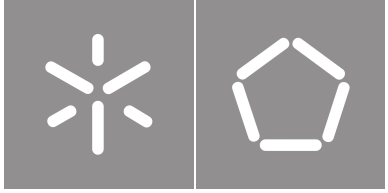
Universidade do Minho
Escola de Engenharia

João Manuel Capela Araújo Ribeiro

AI2SM: Artificial Intelligence to expose plant secondary metabolism

June, 2026

AI2SM: Artificial Intelligence to expose plant secondary metabolism
João Ribeiro
UMinho | 2026



Universidade do Minho
Escola de Engenharia

João Manuel Capela Araújo Ribeiro

AI2SM: Artificial Intelligence to expose plant secondary metabolism

Doctorate Thesis
Doctorate in Informatics

Work developed under the supervision of:
Oscar Dias
Miguel Rocha
Dick de Ridder

June, 2026

COPYRIGHT AND TERMS OF USE OF THIS WORK BY A THIRD PARTY

This is academic work that can be used by third parties as long as internationally accepted rules and good practices regarding copyright and related rights are respected.

Accordingly, this work may be used under the license provided below.

If the user needs permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositoriUM of Universidade do Minho.

License granted to the users of this work



Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International
CC BY-NC-SA 4.0

`https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en`

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.4.2) .

J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: `https://github.com/joaomlourenco/novathesis/raw/main/template.pdf`.

Acknowledgements

The longest four years of my life are finally coming to an end. I cannot describe this feeling as anything other than post-desperation and relief. It has been a long road and some people were key to the success of this thesis.

First of all, I want to address my supervisors. I want to thank Professor Oscar for believing in me, sometimes too unconditionally, since day one. When we first met, I was still a master's student; now I am almost a doctor of philosophy, largely because of him. I also thank him for his constant will to push and challenge me to be better. I want to thank Professor Miguel for all the meetings, off-hour calls, and the vision he had for me and for my thesis. Work aside, thanks to them both for the friendship, late-night conversations, fun barbecues, group dinners and retreats. To Professor Dick, I want to leave a note of acknowledgement for his patience, readiness to help and meaningful contributions to this work; without him, the quality of this work would certainly be poorer.

Aos meus pais e irmã agradeço-lhes por tudo. Deram-me muito, tanto que nunca conseguirei, num tempo de vida, retribuir tudo. O sucesso desta tese, mais que a toda a gente, deve-se a eles por nunca me faltarem em apoio incondicional e carinho. Mas principalmente por nunca deixarem de me aturar e tolerar as minhas ideias progressistas e ousadas, essa tolerância ajudou a que não vivesse preso, com a criatividade castrada.

To my best friends in Wageningen, I want to disclose that you mean so much to me. Nine months were enough for me to get so attached to you that I feel I never left Waga. Every time we see each other again, it's like not a day has passed since I left. And even when I feel detached, I just wish I was there again. To Rosina, Nehir, Anni, Dimitris, Jacky and Cata, a big thank you for all the moments and memories that live rent-free in my heart, all the scientific discussions, all the conversations, all the parties, all the confidence. You made my journey way easier and more fun.

À shrooms crew, Moura, Camila, Constança, Tomás, Bia e Leandro, um obrigado por estarem lá para mim incondicionalmente, mesmo quando eu não me porto tão bem ou mesmo que estejamos longe uns dos outros. Vocês são os meus melhores amigos e fizeram parte desta jornada, mesmo que nao se tenham apercebido. Obrigado por me ajudarem a não me sentir tão só, neste mundo que é tão diferente de mim, de nós. Obrigado por todos os concertos, acampamentos, fugidas de fim de semana, todas essas coisas que ajudam a alienar um pouco da realidade fora da nossa bolha.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

DECLARATION ON THE USE OF ARTIFICIAL INTELLIGENCE (AI) TOOLS

I hereby declare having acted this academic work with integrity and confirm that I have not resorted to any form of undue use of AI tools, nor to the falsification or omission of information, sources, data, results or processes in any of the stages leading to its elaboration.

I further declare that any use of AI tools was done in a transparent and responsible manner, being duly noted in the work, and that I critically verified the accuracy of the content produced with their support, assuming full responsibility for what I submit.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho, as well as the applicable standards of academic integrity and data protection.

_____, _____

(Place) (Date)

_____

(João Manuel Capela Araújo Ribeiro)

Propositions

My time in the Netherlands left its mark on me, through friendships, experiences and science. Inspired by the Dutch tradition of *stellingen*, I am sharing my propositions. They are a mix of what I have learned, questioned, and carried forward both in Portugal and the Netherlands.

1. Even small advances in science are meaningful.
2. Sooner or later, good fundamental science finds its way into industry, rarely on schedule, and often in forms no one foresaw.
3. Being able to collaborate, communicate and put the ego aside is a cheat code in science.
4. Bioinformatics research can only be trusted if code and data are fully available and usable.
5. Artificial intelligence is a powerful tool, it is already outperforming similarity-based and rule-based tools and the tendency, as more data is generated, is to be even better.
6. Not every problem calls for the biggest blade; sometimes a smaller model solves it with far less overhead.
7. A motivated student with a great idea and active supervision can achieve remarkable results, even when they begin with limited preparation.
8. In the age of machine learning, not learning, as humans, has never been so tempting.
9. In an era of abundant data and fast-moving research, rigorous benchmarking is not optional; it is what separates progress from noise.

Resumo

AI2SM: Inteligência Artificial para expor o metabolismo secundário de plantas

Estima-se que as plantas produzam mais de um milhão de metabolitos, mas apenas cerca de 0,1% das suas vias biossintéticas foram elucidadas. Esta lacuna é particularmente acentuada no metabolismo especializado (secundário), apesar do valor destes metabolitos como produtos farmacêuticos, agroquímicos, nutracêuticos e cosméticos. A caracterização das vias requer a reconstrução dos passos de síntese química e a atribuição de cada transformação à reação enzimática subjacente. A retrossíntese automatizada permite prever passos químicos até um metabolito-alvo sem evidência experimental completa, enquanto as ferramentas de anotação de proteínas identificam enzimas e inferem os tipos de reação que catalisam através da previsão do *Enzyme Commission number* (EC). Embora existam várias ferramentas de retrossíntese e anotação enzimática baseadas em inteligência artificial (IA), regras ou similaridade, estas raramente foram comparadas de forma sistemática ou integradas num quadro unificado.

Para colmatar esta lacuna, foi desenvolvido o *DeePlants*, uma plataforma modular que usa IA para apoiar a elucidação do metabolismo especializado das plantas, integrando anotação funcional de enzimas, associação metabolito-via e previsão retrossintética num pacote unificado. Este trabalho contribui para o estudo de representações computacionais de proteínas e produtos naturais, avaliando a sua eficácia na previsão de números EC, especificidade enzima-substrato e associação a vias biossintéticas, estabelecendo uma base sólida para a previsão da função enzimática e a elucidação de vias.

Nos nossos *benchmarks*, os modelos de linguagem de proteínas superaram codificações *one-hot* e BLASTp na previsão de números EC, sobretudo na ausência de homólogos. Para associações metabolito-via, a *Morgan fingerprint* com contagens foi a representação mais eficaz. Para a previsão de especificidade enzima-substrato, uma nova abordagem que combina este *fingerprint*, *embeddings* de modelos de linguagem de proteínas e *gradient boosting* superou os melhores métodos de referência. Adicionalmente, com o *DeepMol*, uma plataforma de aprendizagem máquina automatizada para química computacional, identificámos combinações ideais de modelos e representações de compostos, com o classificador *Ridge* a superar o estado da arte e a prever precursores biossintéticos para mais de 75% dos metabolitos vegetais na LOTUS-DB.

Palavras-chave: catálise enzimática · inteligência artificial · metabolismo especializado · retrossíntese

Abstract

AI2SM: Artificial Intelligence to expose plant secondary metabolism

Plants are estimated to produce more than one million metabolites, yet only about 0.1% of their biosynthetic pathways have been elucidated. This annotation gap is particularly pronounced for specialised (secondary) metabolism, despite the major value of these metabolites as pharmaceuticals, agrochemicals, nutraceuticals, and cosmetics. Pathway characterisation requires not only reconstruction of chemical synthesis steps but also assignment of each transformation to the underlying enzymatic reaction. Automated retrosynthesis offers a computational strategy to predict chemical steps leading to a target metabolite without requiring complete experimental evidence, while protein annotation tools can rapidly predict whether proteins are enzymes and infer the reaction types they catalyse, through Enzyme Commission (EC) number prediction. Despite the availability of many retrosynthesis and enzyme annotation tools, based on artificial intelligence (AI), rules, or sequence similarity, few have been systematically benchmarked or integrated within a unified framework.

To address this gap, DeePlants was developed as a modular framework that leverages AI to support elucidation of plant specialised metabolism, integrating enzyme functional annotation, metabolite-pathway association, and retrosynthetic prediction into a unified package. This work advances the study of computational representations for proteins and natural products, systematically evaluating their effectiveness for EC number prediction, enzyme-substrate specificity prediction, and biosynthetic pathway association tasks, establishing a robust foundation for enzyme function prediction and pathway elucidation.

Protein language models outperformed one-hot encodings for EC number prediction and performed on par with BLASTp, surpassing it when no homologous sequences were available. For metabolite–pathway association, the counted Morgan fingerprint ranked among the strongest representations, matching the state of the art. For enzyme-substrate specificity prediction, a novel framework combining this fingerprint, protein language model embeddings, and gradient boosting surpassed state-of-the-art baselines. Moreover, by developing and using DeepMol, an automated machine learning framework tailored to computational chemistry, we identified optimal model and feature combinations, with Ridge classifier outperforming state-of-the-art approaches and achieving biosynthetic precursor predictions for more than 75% of plant metabolites in LOTUS-DB, a database of over 190,000 natural products.

Keywords: artificial intelligence · enzyme catalysis · retrosynthesis · specialised metabolism

Contents

Acronyms **xvii**

1 Introduction **1**
1.1 Context and motivation . . . 1
1.2 Objectives . . . 3
1.3 Thesis outline . . . 4

2 Background **6**
2.1 Proteins, compounds and reactions . . . 6
2.1.1 Proteins and their computational representation . . . 6
2.1.2 Compounds and their computational representation . . . 7
2.1.3 Reactions and their computational representation . . . 11
2.2 Plant metabolism . . . 15
2.2.1 Plant specialised metabolism . . . 16
2.3 Machine learning . . . 27
2.3.1 Classification tasks . . . 27
2.3.2 Regression tasks . . . 29
2.3.3 Models and algorithms . . . 29
2.4 Deep learning . . . 32
2.4.1 Neural networks . . . 32
2.4.2 Training process . . . 33
2.4.3 Optimising a neural network . . . 34
2.4.4 Bias-variance trade-off . . . 36
2.4.5 Convolutional neural networks . . . 38
2.4.6 Attention mechanism and transformers . . . 39
2.4.7 Graph neural networks . . . 42
2.5 Retrobiosynthesis approaches . . . 43
2.5.1 Reinforcement learning for retrobiosynthesis . . . 44

2.5.2 Template-based approaches . . . 49
2.5.3 Template-free approaches . . . 51

3 DeepMol: an Automated Machine and Deep Learning Framework for Computational Chemistry 53
3.1 Introduction . . . 53
3.2 Implementation . . . 55
3.2.1 Data loading . . . 57
3.2.2 Molecular standardisation . . . 57
3.2.3 Feature extraction . . . 58
3.2.4 Data scaling . . . 59
3.2.5 Feature selection . . . 59
3.2.6 Machine and deep learning models . . . 59
3.2.7 Hyperparameter optimisation . . . 59
3.2.8 Pipelines . . . 60
3.2.9 AutoML: pipeline optimisation . . . 60
3.2.10 Unsupervised learning . . . 61
3.2.11 Data splitting . . . 61
3.2.12 Imbalanced data . . . 63
3.2.13 Feature importance and model interpretability . . . 64
3.3 Discussion: comparison with other tools . . . 65
3.4 Conclusion . . . 69
3.5 Data and code availability . . . 70
3.6 Supplementary information . . . 70

4 Protein LLMs for EC Prediction 71
4.1 Introduction . . . 72
4.2 Results . . . 74
4.2.1 DNN-LLMs perform generally on par with BLASTp . . . 74
4.2.2 BLASTp outperforms DNN-LLMs across identity thresholds . . . 76
4.2.3 DNN-LLMs successfully learn the hierarchy of the EC system . . . 77
4.2.4 DNN-LLMs and BLASTp are complementary . . . 80
4.2.5 DNN-LLMs outperform BLASTp on difficult tasks . . . 82
4.3 Discussion . . . 84
4.4 Methods . . . 85
4.4.1 Defining EC number classification problem . . . 85
4.4.2 UniProtKB data extraction and processing . . . 85
4.4.3 Data splitting . . . 86
4.4.4 Designing the evaluation datasets . . . 86

4.4.5 Language models embeddings for enzyme sequences . . . 87
4.4.6 Model architectures and hyperparameters . . . 88
4.4.7 Pair-wise alignments . . . 89
4.4.8 Metrics . . . 89
4.4.9 Statistical methods . . . 91
4.5 Data and code availability . . . 91
4.6 Supplementary information . . . 91

5 Counted Morgan Fingerprints Best Represent Natural Products 92
5.1 Introduction . . . 92
5.2 Results . . . 97
5.2.1 Qualitative comparison of natural product representations . . . 97
5.2.2 Analysing the distribution and correlations of representation similarities . . . 98
5.2.3 NPClassifierFP and Biosynfoni excel in biosynthetic distance computation . . . 100
5.2.4 NPClassifierFP most consistently predicts biosynthetic relationships . . . 102
5.2.5 Predicting precursors of plant specialised metabolites . . . 107
5.2.6 NPClassifierFP most consistently represents NP classes . . . 109
5.3 Discussion . . . 110
5.4 Conclusion . . . 112
5.5 Methods . . . 112
5.5.1 Natural products representations . . . 112
5.5.2 Building the datasets for masked learning and exploratory analysis . . . 112
5.5.3 Training BERT models with masked learning . . . 113
5.5.4 Similarity calculation . . . 113
5.5.5 Computing biosynthetic distance . . . 114
5.5.6 Building the datasets for multi-label classification . . . 115
5.5.7 Data splitting . . . 116
5.5.8 Supervised model training . . . 117
5.5.9 Model performance evaluation . . . 117
5.6 Data and code availability . . . 119

6 Predicting Enzyme-substrate Specificity 120
6.1 Introduction . . . 120
6.2 Results . . . 122
6.2.1 Dataset and splits . . . 122
6.2.2 XGBoost-NPClassifierFP variants outperform the state of the art . . . 123
6.2.3 Runtime efficiency and memory required . . . 125
6.3 Conclusion . . . 127
6.4 Methods . . . 127

6.4.1 Dataset construction . . . 128
6.4.2 Data splits . . . 129
6.4.3 Calculating substrate and enzyme representations . . . 129
6.4.4 Optimisation and construction of machine learning models . . . 129
6.4.5 Similarity based nearest-neighbour classification . . . 130
6.5 Data and code availability . . . 130

7 Predicting Precursors of Plant Metabolites 131
7.1 Introduction . . . 131
7.2 Results and Discussion . . . 133
7.2.1 AutoML consistently outperforms MGCNN and NP-focused fingerprints . . . 133
7.2.2 Predictions for LOTUS-DB compounds show consistent patterns . . . 135
7.2.3 Ridge classifier can make and explain relevant predictions . . . 137
7.3 Conclusion . . . 140
7.4 Methods . . . 141
7.4.1 Data . . . 141
7.4.2 Metrics . . . 141
7.4.3 Automated machine learning . . . 143
7.4.4 Fingerprints for comparison . . . 144
7.4.5 Statistical methods . . . 144
7.4.6 Precursor predictions for LOTUS-DB plant compounds . . . 145
7.5 Data and code availability . . . 145

8 DeePlants, a Unified Platform for Plant Pathway Prediction 146
8.1 Introduction . . . 146
8.2 General implementation . . . 147
8.3 Proteome annotation . . . 149
8.4 Single-step retrosynthesis prediction . . . 150
8.5 Solutions and Searchers . . . 151
8.6 DeePlants algorithms . . . 152
8.7 Showcase demo . . . 154
8.7.1 One molecule-shot retrobiosynthesis and pathway enrichment analysis . . . 154
8.7.2 Finding candidate enzymes for reactions . . . 156
8.8 Tasty molecules – use cases . . . 156
8.8.1 One-shot genome-guided retrobiosynthesis and pathway associations . . . 159
8.8.2 Single-step prediction of glycosylation for flavour lock – UC 1 . . . 161
8.8.3 Single-step prediction of reductase for β-damascenone precursor production – UC 2 . . . 162
8.9 Discussion and conclusions . . . 164

9 Conclusion and future work **168**
9.1 Main contributions 168
9.2 Publications 170
9.3 Perspectives for future work 171

Bibliography **173**

Appendices
5.1 Supplementary Information - Chapter 5 196
6.1 Supplementary Information - Chapter 6 205
7.1 Supplementary Information - Chapter 7 209
7.1.1 Similarity matrix and t-SNE generation 209
7.1.2 AutoML results 209
7.1.3 Results per label 210
7.1.4 Label scores - overlapping dataset 214
7.1.5 Statistical differences 216
8.1 Supplementary Information - Chapter 8 221

List of Figures

2.1 Generic structure of an amino acid. 6
2.2 Amino-acids separated by their properties 8
2.3 The International Chemical Identifier (InChI) of β-D-glucose. 10
2.4 Classes of non-enzymatic reactions in cellular metabolism. 13
2.5 Main types of enzymatic reactions. Adapted from [49]. 14
2.6 Graphical representation for a SMILES arbitrary target specification (SMARTS) pattern. . 15
2.7 Schematic view of the production of specialised metabolites. 17
2.8 Phenol structure. 17
2.9 Phenolic compounds' biosynthesis: main branches. 19
2.10 Terpenoid biosynthesis: main branches. 24
2.11 The early steps of alkaloid biosynthesis. 25
2.12 Production of the main precursors of alkaloids. 26
2.13 Architecture of a perceptron. 32
2.14 Common activation functions and their derivatives. 34
2.15 Bias-variance trade-off. 37
2.16 Transformer architecture. 41

3.1 The DeepMol AutoML framework offers a comprehensive approach that includes an optimisation framework that samples pipeline configurations from the configuration space. . . 56
3.2 Depiction of dataset splits in the chemical space for the mutagenicity dataset. 63
3.3 Depiction of SHAP values for a ridge classifier and the most relevant MACCS keys in the molecules. 65
3.4 Heatmap showcasing the presence or absence of an integrated way of performing relevant tasks for constructing QSAR/QSPR systems. Green stands for presence, red for absence and yellow for incomplete integration. 66

4.1 Results and evaluation using the whole test set. 75
4.2 The mF1 scores are presented as a function of various identity thresholds in a train-test split setting across five folds. 79

4.3 The performance of each evaluated model in learning the hierarchical structure of the Enzyme Commission (EC) system. 80
4.4 Comparison of DNN-LLM models' and BLAST's F1 scores. 81

5.1 Overview of the ML benchmarking framework for natural product (NP) representations. . 96
5.2 Pairwise correlation matrices and similarity score distributions across NP representations 100
5.3 Biosynthetic distances and distributions of similarities. 101
5.4 t-SNE and PCA projections of NPClassifierFP embeddings with KEGG pathways. 105
5.5 t-SNE and PCA projections of NPClassifierFP with the 30 more represented PlantCyc pathways. 107
5.6 t-SNE and PCA projections of NPClassifierFP with all precursors highlighted. 109

6.1 Comparison of the performance of various Machine Learning (ML) models in predicting proteins and compounds across different identity and similarity thresholds. 123
6.2 Computation and memory requirements as functions of the number of enzyme-substrate pairs for the different methods. 126
6.3 The workflow's upper section illustrates data collection and preprocessing for model training. 128

7.1 Results of each model evaluated on four different settings (see Methods). 133
7.2 Predictions for LOTUS-DB using *Layered FP + Ridge Classifier*. 136
7.3 Predictions made by the *Morgan FP + Ridge Classifier* model and the importance given by the model to structural features in the compounds. 139
7.4 Number of compounds associated with each precursor. Note that one compound can have more than one precursor. 142
7.5 Chemical space of the dataset visualised using t-SNE. 143

8.1 General architecture of DeePlants. 148
8.2 Annotator and classes that extend them. 150
8.3 Reactor and classes that extend them. 151
8.4 Solution classes and search orchestration architecture. 153
8.5 DeePlants algorithm. 155
8.6 DeePlants compound-analysis view. 156
8.7 DeePlants retrobiosynthesis and candidate-enzyme view. 157
8.8 Pathway enrichment and compound classification for UC 1. 160
8.9 One-shot genome-guided route for UC 1. 161
8.10 Pathway enrichment and compound classification for UC 2. 162
8.11 One-shot genome-guided route for UC 2. 163
8.12 Retroformer first 6 solutions for 3-Hydroxy-β-ionone β-D-glucopyranoside. 163
8.13 Retroformer first 6 solutions for Megastigma-6,7-dien-3,5,9-triol (β-Damascenone precursor). 164

S5.1 Distribution of NP classes for the 30k subsample of the whole dataset. 196

S5.2 Validation loss curves in the masked learning modelling task, showing convergence over epochs for models ModernBERT-small and large, DeBERTa-small and BERT-small. 197
S5.3 Validation masked tokens recovery accuracy over epochs. 197
S5.4 Comparison of cross-entropy loss and masked tokens recovery accuracy for training, validation and test sets. 198
S5.5 Normality tests for the KEGG dataset. 199
S5.6 Normality tests for the PlantCyc dataset 199
S5.7 Normality tests for the plant precursors dataset 200
S5.8 KEGG pathway prediction performance and representation analysis. 200
S5.9 Weighted F1, precision and recall for the KEGG dataset. 201
S5.10 PlantCyc pathway prediction performance. 201
S5.11 Weighted F1, precision and recall for the PlantCyc dataset. 202
S5.12 Prediction performance and representation analysis of specialised metabolite precursors. 202
S5.13 Weighted F1, precision and recall for the precursors dataset. 203
S6.1 Similarity distributions for different dataset splits: 80%, 60%, 40%, and 20%. 206
S6.2 Maximum sequence identity values between dataset splits. 207
S6.3 Performance of multiple machine learning models in predicting protein-compound interactions across varying identity and similarity thresholds, using datasets stripped of stereochemical redundancy. 208
S7.1 mF1 score on the validation set in each trial per type of features. 209
S7.2 mF1 score on the validation set in each trial per model. 209
S7.3 F1 scores for the test dataset for each label and model. 210
S7.4 F1 scores for the clustered dataset for each label and model. 211
S7.5 Precision scores for the clustered dataset for each label and model. 212
S7.6 Recall scores for the clustered dataset for each label and model. 213
S7.7 F1 scores for the overlapping dataset for each label and model. 214
S7.8 Precision scores for the overlapping dataset for each label and model. 215
S7.9 Recall scores for the overlapping dataset for each label and model. 216
S8.1 Comparison of enzyme prediction accuracy across protein identity levels for ESM1b, ProtBERT, and ESM2 3B models. 221

List of Tables

2.1 The different line notations of the benzene ring. 9
2.2 Scaffold synthesis of major alkaloid subtypes. 21
2.3 Most used metrics in classification tasks. 28
2.4 Most used metrics in regression tasks. 30
2.5 Last-layer activation and loss functions. 34
2.6 Most common gradient descent variants and respective update rules. 35
2.7 Retrosynthesis methods based on reaction rules. 50

3.1 Performance of DeepMol compared to QSARTuna 68

4.1 wF1 scores for the evidence-level dataset. The levels refer to the different hierarchical levels of the EC number system (i.e. level 1: X.; level 2: X.X.; level 3: X.X.X.; level 4: X.X.X.X) . 77
4.2 wF1 scores for the promiscuous and multi-functional enzymes dataset. 78
4.3 Model wRecall, wPrecision and wF1 scores for the [236] dataset. 82
4.4 Model accuracies for the halogenases dataset. 83

5.1 Overview of natural product representations used in this work 98
5.2 KEGG results expressed in averages ± standard deviation 103
5.3 PlantCyc metabolite-pathways association average performances ± standard deviations . 106
5.4 Performance of all methods in predicting the precursors of plant specialised metabolites 108
5.5 mF1 scores for each of the tasks 110

6.1 Hyperparameter optimisation (HPO) and feature generation time for each pipeline. . . . 126

8.1 Comparison of the four DeePlants inference methods. 165

S5.1 Levene test and max fold difference in variance by metric for KEGG dataset 197
S5.2 Levene test and max fold difference in variance by metric for the PlantCyc dataset 198
S5.3 Levene test and max fold difference in variance by metric for plant precursors dataset . . 198
S5.4 Comparison of Neural Network (NN) and Linear Models for the KEGG dataset. 203

S5.5 Comparison of Neural Network (NN) and Linear Models for the PlantCyc dataset 204
S5.6 Comparison of Neural Network (NN) and Linear Models for the precursors dataset 204
S6.1 Protein identity cross-partition threshold and the resultant datasets 205
S6.2 Compound similarity cross-partition threshold and the resultant datasets 205
S7.1 Significant differences between pipelines for F1 scores in test dataset 217
S7.2 Significant differences between pipelines for precision scores in test dataset 217
S7.3 Significant differences between pipelines for recall scores in test dataset 217
S7.4 Significant differences between pipelines for F1 scores in the alkaloids dataset 218
S7.5 Significant differences between pipelines for precision scores in the alkaloids dataset . . 219
S7.6 Significant differences between pipelines for recall scores in the alkaloids dataset 219
S7.7 Significant differences between pipelines for F1 scores in the clustered dataset 220
S7.8 Significant differences between pipelines for precision scores in the clustered dataset . . 220
S7.9 Significant differences between pipelines for recall scores in the clustered dataset 220
S7.10 Significant differences between pipelines for F1 scores in the overlapping dataset 220
S7.11 Significant differences between pipelines for Precision scores in the overlapping dataset . 221
S7.12 Significant differences between pipelines for Recall scores in the overlapping dataset . . 221

Acronyms

R^2	Coefficient of determination *(p. 30)*
AdaGrad	Adaptive Gradient Algorithm *(p. 36)*
Adam	Adaptive Moment Estimation *(p. 36)*
AI	Artificial Intelligence *(pp. 2, 27, 121)*
AUC	Area Under the Curve *(pp. 28, 29)*
AutoML	Automated Machine Learning *(pp. 2, 4, 5, 54, 55, 59–61, 64, 67, 70, 132–135, 140, 143, 144, 168, 169)*
CASE	Computer-Assisted Structure Elucidation *(p. 7)*
CNN	Convolutional Neural Networks *(pp. 38–40, 72, 73, 75, 78, 82, 88, 89)*
DL	Deep Learning *(pp. 2–4, 32, 37, 39, 40, 43, 44, 46, 53, 54, 59, 63, 67, 69, 71–76, 78, 80, 81, 84, 85, 89, 102, 147, 168, 169)*
DMAPP	Dimethylallyl Diphosphate *(pp. 20, 132)*
DNN	Deep Neural Network *(pp. 2, 33, 38, 72–74, 76–80, 82–84, 88)*
EC	Enzyme Commission *(pp. xiii, xv, 2, 4, 12, 71–78, 80–91, 120, 121, 123, 125, 127–129, 147, 149, 154, 156, 157, 159–168, 170)*
ECFP	Extended Connectivity Fingerprints *(p. 132)*
ESI	Enzyme-Substrate Interaction *(pp. 121, 122, 125, 149)*
ESM	Evolutionary Scale Modelling *(pp. 73, 74, 78–80)*
FCN	Fully Connected Network *(pp. 33, 39)*
FP	Fingerprints *(pp. 92–94, 97–100, 111, 113, 114, 209)*
FPP	Farnesyl Diphosphate *(pp. 132, 134, 136, 137, 141, 164)*
FPR	False Positive Rate *(p. 28)*

GAT Graph Attention Network *(p. 43)*
GCN Graph Convolutional Network *(pp. 42, 43)*
GGPP Geranylgeranyl Diphosphate *(pp. 132, 136, 137, 141, 159, 161, 164)*
GNN Graph Neural Network *(p. 42)*
GO Gene Ontology *(pp. 1, 73, 121)*
GPP Geranyl Diphosphate *(pp. 132, 136, 141)*
GPU Graphical Processing Unit *(pp. 148, 169)*
GSMM Genome-Scale Metabolic Models *(p. 50)*

InChI International Chemical Identifier *(pp. xii, 9, 10)*
IPP Isopentenyl Diphosphate *(pp. 20, 132)*
IUPAC International Union of Pure and Applied Chemistry *(pp. 9, 10)*

KEGG Kyoto Encyclopedia of Genes and Genomes *(pp. 49, 50)*

LLM Large Language Models *(pp. 71, 73, 74, 78, 82, 84, 85, 87, 88, 168)*
LSTM Long Short-Term Memory *(p. 72)*

MAE Mean Absolute Error *(p. 30)*
MCC Matthews Correlation Coefficient *(pp. 28, 120, 123, 124)*
MCTS Monte Carlo Tree Search *(pp. 46–48, 51)*
ME Mean Error *(p. 30)*
MEP Methylerythritol 4-phosphate *(p. 20)*
MHFP MinHashed fingerprints *(pp. 132, 134, 135, 140, 144)*
ML Machine Learning *(pp. xiii, 2–4, 27–29, 32, 37, 53–55, 57–61, 63, 64, 70, 72, 85, 93, 94, 107, 110, 112, 113, 120–125, 127, 129, 130, 147, 169, 171)*
MLP Multilayer Perceptron *(pp. 33, 134)*
MNB Mean Normalised Bias *(p. 30)*
MSE Mean Squared Error *(pp. 30, 34)*
MVA Mevalonate *(p. 20)*

NLP Natural Language Processing *(pp. 27, 40, 51)*
NMR Nuclear Magnetic Resonance *(p. 7)*
NN Neural Network *(p. 93)*
NP Natural Products *(pp. 2, 4, 92–94, 97, 98, 102, 105, 107, 109–113, 115, 117, 132, 140, 141, 144, 147, 154, 156, 159, 160, 162, 165, 168, 169)*
NPV Negative Predictive Value *(p. 28)*

PM Primary Metabolism *(p. 1)*
PPV Positive Predictive Value *(p. 28)*
PR Precision-Recall *(pp. 28, 29)*
PSSM Position Specific Scoring Matrices *(p. 72)*
PUCT Predictor + Upper Confidence Bound for Trees *(pp. 47, 48)*

R Correlation coefficient *(p. 30)*
RF Random Forest *(pp. 72, 121, 132)*
RMSE Root Mean Squared Error *(p. 30)*
RMSProp Root Mean Square Propagation *(p. 36)*
RNN Recurrent Neural Networks *(p. 40)*
ROC Receiver Operating Characteristic *(pp. 28, 29)*
ROS Reactive Oxygen Species *(p. 12)*

SLN Sybyl Line Notation *(p. 9)*
SM Specialised Metabolism *(pp. 1–4, 168–172)*
SMARTS SMILES arbitrary target specification *(pp. xii, 9, 10, 13–15)*
SMILES Simplified Molecular Input Line Entry System *(pp. 9, 13–15, 51, 154, 156, 157, 159, 160)*
SVM Support Vector Machines *(pp. 31, 32, 72)*

TNR True Negative Rate *(p. 28)*
TPR True Positive Rate *(p. 28)*

UC use case *(pp. 156, 161, 164–166)*
UCB Upper Confidence Bound *(p. 47)*

WLN Wiswesser Line Formula Notation *(p. 9)*

XGBoost Extreme Gradient Boosting *(pp. 120, 124–126, 129, 130)*

1 Introduction

1.1 Context and motivation

Genetic and evolutionary events of gene duplication and divergence have caused plants to vary in gene content and function [1]. This is particularly visible in the development of lineage-specific Specialised Metabolism (SM) [2] tailored for defence against biotic and abiotic stressors or taking advantage of external agents [3]. SM enzymes in plants catalyse the production of a remarkable diversity of plant-specific specialised metabolites, compounds that lie outside the conserved pathways of primary metabolism, mediate a lineage's chemical adaptation to its environment, and have broad industrial relevance [4]. In pharmaceutics, plant-derived specialised metabolites include anticancer agents such as paclitaxel, the antimalarial artemisinin, and numerous alkaloids and terpenoids that form the basis of both traditional medicines and modern drug development [5]. Agrotechnology benefits from specialised metabolites that mediate plant–pathogen and plant–insect interactions, offering potential alternatives to synthetic agrochemicals [6], while the cosmetics and food industries rely on plant-derived flavonoids, carotenoids, and aromatic compounds as natural antioxidants, colorants, and flavouring agents [7]. Beyond their direct commercial value, SM represents a largely untapped reservoir of chemical diversity, shaped by millions of years of evolution into complex molecular scaffolds that are often difficult to replicate through chemical synthesis alone [8]. Harnessing this biosynthetic potential, through metabolic engineering or enzymatic approaches, requires a deeper understanding of the enzymes responsible for specialised metabolite biosynthesis, which remains a substantial challenge.

Despite the relevance of these metabolites, their pathways and biosynthetic genes are still poorly characterised [3], especially in plants. Even for the model plant *Arabidopsis thaliana*, a substantial fraction of genes still lack a functional (Gene Ontology (GO)) annotation [9]. Compared to Primary Metabolism (PM), SM genes have restricted phylogenetic distribution [3]. Moreover, plant biosynthetic pathways are frequently not organised into contiguous gene clusters, and the clusters that do exist tend to be partial, encoding only a subset of the pathway's enzymes. Combined with the limited availability of multi-omics data needed to link dispersed genes, this makes cluster-based detection methods (biosynthetic gene cluster detection methods), largely developed for microbial genomes, insufficient for plants. SM enzymes may be associated with convergent evolution events [10], where unrelated enzymes from distant lineages

differing in sequence can use common substrates to produce similar chemicals. Moreover, minor changes to their amino acid sequences can modify their enzymatic function while still being considered homologous genes. Consequently, sequence similarity alone can be an unreliable guide to SM enzyme function, since closely related enzymes can differ in substrate and convergent enzymes can converge on function without converging on sequence.

Machine Learning (ML) and Deep Learning (DL) approaches are promising alternatives to similarity-based protein automatic function prediction strategies. These Artificial Intelligence (AI) tools have delivered promising annotation models with high predictive power at a large scale, especially those using deep self-supervised models of protein sequences [11]. EC numbers are a hierarchical classification system where enzymes are categorised by their catalysed reactions (e.g., EC 1.2.3.4), with the fourth digit specifying the individual reaction, frequently reflecting substrate specificity. This fourth-level annotation is often missing or incomplete, particularly for enzymes acting on specific substrates such as specialised metabolites. Recently, several approaches using Deep Neural Network (DNN)s have been developed to assign EC numbers to sequences with unknown function [12–15], yet predicting the fourth digit with confidence remains difficult. Nonetheless, direct applications of such approaches to plant genomes, and to specialised metabolite biosynthesis in particular, are yet to be explored. Physics-based methods (e.g., docking and molecular dynamics) model binding interactions at atomistic, mechanistic detail [16], but require curated 3D structures, extensive computational resources, and long run times [17], hindering high-throughput applications. ML alternatives offer promising solutions by potentially reducing experimental trial-and-error attempts by suggesting leads and addressing EC annotations without fourth level for a new substrate. However, these studies are still limited by the scarcity of data and integration with EC annotations.

Representing specialised metabolites in a computationally useful way remains a significant challenge in both chemoinformatics and ML. One of the most pressing issues is determining the optimal representation of compounds for tasks related to biosynthesis. Despite advances in chemoinformatics and ML applied to chemistry [18], there is currently no systematic comparison, and no consensus, on which molecular representation, whether based on SMILES strings, molecular fingerprints, graphs, or other encodings, is most effective for predicting biosynthetic pathways or enzyme-substrate interactions. This lack of consensus complicates the development of accurate and scalable models, as the choice of representation can profoundly impact the performance of ML algorithms in tasks such as metabolite-pathway association prediction, enzyme function annotation, or retrosynthetic analysis.

A further challenge, complementary to representation choice, concerns algorithm selection. The “no free lunch” theorem holds that no single algorithm outperforms all others across every problem [19], meaning that the choice of model must be tailored to the specific task at hand. In this context, Automated Machine Learning (AutoML) tools offer a possible alternative as they can search for multiple configurations of feature engineering methods, ML models, and respective hyperparameters. Although several AutoML tools exist for chemoinformatics and drug discovery research [20, 21], they lack customisation for specific problems, standalone usage of the tools inside the frameworks and Natural Products (NP)-tailored representations and benchmarks on predicting biosynthetic information.

At the biological side, a major bottleneck in advancing the study of plant SM lies in the vast number

of metabolites for which no complete biosynthetic pathways have been annotated [22]. The scope of this challenge becomes clear when considering that estimates of plant metabolite diversity range into the hundreds of thousands and possibly beyond a million, yet fewer than 0.1% have characterised biosynthetic pathways [23, 24]. High-throughput metabolomics technologies have been revealing this enormous diversity of plant-derived compounds. This gap reflects not only the intrinsic complexity of plant metabolic networks but also the limitations of current annotation frameworks, which struggle to keep pace with the rate of metabolite discovery.

Furthermore, linking metabolites to their biosynthetic genes or pathways remains largely manual and case-specific, with few automated or scalable alternatives. This bottleneck constrains how fully plant metabolic diversity can be exploited for biotechnology. In this context, computational strategies based on retrosynthesis (retrobiosynthesis) have emerged as a promising alternative to bridge this gap. Retrobiosynthetic approaches aim to infer potential biosynthetic pathways by iteratively decomposing target metabolites into simpler precursor structures, effectively working backwards from the final compound. By integrating chemical transformation rules or predictions with enzymatic knowledge, these methods can generate pathway hypotheses even in the absence of complete experimental evidence. When combined with ML and Deep Learning (DL) models, retrosynthesis frameworks can be further enhanced to prioritise biologically relevant reactions and candidate enzymes, enabling more accurate and scalable predictions.

Altogether, ML-based enzyme function prediction and retrosynthetic pathway reconstruction hold significant potential to overcome current limitations in plant SM annotation. Combining these computational strategies may accelerate the discovery of novel biosynthetic pathways, improve metabolite-to-gene associations, and ultimately facilitate the rational engineering of plant metabolic systems for applications in agriculture, medicine, and industry.

1.2 Objectives

The main objective of this work is to develop an integrative computational framework to improve the functional annotation of plant SM, bridging the gap between metabolite discovery and biosynthetic pathway reconstruction. To achieve this, the specific objectives are:

- To critically evaluate the limitations of traditional similarity- and ML-based annotation approaches in the context of plant SM, particularly their inability to account for convergent evolution and substrate specificity.

- To develop, adapt, and benchmark ML and DL models for enzyme function prediction, with a focus on improving substrate specificity assignment beyond incomplete EC annotations.

- To systematically assess and identify the most effective computational representations for both compounds and proteins in tasks relevant to plant specialised metabolite biosynthesis. This involves comparing molecular fingerprints, graphs, and other encodings to determine their impact on

model performance in metabolite-pathway association prediction and enzyme-substrate specificity modelling.

- To design and implement AutoML tools tailored for specialised metabolite biosynthesis, enabling automated feature engineering, model selection, and hyperparameter optimisation. These tools will be customised to handle NP-tailored representations and benchmarks, addressing the "no free lunch" theorem by adapting to the unique challenges of plant SM.
- To develop retrosynthetic (retrobiosynthetic) approaches for inferring plausible biosynthetic pathways of orphan or partially characterised plant metabolites. This will involve integrating chemical transformation rules and out-of-the-box DL predictions with enzymatic knowledge to generate biologically relevant pathway hypotheses, even in the absence of complete experimental evidence.
- To integrate enzyme function prediction with retrosynthetic pathway reconstruction into a cohesive, scalable framework. This integrative approach will establish reliable associations between metabolites, reactions, and candidate genes.

1.3 Thesis outline

This thesis is organised into nine chapters. A brief description of each is given below:

- Chapter 2 provides an introduction to the foundational concepts and methodologies of ML, DL, bioinformatics, chemoinformatics, and plant SM. It offers a concise yet comprehensive overview of each field, highlighting their interconnections and relevance to the computational framework developed in this thesis.
- Chapter 3 details the implementation and key features of the AutoML tool DeepMol, an automated machine and deep learning framework for computational chemistry.
- Chapter 4 evaluates and compares state-of-the-art protein language models for predicting EC numbers. The analysis includes a performance benchmark not only across different protein representations but also against BLASTp, the gold standard for sequence similarity-based annotation, to contextualise the predictive capabilities of these models.
- Chapter 5 systematically evaluates and compares computational NP representations for the prediction of biosynthetic information such as establishing metabolite-pathway associations and predicting NP classes based on biosynthetic origin and structure.
- Chapter 6 integrates the computational representations benchmarked in Chapters 4 and 5 to develop a ML system for predicting enzyme-substrate specificity. The system's performance is benchmarked against similarity-based and ML-based state-of-the-art approaches. Additionally, large-scale efficiency assessments are conducted to evaluate scalability and computational feasibility.

- Chapter 7 investigates the capability of the AutoML tool DeepMol to predict biosynthetic information about the precursors of plant specialised metabolites.
- Chapter 8 presents a comprehensive overview of the software developed in this work, detailing the implementation and key features the integrative framework DeePlants. It consolidates all tools and machine learning models developed in this thesis, including enzyme annotation, metabolite-pathway associations and retrosynthesis tools, into a unified computational pipeline.
- Chapter 9 concludes the dissertation by summarising the main findings, discussing the contributions, and suggesting directions for future work.

2

Background

2.1 Proteins, compounds and reactions

2.1.1 Proteins and their computational representation

Proteins are essential components of living organisms, performing a wide range of functions [25]. They can catalyse reactions essential to the cell [26], and transport metabolites from one compartment to another [27]. In addition to the roles mentioned above, structural proteins provide mechanical support for cells and are among the most abundant proteins by mass in many organisms [28]. These supramolecular structures act as the skeleton of cells, holding them together and allowing them to function properly [28].

Biochemically, proteins are large molecules built from amino acids linked in chains. Such a chain is called a polypeptide, and a protein consists of one or more polypeptides folded into a functional structure. The distinction between a protein and a shorter oligopeptide is conventional rather than absolute, with a threshold of roughly 50 amino acids commonly used [29].

Chemical structures of amino acids differ only in the side chain, simplistically represented as an *R* group. Generically, all amino acids are composed of a carboxylic group, an amine group and a *R* group, as shown in Figure 2.1.

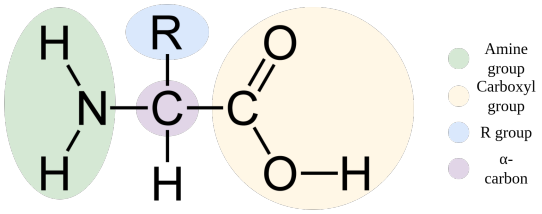


Figure 2.1: Generic structure of an amino acid.

Although there are 20 standard amino acids, two further genetically encoded residues extend the set to 22: selenocysteine, incorporated across all domains of life including in human selenoproteins, and pyrrolysine, restricted to certain archaea and bacteria. Among the standard amino acids, five carry electrically charged *R* groups near neutral pH; see Figure 2.2A. Arginine and lysine are positively charged, whereas aspartic and glutamic acids are negatively charged. Histidine is a special case: with a side-chain

pKa near 6, it is only partially protonated at pH 7 and carries a small, environment-dependent positive charge. The non-standard pyrrolysine also bears a positively charged side chain. On the other hand, serine, threonine, asparagine, and glutamine possess uncharged and polar *R* groups, as enumerated in Figure 2.2B. Histidine stands out because its side-chain pKa lies close to physiological pH, allowing it to shift between protonated and neutral forms and to act as both a weak acid and a weak base. Alanine, isoleucine, leucine, methionine, phenylalanine, valine, proline, and cysteine are commonly classified as hydrophobic, whereas glycine, tyrosine, tryptophan, and selenocysteine are sometimes included but occupy intermediate positions on most hydrophobicity scales. Strongly hydrophobic residues tend to be buried in the protein core, away from the solvent [30].

Charged amino acid side chains are important for protein three-dimensional structure stability. The interaction between negatively and positively charged *R* groups forms "salt bridges", both on the protein surface and buried within its interior [31]. These interactions contribute to thermostability and increase resistance to denaturation at higher temperatures, an effect particularly pronounced in proteins from thermophilic organisms [31].

Though cysteine is considered a hydrophobic residue, it may emerge on the protein surface [32]. Moreover, sulfur-containing residues such as cysteine and methionine contribute to protein structure stability by forming disulfide bridges, connecting different parts of the structure or subunits in a protein complex.

As amino acids combine to form peptides, a water molecule is removed at each peptide bond: the carboxyl group of one amino acid loses a hydroxyl group and the amino group of the next loses a hydrogen atom. The portion of each amino acid that remains within the chain is called a residue. Amino acid residues at the start of a peptide, whose amino groups are not bound to other amino acids, are known as N-terminal residues. Conversely, amino acid residues at the end of a peptide, whose carboxyl groups are not bound to other amino acids, are called C-terminal residues.

During or after synthesis, residues suffer post-translational modifications, which may physically and chemically change them [33]. These modifications include changes in the three-dimensional structure and, consequently, in the function of proteins [33]. Moreover, other non-peptide molecules can be attached to the protein and be important for their function - these are called cofactors. Proteins can also interact with each other and form stable complexes to achieve a specific function.

2.1.2 Compounds and their computational representation

A chemical compound typically consists of two or more atoms of different types bonded together in a well-defined stoichiometry. The atoms are arranged in space in a particular way, and this arrangement, along with the electronic structure, defines the compound's chemical structure [34]. The representation of chemical compounds often relies on the chemical structure, the type of atoms, or their chemical properties [35].

The advent of techniques such as crystallography, Nuclear Magnetic Resonance (NMR), and Computer-Assisted Structure Elucidation (CASE) has allowed for a more comprehensive and detailed description of

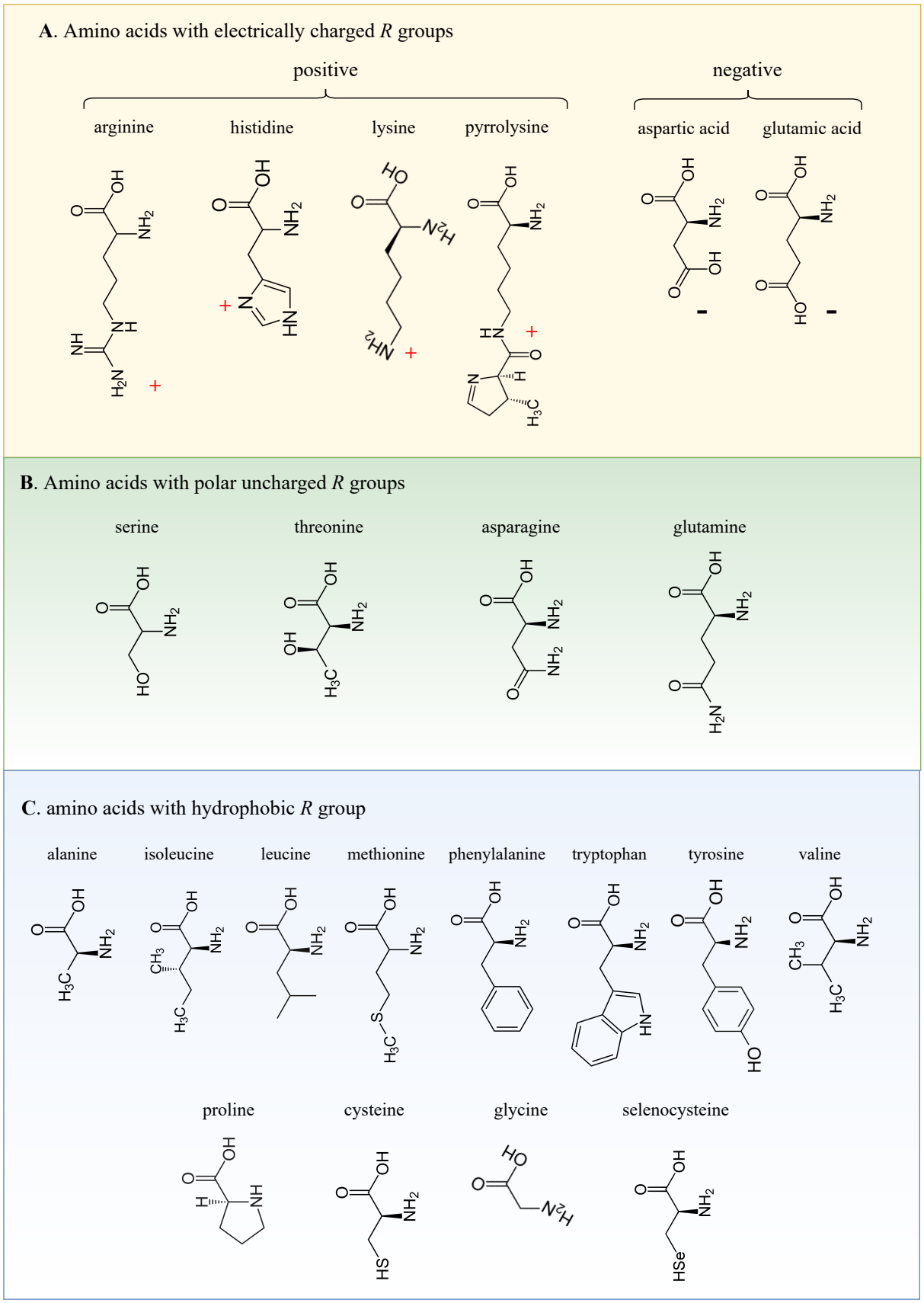


Figure 2.2: Amino-acids separated by their properties

molecular structures [35]. As a result, more complex molecular structures have been identified, and traditional methods of naming compounds, like a systematic chemical nomenclature, have become inadequate due to the long and complex names they produce [35].

Chemical graph formalism

Computationally, chemical structures can be represented as graphs in which atoms are nodes and bonds are edges. These graphs are undirected and labelled, as bonds have no direction and nodes are labelled with the atom symbol [36–38].

The similarity between a compound's structure and a topological graph has inspired the creation of numerous algorithms that allow analysing its structure, including its constituent atoms and bonds [37].

Line notations

Line notations are concise and machine-friendly representations of molecular structures using alphanumeric symbols [36, 39]. The first widely used notation was the Wiswesser Line Formula Notation (WLN) [36]. Although compact, it was difficult to learn and generate by hand, and its use declined through the 1980s as connection tables and Simplified Molecular Input Line Entry System (SMILES) [40] became established. Other line notations have also been developed over time (as shown in Table 2.1 for the structure of benzene). In addition to SMILES, the International Union of Pure and Applied Chemistry (IUPAC)'s InChI [39] is also widely used.

Table 2.1: The different line notations of the benzene ring.

Line notation	Representation	Reference
WLN	R	[41]
SMILES	C1=CC=CC=C1	[40]
Sybyl Line Notation (SLN)	C[1]H:CH:CH:CH:CH:CH:@1	[42]
ROSDAL	1-2=3-4=5-6=1;	[43]
InChI	1S/C6H6/c1-2-4-6-5-3-1/h1-6H	[44]
InChIKey	UHOVQNZJYSORNB-UHFFFAOYSA-N	[44]
CAS	71-43-2	[45]

SMILES

The SMILES is a popular way to represent chemical structures as linear strings of symbols. These strings are compact, easy for humans to read and write, and can be used to generate molecular graphs that can be processed using graph-based algorithms. SMILES strings consist of chemical element symbols, bonds, indices of broken cycles, and parentheses enclosing branching vertices [40, 46]. However, SMILES are not unique, meaning that a single compound may have multiple possible SMILES representations [39, 40]. Efforts have been made to standardise the generation of canonical SMILES strings.

SMARTS is a language used to describe and search for specific patterns within chemical structures. It allows users to query for specific substructures within a molecule by extending the labels of nodes and

edges to include *logical operators* and special atomic and bond symbols. These symbols enable SMARTS to represent atoms and bonds flexibly; for example, the symbols **[c,n;H1]** can be used to represent a molecule containing an aromatic carbon or nitrogen with exactly one hydrogen atom.

InChI and InChIKey

IUPAC developed the InChI in 2005 as a public, open-source tool for uniquely identifying chemical compounds [47]. InChI uses a hierarchical, structure-based approach to generate a unique identifier for each compound. While InChI can represent a wide range of organic compounds, it does not cover the whole set of inorganic compounds [44]. InChI has become a widely-used resource in the chemical community [44].

InChI generates a unique identifier for each chemical compound (example in Table 2.1). The identifier consists of multiple layers, each representing a specific class of structural information, such as the formula, charge, and stereochemistry [36] (see an example in Figure 2.3). The main layer, which includes the formula and atomic bonds, is present in all InChI identifiers. Other layers may vary depending on the properties and structure of the compound.

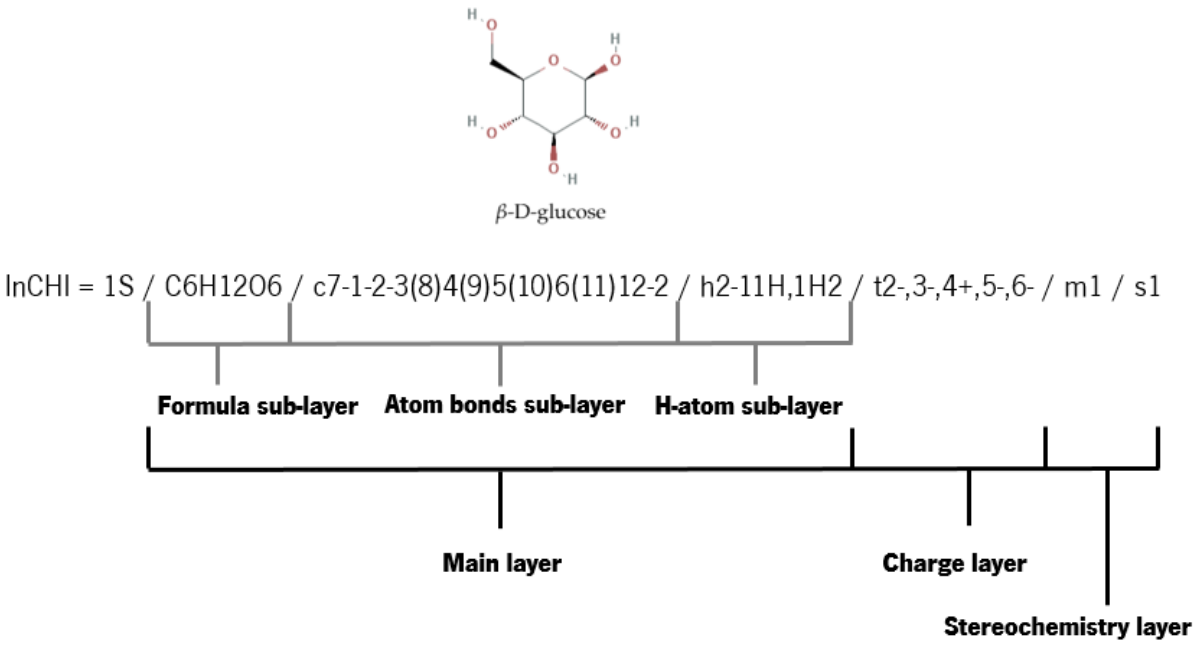


Figure 2.3: The InChI of β-D-glucose.

InChI identifiers can be long and detailed, making them difficult to store and process [36]. To address these limitations, InChI identifiers can be converted into InChIKeys, fixed-length 27-character strings (see the benzene InChIKey in Table 2.1). The first block (14 characters) hashes the molecular skeleton, that is, the connectivity; the second block (8 characters) hashes the remaining layers, including stereochemistry and isotopic composition, and is followed by characters encoding the InChI version and flags, with a

final character indicating the protonation state. This eases storing and indexing chemical structures in databases.

2.1.3 Reactions and their computational representation

Reactions can be defined as transformations applied to molecules, thus converting one or several into other molecules. The consumed molecules are called reactants, whereas the ones produced are called products. Reactions can occur in only one way (reactants ⟶ products) or both ways (reactants ⟶ products and products ⟶ reactants), being irreversible or reversible, respectively.

Generically, reactions can be classified as spontaneous or non-spontaneous. Spontaneous reactions proceed favourably under the conditions in which they occur, releasing usable energy as reactants are converted into products. Whether this happens depends on the combined effect of the enthalpy and entropy changes at a given temperature, not on either quantity alone. Non-spontaneous reactions, by contrast, do not proceed favourably under the given conditions and require an external input of energy.

Whether a reaction is spontaneous or non-spontaneous depends on the change in free energy between its initial and final states. The difference of free energy from one initial state to a final state is defined as *Gibbs free energy*. It can be defined as follows:

$$\Delta G = G_{final} - G_{initial} \tag{2.1}$$

The result expressed by this difference represents the amount of usable energy that is released or absorbed from the system. Nevertheless, *Gibbs free energy* can also be defined as follows:

$$\Delta G = \Delta H - T\Delta S \tag{2.2}$$

where ΔH, the enthalpy change, is approximated by the difference in bond energies between reactants and products, ΔS is the entropy change during the reaction (whether the system becomes more disordered or not), and T is the temperature of the occurring conditions.

According to these definitions, a reaction with negative *Gibbs free energy* is spontaneous, whereas a reaction with positive *Gibbs free energy* is non-spontaneous.

Catalysts can help accelerate the transformation of reactants into products within spontaneous and non-spontaneous reactions [48]. Enzymes are natural catalysts of biochemical reactions, accelerating reaction rates by decreasing their activation energy [49]. The activation energy is required to assemble or break chemical bonds and convert reactants into products. Notably, the active site of an enzyme is where it binds to its substrate (a specific type of reactant), leading to faster chemical transformation. After the substrate binds to the enzyme in the active site, there is a slight modification to find the best fit between the transition state (a structural intermediate between the substrate and product) and the active site. Generally, an enzyme is specific for a substrate, although, in several cases, they are promiscuous to structurally related compounds.

Finally, many enzymes cannot operate properly or efficiently without the help of other molecules, such as coenzymes and cofactors. These molecules bind to the protein, helping to stabilise its conformation and ensuring proper function.

Notice that the same thermodynamic principles apply to enzymatic and non-enzymatic reactions [49, 50]. This means that an enzymatic reaction can occur in a non-enzymatic environment, and enzymes do not prevent non-enzymatic reactions from occurring within a metabolic network [49]. Therefore, it is important to consider enzymatic and non-enzymatic reactions in studying metabolism.

Types of non-enzymatic and enzymatic reactions

According to Keller and collaborators [49], non-enzymatic reactions contributing to cellular metabolism can be divided into three classes, as shown in Figure 2.4. Class I reactions display broad chemical reactivity and low substrate specificity, occurring both enzymatically and non-enzymatically across a wide range of substrates. These include Maillard reactions [51], oxidation reactions driven by Reactive Oxygen Species (ROS), and covalent modifications of lipids and proteins such as alkylation, glycosylation, and acetylation [52]. Class II reactions are highly specific and occur exclusively non-enzymatically as part of the metabolic network, with no enzymatic counterpart. A well-known example is the maturation of vitamin D3, in which a precursor is converted to previtamin D3 by ultraviolet light in the skin, followed by spontaneous thermal isomerisation into vitamin D3 [53]. Class III reactions occur simultaneously through both enzymatic and non-enzymatic routes, closely resembling enzymatic catalysis across all six major enzyme classes that perform chemical transformations. In many cases, the enzyme acts as a negative catalyst, primarily serving to prevent unwanted side products that would otherwise arise from the spontaneous reaction [49]. The EC system defines seven classes. The six that catalyse chemical transformations are illustrated with examples in Figure 2.5, while the seventh, translocases, mediates transport across membranes:

1. In redox reactions, an electron is transferred from one chemical species into another. Enzymes: oxidoreductases (EC 1.#.#.#);

2. In single replacement reactions, a part of a reactant is transferred to another chemical species, replacing an existing part of that species. Enzymes: transferases (EC 2.#.#.#);

3. In hydrolysis reactions, a bond is cleaved by the addition of water. Enzymes: hydrolases (EC 3.#.#.#);

4. In elimination or addition reactions, bonds are broken or formed by routes other than hydrolysis or oxidation, often generating or consuming a double bond. Enzymes: lyases (EC 4.#.#.#);

5. In isomerisation reactions, a reactant is converted into its isomer. Enzymes: isomerases (EC 5.#.#.#);

6. In synthesis reactions, several reactants react to form a product. Enzymes: ligases (EC 6.#.#.#);

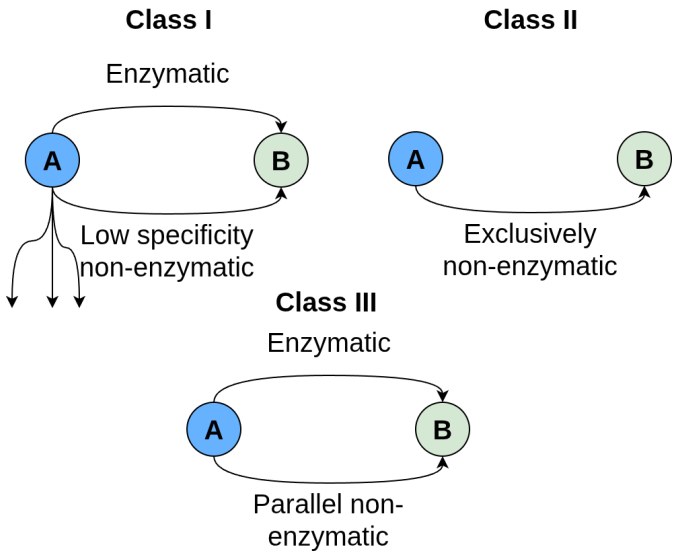


Figure 2.4: Classes of non-enzymatic reactions in cellular metabolism. Each panel illustrates the conversion of a substrate (A) into a product (B). Class I: the reaction proceeds both enzymatically and non-enzymatically, but the non-enzymatic route is unspecific and acts on a broad range of substrates. Class II: the reaction occurs exclusively non-enzymatically, with no enzymatic counterpart known. Class III: the reaction proceeds simultaneously through both enzymatic and non-enzymatic routes. Adapted from [49].

7. These enzymes facilitate the transport or separation of ions or molecules across or within membranes. Enzymes: translocases (EC 7.#.#.#);

Reaction representations

According to Daylight Chemical Information Systems (`https://www.daylight.com`), the developer of the SMILES and SMARTS notations, complete reactions can be represented in three ways:

1. using the reactants' SMILES, each separated by a ".", followed by the notation "»" and, as in reactants, the products' SMILES notation separated by a ".";

2. using the reaction graph, where pairwise mapping of the reactant and product atoms and bonds are established. Differences in atom and bond properties are stored on the edges of the graph;

3. an atom- and bond-change list plus the reactant molecules.

Nevertheless, representing reactions more generally can help capture all reactions involving the same atoms and bond changes. It is also essential to include the indirect effects of activating or deactivating groups on the reaction site. To represent these generic reactions, the SMARTS query language is used within the Daylight system.

The key difference between complete and generic reaction representations lies in how their transformations are defined. A complete reaction is encoded as a strict SMILES representation and can therefore only be applied to one specific substrate. A generic reaction, by contrast, is defined by a SMARTS pattern

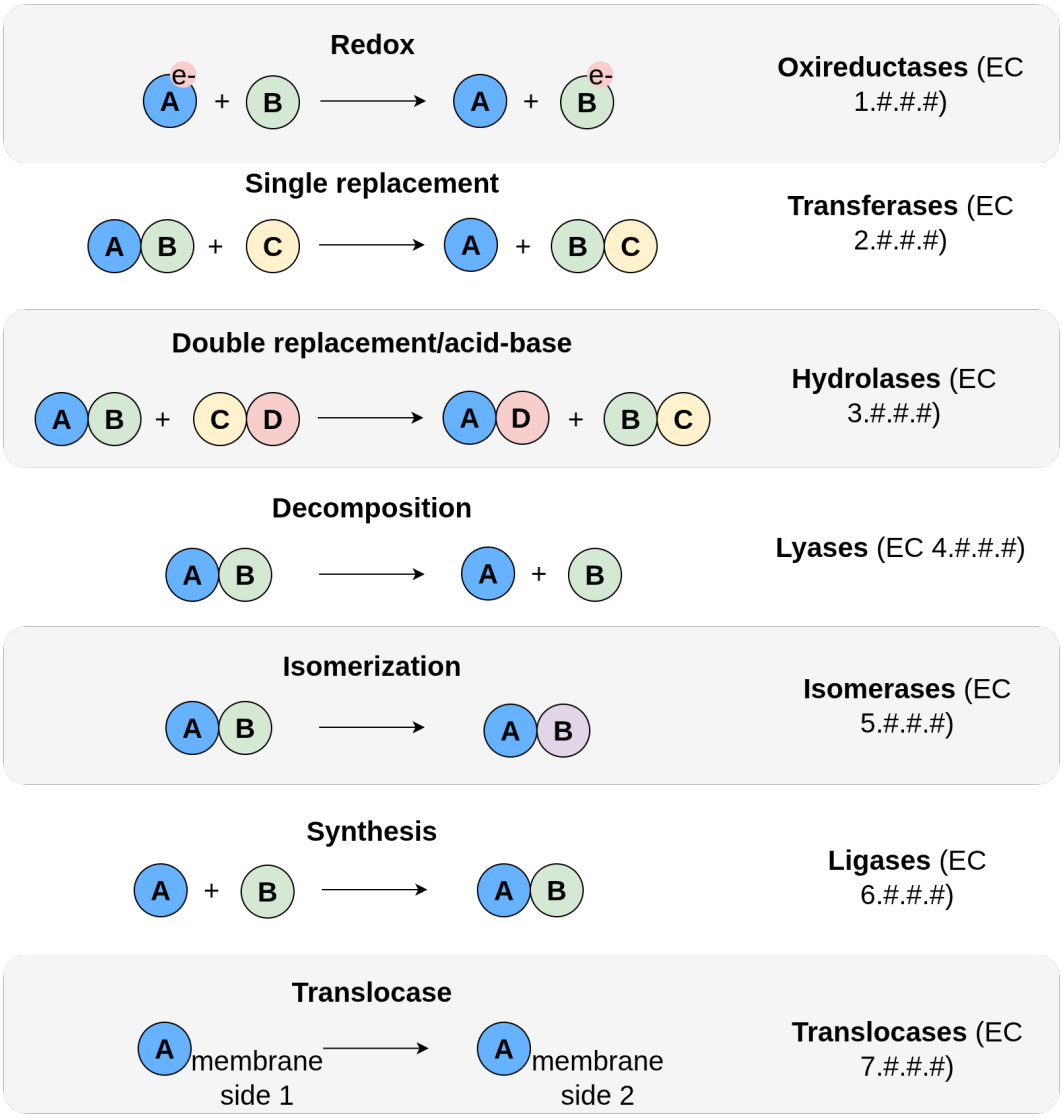


Figure 2.5: Main types of enzymatic reactions. Adapted from [49].

— known as a reaction rule — that matches any compound whose structure satisfies the pattern, allowing the transformation to be applied broadly across chemically related substrates. These reaction rules take the form of a SMARTS-SMILES hybrid representation [54]. The rules for properly constructing reaction SMARTS are the following:

1. All atoms must be paired between the reactants and products, meaning that each atom in the reactants corresponds to a specific atom in the products.

2. The stoichiometric coefficients are 1 to 1 for all atoms in reactants and products. If stoichiometry is to be represented, the reactants and products must be repeated.

3. Explicit hydrogen atoms used on one side of a reaction should be mapped and represented on the other side.

4. Bonds should be represented as in SMILES; SMARTS queries are not allowed; expressions related to atoms should be any valid SMARTS expression where the bonds do not change; otherwise, the representation must be a valid SMILES.

The grammar of the transformation is defined as follows:

$$\begin{gathered} reactant > agent > product \\ or \\ reactant >> product \end{gathered} \tag{2.3}$$

where the reactant, agent and product are defined either with a SMILES or SMARTS notation.

In a generic reaction, the reactants and products are separated by "»", and dots separate the atoms of each reactant on the reactant side or each product on the product side. Each atom is paired between the reactants and products, including explicit hydrogen atoms. An example of a graphical representation of a SMARTS pattern is depicted in Figure 2.6.

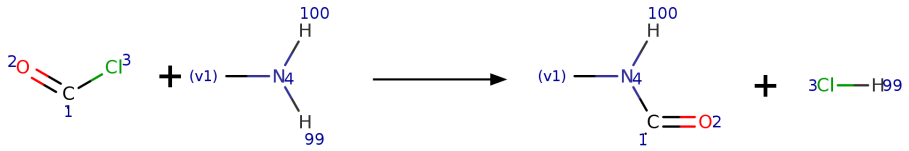


Figure 2.6: Graphical representation for the SMARTS pattern [C:1](=[O:2])[Cl:3].[H:99][N:4]([H:100])[C:0]» [C:1](=[O:2])[N:4]([H:100])[C:0].[Cl:3][H:99]. Image computed with MarvinView software version 22.13.

2.2 Plant metabolism

The scientific community has acknowledged the difference between general (or primary) and specialised (or secondary) plant metabolism for over a century. This differentiation is widely accepted and has played a significant role in the study of plant biology [55]. According to this assumption, primary plant metabolism includes all processes essential to basic life functions, such as reproduction, storage, respiration, and cell division [56]. Based on this distinction, specialised plant metabolism can be considered non-essential for plant growth and development. Regardless, it plays a vital role in how plants interact with their environments, such as through defence mechanisms or adaptation to changing conditions [56, 57]. In this sense, three overarching compound classes must be considered: primary metabolites, specialised metabolites and hormones, which regulate growth and metabolism [58]. Although this functional trichotomy shaped decades of plant biology research, recent advances in chemical and genetic studies blur this strict boundary between definitions [58]. It has been demonstrated that specialised metabolites are multifunctional, potent regulators of plant growth and development [58, 59], defence regulators [60, 61] and can be re-integrated into primary metabolism, serving as biosynthetic precursors [62, 63]. Therefore, specialised

metabolites cannot be considered end-points or end-products but rather integral components of the whole metabolic landscape [58, 64].

In 1891, Albrecht Kossel used the term "secondary metabolite" to refer to any cell component that is not present in a developing cell, in contrast to the term "primary metabolite", which refers to essential cellular components [55]. Around twenty years later, Friedrich Czapek, in his Biochemistry of Plants book, adopted the concept and it has been used ever since. Half a century later, more sophisticated analytical techniques, such as chromatography, allowed the discovery and characterisation of more molecules. Such advances were the basis for establishing the discipline of phytochemistry [56].

Generally, critical components of primary metabolism include photosynthesis, the Krebs or citric acid cycle, and oxidative phosphorylation, among others. The metabolites produced and those that act as intermediates in primary metabolism include sugars, amino acids, tricarboxylic acids, nucleic acids, and polysaccharides [56].

Production of specialised plant metabolites is controlled by a highly complex network of transcription factors in response to external factors [57]. They often accumulate in very small quantities, mostly in specialised cells [56]. They are vastly diverse in their chemical nature, but all are produced through biosynthetic pathways that mainly use compounds from primary metabolism [56].

specialised plant metabolites can be divided into two categories based on their chemical and taxonomic characteristics: high-molecular-weight polymeric compounds, such as lignins, which are present in all higher plants and have a structural function; and low-molecular-weight compounds, such as alkaloids, which have a more restricted taxonomic distribution and are produced in response to biotic and abiotic stressors.

2.2.1 Plant specialised metabolism

Contrary to the highly conserved nature of primary metabolites, specialised metabolites evolve rapidly and dynamically, thus varying in structure and synthesis [58]. Generally, primary metabolism produces the main metabolite starters to produce specialised metabolites. These starters are mainly produced by photosynthesis, glycolysis, the citrate cycle, the shikimate pathway, and the two isoprenoid pathways, mevalonate (MVA) and non-mevalonate (DOXP/MEP). These compounds are then transformed downstream by cyclisation, ligation, glycosylation, methylation, acylation, prenylation and oxidation to produce a diverse pool of specialised metabolites [65]. Figure 2.7 depicts a schematic overview of the production of such plant-derived specialised metabolites.

This thesis will only focus on the biosynthesis of phenols, terpenes and alkaloids, as the most diverse classes of specialised metabolites. Accordingly, the following subsections will describe the structure of each of these classes of plant specialised metabolites and how they are produced.

Phenols

Phenols are aromatic compounds with one or more acidic hydroxyl groups; their generic representation is shown in Figure 2.8. Over 8,000 phenolic structures are known [66], finding wide distribution across the

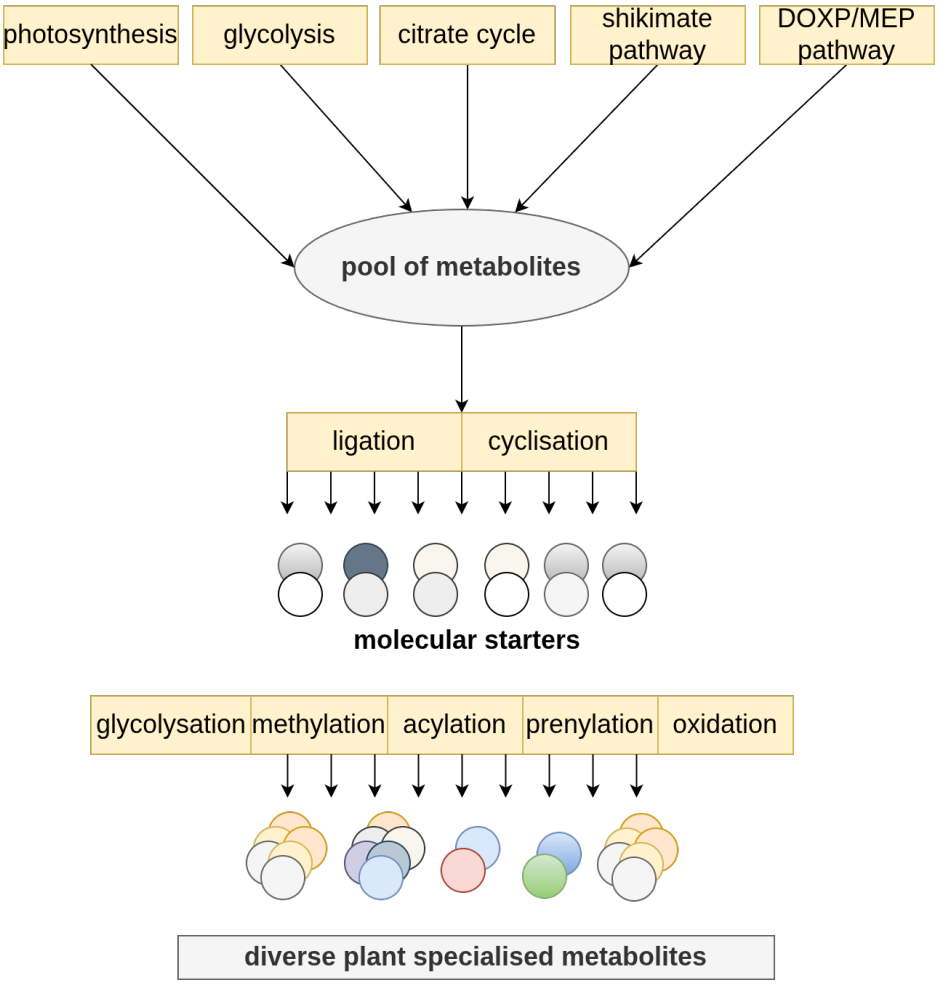


Figure 2.7: Schematic view of the main sources and chemical steps underlying the production of specialised metabolites. Adapted from [65].

plant kingdom and being stunningly diverse in size, ranging from small molecules to highly polymerised ones. Generally, phenolic compounds are classified as flavonoids and non-flavonoids [67].

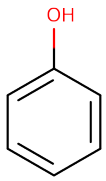


Figure 2.8: Phenol structure.

Flavonoids possess a C6-C3-C6 chemical skeleton and can receive several substituents such as hydroxyl- and methyl- groups, isopentenyl units, and sugars [67]. They are often located in the vacuoles and are responsible for the colour of most flowers and fruits, working as pollinator attractors [57, 67]. Flavonoids also respond to wounding, drought, heavy metals, and starvation stresses [57]. Furthermore, they operate as UV-B radiation protectors and antifeedants [67, 68]. Non-flavonoids include stilbenes, tannins, coumarins, and lignans, among others. Generally, all of these, except lignans, are produced in

response to wounding stress and pathogen attacks [69, 70] and have been closely associated with the response to predation by mammals, birds, and insects [71, 72].

The shikimate and malonic acid pathways synthesise the main substrates for producing phenolic compounds in plants. Phenylalanine is produced in the shikimate pathway, being a molecular starter for the biosynthesis of the phenolic compounds (Figure 2.9). Phenylalanine enters the phenols biosynthesis pathway by being transformed into trans-cinnamic acid in a reaction catalysed by a phenylalanine ammonia-lyase (PAL) [73]. Trans-cinnamic acid is the most upstream compound in this super pathway [73].

Accordingly, trans-cinnamic acid is a biosynthetic precursor of coumarins. Although coumarins are compounds of paramount importance for plants, human health and the food industry, several steps of their biosynthetic pathway and genes remain unknown and uncharacterised, varying from species to species [74, 75]. Trans-cinnamic acid is hydroxylated through a reaction catalysed by a cinnamate 4-hydroxylase (C4H), producing *p*-coumaric acid. Afterwards, several steps undergo the conversion of this compound into *p*-coumaryl alcohol, coniferyl, and sinapyl alcohol to be consumed by the lignan and lignin biosynthesis pathways [76].

4-coumaroyl-CoA is formed with a synthesis reaction catalysed by a 4-coumaric CoA ligase (4CL). It is then combined with malonyl-CoA, branched off from the malonic acid pathway, to form chalcone. Several steps underlie the transformation of chalcone into other molecular starters, most of which are represented in Figure 2.9, for the biosynthesis pathways of flavones, flavonols, anthocyanidins, and proanthocyanidins [68, 77, 78]. Moreover, 4-coumaroyl-CoA serves as a precursor of liquiritigenin that contributes to isoflavone production through a reaction catalysed by an isoflavone synthase (IFS), which later enters the isoflavonoids pathway [77].

Another important branch of this pathway is the stilbene skeleton formation through a reaction catalysed by a stilbene synthase (STS) taking as substrates malonyl-CoA and a CoA ester of a cinnamic acid derivative [71].

Terpenoids

Terpenoids or isoprenoids represent the most diverse class of specialised metabolites produced by plants [79]. These metabolites are specialised in ecological interactions to attract pollinators, repel herbivores or attract other animals [80]. It is thought that the proliferation of cytochrome P450 and terpene synthase gene families across plants accelerated the evolution of their biosynthetic genes [81–83].

The high chemical diversity of terpenoids (over 40,000 compounds) is mainly based on the simplicity of the process of constructing molecules of different sizes [79]. The diversity reflects a genetic and co-evolutionary journey along with herbivore stress. Notably, the general similarity of folds and motifs of plant and animal proteins might boost and ease the process of selecting molecules with a natural affinity to animal proteins [82, 84]. Hence, the exposure to predators might have facilitated the selective process of functionalised terpenoids with potent biological activities towards animals [82, 84].

The isoprene rule, defined by Wallach and Ruzicka in the late nineteenth and mid-twentieth centuries,

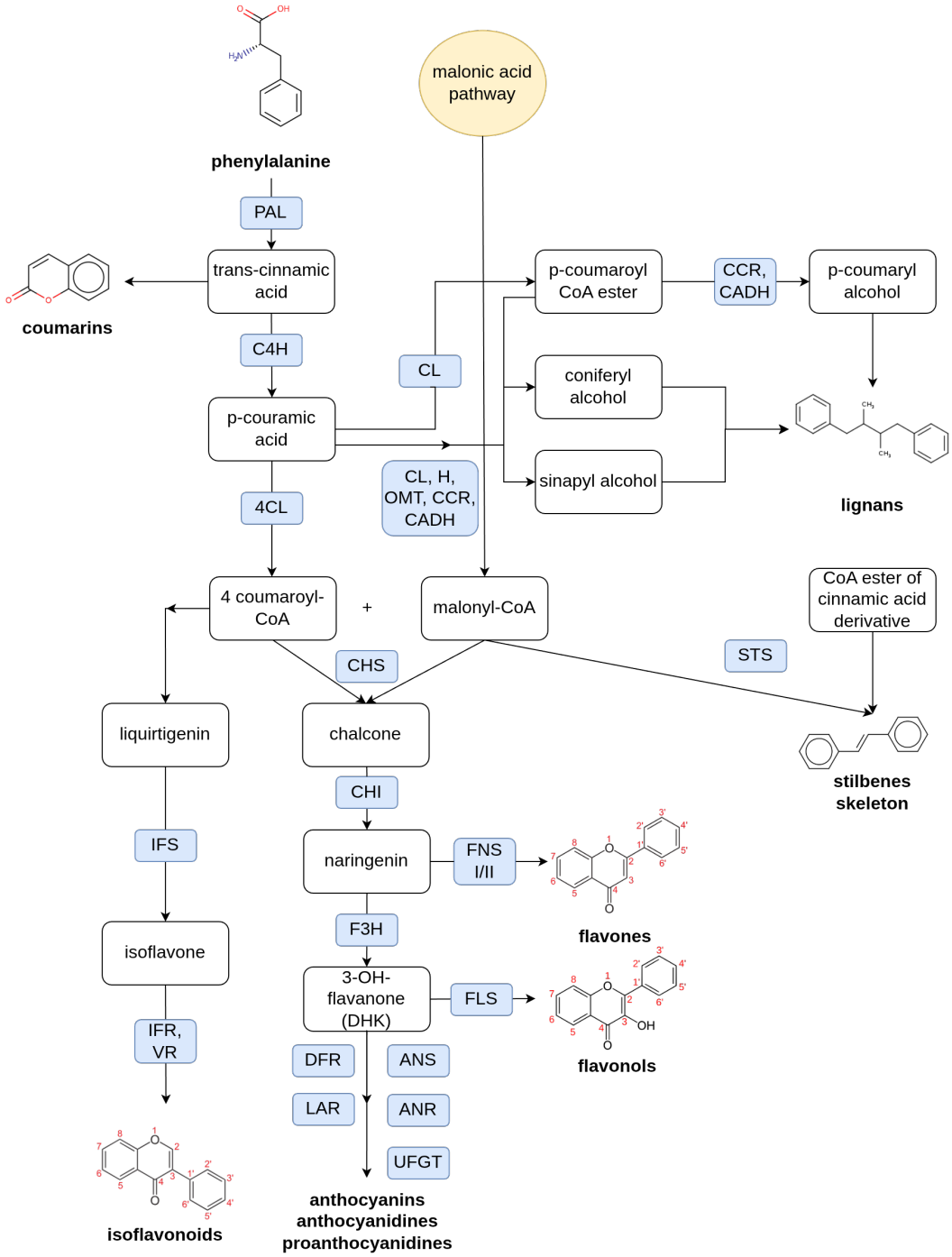


Figure 2.9: Phenolic compounds biosynthesis: main branches. This schematic view is a compilation of information from [68, 71, 73, 74, 76, 77]. PAL: Phenylalanine ammonia-lyase; C4H: Cinnamate 4-hydroxylase; 4CL: 4-coumaric CoA ligase; CL: CoA ligases; CCR: Cinnamoyl-CoA reductase; CADH: Cinnamyl alcohol dehydrogenase; H: Hydroxylases; OMT: O-methyl transferases; CHS: Chalcone synthase; STS: Stilbene synthase; CHI: Chalcone isomerase; FNS I/II: flavone synthase I/II; F3H: Flavanone 3-hydroxylase; FLS: Flavonol synthase; DFR: Dihydroflavonol 4-reductase; ANS: Anthocyanidin synthase; ANR: Anthocyanidin reductase; LAR: Leucoanthocyanidin reductase; UFGT: UDP-glucose flavonoid 3-O glucosyltransferase; IFS: Isoflavone synthase; IFR: Isoflavone reductase; VR: Vestitone reductase.

recognises that all terpenoids derive from the universal five-carbon building blocks Isopentenyl Diphosphate (IPP) and its allylic isomer Dimethylallyl Diphosphate (DMAPP) [79, 85]. As a result, the precursors for the biosynthesis of terpenoids are the prenyl diphosphate intermediates built by condensation of these two building blocks [79, 80, 82, 86].

These two building blocks can be produced both in the plastid via the Methylerythritol 4-phosphate (MEP) pathway or in the cytosol via the Mevalonate (MVA) pathway. The former produces the main precursors for the synthesis of hemiterpenoids (C5), monoterpenoids (C10), diterpenoids (C20) and carotenoids (C40); whereas the latter pathway branches towards the production of the precursors of sesquiterpenoids, sterols, and triterpenoids (C30). Figure 2.10 highlights these two main pathways and the enzymes associated with each chemical transformation.

Alkaloids

In 1819, alkaloids were defined as organic phytochemicals with an alkaline or basic character in a first description in scientific literature [87, 88]. This definition changed over the years as other alkaloids were discovered and structurally elucidated [87, 88]. Alkaloid classifications eventually led to the current definition: nitrogen-containing compounds whose nitrogen derives from an amino acid, typically incorporated into a heterocyclic ring, although in the protoalkaloids the amino-acid-derived nitrogen remains outside such a ring [87].

Alkaloids of plant origin can be further classified based on their relationship to primary metabolites. Those derived from tyrosine include benzylisoquinolines (e.g. the narcotic analgesics morphine and codeine, and antimicrobials sanguinarine and berberine), tetrahydroisoquinolines (e.g. the skeletal muscle relaxant mivacurium, the anti-hypertensive quinapril, fully synthetic pharmaceuticals that contain a tetrahydroisoquinoline moiety), and Amaryllidaceae alkaloids (e.g. galanthamine, used in the treatment of Alzheimer's disease). The tryptophan-derived alkaloids include the monoterpene indole alkaloids, such as the anti-tumorous vinblastine and vincristine. Finally, polyamine-derived alkaloids are tropane (e.g. scopolamine used to treat nausea), quinolizidine, pyrrolizidine and Nicotiana alkaloids (e.g. nicotine) [87].

The early stages of alkaloid biosynthesis are crucial and serve as entry points to a vast chemical space that is not accessible through other pathways [87]. The formation of the main classes of alkaloid scaffolds follows the pattern depicted in Figure 2.11: (i) amine accumulation, (ii) aldehyde accumulation, (iii) formation of the iminium cation and (iv) a Mannich-like reaction to form the alkaloid scaffold.

The accumulation of amine and aldehyde characterises the early steps in the biosynthesis of the main scaffolds of alkaloids. Amine accumulation can occur from polyamines or amino acids, whereas aldehydes can be formed intramolecularly or intermolecularly. This is the main variation in the aldehyde accumulation and the following steps (3 and 4): either the step occurs within the molecule or by addition from others. As shown in Table 2.2, polyamine-derived alkaloids form a cyclic iminium through intramolecular condensation, followed by an intermolecular Mannich-like reaction [87]. On the other hand, the iminium formation of amino-acid-derived alkaloids occurs intermolecularly, followed in most cases by

Table 2.2: Scaffold synthesis of major alkaloid subtypes and whether the synthesis steps are completely elucidated. Specific compounds and reaction types are referred to in italics. Adapted from [87].

Alkaloid origin	**Amine origin**	**Aldehyde origin**	**Iminium**	**Mannich-like reaction**	**Completely elucidated**
Nicotine	Polyamine, *putrescine*	Intramolecular	Cyclic iminium *pyrrolinium*	Intermolecular, *nicotinic acid*	No
Tropane	Polyamine, *putrescine*	Intramolecular	Cyclic iminium, *pyrrolinium*	Intermolecular, polyketide	No
Lycopodium	Polyamine, *cadaverine*	Intramolecular	Cyclic iminium, *piperideine*	Intermolecular, polyketide	No
Granatane	Polyamine, *cadaverine*	Intramolecular	Cyclic iminium, *piperideine*	Intermolecular, polyketide	No
Pyrrolizidine	Polyamine, *homospermidine*	Intramolecular	Cyclic iminium, *pyrrolinium*	Intramolecular	No
Betalains	*Tyrosine*, *L-DOPA*	*Tyrosine*, *betalamic acid*	Intermolecular, *spontaneous*	None	Yes
Benzylisoquinolines	*Tyrosine*, *dopamine*	*Tyrosine*, *4-HPAA*	Intermolecular, *enzyme catalysed*	Intramolecular, *Pictet–Spengler*	No
Monoterpene indoles	Tryptophan, tryptamine	Terpene, *secologanin*	Intermolecular, *enzyme catalysed*	Intramolecular, *Pictet–Spengler*	Yes
Amaryllidaceae	*Tyrosine*, *tyramine*	Phenylpropanoid, benzaldehyde	Intermolecular	Intramolecular, *reduction*	No

an intramolecular Mannich-like reaction.

As shown in Figure 2.12 and denoted in Table 2.2, several enzymes and chemical transformations are not elucidated, especially in the aldehyde accumulation and the Mannich-like reaction steps. Nevertheless, the amine accumulation step is well-known for the most studied plants. Alkaloids do not generally incorporate amino acids directly; they use a primary amine derivative. These compounds are ubiquitous in green plants, accumulating beyond typical concentrations, redirecting flux into alkaloid synthesis without disrupting existing pathways [87].

Polyamine-derived alkaloids take putrescine, spermidine, cadaverine, and homospermidine as precursors. Putrescine can derive from arginine [89] or ornithine [90] and is incorporated into the tropane alkaloid scaffold biosynthesis through *N*-methylpyrrolinium. *N*-methylpyrrolinium is formed after subsequent methyl transfer to putrescine and oxidation of *N*-methylputrescine. These two steps represent steps 1 and 2 of Figure 2.11, respectively. The spontaneous condensation of *N*-methylaminobutanal represents step 3. Finally, step 4 is characterised by the production of tropinone in the Solanaceae family

and methylecgonone in the *Erythroxylum* genus. Both are Mannich-like reactions whose processes are not known. Besides, while the production of tropinone is proven to be catalysed by a tropinone synthase (CYP82M3) [91], the same is not true for the production of methylecgonone [87].

Conversely, cadaverine derives from lysine by decarboxylation, ultimately being the precursor of *Lycopodium* and granatane alkaloids [92]. Oxidative deamination and cyclisation of cadaverine yields 5-aminopentanal [87], representing the aldehyde assimilation step. The spontaneous condensation of 5-aminopentanal yields δ1-piperideine, forming the iminium ion (step 3). Finally, the fourth step is merely hypothetical. It is thought to occur spontaneously to yield oxo-2-piperidine-butanoic acid, branching to either the spontaneous production of pelletierine or to a P450 enzymatic decarboxylation to yield norpseudopelletierine [87].

Homospermidine is the amine origin of the pyrrolizidine alkaloids and is produced from putrescine and spermidine (essential for normal plant growth) in a reaction catalysed by homospermidine synthase [87]. Although it is known that homospermidine is converted into 4-(4-Oxobutylamino)butanal followed by a conversion into pyrrolizidine to serve as a scaffold for the biosynthesis of pyrrolizidine alkaloids, the pathway is not known, both chemically and enzyme-wise [87].

The amine accumulation step can also occur from aromatic amines such as tyrosine and tryptophan *via* their primary amine derivatives, tyramine and tryptamine. The conversions are catalysed by tyrosine decarboxylase (TYDC) and tryptophan decarboxylase (TDC), respectively.

Tyramines undergo several chemical transformations into scaffolds to enter the benzylisoquinolines, Amaryllidaceae alkaloids and betalains biosynthesis pathways (Figure 2.12). One of the most important intermediates is dopamine, which is produced from tyramine or L-DOPA. It is thought that the former process of synthesis consists of hydroxylation, but the enzyme remains unknown. However, the added 3-hydroxy group is paramount for downstream *Pictet–Spengler* reactions. By contrast, the decarboxylation of L-DOPA is elucidated chemically and enzymatically, consisting of a promiscuous decarboxylase that can take L-tyrosine and L-DOPA as substrates. The second step encompasses the aldehyde assimilation from tyrosine and dopamine. To the best of our knowledge, in this step 4-hydroxyphenylacetaldehyde (4-HPAA) is formed *via* three possible branches: tyrosine transamination and subsequent decarboxylation (unknown enzyme); the action of aromatic acetaldehyde synthases (AASs) catalysing the formation of aldehydes (unknown enzyme); and from tyramine, although the metabolic and evolutionary origin remains unclear. The third step occurs spontaneously, while the fourth step occurs in a reaction catalysed by a norcoclaurine synthase (NCS) consuming dopamine (source of amine) and 4-HPAA (source of aldehyde). The product of the latter reaction, (*S*)-norcoclaurine, serves as a gateway for the biosynthesis of benzylisoquinolines.

Betalamic acid, the main precursor of betalains (betaxanthines and betacyanins) [87], can be synthesised through two different pathways depending on the amine donor used. Betacyanins are derived from cyclo-DOPA, while betaxanthins are derived from amino acids or other amines (such as dopamine or tyramine). The formation of the iminium intermediate in this process does not require enzyme catalysis [93].

Tryptamine condenses with the monoterpenoid secologanin (aldehyde origin) to form (*S*)-strictosidine, the main precursor of monoterpene alkaloids. Moreover, tyramine and 3,4-dihydroxybenzaldehyde are

used for producing Amaryllidaceae alkaloids, yielded by a chain of unclear reactions from cinnamic acid [87].

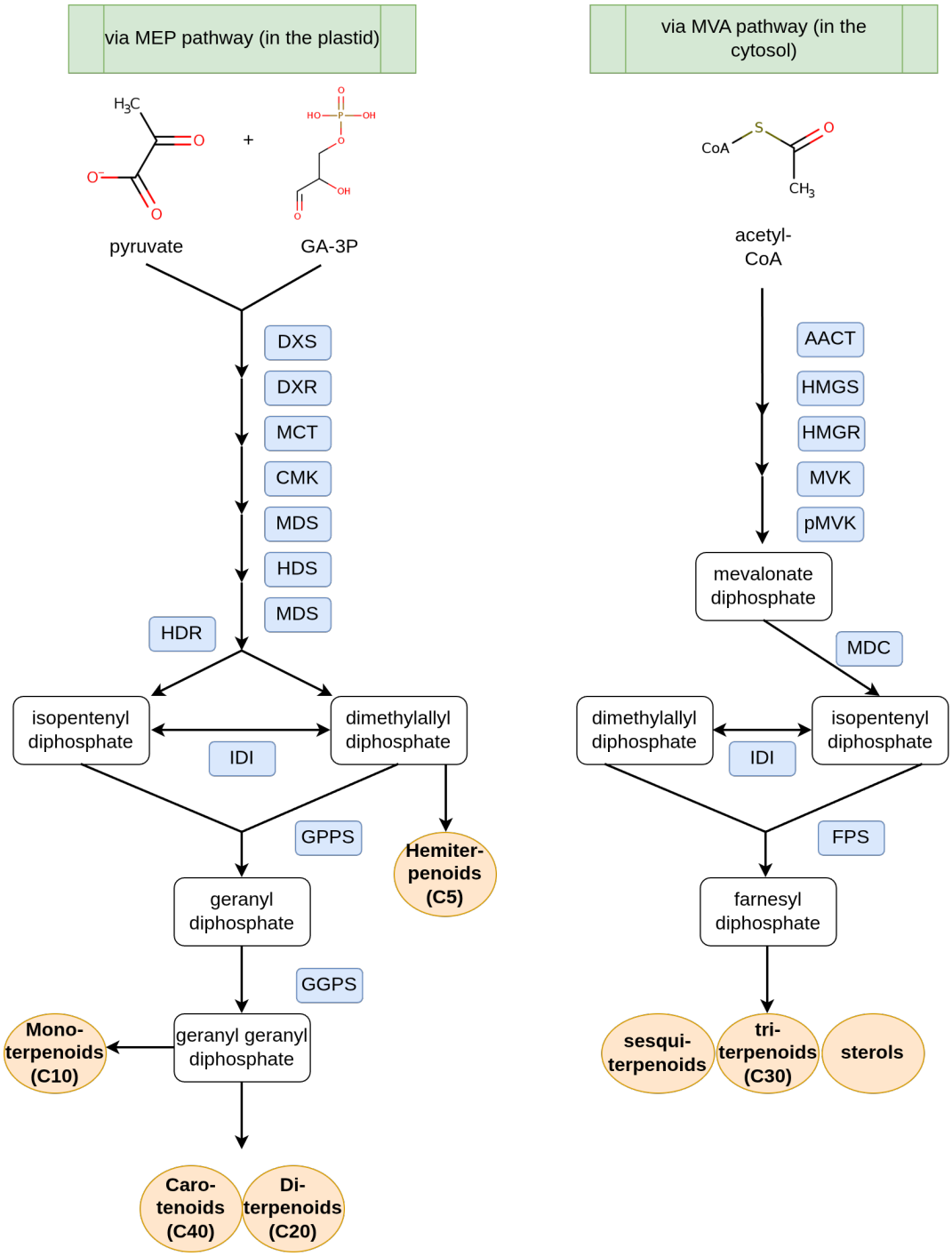


Figure 2.10: Terpenoid biosynthesis: main branches. This schematic view is a compilation of information from [79, 80, 82, 86]. AACT: acetoacetyl-coenzyme A thiolase; HMGS: 3-hydroxy-3-methylglutaryl (HMG)-CoA synthase; HMGR: HMG-CoA reductase; MVK: mevalonate kinase; pMVK: phosphomevalonate kinase; MDC: mevalonate diphosphate decarboxylase; DXS: 1-deoxyxylulose-5-phosphate synthase; DXR: 1-deoxy-d-xylulose-5-phosphate reductoisomerase; MCT: 2-C-methyl-d-erythritol 4-phosphate cytidylyltransferase; CMK: 4-diphosphocytidyl-2-Cmethyl-d-erythritol kinase; MDS: 2C-methyl-d-erythritol 2,4-cyclodiphosphate synthase; HDS: 4-hydroxy-3-methylbut-2-enyl diphosphate synthase; HDR: 4-hydroxy-3-methylbut-2-enyl diphosphate reductase; IDI: IPP/DMAPP isomerase; GPPS: geranyl diphosphate synthase; GGPS: geranylgeranyl diphosphate synthase; FPS: farnesyl diphosphate synthase.

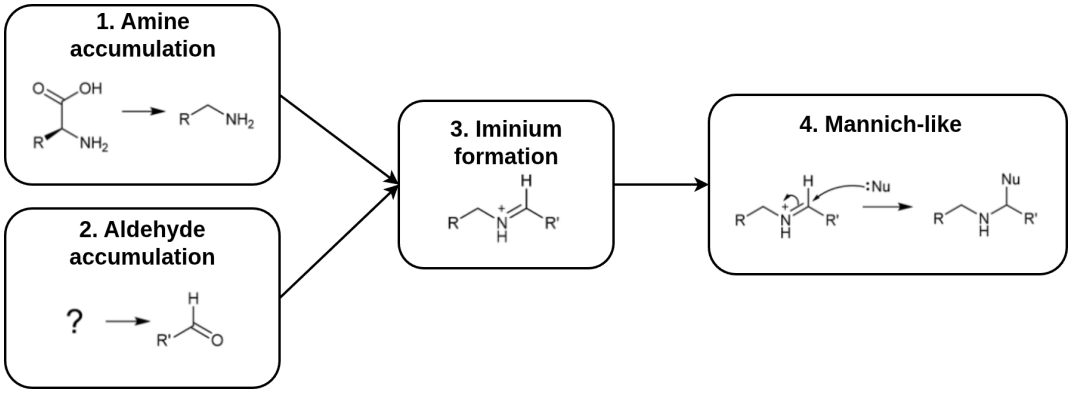


Figure 2.11: The early steps of alkaloid biosynthesis. Adapted from [87].

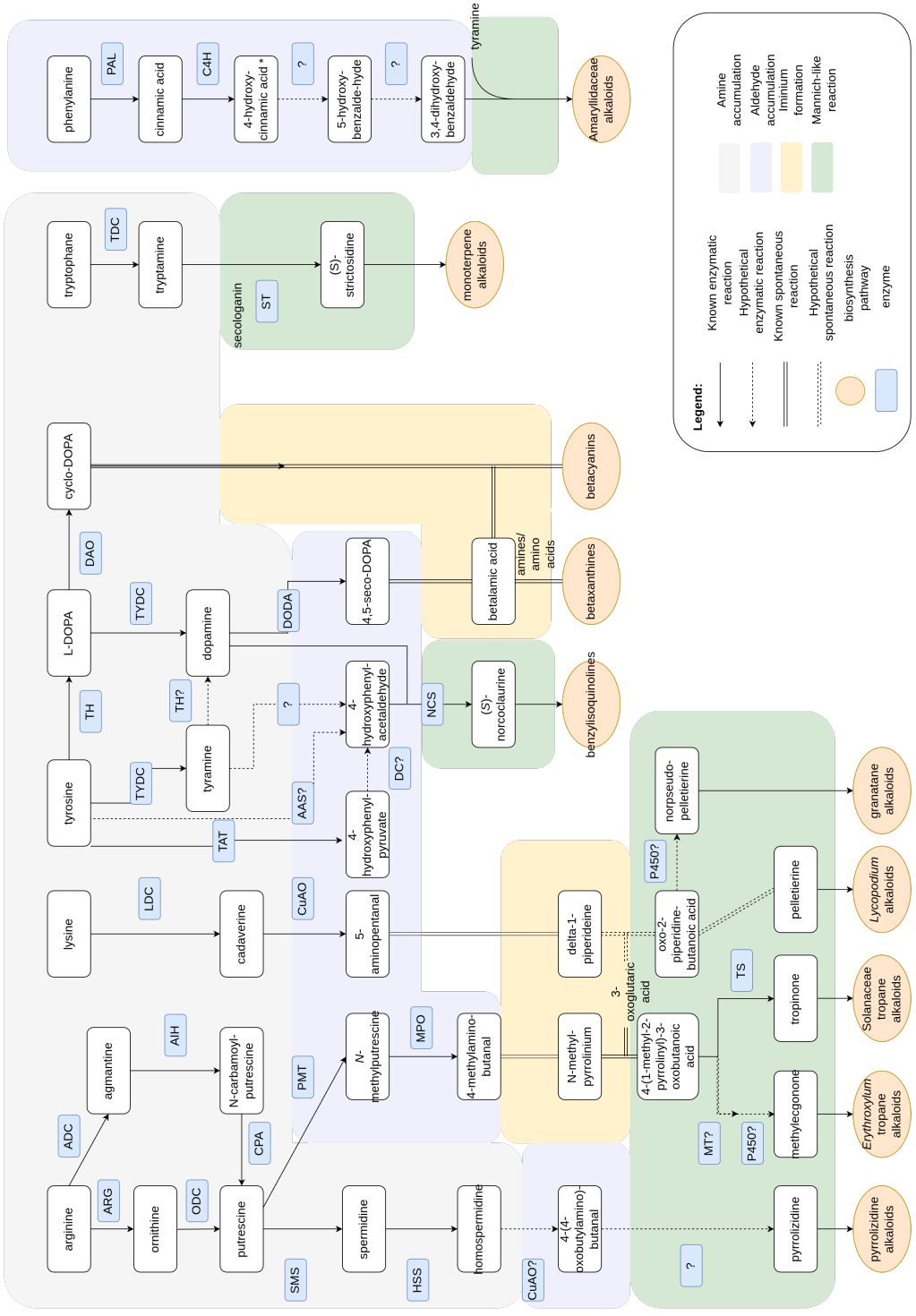


Figure 2.12: Production of the main precursors of alkaloids. Information compiled from the different sections of a remarkably state-of-the-art review by [87]. ARG: arginase; ADC: arginine decarboxylase; ODC: ornithine decarboxylase; AIH: agmatine iminohydrolase; CPA: N-carbamoylputrescine aminohydrolase; LDC: lysine decarboxylase; SMS: spermidine synthase; HSS: homospermidine synthase; TYRAAT: arogenate dehydrogenase; TYDC: tyrosine decarboxylase; TH: tyramine hydroxylase; DAO: DOPA oxidase; DODA: 4,5-DOPA dioxygenase; TAT: tyrosine aminotransferase; AAS: aromatic acetaldehyde synthase; DC: decarboxylase; PMT: putrescine methyltransferase; MPO: N-methylputrescine oxidase; CuAO: copper-containing amine oxidases; P450: cytochrome P450 enzyme; MT: methyltransferases; TS: Tropinone synthase CYP82M3; NCS: norcoclaurine synthase; PAL: phenylalanine ammonia lyase; C4H: cinnamate 4-hydroxylase; ST: strictosidine synthase.

2.3 Machine learning

ML is a sub-field of AI, concerning how computers can learn automatically from data or experiences [94]. ML algorithms are programmed to learn from samples conventionally referred to as training data. After training, such models make predictions based on the patterns learned from the training data [95, 96].

It is common practice in ML to evaluate these models' performance with a test dataset. Accordingly, the performance of the trained models is assessed by making predictions for examples they have not been exposed to. After learning from the training examples, these predictions provide information on their ability to generalise or solve real problems. It is paramount to ensure that examples in the training set are not repeated in the test set. This overlap is one form of data leakage, the broader problem of information from outside the training set influencing the model, which can make a model appear accurate while generalising poorly to real-world cases. In addition, it is essential to ensure the quality and reliability of the data and their endpoints to build reliable models.

The widespread availability of data and advances in hardware have made ML the most popular subfield of AI. Its scalability and efficiency in tackling tasks like computer vision, Natural Language Processing (NLP), and biological problems [94] have played a major role in its success.

ML is commonly divided into supervised, unsupervised, and reinforcement learning, with semi-supervised learning often regarded as a hybrid of the supervised and unsupervised paradigms. Supervised Learning uses labelled datasets to construct predictive models, predicting and classifying data according to well-defined outcomes. In contrast, unsupervised learning uses algorithms to cluster and analyse data, aiming to discover patterns without human intervention or label learning. Semi-supervised learning encompasses model training with a small number of labelled examples and many unlabelled examples. Finally, in reinforcement learning, practitioners develop methods for rewarding desired behaviours and punishing undesirable ones.

Supervised learning is a type of machine learning in which algorithms learn from labelled data by adjusting their parameters to minimise the difference between the predicted and the correct label [95, 96]. Training examples are associated with either a category or a continuous value, and the model adjusts its parameters to minimise misclassification or prediction error. In iterative methods such as logistic regression and neural networks, the parameters are initialised and updated until convergence, whereas other models, such as decision trees and ordinary least squares, reach their solution greedily or in closed form [95]. This learning paradigm is the one considered most relevant for the present work, and will be the focus of the following sections.

Below, the main tasks in supervised learning are enumerated: classification, encompassing binary and multi-class classification, and regression [95].

2.3.1 Classification tasks

Binary classification involves classifying data examples into one of two categories, usually called the negative and positive classes. In this type of classification, a single instance, represented by a numerical

Table 2.3: Most used metrics in classification tasks and remarks concerning their proper usage.

Metric	Formula	Remarks
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Presents performance at a single class threshold and assumes the equal cost for errors [97, 98].
MCC	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	When MCC $= 1$ then the prediction is perfect, 0 is equivalent to a random prediction and -1 means a complete disagreement between the predicted and true labels [97].
PPV or Precision	$\frac{TP}{TP+FP}$	Perfect PPV is 1 and the worst value is 0.
TPR or Sensitivity or Recall	$\frac{TP}{TP+FN}$	Proportion of actual positives that are correctly identified.
TNR or Specificity or selectivity	$\frac{TN}{TN+FP}$	Proportion of actual negatives that are correctly identified.
F1	$2 \times \frac{(PPV \times TPR)}{(PPV+TPR)}$	Harmonic mean of PPV and TPR, ignoring true negatives and depending on which class is designated as positive [99].
Balanced Accuracy	$\frac{TPR+TNR}{2}$	Accommodates imbalanced datasets.
NPV	$\frac{TN}{TN+FN}$	Ideal value is 1 and the worst value is 0.

TP: True Positives; TN: True Negatives; FP: False Positives; FN: False Negatives

vector, is given as input, belonging either to the positive or negative class. Each instance in a dataset is notated as $x \in \mathbb{R}^d$, given as a d-dimensional numerical vector, with a label notated as $y \in \{+, -\}$.

Multi-class classification tasks are an extension of binary classification. In this case, models learn to classify examples according to M labels, where $M \geq 3$. Hence, each data instance is notated as $x \in \mathbb{R}^d$, given as a d-dimensional numerical vector, with a label notated as $y \in \{1, 2, \ldots, M\}$.

As mentioned, after training, models should be assessed. Adequate functions should be used to measure the error between the predictions made and the real data. Table 2.3 lists some of the most used metrics in classification tasks and discusses their proper use.

Other methods of evaluating ML models are based on function curves, such as Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves and their Area Under the Curve (AUC). The ROC curve plots TPR (y axis) and False Positive Rate (FPR) (x axis), both taking a value between 0 and 1 at

different classification thresholds, and should be used on balanced datasets [97, 100]. On the other hand, the PR is mainly applicable in cases of moderate to large class imbalance [97, 100] and is mostly used in binary classification. A PR curve plots precision (y axis) and recall (x axis) at different classification thresholds. AUC reflects how good both curves are with a single value score. The more towards the top-left a ROC curve reaches, the better, whereas the more top-right a PR curve reaches, the better the model. ROC AUC represents how well the model ranks predictions, giving the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance. The PR AUC represents the average of all precision scores in relation to all recall thresholds.

2.3.2 Regression tasks

In regression, a single example, represented by a numerical vector, is given as input, and a continuous numerical value is predicted. Each instance in a dataset is notated as $x \in \mathbb{R}^d$, given as a d-dimensional numerical vector, with a label notated as $y \in \mathbb{R}$.

Table 2.4 enumerates some of the most used metrics in regression tasks together with remarks concerning their proper use.

2.3.3 Models and algorithms

ML depends on high-quality data derived from relevant and reliable sources for effective algorithm training. Standard ML workflows involve integrating these data into curated datasets, followed by pre-processing and partitioning into training, validation, and test sets. Additionally, resampling techniques, such as cross-validation or bootstrapping, are often employed to improve model generalisation, reduce overfitting, and provide more robust estimates of predictive performance.

The training set is used to train the model, the validation set guides hyperparameter optimisation, and the test set evaluates the model's ability to generalise to real-world cases. Subsequent steps involve feature engineering, data selection, and scaling, producing the numerical representations (vectors, matrices, or arrays) required for model training. Throughout training and optimisation, models are rigorously evaluated, and the best-performing model(s) are selected for deployment to make predictions on new data.

Linear models

Linear regression is a statistical method for regression problems which relies on modelling the relationship between a dependent variable (Y) and one or more independent variables ($X = (x_1, x_2, ..., x_{n-1}, x_n)$) by fitting a linear equation to the observed data [101]. The basic formulation of linear regression is the following [101]:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon \tag{2.4}$$

Table 2.4: Most used metrics in regression tasks and remarks concerning their proper usage.

Metric	Formula	Remarks
Error (E)	$Actual(A) - Predicted(P)$	Describes how much predictions differ from their actual values.
ME	$\frac{\sum_{i=1}^{n} E_i}{n}$	Defines the average of all errors in a dataset. It may not be helpful in cases where positive and negative predictions cancel each other out [97].
MAE	$\frac{\sum_{i=1}^{n} \lvert E_i \rvert}{n}$	Measures the absolute difference between predictions and actual values.
MNB	$\frac{\sum_{i=1}^{n} E_i/A_i}{n}$	Average of the errors normalised by the actual values, expressing bias on a relative scale; as with ME, opposite-signed errors may cancel.
MSE	$\frac{\sum_{i=1}^{n} E_i^2}{n}$	Scale-dependent and penalises large errors quadratically, making it sensitive to outliers [97].
RMSE	$\sqrt{\frac{\sum_{i=1}^{n} E_i^2}{n}}$	Scale-dependent and, like MSE, very sensitive to outliers, but expressed in the same units as the target [97].
R	$\frac{\sum_{i=1}^{n}(A_i-\bar{A})(P_i-\bar{P})}{\sqrt{\sum_{i=1}^{n}(A_i-\bar{A})^2}\sqrt{\sum_{i=1}^{n}(P_i-\bar{P})^2}}$	Measures the linear correlation between variables and is invariant to equal scaling. $R > 0.8$ implies a strong correlation [97].
R^2	$1 - \frac{\sum_{i=1}^{n}(P_i-A_i)^2}{\sum_{i=1}^{n}(A_i-\bar{A})^2}$	Proportion of variance explained by the model; values close to 1 indicate a good fit, and the metric can be negative for poorly fitting models.

where Y is the dependent variable, $x_1, x_2, ..., x_n$ are the independent variables, β_0 is the y-intercept (the value of Y when all the Xs are equal to 0), β_1, β_2, ..., β_n are the coefficients or weights of the independent variables and ϵ is a term that represents the random error or noise in the data.

The goal of linear regression is to find the values of $\beta_0, \beta_1, \beta_2, ..., \beta_n$ that minimise the difference between the predicted values $\hat{Y}$ and the actual values of Y. This is done using a method called least squares, which finds the line that minimises the sum of the squares of the residuals (the difference between the predicted and actual values of Y), or gradient descent, which iteratively updates the coefficients in the direction opposite to the loss gradient, scaled by a learning rate, until convergence.

The *logistic regression* model uses the logistic function (also called the sigmoid function) to model a binary dependent variable [102]. The logistic function is defined as follows for a single independent variable:

$$p(x_1) = 1/(1 + e^{(-\beta_0 - \beta_1 x_1)}) \tag{2.5}$$

where $p(x)$ is the probability that the dependent variable equals 1, given the predictor variable x, and β_0 and β_1 are the model parameters that need to be estimated from the data. The logistic regression model can also be extended to multiple independent variables (features) by using the following equation:

$$p(X) = 1/(1 + e^{(-\beta_0 - \beta_1 x_1 - \beta_2 x_2 - ... - \beta_n x_n)}) \tag{2.6}$$

where $x_1, x_2, ..., x_n$ are the feature variables and $\beta_1, \beta_2, ..., \beta_n$ are the corresponding model parameters.

Decision trees and random forests

A decision tree is a tree-like structure in which an internal node represents a feature (or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The basic idea behind any decision tree is to map observations about an item to conclusions about the item's target value. It works by recursively partitioning the data into subsets based on the values of the input features. This process is repeated on each subset, creating a tree-like structure with decision and leaf nodes. The decision nodes represent the input features, and the leaf nodes represent the output class labels or target values. The goal of training is to build an optimal tree, which can be used to predict the target value by traversing from the root to the leaf node. [103]

Random Forest is an ensemble learning method that constructs various decision trees during training. Each tree is trained on a bootstrap sample of the data, drawn with replacement, and at each split only a random subset of the features is considered [103]. This randomness helps to make the model more robust than a single decision tree and less susceptible to overfitting [103].

Support vector machines

Support Vector Machines (SVM)s are a type of supervised learning algorithm that can be used for classification or regression tasks. The basic idea behind SVMs for classification is to find a hyperplane (a line or

a higher-dimensional plane) that separates the different classes in the feature space [103]. The goal of an SVM is to find the best hyperplane that maximally separates the different classes while also maximising the margin, which is the distance between the closest data points of each class to the hyperplane [103].

2.4 Deep learning

Conventional ML techniques have limited capabilities for processing raw data on compounds, proteins, and reactions [104]. In contrast, DL techniques offer an alternative for handling raw biological data. In the past, it was necessary to use expert knowledge and carefully designed features to transform raw data into a format that machine learning models could use. With *representation learning*, however, machines can learn to classify or regress on raw data directly. In general, DL systems are made up of simple, nonlinear modules that can transform raw input data into more abstract, higher-level representations [104].

DL systems offer several advantages over traditional ML methods, including learning from raw data without the need for extensive feature engineering and scaling to larger datasets. However, DL systems have limitations, such as the need for large data and longer training times.

2.4.1 Neural networks

The perceptron, a computational neuron, was the prototype of a neural network designed to imitate the structure of a biological neuron. Its structure is shown in Figure 2.13. It starts by receiving an input vector $X = (x_1, x_2, ..., x_{n-1}, x_n)$, which is then multiplied by the weights of the model and added to a bias parameter (w_0 or b), $W = (w_0, w_1, w_2, ..., w_{n-1}, w_n)$. Afterwards, the dot product $X^T W$ is computed, followed by the application of a step function (f). In short, the resultant output Y is computed as follows: $Y = f\left(\sum_{i=0}^{n} w_i x_i\right)$ (where $x_0 = 1$) or $Y = f(\sum_{i=1}^{n} w_i x_i + b)$.

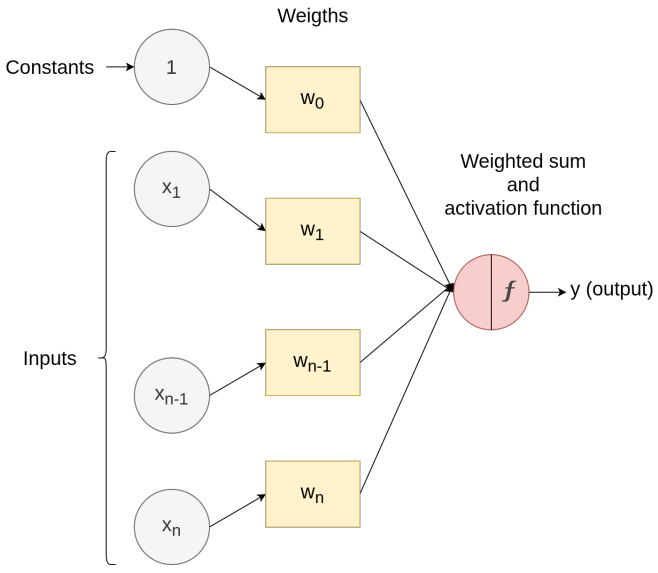


Figure 2.13: Architecture of a perceptron.

A perceptron could use either a Heaviside or sign function, presented in equation 2.7 and 2.8 (assuming a threshold of 0), respectively.

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad (2.7) \qquad sgn(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases} \quad (2.8)$$

Over the years, more complex architectures were constructed based on this *rationale*. While a perceptron is composed of a single layer of neurons, a DNN is implemented with multiple layers, also known as Fully Connected Network (FCN)s, (all the neurons in one layer are connected to all the neurons in the following layer) with multiple neurons stacked in more than one layer. Multilayer Perceptron (MLP)s are DNNs composed of one *input layer*, one or more layers, also known as *hidden layers* and one *output layer*. All layers except the output layer include a fully connected bias neuron.

2.4.2 Training process

The training process of a neural network was revolutionised by the discovery of the *backpropagation* method. Backpropagation is a training algorithm popularised for neural networks in 1986, consisting of using gradient descent along with an efficient technique for computing the gradients automatically based on the chain rule [105].This technique automatically computes the network's error considering every model parameter (weight) in two steps (one forward and another backward). In summary, the algorithm determines how changing each weight and bias parameter affects the error and adjusts them to minimise it. The algorithm works as follows [103]:

1. The network takes a mini-batch at a time as an input, passing through the whole training set.
2. *Forward pass*. Each mini-batch passes through the input layer and is sent to the first hidden layer. Each neuron computes the input and generates an output passed to the next layer. This process happens from the input layer to the output layer. The intermediate results in each layer are preserved as they are then used in the *backward pass* step.
3. Loss calculation. A *loss function* is computed to measure the network's output error.
4. *Backward pass*. The algorithm determines how much each output connection contributes to the error, working backwards through the network.
5. Optimisation step. The algorithm tweaks all the connection weights in the network using the error gradients computed beforehand.

Since step functions have flat segments with no gradient, gradient descent cannot find the optimal solution. Therefore, other types of functions must be used instead. These functions are called *activation*

Table 2.5: Last-layer activation and loss functions according to the problem type. [107].

Problem Type	Last-layer activation	Loss function
Binary classification	sigmoid	binary cross-entropy
Multi-class, single-label classification	softmax	categorical cross-entropy
Multi-class, multi-label classification	sigmoid	binary cross-entropy
Regression to arbitrary values	-	MSE
Regression to values between 0 and 1	sigmoid	MSE or binary cross-entropy

functions and work similarly and in the same step in the pipeline as step functions. However, they have a well-defined non-zero derivative. This feature allows the gradient descent to make progress during training. Figure 2.14 depicts some of the most common activation functions.

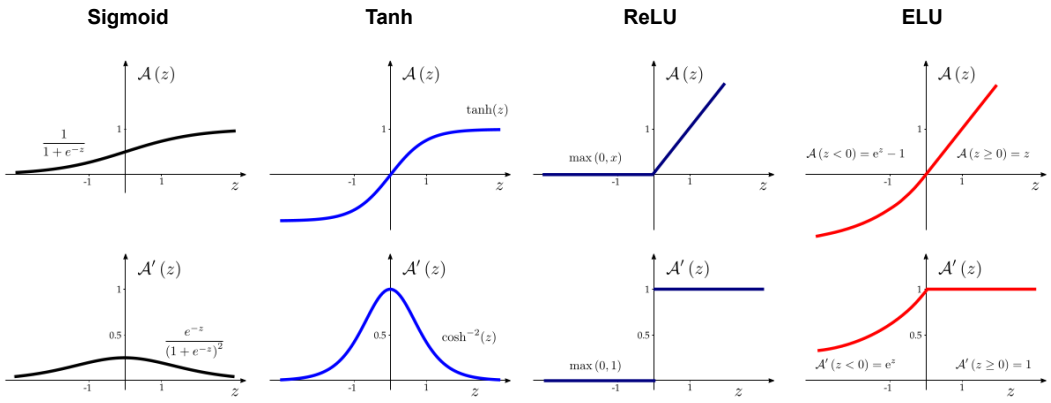


Figure 2.14: Common activation functions and their derivatives. Adapted from [106].

When training a model, it is necessary to select two essential functions: the activation function for the output layer and the loss function. These functions should be chosen based on the specific task and the metric to be optimised. However, deciding on the appropriate loss function is not always straightforward. Table 2.5 lists output activation and loss functions for common problem types.

2.4.3 Optimising a neural network

In the optimisation step, one can use different optimisation configurations (i.e. gradient descent algorithm variants and learning rate) [107]. In this context, optimising a neural network includes minimising an objective function $J(\theta)$, where $\theta \in R^d$ is a set of model parameters. Accordingly, gradient descent updates the parameters θ of a model to minimise the loss function J, quantifying the network error. More specifically, the parameters are updated in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$, considering a learning rate μ that determines the size of the steps one takes to reach a local minimum [108].

Table 2.6 summarises the gradient descent variants and the most common algorithms to optimise gradient descent, along with their parameter update rules.

Table 2.6: Most common gradient descent variants and respective update rules.

Algorithm	Update rule	Remarks
Batch gradient descent	**(a)** $\theta = \theta - \mu.\nabla_\theta J(\theta)$	Computes the gradient of the cost function with regard to θ for the whole training dataset ($\nabla_\theta J(\theta)$). It can be slow as it has to compute the gradients for the whole dataset. Due to this limitation, such a method cannot be used for datasets that do not fit in memory. The update rule is defined as in **(a)**, considering the learning rate parameter μ. [108]
Stochastic gradient descent	**(a)** $\theta = \theta - \mu.\nabla_\theta J(\theta; x_i; y_i)$	Computes the gradients for each training example x_i and label y_i ($\nabla_\theta J(\theta; x_i; y_i)$). It is much faster than batch gradient descent, as it performs one update at a time. Due to the frequent weight updates with high variance, the objective function tends to fluctuate heavily, enabling it to jump from worse local minima to better ones. The update rule is defined as in **(a)**, considering the learning rate parameter μ. [108]
Mini-batch gradient descent	**(a)** $\theta = \theta - \mu.\nabla_\theta J(\theta; x_{(i:i+n)}; y_{(i:i+n)})$	Updates the weights θ for every mini-batch of *n* examples ($\nabla_\theta J(\theta; x_{(i:i+n)}; y_{(i:i+n)})$). The variance of the parameter updates is decreased, leading to more stable convergence. The update rule is defined as in **(a)**, considering the learning rate parameter μ. [108]
Momentum [109]	**(a)** $v_t = \gamma v_{t-1} + \mu.\nabla_\theta J(\theta)$; **(b)** $\theta = \theta - v_t$	Momentum hastens the convergence of stochastic gradient descent and smooths oscillations. This is possible by adding a fraction γ (momentum term) of the last step's update vector (v_{t-1}) to the current step's vector (equation **(a)**). The update rule is defined as in **(b)**, considering the learning rate parameter μ.

AdaGrad [110]	**(a)** $g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$; **(b)** $\theta_{t+1,i} = \theta_{t,i} - \frac{\mu}{\sqrt{G_{t,ii}+\epsilon}}.g_{t,i}$	AdaGrad uses a different learning rate for each parameter θ_i at every time step t, based on the past gradients. This is formulated by the division of the learning rate μ by the squared root of a diagonal matrix $G_{t,ii}$, where each element in the diagonal is the sum of squared gradients g (equation **(b)**). An additional parameter ϵ is added to $G_{t,ii}$ to avoid division by zero. Although manually tuning the learning rate is no longer necessary when using AdaGrad, it accumulates squared gradients in the denominator, decreasing the learning towards an infinitesimally small number, and hindering the algorithm from learning additional knowledge. [108]
RMSProp	**(a)** $E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2$; **(b)** $\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{E[g^2]_t+\epsilon}}.g_t$	RMSProp appears as an extension of AdaGrad to mitigate the aggressive monotonic decrease of the learning rate μ. To do this, instead of accumulating all past squared gradients g^2, this algorithm defines the sum of the gradients g as the decaying average of all the past squared gradients (equation **(a)**). The update rule is defined as in **(b)**.
Adam [111]	**(a)** $v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$; **(b)** $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$; **(c)** $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$; **(d)** $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$; **(e)** $\theta_{t+1} = \theta_t - \frac{\mu}{\sqrt{\hat{v}_t}+\epsilon}.\hat{m}_t$;	Adam is another method for adaptive learning rate computation. This algorithm stores an exponentially decaying average of past squared gradients g^2 (equation **(a)**) and also an exponentially decaying average of past gradients g (equation **(b)**), like momentum. In the initial steps, as m_t and v_t are initialised as vectors of zeros, they are biased towards zero. Therefore, bias-corrected first- and second-moment estimates are applied to v_t and m_t (equation **(c)** and **(d)**), which are then used to update parameters as in equation **(e)**. The authors propose the default values of 0.9 for β_1, 0.999 for β_2 and 10^{-8} for ϵ.

2.4.4 Bias-variance trade-off

The bias-variance trade-off is worth considering when optimising a neural network. *Bias* refers to the difference between the predictions of a neural network and the actual values, on average over the distribution of datasets. A model with high bias underfits the training data, i.e. it cannot fully predict the outcome, for example when it is based on wrong assumptions such as that a relation is linear when it is not [103]. In contrast, *variance* refers to the model predictions' variability in outcome over various dataset instantiations

[103]. A model with high variance overfits the training data and fails to generalise to new examples. As depicted in Figure 2.15, increasing the complexity of the model (e.g., the number of parameters) tends to reduce bias but also increases variance, leading to a U-shaped validation error curve. In practice, high variance translates into a much lower training error than validation error, assuming that the training error is considerably low. A high training error indicates a model with high bias; if the validation error is in turn much higher than the training error, the model also has high variance. On the other hand, if the training and validation errors are both low and not that different, the model will have low bias and variance.

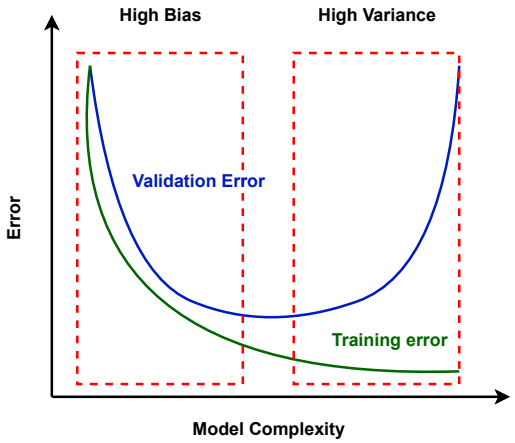


Figure 2.15: Bias-variance trade-off.

Although the bias-variance trade-off has been a dominant dogma in ML [112], recent empirical results with over-parameterised neural networks point to another direction. Rather than following the classic U-shaped test error curve, they exhibit a *double descent*: the error decreases, rises to a peak near the interpolation threshold, and then decreases again as model complexity continues to grow [113, 114].

Analysing bias and variance is an essential step when developing DL models, and standard methodologies have been established to address each problem. A high-bias model underfits the data, while a high-variance model overfits it. A basic ML recipe can be defined as follows:

1. Model with high bias:
    a) Increase model complexity;
    b) Train the model longer;
    c) Enlarge the Neural Network architecture (e.g. number of neurons per layer, number of layers, etc);
    d) Optimise hyperparameters (e.g. learning rate, optimiser, activation function, batch size);
2. Model with high variance:
    a) Find more data;
    b) Apply regularisation techniques to the network;

c) Simplify the Neural Network architecture (e.g. number of neurons per layer, number of layers, etc);

d) Optimise hyperparameters (e.g. learning rate, optimiser, activation function, batch size);

Optimising hyperparameters and neural network architectures can be time-consuming and demanding resource-wise, especially when the data volume is high. Nevertheless, several employed heuristics and other approaches can help achieve competitive results, namely random and grid searches, Bayesian optimisation, and evolutionary algorithms [103].

Models with high variance likely overfit the training data and struggle to predict correctly for unseen examples. Several regularisation techniques can help with this problem:

- **Dropout mask** [115]: in each training step every neuron, except those in the output layer, has a *p* probability of being temporarily ignored (i.e. its output is set to zero) and it can be active again in the next training step;
- **Batch normalisation** [116]: primarily a technique to stabilise and accelerate training, with a regularising side effect, that adds a normalisation operation before or after the activation function of each layer [103]. This is normally a layer that zero-centres and normalises each input, then scales and shifts the result with two learnable parameters. This allows the model to learn the optimal scale and mean of each of the layer's inputs [103, 107].
- **Early stopping**: the process of stopping model training when the validation loss increases after *n* iterations, as it indicates that the model is starting to overfit the training data.
- **L1 regularisation**: a method that adds a penalty term based on model complexity to the loss function. This penalty term consists of a regularisation parameter λ multiplied by the sum of the absolute values of the weight parameters in a weight matrix $||\theta||_1 = \sum_i \sum_j |\theta_{ij}|$. Then, the final loss will be computed as follows: $\hat{J}(\theta) = J(\theta) + ||\theta||_1.\lambda$.
- **L2 regularisation**: a method that adds a penalty term based on model complexity to the loss function. This penalty term consists of a regularisation parameter $\frac{\lambda}{2}$ multiplied by the Euclidean Norm (or L2 norm) of the weight matrices (the sum of all squared weight values of the weight matrix) $||\theta||_2^2 = \sum_i \sum_j \theta_{ij}^2$. Then, the final loss will be computed as follows: $\hat{J}(\theta) = J(\theta) + ||\theta||_2^2 . \frac{\lambda}{2}$.

2.4.5 Convolutional neural networks

Convolutional Neural Networks (CNN)s are DNNs whose neurons are organised into $n + 1$ dimensions: the spatial dimensionality of the data (n, where $n = 1$ for 1D CNN, $n = 2$ for 2D CNN, and $n = 3$ for 3D CNN) and a channel dimension. Channels represent distinct feature maps at each spatial location — for instance, in a protein sequence where each position corresponds to an amino acid, channels may encode different properties of that amino acid, such as its one-hot encoded identity across the 20 canonical amino acids, or learned feature detectors in deeper layers of the network. Moreover, one of the main differences

compared to a FCN is that the neurons in a given layer will only connect to a small region of the preceding layer [117].

Regarding architecture, CNNs are typically composed of three layer types: convolutional layers, pooling layers and fully-connected layers. *Convolutional layers* focus on the use of learnable kernels [117]. These layers convolve each filter (a convolutional layer parameter) across the input data to produce activation maps or feature maps [103]. These activation maps consist of an activation function applied to the scalar product of the weights (filters) and the input data plus bias. Each kernel will have a corresponding activation map and be stacked along the depth dimension to form the full output volume [117].

A convolutional layer has three hyperparameters that control the size of the output volume: depth, stride and zero-padding. The *depth* corresponds to the number of feature maps in the output volume, set by the number of filters in the layer. The *stride* is the parameter that defines the step size with which each filter moves through the input data. If the stride is small, the overlap between receptive fields (small region connected to one neuron) will be high, producing large activation maps. On the contrary, if the stride is high, the overlap is reduced and it will produce an output of lower spatial dimensions. *Zero-padding* is the process of padding the input's border, providing control of the dimensionality of the output volumes. [117]

The output of a neuron in a 2D convolutional layer is computed as follows:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} . w_{u,v,k',k} \text{ with } \begin{cases} i' = i \times s_h + u \\ \\ j' = j \times s_w + v \end{cases} \tag{2.9}$$

where $z_{i,j,k}$ is the output of the neuron located in row i, column j in feature map k of the layer l. s_h and s_w are the vertical and horizontal strides, f_h and f_w are the height and width of the receptive field, and $f_{n'}$ is the number of feature maps in the $(l-1)^{th}$ layer. $x_{i',j',k'}$ is the output of the neuron of the previous layer, b_k is the term of bias for feature map k, and $w_{u,v,k',k}$ is the weights matrix of the neurons located in feature map k of the layer l connected to the input data at row u, column v and feature map k' [103]. These layers can be either 1D, 2D or 3D, providing suitable architectures to deal with protein and compound sequences (1D) [118, 119], 2D encodings [120], and also with 3D structures [118, 121, 122].

The *pooling layer* aims to reduce the representation's dimensionality, the number of parameters and the computational complexity of the model. These layers operate over the feature maps and scale the data using a specific function. Commonly, pooling layers scale the data using the *max* function, thus adopting the name of *max-pooling* layers. However, there are other ways of scaling the data, using the *average* function or applying an L1/L2-normalisation [103, 117].

2.4.6 Attention mechanism and transformers

Arguably, the *attention mechanism* has become one of the most critical concepts in the DL field. It is inspired by human visual perception, which involves focusing on specific regions of an image to perceive

them in high resolution. In contrast, peripheral objects are perceived at a lower resolution [123]. This mechanism was first applied to machine translation in 2014 [124] providing performance improvements and bringing more interpretability to DL models [125].

To truly understand the attention mechanism, one has to consider the encoder-decoder framework. Conceptually, the encoder generates a trainable intermediate representation of the input sequence, whereas the decoder generates the output sequence from that representation. The encoder-decoder framework can employ different models such as Recurrent Neural Networks (RNN)s and CNNs [126].

The attention mechanism computes a context vector c that represents how the network's output relates to the parts of the input sequence. This vector c accesses all the hidden states h_j of an encoder to perform an alignment α_{tj}. This alignment function computes the similarity between the j^{th} segment of the input sequence at the time step t of the encoder and the $(t-1)^{th}$ output of the decoder. The context vector is computed as follows:

$$c_{(t)} = \sum_{j=1}^{T} \alpha_{tj} h_j \tag{2.10}$$

This means that at each time step t, the context vector $c_{(t)}$ is being computed as the weighted sum of the hidden states h, where each attention weight α_{tj} is computed as follows:

$$e_{tj} = a(s_{(t-1)}, h_j) \tag{2.11}$$

and

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})} \tag{2.12}$$

where a is a learnable score function (e.g. an additive feedforward network or a dot product) that reflects the importance of the hidden states h_j to the next hidden state s_t, according to the state s_{t-1}. Essentially, a sequence of words is presented to the attention mechanism and is scored (e_{tj}) against each encoder hidden state h_j, each of which corresponds to a position in the input sequence. Accordingly, the attention mechanism reveals how the word relates to others in the sequence. Finally, it scales ($c_{(t)}$) the values according to the attention weights α_{tj} to focus on the words that are relevant to the sequence [125].

Transformer network

Transformers [127] are a type of DL architecture based primarily on self-attention mechanisms, enabling the parallel processing of all elements in a sequence. They are commonly used in NLP tasks and can process raw sequence data. To take the order of the elements in the sequence into account, Transformers use a positional embedding, which encodes the position of each element in the sequence. This allows the model to correctly process the input data, even when it is processed in parallel.

In the original paper [127], the positional encodings are generated using the sinusoidal functions shown in Equations 2.13 and 2.14. For each sequence position pos (e.g. the pos^{th} amino-acid in the

protein sequence), these functions produce a vector of dimension d_{model}, which is added to the corresponding token embedding to encode positional information. The token embeddings capture semantic information, while the positional encodings provide explicit information about token order.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{2.13}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2.14}$$

Transformers make use of the encoder-decoder framework. The language models used for biological sequences focus on making variations of the transformers' encoder to generate a feature vector of fixed length for supervised learning [126]. Each encoder, represented in Figure 2.16 contains two parts: a multi-head self-attention layer and a feed-forward layer.

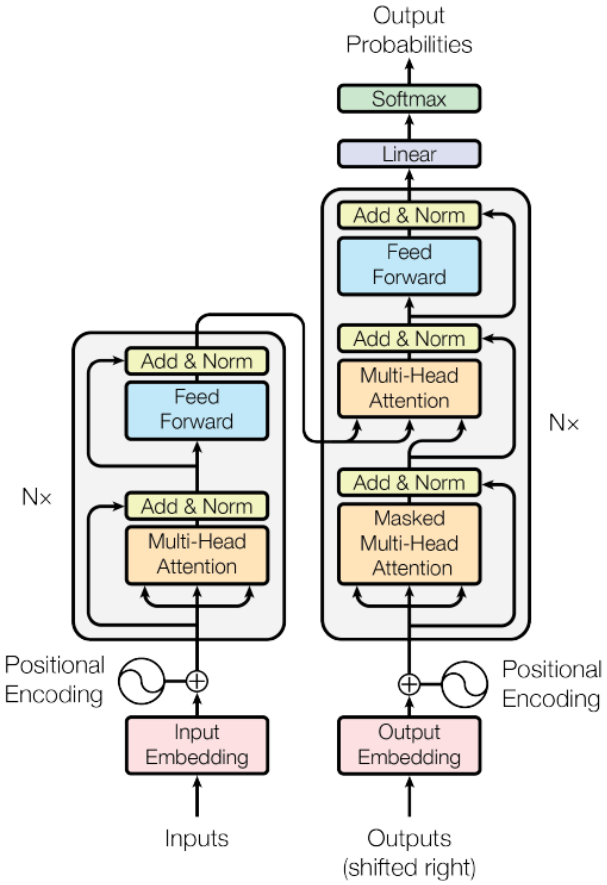


Figure 2.16: Transformer architecture. Adapted from [127].

A multi-head self-attention layer consists of a module that relies on three main definitions: Query Q, Key K, and Value V. The transformers' encoder computes the dot-product attention function of Q, K, and V matrices (equation 2.15). Accordingly, h *heads* are generated based upon the projection of these

three matrices (equation 2.16), resulting in multiple output vectors of d_v dimensions that are later concatenated (equation 2.17). This way, this multi-head approach and a self-attention mechanism can capture relationships between the tokens of a sequence.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.15}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.16}$$

$$MultiHead(Q, K, V) = concat(head_1, head_2, ..., head_h)W^O \tag{2.17}$$

Each token in the sequence is processed independently by the same position-wise feed-forward network. After each sub-layer, residual connections and layer normalisation are applied. Unlike batch normalisation, layer normalisation normalises across feature dimensions within each token representation, making it particularly suitable for sequence models.

Although most Transformer-based language models for biological sequences employ encoder-only architectures, the complete encoder–decoder Transformer remains highly effective for sequence-to-sequence tasks. A representative application is single-step retrosynthesis prediction, where the model receives the SMILES string of the reaction product as input and generates the corresponding SMILES strings of the reactants as output.

The decoder follows a structure similar to the encoder but includes an additional attention sub-layer. Each decoder block contains three components: (i) a masked multi-head self-attention layer, (ii) an encoder-decoder attention layer, and (iii) a position-wise feed-forward network. The masked self-attention mechanism prevents each position from attending to future tokens, ensuring autoregressive generation. The encoder-decoder attention layer uses the decoder representations as queries and the encoder output as keys and values, allowing the model to focus on relevant regions of the input sequence during prediction. This design makes the decoder particularly suitable for sequence generation tasks such as machine translation, text generation, and sequence-to-sequence learning.

2.4.7 Graph neural networks

As previously mentioned, molecules and proteins can be represented as graphs. Graph Neural Network (GNN)s can be trained with that information for prediction tasks [128]. Accordingly, the model herein explained is for a *graph-focused* application, which uses the entire graph to perform the prediction [128]. Normally, in graph representations, one represents a feature matrix X for nodes and an adjacency matrix $\hat{A}$ to represent the edges.

Graph Convolutional Network (GCN)s are used for protein and compound graphs. This architecture allows computing and updating the information of each node based on the aggregation of its neighbourhood nodes' information, as in equation 2.18. In this context, if one stacks two GCN layers, the second layer will aggregate information about the nodes' neighbours. As this information was updated in the previous

layer, considering the neighbour nodes, in the second layer, the feature matrix will be updated based on the surroundings of the neighbours (equation 2.19). This way, the GCN can learn how the interaction between nodes contributes to the output [129]. The first layer takes the input feature matrix X directly:

$$H^1 = \sigma(\hat{A}XW^0) \tag{2.18}$$

and each subsequent layer propagates the node representations as:

$$H^{l+1} = \sigma(\hat{A}H^lW^l) \tag{2.19}$$

where $\hat{A}$ is the normalised adjacency matrix, W^l is the trainable weight matrix of layer l, and H^l is the matrix of node representations at layer l, with $H^0 = X$ being the input feature matrix. Normalising $\hat{A}$ is common practice to keep the scale of the node feature vectors stable during propagation, which in turn helps avoid vanishing or exploding gradients during training.

The above formulation does not consider the information on atom bonds and amino acid relationships. For that, one should implement a relational GCN:

$$H^{l+1} = \sigma\left(\sum_{r=1}^{R} \hat{A}_r H^l W_r^l\right) \tag{2.20}$$

where R is the number of relations, $\hat{A}_r$ is the normalised adjacency matrix describing the edge connections for a given relation r, and W_r^l is a relation-specific weight matrix at layer l. This way, the model can learn how information from neighbouring nodes in a relational graph should be combined [129].

Graph Attention Network (GAT)s are another family of DL models that operate on graph-structured data, using attention mechanisms to weigh and aggregate the features of the nodes in the graph. In other words, the GAT model can selectively weigh and combine the features of different nodes in the graph rather than using a fixed weighting or aggregation function as the GCN does.

2.5 Retrobiosynthesis approaches

Retrobiosynthesis computational methods involve constructing a pathway by repeatedly applying chemical reaction rules or using sequence-to-sequence DL techniques on a target compound and its intermediates until an available precursor is identified [130]. One major issue with these approaches is the potential for a rapid increase in possible paths and intermediate compounds. This is due to the presence of multimolecular reactions, which involve multiple reactants and products. Additionally, it is possible to apply multiple reactions to the same substrate and to apply the same reaction to different parts of the substrate, which contributes to the combinatorial explosion of solutions. This is a pressing issue, and several strategies have been proposed to tackle the problem [130]:

- usage of reinforcement learning for multi-step search and planning, prioritising more promising solutions in the combinatorial space of solutions

- ignore “currency” metabolites / cofactors required by the reaction;
- divide multimolecular reactions into several reactions with only one substrate (e.g. in any reaction of type “A + B -> C + D”, the substrates A and B are never tested together);
- filter the number of rules by:
  - similarity;
  - reaction feasibility;
  - enzyme availability and scoring;
  - diameter around the reaction centre, i.e. atoms that change their configuration when the reaction is applied;
- considerably reduce the set of applicable reactions.
- usage of sequence-to-sequence DL models to predict the k most probable reactants, based on target products.

2.5.1 Reinforcement learning for retrobiosynthesis

Retrosynthetic planning aims to identify a feasible sequence of chemical transformations that can decompose a target molecule into simpler and ultimately purchasable building blocks [131]. In an analogous way, *retrobiosynthesis* in the context of pathway elucidation seeks to infer the sequence of enzymatic reactions that can biologically explain the formation of a plant specialised metabolite from simpler precursors originating in primary metabolism.

This can be interpreted as a recursive pathway reconstruction problem. Starting from a final natural product, such as an alkaloid, terpene, or flavonoid, the goal is to iteratively ask:

> *Which immediate precursor metabolite and enzyme-catalysed transformation could have generated this compound?*

Repeating this reasoning step-by-step gradually traces the molecule back toward well-known precursor pools, such as amino acids, terpenoid building blocks, shikimate derivatives, or central carbon intermediates.

Let the target metabolite be denoted by x_{target}. The goal is to infer a pathway:

$$\mathcal{P} = (x_{\text{target}} \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots \longrightarrow x_T)$$

that terminates in a set of biologically plausible precursor metabolites:

$$\mathcal{G} = \{x : x \in \text{primary metabolism or known precursor pool}\}.$$

This recursive dependency between pathway steps makes retrobiosynthesis naturally suitable for reinforcement learning and search-based formulations, because each enzymatic hypothesis directly constrains the future set of biologically reachable intermediates.

2.5.1.1 Biological markov decision process formulation

A useful way to formalise this problem is through a Markov Decision Process (MDP) [132, 133]. Although this terminology originates from reinforcement learning, each component has a direct biological interpretation.

The **state** (s_t) is defined as the set of metabolite structures whose upstream biosynthetic origin remains unresolved. At pathway step t, the state

$$s_t = \{x_1, x_2, \ldots, x_n\}$$

represents the current frontier of pathway uncertainty. For example, after disconnecting a flavonoid glycoside, the unresolved state may contain the aglycone scaffold and the sugar donor precursor that still need to be explained upstream.

The **action** (a_t) corresponds to selecting a biologically plausible enzymatic reaction that could transform one metabolite into one or more upstream precursors. Examples include cytochrome P450 oxidation, methyltransferase-mediated methylation, glycosyltransferase conjugation, or terpene cyclisation.

Mathematically, the probability of selecting a transformation is described by:

$$\pi(a \mid s),$$

where π represents the pathway policy. Biologically, $\pi(a \mid s)$ expresses:

> *Given the current unresolved metabolite set, how likely is a given enzyme family or reaction mechanism to explain the next precursor step?*

The **transition** corresponds to precursor generation. Applying an enzymatic hypothesis changes the state by replacing the selected metabolite with its predicted immediate precursors. For instance, a methylated flavonoid may be transformed into its demethylated precursor, while introducing *S*-adenosylmethionine-dependent methylation as the corresponding forward biosynthetic step. Thus, each action changes the future biochemical search space.

The **optimisation objective** is defined as cumulative biological plausibility. The objective is to minimise the average cumulative pathway cost:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} C(s_t, a_t, s_{t+1}) \right].$$

The expectation operator $\mathbb{E}_\pi$ represents the average over all plausible biosynthetic pathways that the model may generate. The cost term

$$C(s_t, a_t, s_{t+1})$$

can biologically encode pathway length, enzyme availability, thermodynamic implausibility, generative model cost/score or distance from primary metabolism.

2.5.1.2 Search strategy ladder: from local to global planning

It is useful to present the evolution of search strategies as a progression from local decision-making to globally informed planning.

The simplest strategy is greedy search, which always selects the reaction with the highest immediate confidence score or lowest cost [134]. It tends to excel in systems with straightforward pathways but struggles in complex networks [134]. It is highly dependent on the cost or confidence score and often gets stuck in local optima [134].

A natural improvement is beam search, which preserves the top-k most promising partial solutions at each expansion step [135]. Beam search is especially common in single-step retrosynthesis models based on sequence-to-sequence DL architectures [135–137], where multiple reactant hypotheses are decoded simultaneously. However, because beam width is fixed, promising branches may still be pruned too early if their initial score is low.

The next step in this progression is A* search [138], which extends local scoring by incorporating a heuristic estimate of future cost. Rather than ranking states only by immediate reaction confidence, A* prioritises nodes according to their total expected route cost. This makes it significantly more powerful for multi-step planning, but its success depends strongly on the reliability of the heuristic.

A* relies on a heuristic estimate of future cost, which in the neural-guided setting developed here is realised as a learned value function. In this context, it estimates:

How difficult is it to biologically explain the remaining unresolved metabolites?

This remaining pathway difficulty is represented by the heuristic:

$$f(s) = g(s) + h(s),$$

used in A* search [138].

Here:

- $g(s)$ = cumulative pathway cost accumulated so far,
- $h(s)$ = predicted remaining biological complexity.

Biologically, $g(s)$ corresponds to the total biosynthetic complexity already required to explain the current pathway branch. For example, if the unresolved metabolite is a common phenylpropanoid intermediate, $h(s)$ should be low. Conversely, if it is a rare rearranged alkaloid scaffold with unclear precursor logic, $h(s)$ should be high. Thus, the value function acts as a learned estimate of *remaining biosynthetic uncertainty*.

While A* provides globally informed best-first expansion, it remains sensitive to heuristic bias. This motivates the integration of Monte Carlo Tree Search (MCTS) [139], which explicitly allocates search effort toward uncertain but potentially valuable biosynthetic alternatives.

2.5.1.3 Exploration vs exploitation

The central reinforcement learning principle underlying this framework is the exploration–exploitation trade-off [133]. Exploitation refers to selecting the branch that currently appears best according to the learned heuristic, while exploration allocates computational effort to branches that are uncertain but may later yield better solutions.

In retrosynthesis, this trade-off is particularly acute because heuristic estimates are inherently imperfect. A planner that over-exploits its value function may repeatedly expand an apparently favourable branch that eventually leads to dead ends or unnecessarily long pathways. On the other hand, excessive exploration can be equally harmful. If too many uncertain branches are pursued indiscriminately, computational resources are wasted on unproductive alternatives, reducing the ability to search deeply into the most promising synthesis routes. This problem aligns well with the biological decision of either following the currently most plausible enzyme branch (exploitation) or exploring a less obvious but potentially lineage-specific pathway (exploration).

This trade-off [140] is formalised as:

$$a^* = \arg\max_a \left(Q(s,a) + U(s,a) \right).$$

Here, $Q(s,a)$ represents the accumulated evidence supporting an enzyme branch, while $U(s,a)$ corresponds to a novelty bonus for underexplored pathway hypotheses. This is particularly relevant in plant metabolism, where orphan enzymes and cryptic pathway branches are common.

2.5.1.4 Monte carlo tree search for pathway discovery

MCTS [139] provides a framework for exploring multiple competing biosynthetic hypotheses before committing to a pathway branch. Biologically, this is highly relevant for plant specialised metabolism, where several enzyme families [141, 142], lineage-specific reactions [141], or cryptic intermediates may plausibly explain the same target metabolite.

MCTS incrementally builds a pathway search tree by repeatedly exploring candidate biosynthetic trajectories through four conceptual stages: *selection*, *expansion*, *evaluation* (also called simulation), and *backup* (backpropagation).

The first stage, **selection**, determines which biosynthetic branch should be explored next. During this phase, the algorithm traverses the existing search tree according to a tree policy that balances pathway plausibility (exploitation) with uncertainty (exploration). This mechanism revisits metabolite branches that have previously yielded promising upstream routes while still probing less explored alternatives that may reveal hidden precursor logic.

This balance is commonly implemented using selection criteria derived from the Upper Confidence Bound (UCB) algorithm [143] and its extension Predictor + Upper Confidence Bound for Trees (PUCT) [144], which guide the tree search by rewarding both high estimated action value and uncertainty. UCB selects actions by balancing the estimated reward with an exploration bonus that decreases with the

square root of the action's visit count, while PUCT extends this by incorporating a prior probability $P(s, a)$ over actions, typically provided by a policy network. The next action a^* is then chosen as:

$$a^* = \arg\max_a \left(Q(s, a) + c_{\text{puct}} P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)} \right)$$

where $Q(s, a)$ is the estimated action value, $P(s, a)$ is the prior probability of action a in state s, $N(s, a)$ is the visit count of action a, $\sum_b N(s, b)$ is the total visit count over all actions in state s, and c_{puct} is a constant controlling the degree of exploration.

The second stage, **expansion**, adds newly encountered states to the search tree. From a biological perspective, this corresponds to applying a candidate enzymatic transformation to one unresolved metabolite and generating one or more upstream precursor hypotheses. For example, a methylated flavonoid branch may expand into its demethylated precursor, or a terpene scaffold may be disconnected into a prenyl donor and cyclisation precursor. Each newly generated precursor set becomes a new node in the pathway tree.

The third stage, **evaluation**, assigns a score to the newly expanded state. Biologically, this means estimating how likely the newly proposed precursor set is to be traceable back to primary metabolism. This can be done either through simulated rollouts or, more efficiently, through a learned value estimator that predicts the remaining biosynthetic complexity of the unresolved metabolite set.

The final stage, **backup**, propagates the evaluation signal backward through the visited branch, updating the statistics of all preceding nodes. Biologically, this means that if a precursor hypothesis repeatedly leads to chemically and biologically coherent upstream routes, confidence in earlier enzymatic decisions is progressively reinforced. This dynamic value revision is one of the strongest advantages of MCTS over static heuristic search, as the planner continuously refines its pathway confidence estimates during search.

2.5.1.5 AND-OR tree-based neural-guided search

While MCTS provides an exploration-aware framework for pathway planning, retrobiosynthesis can also be formulated more explicitly as a recursive decomposition problem using an AND-OR search tree [145, 146].

In this representation, **OR nodes** correspond to metabolite hypotheses because several distinct enzymatic transformations may plausibly explain a metabolite biosynthetic origin. For example, a hydroxylated alkaloid scaffold may be explained by cytochrome P450-mediated oxidation or lineage-specific tailoring enzymes. Only one of these alternatives is required to successfully explain the metabolite, making the OR abstraction biologically natural.

The optimal metabolite expansion is selected as:

$$m^* = \arg\min_m f(m),$$

where

$$f(m) = g(m) + h(m)$$

combines the accumulated pathway cost $g(m)$ with the neural estimate of remaining biosynthetic difficulty $h(m)$.

Conversely, **AND nodes** encode precursor co-dependencies. Once a reaction hypothesis is selected, it becomes an AND node because all precursor metabolites generated by that enzymatic step must themselves be traceable to upstream precursor pools. For example, a monoterpene indole alkaloid intermediate may be disconnected into tryptamine and secologanin. The pathway branch is only valid if both precursor branches are biologically feasible.

The recursive cost of a reaction node r is:

$$V(r) = c(r) + \sum_{m \in \text{children}(r)} V(m),$$

where $c(r)$ is the immediate enzymatic transformation cost and $V(m)$ the remaining biosynthetic cost of each precursor metabolite.

The key advantage of neural-guided AND–OR search is that the remaining biosynthetic complexity of each unresolved metabolite can be estimated using a learned value function:

$$h(m) = V_m(m),$$

where V_m is a neural model trained to predict the expected upstream pathway difficulty of metabolite m.

Biologically, this corresponds to estimating how likely a given intermediate is to be recursively connected to primary metabolism, known precursor pools, or canonical pathway modules. For example, common phenylpropanoid intermediates should have low heuristic cost, whereas rare rearranged alkaloid scaffolds should have high heuristic cost.

Finally, the total value of a metabolite node is determined by its best downstream reaction:

$$V(m) = \min_r V(r).$$

This captures the intuitive biological principle that the biosynthetic origin of a metabolite is explained by its most plausible enzymatic decomposition route. Thus, AND-OR neural-guided search explicitly separates alternative enzyme hypotheses (OR) from mandatory precursor co-resolution (AND).

2.5.2 Template-based approaches

Template-based approaches use automatically or manually constructed chemical rules to predict reactants from products. Several computational methods for retrobiosynthesis have been developed in the last two decades. Table 2.7 enumerates some of the most relevant and freely available resources for retrobiosynthesis using reaction patterns. PathPred, RetroPath2.0, and novoPathFinder generate rules automatically, while Ni et al. 2021 rely on manual curation. As for the latter, fewer reactions were generated compared to the automatic approaches, consequently reducing the combinatorial complexity of the problem [147].

PathPred [148] is a method to predict biodegradation pathways of environmental compounds and biosynthesis pathways of secondary metabolites based on the Kyoto Encyclopedia of Genes and Genomes

Table 2.7: Retrosynthesis methods based on reaction rules.

Tool	Number of rules	Rules source	Rule specificity	Combinatorial complexity
PathPred [148]	2,626	Automatic from KEGG RPAIR database	Fixed	Similarity
RetroPath2.0 [149]	19,000 (maximum)	Automatic from MetaNetX	Variable, controlled by diameter	Diameter, enzyme score and multimolecular reactions split
novoPathFinder [150]	4,996	Automatic from Rhea database	Fixed	Compounds similarity
Ni et al. 2021 [147]	1,224	Manual curation of MetaCyc reactions	Fixed	Reduction of the number of reactions

(KEGG) RPAIR database and compound subgraph matching. It starts by performing a maximal common subgraph global search in the KEGG database to find the most similar compounds compared with the query metabolite. Then, the RDM patterns associated with the compounds found are applied to the query compound. RDM patterns represent KEGG atom type changes at the reaction centre atom and its neighbouring atoms on the different regions [148].

RetroPath2.0 [149] attempted to overcome the combinatorial explosion by controlling the diameter over the reaction centre and splitting multimolecular reactions into single substrate reactions. Herein, the rules are computed by selecting the atoms around a sphere of a given radius around the reaction centre. This radius is directly associated with enzyme promiscuity and specificity: a smaller radius implies more promiscuity and less specificity to a given substrate, and vice versa. The authors compute enzyme penalty scores that take this into account, based on the logarithm of the number of similarity clusters of the sequences selected for a given rule. A penalty score of 0 implies high specificity, meaning that the enzymes associated with that rule belong to only one cluster of enzymes. When too many compounds are produced and a selection needs to be made, only the rules with the lowest penalties are applied [149].

Another feature to take into account is the consideration of Genome-Scale Metabolic Models (GSMM)s. novoPathFinder can take into account the GSMMs of chassis organisms, i.e. *Escherichia coli* or *Saccharomyces cerevisiae*, to deal with the combinatorial explosion problem. In this sense, when a multimolecular reaction is applied, novoPathFinder chooses the available heterologous metabolite in the model if it exists. Otherwise, it uses the reactant with the maximum common substructure. Moreover, as this method uses GSMMs of chassis organisms, it can consider overall stoichiometric conversions, growth-coupled theoretical yield, and thermodynamic feasibility of the predicted solutions and choose the most suitable one [150].

2.5.3 Template-free approaches

Chemical reaction predictions have recently been increasingly treated as NLP problems [135]. In particular, chemical reactions are represented as reaction SMILES, which represents *sentences*, and the forward and retrobiosynthesis problem is considered as a translation problem. Notably, the words in one language (products) are translated into the other language (reactants). The significant advantage of these methods is the capability of generating reactions that are not in the training set nor in the spectrum of known transformations [151].

As MolecularTransformer [152] achieved state-of-the-art accuracy in forward synthesis prediction at the time of its release, other approaches for retrosynthesis based on the same architecture [135] (Transformers) became available. In 2019, Zheng and collaborators [137] proposed an approach that consisted of two Transformer networks. The first is a fully-trained retrosynthetic reaction predictor to infer the candidate reactants of a given target. The second allowed the correction of syntax errors when generating the SMILES sequence, mitigating the large number of incorrect SMILES generated. This approach obtained 43.7% top-1 accuracy on the USPTO-50K benchmark (without the reaction class provided), reducing the percentage of invalid candidate precursors from 12.1% to 0.7% using the syntax corrector transformer. They relied on beam search to find optimal solutions for pathways.

Another approach [153] combined heuristic MCTS, proposed by [154], with a Transformer model. Notably, they reported a top-1 accuracy above 63.0% on the same USPTO-50K dataset, with SMILES validity rates of 95% (top-1) and 99.6% (overall) and without a syntax corrector.

Schwaller and collaborators [135], on the other hand, used an extension of the MolecularTransformer architecture combined with a hyper-graph to represent the retrosynthesis tree. Accordingly, they use a single-step retrosynthetic model to generate a certain number of possible disconnections (i.e. precursor set). Furthermore, they compute the likelihood of the applied transformation with the forward synthesis transformations (predict the products, given the substrates) generated by the MolecularTransformer. It also applies an SCScore-based Bayesian-like probability described in [155] to guide the tree expansion. The optimal synthetic pathway is found through a beam search on the hyper-graph. Unlike the preceding tools, it allows for concurrently predicting the reagents, catalysts and reactants.

Tetko and collaborators [136] showed that data augmentation eliminated the effect of data memorisation by neural networks and could improve performance on retrosynthesis tasks. Notably, they applied data augmentation both on the SMILES of the input and on the target data combined with a beam search algorithm.

A recent effort adopted a different approach inspired by chemists' approach to retrosynthesis [151]. RetroPrime starts by (i) disconnecting a molecule at a reaction centre and (ii) converting the synthons (a building block for the synthesis) into reactions. Thus, this tool relies on two Transformers: one to predict the reaction centre, which serves as a framework to extract the synthons; and another to predict the reactants from the products. This tool also adopts the data augmentation strategy to train the two Transformers. Finally, this tool also uses a beam search algorithm to search across the retrosynthesis tree.

Recently, BioNavi-NP [156] was among the earliest tools to extend template-free retrosynthetic planning to the biosynthetic domain of natural products. BioNavi-NP first trains an end-to-end Transformer model to perform single-step retrobiosynthesis prediction, generating plausible precursor candidates for a target natural product by learning from both curated biosynthetic reactions and natural product-like organic reactions. To improve robustness and generalisation, this tool adopts data augmentation through reaction dataset expansion and further combines multiple models via ensemble learning. Based on these single-step predictions, BioNavi-NP then performs iterative multi-step pathway planning using the Retro* framework, an AND-OR tree-based neural-guided search algorithm that efficiently explores biosynthetic routes toward biologically plausible building blocks. Finally, the predicted pathways are ranked according to the accumulated reaction costs, enabling the identification of both known and hypothetical biosynthetic routes.

Later on, Kim *et al.* introduced a retrieval-augmented strategy to improve bio-retrosynthetic pathway prediction for complex natural products [157]. READRetro starts by combining two dual-view single-step retrosynthesis models, Retroformer and Graph2SMILES, which leverage both SMILES and graph representations to capture complementary biochemical features. This tool further incorporates a reaction retriever module to directly recover conserved metabolic reactions from the training database, improving the memorability of known biosynthetic transformations. Based on these single-step predictions, READRetro then performs multi-step pathway planning using the Retro* algorithm, which iteratively expands an AND-OR search tree toward predefined metabolic building blocks. Finally, this tool also integrates a pathway retriever based on KEGG maps, enabling early termination of search branches when intermediates match curated biosynthetic pathways and improving the reconstruction of long and biochemically plausible routes.

A recent effort proposed a hybrid search strategy that integrates the exploratory behaviour of Monte Carlo Tree Search with the efficiency of heuristic best-first search [158]. MEEA* starts by (i) performing a set of MCTS simulations without node expansion to collect an exploratory candidate set and (ii) selecting the state with the lowest heuristic cost for expansion under an A*-like criterion. Thus, this tool combines look-ahead exploration with cost-guided exploitation, which the authors report preserves the optimality guarantees of A* under an admissible heuristic, while mitigating its tendency to become trapped in non-optimal branches. To further improve robustness, this tool also adopts path consistency as a regularisation strategy to train a more reliable cost estimator with improved generalisation performance. Finally, this tool can be directly extended to natural product bio-retrosynthesis by integrating the single-step expansion policy of BioNavi-NP, significantly improving pathway recovery for complex biosynthetic targets.

3

DeepMol: an Automated Machine and Deep Learning Framework for Computational Chemistry

The domain of computational chemistry has experienced a significant evolution due to the introduction of Machine Learning (ML) technologies. Despite its potential to revolutionise the field, researchers are often encumbered by obstacles, such as the complexity of selecting optimal algorithms, the automation of data pre-processing steps, the necessity for adaptive feature engineering, and the assurance of model performance consistency across different datasets. Addressing these issues head-on, DeepMol is an Automated ML (AutoML) tool that simplifies these tasks by automating critical steps of the ML pipeline. DeepMol rapidly and automatically identifies the most effective data representation, pre-processing methods and model configurations for a specific molecular property/activity prediction problem. As one of the first AutoML tools specifically developed for the computational chemistry domain, DeepMol stands out with its open-source code, in-depth tutorials, detailed documentation, and examples of real-world applications, all available at `https://github.com/BioSystemsUM/DeepMol` and `https://deepmol.readthedocs.io/en/latest/`. By introducing AutoML as a groundbreaking feature in computational chemistry, DeepMol establishes itself as the pioneering state-of-the-art tool in the field.

Keywords: AutoML · cheminformatics · QSAR · deep learning

This section holds first co-authorship with PhD João Correia and was published in the following peer-reviewed paper: *Correia, J., Capela, J., & Rocha, M. (2024). Deepmol: an automated machine and deep learning framework for computational chemistry. Journal of Cheminformatics, 16(1), 136.*

3.1 Introduction

In recent years, computational chemistry has undergone a remarkable transformation, driven by advances in ML and DL techniques and by the immense growth of available chemical data [159–162]. These have facilitated exploring and understanding the complex relationships between chemical structures and their properties [163]. As a result, using these computational methods within the chemical discovery pipeline

has emerged as a powerful and promising way to expedite the discovery of new chemicals with improved properties [164].

Quantitative structure-activity/property relationship (QSAR/QSPR) models have always been a focal topic in computational chemistry [165–168]. Initially, simple statistical models were applied to small datasets of molecules characterised by a restricted array of descriptors [169], offering a straightforward way to correlate molecular structure with biological activity [170]. Nowadays, QSAR/QSPR modelling consists of large sets of molecules with an extensive set of molecular descriptors.

As the amount of available data and the complexity of QSAR/QSPR models continue to grow, the demand for advanced techniques has led to the emergence of DL as a viable alternative to traditional ML models [171]. With the current plethora of ML/DL models and diverse chemical descriptors available, researchers face the challenge of selecting the most suitable alternative for their data regarding data representation and processing and ML/DL models [170]. Choosing the optimal combination of features/descriptors and models requires exhaustive testing to comprehensively understand their performance on a given dataset [172, 173]. Recognising this challenge, the importance of AutoML frameworks becomes unequivocal. In the fields of QSAR/QSPR modelling, where pre-processing steps, models, and their combination can significantly impact results, the need for a comprehensive and easily customisable AutoML framework is even more evident.

However, despite this need, only a few resources offer an easily customisable framework capable of providing a wide range of possibilities. ZairaChem [20] provides the first AutoML framework described in the literature that can automatically optimise all the pre-processing steps and model hyperparameters for specific QSAR problems. However, users cannot easily customise this system to suit their needs. For example, ZairaChem does not provide ways of creating custom objective functions, i.e. generating the final metric to give feedback to the AutoML system, whether in a cross- or hold-out validation or other scenarios. Another example is the impossibility of defining the type of models and descriptors to test. Furthermore, ZairaChem is restricted to binary classification tasks, which can be very limiting. Another AutoML tool is QSARTuna [21], which performs the AutoML search using Optuna [174]. However, it offers a limited range of feature extraction methods and ML/DL models. Additionally, like ZairaChem, QSARTuna is unable to create custom objective functions.

Herein, we present DeepMol, an AutoML Python-based open-source framework for the prediction of activities and properties of chemical molecules. Although fully automated, DeepMol is built modularly, allowing for the customisation of every step of the ML pipeline, starting from the data loading and processing to the model prediction and explainability. What sets DeepMol apart is its robust AutoML functionality, allowing the automatic optimisation of pre-processing methods, data engineering techniques, and ML/DL models and respective hyperparameters. Even for users with minimal coding experience, with just a few lines of code, DeepMol allows testing thousands of configurations to determine the most effective ones for their specific datasets. Notably, DeepMol provides documented ways of defining objective functions and customising AutoML experiments. Moreover, it is suitable for several ML tasks such as binary, multi-class, multi-label/multi-task classification and regression. The framework leverages other well-known and established packages, like *RDKit* [175] for molecular operations, *Scikit-Learn* [176], *Tensorflow* [177] and

DeepChem [178] for model building and *Optuna* [174] for end-to-end ML pipeline optimisation.

3.2 Implementation

DeepMol was primarily developed as a comprehensive end-to-end AutoML framework for computational chemistry. Yet, its modular design permits the independent utilisation of its components. The framework provides a wide range of techniques for each step of a general ML pipeline, from data pre-processing and feature extraction to model training, evaluation, and explainability. Furthermore, the package has been extended with additional functionalities of significant relevance, including unsupervised learning models, data-splitting strategies specifically designed for molecular datasets, and approaches to address data imbalance.

The DeepMol AutoML engine, as illustrated in Figure 3.1, comprehensively explores a vast array of potential methodological combinations. In this context, state-of-the-art computational chemistry methods are employed automatically and sequentially in a pipeline, following a specific configuration, ensuring it is tailored to address specific chemical tasks. The configuration space encompasses several data standardisation methods (Figure 3.1a: three methods), feature extraction methods (Figure 3.1b: four options for sets with 34 methods in total, all with their respective parameters), scaling and selection methods (Figure 3.1c: 14 methods and respective parameters), various ML models and ensembles (Figure 3.1d: 140 models), and their respective hyperparameters.

Upon starting the AutoML experiment, the engine first processes the training data following a predefined sequence of steps, known as the pipeline configuration, and then uses this data to train an ML/DL model. Post-training, a separate set of data, the validation set, is processed and assessed to evaluate the model's performance. The outcomes of this evaluation are then fed back into the optimisation framework, guiding it in choosing new parameters and methods for improvement. This cycle of training and evaluating is repeated a user-specified number of times, referred to as *trials*. After completing all the trials, the system analyses the results to identify and select the most effective pipeline. The optimisation framework is efficiently powered by the *Optuna* search engine, which provides nine different optimisation algorithms compatible with DeepMol. Once the optimal pipeline has been identified, it can be applied to analyse new data, enabling predictions on untested data. Additionally, this pipeline facilitates the virtual screening of extensive databases, efficiently identifying potential molecules of interest.

This section provides an overview of the key components and capabilities of DeepMol, highlighting its significant contributions and advantages in chemical compound analysis.

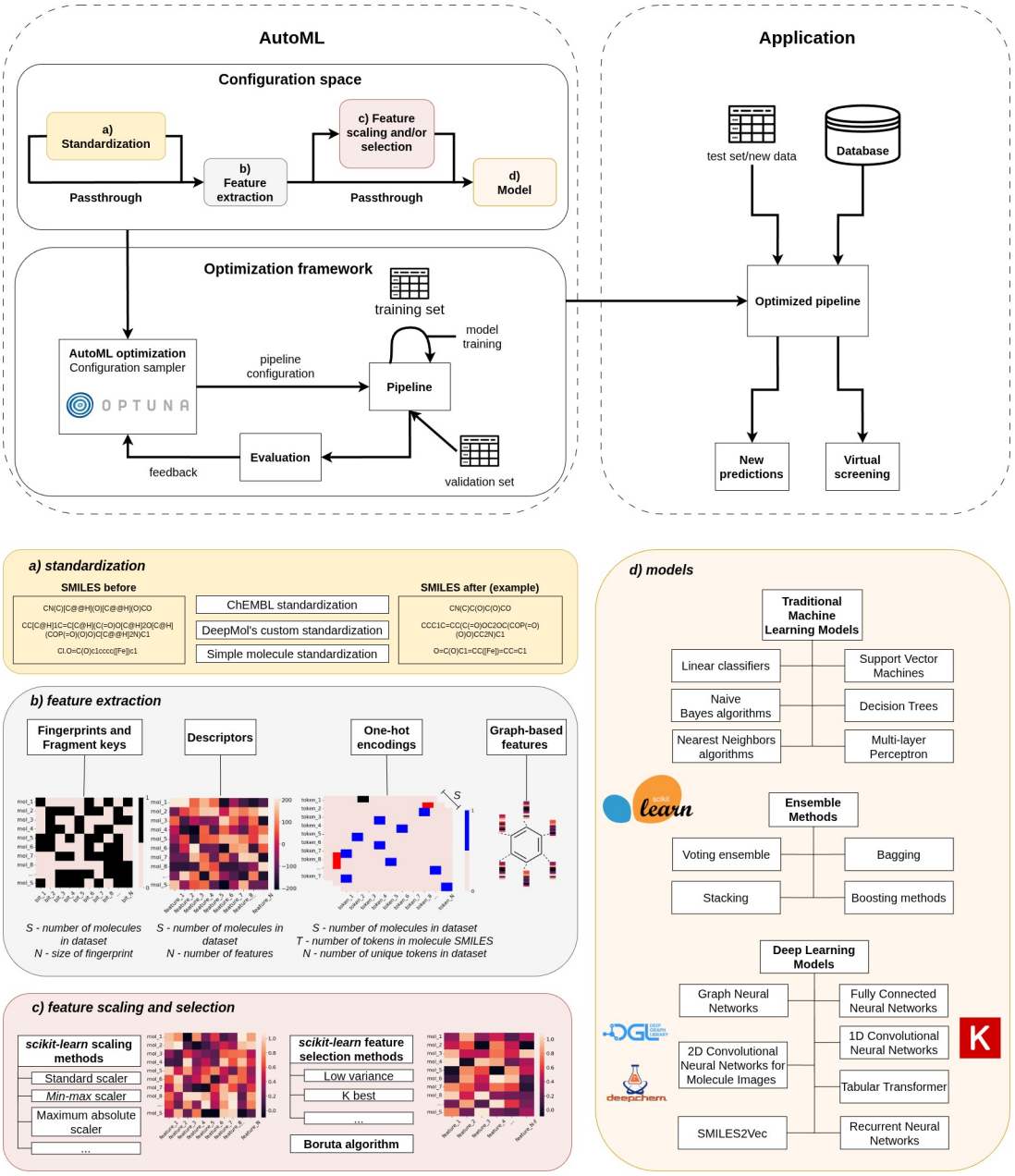


Figure 3.1: The DeepMol AutoML framework offers a comprehensive approach that includes an optimisation framework that samples pipeline configurations from the configuration space. This pipeline begins with an optional **(a)** standardisation step (configuration space: 3 different methods), followed by **(b)** the extraction of features (configuration space: four options for sets with 34 methods in total, all with their respective parameters). These features can then be optionally **(c)** scaled and selected (configuration space: 14 methods and respective parameters), preparing them for **(d)** the training phase of the model (configuration space: 140 models, architectures and hyperparameters). Post-training, the model's performance is evaluated based on a predetermined metric. This evaluation feedbacks to the *Optuna* optimisation engine. *Optuna* uses these results to inform its selection of new parameters and methods for enhancing the pipeline's efficiency using state-of-the-art optimisation algorithms. This process is repeated *n* times (chosen by the user), also called trials, and, in the end, the best pipeline is selected. Each individual box represented in the configuration space is expanded and schematised in the boxes below. Finally, the result of the AutoML engine is an optimised pipeline that can be used to transform new data and perform new predictions.

3.2.1 Data loading

Molecular structures can be represented using various formats. One such format is the Simplified Molecular Input Line Entry System (SMILES), which is readable and concise, as short as single-line ASCII strings [46]. The other format is a connection table, which can be stored as a Structure Data File (SDF) and used to represent three-dimensional structures of molecules [38, 179]. These are the *de facto* standard starting points for representing molecules in ML tasks [46, 180] and DeepMol provides loaders for both formats. Once loaded, the information is converted into a structured data format that allows easy access to relevant details, such as the SMILES representation of all molecules, their identifiers, and known outputs associated with the input data (labels). Alternatively, if one prefers to load molecules from an already-loaded data frame, a *SmilesDataset* class can be created. This dataset accepts SMILES or *RDKit Mol* objects as input, along with their corresponding identifiers and labels.

3.2.2 Molecular standardisation

The availability of vast amounts of chemical data found in molecular databases containing hundreds of millions of compounds can be a double-edged sword. While it pushes the field forward, it makes human curation infeasible, resulting in the frequent occurrence of incorrect and inconsistent molecular structures [181]. Even minor structural errors and inconsistencies within a dataset can result in significant losses in the predictive ability of QSAR/QSPR models [182]. Consequently, the development of strategies that can tackle this problem has received increased attention in recent years [183–185].

With DeepMol, it is possible to standardise molecules using three different options:

- *BasicStandardizer* performs basic sanitization using *RDKit* to ensure that a molecular structure is represented consistently and validly according to a set of predefined rules. These include kekulization, check of valencies, set aromaticity, conjugation, and hybridization.

- *CustomStandardizer* performs a set of steps defined by the user, using *RDKit*. Some steps include molecular sanitization (same as in the BasicStandardizer), removal of isotope and/or stereochemistry information, neutralising charges, fragment removal, and kekulization.

- *ChEMBLStandardizer* performs a set of rules as used by the ChEMBL database [183]. It consists of three components: a *Checker* that tests the validity of the chemical structures, a Standardizer that formats compounds based on the U.S. Food and Drug Administration (FDA) and International Union of Pure and Applied Chemistry (IUPAC) guidelines; and a *GetParent* component that removes any salts and solvents from the compound.

While DeepMol provides these widely applicable standardisation techniques, we recognise there are additional layers of standardisation that can significantly enhance data consistency, depending on the dataset and modelling needs. For example, more specialised transformations such as canonicalising tautomers and specific functional groups can further reduce redundancies and improve model performance.

DeepMol's design is extensible, enabling users to implement these or other tailored standardisation processes to meet specific requirements.

3.2.3 Feature extraction

In ML, feature extraction from molecules is a common task. Different types of molecular features are categorised from 0-dimensional (D) to 4D. 0D features provide overall information about the molecule (e.g. atom and bond counts), while 1D features describe substructures within the molecule (e.g. fingerprints) [186, 187]. 2D features capture molecular topology, and 3D features capture geometric information of the molecule's three-dimensional structure [188]. 4D features are descriptors with an additional dimension representing interactions with receptors' active sites or multiple conformational states.

DeepMol provides a set of 0D, a few 1D and 2D descriptors in only one class. The descriptors contained in this class are enumerated and described in Supplementary Table S1. DeepMol offers a wide range of 1D features provided by *RDKit*, including circular, atom pair, layered and *RDKit* fingerprints. DeepMol also includes Molecular ACCess System (MACCS) keys implemented by *RDKit*, which encode the presence or absence of specific molecular fragments or substructures in a molecule. These are further enumerated and described in Supplementary Table S2.

3D molecular conformations must be generated or loaded before generating 3D descriptors. For this matter, DeepMol provides conformer generation methods like Experimental-Torsion basic Knowledge Distance Geometry (ETKDG) and methods like Merck Molecular Force Field (MMFF) and Universal Force Field (UFF) for optimising the generated conformers. Once generated, DeepMol offers methods to extract features from these conformers. These methods include AutoCorr3D, Radial Distribution Function (RDF), the plane of best fit, MORSE, WHIM descriptors, Radius of Gyration, shape descriptors, and principal moments of inertia. More details on these methods can be found in Supplementary Table S3.

DeepMol also provides one-hot encoding schemes. Herein, DeepMol provides an implementation of an atom-level tokenizer and *k*-mer tokenizer from Li et al. [189]. The former method permits treating each character in the SMILES string linked to an atom (e.g., [C@] or [N@+]) as an identical token, whereas the latter provides groups of *k* characters of the SMILES string. Both create a vocabulary dynamically given a dataset. If no tokenizer is passed as input, the one-hot encoding uses the atom-level tokenizer.

DeepMol extends *DeepChem* to provide inputs for graph neural networks (GNN), including molecular graphs (2D descriptors), representing the molecule by a list of neighbours and a set of initial feature vectors for each atom. The feature vectors represent the atom's local chemical environment, including atom types, charge, and hybridization, among others [190]. Other implementations for using GNNs were also included, such as methods for implementing Duvenaud graph convolutions [191], Weave convolutions [190], and MolGAN [192]. Coulomb matrices and their eigenvalues [193] are also provided as methods of extracting 3D features, providing information on electrostatic interaction between atoms.

Finally, DeepMol also extends *DeepChem* to convert molecules into images and encodings to be passed to an embedding layer. All DeepChem-derived feature extractors are further detailed in Supplementary Table S4.

3.2.4 Data scaling

Many feature extraction methods encode molecules as bit vectors, while others use vectors of real numbers. In the latter, a challenge arises when features have different numerical ranges, which may impact certain algorithms. For instance, scaling is particularly important in models sensitive to feature scales, such as SVMs and neural networks, where larger ranges can cause certain features to dominate the learning process. While, for instance, tree-based models are generally less affected by feature magnitudes. To support these requirements, DeepMol offers a wide range of data scalers, as provided in the *Scikit-Learn* package (Supplementary Table S5).

3.2.5 Feature selection

Feature selection is a common task in ML problems. By choosing the most relevant features, the performance of an ML model can be significantly enhanced. Eliminating irrelevant or redundant features helps prevent overfitting and improve the generalisation power of the model, while reducing the computational burden. DeepMol supports many types of feature selection as provided by *Scikit-Learn*, including removing features with low variability, supervised filters based on statistical tests, as well as wrappers such as Recursive Feature Elimination, or selecting features based on importance weights (Supplementary Table S6). In addition, DeepMol also provides feature selection based on the Boruta algorithm [194].

3.2.6 Machine and deep learning models

DeepMol offers compatibility with three popular ML/DL frameworks: *Tensorflow*, *Scikit-Learn*, and *Deep Chem*. This allows seamless integration of any model built upon these frameworks into DeepMol. A wide range of pre-built models from these frameworks is readily available, supporting single and multi-task problems, binary and multi-class classification and regression, making it convenient for users to utilise these models for specific tasks.

Through *Scikit-Learn*, DeepMol offers an extensive selection of popular ML models, including, among others, linear and logistic regression, support vector machines, decision trees, random forests, and gradient boosting. In conjunction with *DeepChem*, it also provides several DL models specifically tailored for chemical data, including graph neural networks (GNNs), recurrent neural networks (RNNs), and transformer models. Moreover, the flexibility offered by *Tensorflow* enables integration of any DL architecture into DeepMol's pipeline. By consolidating these capabilities, DeepMol serves as a comprehensive and versatile framework, facilitating implementation and comparison of various ML and DL models in one unified platform [195, 196].

3.2.7 Hyperparameter optimisation

Although DeepMol's primary focus is on AutoML, where advanced optimisation techniques can be used to optimise each step and corresponding hyperparameters of a complete ML pipeline, it also provides users with the flexibility to perform standard hyperparameter tuning on individual models. Fine-tuning model

hyperparameters is essential for achieving optimal results, reducing overfitting, and improving overall efficiency. For this purpose, DeepMol includes straightforward options like *randomised* and *grid search*, enabling users to directly control specific tuning tasks if they choose not to use the full AutoML pipeline.

3.2.8 Pipelines

In DeepMol, we have introduced a pipeline class that simplifies the creation of an ML pipeline. This class empowers users to construct an ML pipeline tailored to their specific needs. It offers the flexibility to incorporate a diverse range of steps, enabling the inclusion of any combination of the following methods:

- Data standardisation;
- Feature extraction or transformation (data scaling and selection);
- Model training and hyperparameter tuning.

Pipelines in DeepMol work the same as *Scikit-Learn* pipelines: use the *fit_transform* method to fit and transform the train data and the *transform* method for solely transforming the validation and test sets. The pipeline class also allows users to save and load the pipeline again. Moreover, it allows to evaluate the pipeline and make predictions on new data using the *evaluate* and *predict* methods, respectively. The model herein declared can also be optimised.

Users can design and customise their ML pipeline, adapting it to their unique requirements and desired outcomes. This approach significantly reduces the complexity of constructing an ML pipeline, promoting efficient experimentation and development.

3.2.9 AutoML: pipeline optimisation

Pipelines provide a convenient and efficient way to transform raw data into trained and validated models, but determining the appropriate sequence of steps often requires expertise. Moreover, if one wishes to explore multiple combinations of pre-processing techniques (data *Transformers*) and models (*Predictors*), manually constructing a *Pipeline* for each combination can be repetitive and time-consuming.

Automating the process of building and optimising each step and respective parameters of a *Pipeline* can greatly assist researchers with limited ML expertise in building and deploying effective QSAR/QSPR systems, accelerating the development process. In DeepMol, we offer the *PipelineOptimisation* module (AutoML), which enables the search for the best configurations from a large number of possibilities. This module leverages the powerful *Optuna* library [174] and its state-of-the-art optimisation algorithms.

As with any optimisation problem, pipeline optimisation needs an objective function. By default, DeepMol provides the objective of maximising or minimising a metric (e.g. accuracy) for a given validation set. However, other custom objective functions can be added.

Furthermore, a function can be created with the configuration space that will be optimised by *Optuna*. This function will receive a *Trial* object from *Optuna* and will return a set of steps (i.e. *Transformers* and

Predictors) from the configuration space to define the trial pipeline. Alternatively, DeepMol provides a set of predefined configuration spaces (pre-sets) for convenience. The available pre-sets include:

- 'sklearn': optimises between all available Standardizers, Featurizers (and combinations of those) , Scalers, Feature Selectors in DeepMol, and all available models in *Scikit-Learn*.
- 'deepchem': optimises between all available Standardizers in DeepMol and all available models in *DeepChem* (depending on the model, different Featurizers, Scalers and Feature Selectors may also be optimised).
- 'keras': optimises between Standardizers, Featurizers (and combinations of those), Scalers, Feature Selectors in DeepMol, and some core Keras models (Fully Connected Neural Networks (FCNNs), 1D Convolutional Neural Networks (CNNs), RNNs, Bidirectional RNNs, and transformer models).
- 'all': optimises between all the above.

Finally, a *Voting Pipeline* composed of the best five pipelines is created after running the AutoML, where the voting process for classification occurs through soft and hard voting mechanisms. In soft voting, the predicted probabilities from each classifier are averaged to determine the final prediction. On the other hand, hard voting involves a simple majority vote, where the class predicted by the majority of the individual classifiers is selected. Additionally, for regression tasks, the final prediction is obtained by averaging the predictions from each regressor, providing a robust ensemble prediction from the diverse models in the voting pipeline.

3.2.10 Unsupervised learning

Unsupervised learning has become increasingly important in exploring the vast amounts of available and sometimes unlabelled chemical data. Through techniques like clustering and dimensionality reduction, unsupervised learning allows the identification of patterns and relationships in large datasets without prior knowledge.

DeepMol provides a simple interface to various techniques, such as Principal Component Analysis (PCA), t-distributed Stochastic Neighbour Embedding (t-SNE), K-means clustering, and Uniform Manifold Approximation and Projection (UMAP). These techniques can help scientists to efficiently analyse and visualise complex molecular data, providing valuable insights to guide subsequent analyses.

3.2.11 Data splitting

Splitting the data into training, validation and test datasets or in a cross-validation setting is a crucial step in ML pipelines. It is common for practitioners to require data separation based on molecular structures or similarities. Practitioners may opt for a homogeneous data split to maintain a balanced distribution of molecular structures or similarities across the sets. This approach aims to capture the diversity of the dataset, while providing reliable performance estimates for the ML model. Alternatively, practitioners

may emphasise the separation of highly similar molecules from less similar ones (heterogeneous splits). By isolating similar molecules, practitioners can better assess the model's ability to capture nuanced differences and generalise to unseen data.

DeepMol provides molecular splitters that split the inputted dataset based on similarity, scaffolds or Butina clusters [197], while maintaining the stratification of classes. Figure 3.2 illustrates how the similarity and scaffold splits are distributed in the overall chemical space and how the homogeneity parameters control the split.

Moreover, DeepMol provides random and stratified splitters for single and multitask classification problems. More information about each splitter is provided in Supplementary Table S7.

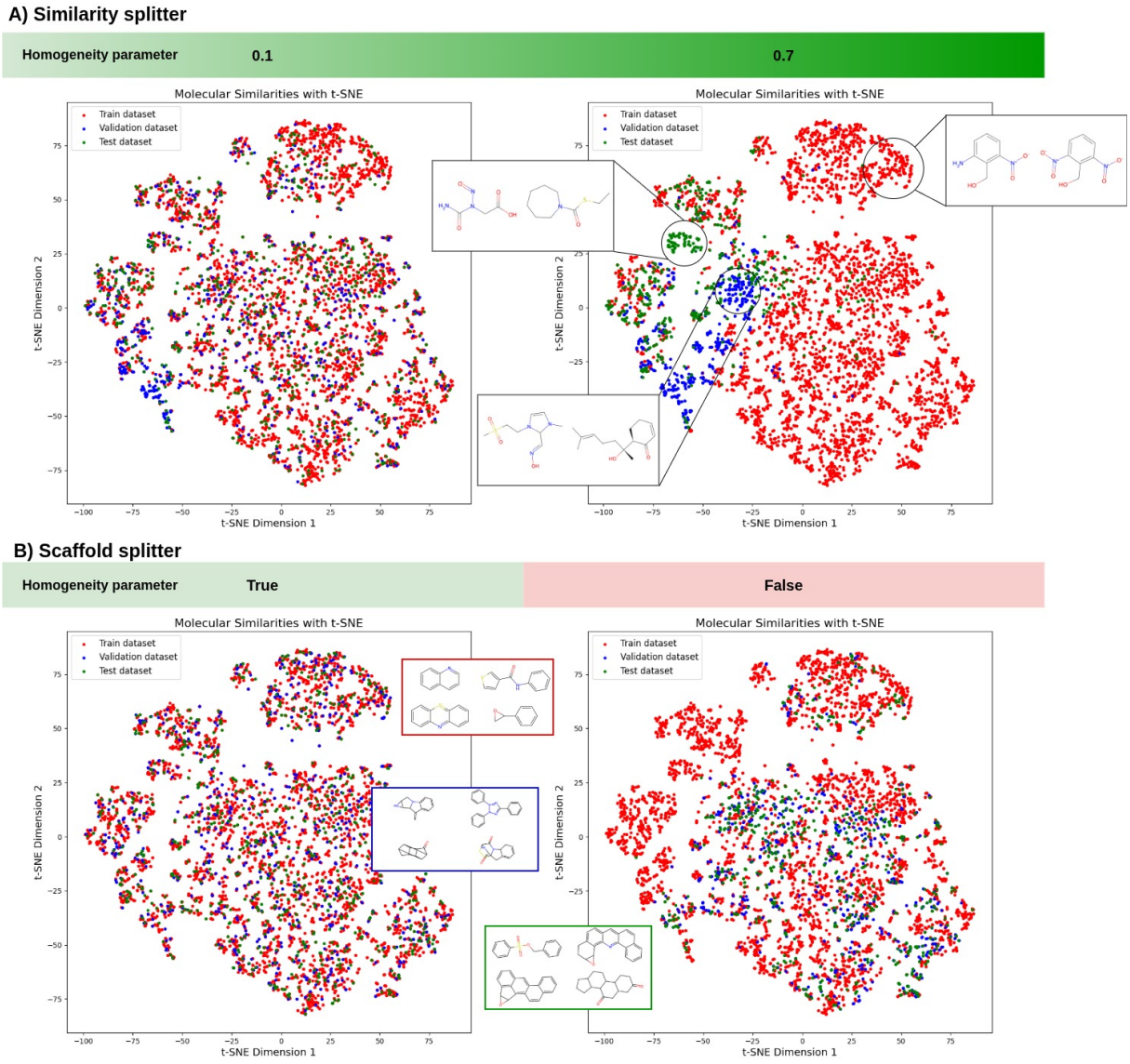


Figure 3.2: Depiction of dataset splits in the chemical space for the mutagenicity dataset from [198]. A similarity matrix was generated using the Tanimoto similarity metric over Morgan fingerprints. A t-SNE was applied to the matrix to reduce dimensionality and visualisation. **A)** represents the similarity splitter and examples of molecules belonging to small clusters. One can regulate the homogeneity of the splits by tweaking the homogeneity parameter, which considers all the compounds with a similarity lower than the homogeneous parameter to be in the same set. The higher the parameter is, the more heterogeneous the split will be. **B)** represents the scaffold splitter and scaffolds belonging to the splits on the plot on the right. This splitter can separate the data by putting compounds with the same scaffolds in different data splits (homogeneous split) or not (heterogeneous split).

3.2.12 Imbalanced data

Imbalanced data is a common problem in many real-world applications that heavily affect the quality and reliability of ML and DL approaches [199]. Likewise, imbalanced data is an issue in QSAR studies, where the difference between the number of active and inactive molecules can be extreme [200]. This difference

generally leads to biased and sub-optimal models, as traditional ML algorithms may not be able to learn the minority class effectively.

Data balancing techniques have shown potential in mitigating the effect of imbalanced data on the model performance [200]. DeepMol provides various methods for handling imbalanced data, including over-sampling, under-sampling, and combination methods, as in the *imbalanced-learn* package [201]. For over-sampling, it provides a random over-sampler and Synthetic Minority Over-sampling Technique (SMOTE) [202], while Random under-sampler and *ClusterCentroids* can be used for under-sampling. Finally, for simultaneous over and under-sampling, it provides SMOTE with Edited Nearest Neighbours (SMOTEENN) [203] and SMOTE using Tomek links (SMOTETomek) [204]. More information about each imbalanced learning technique is provided in Supplementary Table S8.

While techniques like SMOTE have shown efficacy in many fields, including in QSAR studies, we recognise its limitations, specifically for molecular data. SMOTE operates by generating synthetic samples through interpolation in the feature space, which can be problematic for molecular data where structural validity and biological relevance are crucial. For instance, stereoisomers, molecules that have the same connectivity but differ in spatial orientation, can pose significant risks when oversampling. Even though stereoisomers may share similar molecular fingerprints or descriptor-based features, their distinct three-dimensional arrangements often lead to different biological activities and interactions. Using SMOTE in such cases could introduce synthetic samples that fail to respect the nuanced differences between stereoisomers, inadvertently leading to models that overlook critical stereochemical distinctions. This could result in predictions that generalise poorly and even lead to unsafe recommendations in applications like drug discovery or toxicity prediction.

Given these considerations, including the potential issues with synthetic sample generation in molecular datasets, we opted not to integrate imbalanced learning techniques directly into the AutoML pipeline in DeepMol. Instead, we leave the decision to use such techniques to the user, allowing for more control and customisation of their model-building process.

3.2.13 Feature importance and model interpretability

Model interpretability, an essential aspect in the field of ML, refers to the capacity to comprehend and elucidate the internal mechanisms of an ML model. This ability provides insights into the rationale behind the model's predictions or decisions, fostering a sense of trust in the model's outputs.

SHAP (SHapley Additive exPlanations) [205] stands out as a widely utilised technique for achieving model explainability. It leverages the Shapley value, a fundamental concept derived from cooperative game theory, to assign significance to each input feature within a model. By comparing the predictions of various feature subsets, both with and without a specific feature, the Shapley value quantifies the marginal contribution of that feature to a prediction.

Visualising SHAP values can take various forms, including bar charts, scatter plots, and summary plots. These visual representations facilitate the interpretation of how individual features influence the model's predictions, aiding in identifying the most crucial features for a given prediction scenario.

Along with these methods, DeepMol allows one to visualise the structures associated with bits in Morgan and *RDKit* fingerprints and MACCS keys. Therefore, after computing the SHAP values, one can directly cross information from the most relevant features to the molecule and draw biological and chemical conclusions, as depicted in Figure 3.3.

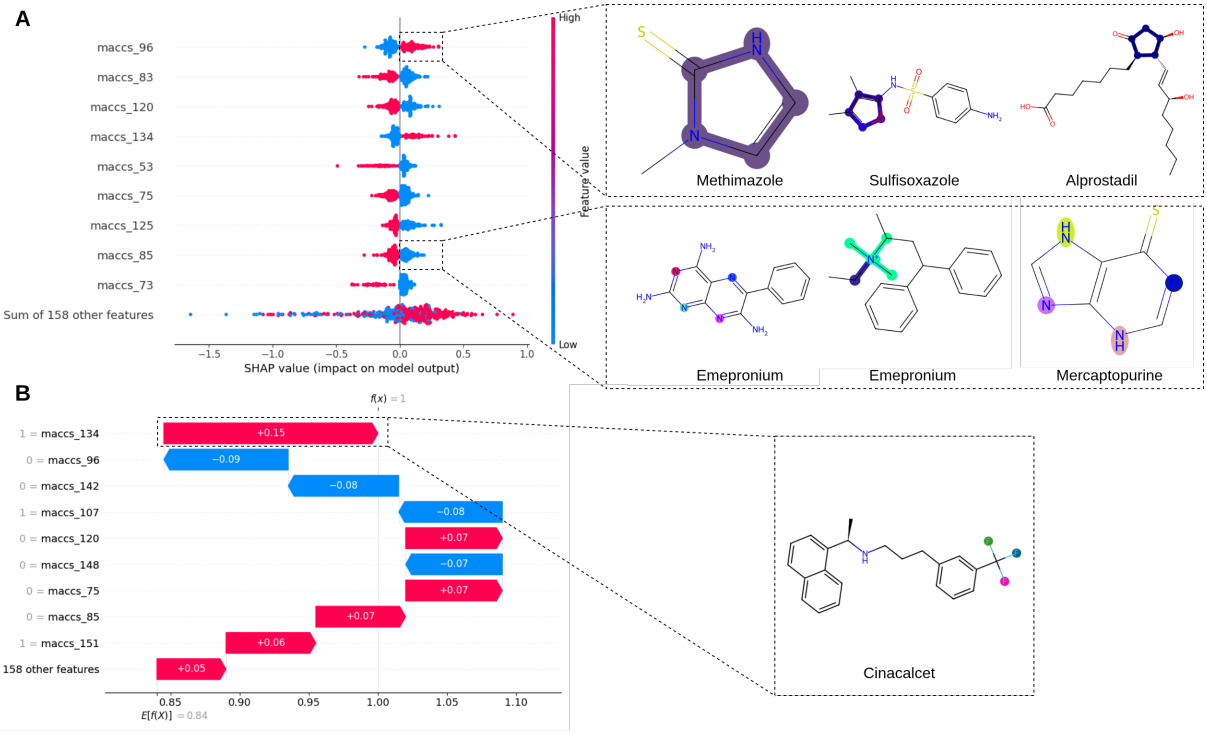


Figure 3.3: Depiction SHAP values for a ridge classifier and the most relevant MACCS keys in the molecules. The dataset used was the bioavailability dataset from [206] provided by Therapeutics Data Commons (TDC) repository [207]. **A)** provides the overall feature importance of all labels for all data points and depicts the two of the most relevant MACCS keys for the prediction. The highlighted atoms and bonds are the ones that correspond to the MACCS key. **B)** the most important features of the drug cinacalcet, showcasing how it is also possible to assess features' importance for individual cases.

3.3 Discussion: comparison with other tools

Several chemoinformatics toolkits have been developed in recent years [168, 208–211], including *Deep Chem* (`https://deepchem.io/`), OpenChem [210], AMPL [209], ZairaChem [20], QSARTuna [21], and MolPipeline [212]. These open-source projects, developed in Python, are used to construct pipelines for QSAR/QSPR modelling. A comparison between DeepMol and the other tools is showcased in Figure 3.4.



Figure 3.4: Heatmap showcasing the presence or absence of an integrated way of performing relevant tasks for constructing QSAR/QSPR systems. Green stands for presence, red for absence and yellow for incomplete integration.

Regarding dataset loading, DeepMol, *DeepChem*, QSARTuna, and MolPipeline support CSV and SDF formats, while OpenChem, AMPL, and ZairaChem only support CSV files. Hence, a limitation of the latter three tools is their incapability to import pre-computed 3D structures.

Molecule standardisation is available in DeepMol through various methods. OpenChem allows sanitising molecules, padding sequences and canonicalising SMILES. In contrast, AMPL allows the striping of salt groups, neutralising and kekulising the molecules and replacing any rare isotopes with the most common ones for each element. *DeepChem* only allows sanitising molecules when loading datasets. ZairaChem applies the MELLODDY-Tuner protocol, which includes disconnecting metal atoms from the rest of the molecule, reionising the molecule, adjusting charges and protonation states, assigning stereochemistry to the molecule, ensuring consistent representation of chiral centres, and additional standardisation procedures. QSARTuna does not incorporate chemical standardisation methods (by the definition described in this work, see Methods); however, it does include other essential preprocessing functionalities such as identifying missing data and duplicates, as well as handling various representations of the same molecule.

MolPipeline provides a wide and complete set of methods for standardisation including tautomer canonicalization, salt removal, molecule sanitization, and largest fragment selection, among others.

All tools except OpenChem can perform data splitting, but only DeepMol can perform a stratified split specific to molecular data and multitask classification. DeepMol stands out in this matter compared to all the other tools because it provides parameterisable stratified split based on scaffolds and similarity, including also the stratified splitting using the Butina algorithm. Additionally, all the tools are capable of generating numerical features from molecules. Hyperparameter and architecture optimisation are possible in all the toolkits except for OpenChem.

Model construction in these toolkits primarily focuses on DL models, but all the tools provide shallow learning models except for OpenChem. Hyperparameter and architecture optimisation are possible in all the tools except OpenChem. Moreover, while all the methods offer pipelines, only DeepMol, ZairaChem, and QSARTuna provide a comprehensive and efficient approach to automating pipeline optimisation (AutoML). However, QSARTuna only provides the optimisation of feature extraction methods and models, while both DeepMol and ZairaChem optimise other methods besides these two steps.

DeepMol and QSARTuna excel in providing feature importance analysis and effectively connecting features to molecular structures. Both ZairaChem and DeepMol also offer methods to address unbalanced datasets and perform integrated and automated feature selection, distinguishing them from the other assessed tools. ZairaChem applies feature selection through Autogluon [213], which offers only one feature selection method. QSARTuna applies variance threshold and co-correlated feature selection filtering only.

Even though DeepMol and ZairaChem appear to perform similarly for these tasks, it is important to note that ZairaChem is limited to binary classification problems. Consequently, the features provided by ZairaChem, as indicated in Figure 3.4, are specifically designed for binary classification tasks.

DeepMol offers a more complete and robust alternative to MolPipeline by also employing an integrated, self-contained pipeline serialisation approach that facilitates deployment under the *fit*, *transform*, and *predict* paradigms. In addition to encompassing all scikit-learn models, it supports a variety of DL models and allows both custom and fully automated pipeline optimisation (AutoML).

For AutoML capabilities, both DeepMol and QSARTuna utilise Optuna for pipeline optimisation. However, QSARTuna is more limited in scope, providing only eight feature extraction methods and approximately 20 models, with Directed Message Passing Neural Network (D-MPNN) and Feed Forward Neural Network (FFNN) as the only DL options. This narrower range can be restrictive for users seeking diverse modelling approaches. QSARTuna also lacks optimisation functionalities for chemical standardisation, feature scaling, and feature selection.

Another key difference lies in user configuration: QSARTuna requires users to create a custom configuration detailing the methods to be used before execution. In contrast, DeepMol simplifies the process by offering predefined configuration sets tailored to user requirements (see implementation in Supplementary Figure S1). For advanced users, DeepMol further enhances flexibility by supporting the implementation of new objective functions for generating metrics to inform the AutoML system in various validation scenarios, such as cross-validation or hold-out validation. However, QSARTuna includes features for probability calibration and uncertainty estimation, which are not yet available in DeepMol.

To compare the performance of DeepMol and QSARTuna, we evaluated them using three datasets: Pgp and CYP2D6 Substrate (both from TDC repository [207]) and DEL (from the QSARTuna publication). We created challenging train-test splits by using DeepMol's similarity splitter with a homogeneity threshold of 0.7. For pipeline optimisation, we set 20, 50, and 100 trials for the Pgp, CYP2D6 Substrate, and DEL datasets, respectively. We aimed to include all available methods by selecting the "all" pre-set for DeepMol and utilising every method documented for QSARTuna. Table 3.1 presents the comparative performance of the tools on these datasets. The metrics were chosen based on class balance, with Pgp having balanced classes and CYP2D6 Substrate and DEL being imbalanced.

Table 3.1: Performance of DeepMol compared to QSARTuna

Dataset	Tool	Metric	Result	Runtime (h:m:s.ms)
PGP (1,218 molecules)	DeepMol	ROC-AUC	**0.9178**	1:43:05.774
		F1	**0.8717**	
		Precision	**0.8263**	
		Recall	0.9224	
		Accuracy	**0.8571**	
		MCC	**0.7170**	
	QSARTuna	ROC-AUC	0.9072	0:15:36.267
		F1	0.8654	
		Precision	0.8150	
		Recall	0.9224	
		Accuracy	0.8489	
		MCC	0.7017	
CYP2D6 (13,130 molecules)	DeepMol	F1	**0.3516**	15:19:52.204
		Precision	0.2857	
		Recall	**0.4572**	
		Average precision	**0.2345**	
		Balanced accuracy	**0.5933**	
		MCC	**0.1593**	
	QSARTuna	F1	0.2382	3:07:44.322
		Precision	**0.3286**	
		Recall	0.1868	

Dataset	Tool	Metric	Result	Runtime (h:m:s.ms)
DEL (108,528 molecules)	DeepMol	Average precision	0.2170	16:12:55.992
		Balanced accuracy	0.5482	
		MCC	0.1219	
		F1	**0.1635**	
		Precision	**0.1854**	
		Recall	0.1463	
		Average precision	**0.0687**	
		Balanced accuracy	0.5566	
		MCC	**0.1269**	
	QSARTuna	F1	0.1340	3:40:33.309
		Precision	0.0882	
		Recall	**0.2785**	
		Average precision	0.0597	
		Balanced accuracy	**0.5655**	
		MCC	0.0782	

DeepMol outperforms QSARTuna across all three datasets in most predictive metrics, showcasing stronger overall performance, particularly in terms of F1 score, precision, and MCC. However, this superior performance comes with a significantly longer runtime, with DeepMol taking considerably more time than QSARTuna, especially as dataset size increases. For the Pgp dataset, DeepMol's higher ROC-AUC and F1 score highlight its advantage for small-scale studies, despite the additional computational cost. In the medium-sized CYP2D6 dataset, DeepMol maintains better metrics, though QSARTuna achieves higher precision, suggesting it is more conservative with its positive predictions. On the large DEL dataset, while DeepMol excels in F1 score and precision, QSARTuna has better recall and balanced accuracy, indicating a trade-off between capturing true positives and precision. It is important to note that DeepMol requires significantly more time due to the use of the "all" pre-set, which directs Optuna to explore the entire configuration space, including computationally intensive and time-consuming methods like DL models. Selecting the 'sklearn' pre-set would result in faster methods and reduced runtime.

3.4 Conclusion

In conclusion, DeepMol emerges as a powerful alternative to similar tools, offering a Python-based open-source framework for predicting activities and properties of chemical molecules. Its modular design allows

researchers to customise every aspect of the ML pipeline, from data processing to model prediction and explainability, catering to users with varying levels of computational expertise. DeepMol's AutoML modules further simplify the process by automatically optimising pre-processing, data engineering techniques, and ML/DL models and their hyperparameters, streamlining the selection of the most suitable combinations for a given dataset. By providing a user-friendly and easily customisable platform, DeepMol empowers researchers to efficiently explore thousands of configurations, making it an invaluable resource for accelerating chemical discovery. With the support of well-established packages like *RDKit*, *Scikit-Learn*, *Tensorflow*, and *DeepChem*, DeepMol represents a promising avenue for optimising and deploying pipelines, offering a vital contribution to the advancement of computational chemistry. The framework's availability as an open-source resource encourages collaboration and innovation in the field, facilitating progress and empowering researchers in their quest for finding new chemicals with improved properties.

Future work should focus on developing advanced methods for feature selection that can effectively handle the challenges posed by correlated features to minimise overfitting and improve model generalisability. Additionally, incorporating tools that assist in the interpretation of feature importance in the presence of feature correlation would be invaluable for guiding users toward more robust and chemically meaningful insights.

3.5 Data and code availability

The DeepMol code can be found in the DeepMol repository, and all models developed with the tool are available in the DeepMol case-studies repository.

3.6 Supplementary information

All the supplementary information can be found in `https://static-content.springer.com/esm/art%3A10.1186%2Fs13321-024-00937-7/MediaObjects/13321_2024_937_MOESM1_ESM.pdf` or in the Supplementary Information of the published, peer reviewed paper (`https://link.springer.com/article/10.1186/s13321-024-00937-7#Sec26`).

4

Comparative Assessment of Protein Large Language Models for Enzyme Commission Number Prediction

Protein Large Language Models (LLM) have been used to extract representations of enzyme sequences to predict their function, which is encoded by EC numbers. However, a comprehensive comparison of different LLMs for this task is still lacking, leaving questions about their relative performance. Moreover, protein sequence alignments (e.g. BLASTp or DIAMOND) are often combined with machine learning models to assign EC numbers from homologous enzymes, thus compensating for the shortcomings of these models' predictions. In this context, LLMs and sequence alignment methods have not been extensively compared as individual predictors, raising unaddressed questions about LLMs' performance and limitations relative to the alignment methods. In this study, we set out to assess the performance of ESM2, ESM1b, and ProtBERT language models in their ability to predict EC numbers, comparing them with BLASTp, against each other and against models that rely on one-hot encodings of amino acid sequences. Our findings reveal that combining these LLMs with fully connected neural networks surpasses the performance of DL models that rely on one-hot encodings. Moreover, although BLASTp provided marginally better results overall, DL models provide results that complement BLASTp's, revealing that LLMs better predict certain EC numbers while BLASTp excels in predicting others. The ESM2 stood out as the best model among the LLMs tested, providing more accurate predictions on difficult annotation tasks and for enzymes without homologues. Crucially, this study demonstrates that LLMs still have to be improved to become the gold standard tool over BLASTp in mainstream enzyme annotation routines. On the other hand, LLMs can provide good predictions for more difficult-to-annotate enzymes, particularly when the identity between the query sequence and the reference database falls below 25%. Our results reinforce the claim that BLASTp and LLM models complement each other and can be more effective when used together.

Keywords: enzyme annotation · deep learning · large language models

---

This chapter was co-authored by Dr. Maria Zimmermann-Kogadeeva and Dr. Aalt D.J. van Dijk and was published in the following peer-reviewed paper: *Capela, J., Zimmermann-Kogadeeva, M., Dijk, A. D. V.,*

de Ridder, D., Dias, O., & Rocha, M. (2025). Comparative assessment of protein large language models for enzyme commission number prediction. BMC Bioinformatics, 26(1), 68.

4.1 Introduction

Enzymes are natural catalysts of biochemical reactions, accelerating reaction rates by decreasing the activation energy required to form or break chemical bonds [49]. Enzymes generally demonstrate specificity for particular substrates, although they can be promiscuous towards structurally similar compounds [26]. Accurately and efficiently predicting their function is important for enhancing genome annotation workflows [13]. These predictions are critical for establishing connections between proteins and reactions, devising new pathways, and gaining precise insights into cellular metabolism.

EC numbers offer a hierarchical framework, organising enzyme functions. In the context of genome annotation procedures, EC numbers are assigned to protein sequences to succinctly encapsulate patterns of chemical reactions [214]. These correspond to the chemical changes facilitated by the enzyme linked to the respective EC number.

Given the low proportion of enzymes assigned with EC numbers [215], many tools have emerged for this annotation task, leveraging similarity and ML techniques. The former is based on the widely held idea that enzymes sharing high sequence similarity will likely have similar functions [15, 196, 216]. This annotation transfer technique is commonly adopted using the gold standard tool BLASTp (Basic Local Alignment Search Tool for protein sequences) [217]. Although it is the most employed approach in practice, it offers no way of assigning a function to proteins with no homologous sequences.

Recently, ML algorithms for the classification of enzymes that learn patterns and make predictions from protein sequence data gained a lot of attention [13, 15, 218–225]. In many of these studies, the first aim is to convert input samples, specifically amino-acid sequences, into a numerical format, referred to as features. Various approaches have been employed to generate these features, including methods based on homology [13, 15, 219, 220, 226], physicochemical properties [15, 226], presence of functional domains [15, 218], amino-acid sequence properties [220, 226], the raw amino-acid sequence [13, 15, 221], or the usage of learned representations from a protein language model [222, 224].

Over the years, several ML models were devised to learn from these numeric representations. Between 2007 and 2018, the predominant models included k Nearest Neighbours (k-NN) [218, 226], Random Forest (RF) [220], and SVM [226]. From 2018 onward, the focus shifted to DL methods, best suited for tasks involving large data volumes, aligning with the rapid expansion of databases of curated enzymes [15, 227]. DEEPre [15], one of the earliest DL models, tackles the issue of nonuniform feature dimensionality, which arises when features depend on sequence length, by developing a hybrid DL architecture. This combines CNN with Long Short-Term Memory (LSTM) networks to learn latent protein representations that depend on sequence length, like one-hot encoding and Position Specific Scoring Matrices (PSSM), and incorporates a fully connected DNN to learn length-independent features, such as the presence of Pfam [228] functional domains. Ultimately, the latent representations are concatenated and processed

by a DNN to predict the EC number. D-SPACE [12], another approach based on CNNs, was developed as a protein annotator capable of performing various tasks using a single model, including classifying EC numbers, GO terms and protein families. DeepEC [13] was developed later, combining CNNs with similarity-based searches performed with DIAMOND [229]. More recently, ProteInfer [221] was developed, leveraging deep dilated CNNs and data parallelism to train with full-length sequences of amino acids and offering an interpretation of the predictions through class activation mapping.

Although these DL models show great promise by themselves, approaches combining the model predictions and those of other approaches, such as pair-wise alignments with BLASTp [221] and DIAMOND [13, 225], or multiple sequence alignments [222] were explored. The systematic usage of these approaches suggests that DL models provide sub-optimal results that are later compensated by alignments' predictions. The authors of ProteInfer provided an analysis of the results of BLASTp and those of DL models, reaching the general conclusion that BLASTp performs slightly better than ProteInfer deep dilated CNN, and an ensemble of the two delivers performance surpassing that achieved by the individual techniques [221]. On the other hand, CLEAN showcased higher predictive capabilities than BLASTp for independent small datasets of understudied enzymes [224] by mitigating the unbalanced distribution of EC numbers within the training dataset through the application of contrastive learning.

Since LLMs have gained significant traction, protein LLMs have been used for EC number prediction [222–225]. These models are based on the transformer network, which employs self-attention mechanisms to weigh the importance of different parts of a sequence, enabling to capture long-range dependencies.

Two types of pre-trained LLMs have been used for EC number prediction so far: Evolutionary Scale Modelling (ESM) [230, 231] and ProtBERT [232]. ESM models are transformer networks pre-trained with UniProtKB data [227] and were used for feature extraction, involving taking the outputs from a layer within these models and using them as features (embeddings) for predicting the EC number [222, 224]. ESM models have been proven to be highly competitive amongst LLMs for protein representation and function prediction [233]. On the other hand, ProtBERT is a transformer network pre-trained with data from UniProtKB and the BFD database [234] and was fine-tuned for EC number prediction [223, 225]. Indeed, although many studies have investigated ESM and ProtBERT models for EC number prediction [222–225], a thorough comparison of these LLMs as feature extractors has not yet been conducted. This lack of comparative analysis leaves unanswered questions about the relative effectiveness of different LLMs in this specific application.

Furthermore, it is important to recognise that the methodologies discussed in these studies typically do not demonstrate their performance independently without the aid of alignments. Moreover, none of the above investigations have thoroughly compared LLMs to BLASTp on large test sets. This leaves open questions about LLMs' performance against BLASTp, their potential advantages, and where LLMs might fall short or require further development to meet or exceed BLASTp's capabilities.

This study presents an in-depth evaluation of advanced protein representations and DL architectures for predicting EC numbers. By addressing the noticeable gap in existing research, we aim to clarify their relative effectiveness. To this end, we have designed a robust experimental framework to evaluate

DL models tailored for EC number prediction, trained with embeddings of ESM2, ESM1b, and ProtBERT. Moreover, we provide implementations of DeepEC and D-SPACE for comparison. Since these methods use one-hot encodings as input rather than LLM embeddings, they offer a useful way to compare different input types. Through this comprehensive approach, we aim to critically assess the performance of both BLASTp and DL models, thereby contributing valuable insights into the effectiveness of DL models, enhanced by state-of-the-art LLM embeddings or one-hot encodings, compared to a traditional, alignment-based protein function prediction methodology.

4.2 Results

4.2.1 DNN-LLMs perform generally on par with BLASTp

Our workflow included a benchmark of models trained with ESM and ProtBERT embeddings compared to BLASTp. Initially, linear baseline models were established, setting a reference for evaluating more complex models. These baselines were then enhanced with hidden layers and batch normalisation and compared against reimplementations of architectures like D-SPACE and DeepEC. A clear enhancement in performance was verified from the baseline models to their counterparts, as shown in Supplementary Figure S2.

Figure 4.1a demonstrates that models using LLM embeddings substantially outperformed those that did not, indicating their advantage for the EC prediction task. This was evident compared to DeepEC CNN3 and D-SPACE EC models, which relied on a one-hot encoding of amino acid sequences. Comparing the performance of BLASTp and DNN-LLM models reveals that BLASTp was slightly better than the DNN-LLM models in terms of mF1 score, with a difference of 0.011 from the worst DNN-LLM model. Figure 4.1c reveals a consistent trend across all EC levels.

Regarding mPrecision and mRecall, the results indicate that all models, except DeepEC CNN3 and D-SPACE EC, surpass BLASTp in mPrecision, while BLASTp demonstrates better mRecall values. This observation is consistent with the Recall-Precision curves reported by [221]. They suggest that BLASTp's higher recall may be due to its direct utilisation of the entire training set, in contrast to neural networks that condense this data into a limited set of weights [221]. On the other hand, BLASTp's comparatively lower precision could result from evolutionary changes in sequences, where proteins might closely match parts of a training sequence but lack key regions crucial for the enzyme function [221]. Additionally, we can observe that combining DNN-LLM models with BLASTp in an ensemble approach (Models + BLASTp) leads to a better performance than using either approach independently (Figure 4.1c), a pattern that is consistent across EC levels, confirming the findings from [221].

Figure 4.1b shows the mF1 scores of the models applied to subsets of the test set organised by sequence identity levels with the training set, calculated by BLASTp. The findings indicate that within the lowest identity interval (below 15%), where BLASTp cannot make any predictions, even the least accurate DL model, DeepEC CNN3, still achieved a mF1 score of 0.04, whereas the best model, DNN ESM2 3B,

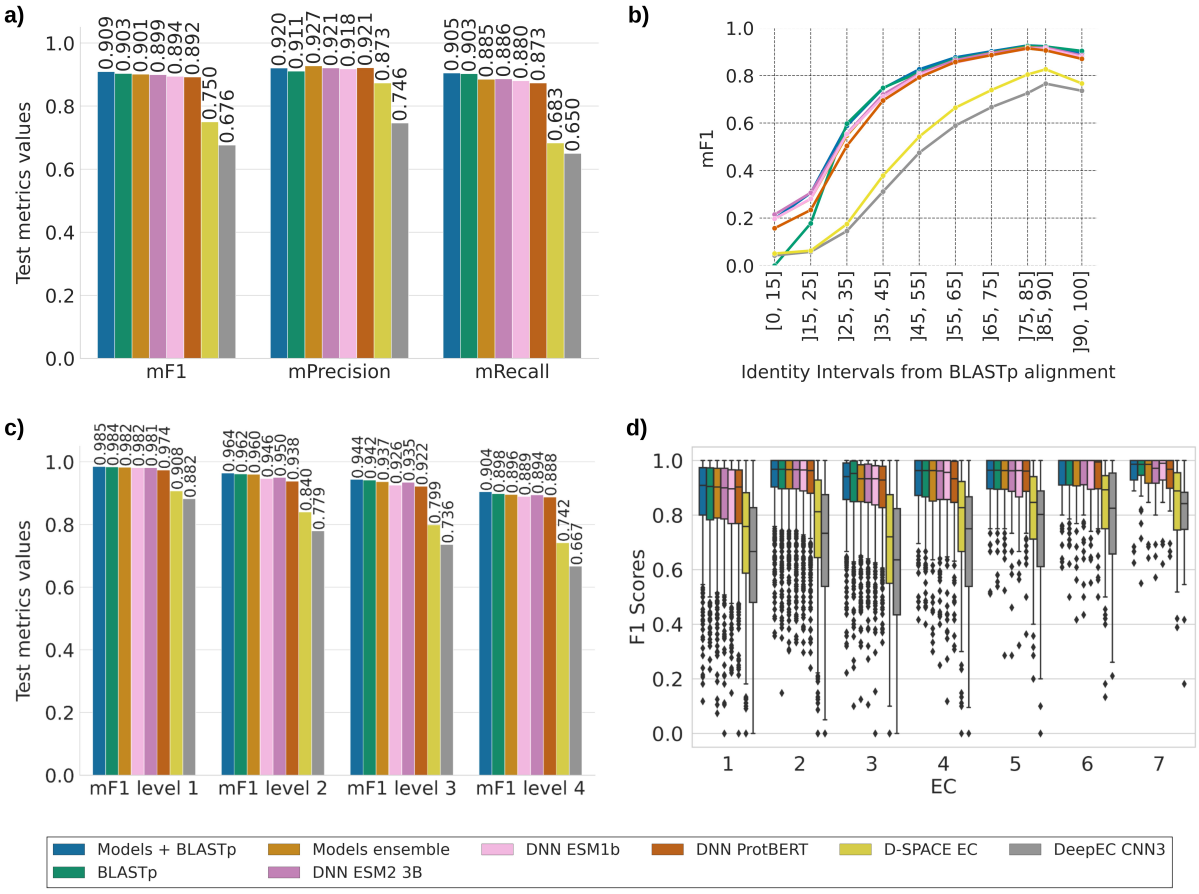


Figure 4.1: Results and evaluation using the whole test set. **a)** mF1, mRecall and mPrecision of all models. **b)** mF1 score for different identity intervals. **c)** mF1 for the different levels of EC numbers (i.e. level 1: X.; level 2: X.X.; level 3: X.X.X.; level 4: X.X.X.X). **d)** Distribution of F1 scores across level 4 EC classifications (X.X.X.X format) for each method. This boxplot displays F1 score distributions organised by EC category of level 1 (1–7), capturing the performance of each method across specific enzyme functions within these categories (e.g. EC 1 includes F1 scores for subcategories (1.X.X.X), EC 2 includes F1 scores for subcategories (2.X.X.X), and so forth.

reached 0.21. Below 25% of identity, the DNN-LLM models provided better predictions than BLASTp; however, they performed marginally worse than BLASTp for higher identity levels.

When no similar sequences were available, BLASTp performed worse than DL models. This is increasingly relevant in functional annotation pipelines, as more sequences from previously uncharacterised organisms become available. For example, current alignment-based tools are unable to determine the functions of one-third of microbial protein sequences [235].

Figure 4.1d provides a detailed view of F1 scores across level 4 EC classifications (X.X.X.X format), where results are aggregated by EC category of level 1 (1–7). In this boxplot, we can have a better perspective on the variability of performances of each method per EC category. The standout finding is clear: DNN-LLM models and BLASTp consistently outperform CNN-based models, and the performance difference is statistically significant (p-value < 0.001 in the WSR test). At the highest precision level (level 4), BLASTp takes a narrow lead, surpassing only EC 3 (hydrolases). Generally, all DL models fell shorter in

predicting ECs across EC 3 than BLASTp. Furthermore, BLASTp performs significantly better than other models like DNN ESM1b and DNN ProtBERT for EC 1, 2, and 3.

When comparing our top combination, Models + BLASTp, with BLASTp alone, the combined approach shows a clear advantage, particularly for EC 2 (transferases). Models + BLASTp also outperformed individual DNN-LLM models across the board, except for EC 7 (translocases), EC 6 (for DNN ProtBERT and DNN ESM1b), and EC 5 (for DNN ProtBERT).

Interestingly, ECs 6 and 7 emerged as the 'easier' categories, with median F1 scores at or near one across all levels and impressively low variability, indicating consistent predictions. EC 2 is another level which was predicted well by most predictors, with low variability of median F1 scores. EC 1, however, presents the greatest challenge, with the lowest median F1 scores and 75th percentiles across the board and all levels.

The models were then evaluated on the subsets of enzymes with additional information, such as literature references (evidence-level dataset), enzyme promiscuity and multi-functionality (see Methods). We identified the wF1 score as the most suitable metric for these cases, given their smaller size and the evident class imbalance, further detailed in Supplementary Figures S3 and S4. Tables 4.1 and 4.2, and Supplementary Figure S5 present the results for these datasets. The results reveal that BLASTp generally outperforms individual DNN-LLM models and the Models ensemble at EC levels 2, 3, and 4, albeit the margin of performance difference is not substantial. In almost all cases, except for level 4 in the evidence-level dataset, the Models + BLASTp ensemble performed better than any of the methods used independently, following the trends observed in the overall test set. Accordingly, among the DNN-LLM models, DNN ESM2 3B emerged as the top performer overall, except for level 1, where DNN ESM1b outperformed it.

While these trends remained consistent, there was a noticeable decline in performance for the dataset comprising promiscuous and multi-functional enzymes (those with multiple EC numbers). This indicates that making predictions for enzymes with more than one EC number is more challenging, highlighting the complexities associated with enzyme promiscuity and multifunctionality.

4.2.2 BLASTp outperforms DNN-LLMs across identity thresholds

Next, we set out to investigate whether the identity thresholds based on which the enzymes are split into train and test datasets affect the observed differences in models' performances. To this end, we performed a 5-fold cross-validation with train-test splits based on different identity thresholds. Figure 4.2 compares the mF1 score of various models across different identity thresholds, demonstrating how each model's performance improves as the identity threshold increases from 40% to 90%.

BLASTp (green line) consistently shows strong performance across all identity levels, achieving significantly higher (p-value < 0.001 in a WSR test) mF1 scores than DL models at nearly all identity levels, lagging behind only Models + BLASTp.

The DNN-LLM models generally show competitive performance, especially as identity increases, but tend to fall slightly below BLASTp. They perform reasonably well at mid-identity levels, with performance

Table 4.1: wF1 scores for the evidence-level dataset. The levels refer to the different hierarchical levels of the EC number system (i.e. level 1: X.; level 2: X.X.; level 3: X.X.X.; level 4: X.X.X.X)

	wF1			
Models	**lvl 1**	**lvl 2**	**lvl 3**	**lvl 4**
DNN ESM2 3B	0.967	0.945	0.928	0.862
DNN ESM1b	0.969	0.943	0.922	0.852
DNN ProtBERT	0.953	0.922	0.902	0.842
DeepEC CNN3	0.818	0.719	0.679	0.587
D-SPACE EC	0.854	0.780	0.751	0.651
BLASTp	0.967	0.948	0.931	**0.878**
Models ensemble	0.970	0.946	0.928	0.861
Models + BLASTp	**0.973**	**0.952**	**0.936**	0.876

stabilising as identity reaches 90%. Among these, DNN ESM2 3B is clearly the best at all identity thresholds, significantly outperforming the rest (p-value < 0.001 in a WSR test), being followed by DNN ESM1b which significantly outperforms DNN ProtBERT except for the identity threshold of 90%.

D-SPACE EC and DeepEC CNN3 models (yellow and grey lines) show significantly lower performance than the other methods (p-value < 0.001 in a WSR test), particularly at lower identity levels. Their mF1 scores improve with increasing identity but remain below those of BLASTp and the DNN-LLM models.

The combination of models with BLASTp (Models + BLASTp) achieves the highest overall performance. This method significantly surpasses all other approaches at every identity threshold (p-value < 0.001 in a WSR test), except for the 40% threshold, where it is slightly outperformed by BLASTp alone, but with no significant differences.

4.2.3 DNN-LLMs successfully learn the hierarchy of the EC system

Figure 4.3 shows how each evaluated model performs in terms of learning the hierarchical structure of the EC system at different thresholds of positive classification (Figure 4.3a and c) and using the threshold of 0.5 (Figure 4.3b).

Figure 4.3a compares models based on remaining uncertainty (missing true information) and misinformation (incorrect predicted hierarchy paths) at different positive classification thresholds, with semantic distance (S-min) combining these measures. DNN ProtBERT, DNN ESM2 3B, and DNN ESM1b (DNN-LLM) have the lowest S-min values, indicating that they capture the most complete and accurate information, minimising both missing and misleading annotations. In contrast, DeepEC CNN3 and D-SPACE

Table 4.2: wF1 scores for the promiscuous and multi-functional enzymes dataset. The levels refer to the different hierarchical levels of the EC number system (i.e. level 1: X.; level 2: X.X.; level 3: X.X.X.; level 4: X.X.X.X)

	wF1			
Models	**lvl 1**	**lvl 2**	**lvl 3**	**lvl 4**
DNN ESM2 3B	0.925	0.910	0.880	0.742
DNN ESM1b	0.941	0.907	0.880	0.735
DNN ProtBERT	0.932	0.888	0.856	0.742
BLASTp	0.940	0.913	0.893	0.770
DeepEC CNN3	0.781	0.685	0.626	0.447
D-SPACE EC	0.839	0.753	0.699	0.502
Models ensemble	0.937	0.912	0.883	0.748
Models + BLASTp ensemble	**0.949**	**0.923**	**0.898**	**0.779**

EC show higher S-min scores, suggesting that they miss more true information or include more incorrect paths, making them less effective overall. Among DNN-LLMs, DNN ESM2 3B stands out as the most optimal model with an S-min of 0.240 and with the most left-cornered remaining uncertainty-misinformation curve, outperforming DNN ESM1b and DNN ProtBERT.

Besides the metrics described above, we recognise that in hierarchical multi-label classification, understanding the relationships between labels is also key to learn the hierarchy. Figure 4.3b shows the Hierarchical Consistency Error (HCE) for each DL model at a positive classification threshold of 0.5, indicating that DL models utilising LLM embeddings, particularly ESM-like, substantially outperformed others in terms of HCE. This shows that LLM-based models can achieve higher mF1 scores and be more proficient at learning the EC hierarchy, with ESM-like embeddings outperforming ProtBERT. However, applying simple heuristics to these predictions could eventually reduce this error and improve the results. For instance, if one model predicted the EC number as 1.1.1.1, one could apply a heuristic to assume that the precedent levels are 1, 1.1 and 1.1.1.

Since a prediction threshold of 0.5 (the output of the sigmoid function in the output layer) may not fully capture how the models perform in terms of mPrecision and mRecall and learn the hierarchy at different levels, we examined their behaviour across a range of prediction thresholds. Figure 4.3c shows precision-recall (PR) curves across these thresholds. The PR curve and respective Area Under the Curve (AUC) of DNN-LLM models show that these models outperform CNN-based models (D-SPACE and DeepEC). Generally, the DNN-LLM-based models show strong initial mPrecision values from 0.96 to 0.91 when mRecall is below 0.85, but slowly decrease to a minimum of 0.85 as recall approaches 0.90, with slight

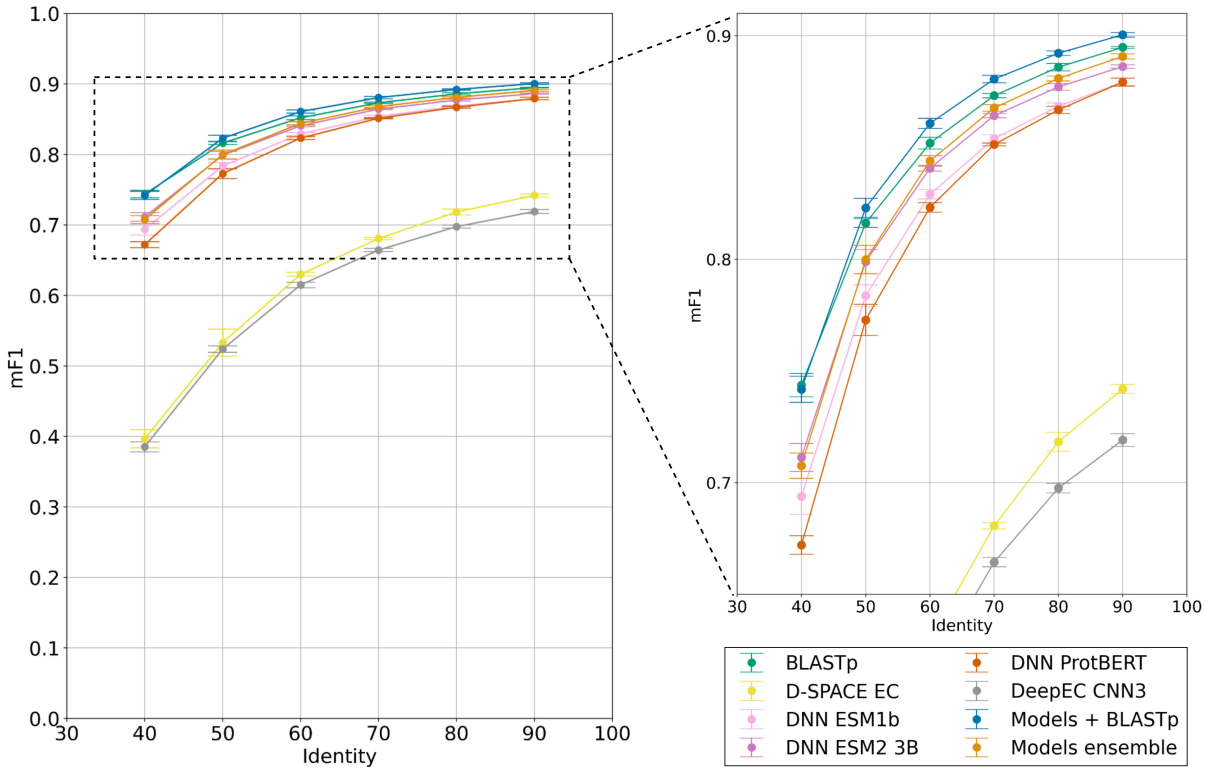


Figure 4.2: The mF1 scores are presented as a function of various identity thresholds in a train-test split setting across five folds. On the left, a comprehensive overview of the results is displayed on a scale from 0 to 1, showing the mF1 scores. On the right, a zoomed-in version of the plot on the right provides a more detailed view of these results.

differences in DNN ESM2 3B, where the decrease is less steep.

Moreover, a closer look at the results shows that DNN ProtBERT and ESM2 3B both achieve very high mPrecision and mRecall values, with DNN ESM2 3B slightly outperforming DNN ProtBERT in both metrics at multiple thresholds. However, PR-AUC is not solely a function of the highest precision or recall values. It captures the trade-off between mPrecision and mRecall across all threshold levels. The differences in the progression of mPrecision and mRecall rates across thresholds result in a PR curve for DNN ProtBERT that covers slightly more AUC in certain conditions. The major difference is when the values of mRecall are lower than 0.8, DNN ProtBERT maintains a higher mPrecision than DNN ESM2 3B. However, for mRecall above 0.8, DNN ESM2 3B mPrecision is higher than for DNN ProtBERT. This suggests that DNN ESM2 3B is better suited for higher threshold settings and is likely less sensitive to threshold changes, particularly when mRecall exceeds 0.8. The higher precision DNN ESM2 3B achieves at high recall levels indicates that it is more stable in maintaining mPrecision as mRecall increases, which is beneficial when high-recall, high-precision predictions happen, which is the case (see Figure 4.1). In contrast, DNN ProtBERT's strength lies in maintaining higher mPrecision at lower recall levels (under 0.8). This means that ProtBERT is more responsive in prioritising correct predictions when aiming for moderate recall but may lose mPrecision as mRecall pushes closer to 1.0. This characteristic implies that DNN ProtBERT could be advantageous in cases where more selective, lower-recall thresholds are acceptable, but it may

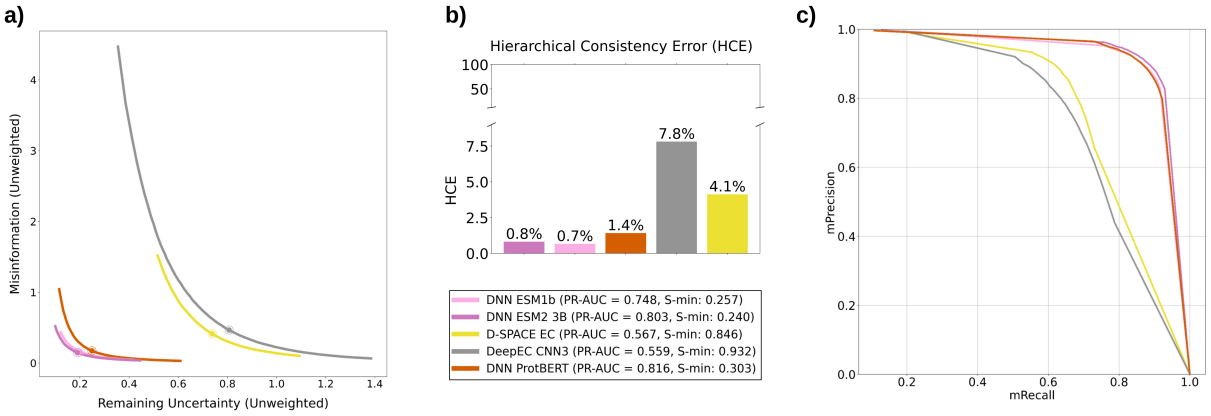


Figure 4.3: The performance of each evaluated model in learning the hierarchical structure of the EC system using the whole test set, with panels **(a)** and **(c)** showing results across different positive classification thresholds, and panel **(b)** focusing specifically on the threshold of 0.5. **a)** Remaining uncertainty and misinformation at different thresholds. S-min is represented in the plot as a circle. **b)** HCE for all DL models. **c)** mRecall and mPrecision across different positive classification thresholds. The area under the curve is shown in the legend.

not be as stable across higher recall ranges compared to ESM 3B.

4.2.4 DNN-LLMs and BLASTp are complementary

To test whether different methods performed better for different enzyme types, we next compared the performance of BLASTp and DNN-LLM models (in terms of the difference in F1 scores from both methods) for each EC class, spanning all levels ($\Delta F1_i = F1_{\text{DNN-LLM},i} - F1_{\text{BLASTp},i}$, where i is one class of EC numbers), as shown in Figure 4.4. This figure indicates that while some enzyme classes have the same F1 scores for both methods, others vary. Figure 4.4 includes scatter plots illustrating how the differences in F1 scores between the DNN-LLM models and BLASTp relate to the number of sequences in each EC class. Also, bar plots show how many EC classes have F1 score differences surpassing a set threshold.

The analysis indicates that BLASTp demonstrated superior performance for more EC numbers than DNN-LLM models. However, it is important to highlight that each DNN-LLM model evaluated in this study outperformed BLASTp on hundreds of classes. Among these, DNN ESM2 3B emerged as the top performer, annotating 792 classes better than BLASTp, followed by DNN ESM1b and DNN ProtBERT, with 746 and 710 classes. Regarding F1 score differences, the leading model, DNN ESM2 3B, exhibited a minimum increase of 0.1, 0.2, and 0.3 in F1 scores for 107, 14, and 3 classes, respectively. A notable mention is DNN ESM1b, which achieved a significant F1 score improvement compared to BLASTp of 0.4 in two labels (1.1.1.274 and 4.3.2.3). Detailed comparisons of specific EC numbers where DNN-LLM models outperformed BLASTp can be found in Supplementary Table S7.

Figure 4.4 highlights underrepresented classes, specifically those with fewer than 63 sequences per class (red squares). There are 69 such classes. The plot delineates which classes are more accurately annotated by BLASTp (indicated by values below 0) versus those better annotated by the DNN-LLM models

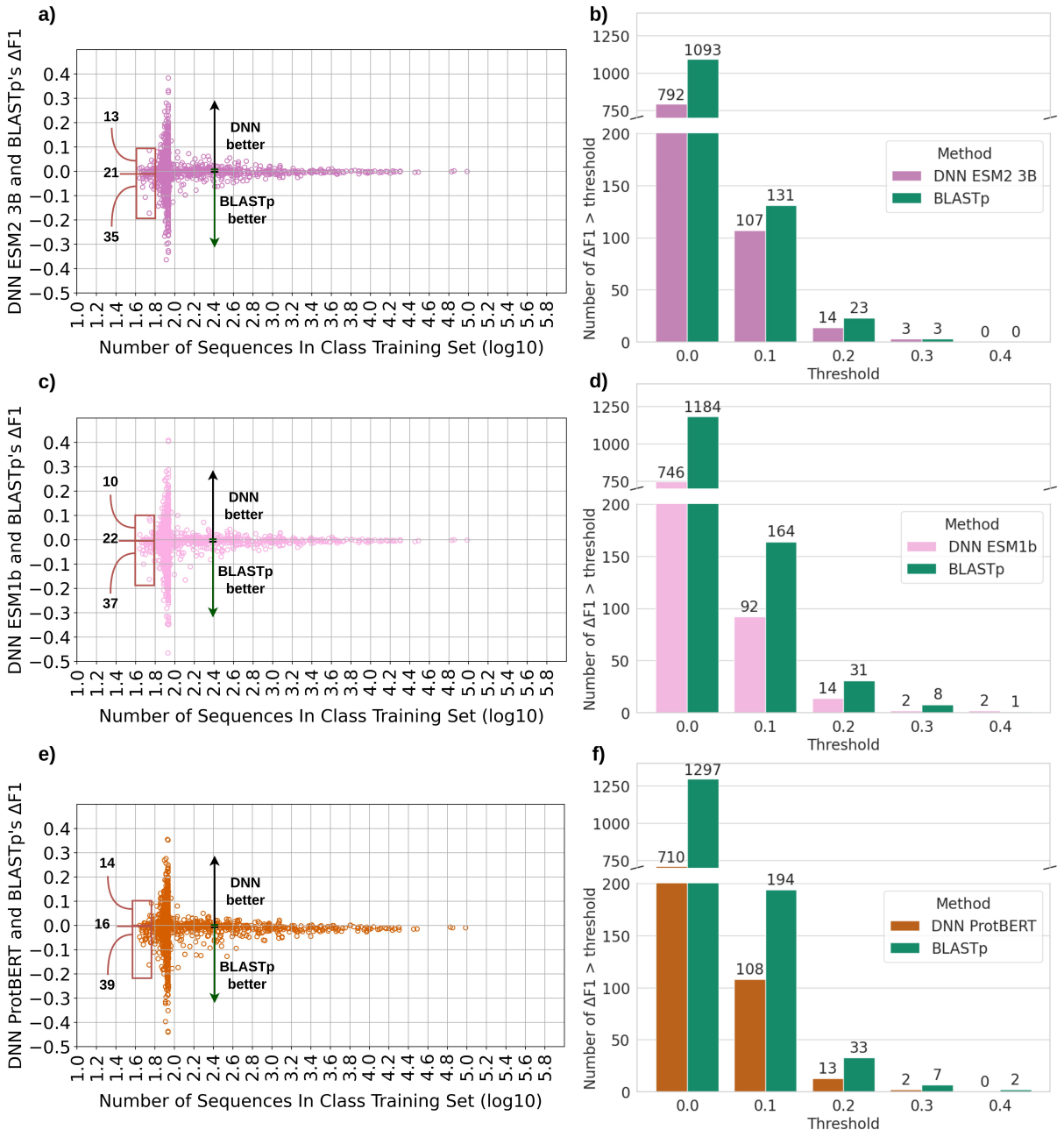


Figure 4.4: Comparison of DNN-LLM models' and BLAST's F1 scores. **a,c,e)** Distribution of DNN-LLM models' and BLASTp's F1 score differences per number of sequences in the training set ($\log_{10}$), where each dot represents an EC class. **b,d,f)** Number of classes with an $\Delta F1_i$ above a given threshold.

(indicated by values above 0). The findings reveal that, among the differentially annotated classes i.e. $\Delta F1_i \neq 0$, the proportion of classes better annotated by DL models is lower for these underrepresented classes (24.9%) than the entire set of classes (38.7%). This suggests that while BLASTp generally annotates more classes effectively, this is more pronounced for classes represented less well in the training set.

4.2.5 DNN-LLMs outperform BLASTp on difficult tasks

To challenge the EC prediction methods with more difficult tasks, we applied them to two published datasets: newly annotated bacterial enzymes [236] and halogenases [224]. Since these datasets are independent and contain enzymes not found in the training sets of any of the following state-of-the-art approaches, we compared our methods with two leading approaches: ProteInfer, a CNN-based method, and CLEAN, a contrastive learning model built using ESM1b embeddings.

The former comprises enzymes whose functions were recently determined through a high-throughput experimental genetic study [236]. The authors provided new EC numbers for enzymes where prior annotations in the SEED or Kyoto Encyclopedia of Genes and Genomes (KEGG) databases were either incorrect or inconsistent. Given this context, this dataset is likely enriched with proteins whose functions pose a significant challenge for computational assessment, making it invaluable for testing and refining computational annotation methods. For evaluation purposes, wRecall, wPrecision, and wF1 scores at level 4 were exclusively employed to address the significant class imbalance and to align with the metrics used by [224], facilitating comparisons with other tools.

Table 4.3: Model wRecall, wPrecision and wF1 scores for the [236] dataset.

Method	**wF1**	**wRecall**	**wPrecision**
CLEAN	**0.495**	**0.584**	0.467
ProteInfer	0.166	0.138	0.243
DNN ESM2 3B	0.433	0.408	0.535
DNN ESM1b	0.421	0.382	0.587
DNN ProtBERT	0.357	0.316	0.526
Models Ensemble	0.392	0.362	0.525
BLASTp	0.372	0.329	0.499
Models + BLASTp	0.444	0.408	**0.600**

BLASTp performs worse than all methods except DNN ProtBERT (Table 4.3). Moreover, ProteInfer performs worse than all methods, which reinforces the idea that LLM-based models outperform CNN-based models. The custom models developed for this work, DNN ESM2 3B and DNN ESM1b, exhibit strong performance. The Models ensemble shows balanced performance across all metrics but does not outperform the individual DNN ESM2 3B model. The Models + BLASTp ensemble outperforms other methods in wPrecision, including CLEAN. On the other hand, CLEAN delivers better wF1 and wRecall than any other approach. This shows that models trained with LLM embeddings are promising to (re-)annotate enzymes that are difficult to predict or have inconsistent annotations across biochemical databases.

The second challenging dataset comprises 36 halogenases, a group of incompletely annotated enzymes identified from the UniProtKB database. The dataset presents a significant challenge since the halogenase family is relatively understudied, and the existing annotations in databases are often incomplete or conflicting, complicating the task of accurate classification. Despite these challenges, expert curation was conducted by [224], leading to the confident assignment of specific EC numbers to all 36 halogenases in the dataset. To assess method performance, accuracy across all levels was computed, given the lesser degree of imbalance, ensuring alignment with the evaluation criteria outlined in [224] and enabling comparative analysis with alternative tools. Furthermore, given the small sample size in this dataset, binomial tests were conducted to compare the rates of correctly and incorrectly annotated proteins for each EC number at each level. The expected proportions of correctly annotated proteins were estimated based on the overall number of correctly annotated proteins within each level.

A halogenase refers to an enzyme that incorporates halogen atoms (such as chlorine, fluorine, or bromine) into organic compounds, often for catalysing halogenation reactions that add bioactive properties to compounds [237]. In the context of EC terminology, these enzymes fall under specific EC classifications based on their reaction types, such as haloperoxidases (classified as oxidoreductases, EC 1.11.1.10) or flavin-dependent (1.14.19.x, depending on the substrate), α-ketoglutarate–dependent (1.14.20.x and 1.14.11.x), and S-adenosyl-L-methionine (SAM)-dependent halogenases, which act as adenosyl transferases (fluorinase, EC 2.5.1.63; chlorinase, EC 2.5.1.94) and share a nucleophilic-substitution mechanism at C5' of SAM with the related SAM hydrolase EC 3.13.1.8 [224].

Table 4.4: Model accuracies for the halogenases dataset.

Method	**Accuracy**			
	level 1	**level 2**	**level 3**	**level 4**
CLEAN	**1.000**	**0.972**	**0.944**	**0.944**
ProteInfer	0.611	0.388	0.083	0.694
DNN ESM2 3B	0.917	0.917	0.500	0.805
DNN ESM1b	0.917	0.917	0.500	0.833
DNN ProtBERT	0.805	0.917	0.528	0.750
Models Ensemble	0.917	0.917	0.500	0.833
BLASTp	0.750	0.639	0.194	0.583
Models + BLASTp	0.917	0.917	0.555	0.833

[a] This assessment was performed based on the ground truth derived only from the manual curation in [224].

BLASTp exhibited lower accuracy than most methods, as presented in Table 4.4. As BLASTp was not able to find homologous sequences for four query sequences from this dataset, this might have

affected the results compared to DL models, since DNN-LLMs could generate correct predictions for these sequences. At the same time, DNN ESM2 3B, DNN ESM1b, and DNN ProtBERT varied in their highest accuracies across levels 1 and 2, but all showed decreased performance at levels 3 and 4. The Models ensemble matched the top performances of DNN ESM2 3B and DNN ESM1b at the first two levels but also dropped in accuracy at the lower levels. Models + BLASTp achieved the highest overall across all levels without considering CLEAN's performance. CLEAN obtained the best accuracies across the board, and ProteInfer fell short at all levels compared to all the other methods. Binomial tests revealed that, for DNN ESM2 3B, the confidence interval for the predictions of haloperoxidases (EC 1.11.1.10) was slightly above the expected value of baseline performance at level 4 (see Supplementary Material). Similarly, DNN ESM1b and DNN ProtBERT demonstrated confidence intervals exceeding the baseline expectation for the prediction of α-ketoglutarate–dependent halogenases (specifically those within EC 1.14.11.x). These results suggest that these models exhibit superior annotation performance for these enzyme classes compared to other enzymes at the same level.

In this case, the challenges across different EC levels arise from the need to distinguish broad functional categories (level 1), reaction-related components like solvents, cofactors, and electron acceptors (levels 2 and 3, especially in oxidoreductases), and specific substrates (level 4). CLEAN effectively identifies subtle patterns in the ESM1b embeddings that indicate substrate-specific information, which helps explain its strong performance at the detailed, substrate-specific level 4 EC classification.

4.3 Discussion

This study compared DL models trained with embeddings derived from LLMs and the gold standard method BLASTp. This study delivers a fully reproducible end-to-end pipeline for benchmarking models and protein representations for EC number prediction. We show that using LLMs as straightforward feature extractors leads to better results than models relying on the one-hot encoding of sequences (e.g. D-SPACE, DeepEC and ProteInfer). Generally, DNN-LLM models were marginally outperformed by BLASTp for large datasets with homologous sequences available from databases. While BLASTp demonstrated a slight performance improvement over DL models, it is important to note that it has an inherent advantage in this context. Since it is one of the most widely used annotation methods, it is used to annotate enzymes in UniProtKB, which we use for testing the methods, thus favouring BLASTp. Considering that, DL models trained with LLMs achieve impressive results, with a minimal difference of 0.004 in the mF1 score compared to the one from BLASTp. Despite this slight performance gap, in scenarios where the training set lacked homologous sequences, LLM models outperformed BLASTp, indicating their potential utility in more challenging annotation tasks. Particularly, when BLASTp fails to find an enzyme with more than 25% identity in the training set, its performance declines, and DL models outperform it. Hence, we claim that DL models should be employed below this mark, instead of BLASTp. Additionally, combining BLASTp with DL models can enhance results across nearly all identity intervals. We also confirm that DL models are slightly more precise and have lower recall than BLASTp, as pointed out by [221].

As future work on integrating BLASTp and DL models, a more nuanced approach could be adopted rather than simply favouring DL models over BLASTp based solely on sequence identity. Specifically, an alternative method could be implemented to adjust predictions according to the number, query coverage, sequence identity and other metrics of identified BLASTp hits. This approach could involve using a linear regression model or a similar ML technique to optimise the integration of predictions from both BLASTp and DL models. By leveraging the strengths of each method under different conditions, this strategy has the potential to enhance overall predictive performance.

It is important to emphasise that this framework predicts EC numbers for a given enzyme but does not determine whether the input molecule is an enzyme. Future efforts should focus on benchmarking protein LLM models for this specific task.

Crucially, we show that protein LLM models should still be improved to further narrow or even surpass the performance gap with BLASTp and potentially become the gold standard in enzyme annotation. Overall, we strengthen the claim that BLASTp and LLM models are complementary tools that combined can be advantageous when employed in mainstream protein annotation routines and hard annotation tasks.

4.4 Methods

4.4.1 Defining EC number classification problem

EC number prediction was defined as a multi-label classification problem incorporating promiscuous and multi-functional enzymes (with more than one EC number). Let X be the set of protein sequences and Y be the set of EC numbers. Each protein sequence x_i in X has an associated binary label vector y_i of length $|Y|$, where $|Y|$ is the total number of unique EC numbers. Each vector element y_{ij} is either 0 or 1, indicating whether protein sequence x_i is associated with the EC number j in Y. It is worth noting that all levels were included in the resulting matrix as proposed by [218], so if an enzyme is assigned the EC numbers 1.1.1.1 and 4.1.1.1, ones are placed in y_i at the positions corresponding to 1, 1.1, 1.1.1, 1.1.1.1, 4, 4.1, 4.1.1 and 4.1.1.1. Herein, we take a global approach for hierarchical multi-label classification, where a single classifier is challenged to predict the entire hierarchy of labels and their relationships [238].

4.4.2 UniProtKB data extraction and processing

The SwissProt (manually annotated), TrEMBL (automatically annotated) protein data and UniRef90 clusters were downloaded from UniProtKB [227] in XML format in February 2023. Then, an XML parser was developed to extract the EC numbers, protein sequences, and identifiers of SwissProt and TrEMBL. Only the UniRef90 cluster representatives were kept in the datasets, as first proposed by [218]. By definition, UniRef cluster representatives are chosen based on entry quality, annotation score, organism relevance, and sequence length, enhancing biologically relevant information retrieval. This procedure ensures that enzymes that share more than 90% identity with these representatives are removed from the dataset, thus avoiding redundancy and prioritising entries with better annotation.

Afterwards, for all EC numbers with less than 100 sequences, we added data from TrEMBL, if available. The pipeline then systematically removes the underrepresented EC numbers, numbers with less than 100 sequences assigned, and provisional EC numbers with the letter "n" in the last level. The final dataset included 380,811 enzymes, categorised under EC numbers as follows: 7 for level 1, 69 for level 2, 230 for level 3, and 2,465 for level 4.

4.4.3 Data splitting

Dividing data into training, validation and test sets poses challenges with highly sparse matrices of labels like the ones in this work. It is crucial to ensure that all labels are well-distributed across the sets. To address this, we opted for a split of 60% for training and 20% each for validation and testing, firstly using an iterative approach of stratification of multi-labelled data from the *scikit-multilearn* package v. 0.2.0, described by [239]. Upon running this tool, the stratification was unsuccessful, as a few EC classes had no sequences assigned in the test or validation sets. We, thus, developed an algorithm (Supplementary Algorithm S1) that iteratively transfers a calculated portion of sequences associated with EC numbers from the training set to the test or validation set. The algorithm targets EC numbers where the test-overall data proportion is lower than 15%, a value determined as optimal after some tests. It calculates the number of samples to be moved to compensate for this gap. This ensures that underrepresented EC numbers are more evenly distributed across the datasets, as corroborated in Supplementary Table S1. All the resulting dataset information can be referred to the Supplementary Tables S1 and S2. This process of stratified splitting may lead to certain EC number classes having fewer than 100 sequences each in the resulting datasets.

Complementarily, we developed a train-test split strategy that focuses on identity thresholds, while resampling the data into five folds. We implemented six distinct identity thresholds (90%, 80%, 70%, 60%, 50% and 40%) using CD-HIT [240] to ensure that no sequences in the training set share more than the specified identity threshold with those in the test set. In total, we got 30 train-test combinations (six identity thresholds times the five folds). This approach allowed us to evaluate how similarity affects the performance of each method.

4.4.4 Designing the evaluation datasets

In this work, we evaluated the predictive capacity of the models on five different datasets. The first three are progressively refined datasets derived from the initial test set based on UniProtKB data and obtained from the employed data-splitting strategy. The first dataset included the entire test set, while the second was refined to proteins with an EC number corroborated by at least one literature reference, termed the evidence-level dataset. While this test set may contain some automatically inferred annotations, we believe that its reliance on manually curated SwissProt entries with literature citations strikes a good balance between annotation quality and dataset size, especially since manually annotating all entries would be too time-consuming, more information is included in the Supplementary Information. The third dataset, taken from the evidence-level dataset, focused on promiscuous and multi-functional enzymes. Two additional,

fully independent datasets were considered for evaluation: one from [236] with 149 bacterial enzymes, known for its experimentally validated yet challenging data, and another comprising 36 incompletely annotated halogenases identified from the UniProtKB database, further curated by [224]. These two datasets have been used for performance comparisons [224, 225]. Supplementary Table S2 contains the number of enzymes and EC numbers per dataset.

4.4.5 Language models embeddings for enzyme sequences

In this study, we used three different pre-trained LLMs as feature extractors of the enzyme amino acid sequences. By extracting the outputs of the last layer of these models and applying an element-wise mean over it, we obtained embeddings for the EC number prediction. The models used as feature extractors were ProtBERT [232], ESM1b [230], and ESM2 [231], trained on protein sequences without label supervision. For details on the pre-processing of protein sequences, refer to Supplementary Information, Figure S1 and Table S3.

Let us denote the output of the transformer's last layer as **M**, a matrix of size $t \times n$, where t represents the number of amino acids and n the dimensionality of the feature representation for each token. In protein LLMs, each token represents an individual amino acid, serving as the smallest data unit processed by the model. Moreover, a special "CLS" token is always added at the beginning of the sequences, and it does not represent any amino acid. For this reason, we exclude it from our calculation. Therefore, we consider the submatrix **M'** of **M**, excluding the first row, represented as:

$$\mathbf{M'} = \begin{bmatrix} m_{1,0} & m_{1,1} & \cdots & m_{1,n} \\ m_{2,0} & m_{2,1} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{t,0} & m_{t,1} & \cdots & m_{t,n} \end{bmatrix} \tag{4.1}$$

where, $\mathbf{M}_{i,j}$ represents the feature value for the j-th dimension of the i-th token. The element-wise average for each dimension across all tokens is then computed as follows:

$$E_j = \frac{1}{(t-1)} \sum_{i=1}^{t} m_{i,j} \quad \text{for } j = 0, 1, ..., n$$

where E_j is the average of the j-th dimension across all tokens. The resulting vector of these averages, E, which is of dimension n, represents the element-wise average of the last layer's output:

$$E = [E_0, E_1, ..., E_n]$$

E provides a single representation that captures the mean features across all tokens in the last layer of the transformer. Seven distinct embeddings were calculated, employing ProtBERT, ESM1b, and five variations of ESM2. The resulting dimensions for the used LLM embeddings are detailed in Supplementary Table S4.

4.4.6 Model architectures and hyperparameters

First, we trained baseline models with each of the embeddings to establish a foundation for evaluating the effectiveness and impact of more complex models. The baseline linear models were shallow networks with only input and output layers. The former has as many neurons as the dimension of the feature vector, and the latter as many neurons as labels (EC numbers to predict) with a sigmoid activation function. The loss function used was binary cross-entropy.

Then, we added hidden layers with the Rectified Linear Unit (ReLU) activation function, followed by batch normalisation, to the baseline models to assess the performance compared with the baselines. In this context, we performed an architecture search of four trials for the seven embeddings. Architectures 1 and 2 featured a single hidden layer with 640 and 2,560 neurons, respectively, while architectures 3 and 4 had two hidden layers, with neuron counts of 640 and 320 for the former and 2,560 and 5,120 for the latter. We trained these networks with the Adam optimizer with a learning rate of 0.001, as after a few preliminary experiments on the training set, we found this value to be the most suitable value for the learning rate. The training process was conducted for 30 epochs. For terminology, these models will be referred to as "DNN", followed by the name of the language model that provided the input embedding. For clarity, models utilising LLM embeddings will henceforth be designated as DNN-LLM models, differentiating them from models that do not employ such embeddings.

Among all the trained models, we chose those with the highest performance on the validation set based on their embeddings and architecture. For example, if architecture 4 delivered the best performance with ESM1b among all tested architectures for DNN ESM1b, we would select it. As ESM2 has five different embedding variations, if the best architecture of DNN ESM2 3B outperformed DNN ESM2 650M, we proceeded with the best architecture of DNN ESM2 3B. These top-performing models were then retrained using training and validation data. Complementarily, we created a model ensemble, a voting classifier consisting of ESM2 3B, ESM1b, and ProtBERT best models. The consensus class prediction for this ensemble is based on hard voting, i.e. the final prediction is determined by a simple majority vote. For terminology, the ensemble of top-performing models is referred to as 'Models ensemble'.

Then, we reimplemented the D-SPACE and DeepEC approaches, as they encode enzymes with one-hot encoding rather than LLM embeddings. Comparing the performance of these models with ours will provide valuable insights into the relative effectiveness of LLM embeddings as encodings of enzyme sequences for EC number prediction. D-SPACE [12] comprises three repeated CNN components and a fully connected network to generate a prediction vector. We reimplemented the EC numbers prediction part, which consists of three CNNs with two layers of 1D convolutions, followed by batch normalisation in each, a max pooling layer at the end, and three fully connected layers and an output layer with as many neurons as the number of labels to predict.

In DeepEC [13], the training process involves three CNNs: CNN-1, which assesses whether a protein is an enzyme; CNN-2, which predicts EC numbers up to the third level; and CNN-3, which extends the prediction to the fourth level. The final EC number prediction is derived when the outputs of CNN-2 and CNN-3 are consistent. However, if either CNN-2 or CNN-3 fail to provide a reliable prediction, DIAMOND

[229] is employed to make the EC prediction. In our revised implementation of DeepEC, the architecture exclusively incorporates the CNN-3 model.

A threshold of 0.5 was used for the output layer to determine positive predictions across all models. By setting the threshold at 0.5, any output probability equal to or greater than this value is classified as positive, while probabilities below it are classified as negative. This approach ensures consistency in evaluation across all models.

All models were implemented and trained using the *pytorch* package v. 2.1.2. Supplementary Section 4 presents more details on the implementation.

4.4.7 Pair-wise alignments

BLAST is designed to compare a query sequence against a database of reference sequences to find regions of similarity. BLAST uses the E-value (Expectation value) as a measure of the significance of a sequence alignment. The E-value is the expected number of random hits with a similar or better score that one would encounter in a database search for the given query sequence purely by chance.

An experiment was conducted using BLASTp to annotate the test datasets and compare the performance with the DL models. We merged the train and validation sets to serve as the reference database for BLASTp. The hit with the lowest E-value was selected, and the EC number was assigned directly from the corresponding sequence. Although we used a fully automated procedure, we acknowledge that results from both BLAST and DL models could be improved with manual annotation. However, this is beyond the scope of this work and would be laborious on a large scale. The BLAST E-value cutoffs were set to 10^{-5}, as suggested by [241] as a typically used value. NCBI BLAST version 2.12.0+ was used. A full list of parameters can be found in Supplementary Table S5. We further created a combined ensemble of the top-performing models and BLASTp, based on hard voting. The system chooses the positive prediction when there is a tie of predictions. In terms of terminology, the ensemble that includes both the top-performing models and BLASTp is called 'Models + BLASTp.'

4.4.8 Metrics

We employed several metrics to evaluate the performance of the models. These metrics provide a comprehensive understanding of the model's predictive capabilities, especially when dealing with multi-labelled datasets. The key metrics are the F1 score macro (mF1), F1 score weighted (wF1), Recall macro (mRecall), Recall weighted (wRecall), Precision macro (mPrecision), Precision weighted (wPrecision) and accuracy.

The mF1 is a metric that calculates each label's harmonic mean of precision and recall and then averages them to obtain a single score for the entire dataset. The same principle is applied to mRecall and mPrecision, where the average of precisions and recalls across labels is computed. Another way of calculating these metrics is using the weighted average version, where the scores of each label are multiplied by the relative frequency of that label in the dataset. These metrics were calculated using the *scikit-learn* package version 1.2.0. Their definitions and formulas can be found in the Supplementary Table S6.

Finally, we evaluated the consistency of the hierarchical predictions of our global multi-label classifier. As a hierarchical system, the EC numbers must be predicted as the following formulation: if a model predicts a more specific label (like 1.1.1.1), it also must predict all the corresponding higher-level labels in the hierarchy (like 1.1.1, 1.1, and 1 in this case). If the prediction aligns with the stated formulation, it is considered consistent; otherwise, it is considered inconsistent. Accordingly, we formulated a metric called Hierarchical Consistency Error (HCE), defined as follows:

$$HCE = \frac{\sum_{i=1}^{T} I(s_i)}{T} \tag{4.2}$$

where T is the number of enzymes in the dataset, and $I(s_i)$ is an inconsistency indicator function for each enzyme s_i, returning one if there is at least one inconsistency in the hierarchical predictions for enzyme s_i, and 0 otherwise. Let $P(s_i)$ be the set of labels predicted for enzyme s_i. For each label l in $P(s_i)$, let $H(l)$ be the set of hierarchical higher-level labels of l. Then, the inconsistency indicator function $I(s_i)$ for enzyme s_i is:

$$I(s_i) = \begin{cases} 1, & \text{if } \exists l \in P(s_i) \text{ such that } H(l) \nsubseteq P(s_i) \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

As EC numbers can be represented as a hierarchical ontology in a directed acyclic graph (DAG), where each node represents an EC, we applied metrics specific to biological hierarchical ontologies. In this definition, level 1 nodes branch into more specific subclasses at level 2, further specified by sub-subclasses at level 3, and finally detailed at a deeper level (level 4), with each level's definition dependent on its predecessor, establishing a cascading specificity from general to precise.

Accordingly, metrics for evaluating how confident each prediction is within an ontology were considered. To interpret these predictions accurately, it is essential to consider the *remaining uncertainty*, *misinformation*, and *S-min* (semantic distance) in relation to a decision threshold. Predictions above or equal to the decision threshold are classified as positive, while those below are negative. These metrics were first described in [242] and implemented in Python in CAFA-evaluator [243]. We adapted this implementation for the EC system.

Remaining uncertainty reflects the information missing from a protein's true annotation that is not captured by the prediction DAG. Formally, the remaining uncertainty is the total information content of nodes in the ontology that exists in the ground truth DAG but is absent in the prediction DAG.

Misinformation reflects the total information content of nodes along incorrect paths in the prediction DAG, quantifying how misleading the predicted annotation is.

Semantic distance or *S-min* helps by combining these two measures and is defined as the minimum distance from the origin to the "remaining uncertainty vs. misinformation" curve. It does not just look at how much information is missing or incorrect individually but balances both to give a single performance score. When semantic distance is low, it indicates that the prediction is both complete (not missing key

information) and accurate (not misleading with incorrect information). This helps rank algorithms in terms of both their accuracy and completeness.

4.4.9 Statistical methods

We utilised the Wilcoxon Signed-Rank (WSR) test to assess the statistical significance of the differences in metric values between the methods across the test sets. In this scenario, the samples comprise the metric values generated by two methods for each EC number within the multi-label classification framework. The goal is to assess the significance of any differences in the median metric values across all EC numbers. In applying the WSR test to evaluate differences in metric values across various identity thresholds and EC numbers, we also incorporated a False Discovery Rate (FDR) correction for multiple hypothesis testing. We applied the Benjamini-Hochberg (BH) FDR correction to the p-values. We deemed a corrected p-value of less than 0.001 sufficient for rejecting the null hypothesis. For more information on how we applied the statistical methods, please refer to Supplementary Information.

4.5 Data and code availability

Code for the model implementation and analysis is available at: `https://github.com/jcapels/ec_numbers_prediction`. P-values of all statistical tests can be found in the Supplementary Information.

4.6 Supplementary information

All the supplementary information can be found in `https://static-content.springer.com/esm/art%3A10.1186%2Fs12859-025-06081-9/MediaObjects/12859_2025_6081_MOESM1_ESM.pdf` or in the Supplementary Information of the published, peer reviewed paper (`https://link.springer.com/article/10.1186/s12859-025-06081-9`).

5

Counted Morgan Fingerprints Best Represent Natural Products

Natural products (NPs) exhibit remarkable structural diversity and are integral to high-value industries, including pharmaceuticals and agricultural biotechnology. This study aims to identify the most effective molecular representation for capturing the unique characteristics of NPs in machine learning tasks. We systematically evaluate several representations, including Biosynfoni, Counted Morgan Fingerprints (FP) (NPClassifierFP), NP_AUX, NPBERT, MHFP, ECFP (MorganFP), and ModernBERT. Our methods involve building datasets for masked learning and exploratory analysis, training BERT models, computing pairwise similarities using cosine and Tanimoto metrics, and evaluating the representations on tasks such as predicting NP structural classes, inferring metabolite-pathway associations, and predicting primary metabolism precursors of plant specialised metabolites. The results indicate that NPClassifierFP performs best overall, excelling in computing biosynthetic distances and predicting biosynthetic relationships and structural classes. We conclude that NPClassifierFP is the most effective representation for natural product machine learning analysis, offering a balanced framework that effectively bridges structural and biosynthetic information.

Keywords: natural products · machine learning · similarity

This chapter was written in collaboration with Dr. Justin J. J. van der Hooft and Pedro Pereira and is currently being prepared for publication.

5.1 Introduction

NP constitute an important class of molecules with high structural diversity. They tend to be more rigid, have more fused ring systems, and have more chiral centres compared to synthetic molecules [244]. Their diversity is the result of the involvement of multiple biosynthetic pathways and tailoring reactions [245]. Their evolutionary diversity and specialisation yield bioactive properties that humans have long exploited in highly profitable industries such as agriculture and pharmaceuticals.

Researchers have sought to computationally capture the distinct diversity of NPs and their divergence from synthetic molecules. Molecular representations for NPs range from binary and count-based fragment

keys to FP and neural continuous-valued embeddings. Molecular fragment keys are 1D vectors where each index represents a predefined substructure, to encode NP features [246]. These keys, often generated via SMARTS pattern-matching algorithms, systematically detect (when they are binary) or quantify (when count-based) substructures within a molecule [247, 248]. While fragment keys rely on expert-defined patterns, FPs automatically extract topological features. In contrast to fragment keys, FPs use a bottom-up approach, where the representation emerges directly from the molecule's topology by systematically traversing the molecular graph, capturing atomic environments, bond connectivity, and other structural motifs without prior bias. Another approach to representing NPs involves neural network–based embeddings, where a Neural Network (NN) is trained to learn dense continuous-valued meaningful molecular representations either in a supervised or self-supervised manner. In supervised learning, the NN is trained to predict specific categorical or continuous endpoints, such as NP class (binary or multi-class classification) or physicochemical properties (via regression). In contrast, self-supervised learning does not rely on labelled data; instead, the model learns to represent molecules by capturing inherent patterns within the input itself.

The Natural Compound Molecular Fingerprint (NC-MFP) [247] was the first structural key to be developed for NPs. This approach represents molecular structures by decomposing them into scaffolds (core ring systems), scaffold-fragment connection points (the atomic positions on a scaffold where the fragments are connected), and fragments (functional group or groups that are chemically bonded to scaffolds). Using a hierarchical scaffold approach to identify the scaffolds, multiple structural levels are generated using the Bemis and Murcko method [249], where larger molecular structures are progressively broken down into simpler core structures. This approach creates a scaffold tree with different levels and prioritises scaffolds belonging to classes in the Dictionary of Natural Products (DNP) [250]. However, the development and re-implementation of NC-MFP are hampered by disconnected processing stages, software dependencies, and the risk of data leakage during the generation of structural keys of connection points and fragments, which are not dataset-agnostic but generated *ad hoc*. Critically, the original implementation computes structural keys using the entire dataset, without respecting train/test splits, thus leaking test-set information into ML models and invalidating performance estimates.

The second-ever-made NP-specific structural key is Biosynfoni [248], a 39-dimensional vector encoding substructures that reflect the biosynthetic building blocks and patterns outlined in Dewick's Medicinal Natural Products (2009) book. This design directly mirrors the logical frameworks of NP biosynthesis, capturing the step-by-step processes and core building blocks described in Dewick's work. Biosynfoni outperforms several widely used FP, such as Morgan, RDKit FPs, and MACCS keys, in capturing biosynthetic similarity and achieves comparable performance in predicting NP classes from ChEBI ontology.

The most commonly used is Morgan FP, which relies on the principles and algorithm of Extended Connectivity FP (ECFP), encoding the chemical environment around each non-hydrogen atom (up to a set radius) into compact bit vectors for ML compatibility. In 2021, NPClassifier [245] introduced a novel method to compute an FP tailored to represent NPs, improving their classification. This method computes a counted, layered Morgan FP that captures the presence and frequency of molecular substructures at different radii. It separates features by radius into distinct segments of the FP vector, allowing each layer to

retain structural context and frequency. All features from individual atoms (radius 0) and larger chemical environments (up to radius 2) are stacked into a single fixed-size vector, capturing both the presence and frequency of substructures in the molecule.

One third example of an FP highlighted for its potential to effectively represent NPs [18] is the MinHashed Fingerprint (MHFP) [251]. This method operates similarly to circular FP, but instead of using atom identifiers, it represents each fragment by its SMILES substring. These fragment identifiers are then encoded into a fixed-size vector using the MinHash algorithm.

NP_AUX and NP_AE, described in [244], are NN-generated molecular embeddings designed for NPs. They are trained on a curated dataset of NPs from COCONUT [252] and synthetic decoys from ZINC [253], selected based on Tanimoto similarity and Natural Product Likeness (NPL) score. The embeddings are extracted from the activations of the last hidden layer of a multi-layer perceptron (MLP) or an autoencoder-like (AE) model, with a 64-dimensional representation encoding NP-relevant features. In their work [244], three neural network architectures were evaluated, all utilising Morgan FP as input: a baseline MLP trained for NP classification, an auxiliary-task MLP (NP_AUX) that simultaneously predicts molecular descriptors and NP classification, and an autoencoder (NP_AE) that reconstructs Morgan FP while performing NP classification. This framework provides a data-driven alternative to traditional handcrafted FP.

As language models (LMs) were gaining traction, NPBERT was developed [254], a transformer-based molecular embedding originally trained on ChEMBL and ZINC data to encode natural products in a 512-dimensional latent space. This embedding was generated based on a masked learning task, where a Bidirectional Encoder Representations from Transformers (BERT) model was trained to predict tokens in a sentence. In this case, tokens are substructures generated by Morgan FP for each atom in a molecule, and a sentence is the concatenation of all the molecular tokens. By learning how to predict masked substructures based on the context, this model learns the interdependence of the molecule substructures. Then, the resulting embedding was used as features to train ML models to predict antimalarial activity [254].

While numerous representations of NPs have been proposed, their comparative effectiveness remains poorly understood in two critical dimensions: computing pairwise similarity between NPs and using such similarity as a proxy for biosynthetic closeness; and identifying the optimal representation for ML tasks involving biosynthetic information prediction and NP classification. To address these gaps, this study conducts a systematic evaluation using two complementary approaches. First, we examine the ability of NP representations to capture biosynthetic similarity by applying two metrics, Tanimoto and Cosine, to pairs of biosynthetically related and unrelated compounds. This includes an exploratory analysis of similarity distributions and cross-representation correlations using a 30,000-NP subsample from LotusDB [255] and COCONUT [252], as well as a focused assessment of their performance in reflecting biosynthetic relationships using PlantCyc pathway data [256]. Second, we benchmark the utility of these representations as features in ML models, comparing their performance against state-of-the-art methods across three tasks: predicting NP structural classes with NPClassifier datasets [245], inferring metabolite-pathway associations from KEGG and PlantCyc, and predicting primary metabolism precursors of plant specialised metabolites using a dataset from our prior work [257]. The ML benchmarking framework is summarised

in Figure 5.1.

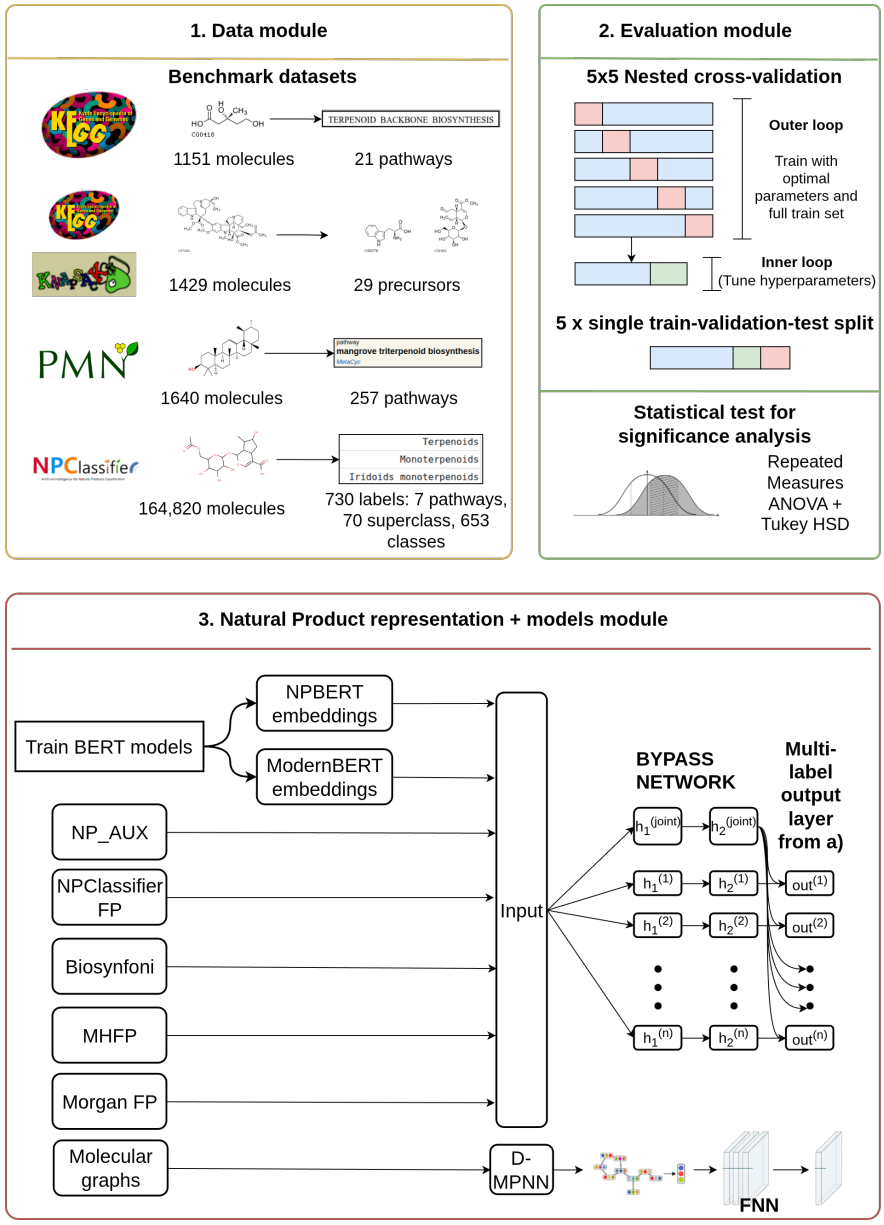


Figure 5.1: Overview of the ML benchmarking framework for natural product (NP) representations. 1) Data module: This section outlines the data sources along with the number of data points in each dataset. Four datasets were used to benchmark NP representations. The first and smallest dataset (1,151 datapoints) includes KEGG plant metabolites. The second dataset is the dataset created in our previous publication [257], which assigns primary metabolism precursors to 1,429 plant specialised metabolites. The third dataset, designed to be more challenging, contains 1,640 metabolites. The fourth dataset is the large-scale NPClassifier dataset, comprising 164,820 compounds annotated with pathway, superclass, and class information. 2) For datasets with fewer than 100,000 datapoints (i.e., the first three), a 5×5 nested cross-validation was employed. For the NPClassifier dataset, which exceeds this size, five model replicates were trained using a fixed train-validation-test split. In the visual representation, blue bars indicate training sets, red bars indicate test sets, and green bars indicate validation sets. To evaluate statistical significance, repeated measures ANOVA was performed, followed by the Tukey HSD post-hoc test. 3) NP representations and models: Each NP representation that produced tabular data (i.e., structured as rows and columns) was used to train a multi-label neural network with a bypass architecture [258]. For graph-based molecular representations, we employed the state-of-the-art Directed Message Passing Neural Network (D-MPNN) [259].

5.2 Results

Our evaluation begins with a qualitative comparison of the NP representations, followed by an exploratory analysis of their similarity distributions and cross-representation correlations. We then assess these representations through two complementary approaches: first, by measuring their ability to capture biosynthetic similarity using pairwise metrics on related and unrelated compounds; second, by benchmarking their utility as machine learning features across three tasks, structural class prediction, metabolite–pathway association inference, and precursor prediction.

5.2.1 Qualitative comparison of natural product representations

A qualitative analysis of the NP representations reveals key characteristics, summarised in Table 5.1.

Biosynfoni encodes biosynthetically relevant fragments based on Dewick's rules, providing a count-based representation that quantifies the occurrence of 39 predefined SMARTS patterns in a compound. This representation offers high interpretability, as each feature directly corresponds to a specific SMARTS pattern, and demonstrates computational efficiency due to its low dimensionality, which reduces model training time and accelerates similarity computations.

NPClassifierFP combines three Morgan FPs with different radii while counting bit collisions. Rather than using binary representation (1 for presence), it records the count of substructures associated with each bit. This count-based approach enhances interpretability, as it builds upon the well-established Morgan FP framework, enabling straightforward tracing of bits back to specific substructures in the compound. However, its high dimensionality (6,144-bit vector) may result in longer training times and slower similarity computations. This also relates to the curse of dimensionality in machine learning, where increasing the number of features can make data sparser, increase computational cost, and require more samples to learn robust and generalisable patterns.

MHFP follows a similar circular fingerprinting approach to NPClassifierFP and MorganFP, but uses SMILES substrings instead of atom identifiers for fragment representation. These substrings are then encoded into fixed-length vectors via MinHash. Like NPClassifierFP, MHFP maintains interpretability as its hash function allows tracing bits back to their corresponding substructures.

NP_AUX, introduced in the NeuralNPFP study [244], is a multi-layer perceptron trained as an auxiliary model to predict molecular descriptors and NP classes. The representation consists of continuous-valued activations from the final hidden layer, extracted after training on the COCONUT database (for NPs) and ZINC compounds (as decoys). The continuous nature of these embeddings prevents the use of Tanimoto similarity, requiring alternative distance metrics. Moreover, the representation's opacity, inherent to NN latent spaces, makes feature interpretation challenging.

NPBERT and other BERT-based models employ transformer architectures pre-trained via masked language modelling. Features are derived from the final layer's activations, yielding continuous-valued representations incompatible with Tanimoto similarity. Among the BERT architectures evaluated, the larger ModernBERT variant was selected for benchmarking due to its superior performance in cross-entropy and

MTRA (see Supplementary Information, Figures S5.2, S5.3 and S5.4). The key distinction lies in their architectural foundations: NPBERT uses the original BERT implementation, while ModernBERT incorporates advancements addressing BERT's limitations in context length, computational efficiency, and data diversity handling. Both models share the same interpretability constraints as NP_AUX due to their neural network-based feature extraction.

Table 5.1: Overview of natural product representations used in this work

Representation	Size / Dimension	Datasets Used (originally)	Category	Type	Source
Biosynfoni	39	Manually curated biosynthetic fragments based on Dewick's logic	Substructure	Count	[248]
NPClassifier FP	6,144	N/A	Circular	Count	[245]
NP_AUX	64	COCONUT + ZINC decoys	NN	Continuous	[244]
NPBERT	512	COCONUT + LotusDB	NN	Continuous	[254]
MinHashed (MHFP)	1,024	N/A	String	Categorical	[251]
Morgan FP	2048	N/A	Circular	Binary	[260]
ModernBERT	1,024	COCONUT + LotusDB	NN	Continuous	N/A

5.2.2 Analysing the distribution and correlations of representation similarities

We evaluated NP representations through correlation analysis and similarity distributions to characterise inter-representation relationships and assess their ability to capture NP chemical space.

Pearson correlations between cosine similarity measures ($n = 30{,}000$) reveal predictable relationships among circular FP (Figure 5.2A). MorganFP and NPClassifierFP exhibit high correlation ($r = 0.70$, cosine; $r = 0.87$, Tanimoto, in Figure 5.2A and B), sharing 49% and 76% of variance depending on the metric. NPClassifierFP's extended radii and explicit counting of overlapping substructures account for the divergence. MHFP shows similarly strong Tanimoto correlations with MorganFP ($r = 0.82$), explaining 67.24% shared variance. Both employ circular topology, though they differ fundamentally: MHFP uses canonicalised SMILES and MinHash hashing in 2^{32}-dimensional space, while MorganFP applies deterministic modular arithmetic hashing. Notably, both expand circular substructures from heavy atoms; their differences lie in fragment encoding and hashing (canonicalised SMILES shingles with MinHash for MHFP versus modular-arithmetic hashing of atomic invariants for MorganFP) rather than in atom coverage.

NPClassifierFP emerges as a particularly effective representation, consistently achieving the highest average correlation across all tested representations ($r = 0.69$ with MHFP, Tanimoto). Even against biosynthesis-aware FP (Biosynfoni), NPClassifierFP shows the strongest correlation ($r = 0.57$ Tanimoto; $r = 0.42$ cosine), suggesting it captures biosynthetically relevant features while maintaining circular topology, a notable result given the algorithms' fundamental differences.

NP_AUX correlates most strongly with MorganFP's cosine similarity ($r = 0.39$), accounting for 15% shared variance, reflecting its training protocol's incorporation of MorganFP features. Interestingly, ModernBERT and NPBERT show substantial correlation in cosine similarities ($r = 0.72$), despite different tokenisation strategies. ModernBERT exhibits near-zero correlations with most representations but shows modest correlation with NP_AUX ($r = 0.20$).

Binary/count representations (NPClassifierFP, Biosynfoni, MorganFP) yield cosine similarities in $[0, 1]$, while real-valued representations (NP_AUX, NPBERT, ModernBERT) span $[-1, 1]$. Within their overlapping range, interpretations remain consistent. Cosine similarity distributions (Figure 5.2C) reveal distinct patterns across representation types. NPClassifierFP and NP_AUX exhibit the broadest and most informative distributions (medians of 0.37 and 0.26; 75th percentiles of 0.51 and 0.46), providing good discrimination across the chemical space. In contrast, MorganFP shows sharp concentration near 0.15 (median = 0.14, 75th percentile = 0.22) due to extreme sparsity (≈2% active bits), which compresses the similarity distribution. NPClassifierFP, though count-based, displays heavier tails with frequent high values (>0.60), even though most pairwise similarities remain modest (median 0.37, 75th percentile 0.51). Biosynfoni, despite low dimensionality (39 dimensions) and moderate sparsity (23% active bits), achieves high average counts per molecule (82), causing high-count dimensions to dominate cosine similarity calculations. This produces a top-heavy distribution (median = 0.84, first quartile = 0.70). Neural embeddings from NPBERT and ModernBERT cluster near 1, showing limited separation despite the dataset containing no duplicates.

Tanimoto similarity distributions (Figure 5.2D) offer different characteristics for count-based representations. Biosynfoni shows the broadest distribution (median = 0.37, 75th percentile = 0.49), while the hashed MHFP and binary MorganFP concentrate sharply (90th percentiles at 0.05 and 0.15 respectively), reflecting their high dimensionality where random overlaps become improbable. NPClassifierFP shares MorganFP's median (0.10) but displays heavier tails, demonstrating how its count-based nature affects similarity calculations.

Overall, the choice of similarity metric interacts substantially with representation type: cosine similarity performs well for count-based representations with appropriate distributions (NPClassifierFP) but becomes problematic for extremely sparse binary FP and dense neural embeddings. Tanimoto is better suited for count-based representations, focusing on absolute feature overlaps rather than proportional patterns.

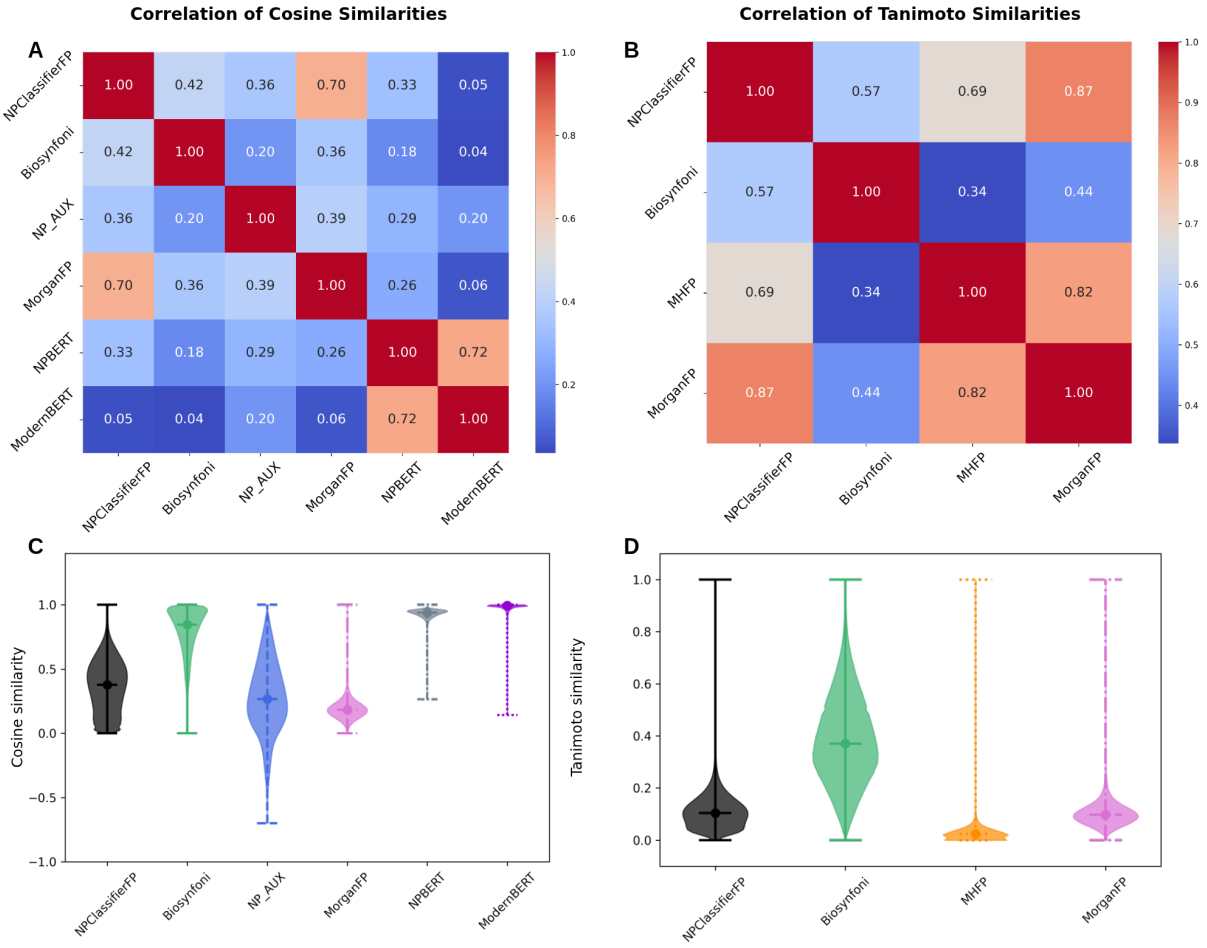


Figure 5.2: A) A pairwise correlation matrix of cosine similarities comparing all NP representations analysed in this study using the NP dataset. B) A pairwise correlation matrix of Tanimoto similarities. C) Distribution of cosine similarity scores for each NP representation, calculated across all pairwise comparisons in the NP dataset. The violin plots illustrate the distribution percentiles, with a circle marking the median similarity. D) Distribution of Tanimoto similarity scores for each NP representation

5.2.3 NPClassifierFP and Biosynfoni excel in biosynthetic distance computation

Next, we aimed to evaluate whether chemical similarities correlate with biosynthetic relationships. We analysed cosine and Tanimoto similarity distributions across varying biosynthetic distances, defined here as the number of reaction steps separating two compounds within a biosynthetic pathway (see Figure 5.3A). Figure 5.3B shows that only NPClassifierFP and NP_AUX exhibit monotonic decreases in cosine similarity with increasing biosynthetic separation. These representations effectively discriminate between closely related (high similarity), distantly related (moderate similarity), and unrelated (low similarity) compound pairs, unlike other FP where random pairs maintain artificially high similarities (e.g., Biosynfoni and NPBERT), consistent with patterns observed in Figure 5.2C. Notably, NPClassifierFP scores exceed MorganFP's while remaining below other FP for closely related pairs.

Figure 5.3C shows that Biosynfoni's Tanimoto similarities effectively serve as a biosynthetic similarity proxy, showing a gradual decline towards zero with increasing biosynthetic distance, though less abruptly than other representations, while maintaining the expected distribution for random pairs (consistent with Figure 5.2D). These findings replicate Nollen et al.'s results [248], as our study and theirs also relied on PlantCyc data to compute biosynthetic distance. Critically, Biosynfoni was designed specifically as a proxy for biosynthetic similarity, unlike other representations tested. NPClassifierFP shows marginal improvement over MorganFP, particularly at biosynthetic distances ≥6, where its similarity distribution shifts slightly closer to 1. MHFP performs the worst among the tested representations, with nearly its entire similarity distribution concentrated below the 0.5 threshold, indicating limited capacity to resolve biosynthetic relationships.

These results demonstrate that NPClassifierFP and NP_AUX cosine similarities, alongside Biosynfoni Tanimoto similarities, effectively serve as quantitative proxies for biosynthetic relatedness.

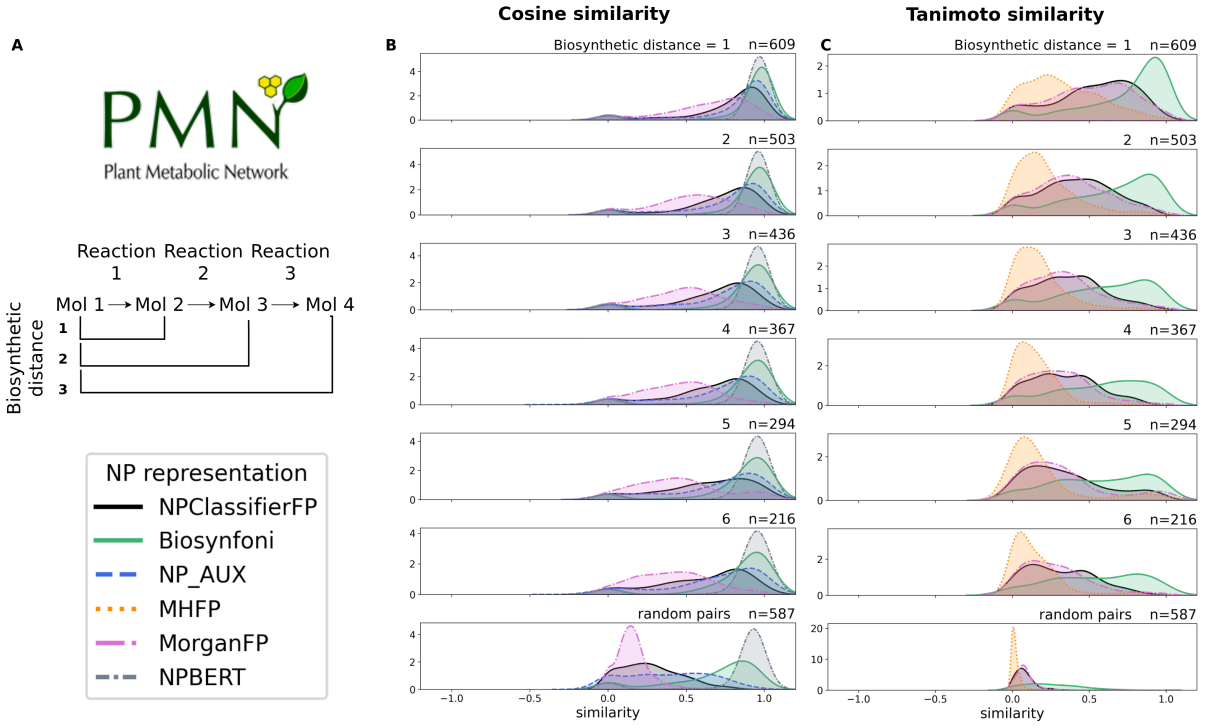


Figure 5.3: A) Illustration of the calculation of biosynthetic distance. By tracing compounds through successive biosynthetic steps, we generated pairs at escalating distances and calculated their corresponding similarity. B) Cosine similarity between pairs of compounds in relation to biosynthetic distance. The degree of representation overlap diminishes in proportion to the biosynthetic divergence between compounds — fewer shared reactions correlate with lower similarity scores. C) Tanimoto similarity between pairs of compounds in relation to biosynthetic distance. The sample size per step is indicated by n.

5.2.4 NPClassifierFP most consistently predicts biosynthetic relationships

To evaluate biosynthetic relationship predictions for plant secondary metabolites, we implemented an AutoML pipeline in DeepMol featuring multi-label neural network classification (see Methods, section 5.5.6) with representation-specific hyperparameter and architecture optimisation. Here, we used three datasets: (1) metabolite-pathway relationships sourced from PlantCyc, (2) metabolite-pathway associations sourced from KEGG, and (3) primary metabolism precursors of specialised metabolites (first three datasets in Figure 5.1). For statistical validation, we applied a repeated measures ANOVA followed by Tukey's HSD *post hoc* test. While a Levene test revealed heterogeneity of variances for some metrics (rejecting the null hypothesis; Supplementary Information, Tables S5.1-S5.3), ANOVA remains robust to moderate violations of this assumption, particularly when group sizes are equal, and the ratio of largest to smallest variance does not exceed 9 [261]. Additionally, we assessed the normality of performance metric distributions. Given the limited sample size, we prioritised visual inspection of distribution shapes (Supplementary Information, Figures S5.5-S5.7) over formal normality tests (e.g., Kolmogorov-Smirnov), as the former offers greater robustness for small datasets [262]. We then evaluated how well each representation serves as features for linear (PCA) versus non-linear (t-SNE) projections in an unsupervised setting. This analysis provides insight into the linearity of the classification problem for each feature set, which we further evaluated by comparing the performance of linear and non-linear supervised models.

5.2.4.1 KEGG metabolite-pathways associations

We set out to assess the predictive power of a multi-label classification model trained with NP representations for KEGG metabolite-pathway associations. Each metabolite's SMILES representation $\mathbf{s}_i$ is mapped to a multi-hot label vector $\mathbf{y}_i \in \{0,1\}^L$, whose ℓ-th entry $y_{i\ell} = 1$ when metabolite i is annotated to KEGG pathway ℓ, that is, when it appears as a metabolite (reactant or product) in one of the reactions constituting that pathway, and 0 otherwise. Since a single metabolite can participate in several pathways simultaneously, multiple entries of $\mathbf{y}_i$ may be active at once, which motivates a multi-label rather than multi-class formulation. For example, quercetin is labelled $y_{i\ell} = 1$ for flavonoid biosynthesis and 0 for diterpenoid biosynthesis, giving a vector of the form $\mathbf{y}_i = [1, 0, \dots]$.

The KEGG pathway prediction performance analysis (Table 5.2 and Supplementary Figure S5.8) reveals distinct patterns across evaluation metrics. The representations are organised by overall performance (highest to lowest), with pairwise performance differences and statistical significance (indicated by asterisks) presented in each cell of the heatmaps. Compared with the representation-based models evaluated here, NPClassifierFP achieves the highest macro F1 score (mF1), indicating the strongest balance between precision and recall. The external state-of-the-art baseline, Baranwal2024 [263], uses a DL architecture based on graph convolutional networks to extract molecular shape features, which are then provided to a random forest classifier. Although the Baranwal2024 outperformed all the other representations, it did not present a statistically significant difference in mF1 performance when compared to

D-MPNN, MorganFP and NPClassifierFP. When examining individual metrics, MorganFP demonstrates superior macro precision (mPrecision) while D-MPNN achieves the highest macro recall (mRecall) among all tested representations. The heatmap comparisons show that while D-MPNN exhibits numerically higher mRecall values than other representations, these differences fail to reach statistical significance when compared to both MorganFP and NPClassifierFP. Conversely, NPClassifierFP and MorganFP display statistically significant improvements in mPrecision over D-MPNN, which contributes to their overall superior mF1 performance. This pattern suggests that NPClassifierFP's strength lies in its balanced performance across both precision and recall metrics, while D-MPNN's advantage in recall does not translate to significantly better overall classification performance.

Table 5.2: KEGG results expressed in averages ± standard deviation

Method	mF1	mPrecision	mRecall
Baranwal 2024	**0.807 ± 0.02**	0.841 ± 0.022	**0.792 ± 0.028**
NPClassifierFP	0.803 ± 0.019	0.854 ± 0.02	0.777 ± 0.027
MorganFP	0.800 ± 0.03	**0.865 ± 0.036**	0.763 ± 0.033
D-MPNN	0.788 ± 0.03	0.814 ± 0.035	0.785 ± 0.036
MHFP	0.772 ± 0.033	0.853 ± 0.043	0.736 ± 0.039
NP_AUX	0.765 ± 0.038	0.799 ± 0.04	0.755 ± 0.047
ModernBERT	0.764 ± 0.032	0.848 ± 0.034	0.719 ± 0.035
Biosynfoni	0.761 ± 0.03	0.798 ± 0.036	0.748 ± 0.036
NPBERT	0.715 ± 0.03	0.771 ± 0.03	0.687 ± 0.034

The analysis shows that NPBERT performed the worst across all metrics. ModernBERT, another BERT-based model, ranked third-to-last in mF1 score, performing comparably to NP_AUX and MHFP. While ModernBERT achieved a relatively higher mPrecision, placing fourth, it did so at the cost of mRecall, where it ranked second-to-last, indicating a tendency to produce false negatives. Biosynfoni also performed poorly, placing second-to-last in both mF1 and mPrecision and fourth-worst in mRecall. MHFP followed a similar pattern, ranking fourth in mF1, third in mPrecision, but struggling in mRecall. Biosynfoni only outperformed the BERT-based models and MHFP in mRecall. These findings suggest that Biosynfoni's biosynthesis-aware design does not improve pathway prediction, as its encoding may not effectively capture the chemical features critical for KEGG pathway classification. Neural embeddings (NPBERT, ModernBERT, and NP_AUX) likewise showed limited effectiveness for this task.

Weighted averages of performance metrics revealed similar trends (Supplementary Information, Figure S5.9), with NPClassifierFP matching the state-of-the-art performance of Baranwal2024 [263] and jointly leading in wF1 score. The most notable shift was that NP_AUX here ranked second-to-last in wF1,

indicating that while it outperformed other methods on average, its performance degraded for the most frequent labels.

The t-SNE projection in Figure 5.4 demonstrates NPClassifierFP's enhanced pathway resolution capability, revealing well-defined clusters for porphyrin, carotenoid, isoquinoline and indole alkaloids, steroids, and diterpenoid biosynthesis. A notable feature is the shared chemical space among phenylpropanoid pathway derivatives, including flavonoids, isoflavonoids, flavone and flavonol biosynthesis, indicating NPClassifierFP's ability to encode both pathway-specific chemical patterns and broader biosynthetic relationships. This visualisation suggests the representation effectively captures biologically meaningful metabolic organisation that extends beyond simple structural similarity, correlating with its superior performance in pathway prediction tasks.

However, when analysing the PCA in Figure 5.4, it is clear that not as many clusters are formed. Particularly, we see a clear (linear) separation between compounds from carotenoid and diterpenoid biosynthesis and the others. However, the overlap between most compounds from different pathways is evident, suggesting that the decision boundary between pathways exhibits non-linear behaviour, as the classes are not separable by linear projections. To confirm this, we trained and optimised a ridge classifier (L2-regularised least-squares classification) in the same terms as the bypass NN (see Methods, Subsection 5.5.8) and applied a t-test for the statistical analysis of differences. Results showed that all methods differed in terms of mF1 score ($^{*}p - value < 0.05$), favouring the non-linear NN, except for ModernBERT, which favoured the ridge classifier, and MHFP and NPBERT, whose differences were not significant (see Supplementary Information, Table S5.4).

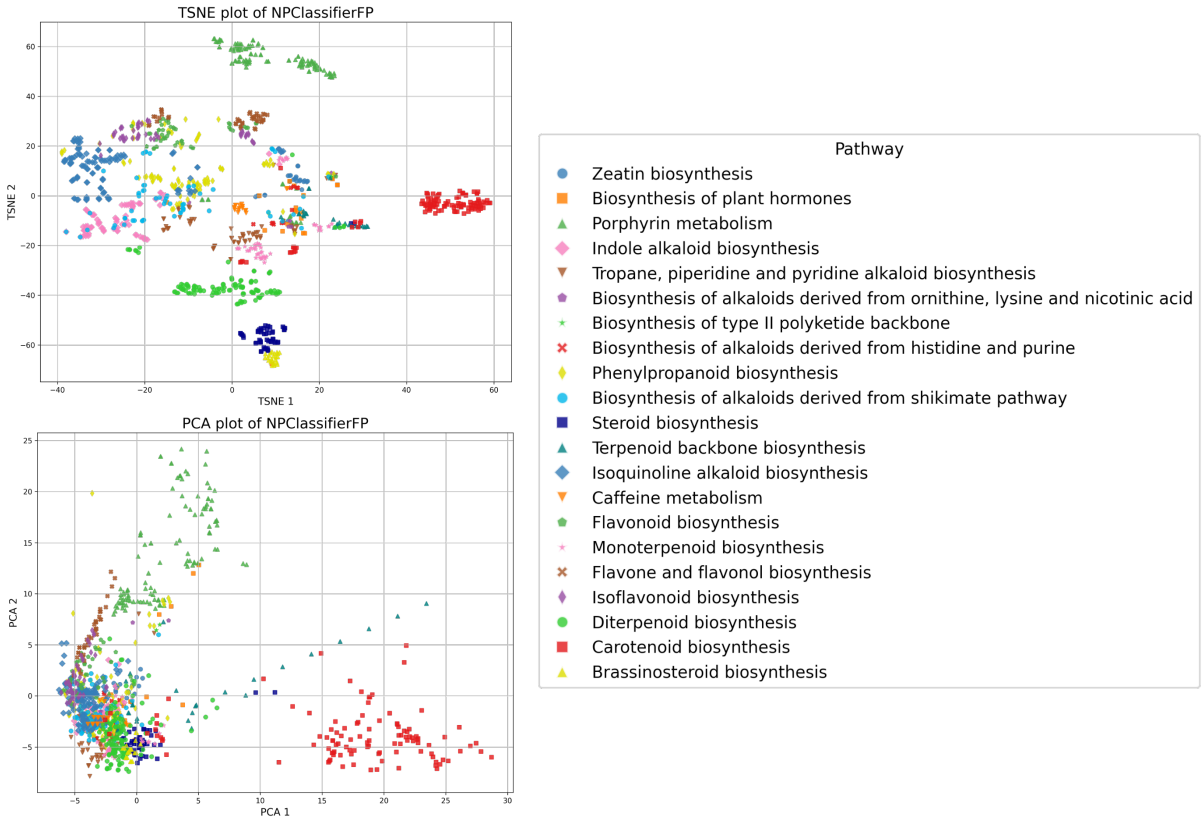


Figure 5.4: t-SNE and PCA projections of NPClassifierFP embeddings with KEGG pathways.

5.2.4.2 PlantCyc metabolite-pathways associations

We set out to assess the predictive power of a multi-label classification model trained with NP representations for PlantCyc metabolite-pathway associations, where each metabolite's SMILES representation $\mathbf{s}_i$ is linked to a binary vector $\mathbf{y}_i \in \{0,1\}^L$ encoding its involvement in L distinct PlantCyc pathways (e.g., quercetin as $[1,0,1,\dots]$ for methylquercetin biosynthesis, α-tomatine biosynthesis, flavonol glucosylation I, etc.).

The PlantCyc pathway prediction analysis (Table 5.3 and Supplementary Figure S5.10) reveals consistent patterns across metrics, unlike the KEGG results, with lower performance overall (e.g., mF1 scores ranging from 0.255 to 0.602). This suggests that the task is inherently harder, likely due to finer pathway granularity and sparse label representation (e.g., pathways with as few as 10 compounds assigned).

NPClassifierFP achieves the highest mF1 and mRecall, while MorganFP leads in mPrecision, though neither difference is statistically significant. Both methods outperform the current state-of-the-art (Baranwal2024 [263]) across all metrics, again without statistical significance. Notably, NPClassifierFP's strength lies in minimising false negatives, as reflected in its superior mRecall. Among the remaining methods, performance varies by metric: MHFP excels in mPrecision and mF1, D-MPNN achieves the highest mRecall, and NP_AUX lags behind both. The weakest representations, Biosynfoni, NPBERT, and ModernBERT, rank third-to-last, second-to-last, and last, respectively. While Biosynfoni consistently approaches the performance of the next-higher method, albeit with statistically significant gaps, ModernBERT remains the

Table 5.3: PlantCyc metabolite-pathways association average performances ± standard deviations

Method	mF1	mPrecision	mRecall
NPClassifierFP	**0.602 ± 0.02**	0.613 ± 0.02	**0.634 ± 0.025**
MorganFP	0.599 ± 0.023	**0.614 ± 0.021**	0.625 ± 0.028
Baranwal 2024	0.589 ± 0.03	0.599 ± 0.033	0.621 ± 0.03
MHFP	0.497 ± 0.023	0.524 ± 0.022	0.509 ± 0.026
D-MPNN	0.496 ± 0.025	0.503 ± 0.026	0.538 ± 0.028
NP_AUX	0.467 ± 0.021	0.476 ± 0.025	0.503 ± 0.023
Biosynfoni	0.436 ± 0.026	0.444 ± 0.025	0.472 ± 0.032
NPBERT	0.358 ± 0.025	0.387 ± 0.026	0.36 ± 0.026
ModernBERT	0.255 ± 0.016	0.292 ± 0.015	0.241 ± 0.017

least effective, posting an mF1 of just 0.255.

The weighted performance metrics (Supplementary Figure S5.11) retain the same ranking order but exhibit modestly higher values (e.g., the top wF1 score reaches 0.646 compared to 0.602 for mF1).

Figure 5.5 presents a t-SNE projection of NPClassifierFP, illustrating only the 30 most represented PlantCyc pathways to avoid overcrowding from the full set of 257 pathways. The visualisation reveals varying degrees of overlap between compounds from different biosynthetic pathways, though many still cluster by chemical class or shared biosynthetic origin. On the far left, a distinct group corresponds to glucosinolate biosynthesis, encompassing pathways like “aliphatic glucosinolate biosynthesis, side chain elongation cycle” and “glucosinolate biosynthesis from (di)homomethionine”. Near the top, compounds linked to sterol biosynthesis are concentrated, while the right region features metabolites from phenylpropanoid biosynthesis, including those involved in anthocyanin, flavonoid, and quercetin production. Meanwhile, the central area contains a dense cluster of alkaloid-related compounds, spanning pathways such as ajmaline, sarpagine, vindoline, vindorosine, vinblastine, morphine, Amaryllidaceae alkaloid, and taxol biosynthesis. In contrast, other pathways, such as those for secologanine and strictosidine biosynthesis or glucosinolate biosynthesis from tryptophan, lack clear spatial cohesion, with their compounds dispersed among unrelated clusters. This scattered distribution, alongside the partial overlap of biosynthetically related metabolites, emphasises the challenges of disentangling pathway-specific signals in this dataset.

The PCA projection in Figure 5.5 reveals limited cluster formation, with substantial overlap between compounds from different pathways. This suggests that pathway discrimination relies on non-linear decision boundaries, as linear projections fail to separate the classes effectively. To validate this hypothesis, we trained and optimised a ridge classifier under the same conditions as the bypass NN and conducted

paired t-tests to assess statistical differences in mF1 scores. The results confirmed that all methods, except ModernBERT, performed significantly better ($^*p < 0.05$) with the non-linear NN compared to the ridge classifier. ModernBERT was the sole exception, favouring the linear model, while MHFP showed no significant difference between the two approaches. (See Supplementary Information, Table S5.5)

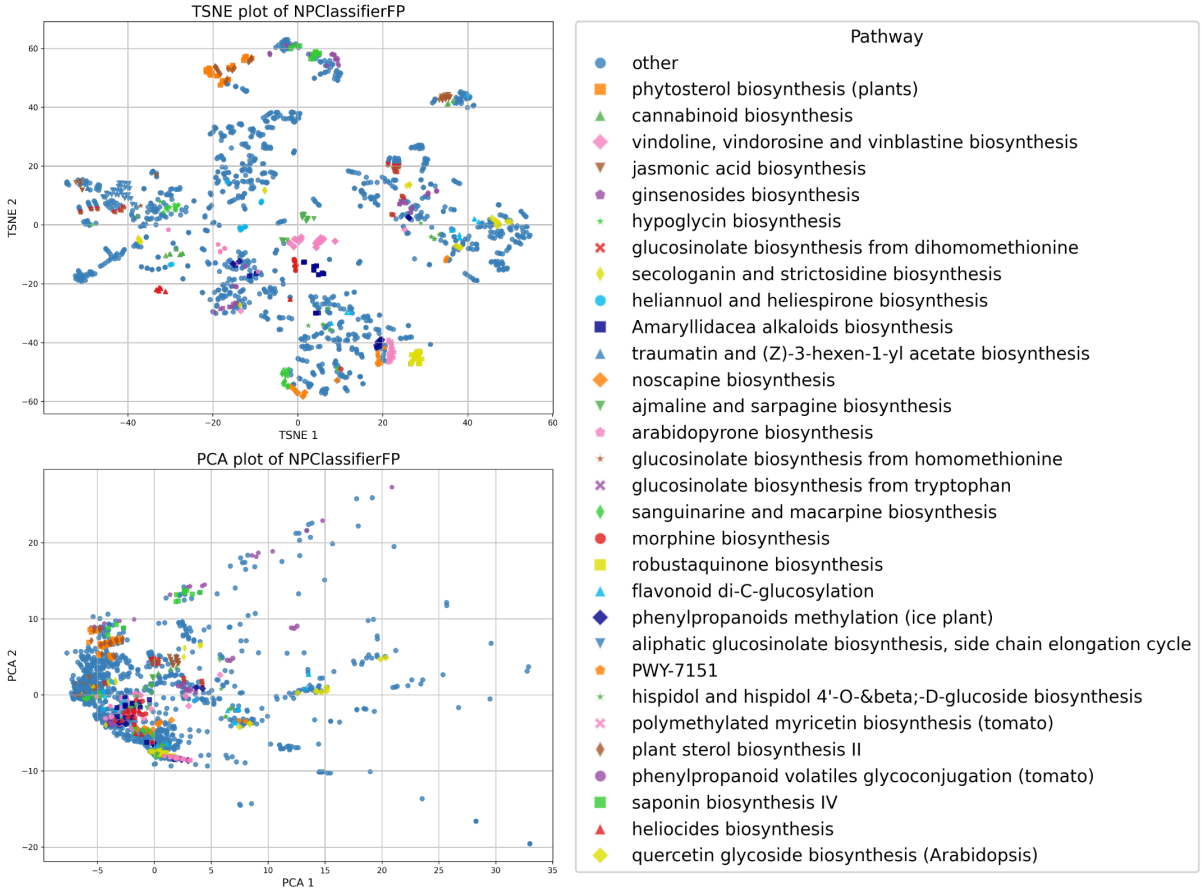


Figure 5.5: t-SNE and PCA projections of NPClassifierFP with the 30 more represented PlantCyc pathways.

5.2.5 Predicting precursors of plant specialised metabolites

Here, we evaluate a multi-label classification model that leverages NP representations to predict biosynthetic precursors of plant specialised metabolites, mapping each metabolite's SMILES string $\mathbf{s}_i$ to a binary vector $\mathbf{y}_i \in \{0,1\}^L$ indicating the utilisation of L candidate precursors (e.g., secologanin $[1]$ and tryptophan $[1]$ for vinblastine, but not phenylalanine $[0]$).

The use of ML to predict precursors of plant specialised metabolites was first proposed by [264] and later expanded upon, and outperformed, by our method from a previous publication [257]. Table 5.4 shows the performance results for this particular task. Moreover, Supplementary Figure S5.12 presents performance differentials between methods as heatmaps, demonstrating that our approach from a previous work [257] significantly outperforms all others, except for MorganFP in mPrecision, where no statistically significant difference was observed.

Table 5.4: Performance of all methods in predicting the precursors of plant specialised metabolites

Method	mF1	mPrecision	mRecall
Capela2025	**0.913 ± 0.017**	**0.945 ± 0.019**	**0.895 ± 0.019**
D-MPNN	0.878 ± 0.019	0.906 ± 0.021	0.87 ± 0.02
NPClassifierFP	0.861 ± 0.025	0.918 ± 0.029	0.832 ± 0.03
MorganFP	0.855 ± 0.024	0.924 ± 0.03	0.816 ± 0.026
MHFP	0.842 ± 0.028	0.918 ± 0.028	0.805 ± 0.037
NP_AUX	0.795 ± 0.023	0.836 ± 0.033	0.783 ± 0.027
Biosynfoni	0.739 ± 0.031	0.79 ± 0.033	0.727 ± 0.035
ModernBERT	0.722 ± 0.022	0.848 ± 0.03	0.663 ± 0.021
NPBERT	0.714 ± 0.025	0.807 ± 0.035	0.674 ± 0.028

Among the representations in the present benchmark, D-MPNN achieved the highest mF1 score overall, though its performance was statistically not different from NPClassifierFP. For mPrecision, D-MPNN ranked fifth but showed no significant difference from methods up to second place. NPClassifierFP performed slightly better in this metric, on par with MHFP and marginally below MorganFP. In terms of mRecall, D-MPNN significantly outperformed all other representations except the state-of-the-art. While NPClassifierFP showed a slight edge over MorganFP, this difference was not statistically significant, yet it still surpassed all remaining methods.

The bottom ranks are occupied by NPBERT, ModernBERT, and Biosynfoni. Among these, Biosynfoni distinguishes itself with the highest mF1 and mRecall scores, though it lags in mPrecision. No statistically significant differences were observed between ModernBERT and NPBERT.

Figure 5.6 shows t-SNE and PCA projections of NPClassifierFP embeddings, where precursors are highlighted using distinct colours and markers. In the t-SNE projection, terpenoid precursors, such as geranylgeranyl diphosphate, farnesyl diphosphate, geranyl diphosphate, and campesterol, form tight clusters in the centre and bottom-right regions. Meanwhile, phenylpropanoid precursors, including naringenin, delphinidin, pelargonidin, cyanidin, cinnamoyl-CoA, p-coumaryl-CoA, and apigenin, predominantly cluster on the left side of the plot. In contrast, alkaloid precursors appear widely dispersed, consistent with their biochemical diversity, often involving multiple precursors from varied biosynthetic origins [257]. This dataset was expected to be more straightforward to classify than pathway-prediction tasks, as the t-SNE projection demonstrates that unsupervised, non-linear methods can inherently group molecules by precursor similarity, indicating clear underlying structural patterns.

In contrast, the PCA projection in Figure 5.6 reveals far fewer distinct clusters. While a linear separation emerges between compounds derived from geranylgeranyl diphosphate/farnesyl diphosphate and

other precursors, most pathway-specific compounds overlap substantially. This overlap suggests that pathway classification relies on non-linear decision boundaries, as linear projections fail to separate the classes effectively. To validate this observation, we trained and optimised a ridge classifier under identical conditions as the bypass NN and conducted a t-test to compare performance. The results confirmed significant differences in F1 scores ($p < 0.05$) across all methods, consistently favouring the non-linear NN, except for ModernBERT, where the ridge classifier performed better, and MHFP and MorganFP, which showed no statistically significant difference (see Supplementary Information, Table S5.6).

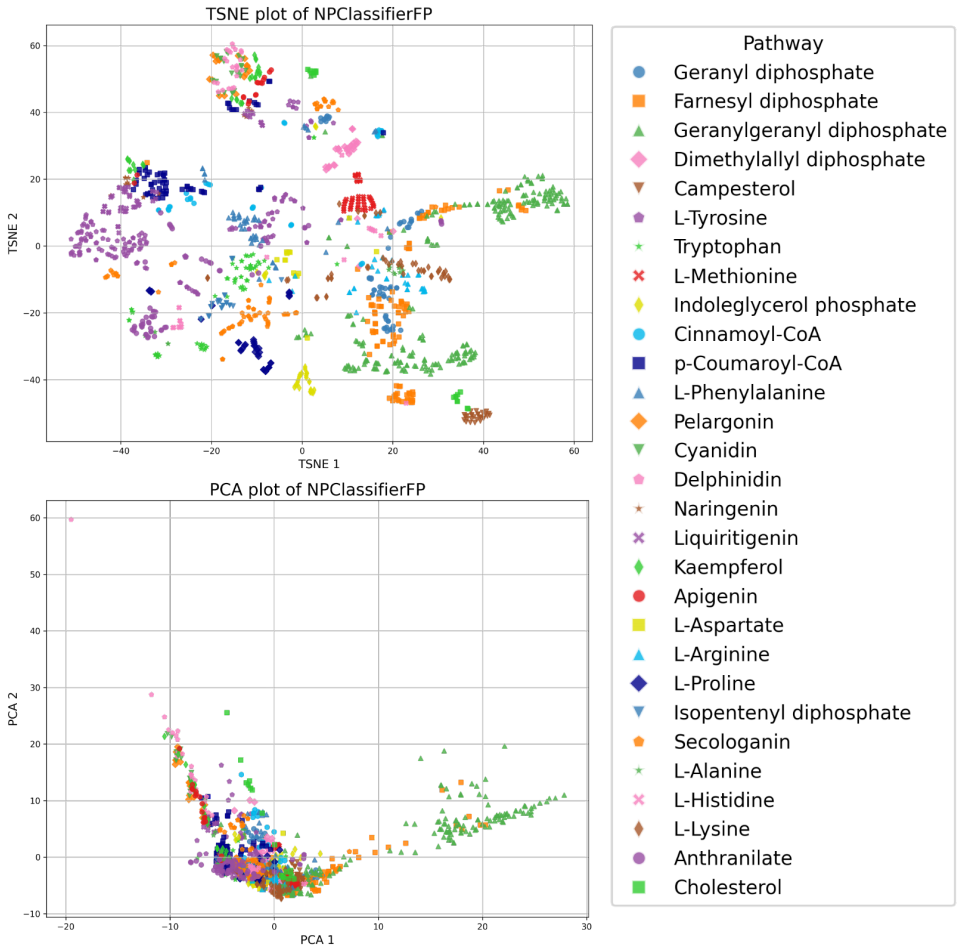


Figure 5.6: t-SNE and PCA projections of NPClassifierFP with all precursors highlighted.

5.2.6 NPClassifierFP most consistently represents NP classes

Quantitative comparisons across our NP dataset from NPClassifier [245] demonstrate that the original NPClassifier method maintains superior performance in class representation tasks, though the proposed NPClassifierFP variant for the benchmark is competitive and shows better results than other evaluated methods. Table 5.5 presents general mF1 and discriminated by pathway, superclass, and class-level performance, with state-of-the-art baselines (NPClassifier model and Ginestra [265]) grouped below the horizontal line for reference. NPClassifierFP demonstrates particularly strong performance among the

tested representations, achieving an mF1 of 0.931 that closely approaches the state-of-the-art. Its perfect pathway classification (1.000) matches the top baselines, while its superclass (0.959) and class (0.927) scores nearly equal those of dedicated classification approaches. D-MPNN and MorganFP similarly show robust performance, with mF1 scores of 0.921 and 0.925, respectively, and comparable class-level results (0.916 and 0.922). This narrow performance gap, just 0.5% in mF1 between NPClassifierFP and the leading baseline (which uses NPClassifierFP but trains a different NN), highlights how architectural modifications can have less impact than the representation used.

The comparison reveals that while other representations like MHFP (0.792 mF1) and transformer-based approaches like ModernBERT (0.760 mF1) struggle with fine-grained discrimination, modern graph-based and MorganFP-based representations consistently close the gap to state-of-the-art performance, as also shown by [265]. The minimal 0.2% difference in class-level mF1 between NPClassifierFP (0.927) and the top baseline that does not use NPClassifierFP as input (0.929) suggests that representation quality has become the primary determinant of classification performance. These findings demonstrate that the most effective representations now perform within 1-2% of state-of-the-art baselines.

Table 5.5: mF1 scores for each of the tasks

Method	mF1	mF1 pathway	mF1 superclass	mF1 class
NPClassifierFP	0.931 ± 0.002	1.000 ± 0.000	0.959 ± 0.001	0.927 ± 0.002
D-MPNN	0.921 ± 0.002	1.000 ± 0.000	0.957 ± 0.002	0.916 ± 0.002
MHFP	0.792 ± 0.003	1.000 ± 0.000	0.507 ± 0.011	0.830 ± 0.003
NP_AUX	0.786 ± 0.003	0.968 ± 0.020	0.858 ± 0.005	0.776 ± 0.004
MorganFP	0.925 ± 0.001	1.000 ± 0.000	0.948 ± 0.001	0.922 ± 0.001
Biosynfoni	0.793 ± 0.002	0.986 ± 0.019	0.884 ± 0.006	0.782 ± 0.002
ModernBERT	0.760 ± 0.013	1.000 ± 0.000	0.851 ± 0.009	0.749 ± 0.013
NPBERT	0.832 ± 0.003	1.000 ± 0.000	0.904 ± 0.003	0.823 ± 0.004
NPClassifier (original)	**0.936 ± 0.002**	0.995 ± 0.010	**0.963 ± 0.003**	**0.933 ± 0.002**
Ginestra	0.933 ± 0.002	0.988 ± 0.001	0.961 ± 0.003	0.929 ± 0.001

5.3 Discussion

The results of this study collectively identify NPClassifierFP as the most effective representation for NP similarity searches and ML tasks associated with biosynthesis and class prediction. Across multiple benchmarks, NPClassifierFP demonstrates a unique ability to balance interpretability, computational efficiency,

and predictive performance. Its count-based circular fingerprint design addresses key limitations of alternative methods: unlike binary FP such as MorganFP, which suffer from excessive sparsity (only 2% active bits on average), NPClassifierFP's substructure counting preserves discriminative information while maintaining compatibility with established similarity metrics. Simultaneously, it avoids the interpretability challenges and artificially inflated similarity scores observed in neural embeddings like NPBERT and ModernBERT, where dense, high-dimensional spaces obscure meaningful chemical relationships.

The representation's strength is particularly evident in its consistent performance across diverse tasks, from capturing biosynthetic similarity (Figure 5.3) to excelling in metabolite-pathway relationship prediction (Figure S5.8 and Table 5.2; Figure S5.10 and Table 5.3), biosynthetic precursors prediction (Figure S5.12 and Table 5.4) and class classification (Table 5.5). The monotonic decline in similarity with increasing biosynthetic distance (Figure 5.3B-C) further confirms its utility as a proxy for biosynthetic relatedness, a property shared only with NP_AUX and Biosynfoni but with greater statistical robustness and broader translation to predicting biosynthetic information.

The comparative analysis reveals that the effectiveness of NP representations is inherently task-dependent, with no single alternative matching NPClassifierFP's versatility. Neural embeddings, though theoretically powerful, underperform in practice, as seen in ModernBERT's preference for linear classifiers, suggesting its features lack the non-linear complexity required for biosynthetic tasks. Biosynfoni, despite its biosynthetic grounding, fails to translate its advantage in capturing biosynthetic distance through Tanimoto similarity into metabolite-pathway relationship prediction performance, highlighting a critical distinction between similarity capture and predictive utility. Even graph-based methods like D-MPNN, which achieve high recall in some tasks, lack the precision and biosynthetic specificity of NPClassifierFP. These patterns underscore that representation quality, rather than model architecture, is the primary determinant of performance, as evidenced by the minimal performance gaps between NPClassifierFP and state-of-the-art baselines that leverage the same underlying features.

The relationship between representations and biosynthetic endpoints exhibits varying degrees of linearity, depending on the task and metric employed. PCA and ridge classifier experiments demonstrate that pathway prediction overwhelmingly relies on non-linear decision boundaries, with linear methods failing to separate overlapping pathway clusters. However, the clear linear separation of structurally distinct precursor classes in PCA (e.g., terpenoid vs. alkaloid precursors) suggests that certain coarse-grained biosynthetic relationships may be linearly encodable. This duality, where broad chemical classes exhibit linear separability while fine-grained pathway distinctions require non-linear models, aligns with the observation that NPClassifierFP's count-based features support both linear and non-linear separability. The superiority of non-linear NN across most tasks ($p < 0.05$) further reinforces that biosynthetic prediction demands non-linear complex feature interactions, which NPClassifierFP's hybrid design effectively captures. Meanwhile, the choice of similarity metric plays a pivotal role: cosine similarity proves uninformative for binary FPs but remains effective for NPClassifierFP, whereas Tanimoto similarity excels for biosynthetic analyses both for NPClassifierFP and Biosynfoni but fails for neural embeddings lacking binary features.

5.4 Conclusion

This study establishes the counted Morgan-based NPClassifierFP as the optimal representation for natural product ML-based analysis, offering a unified framework that bridges structural and biosynthetic information more effectively than existing alternatives. Its success across similarity searches, pathway prediction, biosynthetic precursors and class classification tasks establishes its status as an established baseline for NP modelling. The broader implications of this work extend to drug discovery and synthetic biology, where accurate representation of NP chemical space is critical for predicting bioactivity and designing biosynthetic routes. By addressing the trade-offs between interpretability, performance, and biological relevance, NPClassifierFP provides a foundation for advancing ML applications in NP science.

From a practical standpoint, these findings suggest that NPClassifierFP should be the go-to benchmark representation for NP machine learning, particularly in applications requiring both interpretability and high predictive accuracy. Its near-state-of-the-art performance in class prediction (mF1 = 0.931), competitive pathway resolution (on par with the state of the art for KEGG pathways and outperforming it for PlantCyc pathways), and top-tier biosynthetic precursor prediction (statistically lagging only behind the state of the art) together support this recommendation. However, limitations remain, particularly in potential plant-specific biases that may not generalise to microbial NPs. Future work could explore dimensionality reduction techniques to address computational efficiency while preserving biosynthetic signals, or hybrid approaches combining NPClassifierFP's structural features with neural embeddings to improve rare pathway prediction, such as the approach explored in [265].

5.5 Methods

5.5.1 Natural products representations

The NP representations selected for this comparative analysis include Biosynfoni [248]; NP_AUX and NP_AE (both part of the same method implemented in *DeepMol*) [244], also known as NeuralNPFP; NPBERT [254]; and the Counted Morgan FPs (further referred to as NPClassifierFP) [245]. In addition to these, we explored alternative representations by training language models, ModernBERT, BERT, and DeBERTa, to extract molecular embeddings, and by employing both the Morgan fingerprint (radius 2) and the MinHashed fingerprint (MHFP), the latter having been highlighted as a particularly promising approach for NP representation [18, 251]. Finally, we utilised molecular graph representations as input for training a Directed Message Passing Neural Network (D-MPNN).

5.5.2 Building the datasets for masked learning and exploratory analysis

As masked learning is a form of self-supervised learning, it does not require labelled data. Therefore, we integrated the most comprehensive NP resources available: COCONUT (version 10-2024) [252], containing approximately 695,000 compounds, and LotusDB (February 2021 release) [255], with around 276,000 compounds.

After downloading both databases, we applied a standardisation protocol based on the ChEMBL standardisation routine [266], implemented in the *DeepMol* package [267]. We then generated InChIKeys for each compound and removed duplicates based on the first 26 characters of the InChIKey. The final character, which encodes the protonation state, was ignored to ensure that different protonation forms of the same compound were treated as duplicates.

To prevent data leakage, where training data influences test performance results, we excluded all compounds that appeared in the datasets used for downstream ML tasks (i.e. predicting NP classes, pathways and metabolic precursors). The final dataset was composed of 801,284 compounds. The remaining data were partitioned into a single train-validation-test split using scaffold splitting, which ensures that compounds sharing the same core scaffold are placed within the same subset. To further reduce the risk of data leakage through structural similarity, we removed any compound from the test set that had a Tanimoto similarity greater than 60% (calculated using Morgan FP with radius = 2) to any compound in the training set.

5.5.3 Training BERT models with masked learning

The NPBERT, ModernBERT, BERT, and DeBERTa models were pre-trained using a masked language modelling (MLM) objective, adhering to the standard BERT architecture. These models were trained on SMILES strings sourced from NP databases, as specified in the previous section.

NPBERT was trained according to the parameters and architecture in its original publication [254]. As for ModernBERT, BERT, and DeBERTa, during the training process, 20% of the tokens in each SMILES string were randomly masked. The models were then optimised to predict the original tokens based on their surrounding context. This approach enables models to learn contextual embeddings that capture substructural dependencies and syntactic patterns inherent to NPs. The loss function used was the cross-entropy. The model was trained for 10 epochs. For each model, we evaluated two architectural configurations: a smaller variant (256 hidden size, 8 layers, 8 attention heads) and a larger variant (1,024 hidden size, 24 layers, 16 attention heads). The best-performing configuration, selected via a masked learning test set performance (see subsection 5.5.2), was used for benchmarking.

5.5.4 Similarity calculation

To explore similarity distributions and cross-representation correlations, we computed pairwise similarities for each representation type using a sub-sample of the preprocessed COCONUT and LotusDB databases. Evaluating the full datasets was computationally infeasible, as it would require calculating over 10 billion similarity values. To mitigate this computational burden, we adopted the afore-mentioned sampling strategy in which 30,000 randomly selected NPs were used to compute pairwise similarities. Although the sampling was random, it was ensured that the distribution of NP classes was kept, see Supplementary Figure S5.1.

The resulting similarity values were compiled into a matrix $S \in \mathbb{R}^{f \times n}$, where $n = \binom{30000}{2} =$

$449{,}985{,}000$ denotes the number of unique compound pairs, and f represents the number of representation types. From S, we derived summary statistics (mean, standard deviation, median, and percentiles) for each fingerprint type's pairwise similarity distribution.

Subsequently, we computed the Pearson correlation (r) matrix of S, denoted as $C \in \mathbb{R}^{f \times f}$, to assess how similarly the different representation types characterise compound relationships. This matrix captures the degree to which each representation correlates with others in terms of their similarity profiles across the random batch selected.

To quantify the similarity between molecular representations, we computed *cosine* and *counted Tanimoto similarities* between pairs of representation vectors. Cosine similarity is a metric particularly well-suited for comparing high-dimensional vectors that may differ in scale or numerical domain, such as binary, count-based, or continuous-valued molecular representations. It measures the cosine of the angle between two vectors, effectively capturing their orientation while being invariant to their magnitude.

Given two vectors $\vec{x}$ and $\vec{y}$, cosine similarity is defined as:

$$\text{cosine\_similarity}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \, \|\vec{y}\|}$$

If either vector is a zero vector (i.e., its L2 norm is zero), the similarity is undefined and is treated as a missing value (NaN) in our implementation.

By focusing on vector orientation rather than magnitude, cosine similarity allows for a fair and consistent comparison across molecular FP with differing value distributions and scales. Importantly, this metric yields similarity values in $[-1, 1]$ in general, and in $[0, 1]$ for non-negative representation types such as binary and count-based FP, ensuring that the results are directly comparable across binary, count-based, and real-valued encodings. This normalisation of similarity scores across representations supports meaningful benchmarking and model evaluation under a unified similarity framework. Accordingly, cosine similarity was used uniformly across all representation types in our analysis.

The counted Tanimoto similarity was also considered, though its application was restricted to discrete (count-based or binary) representations, specifically for NPClassifierFP, Biosynfoni, MHFP, and MorganFP. For two discrete or binary vectors x and y, the counted Tanimoto similarity is defined as:

$$\text{counted Tanimoto similarity}(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)},$$

where the numerator $\sum_i \min(x_i, y_i)$ represents the sum of element-wise minima (intersection), and the denominator $\sum_i \max(x_i, y_i)$ represents the sum of element-wise maxima (union). This formula only has mathematical meaning for non-negative, discrete values, as it uses "min" and "max" operations. Negative or continuous values would distort the interpretation of "intersection" and "union".

5.5.5 Computing biosynthetic distance

The motivation for quantifying molecular similarity as a function of biosynthetic distance is to evaluate how well a natural product representation captures underlying biosynthetic relationships, hereafter referred to

as biosynthetic similarity.

Biosynthetic distance refers to the degree of separation between two NPs in a biosynthetic reaction chain, reflecting how many enzymatic or structural transformations are required to convert one into the other. A smaller biosynthetic distance indicates a closer biosynthetic relationship.

To assess biosynthetic distance, we followed the protocol of Biosynfoni [248], where we extracted reaction chains from PlantCyc pathways [256]. It was assumed that all compounds were NPs. For each selected pathway, we identified the longest sequence of consecutive reactions, yielding a total of 113 reaction chains. From each chain, we generated reaction pairs in the format $(x, x+1), (x, x+2), \ldots, (x, x+n)$, where n corresponds to the number of reactions separating the two compounds.

5.5.6 Building the datasets for multi-label classification

The multi-label classification framework employed in this study addresses three distinct prediction tasks in NP research, each utilising datasets prepared according to task-specific requirements (Figure 5.11) but sharing a common mathematical foundation. For all prediction tasks, we consider datasets $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{y}_i)\}_{i=1}^{N}$ where each SMILES string $\mathbf{s}_i$ is associated with binary labels $\mathbf{y}_i \in \{0,1\}^L$ indicating: (1) chemical class membership (e.g., morphine labelled as $[1, 0, 0, ...]$ for [alkaloid, terpenoid, fatty acid]), (2) biosynthetic precursor usage (e.g., vinblastine derived from secologanin $[1]$ and tryptophan $[1]$ but not phenylalanine $[0]$), or (3) pathway involvement (e.g., quercetin participating in flavonoid biosynthesis $[1]$ but not diterpenoid biosynthesis $[0]$).

The NP classifier dataset was downloaded from the GitHub repository `https://github.com/mwang87/NP-Classifier`. The dataset was preprocessed to encode the labels into a matrix of 0s and 1s as in a multi-label classification setting. Given the large dataset size, we adopted the single train-validation-test split from the NPClassifier code, deeming it statistically robust for evaluation. Cross-validation was omitted, as the data volume (with L chemical classes in the NPClassifier hierarchy) ensures reliable performance estimation.

The dataset for the prediction of precursors from the primary metabolism of plant specialised metabolites was downloaded from the repository of the original publication [257] in `https://github.com/jcapels/SMPrecursorPredictor.git`. The dataset was already prepared with the labels in a matrix of 0s and 1s, for use in a multi-label classification setting. In this case, L represents the number of precursor molecules from primary metabolism considered in the analysis.

The dataset for the prediction of the involvement of metabolites in specific pathways from KEGG and PlantCyc (metabolic pathway prediction/metabolite-pathway associations) encompassed the usage of biopython's KEGG API and the development of a parser for PlantCyc 15.1.1 data files, respectively. In this case, we generated datasets annotating compound-pathway associations by evaluating reaction participation in KEGG and PlantCyc (v15.1.1). Specifically, a compound was assigned to a pathway if it appeared as a reactant, product, or cofactor in any pathway-defined reaction. Here, L equals the total number of pathways from KEGG and PlantCyc 15.1.1 databases.

For KEGG, we read the HTML in `https://www.genome.jp/dbget-bin/www_bget?pathway+`

`map01060` related to the pathways of biosynthesis of plant secondary metabolites. This HTML page lists all the secondary metabolism pathways of plants in KEGG. Next, we assessed the structural information of all the metabolites involved in each pathway. We then eliminated pathways that we considered to be primary metabolism, such as 'Glycolysis / Gluconeogenesis', 'Citrate cycle (TCA cycle)', 'Pentose phosphate pathway', 'Arginine biosynthesis', 'Purine metabolism', 'Starch and sucrose metabolism', 'Amino sugar and nucleotide sugar metabolism'. Afterwards, we carefully discarded compounds that were excessively involved and that were not secondary metabolites but rather solvents or cofactors, such as water or ATP. Finally, we discarded compounds that had fewer than 7 hydrogen atoms as first proposed by [268]. In the end, we encoded all the labels and converted them into a matrix of 0s and 1s.

As for PlantCyc data, we developed a parser for the ".dat" raw files for pathways, compounds and PlantCyc pathways ontology. Next, we transformed the pathways' ontology into a tree structure with the Python package "anytree". This tree was used to select all the pathways below the node "SECONDARY-METABOLITE-BIOSYNTHESIS". Upon getting the list of pathways of interest, we selected and extracted the SMILES string of all the metabolites involved in those pathways. Then, similarly to the approach employed for KEGG data, we discarded solvents or cofactors. Also, we removed precursors of secondary metabolites such as amino-acids, GPP, GGPP, FPP, among others. Afterwards, we discarded compounds that had fewer than 7 hydrogens as first proposed by [268], just like we did for the KEGG dataset. In the end, we encoded all labels into a matrix of 0s and 1s.

Finally, we generated InChIKeys for each compound in both datasets and removed duplicates based on the first 14 characters of the InChIKey to ensure chemical uniqueness and avoid redundancy. This approach was chosen because the first 14-character block of an InChIKey encodes the molecular skeleton (atoms and connectivity), providing a fixed-length identifier that enables consistent structural comparison across datasets. Using only this segment balances specificity and efficiency, as it captures core structural information while excluding variable layers such as stereochemical, isotopic or tautomeric differences that most representations cannot capture, therefore being less relevant for most analyses. By removing duplicates in this manner, we ensured that the final datasets contained only structurally distinct compounds, reducing bias from overrepresented scaffolds and improving the reliability of subsequent analyses.

5.5.7 Data splitting

Regarding the NPClassifier dataset, we retained the original single train-validation-test split as specified in the authors' code. As this dataset encompasses over 100,000 data points, we reckon that it is sufficient to maintain statistical robustness without applying resampling techniques [262]. To balance computational efficiency with robustness, we forwent cross-validation and instead trained five model replicates using distinct random seeds (Figure 5.1). This involved shuffling the training data during each training process and randomly initialising the model weights.

On the PlantCyc, KEGG, and plant precursors datasets, we applied a 5×5 nested cross-validation approach to ensure robust model evaluation (Figure 5.12). In nested cross-validation, the dataset is divided twice, in an outer loop and an inner loop. The outer loop is used for evaluating the model's performance,

while the inner loop is used for hyperparameter tuning. Specifically, the dataset is first split into $k = 5$ folds in the outer loop. For each iteration of the outer loop, one fold is held out as the test set, and the remaining $k - 1$ folds are used for the inner loop. Within the inner loop, the data is further split into a single train-validation split for hyperparameter optimisation, as defining folds (as averaging across folds provides a more reliable estimate of hyperparameter performance) would increase the running time by a factor proportional to the number of inner folds, thereby significantly escalating computational costs. This process ensures that the model is trained and validated on different subsets of the data, providing a more reliable estimate of its performance.

5.5.8 Supervised model training

In benchmarking different NP representations, we identified the bypass NN architecture [258] as the most suitable choice for tabular data (i.e., all representations except molecular graphs). This architecture strikes a balance between multitask and progressive networks by combining shared layers for learning general representations with task-specific branches that bypass these shared layers (See Figure 5.13). Unlike progressive networks, the task-specific pathways in bypass networks are independent and non-interacting, avoiding both memory growth and the need for task ordering. This design enables both efficient parameter sharing and flexible task-specific modelling, making it well-suited for diverse predictive tasks. We used binary cross-entropy (BCE) loss to train the models in a multi-label classification setting that is defined as follows:

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{C}\sum_{c=1}^{C}\left(y_{ic}\log(x_{ic}) + (1-y_{ic})\log(1-x_{ic})\right)\right] \tag{5.1}$$

where x_{ic} is the predicted probability for compound i for label c and y_{ic} is the ground truth. C is the total number of labels and N the number of compounds.

For each hyperparameter optimisation loop, the bypass network was tuned for dropout rate, the number and size of fully connected layers, batch size, and number of training epochs. In contrast, for molecular graph representations, we used the D-MPNN [259], optimising the message-passing depth, number of fully connected layers, dropout rate, batch size, and number of training epochs. Hyperparameter tuning for both models was performed using the Tree-structured Parzen Estimator (TPE) algorithm from Optuna, with each search running for 20 trials. All experiments were conducted using the DeepMol framework [267].

5.5.9 Model performance evaluation

For MLM, we used the cross-entropy loss function to train the model and assess its performance over 10 epochs. In addition, we included an additional performance metric to better interpret the results. We applied masked token recovery accuracy (MTRA), defined as follows:

$$\text{MTRA} = \frac{\sum_{i,j}[y_i^{(j)} = \hat{y}_i^{(j)}] \cdot [\text{mask}_i^{(j)}]}{\sum_{i,j}[\text{mask}_i^{(j)}]} \tag{5.2}$$

where $y_i^{(j)}$ represents the ground truth token at position j for compound i, $\hat{y}_i^{(j)}$, the predicted token at position j for compound i, and $\text{mask}_i^{(j)}$ is the binary indicator (1 if the token at position j for compound i is masked, 0 otherwise).

As for the supervised learning tasks, we applied metrics that were specific to multi-label classification, such as macro and weighted F1 (mF1 and wF1), macro and weighted recall (mRecall and wRecall), and macro and weighted precision (mPrecision and wPrecision), defined as follows:

$$\text{mF1} = \frac{1}{C}\sum_{i=1}^{C}\frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{5.3}$$

$$\text{wF1} = \sum_{i=1}^{C} w_i \cdot \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{5.4}$$

where C is the total number of labels, Precision_i is the precision for label i, Recall_i is the recall for label i, and w_i is the weight for label i, calculated as the proportion of compounds in label i relative to the total number of compounds.

$$\text{mRecall} = \frac{1}{C}\sum_{i=1}^{C}\frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i} \tag{5.5}$$

$$\text{mPrecision} = \frac{1}{C}\sum_{i=1}^{C}\frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i} \tag{5.6}$$

$$\text{wRecall} = \sum_{i=1}^{C} w_i \cdot \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i} \tag{5.7}$$

$$\text{wPrecision} = \sum_{i=1}^{C} w_i \cdot \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i} \tag{5.8}$$

where w_i is the weight for label i, calculated as the proportion of compounds in label i relative to the total number of compounds.

To evaluate the statistical significance of our results, we used a repeated measures ANOVA followed by Tukey's HSD post hoc test, chosen for their suitability to our experimental design. Because we compared more than two groups (five replicates or k folds), ANOVA was appropriate for testing significant differences in mean performance, while Tukey's HSD controlled the family-wise error rate across all pairwise comparisons, ensuring rigorous protection against Type I errors. For the NPClassifier dataset, the five replicates, generated using distinct random seeds, are independent, as each involves unique data shuffling and model initialisation. For the PlantCyc, KEGG, and plant precursor datasets, the outer folds

of nested cross-validation yield mutually non-overlapping test sets, and hyperparameter tuning in the inner loop does not influence the outer test sets. We note, however, that the corresponding training sets partially overlap across folds, so the per-fold estimates are not fully independent, a known limitation of significance testing on cross-validation results [269]. Although ANOVA assumes normality, it is robust to mild deviations from this assumption, particularly with balanced group sizes [261]; given the small number of replicates per group, we rely on this robustness rather than on the Central Limit Theorem.

5.6 Data and code availability

All the code and data can be found in the NP fingerprint benchmark repository.

6

Predicting Enzyme-substrate Specificity

Enzymes catalyse biochemical reactions with remarkable specificity and efficiency, yet predicting enzyme–substrate interactions remains challenging due to limited experimental data, incomplete annotations, and the scarcity of validated non-interacting pairs. Traditional computational approaches, such as docking and molecular dynamics, are resource-intensive and poorly scalable, while many ML models struggle to generalise to chemically diverse or previously unseen molecules. We present a novel ML framework that integrates experimentally validated enzyme–substrate pairs with advanced feature representations. Our approach combines protein embeddings from protein language models fine-tuned for EC number prediction with NPClassifierFP molecular fingerprints to capture functional and structural properties of enzymes and substrates. An Extreme Gradient Boosting (XGBoost) classifier is trained using rigorous data partitioning with the GraphPart method to prevent data leakage and ensure robust generalisation across similarity thresholds. Benchmarking against state-of-the-art models (ESP, ProSmith) and similarity-based approaches (BLASTp + Tanimoto similarity) demonstrates that our method consistently outperforms existing approaches, achieving high F1 scores and MCC. Notably, the model maintains strong predictive performance under stringent protein (40–80% sequence identity) and compound (20–80% Tanimoto similarity) similarity constraints, and generalises well to chemically distinct, previously unseen molecules.

Keywords: enzymes · machine learning · substrates

This chapter was written in collaboration with Ana Monteiro and is currently being prepared for publication.

6.1 Introduction

Enzymes, meticulously shaped by evolution for biochemical catalysis, achieve extraordinary rate accelerations (often $> 10^6$-fold vs. spontaneous reactions) by stabilising transition states through precise active-site interactions with substrates [49]. While typically substrate-specific, many enzymes exhibit promiscuity toward structurally related compounds, including substrates that the enzymes did not evolve to catalyse [26]. This catalytic efficiency stems from dynamic active-site conformations that optimise transition-state binding [270].

Understanding enzyme-substrate relationships is fundamental to biochemistry, particularly for applications such as metabolic pathway elucidation in natural and engineered systems [271]. However, establishing these relationships remains challenging because substrate specificity data in public databases are limited, and experimental validation is costly and time-consuming [272]. This challenge is further compounded by incomplete EC annotations, a hierarchical classification system where enzymes are categorised by their catalysed reactions (e.g., EC 1.2.3.4, with the fourth digit indicating substrate specificity), in which this fourth-level specificity is often missing.

Computational approaches have emerged to address these limitations. Physics-based methods (e.g., docking and molecular dynamics) model binding interactions at atomistic, mechanistic detail [16], but require curated 3D structures, extensive computational resources and long run times [17], hindering high-throughput applications.

ML alternatives offer promising solutions potentially reducing experimental trial-and-error attempts by suggesting leads and addressing incomplete EC annotations. Current ML Enzyme-Substrate Interaction (ESI) approaches fall into two categories: (1) enzyme family-specific models trained on comprehensive datasets, and (2) general models covering diverse enzyme families. The latter typically generate negative training examples through random sampling due to the scarcity of experimentally validated non-binding pairs.

Mou et al. [273] developed RF, gradient-boosted decision trees, SVM, and logistic regression models to predict bacterial nitrilase substrate scope, using 3D-structure and active-site features from experimentally validated enzyme-small molecule combinations. Yang et al. [274] took a similar approach for plant glycosyltransferases, generating training data through low-throughput experimental screening of 107 Arabidopsis GT1 enzymes against UDP-d-glucose, two nucleophilic acceptors, and distinct sugar donors/acceptors, training a decision tree-based model. Meanwhile, Goldman et al. [275] compiled high-quality, literature-curated datasets capturing multi-substrate activity profiles for enzyme families.

Recent advances in general models include the ESP model [276], which combines gradient boosting with two representations: fine-tuned Evolutionary Scale Modelling (ESM)-1b embeddings for protein sequences, and graph neural network-derived vectors for molecular features, trained on GO-database-confirmed enzyme-substrate pairs and augmented negatives (molecules similar to known substrates but labelled as non-substrates). However, ESP struggled to generalise to unseen enzymes or molecules. To address these shortcomings, [277] later introduced ProSmith, a multimodal transformer that jointly processes protein and small-molecule representations. ProSmith integrates gradient-boosted predictions from the transformer with independent representations (ESM-1b for proteins, ChemBERTa2 for molecules) via a 6-layer attention network using concatenated embeddings. By expanding the training data to include phylogenetic evidence alongside experimental annotations, and retaining the same negative-sampling strategy, ProSmith achieved superior performance over ESP.

More recently, the MEI model [278] employed cross-attention to integrate molecule and enzyme embeddings, trained on positive pairs from the RHEA database [279] and negative cases generated via a biocatalysis AI model [280], alongside high-throughput data (phosphatases/transaminases) and ESP's training set. While it outperforms ESP and other compound-protein interaction models, reproducibility

is hindered by unmaintained code and restricted data access. While the model is validated on external datasets, including oxidoreductases, hydrolases, lyases, esterases, and DUF849 proteins with sequence similarities below 60% (and excluding shared substrates), data leakage to the training set remains a critical concern in ML-based enzyme-substrate specificity prediction. Even with these precautions, risks persist due to structurally similar compounds in the training data, which can inadvertently introduce bias and overestimate predictive performance.

This study addresses two key limitations in current ESI approaches: (i) the poor generalizability of ML models to chemically distinct unseen molecules, and (ii) the persistent scarcity of experimentally validated non-binding pairs, a challenge explicitly acknowledged in state-of-the-art literature.

6.2 Results

6.2.1 Dataset and splits

The partitioning of compound and protein datasets based on structural and sequence similarity thresholds is a critical step in ensuring rigorous model evaluation. By systematically varying similarity thresholds (20%, 40%, 60%, and 80% for compounds) and identity thresholds (40%, 60%, and 80% for proteins), we analysed their impact on dataset composition and cross-partition similarity. The results, presented in Supplementary Tables S6.1 and S6.2 and visualised in Supplementary Figures S6.1 and S6.2, reveal key insights into how threshold selection influences dataset distribution and the independence of training, validation, and test sets.

As similarity thresholds increase, training set size decreases from 45,647 to 34,895 compounds (without stereochemistry) and from 66,213 to 50,731 (with stereochemistry), while test set size increases from 6,198 to 11,267 and from 6,723 to 15,004, respectively. This pattern reflects stricter partitioning, and the inclusion of stereochemical information consistently increases dataset size.

Protein datasets show similar partitioning trends: at 40% identity, training sets contain 39,048/54,690 proteins (without or with stereochemistry) with test sets of 8,907/12,577; at 80%, these shift to 36,467/51,159 (training) and 10,428/14,725 (test). Lower thresholds enforce greater dissimilarity between training and test sets and therefore provide the stricter tests of generalisation, whereas higher thresholds permit more cross-partition similarity while yielding larger and more reliably enforced test partitions.

Supplementary Figure S6.1 shows similarity distributions across partitions. At 20% threshold, Train-Test median similarity ($\sim$0.6) indicates substantial structural/sequence overlap. GraphPart initially allowed compound leakage at 20% and 40% thresholds, i.e. compound pairs with similarity above these thresholds were allowed to be in different partitions; while 40% leakage could be prevented by excluding compounds with Tanimoto $>$40% to training molecules, lowering this threshold to 20% would empty the test set, making it impractical.

At higher thresholds (40%, 60%, 80%), median Train-Test similarities increase ($\sim$0.4, $\sim$0.5, $\sim$0.6-0.7), as thresholds define maximum allowed cross-partition similarity (the 40% threshold permits $\leq$40% similar compounds/proteins in different sets, and the 80% threshold allows $\leq$80%).

Protein splits show tight identity distributions around thresholds (Supplementary Figure S6.2), with increasing thresholds (40%-80%) yielding higher median similarity and dispersion, particularly at 60%-80% where broader distributions indicate substantial cross-partition sequence overlap, demonstrating reduced partition stringency.

6.2.2 XGBoost-NPClassifierFP variants outperform the state of the art

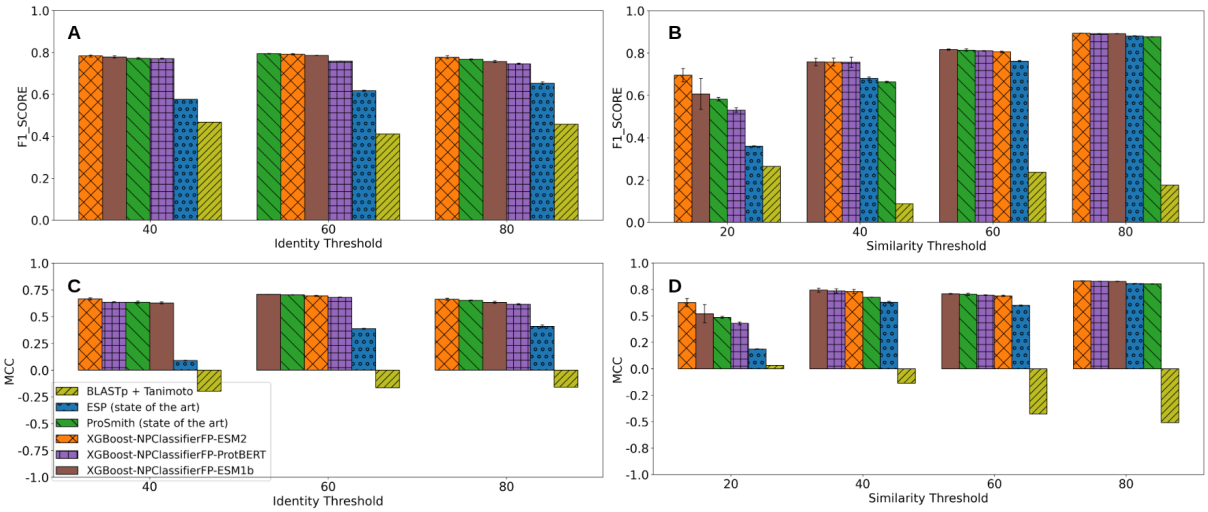


Figure 6.1: Comparison of the performance of various ML models in predicting proteins and compounds across different identity and similarity thresholds. **A** and **B** charts display the F1 Scores for protein identity and compound similarity, respectively.The **C** and **D** charts present the Matthews Correlation Coefficient (MCC) for the same thresholds. Error bars indicate the standard deviation of model performance.

Figure 6.1 compares multiple modelling strategies across a comprehensive panel of evaluation metrics that account only for the positive cases and another that account for both (top and bottom panels, respectively), while being both statistically robust for unbalanced data. Clear and systematic performance differences emerge between traditional similarity-based approaches and ML models built on protein language model embeddings.

Performance remains consistent across different protein identity splits, indicating that varying similarity thresholds between training and test sets have minimal impact on the relative ranking of methods. This robustness suggests that the learned representations capture functional signals that generalise beyond close homologues. Furthermore, larger pre-trained models tend to yield better downstream performance. In particular, embeddings derived from ESM2 fine-tuned for EC number prediction outperform the other variants across most similarity and identity thresholds, the main exceptions being the 40% and 60% compound-similarity splits, where ESM1b takes the lead (discussed below). Nonetheless, the overall trend indicates that increased model capacity and task-aligned pretraining translate into improved predictive performance.

At the 40% and 60% compound-similarity thresholds, XGBoost-NPClassifierFP-ESM1b outperformed all other methods, including its ESM2 equivalent, and consistently surpassed ProSmith, the closest state-of-the-art method. Since both methods use ESM1b embeddings, this suggests that NPClassifierFP with fine-tuned ESM1b improves generalisation for unseen NPs over ChemBERTa and the foundational Transformer model.

Meanwhile, our ProtBERT variant was the least effective among our methods for most compound thresholds, except at 40% and 80%, where it matched the performance of other models. This trend persisted across all protein identity thresholds, with ProtBERT either performing comparably or worse than the rest of our models.

Hence, the XGBoost-based classifiers built on pre-trained protein language model embeddings [281] and NPClassifierFP [245], our proposed approach, consistently achieve top performance. Across nearly all metrics, these models cluster at the upper end of the scale and are accompanied by very small standard deviation bars, indicating high stability and low variance across folds or replicates. Compared to the current state of the art, including ProSmith and ESP, our method consistently performs better across various evaluation settings, except for the identity threshold of 60%, although differences are small. Notably, our method does not deteriorate as abruptly in performance compared to other state-of-the-art approaches when evaluated on cross-partition thresholds with lower compound similarity. This suggests stronger generalisation capabilities, even for compounds that diverge significantly from the training set, e.g. those with Tanimoto similarities below 20% or 40%.

Overall, classical similarity methods are consistently and substantially outperformed by ML models. In particular, the BLASTp + Tanimoto baseline shows markedly lower scores across all metrics and cross-partition thresholds. This highlights that combining sequence similarity with chemical fingerprint similarity is insufficient to capture the functional and structural complexity required for robust classification of enzyme-substrate specificity. The results of the similarity methods in terms of MCC score across different identity thresholds are consistent with what was expected when analysing the boxplots in Supplementary Figure S6.2, since, as the thresholds increase, the MCC score follows accordingly. The same is not true for the F1 score, as this is lower at the 60% threshold than for the 40% score. As for compound splits, when the similarity threshold increases, we observe a marked divergence between MCC and F1 score trends, with MCC progressively decreasing from near zero to strongly negative values while F1 remains low but comparatively stable. This pattern indicates that the method's global predictive power deteriorates substantially at higher similarity levels. Unlike F1, which only captures the balance between precision and recall for the positive class, MCC incorporates all four components of the confusion matrix, including the true negatives. The strongly negative MCC values at higher similarity thresholds suggest that predictions become anti-correlated with the true labels, reflecting a collapse in balanced classification performance rather than a simple reduction in positive-class detection. In contrast, the relatively moderate F1 scores mask this broader degradation because they ignore true negatives. Together, these results indicate that increasing similarity thresholds introduces a distributional shift or structural bias that disrupts the method's overall discriminative capacity, despite maintaining limited positive-class retrieval performance. Nevertheless, it is possible to observe that the best result obtained was for the 20% threshold, which is aligned with

the leakage of compounds to the test set by this split (see Supplementary Figure S6.1).

Across protein-based splits, ESP consistently performs worst among the ML models at every similarity threshold and for every metric, suggesting that the fine-tuned ESM-1b representation for enzyme-substrate interaction may introduce training-set biases that limit generalisation. Regarding compound splits, ESP is outperformed by all models, nearly for all metrics and thresholds, except for ProSmith, which performs on par only at the thresholds of 80% and 40%.

ProSmith, on the other hand, provides performance values that are on par with our approaches for protein splits, independently of the threshold. When it comes to compound splits, ProSmith remains competitive with our approaches for thresholds of 60% and 80%, but fails to reach performances that are as good as our approaches for lower similarity thresholds, such as 20% and 40%, which seem to be the most challenging.

When analysing the datasets without considering stereochemistry, performances drop slightly, but tendencies in top-performing models are consistent with what is observed in Supplementary Figure S6.3.

Taken together, these results demonstrate that modern protein language model embeddings fine-tuned for EC number prediction combined with NPClassifierFP and gradient boosting provide a robust and consistently superior framework for this classification task, substantially advancing beyond traditional similarity-based baselines and improving upon existing state-of-the-art methods.

6.2.3 Runtime efficiency and memory required

Figure 6.2 shows the runtime and memory required by each method to infer ESI from protein sequences and SMILES strings pairs.

ESP starts with a low computation time but this increases significantly as the number of pairs grows. ProSmith maintains a much lower and more gradual increase in computation time. XGBoost-NPClassifierFP-ESM2 exhibits the steepest rise in computation time. On the other hand, XGBoost-NPClassifierFP-ESM1b presents an even lower computation time than ProSmith, due to the difference in computing the compound features: our method relies on fingerprint computation that is much faster than a language model embedding such as ChemBERTa. Finally, XGBoost-NPClassifierFP-ProtBERT is the fastest method, due to ProtBERT being the smallest protein language model among all methods. This comparison highlights that XGBoost-NPClassifierFP-ProtBERT is the most time-efficient method for inference across all tested, while XGBoost-NPClassifierFP-ESM2 is the most time-consuming since it uses ESM2 which represents the largest language model (3B parameters). ESP falls in between, with a moderate but notable increase in computation time compared to XGBoost-NPClassifierFP-ProtBERT and XGBoost-NPClassifierFP-ESM1b.

The memory requirements of the evaluated methods reveal distinct scalability profiles: ESP maintains consistently low memory usage across all dataset sizes, making it the most resource-efficient option. In contrast, ProSmith exhibits a sharp increase in memory demand with larger datasets (22GB), becoming prohibitively resource-intensive for high-throughput tasks. The XGBoost-NPClassifierFP variants (ProtBERT, ESM2, and ESM1b) demonstrate moderate and more predictable memory scaling, with ESM2 requiring slightly more memory than ESM1b and ProtBERT, yet all remaining far more efficient than ProSmith.

Table 6.1: Hyperparameter optimisation (HPO) and feature generation time for each pipeline.

Pipeline	HPO time	Feat. time	Total
XGBoost-NPClassifierFP-ESM2	~4h	~5h	~9h
XGBoost-NPClassifierFP-ESM1b	~4h	~35m	~4h35m
XGBoost-NPClassifierFP-ProtBERT	~4h	~20m	~4h20m
ESP	~1d30m	~4h	~1d4h
ProSmith	~1w	~45m	~1w45m

This balance positions the XGBoost-based methods as practical choices for applications requiring both performance and reasonable resource consumption, especially as dataset size grows.

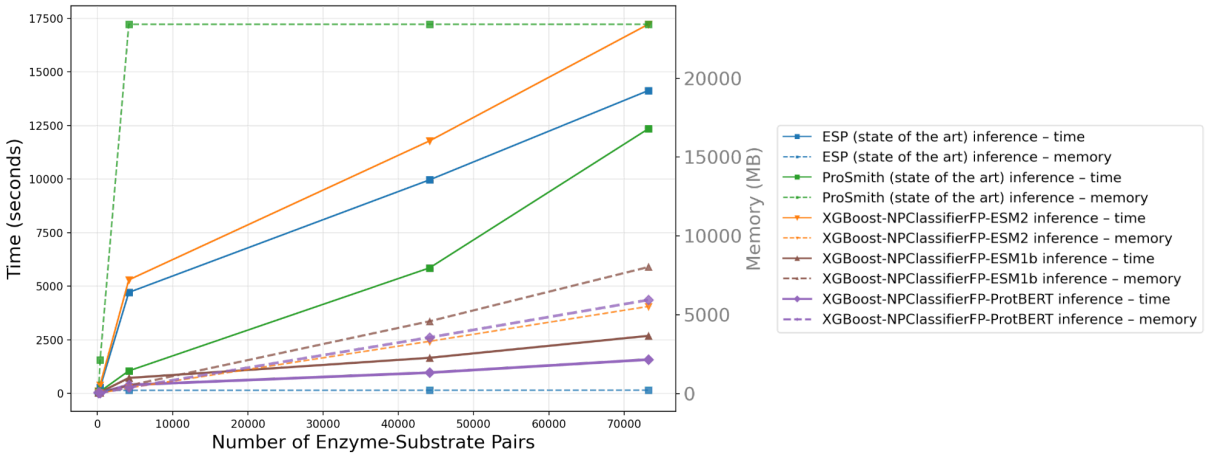


Figure 6.2: Computation and memory requirements as functions of the number of enzyme-substrate pairs for different methods: ESP, ProSmith, and XGBoost-NPClassifierFP-ESM2. The x-axis represents the number of enzyme-substrate pairs, while the y-axis shows the computation time in seconds (left) and memory requirements in MB (right)

Table 6.1 compares the total computation times for training across five pipelines: XGBoost-NPClassifierFP-ESM2, XGBoost-NPClassifierFP-ESM1b, XGBoost-NPClassifierFP-ProtBERT, and ESP, as well as ProSmith. Among the XGBoost-NPClassifierFP variants, XGBoost-NPClassifierFP-ESM2 requires the longest total time due to its extended feature generation phase, though its hyperparameter optimisation (HPO) remains consistent with the other XGBoost-based methods. The shorter total times for XGBoost-NPClassifierFP-ESM1b and XGBoost-NPClassifierFP-ProtBERT stem from their more efficient feature generation, despite sharing similar HPO durations. ESP stands out for its prolonged HPO process, which involves 10 times more trials than the XGBoost variants, resulting in a total time exceeding one day. ProSmith is the most computationally demanding pipeline, primarily due to its HPO phase, which includes both training a foundational Transformer model and optimising an XGBoost over 500 trials, leading to a total time of over one week. Thus, while the XGBoost-based pipelines offer a balance of efficiency and performance, ESP and ProSmith

are significantly slower, with ProSmith being the most time-consuming overall.

6.3 Conclusion

This study introduces a novel ML framework for predicting enzyme-substrate interactions, directly addressing persistent challenges in generalizability, negative data scarcity, and real-world applicability, where performance on highly similar compounds or enzymes is often irrelevant. While modern machine learning models outperform traditional similarity-based approaches (e.g., BLAST or Tanimoto similarity), our framework further improves prediction by combining fine-tuned protein language model embeddings, NP-specific fingerprints (NPClassifierFP), and gradient boosting, achieving superior performance across datasets, particularly for substrates with low similarity to the training data (<40%). These thresholds are critical because industrial and biotechnological applications demand robustness to novel, unseen substrates, not minor variations of known ones. Crucially, our framework achieves this without the computational overhead of methods like ProSmith: while slightly more memory-intensive than ESP during inference, it trains and optimises faster than both ESP and ProSmith, and in its lighter ESM1b and ProtBERT configurations also infers faster than both, while delivering superior accuracy. The results underscore the importance of advanced feature representation, stringent data partitioning, and high-quality negative pairs, components often overlooked in favour of benchmarking on artificially similar test cases. By prioritising scalability and efficiency alongside performance, this work provides a practical tool for enzyme engineering, metabolic pathway discovery, and biocatalyst identification in real-world scenarios where substrate diversity is the norm.

6.4 Methods

We developed a unified pipeline for enzyme–substrate specificity prediction using experimentally validated cases, both for binding and non-binding pairs (Figure 6.3). We curated interactions from the literature and retained only compounds with computable 3D structures and descriptors to ensure feature consistency and comparability with structure-based methods.

The dataset was partitioned with the GraphPart method [282] into balanced training, validation, and test sets. Prior clustering of proteins and compounds ensured that each cluster was assigned to a single partition, enforcing maximum sequence identity (proteins) and Tanimoto similarity (compounds) thresholds across splits and preventing data leakage.

The architecture comprises two modules: a protein encoder based on neural networks pre-trained for EC number prediction, using final hidden-layer activations as fixed-length embeddings, and a compound encoder using counted Morgan (NPClassifier) fingerprints. The concatenated embeddings were provided to a gradient boosting classifier to predict enzyme–substrate interactions. Performance was benchmarked against state-of-the-art models.

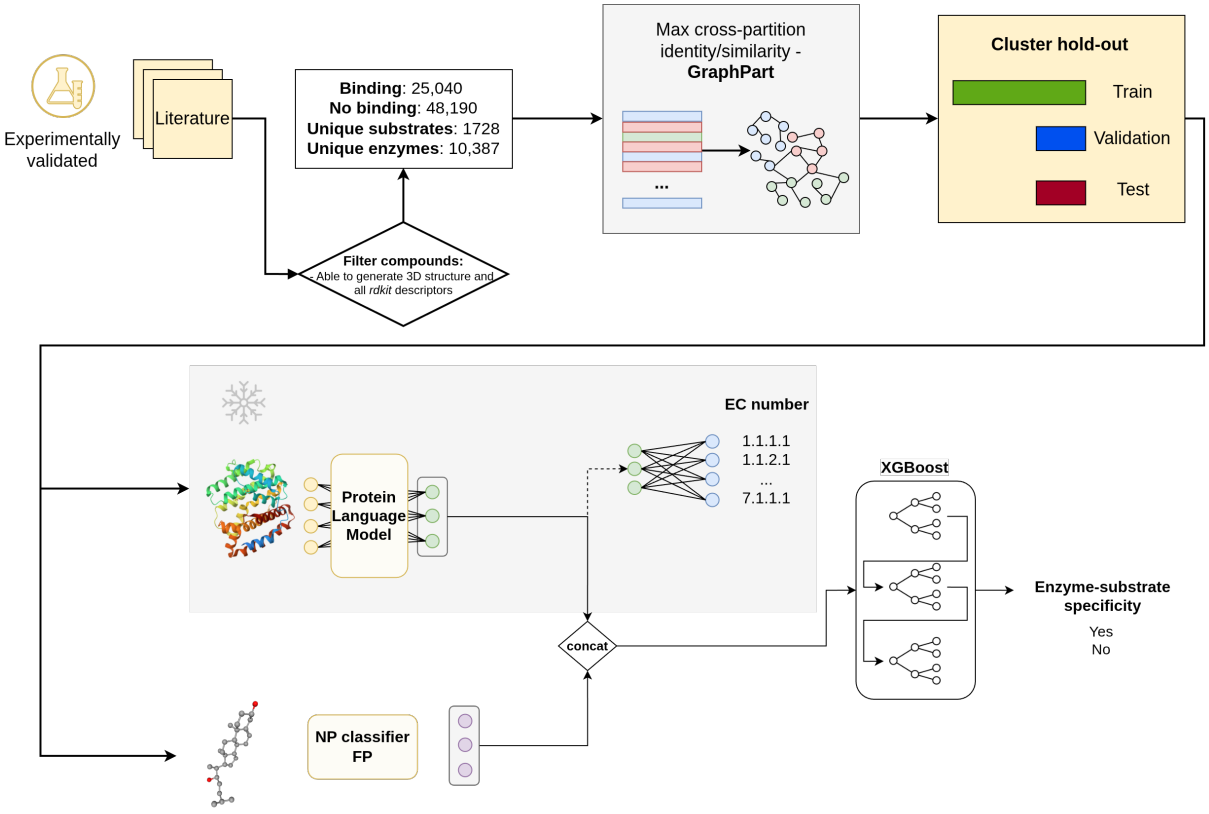


Figure 6.3: The workflow's upper section illustrates data collection and preprocessing for model training. We compiled all available data on experimentally validated enzyme-substrate pairs, then filtered to retain only those compounds for which both 3D structures and rdkit descriptors could be generated. The dataset was then partitioned using GraphPart to create balanced training, validation, and test sets, with clusters of proteins and compounds held out according to a defined maximum cross-partition identity or Tanimoto similarity. The model comprises two modules: one for processing proteins and another for compounds. The protein module utilises the weights of pre-trained neural networks to predict EC numbers, extracting the activations from the final hidden layer as protein representations, while the compound module employs counted Morgan fingerprints, known as NP classifier fingerprints. The two representations are concatenated and used to train a gradient boosting model, which predicts enzyme-substrate specificity.

6.4.1 Dataset construction

Our analysis integrates data from 10 studies, each employing rigorous experimental assays—such as coupled enzymatic reporters, mass spectrometry, or fluorescence-based detection—to quantify enzymatic performance metrics, including substrate conversion, catalytic yield, and activity. The first dataset, sourced from the Gene Ontology (GO) knowledge base, was used to train the ESP model and included only experimentally validated enzyme-substrate pairs (positive interactions). Unlike the family-specific datasets, the GO data represent general enzymatic functions without focusing on particular enzyme families. For the remaining nine datasets, which characterise the substrate scope of phosphatases [283], esterases [284], DUF849 [285], flavin-dependent halogenases [286], glycosyltransferases [274], thiolases [287], nitrilases (two datasets) [273, 288], and transaminases [289], we utilised the processed files provided

by [275]. These files were pre-filtered to include only protein-compound pairs derived from experimental screening, ensuring reliability and comparability in our analysis.

To eliminate stereochemical redundancy, we implemented a filtering pipeline to retain only one representative compound per molecular core structure. This was achieved by computing the InChIKey for each compound in the dataset. By comparing the first 14-character block of each InChIKey, we removed all isomers sharing identical core connectivity, ensuring that only one structural variant per molecular scaffold was retained. This approach minimises redundancy and prevents inflating performance results.

6.4.2 Data splits

Using GraphPart [282], we generated data partitions by enforcing three protein cross-partition identity thresholds (40, 60 and 80%) and four compound Tanimoto similarity thresholds (20, 40, 60, 80%). The graph-based clustering algorithm prevents data leakage by iteratively removing or reassigning sequences that violate a predefined similarity cutoff. Pairwise comparisons using Needleman-Wunsch sequence alignment for proteins and Tanimoto similarities of 2048-bit Morgan fingerprints across seven resulting splits were performed, enabling assessment of performance under varying compound diversity (training on compounds below Tanimoto thresholds) and protein diversity (training on proteins below sequence identity) thresholds.

6.4.3 Calculating substrate and enzyme representations

Enzyme-substrate interaction prediction involves natural products (NPs), which exhibit greater structural diversity, rigidity, fused ring systems, and chiral centres than synthetic molecules [244], features arising from millions of years of evolutionary biosynthetic pathway diversification [245]. To capture this chemical complexity, we employed counted Morgan fingerprints (NPClassifierFP) [245], selected based on our prior benchmarking of NP representations for ML (chapter 5), and computed using the DeepMol package [267].

We selected ESM2 (3B parameters), ESM1b, and ProtBERT for enzyme embeddings based on our comparative analysis [281], which demonstrated superior performance for EC number prediction. We extracted the final hidden-layer activations, capturing catalytic activity information from the EC-prediction model, to represent functional properties of all enzymes in our dataset for enzyme-substrate interaction prediction.

6.4.4 Optimisation and construction of machine learning models

We optimised XGBoost across seven hyperparameters including learning rate, maximum tree depth, L1/L2 regularisation terms, constraints on leaf output values, minimum child node weights, continuous feature binning parameters, and class weighting factors.

For comparative benchmarking, we evaluated our approach against two state-of-the-art methods: ESP and ProSmith. ESP [276] employs a computationally efficient pipeline using ESM1b-derived enzyme embeddings combined with small graph neural network based substrate representations fed into an XGBoost

classifier, enabling rapid training and optimisation. In contrast, ProSmith [277] follows a more resource-intensive workflow involving ESM1b enzyme embeddings combined with ChemBERTa substrate representations, followed by pretraining of a large transformer network and subsequent XGBoost training. This approach restricts the practicality of applying extensive resampling techniques, such as cross-validation, which are often critical for robust model evaluation.

The ESP model's hyperparameter optimisation followed the original protocol, with a 2000-trial randomised search that minimised a custom asymmetric loss function. The loss penalised false negatives more heavily than false positives by applying a quadratic penalty to false negatives and a softer, sublinear penalty to false positives. This weighting reflects the higher cost of missing true enzyme-substrate interactions in our application domain. For ProSmith, the optimisation process first used 100 epochs of transformer pretraining with mean squared error loss, after which the XGBoost component underwent 500 randomised hyperparameter trials to maximise the Matthews Correlation Coefficient on the validation set.

Following hyperparameter optimisation for all models, we created final versions by retraining on the combined training and validation datasets, using the independent test sets solely for reporting the ultimate performance metrics.

6.4.5 Similarity based nearest-neighbour classification

To benchmark ML approaches against similarity-based methods, we implemented a baseline combining BLASTp for enzyme similarity and Tanimoto coefficients for substrate similarity. For each test enzyme, we identified the closest training set match via BLASTp (lowest e-value), while for substrates we used ECFP4 (radius-2 Morgan) fingerprints to find the most similar training compound (highest Tanimoto similarity). Test pair labels were then transferred from their nearest training neighbours based on this combined similarity matching.

6.5 Data and code availability

The data and code of this project are available in the enzyme-substrate-interaction repository.

7

Predicting Precursors of Plant Metabolites

Plants produce specialised metabolites, which play critical roles in defending against biotic and abiotic stresses. Due to their chemical diversity and bioactivity, these compounds have significant economic implications, particularly in the pharmaceutical and agrotechnology sectors. Despite their importance, the biosynthetic pathways of these metabolites remain largely unresolved. Automating the prediction of their precursors, derived from primary metabolism, is essential for accelerating pathway discovery. Using DeepMol's automated machine learning engine, we found that regularised linear classifiers offer optimal, accurate, and interpretable models for this task, outperforming state-of-the-art models while providing chemical insights into their predictions. The pipeline and models are available at the repository: `https://github.com/jcapels/SMPrecursorPredictor`.

Keywords: machine learning · plant specialised metabolites · biosynthesis

---

This chapter was co-authored by João Cheixo and was published in the following peer-reviewed paper: *Capela, J., Cheixo, J., de Ridder, D., Dias, O., & Rocha, M. (2025). Predicting precursors of plant specialised metabolites using DeepMol automated machine learning. Journal of integrative bioinformatics, 22(2), 20240050.*

7.1 Introduction

Gene duplication events have played a key role in shaping plant evolution, contributing to the expansion of gene functions and variability in gene content [1]. This genetic diversity is particularly noticeable in the evolution of lineage-specific specialised metabolites [3], which serve to protect plants from both biotic and abiotic stressors, while also attracting and exploiting external agents such as pollinators [3]. The enzymes responsible for specialised metabolite biosynthesis form distinct metabolic pathways, generating a wide range of compounds that hold significant value in industry, being widely used in agricultural, cosmetic and pharmaceutical products [4].

Despite their importance, the pathways and genes responsible for the biosynthesis of these specialised metabolites remain largely uncharted [3]. Even for the model plant *Arabidopsis thaliana*, nearly half of

its genes still lack an assigned Gene Ontology term, possibly due to convergent evolution events, where unrelated enzymes from distant lineages with different sequences utilise common substrates to produce similar compounds [9, 10]. However, it is well understood that all specialised metabolites are synthesised through biosynthetic routes that utilise compounds derived from primary metabolic processes as building blocks [56]. These precursor compounds arise from key primary metabolic pathways, including photosynthesis, glycolysis, the citrate cycle, the shikimate pathway, mevalonate and the DOXP/MEP pathway [65], which provide the foundation for specialised metabolite production.

Alkaloids are synthesised from amino acids, which provide the essential amine groups for their structure. These amino acids and various other components such as terpenoids, polyketides, and fatty acids contribute to alkaloids' diverse structures [264].

For terpenoids, the isoprene rule states that their fundamental building blocks are the universal five-carbon precursors IPP and DMAPP [79]. The condensation of these two precursors leads to the formation of key prenyl diphosphate intermediates, such as Geranyl Diphosphate (GPP), Farnesyl Diphosphate (FPP), and Geranylgeranyl Diphosphate (GGPP), which serve as starting points for the synthesis of diverse terpenoid structures [79].

Another key group of plant-specialised metabolites is the phenylpropanoid class. The shikimate pathway in plants acts as the entry point for their production by synthesising the amino acid phenylalanine. The enzyme phenylalanine ammonia-lyase then converts phenylalanine into cinnamic acid, initiating the phenylpropanoid biosynthetic pathway [65].

In 2019, Eguchi and colleagues introduced a Molecular Graph Convolutional Neural Network (MGCNN) specifically designed for the multi-label prediction of alkaloid precursors [264]. This model leverages atomic information and implements a "message passing" strategy, where data about neighbouring atoms is transmitted via the edges in the molecular graph. By aggregating and updating information at each atom, the network learns atom-level representations that capture the surrounding chemical environment. While the model outperformed traditional approaches like a neural networks and RF that were trained with Extended Connectivity Fingerprints (ECFP), the use of accuracy as the evaluation metric raises concerns given its limited suitability for unbalanced, multi-label datasets. Additionally, MGCNN produces predictions that lack interpretability, offering no insights into the key structural features driving the outcomes. Lastly, the study's scope was restricted to alkaloids.

The present study seeks to address these limitations by exploring more suitable metrics for multi-label classification in unbalanced datasets and broadening the focus to include terpenoids, phenylpropanoids, glucosinolates, and benzoxazinoids. Furthermore, we utilised DeepMol's AutoML module [267] to identify the optimal model for this specific task. This model was applied to predict the precursors of all plant compounds in LOTUS-DB [290], allowing for a critical analysis of the main prediction patterns across different compound classes on a larger scale. To demonstrate how our method offers improvements over the state-of-the-art, we compared it not only with MGCNN, but also with NeuralNPFP, a fingerprint specifically designed for NP, as well as other fingerprints that have been effective in classifying NPs by pathway, class, and subclass [245]. Additionally, we included MinHashed fingerprints (MHFP), which have shown promise for representing NPs in supervised bioactivity prediction tasks [18]. Finally, we aim

to provide a method that offers interpretability regarding predictions.

7.2 Results and Discussion

7.2.1 AutoML consistently outperforms MGCNN and NP-focused fingerprints

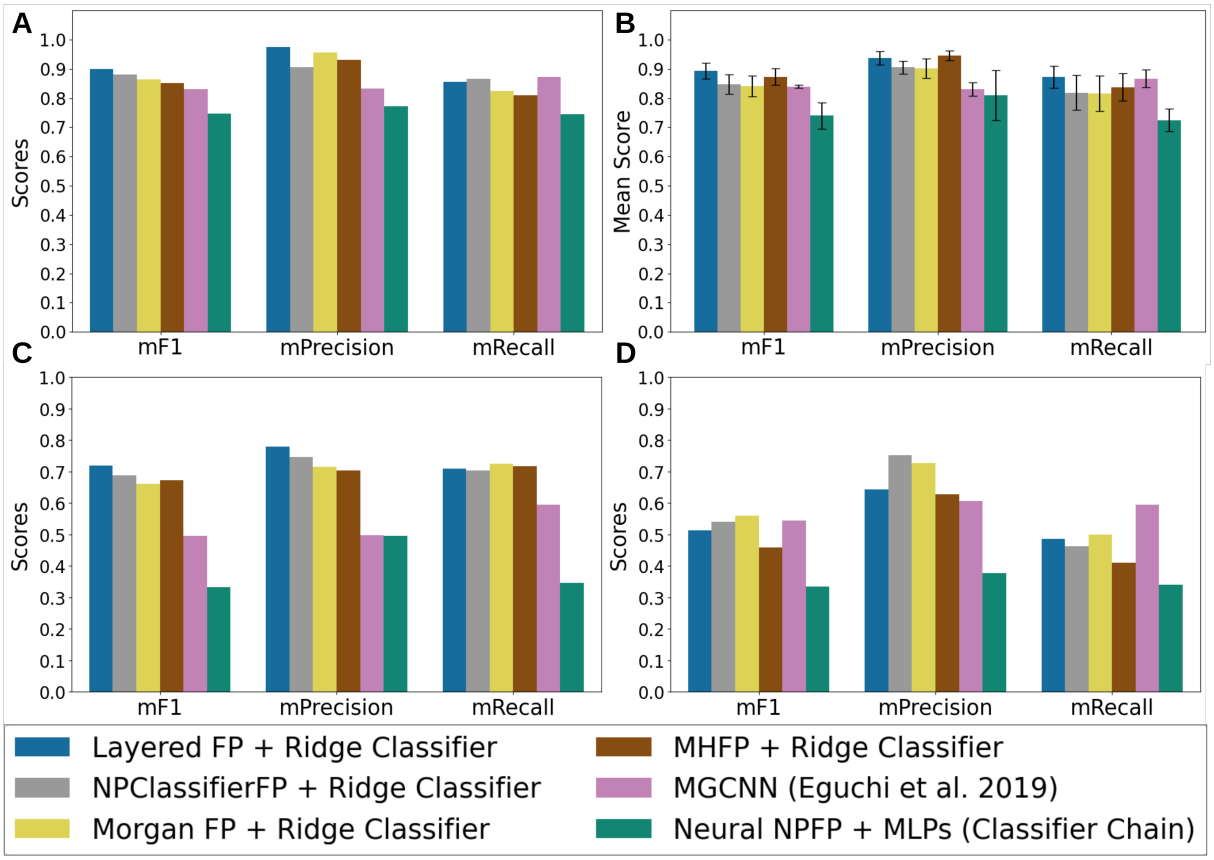


Figure 7.1: Results of each model evaluated on four different settings (see Methods). (**A**) The test set originated from the stratified split. (**B**) Cross-validation datasets for predicting the precursors of alkaloids from the Eguchi *et al.* 2019 paper. (**C**) The distant similarity clustered compounds dataset. (**D**) The overlapping natural product classes dataset.

Figure 7.1 provides a comparative analysis of several classification models across three metrics: macro F1-score (mF1), macro Precision (mPrecision), and macro Recall (mRecall).

The models were selected based on the outcomes of an AutoML search, where Ridge classifiers using Morgan FPs and Layered FPs emerged as the top performers in terms of model and feature extraction methods, respectively. Among the models, the standout performer was a Ridge classifier trained on Layered FPs. The Layered FPs are molecular FPs that capture multiple structural attributes of a molecule using a multi-layered approach, where each layer captures different molecular features, including topology, bond order, atom types, ring presence and sizes, and aromaticity. The construction process involves

generating a multi-layered hash for each molecular subgraph to set bits in the fingerprint, allowing for flexible representation. On top of this, the AutoML engine from DeepMol improved this model's validation results by applying feature selection, specifically filtering out features with a variance below 0.0628, which was identified as the optimal threshold by the AutoML. The approach that combines Layered FPs, a variance threshold filter, and a Ridge classifier is called the *Layered FP + Ridge Classifier*. In addition, we evaluated a Ridge classifier trained with Morgan FPs, known for their straightforwardness in mapping molecular substructures directly to specific FP bits. Similar performance patterns were observed for NPClassifier FP and MHFP, where Ridge classifiers achieved the best results. However, for the Neural NPFP, a classifier chain model leveraging a MLP as the base estimator delivered the top performance.

Layered FP + Ridge Classifier delivered the best performance on the test dataset (Figure 7.1A), achieving an mF1 score of 0.898 and an mPrecision of 0.975. This classifier demonstrated a well-balanced trade-off between mPrecision and mRecall, with a slight bias towards mPrecision. Notably, this model exhibited significant improvements over the MGCNN model [264]. It achieved markedly higher F1 and precision scores than MGCNN, with statistical significance ($p\text{-value}_{F1} < 0.05$, $p\text{-value}_{precision} < 0.01$, in a WSR test, see Supplementary Information). While other models, such as *NPClassifierFP + Ridge Classifier*, *MHFP + Ridge Classifier*, and *Morgan FP + Ridge Classifier*, also outperformed MGCNN in terms of mF1 score, the *Layered FP + Ridge Classifier* was the only one to achieve a statistically significant improvement. As depicted in Figure 7.1A, MGCNN yielded a marginally higher mRecall (0.871) than the other models, though this difference was statistically significant only in comparison to *MHFP + Ridge Classifier* ($p\text{-value}_{recall} < 0.05$, in a WSR test) and *NeuralNPFP + MLPs* ($p\text{-value}_{recall} < 0.01$, in a WSR test). The pipeline utilising NeuralNPFP underperformed across all metrics when compared to the other approaches.

The *Layered FP + Ridge Classifier* with the Ridge classifier outperformed MGCNN in predicting nearly all precursors, except cholesterol, campesterol, and FPP. This indicates that MGCNN exhibits slightly stronger predictive capabilities for steroid, sesquiterpenoid, and triterpenoid precursors (for further details, refer to the Supplementary Information). When focusing specifically on the alkaloids dataset from Eguchi *et al.* (2019) [264] (Figure 7.1B), *Layered FP + Ridge Classifier* consistently outperformed the state-of-the-art MGCNN across all metrics, with statistically significant improvements in mF1 and mPrecision. Although the *MHFP + Ridge Classifier* achieved a higher mPrecision than the *Layered FP + Ridge Classifier*, this difference was not statistically significant, and the MHFP model underperformed in terms of mRecall and mF1. Furthermore, the *Layered FP + Ridge Classifier* was the only approach that surpassed MGCNN across all metrics.

These performance trends persisted with the clustered dataset (Figure 7.1C), where the Ridge classifiers consistently outperformed MGCNN across all metrics. However, the differences were not statistically significant, likely due to the small number of labels evaluated (13) and the high variability in label scores across different methods (see Supplementary Information). The only statistically significant differences were observed when comparing all Ridge classifiers to the *NeuralNPFP + MLPs* pipeline.

In the overlapping dataset (Figure 7.1D), the *Morgan FP + Ridge Classifier* and MGCNN models showed

the best performance, although their advantage was slight and not statistically significant, except in precision scores, where the *Morgan FP + Ridge Classifier* significantly outperformed the *Layered FP + Ridge Classifier*. All models showed a substantial improvement over the *NeuralNPFP + MLPs* in terms of F1 scores. While other models exhibited lower precision scores, these differences were not statistically significant due to the large variability across labels (more details can be found in the Supplementary Information).

The ML pipelines discovered by DeepMol AutoML demonstrated clear advantages over models utilising Natural Product-focused fingerprints and the state-of-the-art tool MGCNN. While the Natural Product-focused representations such as NPClassifierFP and MHFP had isolated strengths in certain metrics, they generally underperformed compared to the Ridge classifiers using Layered and Morgan FPs. These findings highlight the importance of feature selection and optimisation, as facilitated by DeepMol's AutoML framework, in achieving state-of-the-art performance in molecular classification tasks.

7.2.2 Predictions for LOTUS-DB compounds show consistent patterns

Next, we predicted the precursors for all plant compounds in LOTUS-DB to annotate them and provide comprehensive insights. We also conducted a large-scale analysis of the patterns in our predictions across all the 190,273 compounds.

The pie charts in Figure 7.2 offer insights into the predictions made by the *Layered FP + Ridge Classifier* for plant compounds in LOTUS-DB [290], a comprehensive database of NPs with taxonomic details and chemical class ontologies. This database enables the filtering of compounds based on their chemical class and the organisms responsible for their production, facilitating our further analysis.

The top chart (Figure 7.2A) illustrates that our tool predicts the precursors of 77.6% of plant compounds in LOTUS-DB, leaving 22.4% unannotated. These unannotated compounds likely stem from primary metabolism or have precursors not included in the current dataset. While compounds from primary metabolism probably represent only a small portion of this group, the majority might consist of metabolites with precursors outside our current scope. For example, polyketides are synthesised from precursors like acetyl-CoA, malonyl-CoA, and propionyl-CoA, none of which were included in our dataset, as plant polyketide production was not a focus of this study. Expanding the dataset to include additional precursors, especially those involved in the biosynthesis of polyketides and other important plant metabolite classes, will be an important next step.

The bottom charts display the percentage of predicted precursors involved in the biosynthesis of alkaloids, terpenoids, phenylpropanoids, and glucosinolates. Benzoxazinoids were excluded from the analysis as this class of compounds was not in the database.

In the first chart (Figure 7.2B), which focuses on predictions for alkaloids, tryptophan is identified as the most prevalent precursor, accounting for 20.9% of the predictions, followed by L-tyrosine at 17.7%, and secologanin at 13.8%. Other precursors contribute smaller percentages. The prevalence of amino acids, some terpenoids such as secologanin, and the aromatic acid anthranilate align well with expectations, as these are known precursors of alkaloids. However, the dominance of tryptophan, L-tyrosine, and

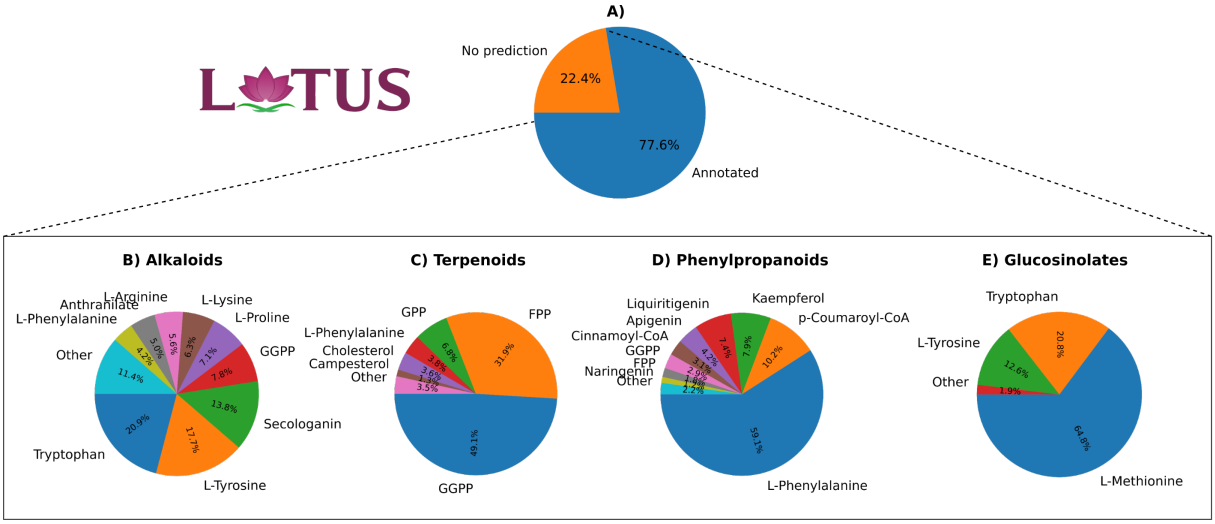


Figure 7.2: Predictions for LOTUS-DB using *Layered FP + Ridge Classifier*. The top pie chart illustrates the percentage of annotated and unannotated plant compounds in LOTUS-DB. A majority (77.6%) of the compounds are annotated, while 22.4% remain unannotated. The bottom four pie charts show the percentage of predictions for each precursor. Each chart represents a different subset of compound classes.

secologanin could indicate a bias in the predictions, potentially caused by label imbalance, where overrepresented labels skew predictions in their favour (see Methods, Figure 7.4). Alternatively, it may reflect the actual distribution of compounds in the database, where a significantly larger proportion of compounds are produced from these most prevalent metabolites compared to others. Interestingly, despite the label imbalance, anthranilate did not appear as the 5^{th} most common label as expected, even though it holds that position (as 5^{th} most prevalent label for alkaloids) in our dataset (see Methods, Figure 7.4). Although GGPP was not included in the dataset as an alkaloid precursor, alkaloids are known to be hybrid structures that can incorporate terpenoids into their carbon skeletons. This observation suggests that the model has learned chemical patterns from terpenoids, enabling it to generalise and apply this knowledge to predict for other compound classes, even when specific precursors, like GGPP, are not explicitly present in the alkaloid dataset.

In the second chart (Figure 7.2C), which focuses on the precursors of terpenoids, GGPP is the most prevalent label, accounting for 49.1% of the predictions, while FPP contributes 31.9%. The contributions from other precursors are significantly smaller, with the third precursor (GPP) contributing 6.8%. Another precursor, campesterol, had a lower prevalence compared to L-phenylalanine and cholesterol. The predictions involving L-phenylalanine are questionable, as terpenoids are not synthesised from this amino acid. Cholesterol, on the other hand, is a terpenoid, and despite not being explicitly included in the dataset as a terpenoid precursor, it is understandable that the model would make this prediction, especially when steroids are being analysed. Once again, the overrepresented labels also appeared most frequently in the predictions.

The third chart (Figure 7.2D) focuses on the prediction of phenylpropanoid precursors. Unsurprisingly,

the dominant precursor, responsible for 59.1% of the predictions, was L-phenylalanine, followed by p-Coumaroyl-CoA at 10.2% and Kaempferol at 7.9%. Most of the other precursors were expected as they were assigned as phenylpropanoid precursors. FPP and GGPP were also predicted as precursors, which is questionable, as they are not known to be involved in phenylpropanoid biosynthesis. However, the fact that L-phenylalanine was predicted as a precursor for some terpenoids, and FPP and GGPP as phenylpropanoid precursors, suggests that these metabolites could be part of complex natural products, such as hybrid molecules that incorporate elements from both pathways. Once again, the most frequent predictions were associated with the overrepresented labels.

Finally, in the fourth chart (Figure 7.2E), focused on the production of glucosinolates, L-methionine accounts for 64.8% of the predictions, with tryptophan contributing 20.8% and L-tyrosine 12.6%. These results are in accordance with the precursors defined for glucosinolates.

The predictions made for all plant compounds in LOTUS-DB highlight the effectiveness of our model in annotating a dataset of 190,273 compounds and revealing insights into the underlying biosynthetic pathways. The detailed breakdown of precursor predictions across various compound classes reveals clear patterns and interesting challenges. The model correctly identified common precursors such as tryptophan, L-phenylalanine, and GGPP, but also seems to have faced issues with overrepresented labels, which may have skewed certain predictions. For instance, in alkaloid biosynthesis, the high prevalence of tryptophan and L-tyrosine may reflect a bias in the data, while in terpenoid and phenylpropanoid classes, a few questionable predictions might indicate the complexity of natural product biosynthesis and the possibility of hybrid structures.

Overall, these results demonstrate the robustness of our predictive model, while also highlighting areas for future improvement, such as refining the handling of label imbalances and expanding the dataset to cover other biosynthetic pathways.

7.2.3 Ridge classifier can make and explain relevant predictions

Next, we aimed to evaluate the predictive capabilities of the *Morgan FP + Ridge Classifier* pipeline on several complex cases and analyse the significance assigned to specific features. This pipeline was selected for its ability to trace FP bits back to specific molecular structures, a feature that is not available with Layered FPs in the *Layered FP + Ridge Classifier*. We selected four compounds for this analysis: two monoterpenoid indole alkaloids recognised for their cryptic biosynthesis and two others with partially elucidated biosynthetic pathways.

Figure 7.3 illustrates the relationship between the model coefficients of the *Morgan FP + Ridge Classifier* and the substructures of the compounds. These coefficients represent the weights assigned to each feature, defining the hyperplane that separates the labels. Each coefficient corresponds to a specific feature (or molecular substructure) and indicates the extent and direction (positive or negative) of its influence on the label prediction.

The carbon skeletons of the selected monoterpenoid indole alkaloids, quinine and ajmaline, are derived from both tryptophan and secologanin. Quinine, an alkaloid obtained from the Cinchona trees of the

Rubiaceae family, has a long history as an antimalarial treatment [291]. Ajmaline, on the other hand, is an antiarrhythmic drug used in diagnosing Brugada syndrome, a rare type of arrhythmia [292].

The biosynthesis of these compounds begins with the condensation of tryptamine (derived from tryptophan) and secologanin, catalysed by strictosidine synthase. Despite this, the substructures originating from secologanin largely disappear in the final structures, especially in the case of ajmaline. This suggests that predicting these molecules as derivatives of secologanin is challenging, given that the distinct features of secologanin may not be evident in the final chemical structure.

An experiment was conducted in which we identified the key substructures of secologanin and selected 18 molecules where these substructures largely disappear in the final structures. These molecules are still related to secologanin but with few of its substructures retained during biosynthesis, making them hard to annotate and link to the precursor. They were excluded from the training set. Despite this, secologanin was still predicted as a major precursor for these molecules, achieving a recall of 0.89, precision of 1.00, and an F1 score of 0.94, showing the model's strong predictive ability.

Figure 7.3 presents the interpretability of the predictions for quinine. The most important features for predicting quinine primarily include the quinoline heterocyclic ring and a propene fragment. The high coefficient for the FP bit corresponding to the quinoline ring system may suggest an unknown step in the biosynthesis of quinine. It is believed that the quinoline structure found in cinchoninone arises from the rearrangement of cinchonaminal, which is thought to be involved in quinine's biosynthesis [291]. Interestingly, the strong signal for the quinoline ring cannot be attributed to any significant presence of compounds with similar structures in the training set, as it did not contain such compounds; instead, these were all reserved for the test set. In contrast, the high coefficient for the propene fragment's FP bit aligns with chemical intuition, as this fragment is the only part of the secologanin carbon skeleton retained in quinine [291].

For ajmaline, no substructure directly associated with secologanin was identified. However, rather than a propene fragment, a propane fragment exhibited a strong signal and played a crucial role in model prediction. This observation could be linked to a reduction event during the biosynthesis of ajmaline, where a propene fragment derived from the secoiridoid moiety of secologanin undergoes reduction by the 1,2-dihydrovomilenine 19,20(S)-reductase enzyme [292]. This reduction results in the propane moiety, which is highlighted in dark blue in Figure 7.3, further supporting the model's predictive power in capturing biosynthetic subtleties.

The third compound featured in Figure 7.3, tangutorid E, is a β-carboline alkaloid found naturally in tomatoes and recently identified as a biomarker for human consumption of tomato juices [293]. β-carboline alkaloids are produced through a Pictet-Spengler reaction between indolethylamines, such as tryptophan-derived compounds, and aldehydes or α-ketoacids [294].

The model predicts that tangutorid E originates from tryptophan, with its β-carboline moiety playing a crucial role in this prediction. The presence of the indole structure and the α-amino group in tryptophan are key features influencing this outcome. This highlights the model's focus on critical structural elements related to the biosynthesis of β-carboline alkaloids, emphasising its ability to capture essential chemical patterns associated with these complex molecules.

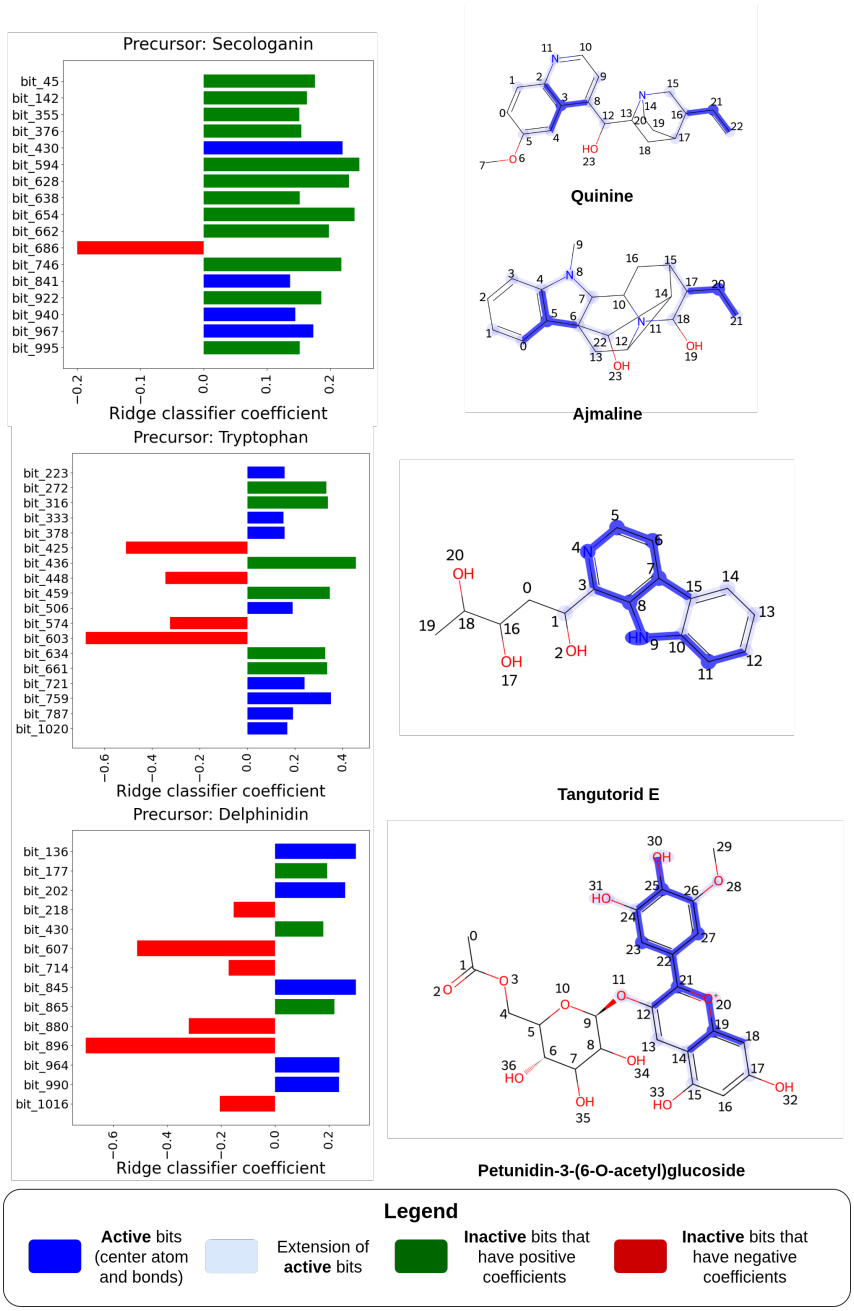


Figure 7.3: Predictions made by the *Morgan FP + Ridge Classifier* model and the importance given by the model to structural features in the compounds. On the left, bar plots show the top Ridge classifier coefficients, which correspond to bits in the FP. Red bars represent inactive negative coefficients, green bars show inactive positive bits and dark blue bars indicate active positive coefficients. The molecules subjected to the predictions are presented on the right, and the most important bits of the FP (represented in dark blue in each respective bar plot) are highlighted, where the extension of the bits (atoms and bonds within the radius defined by MorganFPs) is shaded by faded blue.

The fourth compound is a phenylpropanoid produced by *Vitis vinifera*, Petunidin-3-(6-O-acetyl)glucoside, an anthocyanin discovered in the grapevine berry, which contributes to its blue colour. It is produced through the methylation of delphinidin glycosides [295, 296].

The model identifies delphinidin as its precursor and focuses on the structural features of anthocyanin

precursors —pelargonidin, cyanidin, and delphinidin — that differ primarily by the number of hydroxyl groups attached to the benzene ring (atoms 22–27 in Figure 7.3). Delphinidin, for instance, contains three hydroxyl groups, while pelargonidin and cyanidin have one and two, respectively. The model assigns particular importance to the bits corresponding to these hydroxyl fragments to differentiate between the anthocyanin precursors.

Additionally, the model highlights the central pyran ring in its cationic form, which is a distinguishing feature for anthocyanin precursors, setting them apart from other phenylpropanoid precursor groups such as flavones, flavanols, and isoflavonoids. This emphasises the model's sensitivity to the key structural elements that define the biosynthetic pathways of anthocyanin compounds.

7.3 Conclusion

Our work extends the dataset of Eguchi and collaborators [264] to include terpenoid, phenylpropanoid, glucosinolate and benzoxazinoid precursors rather than only alkaloids. By leveraging DeepMol's AutoML engine to select the optimal combination of ML models, feature extraction, and selection methods, we improved the prediction of not only alkaloid precursors but also those of all other compound classes. We proved that Ridge classifiers outperform the state-of-the-art model, MGCNN, across multiple datasets, offering both accuracy and interpretability. Additionally, the combination of features and models identified by DeepMol's AutoML exceeds the performance of models trained with NP-focused FPs such as NeuralNPFP [244] and NP Classifier FP [245], or one of the most promising FPs for representing NPs for bioactivity prediction tasks (MHFP) [18, 251]. The structural features that influence model predictions are also closely aligned with the biosynthetic pathways leading to the final compounds, further validating the effectiveness of the model.

Upon predicting precursors for 190,273 plant metabolites from LOTUS-DB, we successfully predicted precursors for over three-quarters of the compounds, which is a strong result. However, this also highlights that certain metabolite classes, such as polyketides, were not predicted due to missing precursors in our dataset. Additionally, some compounds from general or primary metabolism, as expected, were not annotated. The prediction patterns in this large database showcase the robustness and efficiency of our model, which accurately predicted the expected biosynthetic precursors for various compound classes, like alkaloids. However, overrepresented labels posed challenges, potentially skewing some predictions.

Improvements to our method include expanding the dataset to include more precursors and addressing the issue of label imbalance. Future work will harness the strengths of this model to enhance pathway enumeration algorithms and retrobiosynthesis tools like BioNavi-NP [297] and READRetro [298], enabling the rapid identification of potential starting points and critical building blocks while excluding irrelevant precursor candidates. This approach will streamline the exploration of biosynthetic pathways, potentially accelerating the discovery and understanding of natural product biosynthesis.

7.4 Methods

7.4.1 Data

This study employed a manually curated dataset consisting of 566 alkaloids and their biosynthetic precursors, sourced from scientific literature and the Kyoto Encyclopedia of Genes and Genomes (KEGG), as described by Eguchi *et al.* (2019) [264]. The dataset features 17 precursors, including ten amino acids, one aromatic acid (anthranilate), five terpenoids, and indole-3-glycerol phosphate (IGP).

To further expand this dataset, we extracted the biosynthetic pathways for terpenoids, phenylpropanoids, glucosinolates, and benzoxazinoids from KEGG. These pathways were converted into directed acyclic graphs, with compounds represented as nodes and reactions as edges, and we employed depth-first search to identify precursors.

For terpenoids, the identified precursors included GPP (precursor of monoterpenoids [299]), GGPP (precursors of diterpenoids [299] and carotenoids [300]), FPP (precursors of sesquiterpenoids and triterpenoids [299]), and campesterol (precursor of brassinosteroids [301]).

The precursors for phenylpropanoids included L-phenylalanine, the universal starter for all phenylpropanoids, p-Coumaroyl-CoA (precursors of flavonoids and stilbenoids [65]), naringenin and liquiritigenin (isoflavonoids), as well as apigenin and kaempferol (flavones and flavonols [302]), and pelargonidin, cyanidin, and delphinidin (anthocyanins [302]).

The precursors for glucosinolates are amino acids [303], which are already included in the dataset, while the precursor for benzoxazinoids is IGP [304]. Figure 7.4 illustrates the number of compounds associated with each precursor.

The dataset was divided into training (70%), validation (20%), and test (10%) sets. Figure 7.5 shows clusters of compounds grouped by their chemical similarities using t-distributed Stochastic Neighbour Embedding (t-SNE) from a similarity matrix (Supplementary Information), positioning compounds with greater chemical similarity closer together, while those with significant differences are placed further apart. Figure 7.5A and B show the distribution of compounds by chemical class and dataset, respectively.

Additionally, the dataset was split into two alternative ways to evaluate model robustness under more challenging training-test conditions. First, we selected more distant compound clusters (Figure 7.5C), ensuring at least minimal precursor representation by including a few compounds in the training set; otherwise, the model could not predict that specific precursor. Second, compounds that are highly similar but belong to different NP classes were excluded from the training set and included in the test set, as illustrated in Figure 7.5D. This configuration creates a challenging prediction task because, despite the molecules' high similarity, they originate from different precursors.

7.4.2 Metrics

The metrics used include the F1 score, macro F1 score (mF1), recall, macro recall (mRecall), precision, and macro precision (mPrecision), each offering insights into different aspects of model performance.

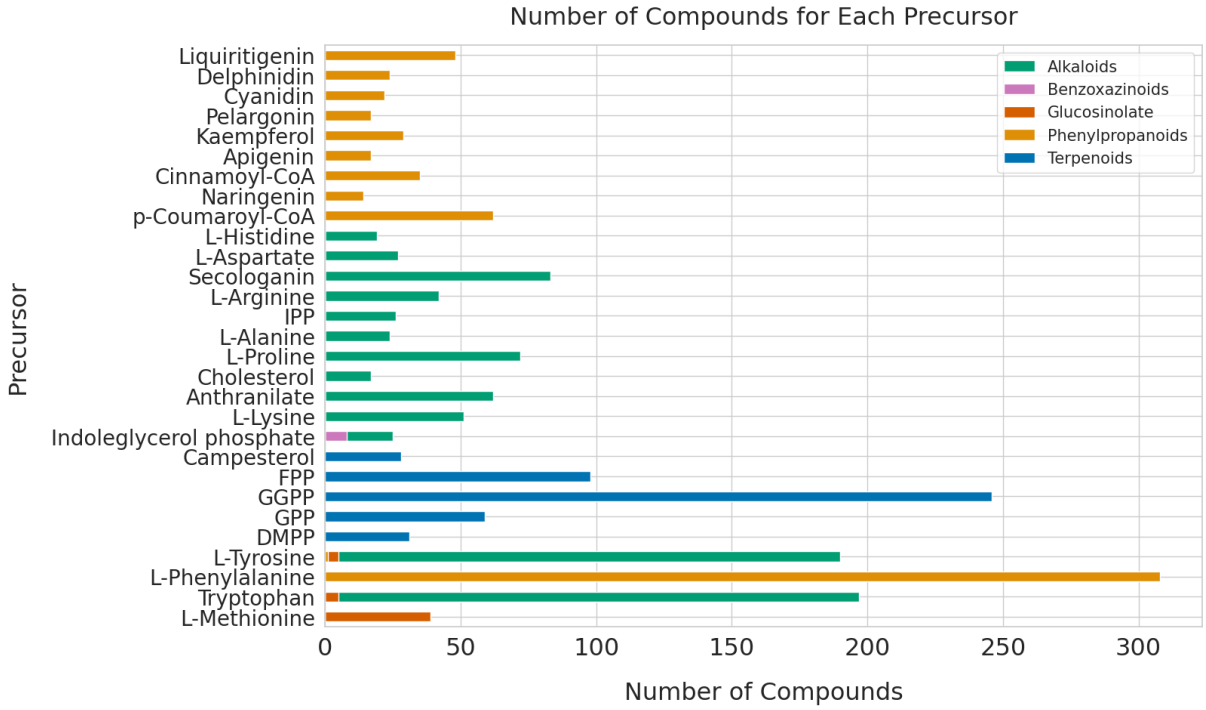


Figure 7.4: Number of compounds associated with each precursor. Note that one compound can have more than one precursor.

The mF1 score is calculated as the harmonic mean of precision and recall for each label individually, and these values are then averaged to produce a single score for the entire dataset. A similar approach is used for mRecall and mPrecision, where the averages of precision and recall across labels are computed. The exact formulations can be found below.

$$\text{mPrecision} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i} \tag{7.1}$$

$$\text{mRecall} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i} \tag{7.2}$$

$$\text{mF1} = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{7.3}$$

where N represents the total number of labels, with Precision_i and Recall_i indicating the precision and recall for label i, respectively. True Positives_i are the true positive predictions for label i, and False Negatives_i are the missed predictions for label i. Finally, the False Positives_i represent the incorrect positive predictions.

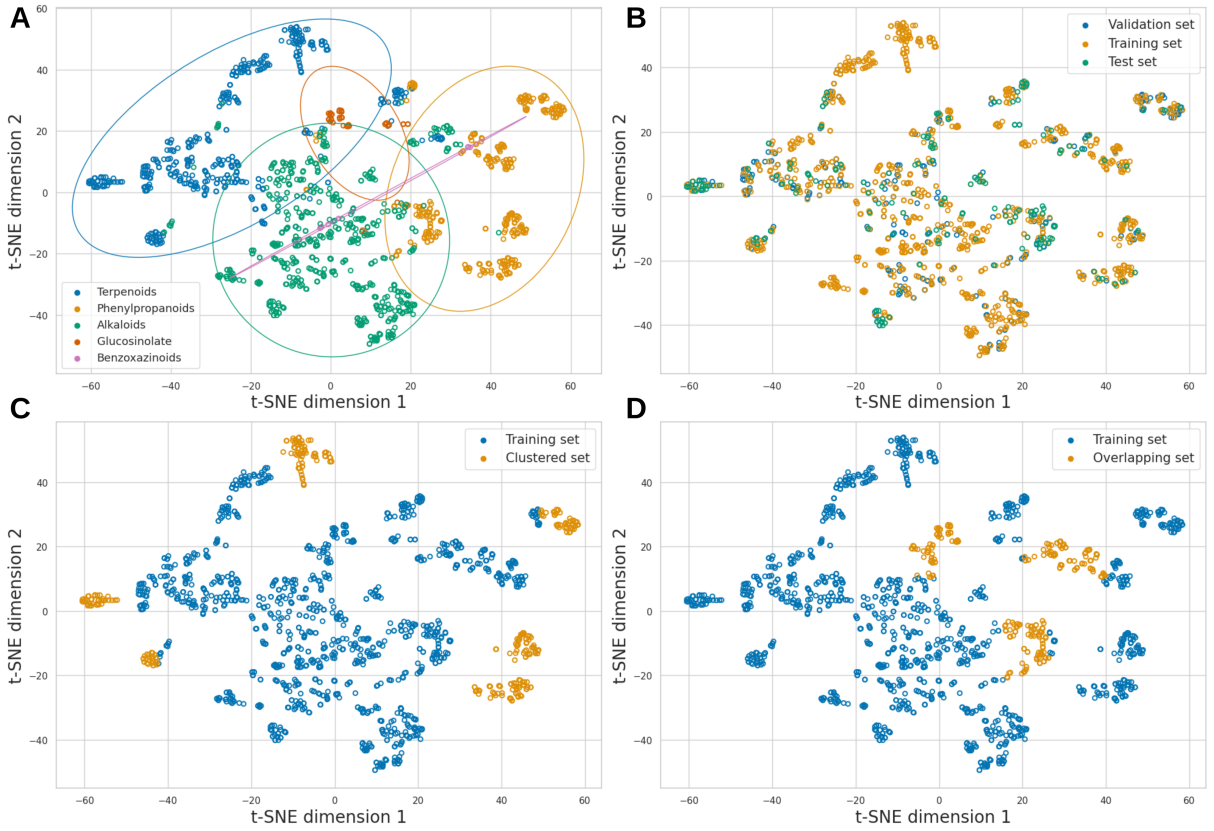


Figure 7.5: Chemical space of the dataset visualised using t-SNE. (**A**) Natural product classes. (**B**) Training, validation and test sets. (**C**) A training and test set with compounds from distant similarity clusters. (**D**) A training and test set where compounds from different natural product classes overlap in similarity.

7.4.3 Automated machine learning

Our experimental framework utilised the DeepMol AutoML engine (version 1.1.5) [267] to train all classifiers for multi-label classification available in *scikit-learn* version 1.2.0, employing molecular fingerprints (FPs) and descriptors as inputs on the training set, and evaluating their performance on the validation set. The models implemented include ridge classifiers, decision trees, random forests, and k-nearest neighbours, among others.

AutoML streamlines ML tasks by automating the search for the optimal pipeline, which integrates a feature set, feature selection, and classifier. The main goal during the optimisation process was to maximise the mF1 score on the validation set. We employed the Tree-structured Parzen Estimator (TPE) algorithm over 500 trials to determine the most appropriate hyperparameters and methods. Upon completing these trials, we identified the best pipeline and subsequently retrained it using the combined training and validation data to assess its performance on the test set.

7.4.4 Fingerprints for comparison

To benchmark our method, we compared it against several established molecular fingerprints (FPs) that have shown promise for NP classification and bioactivity prediction.

The following fingerprints were considered:

- MinHashed Fingerprints (MHFP) is a string-based FP algorithm that operates similarly to circular fingerprints, but instead of using atom identifiers, it uses specialised metabolite SMILES substrings as fragment identifiers. These identifiers are then hashed and stored in a fixed-size vector using the MinHash algorithm, as shown by Probst *et al.* [251]. A recent study also highlighted MHFP's effectiveness in NP bioactivity prediction [18].

- Neural Natural Product Fingerprints (NeuralNPFP) is a neural network-based molecular representation specifically designed for NPs. This FP encodes implicit NP-relevant information while maintaining details about their chemical space by training neural networks to differentiate between natural and synthetic compounds [244].

- NPClassifier Fingerprints are a variation of Morgan FPs designed for NP classification, where non-negative integers represent the count of repeated atomic substructures (or molecular fragments) rather than binary values. Morgan fingerprints represent molecular structures by encoding the substructures within a circular region around each atom into a fixed-length bit vector. These substructures are hashed into unique identifiers, with corresponding bits in the vector set to indicate the presence of specific atomic environments. The fingerprint's size and detail are controlled by adjusting the radius, which defines the neighbourhood size around each atom. NPClassifier FPs combine three Morgan FPs, each with different radii (0, 1, 2), to capture structural information at varying distances [245].

This set of FPs highlights the diversity of approaches in molecular representation, from neural network-based embeddings to substructure counting, which provides a strong foundation for assessing the strengths of our method.

To ensure a fair comparison, we implemented each FP in DeepMol and used its optimisation framework to identify the best combination of methods and models, while keeping the same fingerprint consistent across all trials. For each FP, we conducted 50 optimisation trials using the TPE algorithm, following the standard AutoML approach in DeepMol.

7.4.5 Statistical methods

We employed the Wilcoxon Signed-Rank (WSR) Test to evaluate differences in the metric values between the models for all the test sets. In this context, the samples consist of the metric values obtained by two models for each label within the multi-label classification framework. The objective is to determine the significance of any difference in the median value of a metric across all labels. This hypothesis test is formulated as follows. Given metric values for two models across n labels, m_{1i} and m_{2i}, calculate the

differences $d_i = m_{1i} - m_{2i}$ for each label i. For these differences, ignore $d_i = 0$ and rank the absolute differences $|d_i|$, assign ranks R_i and compute $W^+ = \sum_{d_i>0} R_i$ and $W^- = \sum_{d_i<0} R_i$, the test statistic W is defined as $W = \min(W^+, W^-)$. A p-value is calculated as the probability of observing a value of W_{ref}, determined by a reference distribution under the null hypothesis, as extreme as or more extreme than the observed value (W). The null hypothesis is that there are no significant differences between the median metric values between the two models. We considered a p-value lower than 0.05 sufficient to reject the null hypothesis.

Additionally, we applied the same test to each pair of models for the experimental setup outlined in [264], where cross-validation is conducted solely for predicting the precursors of alkaloids. This hypothesis test is formulated as follows. Given two models evaluated across n labels and r folds, resulting in performance metrics m_{Aij} and m_{Bij} for models A and B respectively, for each label i and fold j, we perform the following steps: calculate the differences $d_{ij} = m_{Aij} - m_{Bij}$, rank the absolute differences $|d_{ij}|$, and apply the Wilcoxon Signed-Rank Test as explained above.

7.4.6 Precursor predictions for LOTUS-DB plant compounds

After selecting the best model, we retrained it with the collected data (train, validation and test sets). Next, we downloaded all the data from LOTUS-DB and filtered the compounds annotated as plant compounds. We then predicted the precursors for all these compounds using the retrained model and analysed the main patterns of predictions by class of compounds, afterwards.

7.5 Data and code availability

The pipeline and models are available at the repository: SMPrecursorPredictor repository.

8

DeePlants, a Unified Platform for Plant Pathway Prediction

This chapter introduces DeePlants, an open-source, modular framework that brings together the enzyme-annotation and pathway-support methods developed across this thesis into one platform for plant retrobiosynthesis. Its design rests on a small set of abstract classes, Annotator, Reactor, Solution, and Searcher, that join chemical route prediction to biological feasibility: single-step backends (RetroRules, Retroformer, BioNaviNP) and a MEEA* planner suggest ranked precursor routes, while EC-number prediction, proteome-wide enzyme classification, reaction-EC annotation, and enzyme–substrate scoring judge whether a given proteome can actually catalyse each step. A coordinating layer, ProteomeCompass, folds these signals into a single cost that guides the search and converts plausible disconnections into shortlists of candidate enzymes.

We test the framework on two tomato flavour volatiles whose biosynthesis still hides an unassigned step, the glycosyltransferase/β-glucosidase pair behind 3-hydroxy-β-ionone and the lone reductase behind β-damascenone, run against the ITAG4.1 proteome. Fully automated, the one-shot pipeline is fragile: the single-step reaction sits at the head of a strict dependency chain, so one lumped or unbalanced transformation quietly spoils every prediction that follows. Driven by hand, with pathway enrichment pointing the way and each module run on its own, DeePlants instead recovers the right reactions and returns focused, annotated candidate lists, testable hypotheses for two flavour gaps the tomato literature leaves open.

Keywords: plants · pathway discovery · retrobiosynthesis · candidate enzymes

---

8.1 Introduction

Pathway enumeration and retrobiosynthesis methods are designed to identify feasible conversion routes between source and target metabolites [131, 305]. Although both aim at the same goal, they rely on different assumptions. Pathway enumeration approaches operate over an explicit metabolic network and therefore depend on the completeness and quality of that network. In contrast, retrobiosynthesis approaches do not require a predefined global network and can be applied more flexibly. As explained in

Section 2.5, these methods can be divided into template-based approaches, which use predefined reaction rules, and template-free approaches, which infer transformations directly from data; notably, template-free methods can leverage DL models to predict plausible substrates from a given product.

A major challenge across these methods is computational scalability. In pathway enumeration, algorithms must often handle branched and cyclic routes. In retrobiosynthesis, iterative expansion frequently creates highly branched search trees, leading to a combinatorial explosion in both candidate pathways and intermediate compounds. This rapidly increases runtime and memory requirements, making practical use difficult without careful filtering and prioritisation strategies.

These limitations motivate the development of a modular software framework that can integrate multiple retrobiosynthesis components in a unified workflow. A modular design enables users to combine search strategies, single-step expansion policies, enzyme availability and annotation, while keeping the system extensible as new models and algorithms emerge.

Assigning enzymes to each predicted reaction step is a critical part of this process. A chemically plausible transformation is not necessarily biologically feasible unless at least one enzyme can catalyse it under realistic conditions. Enzyme assignment therefore acts as a biological feasibility filter, reducing false-positive pathways and prioritising routes that are actionable for metabolic engineering. For these reasons, integrating reaction prediction with systematic enzyme assignment is essential for turning retrobiosynthesis outputs into experimentally testable pathway designs.

Based on these assumptions, we developed DeePlants, an open-source framework that integrates the methods created throughout this project into a single modular platform. DeePlants combines enzyme-annotation and pathway-support components. These include EC number prediction, proteome-level enzyme/non-enzyme classification, NP representation benchmarking, precursor prioritisation, and enzyme-substrate specificity analysis. These modules are designed for efficient inference, GPU support, and integrated visualisation of ranked suggestions.

In addition, DeePlants provides a modular interface for coupling retrobiosynthesis engines with the recommendation modules above. As a case study, we integrated MEEA* (MCTS-Exploration-Enhanced A*) [158], which incorporates the exploratory behaviour of Monte Carlo Tree Search into A* through look-ahead search (see section 2.5.3). This strategy, combined with path-consistency regularisation to improve heuristic generalisation, represents a state-of-the-art approach and identifies bio-retrosynthetic pathways for 97.68% of test NPs.

8.2 General implementation

DeePlants was designed as an open-source framework to address retrobiosynthesis for plant NPs through a modular, integrated approach. The software not only provides standalone tools but also supports interoperability among its components. Its architecture enables the execution of ML and DL tasks, model deployment, and component integration. The main modules of the package are illustrated in Figure 8.1.

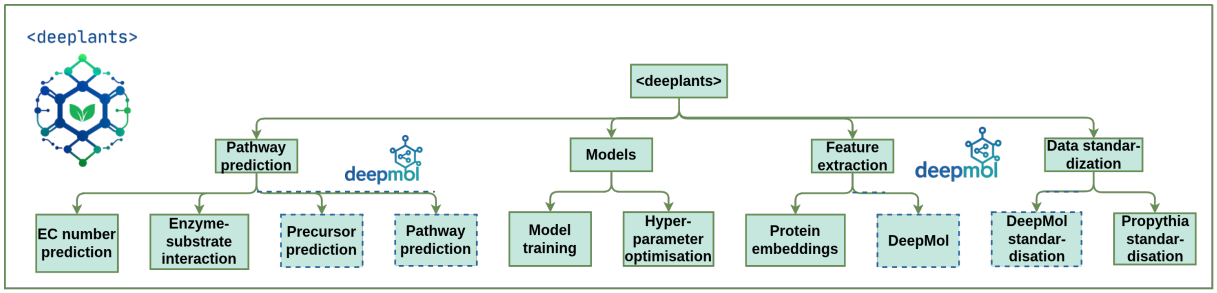


Figure 8.1: General architecture of DeePlants.

The framework includes methods for standardising and generating features from molecular and protein datasets, such as SMILES strings and amino acid sequences. Molecular data processing relies on DeepMol (see Sections 3.2.2 and 3.2.3), while protein standardisation uses the tool from Propythia [306].

Protein embeddings are generated with ESM2, ESM1b, and ProtBERT. ESM1b and ProtBERT were seamlessly integrated into the platform and can run on modest Graphical Processing Unit (GPU) hardware because they contain fewer Transformer parameters (650 and 420 million, respectively) than the largest ESM2 models. In contrast, inference with ESM2 models of 3 and 15 billion parameters is often not feasible on small GPUs due to VRAM limitations. To address this constraint, we implemented a multi-GPU architecture-parallelisation strategy inspired by [307]. In this setup, each GPU stores a distinct slice of model weights while the batch and activations are transferred across devices during forward and backward passes. This distributes memory-intensive Transformer and fully connected layers across available GPUs. Overall, the approach enables inference with larger models by trading additional inter-GPU communication for reduced per-device memory requirements.

At its core, DeePlants incorporates modules for model training with PyTorch and PyTorch Lightning, as well as hyperparameter and architecture optimisation via Optuna. These modules streamline the development of deep learning models. The remaining modules, depicted on the left in Figure 8.1, are specific to DeePlants and focus on inference tasks:

- **EC number prediction:** Provides models to predict EC numbers for enzymes and classify proteins as enzymes or non-enzymes.

- **Enzyme-substrate prediction:** Implements XGBoost-NPClassifier models, leveraging ESM2, ESM1b, and ProtBERT embeddings to predict enzyme-substrate specificity.

- **Precursor prediction:** Uses DeepMol to identify precursors from plant specialised metabolites.

- **Pathway prediction:** Integrates KEGG and PlantCyc pathway associations for known metabolites and supports *de novo* pathway prediction for metabolites with unresolved biosynthesis, including MEEA* execution.

8.3 Proteome annotation

For annotation of proteomes, as in predicting EC numbers and ESI prediction we developed an abstract *Annotator* class that forces the classes of annotation that extend it to implement methods such as *annotate*, *annotate_from_file* and *validate_input*. *validate_input* is always called before the annotation regardless of the method.

Figure 8.2 shows the classes that extend the *Annotator* class. *EnzymeAnnotator* annotates enzymes based on the prediction of EC numbers (see chapter 4) for proteins that are enzymes based on a binary classifier trained with the SwissProt data (see Figure S8.1) that is able to distinguish proteins that are enzymes from those that are not. For this, the three different embeddings that were benchmarked in chapter 4 were used: ProtBERT, ESM2 and ESM1b. The method accepts as input a list of amino-acid sequences, a FASTA file or a pandas dataframe.

Moreover, the models developed in chapter 6 were integrated in a single class called *ESIAnnotator*, extend the class *Annotator* and predict the specificity of enzymes and their respective substrates in a binary classification setting. They all compute NPClassifierFP features (see chapter 5) and protein embeddings from the activations of the last hidden layer of the models developed in chapter 4. The method accepts as input a pandas dataframe with enzyme-substrate pairs.

Finally, *ReactionECAnnotator* extends the model's functionality by annotating the EC number of a given reaction. This module was designed to prioritise deployment efficiency and practical usability, rather than achieving state-of-the-art performance metrics. The model was trained on the comprehensive *ECREACT* dataset [280, 308], which integrates reaction-EC pairs from multiple sources, including Rhea [309], BRENDA [310], PathBank [311], and MetaNetX [312]. The dataset, initially comprising 62,222 entries, was refined by the authors by excluding EC numbers with fewer than 10 reaction entries, resulting in 61,817 high-quality EC-reaction pairs. These pairs span seven 1st-level, 63 2nd-level, and 175 3rd-level EC numbers. To enhance robustness, data augmentation was applied by the authors by shuffling reactants and products, approximately tripling the size of the training (n = 150,226), validation (n = 16,692), and testing (n = 18,816) sets. We ran a small experiment by training this model with the same partition, obtaining 89% accuracy, very close to CLAIRE [308].

For this small experiment, we conducted a workflow using chemprop [313] for preparing and featurizing chemical reaction data to annotate EC numbers, using advanced ML techniques tailored for chemoinformatics. The process started with the encoding of categorical EC numbers into numerical values. Each chemical reaction, encoded as a SMILES string, was parsed into a structured data object that encapsulates both the reaction's molecular structure and its corresponding EC number label. Atom-level features, such as atomic number, degree, formal charge, chirality, and hybridization, were systematically encoded into a multi-hot binary vector, specifically optimised for organic molecules. These atom features were then used to construct a condensed graph representation of each reaction. The condensed graph representation is a specialised approach that transforms a chemical reaction into a unified graph structure, where both reactants and products are integrated into a single, connected graph. In this representation, the reactant and product graphs are superimposed into a single graph, and bonds that change during the

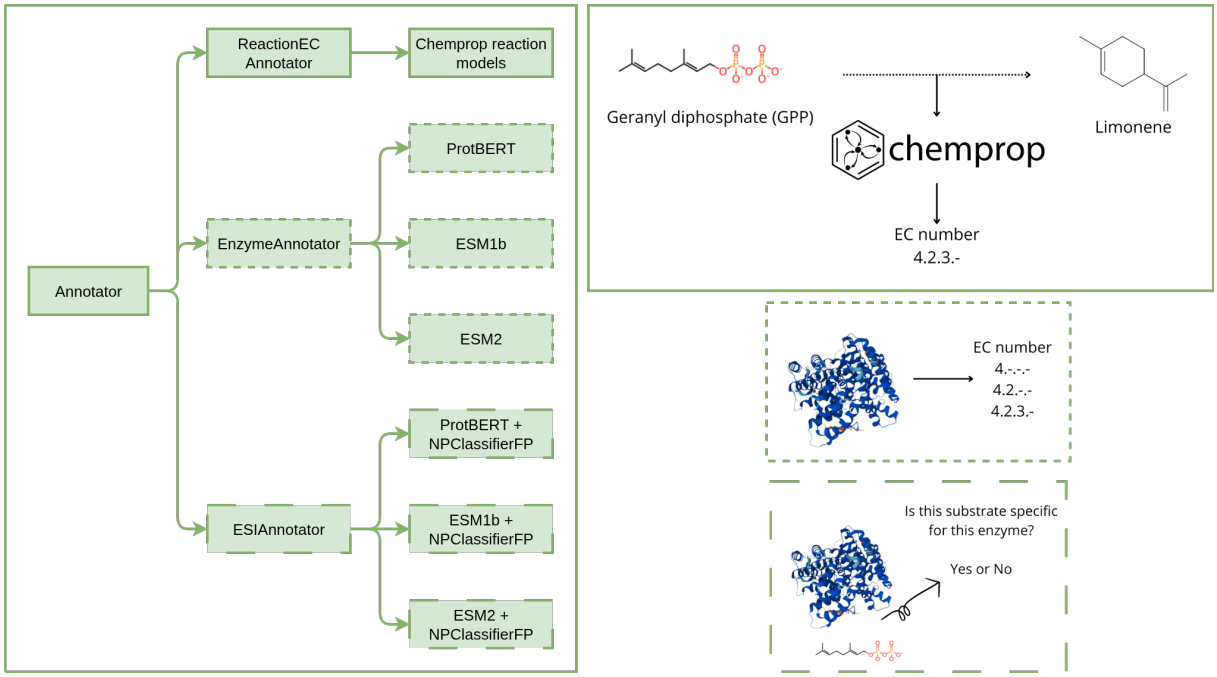


Figure 8.2: Annotator and classes that extend them. On the left, the diagram presents the most abstract classes, transitioning to more specific ones as you move right. On the right, examples of second-level classes (e.g., EnzymeAnnotator) illustrate the scientific questions they address. Each example is visually linked to its corresponding class by matching line styles.

reaction are annotated with dynamic-bond labels encoding their formation or cleavage. This approach effectively highlights the structural changes occurring during the reaction, preserving the mapping of atoms from reactants to products and emphasising the reaction centre. By capturing these transformations in a graph format, the model can more effectively learn the underlying chemical rules and patterns of the reaction, rather than treating reactants and products as isolated entities. The featurized reactions were subsequently compiled into a cohesive dataset, designed to facilitate efficient training and evaluation. For comparison, we trained and tested this model and compared with the state-of-the-art using the same test set.

8.4 Single-step retrosynthesis prediction

For modular retrosynthesis and biocatalytic route planning we developed an abstract *Reactor* class that forces the concrete backends that extend it to implement a method *react*. This method receives a list of either *rdkit* molecules or SMILES strings as input and applies the different algorithms to predict reactants of a reaction based on the list of products. This *Reactor* is the equivalent of single-step expansion policies in retrobiosynthesis searchers/planners.

Figure 8.3 shows the three backend classes connected to the abstract Reactor. The top branch is handled by RetroRules [314], a template/rule-driven engine that applies curated reaction rules to the target

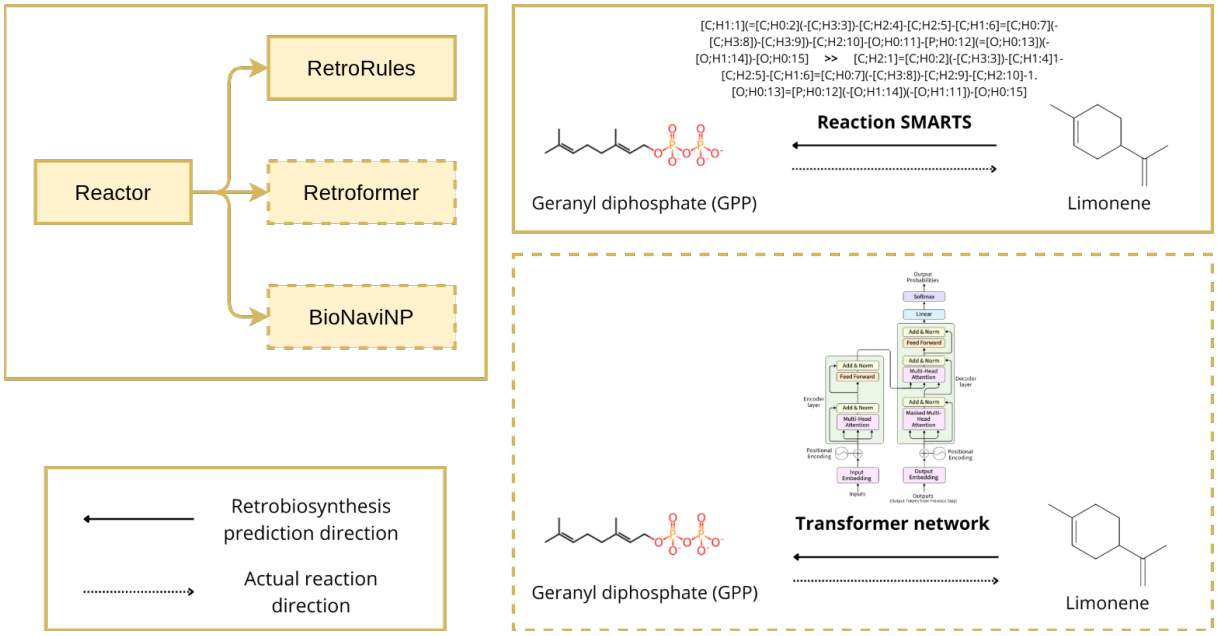


Figure 8.3: The diagram starts with the abstract Reactor class on the left, progressing to more specific implementations (e.g., classes that define concrete methods) toward the right. On the right side, examples of these specific classes (e.g., Retroformer) highlight the scientific questions they address, with each example visually matched to its class through consistent line styles.

molecule (illustrated in the upper right panel: a target structure on the left and a predicted disconnection/precursor on the right). The middle branch is implemented by Retroformer [315], and BioNaviNP [297], sequence-to-sequence Transformer-style models (schematic shown in the centre-right panel) that translates product SMILES into probable reactant SMILES; the figure includes the model architecture and example input to output translation.

Each backend returns ranked precursor sets together with confidence scores and metadata (predicted transformations, required reagents/enzymes when available). This modular design allows the *Reactor* to route a single target through multiple strategies and to aggregate or re-score results downstream, enabling flexible retrosynthesis and biosynthetic-route planning workflows.

8.5 Solutions and Searchers

For the aggregation and standardised reporting of retrosynthetic, enzyme-class and enzyme–substrate inference, we developed an abstract *Solution* class that forces concrete solution types that extend it to implement methods such as *to_csv* and *to_json* that export the solutions into a CSV format file or JSON.

Figure 8.4 shows the concrete solution classes that extend the abstract *Solution* class and the structured outputs each produces. *ReactionSolution* collects the chemistry-centred outputs of a disconnection or reaction proposal: ranked *Reactants*, the derived *Products* and any associated *EC numbers* that were predicted or recommended for biocatalytic steps. *ECSolution* collects enzyme-class related information used to map enzymatic function back to biological actors: hierarchical *Levels* and the *Proteins*

to EC mappings that link protein identifiers to their predicted EC annotations. *ESISolution* contains the enzyme–substrate inference results, returning candidate *Substrate–enzyme matches* (enzyme identifiers, match scores and relevant metadata). Finally, *SearcherSolution* aggregates route-search outputs as ordered *Routes* (each route is an ordered list of intermediate molecules and the transformations connecting them).

The right-hand panel illustrates the search/planning orchestration component: a *Searcher* object holds a list of reactors (reactors: List[*Reactor*]) and exposes abstract methods (`@abstractmethod search` and `@abstractmethod run_reactor`) that concrete search strategies must implement; these methods accept a query *molecule* as input and coordinate calls to the available reactors. Searcher is therefore responsible for exploring the search space (target → intermediate molecules → final precursors) and assembling the corresponding *Solution* objects.

At the pipeline level (lower right) the figure shows how a *Searcher* expands a *target* into a sequence of candidate molecules (mol 1 ... mol n), then collects the chemical and biological evidence (visualised as rendered 3D structures in the result panel) into *Solution* instances that downstream tools or users can inspect, filter or re-score. This modular design decouples result representation from search strategy and reactor implementation, enabling interchangeable search policies and straightforward downstream analysis.

8.6 DeePlants algorithms

The diagrams in Figure 8.5 show how chemical-route proposals and protein-scale annotations are integrated into DeePlants for enzyme-informed retrosynthesis and pathway discovery. Two iterations of the algorithm are presented: an initial version (Figure 8.5, excluding optional steps 5 and 6), which identifies candidate enzymes without using their annotations to score transformations, and a refined version (Figure 8.5, including steps 5 and 6), which aggregates these annotations into a single-step cost passed to the search planner.

Across the top of both diagrams is a small-pathway inset (example of transformation) that represents the type of chemistry handled by the system; these transformations are encapsulated as *ReactionSolution* objects. In the second iteration, these solutions are processed within a central orchestration component, *ProteomeCompass*, which aggregates and coordinates all chemical and biological information.

The biological annotation pipeline is shared between both versions and operates on proteomic data (e.g., FAA files). Protein sequences are processed by the *EnzymeAnnotator* (step 1.1), producing *ECSolution* objects that map proteins to predicted EC numbers (step 1.2).

In parallel, the orchestration layer incorporates a search module implemented by the *Searcher*, which explores candidate transformations using one or more *Reactor* objects (step 2.1 and 2.2). These reactors generate and evaluate precursor candidates, returning ranked chemical proposals as *ReactionSolution* instances. The *ReactionSolution* serves the *ReactionECAnnotator* that predicts reaction EC up to the third level (step 2.3). These annotations are integrated with enzyme-level information (step 1.2) through the

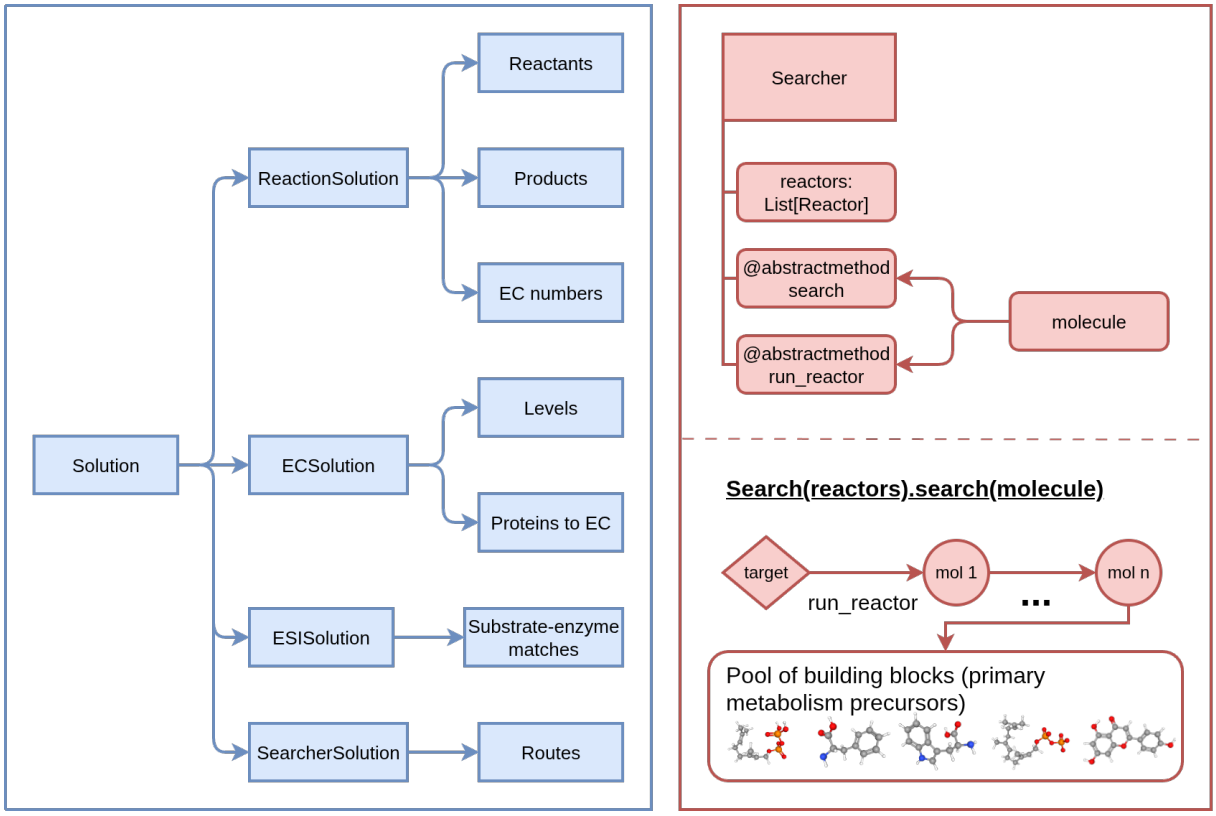


Figure 8.4: Solution classes and search orchestration architecture. Left panel: concrete solution classes extending abstract *Solution* with their outputs - *ReactionSolution* (reactants, products, EC numbers), *ECSolution* (enzyme hierarchies, protein-EC mappings), *ESISolution* (substrate-enzyme matches), and *SearcherSolution* (ordered synthetic routes). Right panel: *Searcher* orchestrates search strategies by coordinating calls to multiple reactors, exploring the search space from target to precursors and aggregating results into *Solution* objects. Lower right: pipeline schematic showing target expansion into candidate molecules with assembled chemical and biological evidence returned as filterable *Solution* instances.

AnnotatorLinker, which associates enzymatic functions with candidate transformations by EC number match. Since these solutions are specific only up to the third level of EC number and lack substrate specificity, they are used to query the *ESIAnnotator* about enzyme-substrate compatibility (step 3.1 and 3.2), which produces *ESISolution* objects. These predictions are consolidated into a list of matched enzymes, representing candidate catalysts compatible with the proposed chemistry.

There are two workflows, the workflow (I), that terminates at this point: the matched enzymes from the *ESISolution*, together with the *ECSolution* produced by the *EnzymeAnnotator*, are simply collected into a final list of candidate enzymes that catalyse the reaction. The biological annotations are thus used to surface candidate catalysts but do not feed back into the search itself, meaning the single-step cost guiding the planner is determined purely by the chemistry-driven evaluation performed by the *Reactor*.

The second workflow extends this pipeline by introducing the *ProteomeCompass* as a closing orchestration layer. Rather than outputting a static list of candidates, the solutions are iteratively refined and

accumulated within the *ProteomeCompass*, which averages the costs derived from EC numbers, enzyme-substrate specificity, and reaction predictions. This aggregated value is passed as the single-step tree policy cost to the searcher/planner (step 4.1, 5 and 6), enabling systematic exploration of the chemical search space guided jointly by chemical plausibility and enzymatic feasibility.

Overall, the figure highlights two complementary components present in both iterations: a chemistry-driven exploration of possible reaction routes and a biology-driven evaluation of enzymatic feasibility. The key distinction lies in how these components interact: the first iteration treats them as parallel pipelines that converge into a list of candidate enzymes, whereas the second couples them through the *Proteome Compass*, integrating outputs from *ReactionSolution*, *ECSolution*, and *ESISolution* into a unified score that prioritises pathways which are not only chemically plausible but also supported by candidate enzymes present in the input proteome.

Although DeePlants is an integrated system that combines chemical and proteome-level information, MEEA* and any search algorithm extending the *Searcher* class can be executed independently of the full annotation workflow. Additionally, users can run the precursor prediction described in Chapter 7 to identify potential precursors or building blocks from primary metabolism. The algorithm can then be directed to target these compounds for synthesis.

8.7 Showcase demo

8.7.1 One molecule-shot retrobiosynthesis and pathway enrichment analysis

To illustrate how the components described throughout this chapter operate as a single workflow, we deployed DeePlants as a web platform and applied it to 3-hydroxy-β-ionone β-D-glucopyranoside. The platform exposes the framework's inference modules through a small set of views, and a single submission of the target SMILES drives every prediction shown in Figure 8.6.

Figure 8.6 presents the compound-level analysis view. From the target compound structure alone, the precursor prediction module (Chapter 7) proposes probable building blocks from primary metabolism (shown as the *probable precursors* tags), while the de-novo retrobiosynthesis engine returns a ranked, multi-step route in which each disconnection is labelled with a predicted EC number and a confidence score (e.g. EC 4.2.3 and EC 1.14.13). These steps are produced by the *Searcher/Reactor* machinery of Section 8.5, expanding the target through single-step backends. In parallel, the pathway-prediction module maps the metabolite to its biosynthetic context using KEGG and PlantCyc associations, recovering the apocarotenoid branch (PlantCyc `PWY-5397`, safranal/apocarotenoid biosynthesis), and the NP-Classifier panel places it in the natural-product taxonomy as a glycosylated apocarotenoid of the megastigmane class. The view therefore answers, from a structure alone, both how the metabolite is likely produced and what it is chemically.

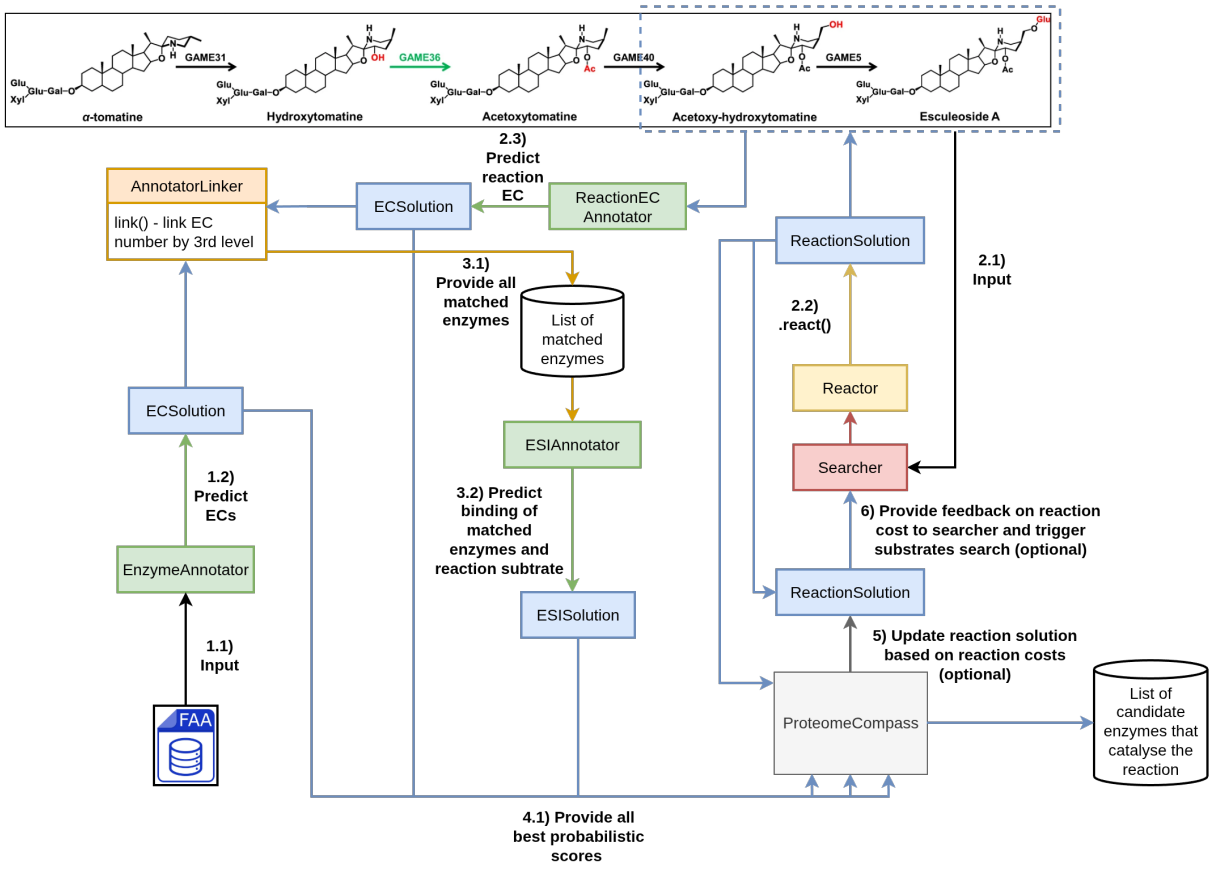


Figure 8.5: DeePlants algorithm. Integrated enzyme-informed retrosynthesis framework combining chemical-route proposals with proteome-scale annotations. Example transformations shown at the top of each panel are encoded as *ReactionSolution* objects. Two algorithmic iterations are shown. (I) Initial workflow, excluding optional steps 5 and 6, in which candidate enzymes are identified without using their annotations to score chemical transformations. (II) Refined workflow, including steps 5 and 6, in which *ProteomeCompass* aggregates enzyme annotations into a single-step cost that is passed back to the search planner by coordinating: (1) biological annotation - protein sequences processed by *Enzyme Annotator* to generate EC predictions (*ECSolution*); (2) chemical search -*Searcher* and *Reactor* objects generate ranked precursor proposals (*ReactionSolution*), which are annotated by *ReactionECAnnotator*; (3) substrate specificity - *AnnotatorLinker* associates enzymatic functions by EC match, then *ESIAnnotator* predicts enzyme-substrate compatibility (*ESISolution*); (4) cost integration - *ProteomeCompass* averages costs across EC, enzyme-substrate, and reaction predictions; (5) solution refinement - the resulting *ReactionSolution* is updated to reflect the aggregated reaction cost; (6) search feedback - the aggregated cost is propagated back to the *Searcher* as a single-step tree-policy cost, triggering the next substrate search and guiding the exploration of the chemical space. The modular design enables independent execution of chemical search (*MEEA**) and optional precursor identification from primary metabolism, while maintaining chemical plausibility and enzymatic feasibility constraints.

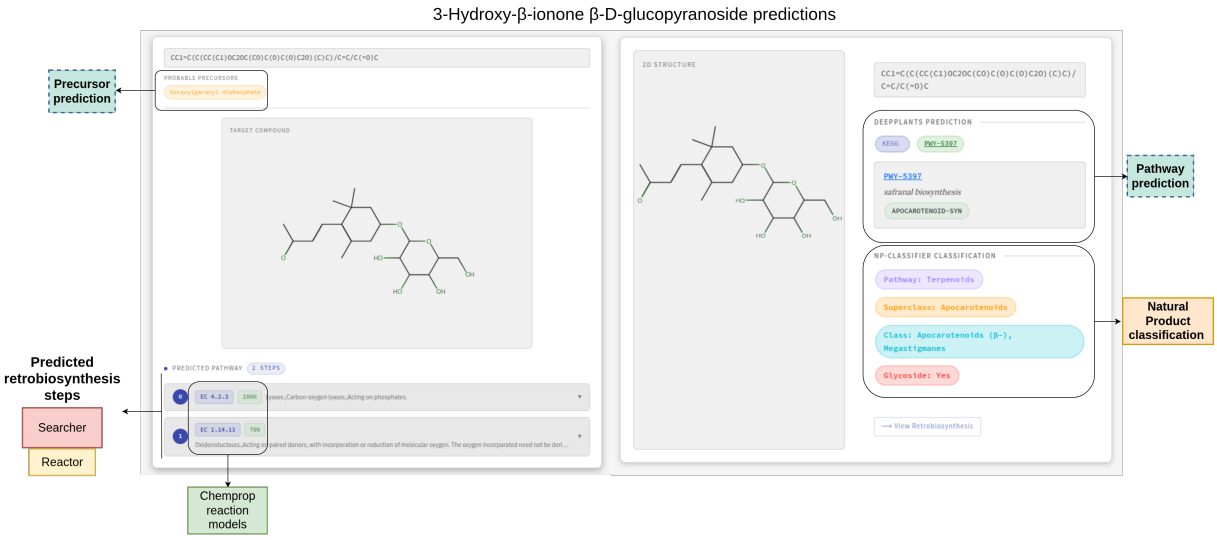


Figure 8.6: Compound-analysis view of the DeePlants platform for 3-hydroxy-β-ionone β-D-gluco pyranoside. Left: the target structure, module-predicted probable precursors, and the ranked retrobiosynthetic route, where each step carries a predicted EC number and confidence score produced by the *Searcher*/*Reactor* backends. Right: pathway prediction from KEGG and PlantCyc annotations (apocarotenoid branch, `PWY-5397`) and NP-Classifier classification of the compound as a glycosylated apocarotenoid megastigmane.

8.7.2 Finding candidate enzymes for reactions

Figure 8.7 shows the retrobiosynthesis view, where the predicted route is coupled to candidate-enzyme discovery and the glucosylation step (EC 2.4.1, *Transferases / Glycosyltransferases / Hexosyltransferases*) as expandable into a full reaction record: the reactant and product structures, the reaction SMILES, and a ranked list of candidate enzymes drawn from the input proteome. The enzyme card is assembled by the annotation stack described in section 8.3: the *EnzymeAnnotator* screens the proteome and predicts EC numbers for enzymes, the *ReactionECAnnotator* assigns an EC class to the proposed reaction, and the *ESIAnnotator* scores enzyme-substrate compatibility, with the *ProteomeCompass* consolidating these signals into the displayed candidates and surfacing UniProt-level metadata (Gene Ontology terms such as *UDP-glucosyltransferase activity*, sequence length, molecular weight and cross-references). Notably, the platform returns candidate UDP-glucosyltransferases, demonstrating how DeePlants turns a chemically plausible route into a shortlist of experimentally testable enzyme hypotheses, and closing the loop between the chemistry- and biology-driven components of Figure 8.5.

8.8 Tasty molecules – use cases

In this section, we develop two apocarotenoid volatiles from tomato flavour chemistry as motivating use case (UC)s. Both UCs share the same evaluation configuration: a chemically well-defined volatile endpoint whose biosynthetic route contains enzymatic steps that have not been biochemically assigned. However, they isolate complementary gaps. The 3-hydroxy-β-ionone case leaves an unmapped glycosyltransferase,

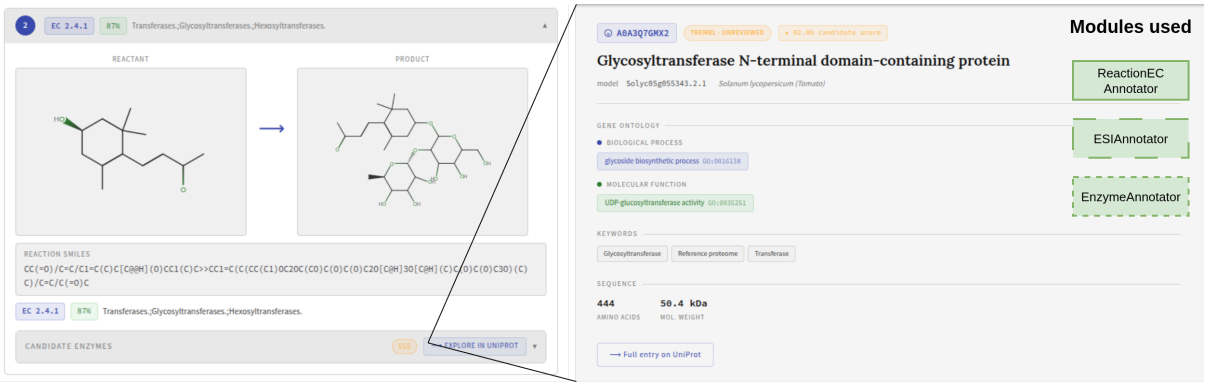


Figure 8.7: Retrobiosynthesis and candidate-enzyme view. The glucosylation step (EC 2.4.1) of the predicted route is expanded into a full reaction record, reactant and product structures and reaction SMILES, together with candidate enzymes from the input proteome. The annotation stack of Section 8.3 populates the enzyme card: *EnzymeAnnotator* (proteome EC prediction), *ReactionECAnnotator* (reaction EC assignment) and *ESIAnnotator* (enzyme-substrate specificity), consolidated by *ProteomeCompass* and linked to UniProt-level metadata such as the *UDP-glucosyltransferase activity* Gene Ontology term.

whereas the β-damascenone case leaves a single unmapped reductase.

The first oxygenated derivative considered is 3-hydroxy-β-ionone. This compound is a C_{13} apocarotenoid (norisoprenoid) [316] that arises from oxidative cleavage of the C_{40} carotenoid pool of the tomato chromoplast. In red-ripe *Solanum lycopersicum* fruit, the parent ketone β-ionone is one of the apocarotenoid volatiles. It is derived from β-carotene through C9-C10 cleavage by the carotenoid cleavage dioxygenases LeCCD1A and LeCCD1B [317]. *In vitro* assays of the same enzymes show that they also cleave 3-hydroxylated apocarotenoid substrates. This reaction releases oxygenated C_{13} products together with their corresponding dialdehydes [318]. The biological pathway therefore links hydroxylated xanthophylls of the β,β-branch, such as zeaxanthin and β-cryptoxanthin, to 3-hydroxy-β-ionone through carotenoid cleavage activity. This aglycone has been detected in the free volatile fraction of red-ripe tomato fruit in the cultivars Alida and Merve. It occurs alongside the dominant C_6 aldehydes and the C_{13} apocarotenoid background typical of the species [319].

Beyond its occurrence as a free volatile, 3-hydroxy-β-ionone is also relevant to the tomato glycoside pool. In the same cultivar survey, norisoprenoids of the β-ionone series were recovered from the solid-phase-extracted glycosidic fraction and identified after enzymatic hydrolysis [319]. This is consistent with the general logic of fruit aroma. Many volatiles accumulate as non-volatile, flavourless O-glycosides and are released as free aglycones only after tissue disruption [320]. Despite this evidence, no tomato enzyme has been biochemically assigned to the conjugation of 3-hydroxy-β-ionone into its corresponding glucoside. The cleavage step is partially mapped through LeCCD1A and LeCCD1B [317, 318]. The structurally adjacent diglucosylation of phenolic volatiles is characterised through NSGT1 [321]. However, the central conjugation step of this apocarotenoid glycoside branch remains unmapped.

The same apocarotenoid logic, taken one step further along the β,β-branch, produces the second and more sharply defined case: β-damascenone. This compound is a C_{13} norisoprenoid [322] and

one of the most potent carotenoid-derived odorants known. Its odour-detection threshold lies in the low parts-per-trillion range [323]. β-Damascenone is a natural constituent of tomato. During the thermal concentration of fruit into paste, it becomes one of the highest odour-activity, character-impact volatiles of the product [324]. The abundant but labile green-note C_6 aldehydes are driven off during processing. At the same time, additional aglycone is liberated from the glycoside pool. As a result, the sensory weight of β-damascenone and its C_{13} relatives rises sharply. It is therefore regarded as one of the signature odorants of authentic, high-quality cooked tomato paste [324, 325].

Unlike the ionones, β-damascenone is not released by a single cleavage event. Instead, it is the endpoint of a multi-step degradation. Oxidative cleavage of 9′-cis-neoxanthin at the 9,10 (9′,10′) double bond yields the C_{13} ketodiol grasshopper ketone. This activity has been demonstrated *in vitro* for the carotenoid cleavage dioxygenases RdCCD1 and AtCCD1 [326, 327]. It is also consistent with the relaxed double-bond specificity of the tomato orthologues LeCCD1A and LeCCD1B [318]. Grasshopper ketone is then reduced to the allenic triol megastigma-6,7-dien-3,5,9-triol. This triol is stored as its non-volatile 9-O-β-D-glucopyranoside and was identified as the direct damascenone precursor in *Rosa damascena* [328]. Under the mildly acidic conditions of ripe fruit and paste manufacture, the triol undergoes acid-catalysed dehydration and 1,3-transposition of oxygen. This process releases free β-damascenone [323].

In this branch, the entry cleavage is assigned to characterised CCDs [318, 326, 327]. The terminal triol-to-damascenone conversion is also well described as a non-enzymatic, acid-catalysed rearrangement [323]. What remains unassigned is the single oxidoreductase that converts grasshopper ketone into the triol. No candidate enzyme has been proposed for this step [329], and the biosynthesis of β-damascenone as a whole is still described as unresolved [330]. This makes the reaction an unusually clean target for sequence-based annotation. It has a defined substrate, a defined product, and a single intervening reaction of predictable mechanistic class. Specifically, the expected chemistry is a carbonyl reduction, corresponding to an EC 1.1.1-type short-chain dehydrogenase/reductase or aldo-keto reductase. The search can therefore focus on the oxidoreductase complement of the tomato genome.

The practical interest of resolving either branch derives from the flavour relevance of the apocarotenoid pool. The C_{13} volatiles of the β-ionone series, together with geranylacetone and 6-methyl-5-hepten-2-one, contribute fruity and floral notes that are consistently associated with preferred tomato cultivars [331]. Tieman et al. [332] further demonstrated that apocarotenoid volatiles enhance perceived sweetness independently of sugar concentration. Carotenoid-biosynthesis mutants deficient in these volatiles, but with unaltered sugars and acids, are rated as less sweet by consumers. As a hydroxylated derivative of the β-ionone scaffold, 3-hydroxy-β-ionone contributes to the norisoprenoid fingerprint that distinguishes flavourful from bland cultivars [331]. In contrast, β-damascenone carries disproportionate sensory weight in the cooked product because of its very low threshold [324, 325]. In both cases, the glycoside-bound counterpart acts as a latent reservoir. Its enzymatic release during chewing and processing shapes retronasal aroma.

Assigning enzymes to either branches would therefore have two benefits. A UGT (UDP-glucosyl transferase) assigned to the 3-hydroxy-β-ionone branch would provide a candidate target for breeding and metabolic engineering of cultivar flavour. The same would be true for an oxidoreductase assigned to

grasshopper-ketone reduction in the β-damascenone branch. In both cases, the assigned enzyme would also provide a labelled positive data point for validating the predictive accuracy of the framework. Together, these two tomato apocarotenoids test the framework across complementary reaction classes: conjugation and carbonyl reduction. They are therefore not only useful benchmarks, but also cases of independent biological interest.

8.8.1 One-shot genome-guided retrobiosynthesis and pathway associations

We applied the algorithm presented in Figure 8.5 using Retroformer as a reactor, ProtBERT as ESIAnnotator and EnzymeAnnotator, and MEEA* as Searcher. The used genome was ITAG4.1, the gene annotation version for the tomato genome (*Solanum lycopersicum*) released by the International Tomato Annotation Group (ITAG).

Use case 1

Having examined the single-step prediction behaviour in isolation, we now report the full one-shot, genome-guided run for the 3-hydroxy-β-ionone β-D-glucopyranoside target, in which a single submission of the target SMILES drives both the pathway-association analysis and the *de novo* retrobiosynthetic search against the ITAG4.1 proteome.

Figure 8.8 shows the pathway-enrichment and compound-classification output. From the target structure alone, the pathway-prediction module recovers the apocarotenoid branch, associating the metabolite with PlantCyc `PWY-5397` (safranal/apocarotenoid biosynthesis, tagged `APOCAROTENOID-SYN`). This recovers the biologically relevant logic for the compound, zeaxanthin degradation, followed by UDP-glucosyl transferase glycosylation to the apocarotenoid glycoside, and thermal release of the free apocarotenoid, which is precisely the route expected from the literature reviewed above. In parallel, the NP-Classifier panel assigns the compound to the *Terpenoids* pathway, the *Apocarotenoids* superclass and the *Apocarotenoids (β-), Megastigmanes* class, and correctly flags it as a glycoside. The classification and pathway association are therefore both correct, confirming that the framework places the metabolite in the right biosynthetic and chemical context before any route is enumerated.

Figure 8.9 presents the corresponding *de novo* route returned by the one-shot search. The planner reaches the target in two displayed steps, departing from a GGPP precursor. The first disconnection is annotated as EC 4.2.3 (carbon-oxygen lyase) with full confidence (100%) and surfaces 47 candidate enzymes; chemically, however, this single arrow compresses several real biosynthetic events (cyclisation, oxidative cleavage and oxidation) into one transformation, the step does not correspond to an elementary, mass-balanced reaction. The second disconnection, from 3-hydroxy-β-ionone to the glucoside target, is annotated as EC 1.14.13 (a monooxygenase) with 70% confidence and 30 candidate enzymes; this prediction is incorrect on three counts. The assigned EC class does not match the chemistry actually required, the step is not mass-balanced because the glucose moiety is absent from the reactant side, and

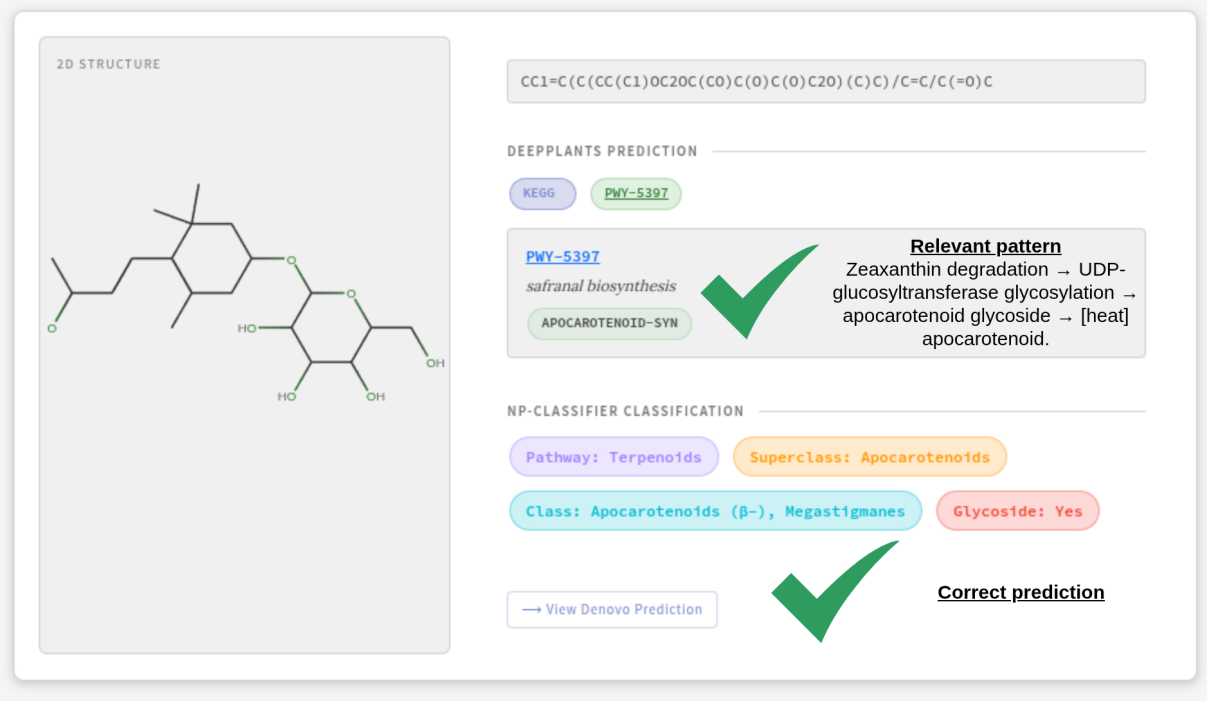


Figure 8.8: Pathway enrichment and compound-classification view for 3-hydroxy-β-ionone β-D-gluco pyranoside. The pathway-prediction module associates the target with PlantCyc `PWY-5397` (safranal/ apocarotenoid biosynthesis, `APOCAROTENOID-SYN`) and a KEGG entry, recovering the relevant biological pattern (zeaxanthin degradation ⟶ UDP-glucosyltransferase glycosylation ⟶ apocarotenoid glycoside ⟶ thermal release of the apocarotenoid). The NP-Classifier panel correctly assigns the *Terpenoids* pathway, *Apocarotenoids* superclass, *Apocarotenoids (β-), Megastigmanes* class and glycoside status.

the route conflates the glycosylation and the 3-hydroxylation into a single arrow rather than resolving them as the distinct UDP-glucosyltransferase (EC 2.4.1) and hydroxylation events that the biology requires.

Use case 2

We repeated the one-shot, genome-guided run for the β-damascenone precursor, megastigma-6,7-dien-3,5,9-triol, again submitting a single target SMILES to drive both the pathway-association analysis and the *de novo* retrobiosynthetic search against the ITAG4.1 proteome.

Figure 8.10 shows the pathway-enrichment and compound-classification output. From the target structure alone, the pathway-prediction module associates the allenic triol with PlantCyc `PWY-695` (ABA synthesis, tagged `Abscisic-Acid-Biosynthesis`) and a KEGG entry, recovering the biologically relevant logic for this branch, neoxanthin degradation followed by apocarotenoid reduction, which matches the grasshopper-ketone-to-triol reduction expected from the literature reviewed above. The NP-Classifier panel correctly assigns the *Terpenoids* pathway, the *Apocarotenoids* superclass and the *Apocarotenoids (β-)* class, and correctly flags the compound as a non-glycoside. As before, the classification and pathway association are both correct, placing the metabolite in the right biosynthetic and chemical context before any route is enumerated.

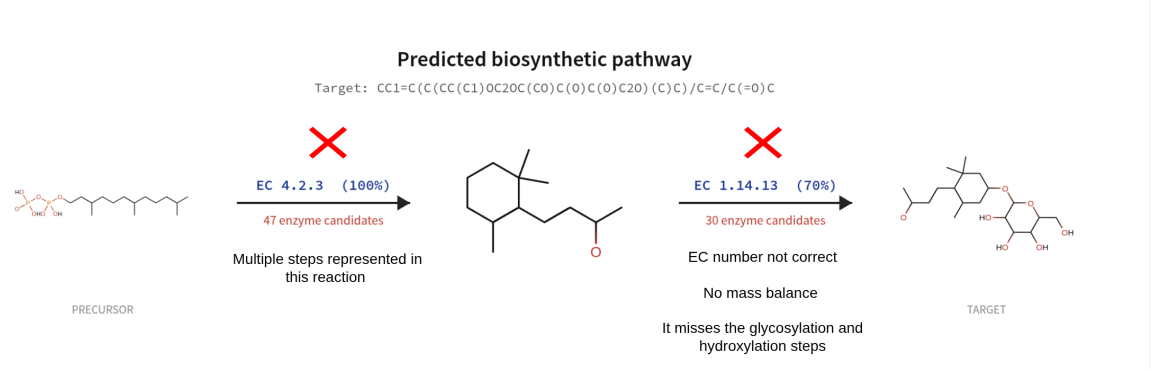


Figure 8.9: One-shot genome-guided retrobiosynthetic route for 3-hydroxy-β-ionone β-D-glucopyranoside, obtained with Retroformer as *Reactor* and MEEA* as *Searcher* against the ITAG4.1 proteome. The route reaches the target in two displayed steps from a prenyl-diphosphate precursor. The first step (EC 4.2.3, 100%, 47 enzyme candidates) compresses several biosynthetic events into one transformation. The second step (EC 1.14.13, 70%, 30 enzyme candidates) is mispredicted: the assigned EC class is incorrect, the reaction is not mass-balanced (the glucose moiety is missing on the reactant side), and it merges the glycosylation and 3-hydroxylation into a single arrow.

Figure 8.11 presents the corresponding *de novo* route returned by the one-shot search. Here the planner reaches the triol target in a single displayed step from a GGPP precursor, annotated as EC 5.4.99 (an intramolecular isomerase) with 91% confidence and 32 candidate enzymes. This step is incorrect on two counts. First, as in UC 1, the single arrow compresses the entire multi-step degradation, cyclisation, oxidative cleavage and the terminal carbonyl reduction, into one transformation rather than resolving the elementary reaction. Second, the assigned EC class is wrong: the chemistry actually required to reach the triol from grasshopper ketone is a carbonyl reduction (EC 1.1.1-type oxidoreductase), not an isomerisation. This exactly mirrors the misassignment already seen in the single-step analysis (Figure 8.13), where the correct reduction was also mislabelled as EC 5.4.99.

8.8.2 Single-step prediction of glycosylation for flavour lock – UC 1

We started by running the single-step prediction of the glycosylation of 3-hydroxy-β-ionone β-D-gluco pyranoside with Retroformer and applied our ProteomeCompass method to find candidate enzymes to the step using the EC number prediction using ProtBERT as EnzymeAnnotator and ESIAnnotator.

Figure 8.12 shows the top six single-step predictions produced by Retroformer. The first solution proposes an isomeric 3-hydroxy-β-ionone as reactant and 3-hydroxy-β-ionone β-D-glucopyranoside as product. However, this reaction is not mass-balanced because β-D-glucose is absent from the reactant side. The same issue occurs in solutions 3 and 6. Solutions 4 and 5 are chemically suggestive, but still incorrect, because they use a glucosylated zeaxanthin-like reactant. This is close to the expected biosynthetic logic, since 3-hydroxy-β-ionone can arise from zeaxanthin cleavage. Nevertheless, this route is not documented and is unlikely to represent the specific glucosylation step considered here. Solution 2 is the

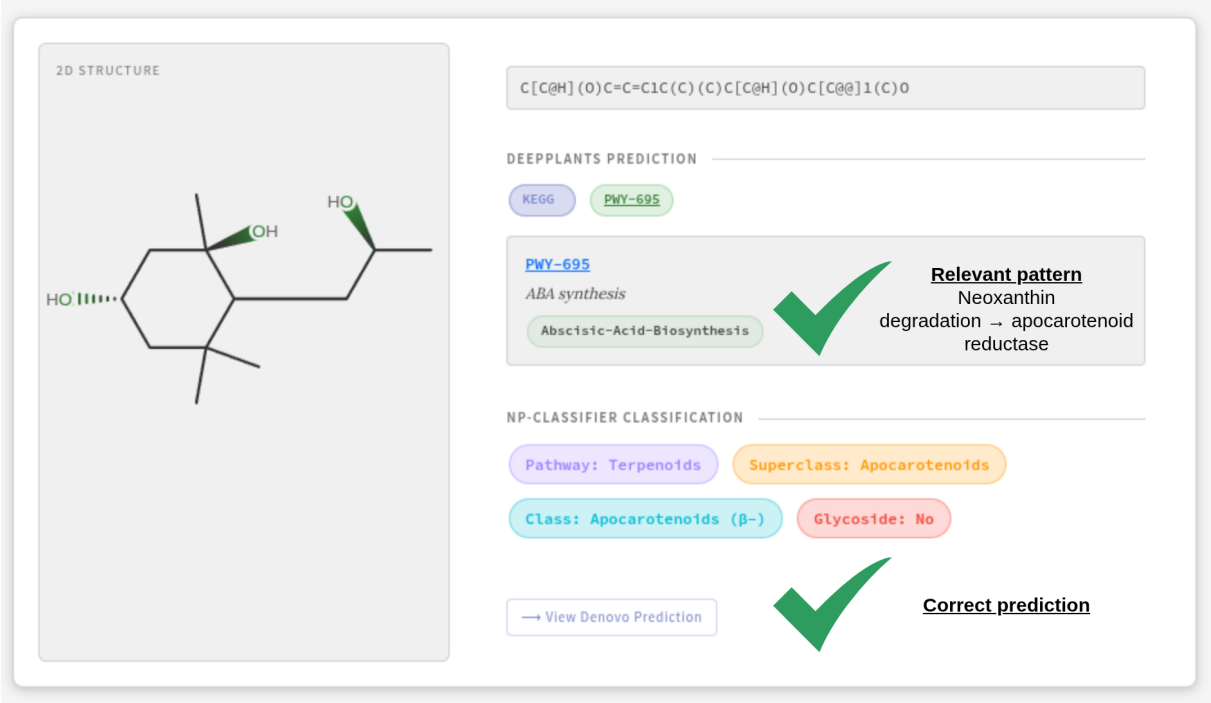


Figure 8.10: Pathway enrichment and compound-classification view for megastigma-6,7-dien-3,5,9-triol. The pathway-prediction module associates the target with PlantCyc `PWY-695` (ABA synthesis, `Abscisic-Acid-Biosynthesis`) and a KEGG entry, recovering the relevant biological pattern (neoxanthin degradation ⟶ apocarotenoid reductase). The NP-Classifier panel correctly assigns the *Terpenoids* pathway, *Apocarotenoids* superclass and *Apocarotenoids (β-)* class, and correctly flags the compound as a non-glycoside.

best-supported prediction because it is mass-balanced and corresponds to the glucosyltransferase reaction expected from the literature.

We further used the correct solution. Its reaction EC was mispredicted as 4.1.2 (an aldolase, with a very low score of 21%), where it should be a UDP-glucosyltransferase (EC 2.4.1). Upon submission to find candidate enzymes, we found 555 candidates out of 34,688 proteins, 82 of them with UniProt annotations for UDP-glycosyltransferase activity, five of which are uncharacterised proteins.

8.8.3 Single-step prediction of reductase for β-damascenone precursor production – UC 2

We applied the approach from the previous section for megastigma-6,7-dien-3,5,9-triol (β-Damascenone precursor) as well.

Figure 8.13 showcases the single step prediction of the production of megastigma-6,7-dien-3,5,9-triol. No single solution is the correct one, which would be the reduction of the ketone group of the grasshopper ketone. Solution 1 (the highest in score) and 5 provide what seems to be a multi-step solution, where the reduction of the methyl ketone takes place, and the formation of the allene group and an hydroxylation that seems to not happen according to literature. Solution 2 is the hydroxylation step and allene formation,

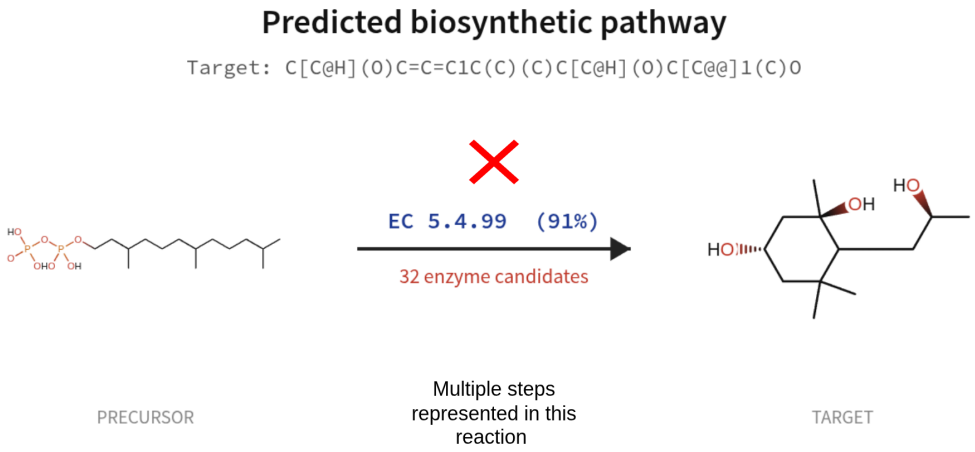


Figure 8.11: One-shot genome-guided retrobiosynthetic route for megastigma-6,7-dien-3,5,9-triol, obtained with Retroformer as *Reactor* and MEEA* as *Searcher* against the ITAG4.1 proteome. The route reaches the target in a single displayed step from a prenyl-diphosphate precursor, annotated as EC 5.4.99 (an isomerase, 91% confidence, 32 enzyme candidates). The step is incorrect: it compresses the full multi-step degradation into one transformation and assigns an isomerase class, whereas the elementary reaction is a carbonyl reduction (EC 1.1.1).

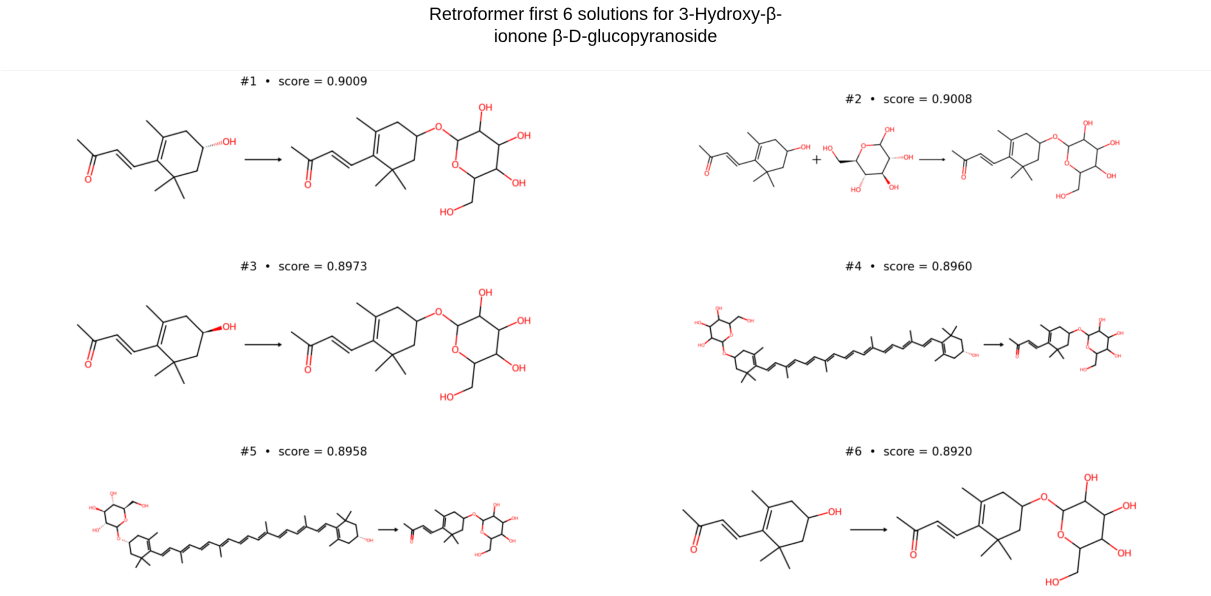


Figure 8.12: Retroformer first 6 solutions for 3-Hydroxy-β-ionone β-D-glucopyranoside.

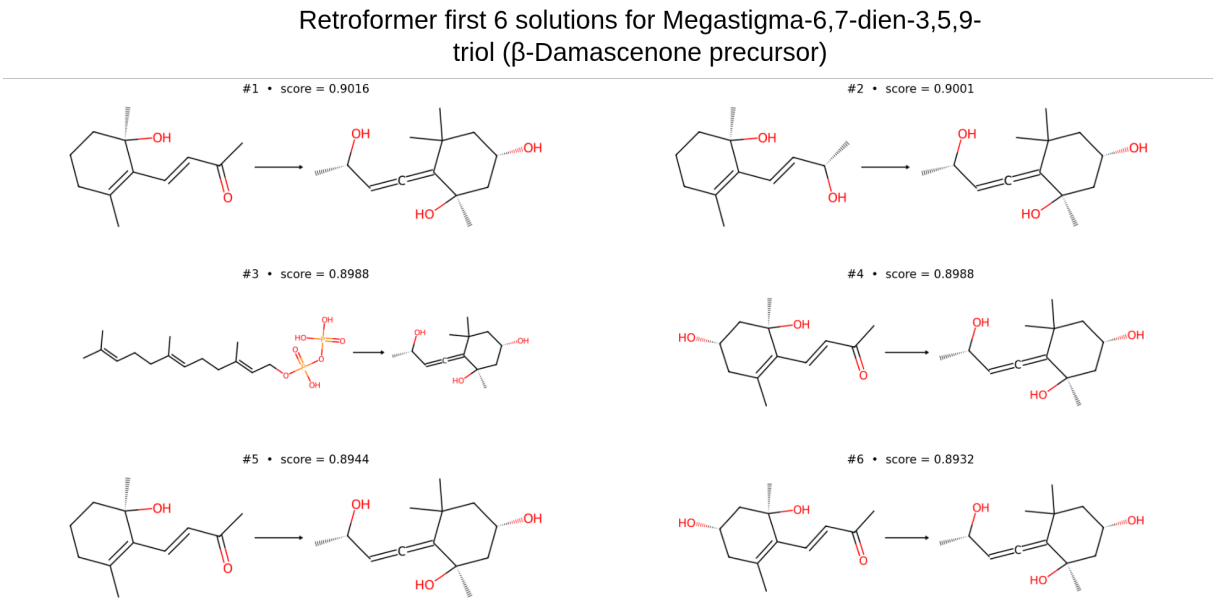


Figure 8.13: Retroformer first 6 solutions for Megastigma-6,7-dien-3,5,9-triol (β-Damascenone precursor).

solution 3 provides a multi-step approach directly to the (wrong) precursor FPP, because it is known that the carotenoid precursor is GGPP. Solutions 4 and 6 are almost correct, where there is a reduction of the methyl ketone; however it still considers the formation of the allene group, which is formed before in neoxanthin production.

We therefore evaluated the intended reaction directly: grasshopper ketone ⟶ megastigma-6,7-dien-3,5,9-triol. This transformation was not present among the returned Retroformer solutions, but it represents the expected ketone-reduction step. The reaction-level EC predictor still assigned the wrong class, predicting EC 5.4.99, an isomerase, instead of the expected ketone reductase class (EC 1.1.1). After submitting this corrected reaction to the enzyme-candidate search, ProteomeCompass returned 246 candidates from 34,688 tomato proteins. Of these, 53 had UniProt annotations consistent with oxidoreductase activity on CH-OH donors using NAD or NADP as acceptor, matching the definition of EC 1.1.1. Nineteen of the 53 annotated candidates were uncharacterised proteins.

8.9 Discussion and conclusions

The two tomato apocarotenoid targets analysed in this chapter were chosen because each couples a chemically well-defined endpoint with a biochemically unassigned step - the glycosyltransferase/β-glucosidase pair of the 3-hydroxy-β-ionone branch (UC 1) and the single reductase of the β-damascenone branch (UC 2) - and running both through DeePlants challenged its four inference methods: pathway enrichment, *de novo* retrobiosynthesis, single-step reaction prediction, and the enzyme candidate finder (summarised in Table 8.1). Of the four, pathway enrichment proved the most dependable: from the target structure alone it placed both compounds in the correct biosynthetic and chemical context, the apocarotenoid/

megastigmane glycoside logic for UC 1 and the neoxanthin-degradation/ABA branch for UC 2, and was correct in both runs. Its value is contextual rather than mechanistic, since it is an association over curated KEGG and PlantCyc entries and neither enumerates novel routes nor resolves elementary reactions or enzymes, so its role is to supply the orienting clues that make the more brittle downstream predictions interpretable.

Table 8.1: Comparison of the four DeePlants inference methods exercised in the two UCs, summarising the principal strengths, limitations, and observed behaviour.

Method	Strengths	Limitations	Use-case remarks
Pathway enrichment	• Operates from the target structure alone. • Places metabolites in biosynthetic and chemical (NP-Classifier) context. • Provides biological clues for downstream interpretation.	• Association driven; limited by database coverage. • Recovers known biosynthetic logic. • Does not generate novel routes or resolve enzyme-level detail.	• **UC1:** PWY-5397, apocarotenoid megastigmane glycoside. • **UC2:** PWY-695, ABA synthesis, non-glycoside. • Correct pathway association and NP-Classifier output in both cases.
***De novo* retrobiosynthesis (one-shot)**	• Generates ranked, multi-step routes from molecular structure alone. • Reaches the target molecule. • Couples chemical search to enzyme annotation. • Keeps *MEEA** and *Reactor* modules independently executable.	• Lumps multiple elementary reactions into single arrows. • Often violates mass balance. • Misassigns EC classes at individual steps. • Propagates errors into enzyme candidate finding.	• **UC1:** two-step route; first EC 4.2.3 is lumped, second EC 1.14.13 is incorrect. • **UC2:** single EC 5.4.99 isomerase step, although the elementary reaction is a carbonyl reduction.
Single-step reaction prediction	• Returns ranked candidate disconnections. • Can propose valid mass-balanced transformations. • Reaches chemically plausible precursor logic. • Supports manual inspection and solution selection.	• Correct solution is not always top-ranked or present. • Some top-ranked solutions break mass balance. • Reaction EC prediction can be unreliable.	• **UC1:** solution 2 is the correct UDP-glucosyltransferase reaction; reaction EC predicted as 4.1.2 at 21%. • **UC2:** correct reduction absent, but solutions 4 and 6 are near-correct and literature-consistent; reaction EC predicted as 5.4.99.

Method	Strengths	Limitations	Use-case remarks
Enzyme candidate finder (*Proteome Compass*)	• Returns focused, ranked proteome candidate shortlists. • Includes UniProt-level annotation. • Highlights uncharacterised proteins as testable hypotheses.	• Depends on agreement between enzyme-based and reaction-based EC predictions. • A wrong reaction EC collapses candidate matching. • Requires mass-balanced single-step reactions. • Training data are not automatically updated.	• **UC1:** 555 of 34,688 candidates after correcting reaction EC to 2.4.1; 82 UGT-annotated, including 5 uncharacterised. • **UC2:** 246 of 34,688 candidates after correcting reaction EC to 1.1.1; 53 oxidoreductase candidates, including 19 uncharacterised.

The enzyme candidate finder is the integrative endpoint of the pipeline, and its operation exposes the central vulnerability of the one-shot design. Candidates are identified by reconciling two independent EC predictions, the enzyme-based prediction obtained by screening the proteome with the *EnzymeAnnotator* and the reaction-based prediction obtained by classifying the proposed transformation with the *ReactionEC Annotator*, the *AnnotatorLinker* retains only enzymes whose class matches the reaction class. If the reaction-based EC prediction is wrong, the candidate finder is compromised from the start. The shortlist is then built around the wrong enzyme class, so even an accurate proteome-level screening cannot recover the correct candidates. This is worsened by where the reaction EC comes from: the *ReactionECAnnotator* was trained on ECREACT [280, 308], a dataset of single-step, mass-balanced transformations. When the reaction it is asked to classify is not mass-balanced, has a reactant missing, or lumps several elementary events into one arrow, it falls outside that training distribution and its prediction becomes unreliable. Since the reaction is produced upstream by the single-step *Reactor*, a poorly generated reaction corrupts the reaction EC, which in turn, corrupts the enzyme match: failure originates at the very first step of the one-shot pipeline and propagates, unrecovered, to the final enzyme shortlist. Both genome-guided one-shot runs displayed exactly this cascade - UC 1 (Figure 8.9) merging glycosylation and 3-hydroxylation into a mass-unbalanced arrow mislabelled EC 1.14.13 instead of a glycosyltransferase, and UC 2 (Figure 8.11) collapsing the entire degradation into one EC 5.4.99 isomerase step where the elementary chemistry is a carbonyl reduction (EC 1.1.1).

The same modules nonetheless recovered the biology once they were used individually and guided by the pathway-enrichment output. Pathway enrichment indicated which elementary reaction to isolate - the apocarotenoid-glycoside conjugation in UC 1 and the grasshopper-ketone reduction in UC 2. Running the single-step *Reactor* on that disconnection yielded, for UC 1, the correct mass-balanced glucosyltransferase reaction (solution 2, Figure 8.12); for UC 2 the correct reduction was absent from the returned set but was identified unambiguously from the near-correct solutions 4 and 6 read against the literature (Figure 8.13). With the reaction EC corrected by hand, the candidate finder returned focused, annotated shortlists: 555 of 34,688 proteins for UC 1 (82 UDP-glycosyltransferase-annotated, 5 uncharacterised) and 246 of 34,688 for UC 2 (53 oxidoreductase-annotated matching EC 1.1.1, 19 uncharacterised). Analysing the pipeline

step by step, following the clues from pathway enrichment, and running the predictions individually thus produced candidate-enzyme proposals for two reactions the tomato literature leaves unassigned - the UDP-glucosyltransferase (EC 2.4.1) conjugation of 3-hydroxy-β-ionone and the carbonyl reductase (EC 1.1.1) that reduces grasshopper ketone to megastigma-6,7-dien-3,5,9-triol, the immediate β-damascenone precursor. These uncharacterised candidates are directly testable hypotheses that, if confirmed, would close two flavour-relevant biosynthetic gaps and provide labelled positives validating the framework.

The clearest lesson concerns how DeePlants should be used: as a fully automated one-shot predictor it is brittle, because the quality of the single-step reaction sits at the head of a strict dependency chain and silently determines everything downstream; as an interpretable, modular, human-in-the-loop toolkit it is powerful, and the most direct route to greater autonomy - enforcing mass balance and atom mapping on *Reactor* outputs, decomposing lumped predictions into elementary steps before EC classification, and making the *ReactionECAnnotator* robust to multi-step reactions - follows directly from removing each failure mode of that same cascade.

9

Conclusion and future work

This thesis presents a computational framework designed to bridge research gaps in plant SM annotation, addressing both the scarcity of biosynthetic pathway data and the limitations of traditional sequence similarity-based approaches. A central contribution is the development of DeePlants, a framework that consolidates and integrates enzyme annotation and prediction, metabolite-pathway associations, and retrosynthetic tools into a unified and modular package. The work advances the understanding of computational representations for proteins and NPs, systematically evaluating their effectiveness for tasks such as EC number and enzyme-substrate specificity prediction, and biosynthetic pathway associations. This research provides a robust foundation for predicting enzyme functions and elucidating biosynthetic pathways, even in the absence of complete experimental evidence.

9.1 Main contributions

Chapter 3 presented DeepMol, an AutoML framework tailored for computational chemistry. Its open-source availability, comprehensive documentation, and competitive performance make it a pioneering tool in chemoinformatics, particularly for addressing challenges like feature engineering, model selection, and hyperparameter optimisation in QSAR/QSPR and NP studies.

Chapter 4 focused on evaluating state-of-the-art protein language models for predicting EC numbers, benchmarking their performance against the gold standard BLASTp. The study demonstrated that DL models using LLM embeddings outperform more traditional one-hot encoding approaches. While BLASTp exhibited a marginal advantage in overall performance, partly due to its role in annotating the UniProtKB test set, its performance declined significantly for sequences lacking homologues, where DL proved superior.

These findings brought up the complementary nature of BLASTp and DL models, particularly for challenging annotation tasks. For sequences with less than 25% identity to training data, DL models consistently outperformed BLASTp, and their combination further improved predictive performance across most identity intervals. This work also confirmed that DL models tend to be more precise, albeit with lower recall than BLASTp, highlighting the potential for hybrid approaches to enhance enzyme function annotation in plant SM research.

Chapter 5 follows by systematically benchmarking computational representations for NPs, identifying NPClassifierFP as the most effective choice for tasks such as biosynthetic similarity, metabolite-pathway prediction, and NP class classification. The study demonstrates that NPClassifierFP's count-based circular fingerprint matches or outperforms alternatives such as binary Morgan fingerprints, DL models' embeddings, and graph-based methods, excelling in both interpretability and predictive performance. Its ability to balance structural and biosynthetic information, evidenced by consistent results across diverse benchmarks positions it as the standard benchmark for NP machine learning. The findings also reveal that biosynthetic prediction often requires non-linear decision boundaries.

Chapter 6 provides an ML framework for predicting enzyme-substrate specificity, specifically designed to address the unique challenges of plant SM, where generalisability to novel substrates is paramount. By integrating fine-tuned protein language model embeddings, NPClassifierFP, and gradient boosting, the framework achieves superior performance across datasets, particularly for substrates with low similarity to training data. This capability is critical for elucidating biosynthetic pathways of plant specialised metabolites, where enzymes often act on structurally diverse, lineage-specific compounds that exhibit limited similarity to known substrates. The model outperforms similarity-based approaches and state-of-the-art models, while also providing a more scalable and practical solution for real-world applications.

Beyond predictive accuracy, the framework's efficiency and focus on high-quality negative pairs make it a powerful tool for attributing enzymes to specific reactions. Its ability to handle substrate diversity, without the computational overhead of methods like ProSmith, enables researchers to systematically link metabolites to enzymes, and, ultimately, to genes.

Chapter 7 extends the dataset of Eguchi and collaborators to include precursors of terpenoids, phenylpropanoids, glucosinolates, and benzoxazinoids, alongside alkaloids, enabling a more comprehensive analysis of plant specialised metabolite biosynthesis. By leveraging DeepMol's AutoML engine, the study identifies optimal combinations of ML models, feature extraction, and selection methods, demonstrating that Ridge classifiers outperform the state-of-the-art MGCNN across multiple datasets. The integration of features and models selected by DeepMol also surpasses the performance of NP-focused fingerprints, while aligning structural features with biosynthetic pathways, further validating the model's biological relevance. At scale, the framework successfully predicted precursors for over three-quarters of plant metabolites from LOTUS-DB.

Chapter 8 provides an in-depth overview of the main software tool developed in this thesis: DeePlants, a modular platform integrating enzyme annotation, pathway prediction, and retrosynthesis tools. DeePlants consolidates the computational methods developed throughout this work into a unified, extensible framework designed for plant SM research. By integrating enzyme annotation, metabolite-pathway associations, and retrosynthetic pathway elucidation, including GPU-accelerated inference and modular interfaces for retrobiosynthesis engines, DeePlants enables the systematic exploration of biosynthetic routes that are both chemically plausible and biologically feasible. Its architecture supports seamless interoperability between ML models, pathway prediction tools, and enzyme-substrate specificity analysis, providing a scalable solution for discovering and prioritising biosynthetic pathways in plants.

Beyond its architecture, the chapter exercised DeePlants on two tomato apocarotenoid targets - 3-hydroxy-β-ionone β-D-glucopyranoside and the β-damascenone precursor megastigma-6,7-dien-3,5,9-triol - each pairing a chemically defined endpoint with a biochemically unassigned step. These case studies exposed both the limits and the strengths of the integrated workflow: in fully automated one-shot mode the framework is constrained by the quality of the single-step reactions it generates, since an ill-formed or mass-unbalanced transformation propagates errors through reaction EC prediction to the enzyme candidate match, whereas the same components, used modularly under expert guidance and anchored by pathway-enrichment context, allowed us to propose candidate enzymes, including uncharacterised proteins, for two biosynthetic steps that the tomato literature leaves unassigned. DeePlants therefore delivers not only a reusable, extensible software framework but also concrete, experimentally testable hypotheses for plant SM, while pointing to clear avenues for refinement, such as enforcing mass balance on retrosynthetic outputs and improving robustness to multi-step reactions, that would bring its automated inference closer to the results currently achievable through guided, step-by-step analysis.

9.2 Publications

Chapter 3 was published in two peer-reviewed scientific journals:

- Correia, J., Capela, J., & Rocha, M. (2024). Deepmol: an automated machine and deep learning framework for computational chemistry. Journal of Cheminformatics, 16(1), 136.
- Capela, J., Correia, J., & Rocha, M. (2026). DeepMol: o Pacote de Quimioinformática Pioneiro em Portugal. Química, 50, 180

Chapter 4 was published in a peer-reviewed scientific journal: Capela, J., Zimmermann-Kogadeeva, M., Dijk, A. D. V., de Ridder, D., Dias, O., & Rocha, M. (2025). Comparative assessment of protein large language models for enzyme commission number prediction. BMC bioinformatics, 26(1), 68.

At the time of writing, Chapter 5 is being prepared for submission to a peer-reviewed scientific journal.

At the time of writing, Chapter 6 is being prepared for submission to a peer-reviewed scientific journal. However, preliminary results were presented at XV Symposium of Bioinformatics – JBI 2025. Reference: Capela, J., Dias, O., Rocha, M. Towards a Deep Learning Framework to Annotate Enzyme-Substrate Catalytic Interactions. XV Symposium of Bioinformatics – JBI 2025, 22-24.

Chapter 7 was published in two peer-reviewed scientific journals and presented in two conferences:

- Journals:
  - Capela, J., Cheixo, J., de Ridder, D., Rocha, M., & Dias, O. (2024, June). Automated machine learning to predict the precursors of plant specialised metabolites. In International Conference on Practical Applications of Computational Biology & Bioinformatics (pp. 153-162). Cham: Springer Nature Switzerland.

  - Capela, J., Cheixo, J., de Ridder, D., Dias, O., & Rocha, M. (2025). Predicting precursors of plant specialised metabolites using DeepMol automated machine learning. Journal of integrative bioinformatics, 22(2), 20240050.

- Conferences:

  - Capela, J., Cheixo, J., Aalt Jan van Dijk, de Ridder, D., Rocha, M., & Dias, O. (2023, September). Predicting the biosynthetic precursors of plant specialised metabolites using machine learning. In International Conference on Plant Systems Biology and Biotechnology
  - Capela, J., Cheixo, J., de Ridder, D., Rocha, M., & Dias, O. (2024, June). Automated machine learning to predict the precursors of plant specialised metabolites. In International Conference on Practical Applications of Computational Biology & Bioinformatics (PACBB)

Additionally, the author contributed to the following peer-reviewed scientific publications during the PhD. Related to modelling plant systems and biosynthesis of natural products:

- Capela, J., Lagoa, D., Rodrigues, R., Cunha, E., Cruz, F., Barbosa, A., ... & Dias, O. (2022). Merlin, an improved framework for the reconstruction of high-quality genome-scale metabolic models. Nucleic Acids Research, 50(11), 6052-6066.
- Cruz, F., Capela, J., Ferreira, E. C., Rocha, M., & Dias, O. (2024). BioISO: an objective-oriented application for assisting the curation of genome-scale metabolic models. IEEE/ACM transactions on computational biology and bioinformatics, 21(2), 215-226.
- Martins, L., Capela, J., Cunha, E., Sampaio, M., & Dias, O. (2025). diel_models: a python package for systematic integration of day–night cycles into plant genome-scale metabolic models. Bioinformatics Advances, 5(1), vbaf087.
- Zdouc, M. M., Blin, K., Louwen, N. L., Navarro, J., Loureiro, C., Bader, C. D., ... Capela, J. ... & Kang, S. (2025). MIBiG 4.0: advancing biosynthetic gene cluster curation through global collaboration. Nucleic acids research, 53(D1), D678-D690.

9.3 Perspectives for future work

Although the present work provides a meaningful contribution toward expanding the application of ML methodologies for uncovering plant SM, several promising avenues remain open for further development. These future directions span data curation, methodological improvements, systems-level biological integration, and pathway reconstruction reliability.

A primary direction for future work involves the continuous updating and expansion of the training datasets used throughout this framework. As databases such as LOTUS and COCONUT, plant metabolomics repositories, enzymatic annotations, and genomic resources continue to grow, incorporating newly available compounds, reactions, and protein sequences will be essential to maintain the predictive relevance

of the developed models. Periodic retraining with larger and more diverse datasets may improve generalisation, especially for underrepresented taxa and rare biosynthetic transformations.

At the methodological level, DeePlants can be extended. In particular, the searcher/planner components represent highly promising areas for improvement, since only MEEA* is implemented. The inclusion of alternative tree-search strategies, retrosynthesis-inspired planners, and biologically constrained route exploration algorithms may significantly enhance pathway discovery. Benchmarking these approaches against experimentally validated plant biosynthetic pathways will be critical to assess biological plausibility, computational efficiency, and reconstruction completeness.

Another important future direction goes in enriching pathway predictions with cellular and biochemical context. Biosynthetic pathways in plants are strongly shaped by compartmentalisation, with reactions often distributed across plastids, cytosol, vacuoles, endoplasmic reticulum, mitochondria, and peroxisomes. Therefore, integrating predictors of subcellular localisation for enzymes and reactions would provide a more realistic systems-level representation of metabolism. In parallel, adding thermodynamic feasibility analysis and enzyme kinetic constraints would enable the prioritisation of pathways that are not only structurally possible but also biologically executable under physiological conditions.

A further limitation that deserves attention concerns the use of sequence-to-sequence models for reaction prediction. While these models are powerful for learning reaction patterns, they do not inherently guarantee atom conservation or mass-balanced transformations. Future work should therefore focus on incorporating explicit chemical constraints into model architectures or post-processing validation steps to ensure stoichiometric consistency.

Beyond pathway reconstruction itself, an exciting long-term perspective is the integration of multi-omics evidence into the framework. Transcriptomics, proteomics, co-expression networks, phylogenomics, and metabolomics data could be leveraged to prioritise pathway candidates that are condition-specific, tissue-specific, or lineage-specific. Such integration would move the framework closer to a holistic platform for hypothesis generation in plant SM.

Ultimately, future developments should aim to transform the current methodology into a continuously learning, biologically grounded discovery engine capable of supporting the identification of novel enzymes, missing pathway steps, and previously uncharacterised metabolic routes in plants.

Bibliography

[1] N. Panchy, M. Lehti-Shiu, and S.-H. Shiu. “Evolution of Gene Duplication in Plants”. In: *Plant Physiology* 171 (4 2016-08), pp. 2294–2316 (cit. on pp. 1, 131).

[2] L. Chae et al. “Genomic Signatures of Specialized Metabolism in Plants”. In: *Science* 344 (6183 2014-05), pp. 510–513 (cit. on p. 1).

[3] B. M. Moore et al. “Robust predictions of specialized metabolism genes through machine learning”. In: *Proceedings of the National Academy of Sciences* 116 (6 2019-02), pp. 2344–2353 (cit. on pp. 1, 131).

[4] F. Zhou and E. Pichersky. “More is better: the diversity of terpene metabolism in plants”. In: *Current Opinion in Plant Biology* 55 (2020), pp. 1–10 (cit. on pp. 1, 131).

[5] D. J. Newman and G. M. Cragg. “Natural products as sources of new drugs from 1981 to 2014”. In: *Journal of natural products* 79.3 (2016), pp. 629–661 (cit. on p. 1).

[6] M. Erb and D. J. Kliebenstein. “Plant secondary metabolites as defenses, regulators, and primary metabolites: the blurred functional trichotomy”. In: *Plant physiology* 184.1 (2020), pp. 39–52 (cit. on p. 1).

[7] A.-M. Boudet. “Evolution and current status of research in phenolic compounds”. In: *Phytochemistry* 68.22-24 (2007), pp. 2722–2735 (cit. on p. 1).

[8] J.-K. Weng, R. N. Philippe, and J. P. Noel. “The rise of chemodiversity in plants”. In: *Science* 336.6089 (2012), pp. 1667–1670 (cit. on p. 1).

[9] M. E. Bolger, B. Arsova, and B. Usadel. “Plant genome and transcriptome annotations: from misconceptions to simple solutions”. In: *Briefings in Bioinformatics* (2017-01), bbw135 (cit. on pp. 1, 132).

[10] E. Pichersky and E. Lewinsohn. “Convergent Evolution in Plant Specialized Metabolism”. In: *Annual Review of Plant Biology* 62 (1 2011-06), pp. 549–566 (cit. on pp. 1, 132).

[11] A. Villegas-Morcillo et al. “Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function”. In: *Bioinformatics* 37 (2 2021-04), pp. 162–170 (cit. on p. 2).

[12] A. S. Schwartz et al. “Deep semantic protein representation for annotation, discovery, and engineering”. In: *BioRxiv* (2018), p. 365965 (cit. on pp. 2, 73, 88).

[13] J. Y. Ryu, H. U. Kim, and S. Y. Lee. “Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers”. In: *Proceedings of the National Academy of Sciences* 116.28 (2019-07), pp. 13996–14001 (cit. on pp. 2, 72, 73, 88).

[14] A. S. Rifaioglu et al. “MDeePred: novel multi-channel protein featurization for deep learning-based binding affinity prediction in drug discovery”. In: *Bioinformatics* 37.5 (2021-05), pp. 693–704 (cit. on p. 2).

[15] Y. Li et al. “DEEPre: sequence-based enzyme EC number prediction by deep learning”. In: *Bioinformatics* 34.5 (2018-03). Ed. by J. Hancock, pp. 760–769 (cit. on pp. 2, 72).

[16] A. A. Naqvi et al. “Advancements in docking and molecular dynamics simulations towards ligand-receptor interactions and structure-function relationships”. In: *Current topics in medicinal chemistry* 18.20 (2018), pp. 1755–1768 (cit. on pp. 2, 121).

[17] S. R. Zia, A. Coricello, and G. Bottegoni. “Increased throughput in methods for simulating protein ligand binding and unbinding”. In: *Current Opinion in Structural Biology* 87 (2024), p. 102871 (cit. on pp. 2, 121).

[18] D. Boldini et al. “Effectiveness of molecular fingerprints for exploring the chemical space of natural products”. In: *Journal of Cheminformatics* 16.1 (2024), p. 35 (cit. on pp. 2, 94, 112, 132, 140, 144).

[19] D. H. Wolpert and W. G. Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (2002), pp. 67–82 (cit. on p. 2).

[20] G. Turon et al. “First fully-automated AI/ML virtual screening cascade implemented at a drug discovery centre in Africa”. In: *Nat Commun* 14 (2023), p. 5736. doi: 10.1038/s41467-023-41512-2 (cit. on pp. 2, 54, 65).

[21] L. Mervin et al. “QSARtuna: An Automated QSAR Modeling Platform for Molecular Property Prediction in Drug Design”. In: *Journal of Chemical Information and Modeling* 64.14 (2024), pp. 5365–5374. issn: 1549-960X. doi: 10.1021/acs.jcim.4c00457 (cit. on pp. 2, 54, 65).

[22] K. Zhao and S. Y. Rhee. “Omics-guided metabolic pathway discovery in plants: Resources, approaches, and opportunities”. In: *Current opinion in plant biology* 67 (2022), p. 102222 (cit. on p. 3).

[23] F. M. Afendi et al. “KNApSAcK family databases: integrated metabolite–plant species databases for multifaceted plant research”. In: *Plant and Cell Physiology* 53.2 (2012), e1–e1 (cit. on p. 3).

[24] P. Schläpfer et al. “Genome-wide prediction of metabolic enzymes, pathways, and gene clusters in plants”. In: *Plant physiology* 173.4 (2017), pp. 2041–2059 (cit. on p. 3).

[25] T. Bepler and B. Berger. “Learning the protein language: Evolution, structure, and function”. In: *Cell Systems* 12.6 (2021-06), 654–669.e3 (cit. on p. 6).

[26] R. B. Leveson-Gower, C. Mayer, and G. Roelfes. “The importance of catalytic promiscuity for enzyme design and evolution”. In: *Nature Reviews Chemistry* 3.12 (2019-12), pp. 687–705 (cit. on pp. 6, 72, 120).

[27] M. H. Saier et al. “The Transporter Classification Database (TCDB): recent advances”. In: *Nucleic Acids Research* 44.D1 (2016-01), pp. D372–D379 (cit. on p. 6).

[28] S.-H. Kim, J. Turnbull, and S. Guimond. “Extracellular matrix and cell signalling: the dynamic cooperation of integrin, proteoglycan and growth factor receptor”. In: *Journal of Endocrinology* 209.2 (2011-05), pp. 139–151 (cit. on p. 6).

[29] A. A. Zamyatnin. “The features of an array of natural oligopeptides”. In: *Neurochemical Journal* 10.4 (2016-10), pp. 249–257 (cit. on p. 6).

[30] M. Blaxter. “Bioinformatics for Geneticists . Edited by M. R. Barnes and I. C. Grey. John Wiley and Sons, Chichester, UK. 2003. 350 pages. ISBN 0 470 84394 2. Price £45.00 (paperback). ISBN 0 470 84393 4. Price £90.00 (hardback).” In: *Genetical Research* 82.3 (2003-12), pp. 223–224 (cit. on p. 7).

[31] J. E. Donald, D. W. Kulp, and W. F. DeGrado. “Salt bridges: Geometrically specific, designable interactions”. In: *Proteins: Structure, Function, and Bioinformatics* 79.3 (2011-03), pp. 898–915 (cit. on p. 7).

[32] S. M. Marino and V. N. Gladyshev. “Cysteine Function Governs Its Conservation and Degeneration and Restricts Its Utilization on Protein Surfaces”. In: *Journal of Molecular Biology* 404.5 (2010-12), pp. 902–916 (cit. on p. 7).

[33] Y. Zhao and O. N. Jensen. “Modification-specific proteomics: Strategies for characterization of post-translational modifications using enrichment techniques”. In: *PROTEOMICS* 9.20 (2009-10), pp. 4632–4641 (cit. on p. 7).

[34] M. S. (S. Silberberg. *Principles of general chemistry*. 2013 (cit. on p. 7).

[35] J. Hastings et al. “Structure-based classification and ontology in chemistry”. In: *Journal of Cheminformatics* 4.1 (2012-12), p. 8 (cit. on pp. 7, 9).

[36] W. A. Warr. “Representation of chemical structures”. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 1.4 (2011-07), pp. 557–579 (cit. on pp. 9, 10).

[37] J. (Gasteiger and T. Engel. *Chemoinformatics: a textbook*. Wiley-VCH, 2003, p. 649 (cit. on p. 9).

[38] J. Polanski and J. Gasteiger. “Computer Representation of Chemical Compounds”. In: *Handbook of Computational Chemistry*. Cham: Springer International Publishing, 2017, pp. 1997–2039 (cit. on pp. 9, 57).

[39] N. M. O’Boyle. “Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI”. In: *Journal of Cheminformatics* 4.1 (2012-12), p. 22 (cit. on p. 9).

[40] D. Weininger. “SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules”. In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988-02), pp. 31–36 (cit. on p. 9).

[41] J. J. Vollmer. “Wiswesser line notation: an introduction”. In: *Journal of Chemical Education* 60.3 (1983-03), p. 192 (cit. on p. 9).

[42] S. Ash et al. “SYBYL Line Notation (SLN): A versatile language for chemical structure representation”. In: *Journal of Chemical Information and Computer Sciences* 37.1 (1997), pp. 71–79 (cit. on p. 9).

[43] H.-G. Rohbeck. “Representation of Structure Description Arranged Linearly”. In: *Software Development in Chemistry 5*. Springer Berlin Heidelberg, 1991, pp. 49–58 (cit. on p. 9).

[44] S. R. Heller et al. “InChI, the IUPAC International Chemical Identifier”. In: *Journal of Cheminformatics* 7.1 (2015-05) (cit. on pp. 9, 10).

[45] A. H. Lipkus et al. “Structural diversity of organic chemistry. A scaffold analysis of the CAS Registry”. In: *Journal of Organic Chemistry* 73.12 (2008-06), pp. 4443–4451 (cit. on p. 9).

[46] D. Probst and J. L. Reymond. “SmilesDrawer: Parsing and Drawing SMILES-Encoded Molecular Structures Using Client-Side JavaScript”. In: *Journal of Chemical Information and Modeling* 58.1 (2018-01), pp. 1–7 (cit. on pp. 9, 57).

[47] S. Heller et al. *InChI - The worldwide chemical structure identifier standard*. 2013-01 (cit. on p. 10).

[48] “catalyst”. In: *The IUPAC Compendium of Chemical Terminology*. Research Triangle Park, NC: International Union of Pure and Applied Chemistry (IUPAC), 2011-08 (cit. on p. 11).

[49] M. A. Keller, G. Piedrafita, and M. Ralser. “The widespread role of non-enzymatic reactions in cellular metabolism”. In: *Current Opinion in Biotechnology* 34 (2015-08), pp. 153–161 (cit. on pp. 11–14, 72, 120).

[50] R. Pascal, A. Pross, and J. D. Sutherland. “Towards an evolutionary theory of the origin of life based on kinetics and thermodynamics”. In: *Open Biology* 3.11 (2013-11), p. 130156 (cit. on p. 12).

[51] F. Tessier. “The Maillard reaction in the human body. The main discoveries and factors that affect glycation”. In: *Pathologie Biologie* 58.3 (2010-06), pp. 214–219 (cit. on p. 12).

[52] F. Wold. “In Vivo Chemical Modification of Proteins (Post-Translational Modification)”. In: *Annual Review of Biochemistry* 50.1 (1981-06), pp. 783–814 (cit. on p. 12).

[53] H. F. DeLuca. “Vitamin D: historical overview”. In: *Vitamins & Hormones* 100 (2016), pp. 1–20 (cit. on p. 12).

[54] N. Kochev, S. Avramova, and N. Jeliazkova. “Ambit-SMIRKS: a software module for reaction representation, reaction search and structure transformation”. In: *Journal of Cheminformatics* 10.1 (2018-12), pp. 1–29 (cit. on p. 14).

[55] T. Hartmann. “The lost origin of chemical ecology in the late 19th century”. In: *Proceedings of the National Academy of Sciences* 105.12 (2008-03), pp. 4541–4546 (cit. on pp. 15, 16).

[56] R. A. Hussein and A. A. El-Anssary. “Plants Secondary Metabolites: The Key Drivers of the Pharmacological Actions of Medicinal Plants”. In: *Herbal Medicine*. IntechOpen, 2019-01 (cit. on pp. 15, 16, 132).

[57] M. A. Alvarez. “Plant Secondary Metabolism Chapter 3”. In: *Plant Biotechnology for Health*. Springer International Publishing, 2014, pp. 15–31 (cit. on pp. 15–17).

[58] M. Erb and D. J. Kliebenstein. “Plant Secondary Metabolites as Defenses, Regulators, and Primary Metabolites: The Blurred Functional Trichotomy”. In: *Plant Physiology* 184.1 (2020-09), pp. 39–52 (cit. on pp. 15, 16).

[59] Q. Guo, I. T. Major, and G. A. Howe. “Resolution of growth–defense conflict: mechanistic insights from jasmonate signaling”. In: *Current Opinion in Plant Biology* 44 (2018-08), pp. 72–81 (cit. on p. 15).

[60] N. K. Clay et al. “Glucosinolate Metabolites Required for an Arabidopsis Innate Immune Response”. In: *Science* 323.5910 (2009-01), pp. 95–101 (cit. on p. 15).

[61] B. Li et al. “Convergent evolution of a metabolic switch between aphid and caterpillar resistance in cereals”. In: *Science Advances* 4.12 (2018-12) (cit. on p. 15).

[62] E. Soubeyrand et al. “The Peroxidative Cleavage of Kaempferol Contributes to the Biosynthesis of the Benzenoid Moiety of Ubiquinone in Plants”. In: *The Plant Cell* 30.12 (2018-12), pp. 2910–2921 (cit. on p. 15).

[63] R. Jenrich et al. “Evolution of heteromeric nitrilase complexes in Poaceae with new functions in nitrile metabolism”. In: *Proceedings of the National Academy of Sciences* 104.47 (2007-11), pp. 18848–18853 (cit. on p. 15).

[64] D. F. Rhoades. “Integrated antiherbivore, antidesiccant and ultraviolet screening properties of creosotebush resin”. In: *Biochemical Systematics and Ecology* 5.4 (1977-01), pp. 281–290 (cit. on p. 16).

[65] T. Vogt. “Phenylpropanoid Biosynthesis”. In: *Molecular Plant* 3.1 (2010-01), pp. 2–20 (cit. on pp. 16, 17, 132, 141).

[66] J. Dai and R. J. Mumper. “Plant Phenolics: Extraction, Analysis and Their Antioxidant and Anticancer Properties”. In: *Molecules* 15.10 (2010-10), pp. 7313–7352 (cit. on p. 16).

[67] S. Kumar et al. “Plant Secondary Metabolites: Their Food and Therapeutic Importance”. In: *Plant Secondary Metabolites*. Singapore: Springer Nature Singapore, 2022, pp. 371–413 (cit. on p. 17).

[68] M. L. Falcone Ferreyra, S. P. Rius, and P. Casati. “Flavonoids: biosynthesis, biological functions, and biotechnological applications”. In: *Frontiers in Plant Science* 3 (2012) (cit. on pp. 17–19).

[69] F. Bourgaud et al. “Biosynthesis of coumarins in plants: a major pathway still to be unravelled for cytochrome P450 enzymes”. In: *Phytochemistry Reviews* 5.2-3 (2006-11), pp. 293–308 (cit. on p. 18).

[70] N. E. Nagy et al. “Patterns and roles of lignan and terpenoid accumulation in the reaction zone compartmentalizing pathogen-infected heartwood of Norway spruce”. In: *Planta* 255.3 (2022-03), p. 63 (cit. on p. 18).

[71] A. Valletta, L. M. Iozia, and F. Leonelli. “Impact of Environmental Factors on Stilbene Biosynthesis”. In: *Plants* 10.1 (2021-01), p. 90 (cit. on pp. 18, 19).

[72] R. V. Barbehenn and C. Peter Constabel. “Tannins in plant–herbivore interactions”. In: *Phytochemistry* 72.13 (2011-09), pp. 1551–1565 (cit. on p. 18).

[73] W. Hu et al. “Biosynthesis of Phenolic Compounds and Antioxidant Activity in Fresh-Cut Fruits and Vegetables”. In: *Frontiers in Microbiology* 13 (2022-05) (cit. on pp. 18, 19).

[74] L. D. Antika et al. “Scopoletin: a review of its source, biosynthesis, methods of extraction, and pharmacological activities”. In: *Zeitschrift für Naturforschung C* 77.7-8 (2022-07), pp. 303–316 (cit. on pp. 18, 19).

[75] K. Robe et al. “The Coumarins: Secondary Metabolites Playing a Primary Role in Plant Nutrition and Health”. In: *Trends in Plant Science* 26.3 (2021-03), pp. 248–259 (cit. on p. 18).

[76] N. G. Lewis and L. B. Davin. “Lignans: biosynthesis and function”. In: *Comprehensive natural products chemistry* 1 (1999), pp. 639–712 (cit. on pp. 18, 19).

[77] W. Dong and Y. Song. “The Significance of Flavonoids in the Process of Biological Nitrogen Fixation”. In: *International Journal of Molecular Sciences* 21.16 (2020-08), p. 5926 (cit. on pp. 18, 19).

[78] A. Mouradov and G. Spangenberg. “Flavonoids: a metabolic network mediating plants adaptation to their real estate”. In: *Frontiers in Plant Science* 5 (2014-11) (cit. on p. 18).

[79] D. Tholl. “Biosynthesis and biological functions of terpenoids in plants”. In: *Biotechnology of isoprenoids* (2015), pp. 63–106 (cit. on pp. 18, 20, 24, 132).

[80] A. Champagne and M. Boutry. “Proteomics of terpenoid biosynthesis and secretion in trichomes of higher plant species”. In: *Biochimica et Biophysica Acta (BBA) - Proteins and Proteomics* 1864.8 (2016-08), pp. 1039–1049 (cit. on pp. 18, 20, 24).

[81] J. Bohlmann, G. Meyer-Gauen, and R. Croteau. “Plant terpenoid synthases: Molecular biology and phylogenetic analysis”. In: *Proceedings of the National Academy of Sciences* 95.8 (1998-04), pp. 4126–4133 (cit. on p. 18).

[82] M. E. Bergman, B. Davis, and M. A. Phillips. “Medically Useful Plant Terpenoids: Biosynthesis, Occurrence, and Mechanism of Action”. In: *Molecules* 24.21 (2019-11), p. 3961 (cit. on pp. 18, 20, 24).

[83] A. M. Boutanaev et al. “Investigation of terpene diversification across multiple sequenced plant genomes”. In: *Proceedings of the National Academy of Sciences* 112.1 (2015-01) (cit. on p. 18).

[84] M. Wink. “Modes of Action of Herbal Medicines and Plant Secondary Metabolites”. In: *Medicines* 2.3 (2015-09), pp. 251–286 (cit. on p. 18).

[85] K.-H. Kubeczka. “History and sources of essential oil research”. In: *Handbook of essential oils*. CRC Press, 2020, pp. 3–39 (cit. on p. 20).

[86] L. K. Henry et al. “Contribution of isopentenyl phosphate to plant terpenoid metabolism”. In: *Nature plants* 4.9 (2018), pp. 721–729 (cit. on pp. 20, 24).

[87] B. R. Lichman. “The scaffold-forming steps of plant alkaloid biosynthesis”. In: *Natural Product Reports* 38.1 (2021), pp. 103–129 (cit. on pp. 20–23, 25, 26).

[88] W. Kukula-Koch and J. Widelski. “Alkaloids”. In: *Pharmacognosy*. Elsevier, 2017, pp. 163–198 (cit. on p. 20).

[89] C. Hanfrey et al. “Arabidopsis polyamine biosynthesis: absence of ornithine decarboxylase and the mechanism of arginine decarboxylase activity”. In: *The Plant Journal* 27.6 (2001-09), pp. 551–560 (cit. on p. 21).

[90] T. Zhao et al. “Metabolic Characterization of Hyoscyamus niger Ornithine Decarboxylase”. In: *Frontiers in Plant Science* 10 (2019-02) (cit. on p. 21).

[91] J.-P. Huang et al. “Tropane alkaloids biosynthesis involves an unusual type III polyketide synthase and non-enzymatic condensation”. In: *Nature Communications* 10.1 (2019-12), p. 4036 (cit. on p. 22).

[92] S. Bunsupa et al. “Lysine Decarboxylase Catalyzes the First Step of Quinolizidine Alkaloid Biosynthesis and Coevolved with Alkaloid Production in Leguminosae”. In: *The Plant Cell* 24.3 (2012-03), pp. 1202–1216 (cit. on p. 22).

[93] W. Schliemann, N. Kobayashi, and D. Strack. “The Decisive Step in Betaxanthin Biosynthesis Is a Spontaneous Reaction1”. In: *Plant Physiology* 119.4 (1999-04), pp. 1217–1232 (cit. on p. 22).

[94] T. M. Mitchell and T. M. Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997 (cit. on p. 27).

[95] T. Jo. Springer International Publishing, 2021, pp. 3–22 (cit. on p. 27).

[96] A. C. Müller and S. Guido. *Introduction to machine learning with Python: a guide for data scientists*. ” O’Reilly Media, Inc.”, 2016 (cit. on p. 27).

[97] M. Z. Naser and A. H. Alavi. “Error Metrics and Performance Fitness Indicators for Artificial Intelligence and Machine Learning in Engineering and Sciences”. In: *Architecture, Structures and Construction* (2021-11) (cit. on pp. 28–30).

[98] H. Huang and H. V. Burton. “Classification of in-plane failure modes for reinforced concrete frames with infills using machine learning”. In: *Journal of Building Engineering* 25 (2019-09), p. 100767 (cit. on p. 28).

[99] R. Caruana and A. Niculescu-Mizil. “Data mining in metric space”. In: ACM Press, 2004, p. 69 (cit. on p. 28).

[100] J. Davis and M. Goadrich. “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd international conference on Machine learning - ICML ’06*. New York, New York, USA: ACM Press, 2006, pp. 233–240. isbn: 1595933832 (cit. on p. 29).

[101] X. Su, X. Yan, and C.-L. Tsai. “Linear regression”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4 (3 2012-05), pp. 275–294 (cit. on p. 29).

[102] X. Zou et al. “Logistic Regression Model Optimization and Case Analysis”. In: IEEE, 2019-10, pp. 135–139. isbn: 978-1-7281-3299-0 (cit. on p. 31).

[103] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. ” O’Reilly Media, Inc.”, 2022 (cit. on pp. 31–33, 36–39).

[104] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015-05), pp. 436–444 (cit. on p. 32).

[105] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (6088 1986-10), pp. 533–536 (cit. on p. 33).

[106] F. Masi et al. “Thermodynamics-based Artificial Neural Networks for constitutive modeling”. In: *Journal of the Mechanics and Physics of Solids* 147 (2021-02), p. 104277 (cit. on p. 34).

[107] F. Chollet. *Deep learning with Python*. Simon and Schuster, 2021 (cit. on pp. 34, 38).

[108] S. Ruder. “An overview of gradient descent optimization algorithms”. In: (2016-09). arXiv: 1609.04747 (cit. on pp. 34–36).

[109] N. Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural Networks* 12.1 (1999-01), pp. 145–151 (cit. on p. 35).

[110] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011) (cit. on p. 36).

[111] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 36).

[112] S. Geman, E. Bienenstock, and R. Doursat. “Neural networks and the bias/variance dilemma”. In: *Neural computation* 4.1 (1992), pp. 1–58 (cit. on p. 37).

[113] B. Neal et al. “A Modern Take on the Bias-Variance Tradeoff in Neural Networks”. In: (2018-10). arXiv: 1810.08591 (cit. on p. 37).

[114] Y. Dar, V. Muthukumar, and R. G. Baraniuk. “A Farewell to the Bias-Variance Tradeoff? An Overview of the Theory of Overparameterized Machine Learning”. In: (2021-09). arXiv: 2109.02355 (cit. on p. 37).

[115] N. Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (2014-01), pp. 1929–1958 (cit. on p. 38).

[116] S. Ioffe and C. Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015 (cit. on p. 38).

[117] K. O’Shea and R. Nash. “An Introduction to Convolutional Neural Networks”. In: (2015-11) (cit. on p. 39).

[118] Y. Xu et al. “Deep Dive into Machine Learning Models for Protein Engineering”. In: *Journal of Chemical Information and Modeling* 60 (6 2020-06), pp. 2773–2790 (cit. on p. 39).

[119] M. Hirohara et al. “Convolutional neural network based on SMILES representation of compounds for detecting chemical motif”. In: *BMC Bioinformatics* 19 (S19 2018-12), p. 526 (cit. on p. 39).

[120] N. Q. K. Le and V.-N. Nguyen. “SNARE-CNN: a 2D convolutional neural network architecture to identify SNARE proteins from high-throughput sequencing data”. In: *PeerJ Computer Science* 5 (2019-02), e177 (cit. on p. 39).

[121] W. Torng and R. B. Altman. “3D deep convolutional neural networks for amino acid environment similarity analysis”. In: *BMC Bioinformatics* 18 (1 2017-12), p. 302 (cit. on p. 39).

[122] D. Kuzminykh et al. “3D Molecular Representations Based on the Wave Transform for Convolutional Neural Networks”. In: *Molecular Pharmaceutics* 15 (10 2018-10), pp. 4378–4385 (cit. on p. 39).

[123] R. M. Schmidt. “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview”. In: (2019-11) (cit. on p. 40).

[124] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on p. 40).

[125] Z. Niu, G. Zhong, and H. Yu. “A review on the attention mechanism of deep learning”. In: *Neurocomputing* 452 (2021), pp. 48–62 (cit. on p. 40).

[126] F. Cui, Z. Zhang, and Q. Zou. “Sequence representation approaches for sequence-based protein prediction tasks that use deep learning”. In: *Briefings in Functional Genomics* 20 (1 2021-03), pp. 61–73 (cit. on pp. 40, 41).

[127] A. Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 40, 41).

[128] F. Scarselli et al. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80 (cit. on p. 42).

[129] T. Thanapalasingam et al. “Relational graph convolutional networks: a closer look”. In: *PeerJ Computer Science* 8 (2022), e1073 (cit. on p. 43).

[130] A. Küken and Z. Nikoloski. “Computational Approaches to Design and Test Plant Synthetic Metabolic Pathways”. In: *Plant Physiology* 179 (3 2019-03), pp. 894–906 (cit. on p. 43).

[131] C. Yan et al. “Retroxpert: Decompose retrosynthesis prediction like a chemist”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11248–11258 (cit. on pp. 44, 146).

[132] R. Bellman. “Dynamic programming”. In: *science* 153.3731 (1966), pp. 34–37 (cit. on p. 45).

[133] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998 (cit. on pp. 45, 47).

[134] Y. Matsumura, K. Tabata, and T. Komatsuzaki. “Comparative Analysis of Reinforcement Learning Algorithms for Finding Reaction Pathways: Insights from a Large Benchmark Data Set”. In: *Journal of Chemical Theory and Computation* 21.7 (2025), pp. 3523–3535 (cit. on p. 46).

[135] P. Schwaller et al. “Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy”. In: *Chemical Science* 11 (12 2020), pp. 3316–3325 (cit. on pp. 46, 51).

[136] I. V. Tetko et al. “State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis”. In: *Nature Communications* 11 (1 2020-11), p. 5575 (cit. on pp. 46, 51).

[137] S. Zheng et al. “Predicting Retrosynthetic Reaction using Self-Corrected Transformer Neural Networks”. In: (2019-07) (cit. on pp. 46, 51).

[138] P. E. Hart, N. J. Nilsson, and B. Raphael. “A formal basis for the heuristic determination of minimum cost paths”. In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 46).

[139] C. B. Browne et al. “A survey of monte carlo tree search methods”. In: *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012), pp. 1–43 (cit. on pp. 46, 47).

[140] L. Kocsis and C. Szepesvári. “Bandit based monte-carlo planning”. In: *European conference on machine learning*. Springer. 2006, pp. 282–293 (cit. on p. 47).

[141] G. D. Moghe and L. H. Kruse. “The study of plant specialized metabolism”. In: *American Journal of Botany* 105.6 (2018), pp. 959–962 (cit. on p. 47).

[142] W. Kreis and J. Munkert. “Exploiting enzyme promiscuity to shape plant specialized metabolism”. In: *Journal of experimental botany* 70.5 (2019), pp. 1435–1445 (cit. on p. 47).

[143] P. Auer, N. Cesa-Bianchi, and P. Fischer. “Finite-time analysis of the multiarmed bandit problem”. In: *Machine learning* 47.2 (2002), pp. 235–256 (cit. on p. 47).

[144] C. D. Rosin. “Multi-armed bandits with episode context”. In: *Annals of Mathematics and Artificial Intelligence* 61.3 (2011), pp. 203–230 (cit. on p. 47).

[145] B. Chen et al. “Retro*: learning retrosynthetic planning with neural guided A* search”. In: *International conference on machine learning*. PMLR. 2020, pp. 1608–1616 (cit. on p. 48).

[146] M. H. Segler, M. Preuss, and M. P. Waller. “Planning chemical syntheses with deep neural networks and symbolic AI”. In: *Nature* 555.7698 (2018), pp. 604–610 (cit. on p. 48).

[147] Z. Ni et al. “Curating a comprehensive set of enzymatic reaction rules for efficient novel biosynthetic pathway design”. In: *Metabolic Engineering* 65 (2021-05), pp. 79–87 (cit. on pp. 49, 50).

[148] Y. Moriya et al. “PathPred: an enzyme-catalyzed metabolic pathway prediction server”. In: *Nucleic Acids Research* 38 (Web Server 2010-07), W138–W143 (cit. on pp. 49, 50).

[149] B. Delépine et al. “RetroPath2.0: A retrosynthesis workflow for metabolic engineers”. In: *Metabolic Engineering* 45 (2018-01), pp. 158–170 (cit. on p. 50).

[150] S. Ding et al. “novoPathFinder: a webserver of designing novel-pathway with integrating GEM-model”. In: *Nucleic Acids Research* 48 (W1 2020-07), W477–W487 (cit. on p. 50).

[151] X. Wang et al. “RetroPrime: A Diverse, plausible and Transformer-based method for Single-Step retrosynthesis predictions”. In: *Chemical Engineering Journal* 420 (2021-09), p. 129845 (cit. on p. 51).

[152] P. Schwaller et al. “Molecular Transformer: A Model for Uncertainty-Calibrated Chemical Reaction Prediction”. In: *ACS Central Science* 5 (9 2019-09), pp. 1572–1583 (cit. on p. 51).

[153] K. Lin et al. “Automatic Retrosynthetic Pathway Planning Using Template-free Models”. In: (2019-05) (cit. on p. 51).

[154] M. H. S. Segler, M. Preuss, and M. P. Waller. “Planning chemical syntheses with deep neural networks and symbolic AI”. In: *Nature* 555 (7698 2018-03), pp. 604–610 (cit. on p. 51).

[155] C. W. Coley et al. “SCScore: Synthetic Complexity Learned from a Reaction Corpus”. In: *Journal of Chemical Information and Modeling* 58 (2 2018-02), pp. 252–261 (cit. on p. 51).

[156] S. Zheng et al. “Deep learning driven biosynthetic pathways navigation for natural products with BioNavi-NP”. In: *Nature Communications* 13.1 (2022), p. 3342 (cit. on p. 52).

[157] T. Kim et al. “READRetro: natural product biosynthesis predicting with retrieval-augmented dual-view retrosynthesis”. In: *New Phytologist* 243.6 (2024), pp. 2512–2527 (cit. on p. 52).

[158] D. Zhao, S. Tu, and L. Xu. “Efficient retrosynthetic planning with MCTS exploration enhanced A* search”. In: *Communications Chemistry* 7.1 (2024), p. 52 (cit. on pp. 52, 147).

[159] G. Hessler and K.-H. Baringhaus. “Artificial Intelligence in Drug Design”. In: *Molecules* 23.10 (2018-10), p. 2520. doi: 10.3390/molecules23102520 (cit. on p. 53).

[160] J. Shen and C. A. Nicolaou. “Molecular property prediction: recent trends in the era of artificial intelligence”. In: *Drug Discovery Today: Technologies* 32-33 (2019-12), pp. 29–36. doi: 10.1016/j.ddtec.2020.05.001 (cit. on p. 53).

[161] J. Gasteiger. “Chemistry in Times of Artificial Intelligence”. In: *ChemPhysChem* 21.20 (2020-09), pp. 2233–2242. doi: 10.1002/cphc.202000518 (cit. on p. 53).

[162] W. P. Walters and R. Barzilay. “Applications of Deep Learning in Molecule Generation and Molecular Property Prediction”. In: *Accounts of Chemical Research* 54.2 (2020-12), pp. 263–270. doi: 10.1021/acs.accounts.0c00699 (cit. on p. 53).

[163] G. Montavon et al. “Machine learning of molecular electronic properties in chemical compound space”. In: *New Journal of Physics* 15 (9 2013-09), p. 095003. issn: 1367-2630. doi: 10.1088/1367-2630/15/9/095003 (cit. on p. 53).

[164] A. Tkatchenko. “Machine learning for chemical discovery”. In: *Nature Communications* 11 (1 2020-08), p. 4125. issn: 2041-1723. doi: 10.1038/s41467-020-17844-8 (cit. on p. 54).

[165] M. Karelson, V. S. Lobanov, and A. R. Katritzky. “Quantum-Chemical Descriptors in QSAR/QSPR Studies”. In: *Chemical Reviews* 96 (3 1996-01), pp. 1027–1044. issn: 0009-2665. doi: 10.1021/cr950202r (cit. on p. 54).

[166] W. M. Berhanu et al. “Quantitative Structure-Activity/Property Relationships: The Ubiquitous Links between Cause and Effect”. In: *ChemPlusChem* 77 (7 2012-07), pp. 507–517. issn: 21926506. doi: 10.1002/cplu.201200038 (cit. on p. 54).

[167] P. C. S. Costa et al. “Chemical Graph Theory for Property Modeling in QSAR and QSPR—Charming QSAR and QSPR”. In: *Mathematics* 9 (1 2020-12), p. 60. issn: 2227-7390. doi: 10.3390/math9010060 (cit. on p. 54).

[168] I. L. Guidotti et al. “Bambu and its applications in the discovery of active molecules against melanoma”. In: *Journal of Molecular Graphics and Modelling* 124 (2023-11), p. 108564. issn: 10933263. doi: 10.1016/j.jmgm.2023.108564 (cit. on pp. 54, 65).

[169] G. Sliwoski et al. “Computational Methods in Drug Discovery”. In: *Pharmacological Reviews* 66.1 (2014). Ed. by E. L. Barker, pp. 334–395. issn: 0031-6997. doi: 10.1124/pr.112.007336 (cit. on p. 54).

[170] Z. Wu et al. “Do we need different machine learning algorithms for QSAR modeling? A comprehensive assessment of 16 machine learning algorithms on 14 QSAR data sets”. In: *Briefings in Bioinformatics* 22 (4 2021-07). issn: 1467-5463. doi: 10.1093/bib/bbaa321 (cit. on p. 54).

[171] Y. Xu. “Deep Neural Networks for QSAR”. In: 2022, pp. 233–260. doi: 10.1007/978-1-0716-1787-8_10 (cit. on p. 54).

[172] Z. Li et al. “Deep learning methods for molecular representation and property prediction”. In: *Drug Discovery Today* 27.12 (2022-12), p. 103373. doi: 10.1016/j.drudis.2022.103373 (cit. on p. 54).

[173] Á. Orosz, K. Héberger, and A. Rácz. “Comparison of Descriptor- and Fingerprint Sets in Machine Learning Models for ADME-Tox Targets”. In: *Frontiers in Chemistry* 10 (2022-06). doi: 10.3389/fchem.2022.852893 (cit. on p. 54).

[174] T. Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019 (cit. on pp. 54, 55, 60).

[175] *RDKit: Open-source cheminformatics*. http://www.rdkit.org (cit. on p. 54).

[176] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 54).

[177] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. url: https://www.tensorflow.org/ (cit. on p. 54).

[178] B. Ramsundar et al. *Deep Learning for the Life Sciences*. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837. O’Reilly Media, 2019 (cit. on p. 55).

[179] J. Gasteiger et al. “Chemical Information in 3D Space”. In: *Journal of Chemical Information and Computer Sciences* 36.5 (1996), pp. 1030–1037. doi: 10.1021/ci960343+ (cit. on p. 57).

[180] D. S. Wigh, J. M. Goodman, and A. A. Lapkin. “A review of molecular representation in the age of machine learning”. In: *WIREs Computational Molecular Science* 12 (5 2022-09). issn: 1759-0876. doi: 10.1002/wcms.1603 (cit. on p. 57).

[181] K. Mansouri et al. “An automated curation procedure for addressing chemical errors and inconsistencies in public datasets used in QSAR modelling”. In: *SAR and QSAR in Environmental Research* 27 (11 2016-11), pp. 911–937. issn: 1062-936X. doi: 10.1080/1062936X.2016.1253611 (cit. on p. 57).

[182] D. Fourches, E. Muratov, and A. Tropsha. “Trust, But Verify: On the Importance of Chemical Structure Curation in Cheminformatics and QSAR Modeling Research”. In: *Journal of Chemical Information and Modeling* 50.7 (2010-06), pp. 1189–1204. doi: 10.1021/ci100176x (cit. on p. 57).

[183] A. P. Bento et al. “An open source chemical structure curation pipeline using RDKit”. In: *Journal of Cheminformatics* 12.1 (2020-09). doi: 10.1186/s13321-020-00456-1 (cit. on p. 57).

[184] V. D. Hähnke, S. Kim, and E. E. Bolton. “PubChem chemical structure standardization”. In: *Journal of Cheminformatics* 10.1 (2018-08). doi: 10.1186/s13321-018-0293-8 (cit. on p. 57).

[185] K. Karapetyan et al. “The Chemical Validation and Standardization Platform (CVSP): large-scale automated validation of chemical structure datasets”. In: *Journal of Cheminformatics* 7.1 (2015-06). doi: 10.1186/s13321-015-0072-8 (cit. on p. 57).

[186] V. Consonni and R. Todeschini. *Molecular descriptors for chemoinformatics*. John Wiley & Sons, 2009 (cit. on p. 58).

[187] R. Todeschini and V. Consonni. “Molecular descriptors”. In: *Recent Advances in QSAR Studies* (2010), pp. 29–102 (cit. on p. 58).

[188] H. Moriwaki et al. “Mordred: a molecular descriptor calculator”. In: *Journal of Cheminformatics* 10 (1 2018-12), p. 4. issn: 1758-2946. doi: 10.1186/s13321-018-0258-y (cit. on p. 58).

[189] X. Li and D. Fourches. “SMILES Pair Encoding: A Data-Driven Substructure Tokenization Algorithm for Deep Learning”. In: *Journal of Chemical Information and Modeling* 61 (4 2021-04), pp. 1560–1569. issn: 1549-9596. doi: 10.1021/acs.jcim.0c01127 (cit. on p. 58).

[190] S. Kearnes et al. “Molecular graph convolutions: moving beyond fingerprints”. In: *Journal of Computer-Aided Molecular Design* 30 (8 2016-08), pp. 595–608. issn: 0920-654X. doi: 10.1007/s10822-016-9938-8 (cit. on p. 58).

[191] D. K. Duvenaud et al. “Convolutional networks on graphs for learning molecular fingerprints”. In: *Advances in neural information processing systems* 28 (2015) (cit. on p. 58).

[192] N. De Cao and T. Kipf. “MolGAN: An implicit generative model for small molecular graphs”. In: *arXiv preprint arXiv:1805.11973* (2018) (cit. on p. 58).

[193] G. Montavon et al. “Learning invariant representations of molecules for atomization energy prediction”. In: *Advances in neural information processing systems* 25 (2012) (cit. on p. 58).

[194] M. B. Kursa and W. R. Rudnicki. “Feature Selection with the Boruta Package”. In: *Journal of Statistical Software* 36.11 (2010). doi: 10.18637/jss.v036.i11 (cit. on p. 59).

[195] D. Baptista et al. “Evaluating molecular representations in machine learning models for drug response prediction and interpretability”. In: *Journal of Integrative Bioinformatics* 19 (3 2022-09) (cit. on p. 59).

[196] J. Capela et al. “merlin, an improved framework for the reconstruction of high-quality genome-scale metabolic models”. In: *Nucleic Acids Research* 50.11 (2022-06), pp. 6052–6066 (cit. on pp. 59, 72).

[197] D. Butina. “Unsupervised Data Base Clustering Based on Daylight’s Fingerprint and Tanimoto Similarity: A Fast and Automated Way To Cluster Small and Large Data Sets”. In: *Journal of Chemical Information and Computer Sciences* 39.4 (1999), pp. 747–750. doi: 10.1021/ci9803381 (cit. on p. 62).

[198] C. Xu et al. “In silico prediction of chemical Ames mutagenicity”. In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2840–2847 (cit. on p. 63).

[199] J. M. Johnson and T. M. Khoshgoftaar. “Survey on deep learning with class imbalance”. In: *Journal of Big Data* 6.1 (2019-03). doi: 10.1186/s40537-019-0192-5 (cit. on p. 63).

[200] S. Korkmaz. “Deep Learning-Based Imbalanced Data Classification for Drug Discovery”. In: *Journal of Chemical Information and Modeling* 60.9 (2020-06), pp. 4180–4190. doi: 10.1021/acs.jcim.9b01162 (cit. on pp. 63, 64).

[201] G. Lemaître, F. Nogueira, and C. K. Aridas. “Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning”. In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. url: http://jmlr.org/papers/v18/16-365.html (cit. on p. 64).

[202] N. V. Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique”. In: *Journal of Artificial Intelligence Research* 16 (2002-06), pp. 321–357. doi: 10.1613/jair.953 (cit. on p. 64).

[203] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard. “Balancing Training Data for Automated Annotation of Keywords: a Case Study”. In: *WOB*. 2003 (cit. on p. 64).

[204] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. “A study of the behavior of several methods for balancing machine learning training data”. In: *ACM SIGKDD Explorations Newsletter* 6.1 (2004-06), pp. 20–29. doi: 10.1145/1007730.1007735 (cit. on p. 64).

[205] S. M. Lundberg and S.-I. Lee. “A unified approach to interpreting model predictions”. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 64).

[206] C.-Y. Ma et al. “Prediction models of human plasma protein binding rate and oral bioavailability derived by using GA–CG–SVM method”. In: *Journal of pharmaceutical and biomedical analysis* 47.4-5 (2008), pp. 677–682 (cit. on p. 65).

[207] K. Huang et al. “Therapeutics Data Commons: Machine Learning Datasets and Tasks for Drug Discovery and Development”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. Curran, 2021. url: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4c56ff4ce4aaf9573aa5dff913df997a-Paper-round1.pdf (cit. on pp. 65, 68).

[208] S. R. Tangadpalliwar et al. “ChemSuite: A package for chemoinformatics calculations and machine learning”. In: *Chemical Biology & Drug Design* 93.5 (2019-05), pp. 960–964. issn: 1747-0277. doi: 10.1111/cbdd.13479 (cit. on p. 65).

[209] A. J. Minnich et al. “AMPL: A Data-Driven Modeling Pipeline for Drug Discovery”. In: *Journal of Chemical Information and Modeling* 60.4 (2020-04), pp. 1955–1968. issn: 1549-9596. doi: 10.1021/acs.jcim.9b01053 (cit. on p. 65).

[210] M. Korshunova et al. “OpenChem: A Deep Learning Toolkit for Computational Chemistry and Drug Design”. In: *Journal of Chemical Information and Modeling* 61.1 (2021-01), pp. 7–13. issn: 1549-9596. doi: 10.1021/acs.jcim.0c00971 (cit. on p. 65).

[211] B. P. Brown et al. “Introduction to the BioChemical Library (BCL): An Application-Based Open-Source Toolkit for Integrated Cheminformatics and Machine Learning in Computer-Aided Drug Discovery”. In: *Frontiers in Pharmacology* 13 (2022-02). issn: 1663-9812. doi: 10.3389/fphar.2022.833099 (cit. on p. 65).

[212] J. Sieg et al. “MolPipeline: A python package for processing molecules with RDKit in scikit-learn”. In: *Journal of Chemical Information and Modeling* (2024) (cit. on p. 65).

[213] N. Erickson et al. “Autogluon-tabular: Robust and accurate automl for structured data”. In: *arXiv preprint arXiv:2003.06505* (2020) (cit. on p. 67).

[214] M. Kotera, Y. Okuno, M. Hattori, et al. “Computational Assignment of the EC Numbers for Genomic-Scale Analysis of Enzymatic Reactions”. In: *JACS* 126.50 (2004), pp. 16487–98 (cit. on p. 72).

[215] D. Baldazzi, C. Savojardo, P. L. Martelli, et al. “BENZ WS: the Bologna ENZyme Web Server for four-level EC number annotation”. In: *Nucleic Acids Res* 49.W1 (2021), W60–W66 (cit. on p. 72).

[216] S. Quester and D. Schomburg. “EnzymeDetector: an integrated enzyme function prediction tool and database”. In: *BMC Bioinf.* 12.1 (2011), pp. 1–13 (cit. on p. 72).

[217] S. F. Altschul, W. Gish, W. Miller, et al. “Basic local alignment search tool”. In: *J Mol Biol* 215.3 (1990), pp. 403–410 (cit. on p. 72).

[218] L. D. Ferrari et al. “EnzML: multi-label prediction of enzyme classes using InterPro signatures”. In: *BMC Bioinf.* 13 (1 2012-12), p. 61 (cit. on pp. 72, 85).

[219] N. Kumar and J. Skolnick. “EFICAz2.5: application of a high-precision enzyme function predictor to 396 proteomes”. In: *Bioinformatics* 28 (20 2012-10), pp. 2687–2688 (cit. on p. 72).

[220] C. Nagao, N. Nagano, and K. Mizuguchi. “Prediction of Detailed Enzyme Functions and Identification of Specificity Determining Residues by Random Forests”. In: *PLoS One* 9 (1 2014). Ed. by V. de Crécy-Lagard (cit. on p. 72).

[221] T. Sanderson, M. L. Bileschi, D. Belanger, et al. “ProteInfer, deep neural networks for protein functional inference”. In: *eLife* 12 (2023-02). Ed. by V. Dötsch and M. V. Staller, e80942 (cit. on pp. 72–74, 84).

[222] Z. Shi, R. Deng, Q. Yuan, et al. “Enzyme Commission Number Prediction and Benchmarking with Hierarchical Dual-core Multitask Learning Framework”. In: *Research* 6 (2023-01) (cit. on pp. 72, 73).

[223] N. Buton, F. Coste, and Y. L. Cunff. “Predicting enzymatic function of protein sequences with attention”. In: *Bioinformatics* 39 (10 2023-10). Ed. by A. Valencia (cit. on pp. 72, 73).

[224] T. Yu, H. Cui, J. C. Li, et al. “Enzyme function prediction using contrastive learning”. In: *Science* 379 (6639 2023), pp. 1358–63 (cit. on pp. 72, 73, 82, 83, 87).

[225] G. B. Kim, J. Y. Kim, J. A. Lee, et al. “Functional annotation of enzyme-encoding genes using deep learning with transformer layers”. In: *Nat. Commun.* 14 (1 2023-11), p. 7370 (cit. on pp. 72, 73, 87).

[226] A. Dalkiran et al. “ECPred: a tool for the prediction of the enzymatic functions of protein sequences based on the EC nomenclature”. In: *BMC Bioinformatics* 19.1 (2018-12), p. 334 (cit. on p. 72).

[227] A. Bateman, M.-J. Martin, S. Orchard, et al. “UniProt: the Universal Protein Knowledgebase in 2023”. In: *Nucleic Acids Res* 51 (D1 2023-01), pp. D523–D531 (cit. on pp. 72, 73, 85).

[228] J. Mistry, S. Chuguransky, L. Williams, et al. “Pfam: The protein families database in 2021”. In: *Nucleic Acids Res* 49.D1 (2021), pp. D412–D419 (cit. on p. 72).

[229] B. Buchfink, C. Xie, and D. H. Huson. “Fast and sensitive protein alignment using DIAMOND”. In: *Nat. Methods* 12.1 (2015), pp. 59–60 (cit. on pp. 73, 89).

[230] A. Rives, J. Meier, T. Sercu, et al. “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”. In: *Proc Natl Acad Sci* 118 (15 2021-04) (cit. on pp. 73, 87).

[231] Z. Lin et al. “Evolutionary-scale prediction of atomic level protein structure with a language model”. In: *bioRxiv* (2022) (cit. on pp. 73, 87).

[232] A. Elnaggar, M. Heinzinger, C. Dallago, et al. “ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.10 (2022), pp. 7112–7127 (cit. on pp. 73, 87).

[233] E. Fenoy, A. A. Edera, and G. Stegmayer. “Transfer learning in proteins: evaluating novel protein learned representations for bioinformatics tasks”. In: *Brief Bioinf.* 23 (4 2022) (cit. on p. 73).

[234] M. Steinegger, M. Mirdita, and J. Söding. “Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold”. In: *Nature methods* 16.7 (2019), pp. 603–606 (cit. on p. 73).

[235] M. Bileschi, D. Belanger, D. Bryant, et al. “Using deep learning to annotate the protein universe”. In: *Nat Biotech* 40.6 (2022), pp. 932–937 (cit. on p. 75).

[236] M. N. Price, K. M. Wetmore, R. J. Waters, et al. “Mutant phenotypes for thousands of bacterial genes of unknown function”. In: *Nature* 557.7706 (2018), pp. 503–509 (cit. on pp. 82, 87).

[237] C. Crowe et al. “Halogenases: a palette of emerging opportunities for synthetic biology–synthetic chemistry and C–H functionalisation”. In: *Chemical Society Reviews* 50.17 (2021), pp. 9443–9481 (cit. on p. 83).

[238] J. Wehrmann, R. Cerri, and R. Barros. “Hierarchical Multi-Label Classification Networks”. In: *Proc. 35th ICML*. Ed. by J. Dy and A. Krause. Vol. 80. Proc. Mach. Learn. Res. 2018-10–15 Jul, pp. 5075–5084 (cit. on p. 85).

[239] K. Sechidis, G. Tsoumakas, and I. Vlahavas. “On the Stratification of Multi-label Data”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by D. Gunopulos et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 145–158. isbn: 978-3-642-23808-6 (cit. on p. 86).

[240] L. Fu et al. “CD-HIT: accelerated for clustering the next-generation sequencing data”. In: *Bioinformatics* 28.23 (2012), pp. 3150–3152 (cit. on p. 86).

[241] S. Choudhuri. “Sequence Alignment and Similarity Searching in Genomic Databases”. In: Elsevier, 2014, pp. 133–155 (cit. on p. 89).

[242] W. T. Clark and P. Radivojac. “Information-theoretic evaluation of predicted ontological annotations”. In: *Bioinformatics* 29.13 (2013), pp. i53–i61 (cit. on p. 90).

[243] D. Piovesan et al. “CAFA-evaluator: a Python tool for benchmarking ontological classification methods”. In: *Bioinformatics Advances* 4.1 (2024), vbae043 (cit. on p. 90).

[244] J. Menke, J. Massa, and O. Koch. “Natural product scores and fingerprints extracted from artificial neural networks”. In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 4593–4602 (cit. on pp. 92, 94, 97, 98, 112, 129, 140, 144).

[245] H. W. Kim et al. “NPClassifier: a deep neural network-based structural classification tool for natural products”. In: *Journal of Natural Products* 84.11 (2021), pp. 2795–2807 (cit. on pp. 92–94, 98, 109, 112, 124, 129, 132, 140, 144).

[246] L. David et al. “Molecular representations in AI-driven drug discovery: a review and practical guide”. In: *Journal of Cheminformatics* 12.1 (2020), p. 56 (cit. on p. 93).

[247] M. Seo et al. “Development of Natural Compound Molecular Fingerprint (NC-MFP) with the Dictionary of Natural Products (DNP) for natural product-based drug development”. In: *Journal of Cheminformatics* 12.1 (2020), p. 6 (cit. on p. 93).

[248] L.-M. Nollen et al. “Biosynfoni: A Biosynthesis-informed and Interpretable Lightweight Molecular Fingerprint”. In: (2025) (cit. on pp. 93, 98, 101, 112, 115).

[249] G. W. Bemis and M. A. Murcko. “The properties of known drugs. 1. Molecular frameworks”. In: *Journal of medicinal chemistry* 39.15 (1996), pp. 2887–2893 (cit. on p. 93).

[250] CRC Press. *Dictionary of Natural Products on CD-ROM*. Accessed: 5 January 2026. Taylor & Francis Group. Boca Raton, FL, 2019. url: `https://dnp.chemnetbase.com/HelpFiles/DNP_Introduction.pdf` (cit. on p. 93).

[251] D. Probst and J.-L. Reymond. “A probabilistic molecular fingerprint for big data settings”. In: *Journal of Cheminformatics* 10 (2018), pp. 1–12 (cit. on pp. 94, 98, 112, 140, 144).

[252] V. Chandrasekhar et al. “COCONUT 2.0: a comprehensive overhaul and curation of the collection of open natural products database”. In: *Nucleic Acids Research* 53.D1 (2025), pp. D634–D643 (cit. on pp. 94, 112).

[253] B. I. Tingle et al. “ZINC-22 A free multi-billion-scale database of tangible compounds for ligand discovery”. In: *Journal of chemical information and modeling* 63.4 (2023), pp. 1166–1176 (cit. on p. 94).

[254] T.-H. Nguyen-Vo et al. “Predicting antimalarial activity in natural products using pretrained bidirectional encoder representations from transformers”. In: *Journal of Chemical Information and Modeling* 62.21 (2021), pp. 5050–5058 (cit. on pp. 94, 98, 112, 113).

[255] A. Rutz et al. “The LOTUS initiative for open knowledge management in natural products research”. In: *elife* 11 (2022), e70780 (cit. on pp. 94, 112).

[256] C. Hawkins et al. “Plant Metabolic Network 15: A resource of genome-wide metabolism databases for 126 plants and algae”. In: *Journal of integrative plant biology* 63.11 (2021), pp. 1888–1905 (cit. on pp. 94, 115).

[257] J. Capela et al. “Predicting precursors of plant specialized metabolites using DeepMol automated machine learning”. In: *Journal of integrative bioinformatics* 0 (2025), p. 20240050 (cit. on pp. 94, 96, 107, 108, 115).

[258] B. Ramsundar et al. “Is multitask deep learning practical for pharma?” In: *Journal of chemical information and modeling* 57.8 (2017), pp. 2068–2076 (cit. on pp. 96, 117).

[259] K. Yang et al. “Analyzing learned molecular representations for property prediction”. In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3370–3388 (cit. on pp. 96, 117).

[260] D. Rogers and M. Hahn. “Extended-Connectivity Fingerprints”. In: *Journal of Chemical Information and Modeling* 50.5 (2010). PMID: 20426451, pp. 742–754. doi: 10.1021/ci100050t (cit. on p. 98).

[261] M. J. Blanca et al. “Effect of variance ratio on ANOVA robustness: Might 1.5 be the limit?” In: *Behavior Research Methods* 50.3 (2018), pp. 937–962 (cit. on pp. 102, 119).

[262] J. R. Ash et al. “Practically significant method comparison protocols for machine learning in small molecule drug discovery”. In: *Journal of chemical information and modeling* 65.18 (2025), pp. 9398–9411 (cit. on pp. 102, 116).

[263] M. Baranwal et al. “A deep learning architecture for metabolic pathway prediction”. In: *Bioinformatics* 40.7 (2024), btae359 (cit. on pp. 102, 103, 105).

[264] R. Eguchi et al. “Classification of alkaloids according to the starting substances of their biosynthetic pathways using graph convolutional neural networks”. In: *BMC bioinformatics* 20.1 (2019), pp. 1–13 (cit. on pp. 107, 132, 134, 140, 141, 145).

[265] A. L. Prete et al. “Leveraging molecular graphs for natural product classification”. In: *Computational and Structural Biotechnology Journal* (2025) (cit. on pp. 109, 110, 112).

[266] A. P. Bento et al. “An open source chemical structure curation pipeline using RDKit”. In: *Journal of Cheminformatics* 12.1 (2020), p. 51 (cit. on p. 113).

[267] J. Correia, J. Capela, and M. Rocha. “DeepMol: An Automated Machine and Deep Learning Framework for Computational Chemistry”. In: *bioRxiv* (2024), pp. 2024–05 (cit. on pp. 113, 117, 129, 132, 143).

[268] E. D. Huckvale et al. “Benchmark dataset for training machine learning models to predict the pathway involvement of metabolites”. In: *Metabolites* 13.11 (2023), p. 1120 (cit. on p. 116).

[269] C. Nadeau and Y. Bengio. “Inference for the Generalization Error”. In: *Machine Learning* 52.3 (2003), pp. 239–281 (cit. on p. 119).

[270] J. P. Richard. “Protein Flexibility and Stiffness Enable Efficient Enzymatic Catalysis”. In: *JACS* 141.8 (2019-02), pp. 3320–3331 (cit. on p. 120).

[271] V. Porokhin, S. A. Amin, T. B. Nicks, et al. “Analysis of metabolic network disruption in engineered microbial hosts due to enzyme promiscuity”. In: *Metabolic Engineering Communications* 12 (2021), e00170 (cit. on p. 121).

[272] L. F. Salas-Nunez et al. “Machine learning to predict enzyme–substrate interactions in elucidation of synthesis pathways: a review”. In: *Metabolites* 14.3 (2024), p. 154 (cit. on p. 121).

[273] Z. Mou et al. “Machine learning-based prediction of enzyme substrate scope: application to bacterial nitrilases”. In: *Proteins: Structure, Function, and Bioinformatics* 89.3 (2021), pp. 336–347 (cit. on pp. 121, 128).

[274] M. Yang et al. “Functional and informatics analysis enables glycosyltransferase activity prediction”. In: *Nature chemical biology* 14.12 (2018), pp. 1109–1117 (cit. on pp. 121, 128).

[275] S. Goldman et al. “Machine learning modeling of family wide enzyme-substrate specificity screens”. In: *PLoS computational biology* 18.2 (2022), e1009853 (cit. on pp. 121, 129).

[276] A. Kroll et al. “A general model to predict small molecule substrates of enzymes based on machine and deep learning”. In: *Nature communications* 14.1 (2023), p. 2787 (cit. on pp. 121, 129).

[277] A. Kroll, S. Ranjan, and M. J. Lercher. “A multimodal Transformer Network for protein-small molecule interactions enhances predictions of kinase inhibition and enzyme-substrate relationships”. In: *PLOS Computational Biology* 20.5 (2024), e1012100 (cit. on pp. 121, 130).

[278] W. Qian, X. Wang, et al. “Deep Learning-Driven Insights into Enzyme–Substrate Interaction Discovery”. In: *Journal of Chemical Information and Modeling* 65.1 (2024), pp. 187–200 (cit. on p. 121).

[279] P. Bansal et al. “Rhea, the reaction knowledgebase in 2022”. In: *Nucleic acids research* 50.D1 (2022), pp. D693–D700 (cit. on p. 121).

[280] D. Probst et al. “Biocatalysed synthesis planning using data-driven learning”. In: *Nature Communications* 13.1 (2022), p. 964. doi: 10.1038/s41467-022-28536-w (cit. on pp. 121, 149, 166).

[281] J. Capela et al. “Comparative assessment of protein large language models for enzyme commission number prediction”. In: *BMC bioinformatics* 26.1 (2025), p. 68 (cit. on pp. 124, 129).

[282] F. Teufel et al. “GraphPart: homology partitioning for biological sequence analysis”. In: *NAR genomics and bioinformatics* 5.4 (2023), lqad088 (cit. on pp. 127, 129).

[283] H. Huang et al. “Panoramic view of a superfamily of phosphatases through substrate profiling”. In: *Proceedings of the National Academy of Sciences* 112.16 (2015), E1974–E1983 (cit. on p. 128).

[284] M. Martínez-Martínez et al. “Determinants and prediction of esterase substrate promiscuity patterns”. In: *ACS chemical biology* 13.1 (2017), pp. 225–234 (cit. on p. 128).

[285] K. Bastard et al. “Revealing the hidden functional diversity of an enzyme family”. In: *Nature chemical biology* 10.1 (2014), pp. 42–49 (cit. on p. 128).

[286] B. F. Fisher, H. M. Snodgrass, et al. “Site-selective C–H halogenation using flavin-dependent halogenases identified via family-wide activity profiling”. In: *ACS central science* 5.11 (2019), pp. 1844–1856 (cit. on p. 128).

[287] S. L. Robinson, M. D. Smith, J. E. Richman, et al. “Machine learning-based prediction of activity and substrate specificity for OleA enzymes in the thiolase superfamily”. In: *Synthetic Biology* 5.1 (2020), ysaa004 (cit. on p. 128).

[288] G. W. Black, N. L. Brown, J. J. Perry, et al. “A high-throughput screening method for determining the substrate scope of nitrilases”. In: *Chemical Communications* 51.13 (2015), pp. 2660–2662 (cit. on p. 128).

[289] T. Li et al. “Exploration of transaminase diversity for the oxidative conversion of natural amino acids into 2-ketoacids and high-value chemicals”. In: *ACS Catalysis* 10.14 (2020), pp. 7950–7957 (cit. on p. 128).

[290] K. Wang et al. “LOTUS-DB: an integrative and interactive database for Nelumbo nucifera study”. In: *Database* 2015 (2015), bav023 (cit. on pp. 132, 135).

[291] F. Trenti et al. “Early and late steps of quinine biosynthesis”. In: *Organic Letters* 23.5 (2021), pp. 1793–1797 (cit. on p. 138).

[292] J. Guo et al. “De novo biosynthesis of antiarrhythmic alkaloid ajmaline”. In: *Nature Communications* 15.1 (2024), p. 457 (cit. on p. 138).

[293] Y. Hövelmann et al. “Identification of potential human urinary biomarkers for tomato juice intake by mass spectrometry-based metabolomics”. In: *European Journal of Nutrition* 59 (2020), pp. 685–697 (cit. on p. 138).

[294] T. Herraiz. “β-Carboline alkaloids”. In: *Bioactive Compounds in Foods* (2008), pp. 199–223 (cit. on p. 138).

[295] P. Minnaar, M. Van Der Rijst, and J. Hunter. “Grapevine row orientation, vintage and grape ripeness effect on anthocyanins, flavan-3-ols, flavonols and phenolic acids: I. Vitis vinifera L. cv. Syrah grapes”. In: *OENO One* 56.1 (2022), pp. 275–293 (cit. on p. 139).

[296] J. Ng and S. D. Smith. “How to make a red flower: the combinatorial effect of pigments”. In: *AoB PLANTS* 8 (2015-08), plw013. issn: 2041-2851. eprint: `https://academic.oup.com/aobpla/article-pdf/doi/10.1093/aobpla/plw013/47415390/plw013.pdf` (cit. on p. 139).

[297] S. Zheng et al. “Deep learning driven biosynthetic pathways navigation for natural products with BioNavi-NP”. In: *Nature Communications* 13.1 (2022), p. 3342 (cit. on pp. 140, 151).

[298] T. Kim et al. “READRetro: natural product biosynthesis predicting with retrieval-augmented dual-view retrosynthesis”. In: *New Phytologist* (2024) (cit. on p. 140).

[299] E. M. Davis and R. Croteau. “Cyclization enzymes in the biosynthesis of monoterpenes, sesquiterpenes, and diterpenes”. In: *Biosynthesis: aromatic polyketides, isoprenoids, alkaloids* (2000), pp. 53–95 (cit. on p. 141).

[300] M. Shumskaya and E. T. Wurtzel. “The carotenoid biosynthetic pathway: thinking in all dimensions”. In: *Plant Science* 208 (2013), pp. 58–63 (cit. on p. 141).

[301] M. A. T. Zullo and A. Bajguz. “The brassinosteroids family–structural diversity of natural compounds and their precursors”. In: *Brassinosteroids: plant growth and development*. Springer, 2019, pp. 1–44 (cit. on p. 141).

[302] W. Liu et al. “The flavonoid biosynthesis network in plants”. In: *International Journal of Molecular Sciences* 22.23 (2021), p. 12824 (cit. on p. 141).

[303] I. E. Sønderby, F. Geu-Flores, and B. A. Halkier. “Biosynthesis of glucosinolates–gene discovery and beyond”. In: *Trends in Plant Science* 15.5 (2010), pp. 283–290 (cit. on p. 141).

[304] M. Frey, K. Schullehner, R. Dick, et al. “Benzoxazinoid biosynthesis, a model for evolution of secondary metabolic pathways in plants”. In: *Phytochemistry* 70.15-16 (2009), pp. 1645–1651 (cit. on p. 141).

[305] A. Ravikrishnan, M. Nasre, and K. Raman. “Enumerating all possible biosynthetic pathways in metabolic networks”. In: *Scientific Reports* 8.1 (2018), p. 9932 (cit. on p. 146).

[306] A. M. Sequeira, D. Lousa, and M. Rocha. “ProPythia: A Python package for protein classification based on machine and deep learning”. In: *Neurocomputing* 484 (2022-05), pp. 172–182 (cit. on p. 148).

[307] W. Fedus, B. Zoph, and N. Shazeer. “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity”. In: *Journal of Machine Learning Research* 23.120 (2022), pp. 1–39 (cit. on p. 148).

[308] Z. Zeng et al. “CLAIRE: a contrastive learning-based predictor for EC number of chemical reactions”. In: *Journal of Cheminformatics* 17.1 (2025), p. 2 (cit. on pp. 149, 166).

[309] P. Bansal et al. “Rhea, the reaction knowledgebase in 2022”. In: *Nucleic Acids Research* 50.D1 (2022-01), pp. D693–D700 (cit. on p. 149).

[310] A. Chang et al. “BRENDA, the ELIXIR core data resource in 2021: new developments and updates”. In: *Nucleic acids research* 49.D1 (2021), pp. D498–D508 (cit. on p. 149).

[311] D. S. Wishart et al. “PathBank 2.0—the pathway database for model organism metabolomics”. In: *Nucleic acids research* 52.D1 (2024), pp. D654–D662 (cit. on p. 149).

[312] S. Moretti et al. “MetaNetX/MNXref: unified namespace for metabolites and biochemical reactions in the context of metabolic models”. In: *Nucleic acids research* 49.D1 (2021), pp. D570–D574 (cit. on p. 149).

[313] D. E. Graff et al. “Chemprop v2: An Efficient, Modular Machine Learning Package for Chemical Property Prediction”. In: *Journal of Chemical Information and Modeling* (2025) (cit. on p. 149).

[314] T. Duigou et al. “RetroRules: a database of reaction rules for engineering biology”. In: *Nucleic Acids Research* 47.D1 (2019-01), pp. D1229–D1235 (cit. on p. 150).

[315] Y. Wan et al. “Retroformer: Pushing the limits of end-to-end retrosynthesis transformer”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 22475–22490 (cit. on p. 151).

[316] ChEBI Database. *(3R)-hydroxy-β-ionone (CHEBI:53173)*. https://www.ebi.ac.uk/chebi/CHEBI:53173. European Bioinformatics Institute, accessed 2026. 2015 (cit. on p. 157).

[317] A. J. Simkin et al. "The tomato carotenoid cleavage dioxygenase 1 genes contribute to the formation of the flavor volatiles β-ionone, pseudoionone, and geranylacetone". In: *The Plant Journal* 40.6 (2004), pp. 882–892. doi: 10.1111/j.1365-313X.2004.02263.x (cit. on p. 157).

[318] A. Ilg et al. "Tomato carotenoid cleavage dioxygenases 1A and 1B: Relaxed double bond specificity leads to a plenitude of dialdehydes, mono-apocarotenoids and isoprenoid volatiles". In: *FEBS Open Bio* 4 (2014), pp. 584–593. doi: 10.1016/j.fob.2014.06.005 (cit. on pp. 157, 158).

[319] O. Özkaya et al. "Characterization of the free and glycosidically bound aroma potential of two important tomato cultivars grown in Turkey". In: *Journal of Food Science and Technology* 55.11 (2018), pp. 4440–4449 (cit. on p. 157).

[320] P. Winterhalter and G. K. Skouroumounis. "Glycoconjugated aroma compounds: occurrence, role and biotechnological transformation". In: *Advances in Biochemical Engineering / Biotechnology*. Vol. 55. Biotechnology of Aroma Compounds. Berlin, Heidelberg: Springer, 1997, pp. 73–105. doi: 10.1007/BFb0102063 (cit. on p. 157).

[321] Y. M. Tikunov et al. "NON-SMOKY GLYCOSYLTRANSFERASE1 prevents the release of smoky aroma from tomato fruit". In: *The Plant Cell* 25.8 (2013), pp. 3067–3078. doi: 10.1105/tpc.113.114231 (cit. on p. 157).

[322] ChEBI. *β-damascenone (CHEBI:67251)*. Chemical Entities of Biological Interest (ChEBI) database. https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI:67251 (cit. on p. 157).

[323] M. A. Sefton et al. "Occurrence, Sensory Impact, Formation, and Fate of Damascenone in Grapes, Wines, and Other Foods and Beverages". In: *Journal of Agricultural and Food Chemistry* 59.18 (2011), pp. 9717–9746. doi: 10.1021/jf201450q (cit. on p. 158).

[324] R. G. Buttery et al. "Quantitative and sensory studies on tomato paste volatiles". In: *Journal of Agricultural and Food Chemistry* 38.1 (1990), pp. 336–340. doi: 10.1021/jf00091a074 (cit. on p. 158).

[325] M. Distefano et al. "Aroma Volatiles in Tomato Fruits: The Role of Genetic, Preharvest and Postharvest Factors". In: *Agronomy* 12.2 (2022), p. 376. doi: 10.3390/agronomy12020376 (cit. on p. 158).

[326] F.-C. Huang et al. "Substrate promiscuity of RdCCD1, a carotenoid cleavage oxygenase from *Rosa damascena*". In: *Phytochemistry* 70.4 (2009), pp. 457–464. doi: 10.1016/j.phytochem.2009.01.020 (cit. on p. 158).

[327] S. H. Schwartz, X. Qin, and J. A. D. Zeevaart. "Characterization of a novel carotenoid cleavage dioxygenase from plants". In: *Journal of Biological Chemistry* 276.27 (2001), pp. 25208–25211. doi: 10.1074/jbc.M102146200 (cit. on p. 158).

[328] M. Suzuki et al. “Identification of (3*S*,9*R*)- and (3*S*,9*S*)-megastigma-6,7-dien-3,5,9-triol 9-*O*-*β*-D-glucopyranosides as damascenone progenitors in the flowers of *Rosa damascena* Mill.” In: *Bioscience, Biotechnology, and Biochemistry* 66.12 (2002), pp. 2692–2697. doi: 10.1271/bbb.66.2692 (cit. on p. 158).

[329] I. Šikuten et al. “Grape volatile organic compounds: analysis, biosynthesis, and profiling”. In: *Journal of Experimental Botany* 76.11 (2025), pp. 3016–3037. doi: 10.1093/jxb/eraf082 (cit. on p. 158).

[330] X. Meng et al. “Roles of enzymes VvNCED1 and VvCCD1 in *β*-damascenone biosynthesis: insights into aroma development in grapes”. In: *Plant Physiology* 199.3 (2025), kiaf496. doi: 10.1093/plphys/kiaf496 (cit. on p. 158).

[331] H. J. Klee and D. M. Tieman. “The genetics of fruit flavour preferences”. In: *Nature Reviews Genetics* 19.6 (2018), pp. 347–356. doi: 10.1038/s41576-018-0002-5 (cit. on p. 158).

[332] D. Tieman et al. “The chemical interactions underlying tomato flavor preferences”. In: *Current Biology* 22.11 (2012), pp. 1035–1039. doi: 10.1016/j.cub.2012.04.016 (cit. on p. 158).

5.1 Supplementary Information - Chapter 5

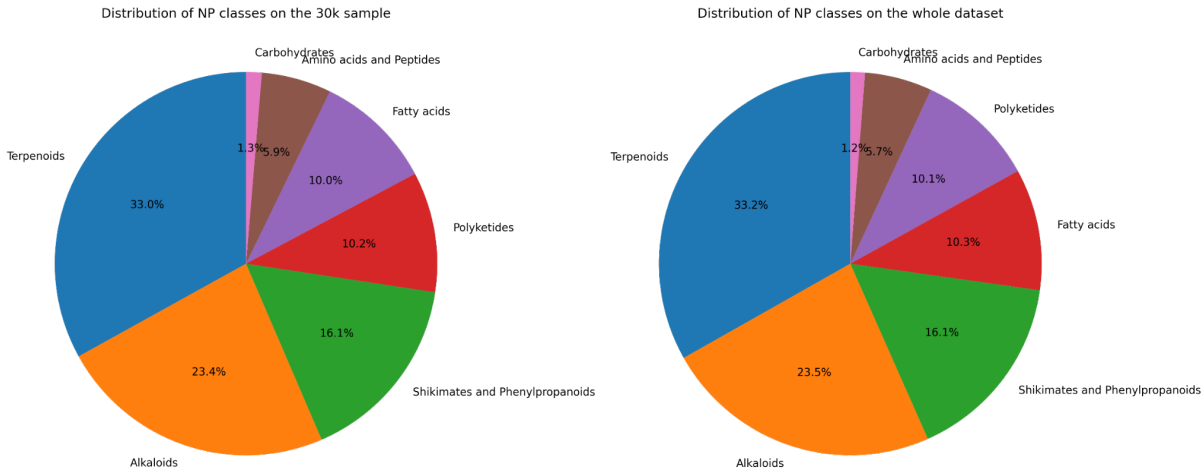


Figure S5.1: Distribution of NP classes for the 30k subsample of the whole dataset.

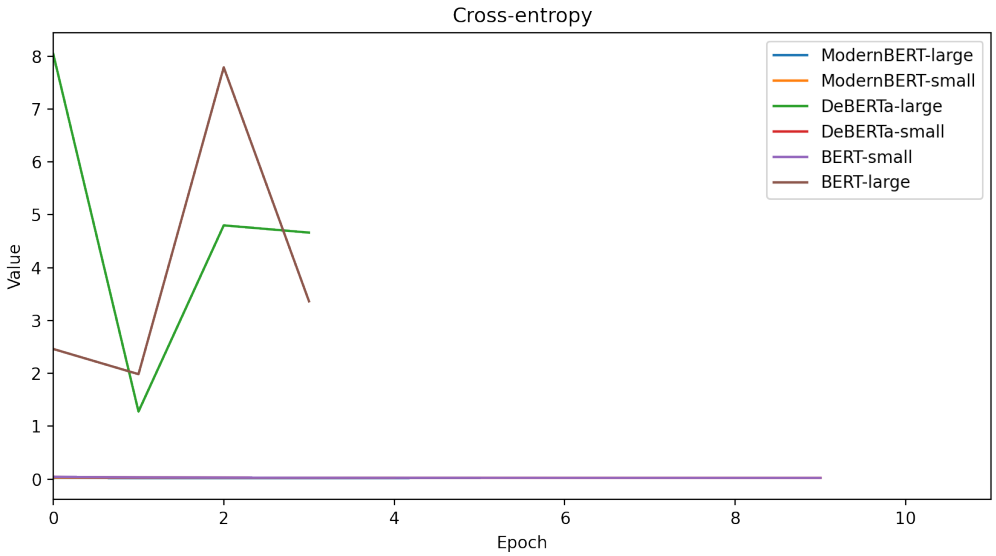


Figure S5.2: Validation loss curves in the masked learning modelling task, showing convergence over epochs for models ModernBERT-small and large, DeBERTa-small and BERT-small.

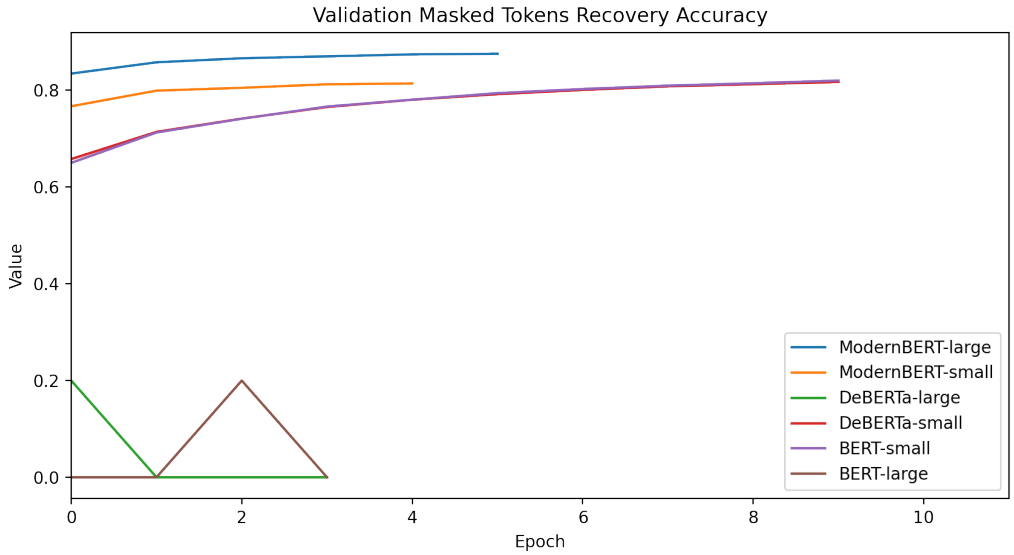


Figure S5.3: Validation masked tokens recovery accuracy over epochs.

Table S5.1: Levene test and max fold difference in variance by metric for KEGG dataset

Metric	**Levene Test p-value**	**Max. Fold Difference**
mF1 score	0.1109	3.93
mPrecision score	0.0099	4.44
mRecall score	0.3191	2.99

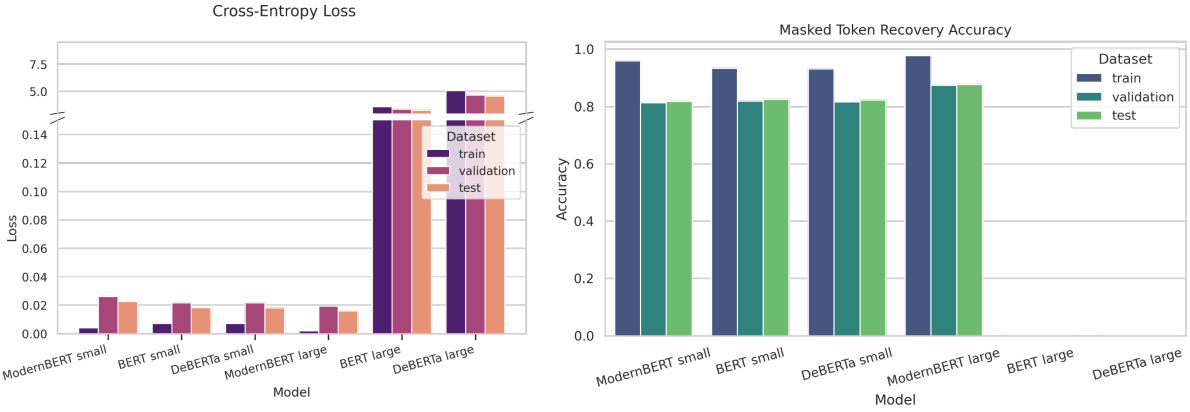


Figure S5.4: Comparison of cross-entropy loss and masked tokens recovery accuracy for training, validation and test sets.

Table S5.2: Levene test and max fold difference in variance by metric for the PlantCyc dataset

Metric	**Levene Test p-value**	**Max. Fold Difference**
mF1 score	0.1564	3.61
mPrecision score	0.0392	4.81
mRecall score	0.4240	3.53

Table S5.3: Levene test and max fold difference in variance by metric for plant precursors dataset

Metric	**Levene Test p-value**	**Max Fold Difference**
mF1 score	0.0490	3.10
mPrecision score	0.0574	3.53
mRecall score	0.0123	3.58

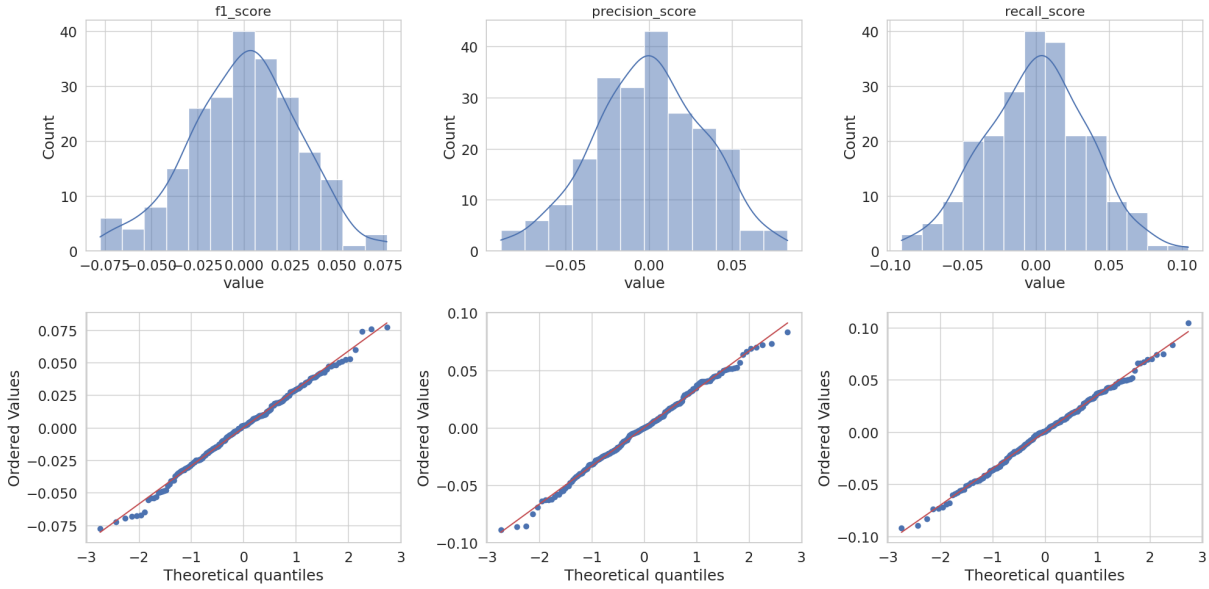


Figure S5.5: Normality tests for the KEGG dataset.

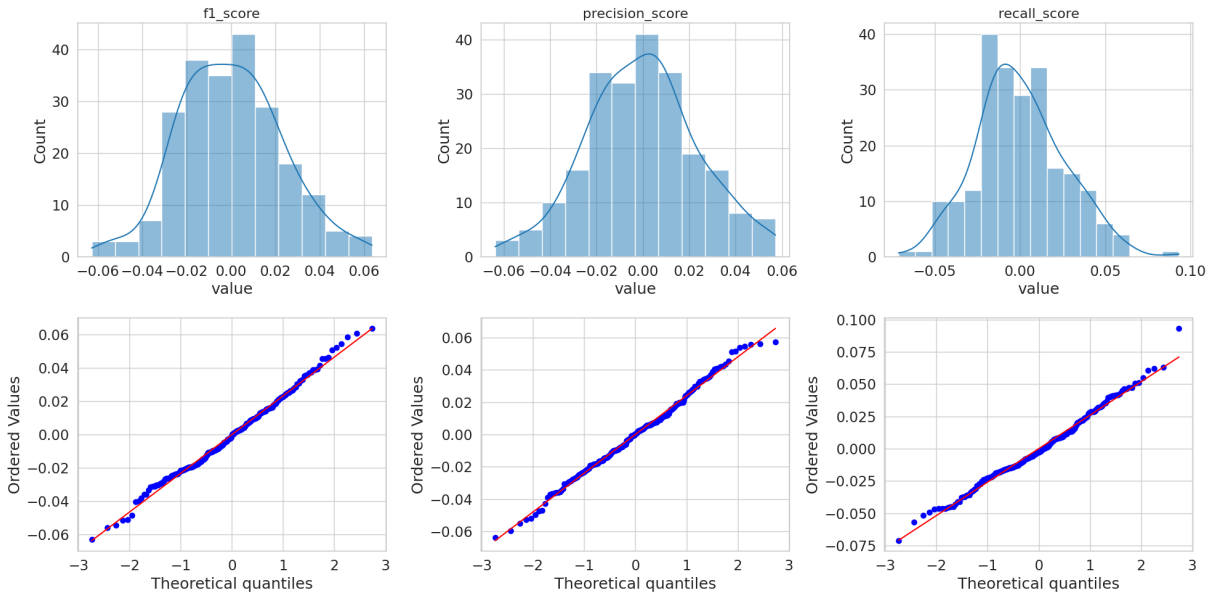


Figure S5.6: Normality tests for the PlantCyc dataset

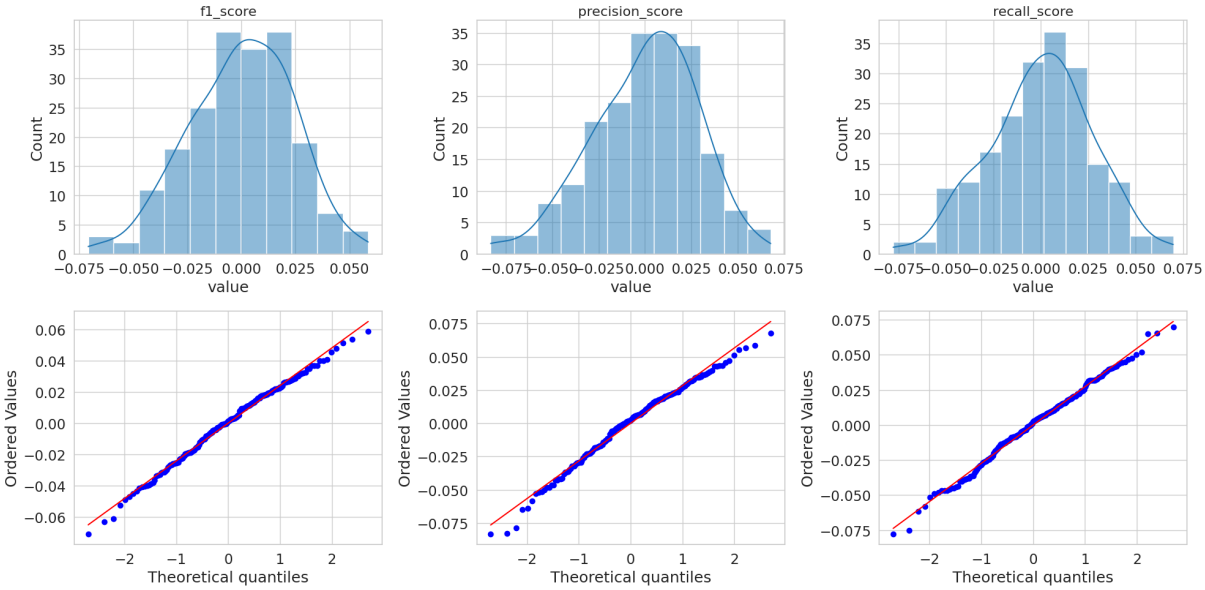


Figure S5.7: Normality tests for the plant precursors dataset

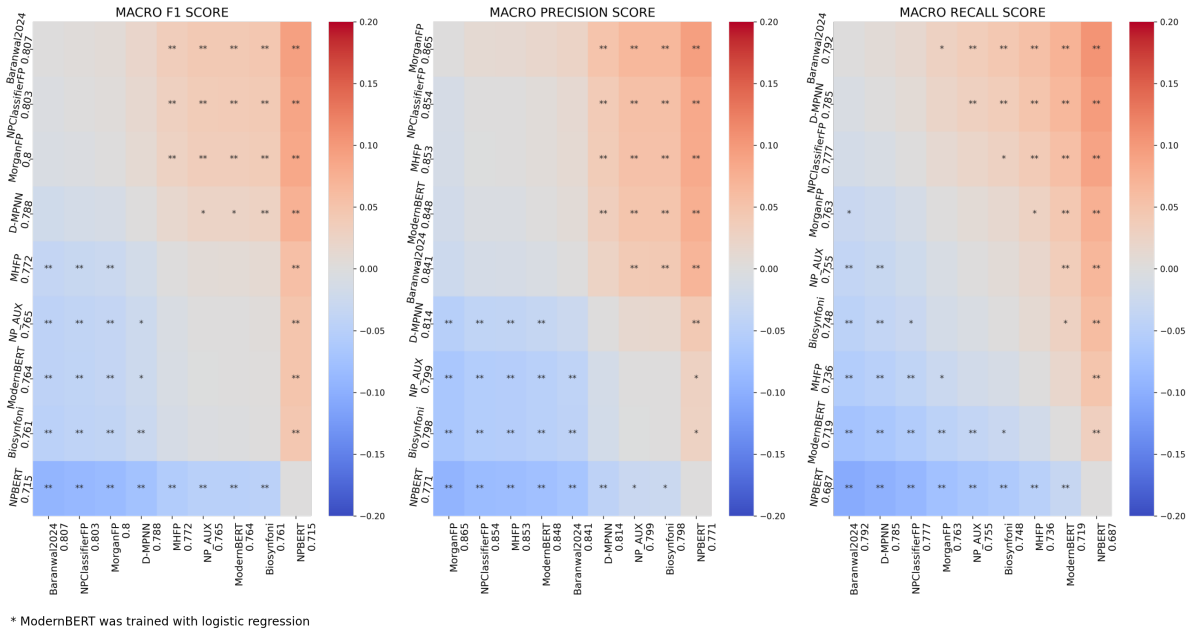


Figure S5.8: KEGG pathway prediction performance and representation analysis. Heatmaps comparing F1 macro, precision macro, and recall macro scores across NP representations, ranked by overall performance (top to bottom). Cells indicate pairwise performance differences with statistical significance denoted by asterisks (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$). Exact metric values are shown below each representation label.

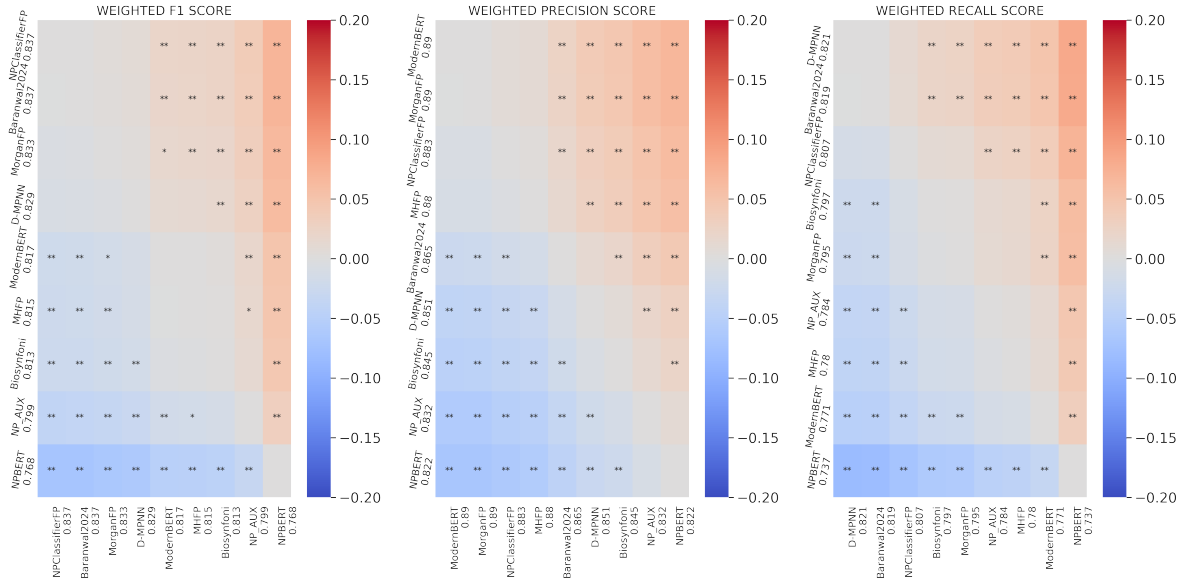


Figure S5.9: Weighted F1, precision and recall for the KEGG dataset.

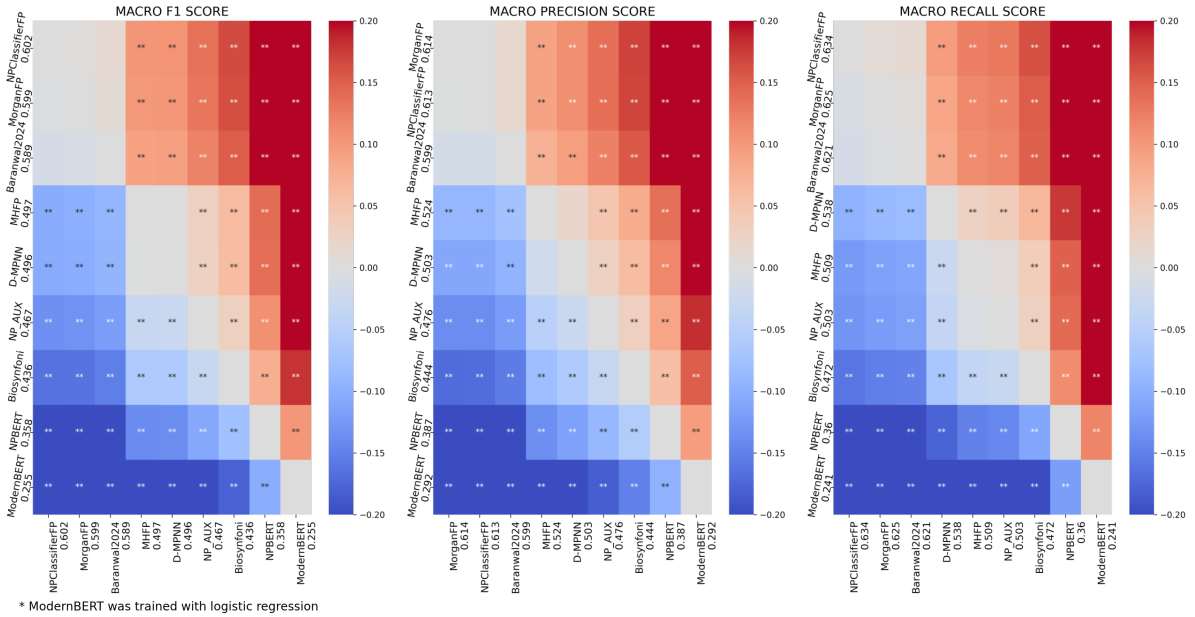


Figure S5.10: PlantCyc pathway prediction performance and representation analysis. Heatmaps comparing F1 macro, precision macro, and recall macro scores across NP representations, ranked by overall performance (top to bottom). Cells indicate pairwise performance differences with statistical significance denoted by asterisks (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$). Exact metric values are shown below each representation label.

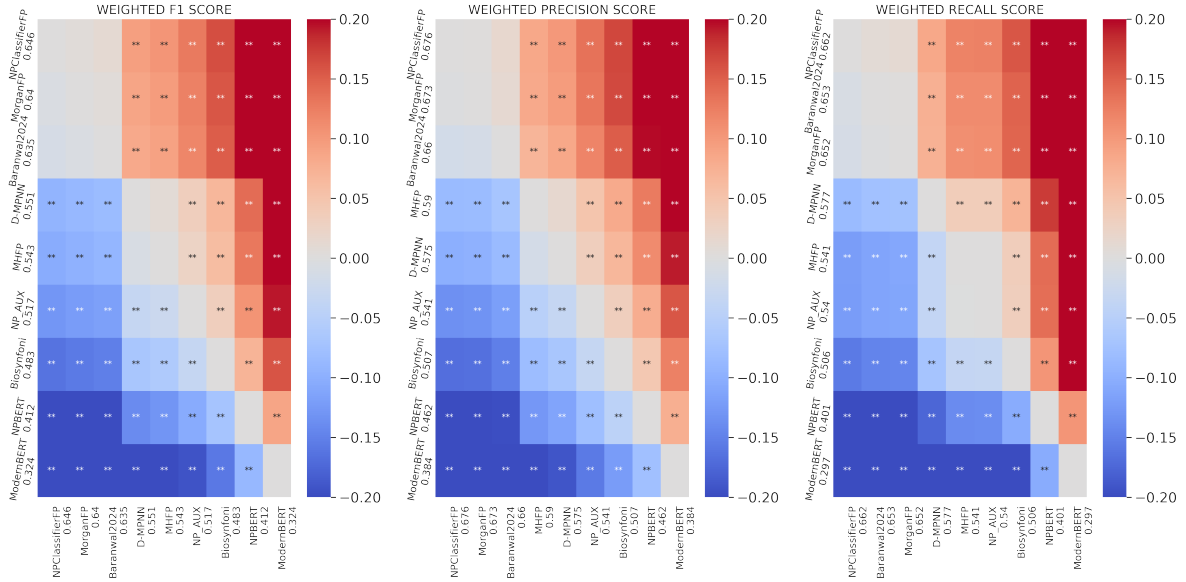


Figure S5.11: Weighted F1, precision and recall for the PlantCyc dataset.

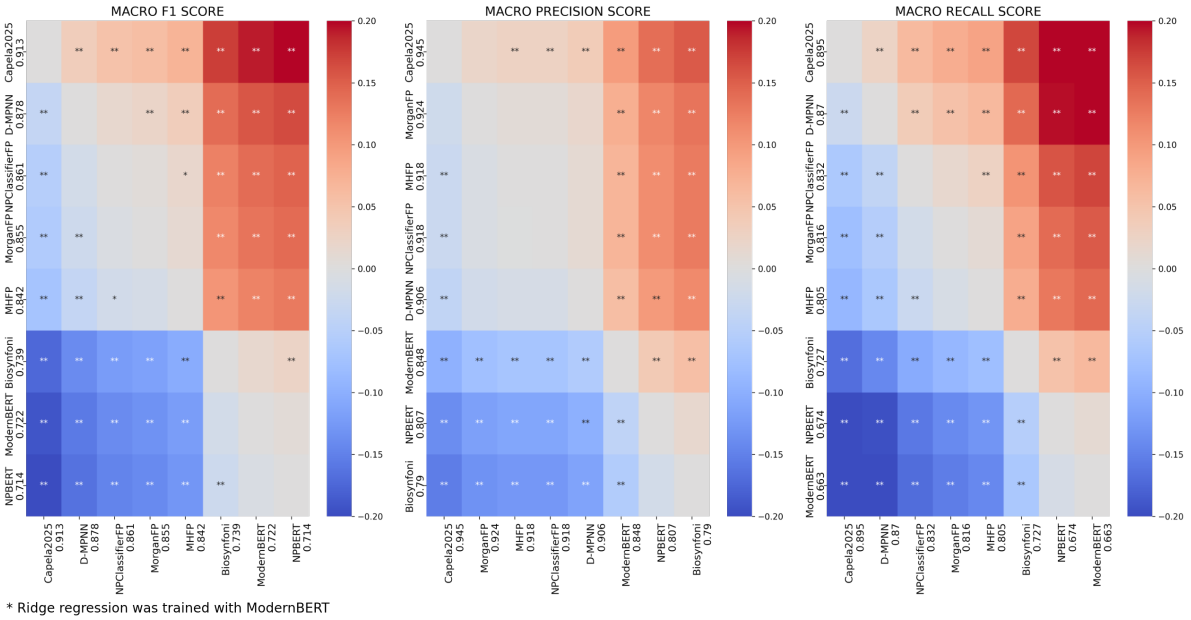


Figure S5.12: Prediction performance and representation analysis of specialised metabolite precursors. Heatmaps comparing F1 macro, precision macro, and recall macro scores across NP representations, ranked by overall performance (top to bottom). Cells indicate pairwise performance differences with statistical significance denoted by asterisks (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$). Exact metric values are shown below each representation label.

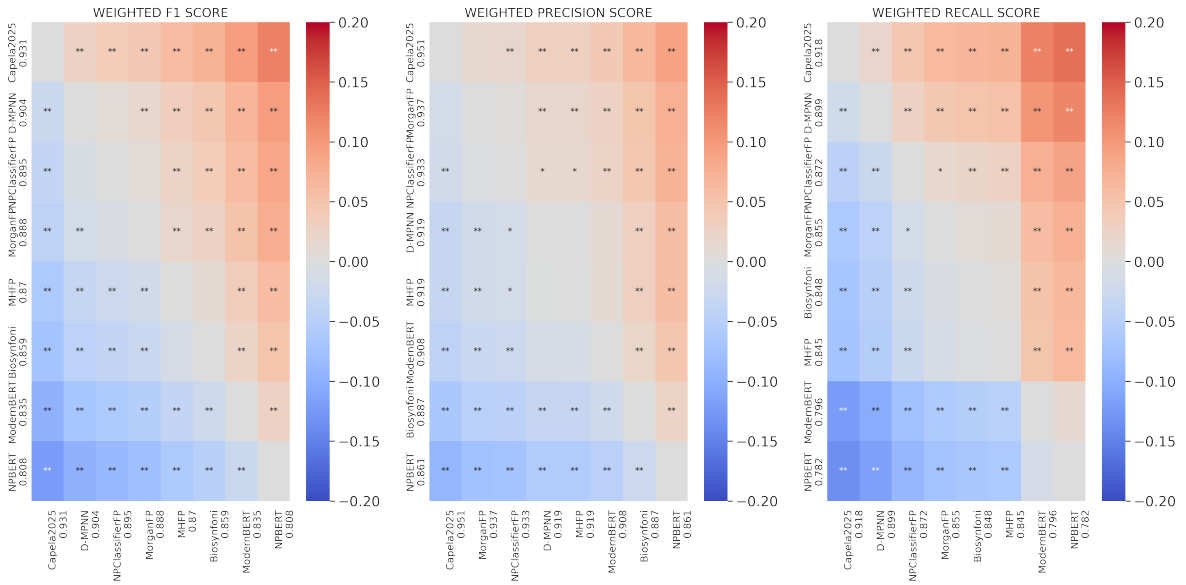


Figure S5.13: Weighted F1, precision and recall for the precursors dataset.

Table S5.4: Comparison of Neural Network (NN) and Linear Models for the KEGG dataset.

Model	mF1		mRecall		mPrecision	
	NN	Linear	NN	Linear	NN	Linear
Biosynfoni	0.761***	0.321	0.748***	0.277	0.798***	0.472
NP_AUX	0.765***	0.526	0.755***	0.459	0.799***	0.696
NPClassifierFP	0.803**	0.779	0.777**	0.743	0.854	0.846
NPBERT	0.715	0.713	0.687*	0.673	0.771**	0.791
ModernBERT	0.629	0.764***	0.596	0.719***	0.740	0.848***
MorganFP	0.800**	0.778	0.763**	0.738	0.865*	0.848
MHFP	0.772	0.765	0.736**	0.762	0.853***	0.800

Note: *** ($p \leq 0.001$), ** ($p \leq 0.01$), * ($p \leq 0.05$)

Table S5.5: Comparison of Neural Network (NN) and Linear Models for the PlantCyc dataset

Model	mF1		mRecall		mPrecision	
	NN	Linear	NN	Linear	NN	Linear
Biosynfoni	0.436***	0.014	0.472***	0.013	0.444***	0.017
NP_AUX	0.467***	0.016	0.503***	0.014	0.476***	0.024
NPClassifierFP	0.602***	0.558	0.634***	0.598	0.613***	0.565
NPBERT	0.358***	0.220	0.360***	0.207	0.387***	0.253
ModernBERT	0.111	0.255***	0.113	0.241***	0.123	0.292***
MorganFP	0.599***	0.567	0.625*	0.607	0.614***	0.571
MHFP	0.497	0.500	0.509	0.591***	0.524***	0.492

Note: *** ($p \leq 0.001$), ** ($p \leq 0.01$), * ($p \leq 0.05$)

Table S5.6: Comparison of Neural Network (NN) and Linear Models for the precursors dataset

Model	mF1		mRecall		mPrecision	
	NN	Linear	NN	Linear	NN	Linear
Biosynfoni	0.739***	0.225	0.727***	0.183	0.790***	0.367
NP_AUX	0.794***	0.393	0.783***	0.336	0.833***	0.547
NPClassifierFP	0.861*	0.873	0.832*	0.844	0.918	0.925
NPBERT	0.714***	0.684	0.674***	0.635	0.807	0.796
ModernBERT	0.512	0.722***	0.490	0.663***	0.618	0.848***
MorganFP	0.855*	0.867	0.816*	0.828	0.924*	0.934
MHFP	0.842	0.834	0.805	0.847***	0.918***	0.851

Note: *** ($p \leq 0.001$), ** ($p \leq 0.01$), * ($p \leq 0.05$)

6.1 Supplementary Information - Chapter 6

Table S6.1: Protein identity cross-partition threshold and the resultant datasets

Identity threshold	Stereo information	Train	Validation	Test
40%	No	39,048	4,181	8,907
40%	Yes	54,690	5,963	12,577
60%	No	35,749	5,413	10,974
60%	Yes	50,220	7,532	15,478
80%	No	36,467	5,241	10,428
80%	Yes	51,159	7,346	14,725

Table S6.2: Compound similarity cross-partition threshold and the resultant datasets

Similarity threshold	Stereo information	Train	Validation	Test
20%	No	45,647	291	6,198
20%	Yes	66,213	294	6,723
40%	No	43,666	4,704	3,766
40%	Yes	63,768	5,237	4,225
60%	No	37,124	5,367	9,645
60%	Yes	56,293	5,836	11,101
80%	No	34,895	5,974	11,267
80%	Yes	50,731	7,495	15,004

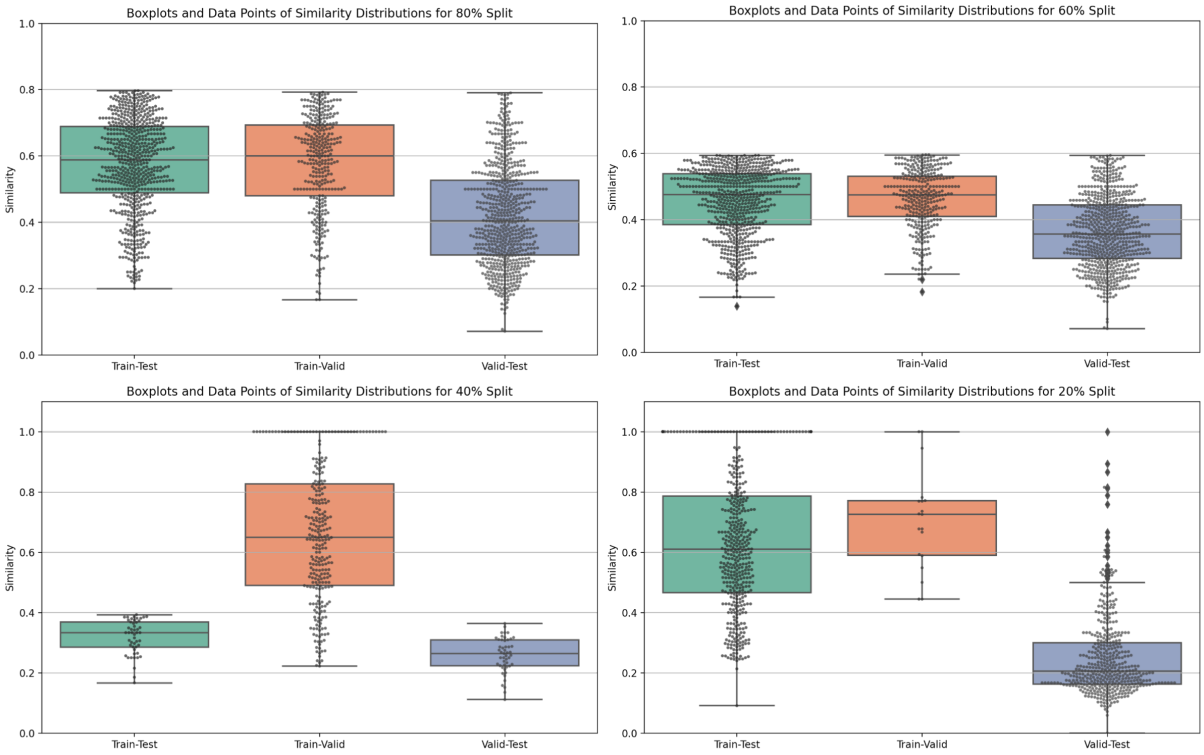


Figure S6.1: Similarity distributions for different dataset splits: 80%, 60%, 40%, and 20%. Each subplot compares three pairs of datasets: Train-Test, Train-Valid, and Valid-Test. The y-axis represents Tanimoto similarity, ranging from 0 to 1. The boxplots illustrate the spread and central tendency of similarity values, while individual data points are overlaid to show the distribution in detail.

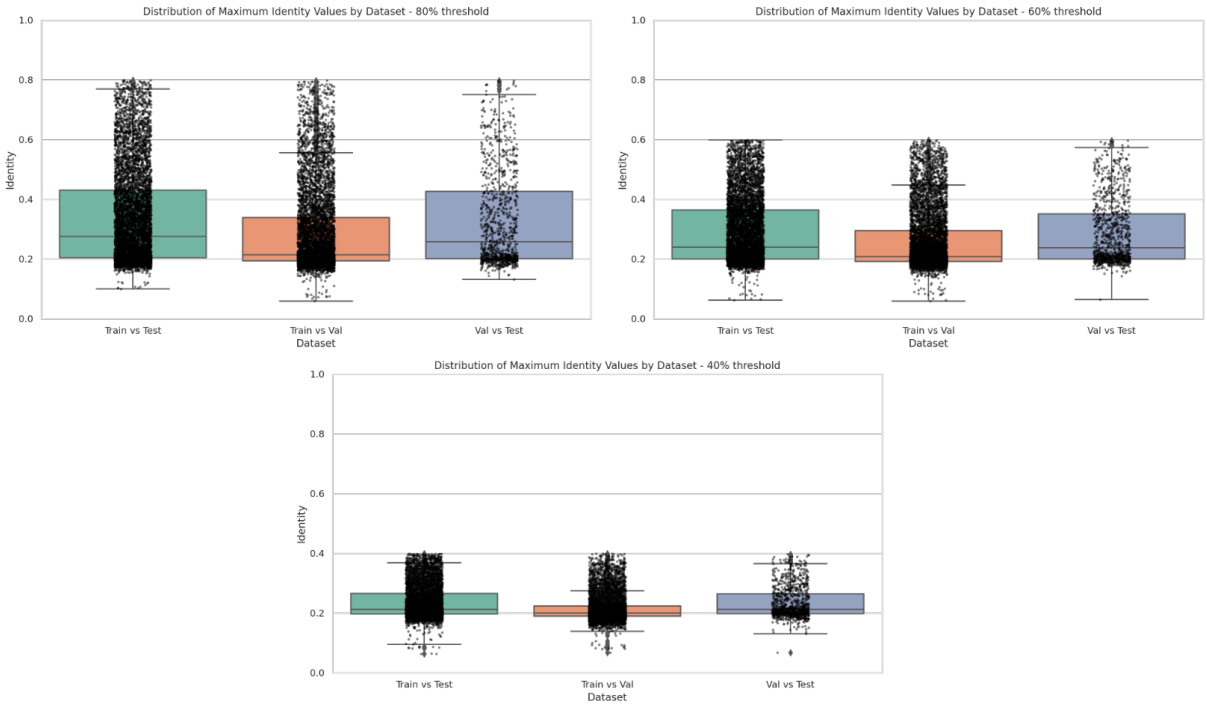


Figure S6.2: Maximum sequence identity values between dataset splits (Train vs Test, Train vs Validation, and Validation vs Test) under four clustering thresholds: 20% (top left), 40% (top right), 60% (bottom left), and 80% (bottom right). For each threshold, the maximum identity of each sequence relative to sequences in the compared split is reported.

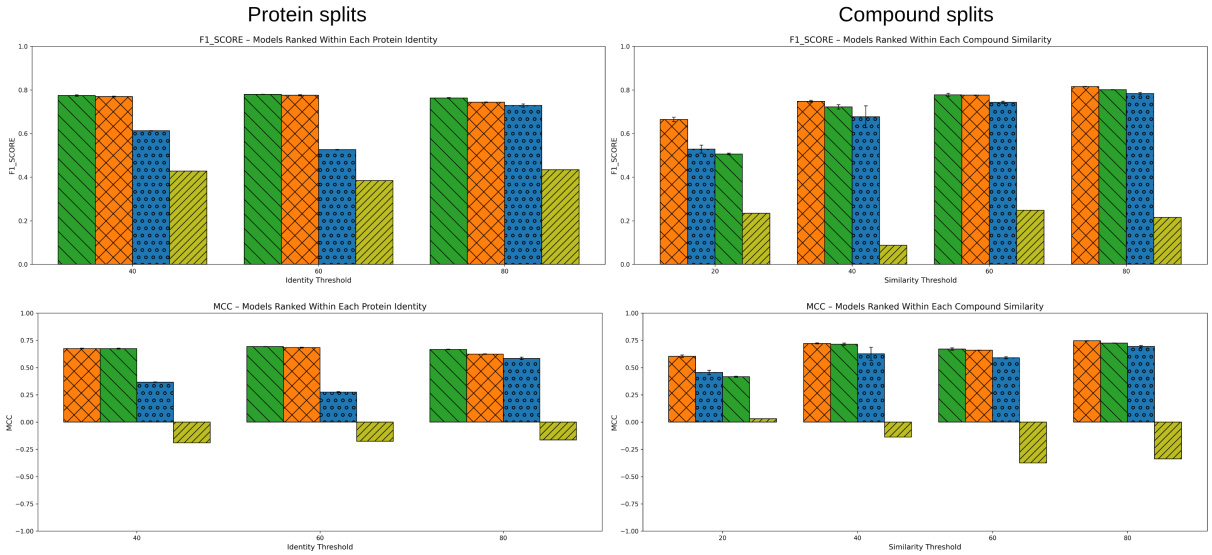


Figure S6.3: Performance of multiple machine learning models in predicting protein-compound interactions across varying identity and similarity thresholds, using datasets stripped of stereochemical redundancy. The top-left and top-right panels illustrate F1 scores as functions of protein sequence identity and compound similarity, respectively, highlighting performance degradation in certain models at higher thresholds. The bottom-left and bottom-right panels depict the Matthews Correlation Coefficient (MCC) under identical conditions, exposing broader performance fluctuations—including negative MCC values at elevated protein identity thresholds. Error bars represent the standard deviation of model performance across trials.

7.1 Supplementary Information - Chapter 7

7.1.1 Similarity matrix and t-SNE generation

A similarity matrix between all the Morgan fingerprints of the compounds in the whole dataset was generated to assess their similarity. The similarity function was the Tanimoto similarity index. A t-distributed Stochastic Neighbour Embedding (t-SNE) was created from this matrix to reduce dimensionality and for visualisation.

7.1.2 AutoML results

Figure S7.1 and S7.2 show the automatic machine learning model results. Figure S7.1 shows the features used during the optimisation and the mF1 score on the validation set for each trial. Morgan and layered FP stood out as the best features. Fig. S7.2 shows the models trained and the mF1 scores obtained by each model on the validation set. The ridge classifiers stood out unequivocally.

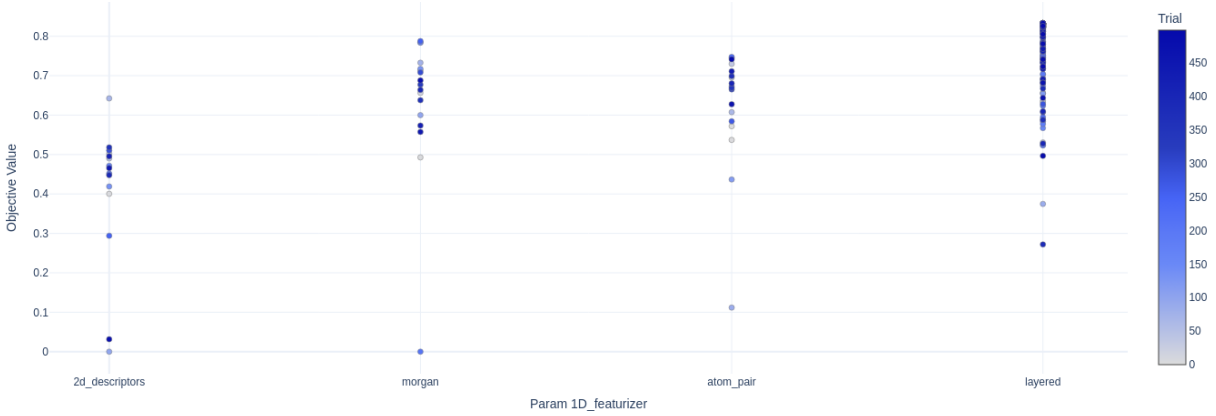


Figure S7.1: mF1 score on the validation set in each trial per type of features.

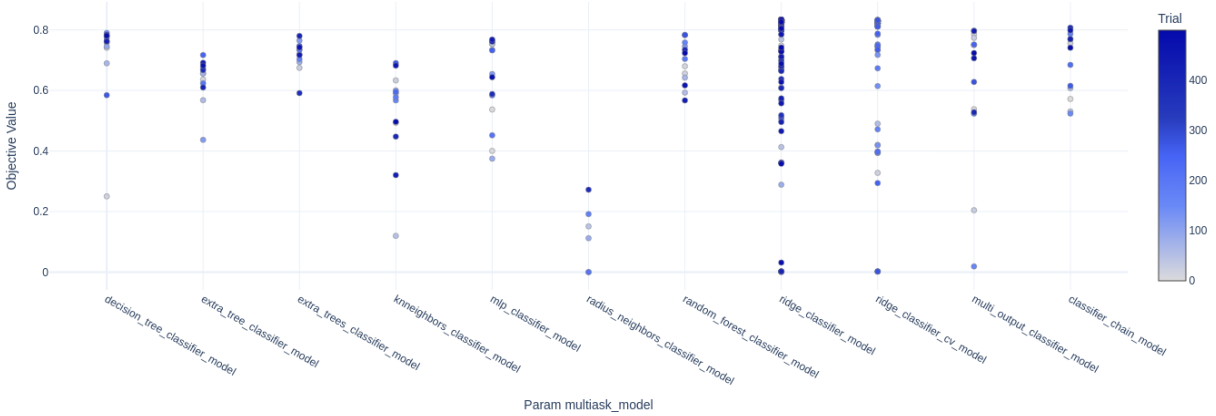


Figure S7.2: mF1 score on the validation set in each trial per model.

7.1.3 Results per label

7.1.3.1 Label F1 scores - test dataset

Fig. S7.3 shows the F1 scores for each label and model.

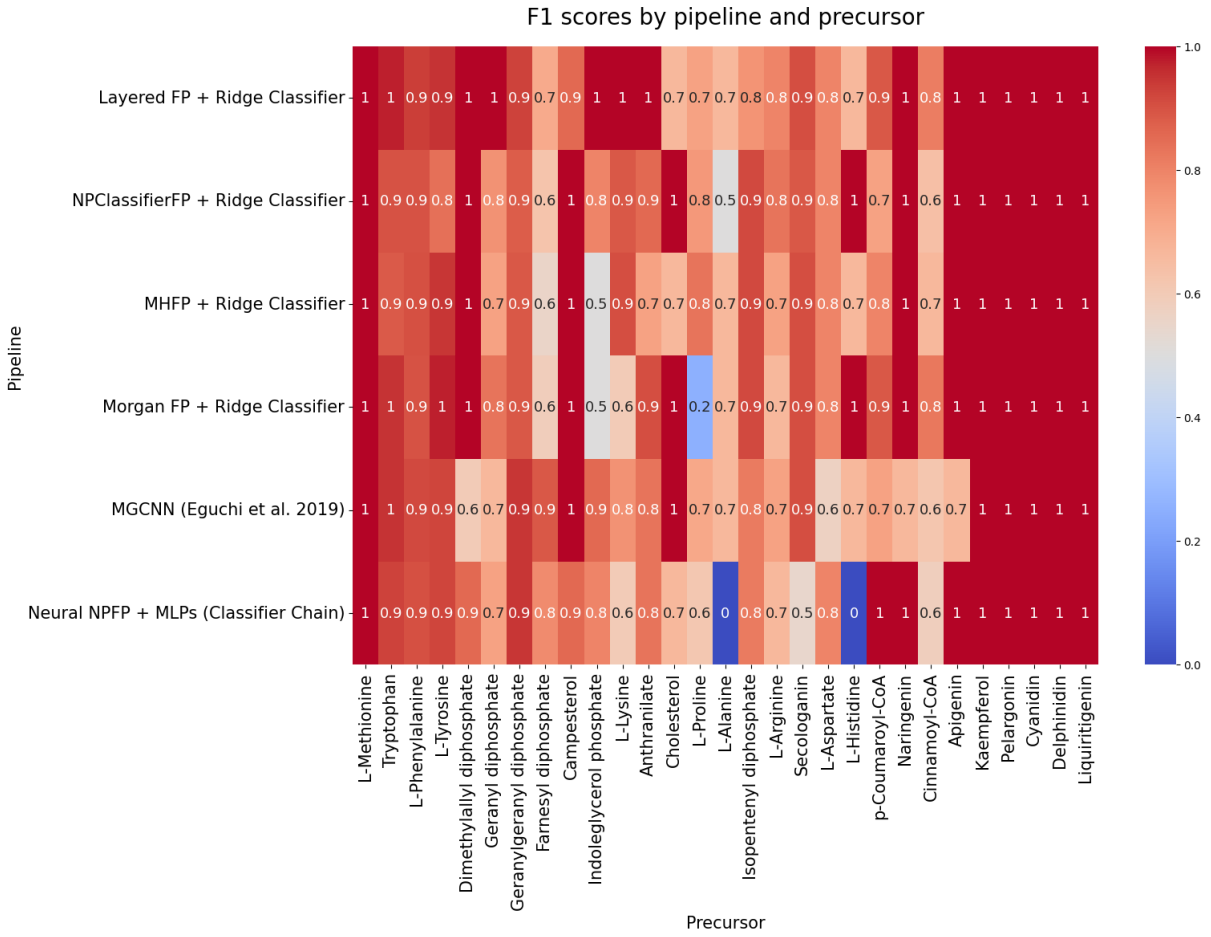


Figure S7.3: F1 scores for each label and model.

7.1.3.2 Label scores - clustered dataset

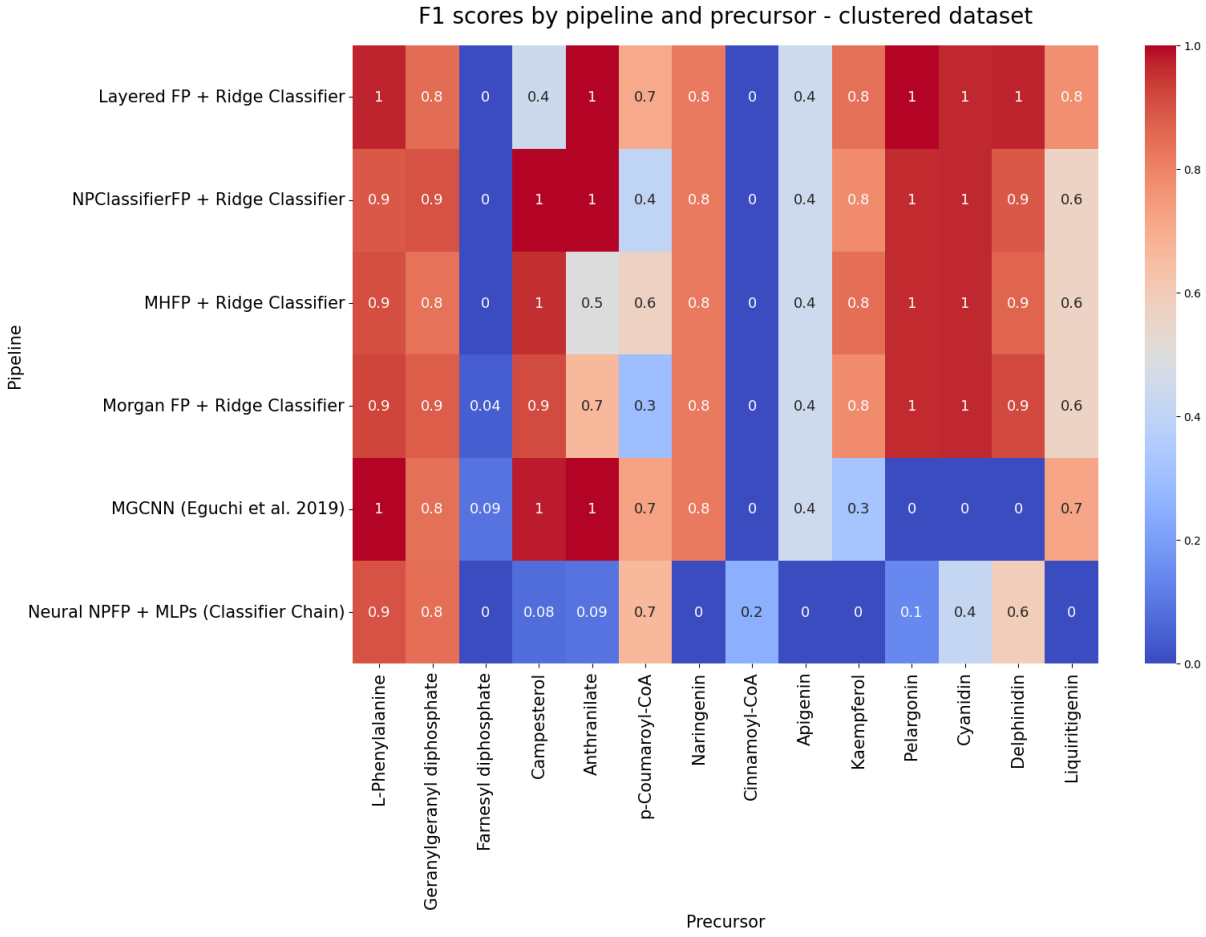


Figure S7.4: F1 scores for each label and model.

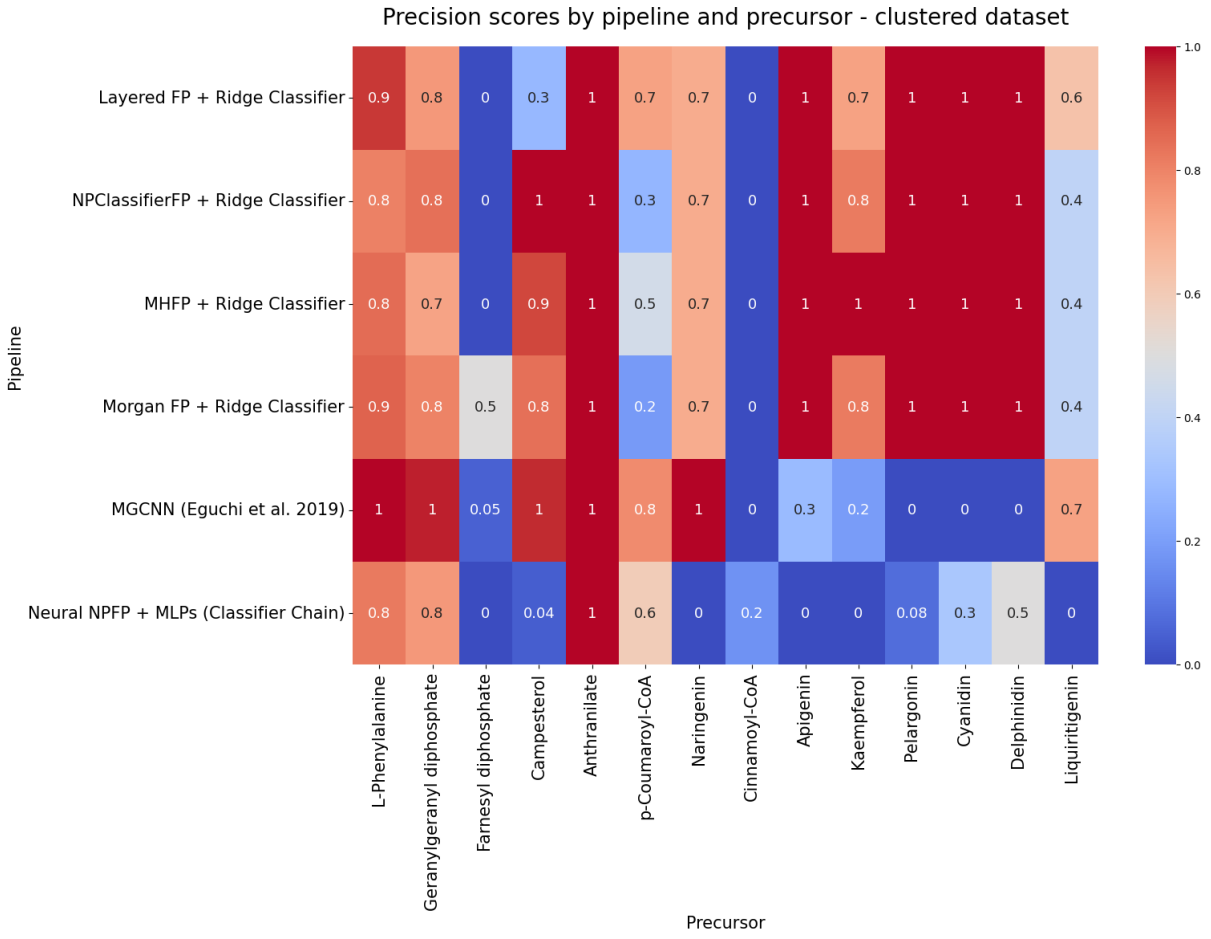


Figure S7.5: Precision scores for each label and model.

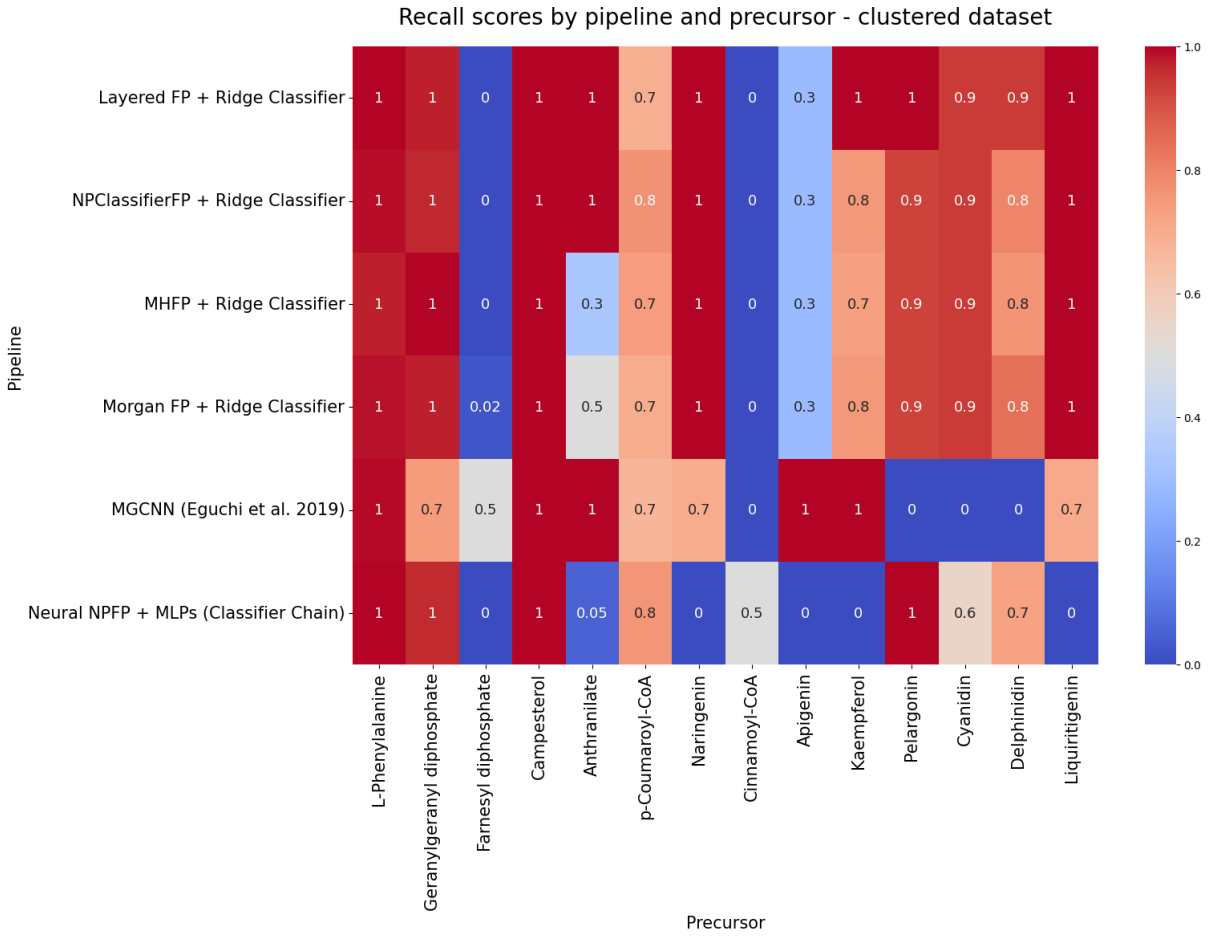


Figure S7.6: Recall scores for each label and model.

7.1.4 Label scores - overlapping dataset

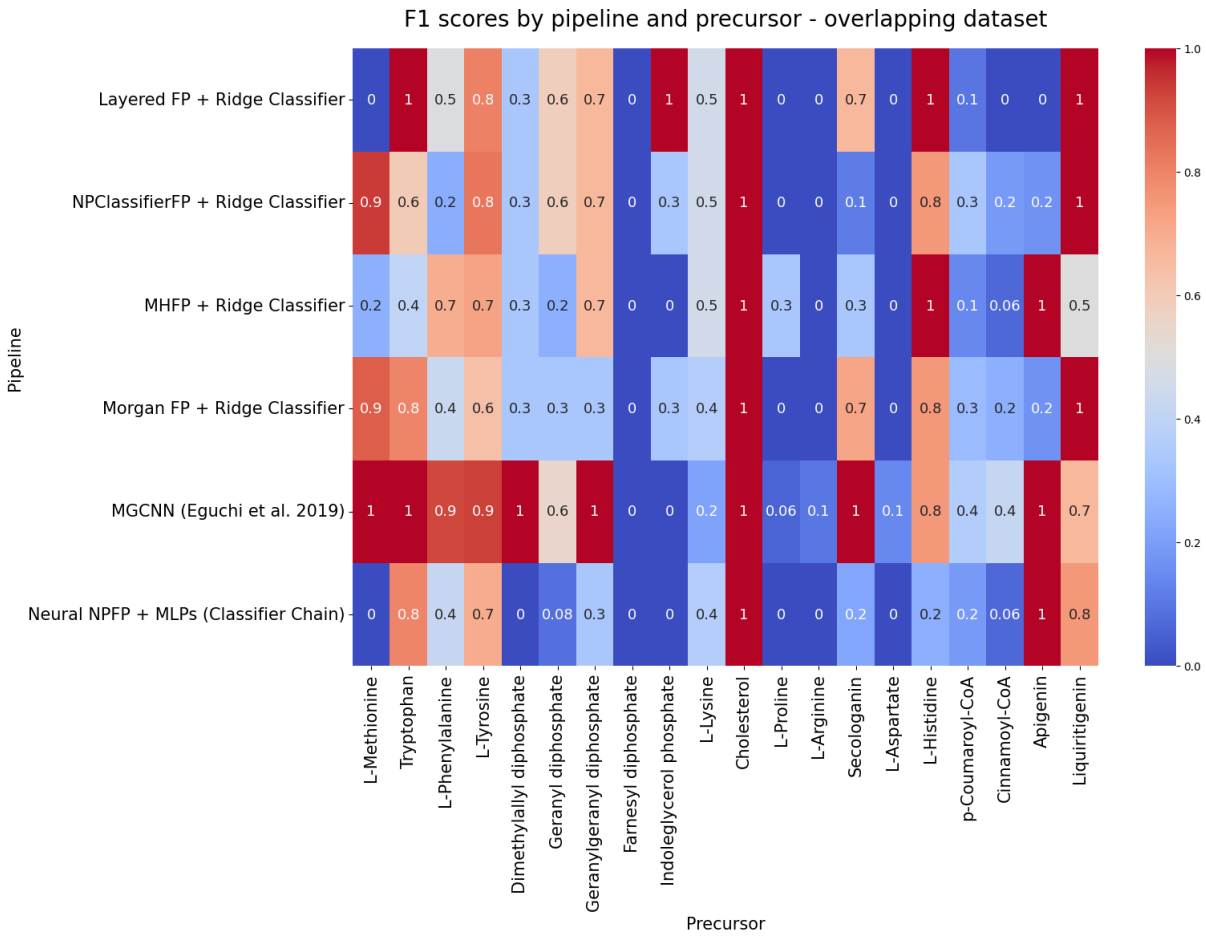


Figure S7.7: F1 scores for each label and model.

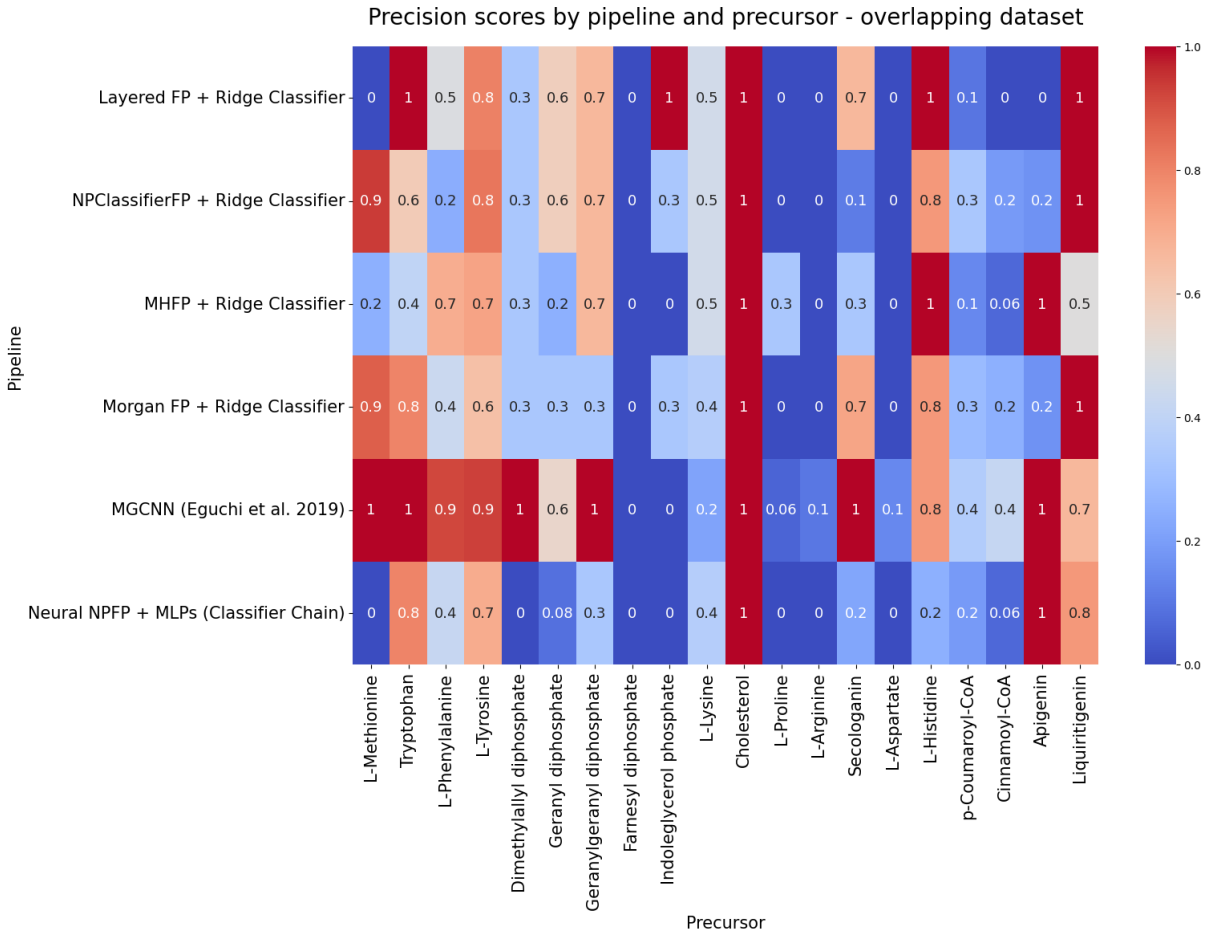


Figure S7.8: Precision scores for each label and model.

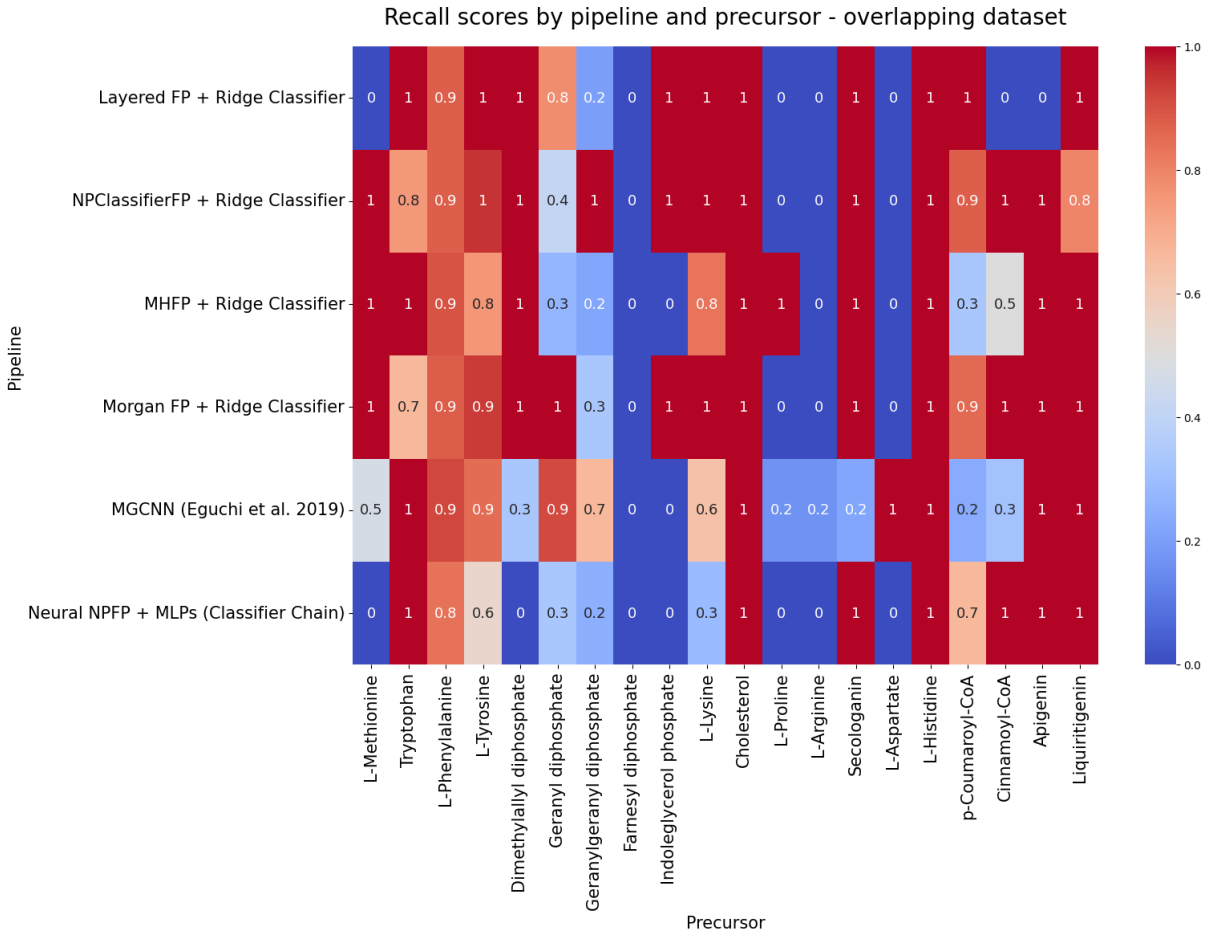


Figure S7.9: Recall scores for each label and model

7.1.5 Statistical differences

In all the tables below, the asterisks are used to denote different levels of statistical significance based on p-values estimated from a Wilcoxon Rank Test. Four asterisks (****) indicate a p-value ≤ 0.0001, signaling a highly significant result. Three asterisks (***) represent a p-value ≤ 0.001, denoting a very significant result. Two asterisks (**) correspond to a p-value ≤ 0.01, marking a significant result, while one asterisk (*) reflects a p-value ≤ 0.05, indicating marginal significance.

7.1.5.1 Test dataset

Table S7.1: Significant differences between pipelines for F1 scores in test dataset

Pipeline 1	Pipeline 2	Differences (F1 score)
Layered FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	*
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*

Table S7.2: Significant differences between pipelines for precision scores in test dataset

Pipeline 1	Pipeline 2	Differences (Precision score)
Layered FP + Ridge Classifier	NPClassifierFP + Ridge Classifier	*
Layered FP + Ridge Classifier	MHFP + Ridge Classifier	*
Layered FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	**
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**
MHFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	*
Morgan FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	**
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*

Table S7.3: Significant differences between pipelines for recall scores in test dataset

Pipeline 1	Pipeline 2	Differences (Recall score)
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*
MHFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	*
MGCNN (Eguchi et al. 2019)	Neural NPFP + MLPs (Classifier Chain)	**

7.1.5.2 Alkaloids dataset

Table S7.4: Significant differences between pipelines for F1 scores in the alkaloids dataset

Pipeline 1	Pipeline 2	Differences (F1 score)
Layered FP + Ridge Classifier	NPClassifierFP + Ridge Classifier	*
Layered FP + Ridge Classifier	MHFP + Ridge Classifier	****
Layered FP + Ridge Classifier	Morgan FP + Ridge Classifier	****
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
Layered FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	***
NPClassifierFP + Ridge Classifier	MHFP + Ridge Classifier	*
NPClassifierFP + Ridge Classifier	Morgan FP + Ridge Classifier	**
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
NPClassifierFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	**
MHFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
Neural NPFP + MLPs (Classifier Chain)	MGCNN (Eguchi et al. 2019)	****

Table S7.5: Significant differences between pipelines for precision scores in the alkaloids dataset

Pipeline 1	Pipeline 2	Differences (Precision score)
Layered FP + Ridge Classifier	MHFP + Ridge Classifier	**
Layered FP + Ridge Classifier	Morgan FP + Ridge Classifier	*
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
Layered FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	****
NPClassifierFP + Ridge Classifier	MHFP + Ridge Classifier	***
NPClassifierFP + Ridge Classifier	Morgan FP + Ridge Classifier	***
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
NPClassifierFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	****
MHFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
MHFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	***
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
Morgan FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	***

Table S7.6: Significant differences between pipelines for recall scores in the alkaloids dataset

Pipeline 1	Pipeline 2	Differences (Recall score)
Layered FP + Ridge Classifier	NPClassifierFP + Ridge Classifier	**
Layered FP + Ridge Classifier	MHFP + Ridge Classifier	***
Layered FP + Ridge Classifier	Morgan FP + Ridge Classifier	***
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
MHFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
MHFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	**
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	****
Morgan FP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	**
Neural NPFP + MLPs (Classifier Chain)	MGCNN (Eguchi et al. 2019)	****

7.1.5.3 Clustered dataset

Table S7.7: Significant differences between pipelines for F1 scores in the clustered dataset

Pipeline 1	Pipeline 2	Differences (F1 score)
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**
MHFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**

Table S7.8: Significant differences between pipelines for precision scores in the clustered dataset

Pipeline 1	Pipeline 2	Differences (Precision score)
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*

Table S7.9: Significant differences between pipelines for recall scores in the clustered dataset

Pipeline 1	Pipeline 2	Differences (Recall score)
Layered FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**
NPClassifierFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*
MHFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	**

7.1.5.4 Overlap dataset

Table S7.10: Significant differences between pipelines for F1 scores in the overlapping dataset

Pipeline 1	Pipeline 2	Differences (F1 score)
Neural NPFP + MLPs (Classifier Chain)	MGCNN (Eguchi et al. 2019)	**

Table S7.11: Significant differences between pipelines for Precision scores in the overlapping dataset

Pipeline 1	Pipeline 2	Differences (Precision score)
MHFP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*
Morgan FP + Ridge Classifier	Neural NPFP + MLPs (Classifier Chain)	*

Table S7.12: Significant differences between pipelines for Recall scores in the overlapping dataset

Pipeline 1	Pipeline 2	Differences (Recall score)
NPClassifierFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	**
MHFP + Ridge Classifier	MGCNN (Eguchi et al. 2019)	*
Neural NPFP + MLPs (Classifier Chain)	MGCNN (Eguchi et al. 2019)	**

8.1 Supplementary Information - Chapter 8

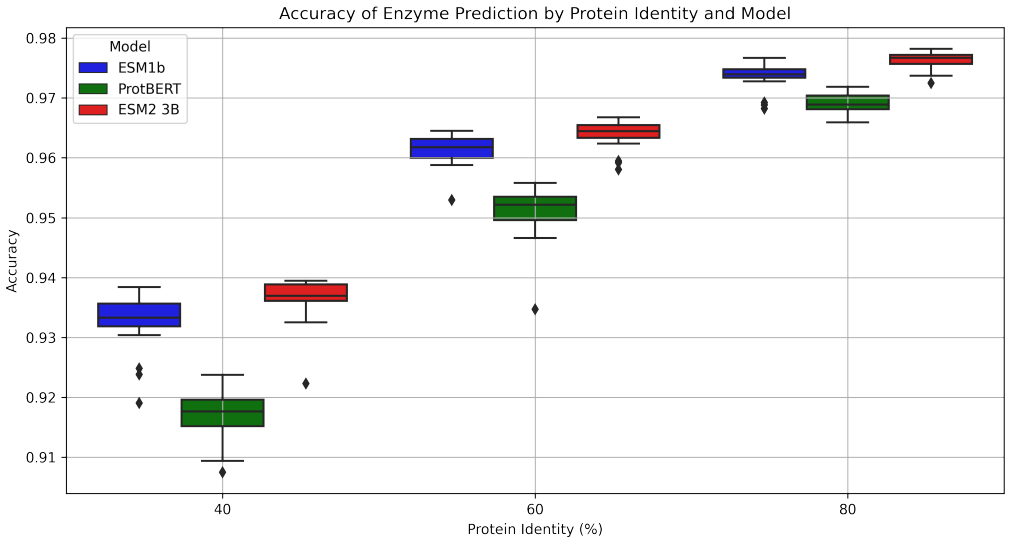


Figure S8.1: Comparison of enzyme prediction accuracy across protein identity levels for ESM1b, ProtBERT, and ESM2 3B models.