Universidade do Minho
Escola de Engenharia
Departamento de Informática

Pedro Daniel Martins Moreira

Implementing a webserver for managing and detecting viral fusion proteins

April 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Pedro Daniel Martins Moreira

Implementing a webserver for managing and detecting viral fusion proteins

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Prof. Miguel Francisco de Almeida Pereira da Rocha
Dr. Diana Andreia Pereira Lousa

April 2021

ACKNOWLEDGEMENTS

First and foremost, this dissertation is funded by COMPETE 2020, Portugal 2020 and FCT - Fundação para a Ciência e a Tecnologia, under the project ″Using computational and experimental methods to provide a global characterization of viral fusion peptides″, through the funding program ″02/SAICT/2017 - Projetos de Investigação Científica e Desenvolvimento Tecnológico (IC&DT)″, with the reference ″NORTE-01-0145-FEDER-028200″, who I would like to thank for their trust.

I would like to thank both of my supervisors for giving all the best support and orientation throughout the development of this dissertation.

I would also like to thank Rúben Rodrigues by being the main help in the development of the webserver and the deployment into the server. Another special appreciation to Ana Marta Sequeira, for providing much needed support to understand the Propythia package and to provide machine learning models and datasets.

I would also like to thank all the people from the cabinet and all my master's colleagues, for all the support and advice provided along these years.

And last, but not least, I would like to thank my family and friends, for all the much-needed motivation they gifted me during these years.

STATEMENT OF INTEGRITY

ABSTRACT

Viral fusion proteins are essential to allow enveloped viruses (such as Influenza, Dengue, HIV and SARS-CoV-2) to enter their hosts' cells, in a mechanism referred to as membrane fusion. This makes these proteins (with special relevance to their fusion peptides, the component of the protein that can insert into the host's membrane by itself) interesting potential therapeutic targets for preventing or treating for some well-known diseases. However, there is no centralized data repository containing all the relevant information regarding viral fusion proteins.

With that in mind, the main purpose of this work is to develop a CRUD (Create, Read, Update and Delete) web server that will allow researchers to find all the necessary data regarding viral fusion proteins, through an easy-to-use web interface. The web application will also contain other bioinformatics functionalities, such as sequence alignment (through BLAST, Clustal and Weblogo) to allow researchers to retrieve key pieces of information regarding a fusion protein, as well as machine learning models capable of predicting the location of fusion peptides inside the viral fusion protein sequence.

The implementation of the server used Django as its back-end, retrieving the data from a MySQL database, and Angular as its front-end.

The main result of the work is, therefore, a working webserver, with a web interface available online through the URL: https://viralfp.bio.di.uminho.pt/.

The web application allows users to explore the gathered data related to viral fusion proteins in a user-friendly way. This tool contains all the proposed functionalities and machine learning models. As expected in an application's development, there are several aspects that require future work to improve the usefulness of this tool to the scientific community.

Key words: Fusion Proteins, Fusion Peptides, Web Server, Machine Learning.

RESUMO

Proteínas virais de fusão são essenciais para que vírus encapsulados (tais como Influenza, Dengue, HIV e SARS-CoV-2) sejam capazes de se inserir nos seus hospedeiros, num mecanismo conhecido como fusão membranar. Por este motivo, estas proteínas (com especial relevância para os seus péptidos de fusão, que são a parte da proteína que se insere na membrana do hospedeiro por si mesma) são potenciais alvos terapêuticos interessantes para prevenir ou tratar algumas doenças bem conhecidas. No entanto, não existe nenhuma fonte de dados centralizada disponível que contenha toda a informação relativa a proteínas virais de fusão.

Sabendo isto, o propósito primário deste trabalho é desenvolver um web server CRUD (*Create, Read, Update and Delete*) que permitirá investigadores encontrar toda a informação necessária relacionada com proteínas virais de fusão, através de um interface user-friendly. Este web server incluirá outras funcionalidades bioinformáticas, tais como ferramentas de alinhamento de sequências (como BLAST, Clustal e Weblogo), que permitirá investigadores extrair informações importantes acerca de uma proteína de fusão. Por fim, incluirá modelos de machine learning capazes de prever a localização de péptidos de fusão na sequência da proteína de fusão.

A implementação do servidor usou Django como seu back-end, que permite extrair a informação da base de dados MySQL, e Angular como front-end.

O principal resultado deste trabalho é, portanto, um web server funcional, com a interface web disponível através do URL: https://viralfp.bio.di.uminho.pt/.

Esta aplicação web permite que utilizadores possam explorar a informação acumulada acerca de proteínas virais de fusão através de uma interface user-friendly. Esta ferramenta contém todas as funcionalidades e modelos de machine learning propostos. Como seria de esperar no desenvolvimento de uma aplicação, existem vários aspetos que requerem trabalho futuro para melhorar a utilidade desta ferramenta para a comunidade científica.

Palavras-chave: Proteínas de Fusão, Péptidos de Fusão, Web Server, Machine Learning.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

A

AAC	Amino Acid Composition.
API	Application Programming Interface.

B

BLAST	Basic Local Alignment Search Tool.

C

CORS	Cross-Origin Resource Sharing.
CTD	Composition, Transition and Distribution.

F

FPEP	Fusion Peptide.

H

HMM	Hidden Markov Model.

K

KNN	K-Nearest Neighbour.

M

MCC	Matthews Correlation Coefficient.
MEI	Mestrado em Engenharia Informática.
ML	Machine Learning.

N

NB	Naïve Bayes.

P

PAAC	Pseudo Amino Acid Composition.

R

REST	Representational State Transfer.
RF	Random Forest.

S

SVM	Support Vector Machine.

U

UM	Universidade do Minho.

V

VFP	Viral Fusion Protein.

1 INTRODUCTION

1.1 MOTIVATION

Enveloped viruses, a type of viruses that includes Dengue, Influenza, HIV and SARS-CoV-2, have an outer lipid envelope that contains one or more types of glycoproteins on its surface (Marsh, 2005, Preface I) (Marsh and Helenius, 2006) (Weissenhorn et al., 2007). These proteins are crucial for viral entry into the host cell, since they bind to receptors on the host cell and catalyse the fusion between the viral envelope and the host cell membrane.

In some viruses, binding to the host and fusion are carried out by the same protein, whereas in other viruses these functions are executed by different proteins. Fusion proteins (VFP) are essential in a process called membrane fusion, which allows the virus to insert its genetic material into the host cell, enabling it to replicate. These proteins function as catalysts in membrane fusion reactions, since without them this process would have a very high energetic barrier (Apellániz et al., 2014) (Harrison, 2015).

Over the years VFPs have been divided into 3 distinct classes: I, II and III, each with their own features, such as its structure, orientation within virus' membrane and location of their fusion peptide (FPep). The FPep corresponds to the segment of the VFP that inserts into the host's membrane, promoting the fusion between the viral envelope and the host cell membrane. This peptide can induce lipid mixing of membrane vesicles even if detached from the rest of the protein (Epand, 2003).

The importance of the VFP and, in particular, the FPep in the process of membrane fusion makes them very promising therapeutic targets for which drugs can be developed to inhibit or treat diseases related to those viruses (Apellániz et al., 2014) (White et al., 2008).

However, even though a lot of information for each of these proteins is available in databases such as UniProt, NCBI Protein and PDB, there no centralized repository that has all known information regarding these proteins, making it necessary to search several data sources when the aim is to have a comprehensive characterization of one or several of these proteins. Thus, it would be very useful to have all the available data on VFPs and FPep centralized in a common repository, which could be easily accessed and used to retrieve important information, analyse different protein features and make useful predictions.

A possible way to display large amounts of data is through a web server. This is a service that allows multiple users to access and utilize stored data (for example within a database) by a web browser in an easy, shared and fast manner (MDN Web Docs, 2019).

To understand better how VFPs (and their components) are characterized, some tools can be deployed, like sequence alignment tools, that allow computational biologists and bioinformaticians to discover a peptide's conserved sequences and several other key pieces of information regarding their features, giving valuable insights about the way they operate (Madden, 2013).

Also, in recent years, there is a growing interest in applying machine learning (ML) models to augment our comprehension regarding these proteins. This type of algorithm has been, for instance, applied on predicting the location of FPeps in the VFP sequences of retroviruses (Wu et al., 2016) (Wu et al., 2019) and predicting inhibitors for different VFPs so that possibly they can be used as potential therapies for some diseases (Xu et al., 2015).

1.2 OBJECTIVES

In this context, the main purpose of this thesis is to develop a CRUD (Create, Read, Update and Delete) web server that will allow to easily store, retrieve, display and analyse the gathered data on VFPs and their respective FPeps, as well as allow authorized users to alter the database, all of which through user-friendly interfaces. This will be done using a relational database, which contains relevant data regarding several hundred VFPs, collected by one of the host groups. The back-end part of this proposed application will be built using Django (and Django REST API) library, which is a Python package built for that purpose, and its front-end will be built using Typescript, through the use of the Angular framework. This webserver and its interface will be freely available and will be useful for the scientific community working with VFPs and FPep.

This web application will include selected bioinformatics tools, such as sequence alignment tools, and ML models developed in previous work capable to detect FPeps within the VFP sequence to expand the capabilities of the web server. Finally, manual curation of the database information will also be performed in order to add / correct entries in the database, by searching several protein databases, such as UniProt, NCBI Protein and PDB, and taxonomy databases (useful for the VFP's host virus and those viruses' host organisms), like NCBI Taxonomy, as well researching through scientific publications.

The following list describes the main tasks performed during the current work:

- Review the relevant literature on viral VFPs to find important information that needs to be added to the database and related applications/software that can be useful to implement the proposed web server (e.g. by linking our webserver to existing tools);

- Develop a CRUD dynamic web server that will allow users to explore the data and authorized users to alter the database, all of which through user-friendly interfaces;
- Include selected bioinformatics tools into the application, such as sequence alignment tools, to expand the capabilities of the web server;
- Include ML models developed in previous work capable to detect FPeps within the VFP sequence;
- Perform manual curation of the database information in order to add / correct entries in the database;
- Writing the master dissertation.

1.3 DISSERTATION'S STRUCTURE

This document is described by the following structure:

Chapter 2: State of the art

Description of the biological problem, referring the already known information regarding the viral fusion process and the viral fusion proteins and its fusion peptides' role in this process. The technological implementation aspects of the tools to be used will be detailed, as well as all the necessary pipelines, that explain how web servers and ML models operate. A review of the proposed ML methodology applied in this field will also be performed.

Chapter 3: Methods

Description of all the methodologies necessary to create the proposed web server application, as well to implement sequence alignment tools and ML models into it. This chapter will mostly contain a deeper description of relevant frameworks that will be heavily utilized during the development of the work. Finally, this chapter contains brief review of the biological databases relevant to this work will be provided.

Chapter 4: Development

Detailing of the development of application and its features. This will contain all the descriptions necessary to explain the functioning of the different components of the application, as well of all relevant complementary procedures.

Chapter 5: Results and Discussion

The main results and issues detected during this dissertation; also includes a small manual of how to use the developed application.

Chapter 6: Conclusion and Future Work

A short analysis whether the proposed objectives were achieved. This chapter presents its contribution for the scientific community and possible directions for future work.

2

STATE OF THE ART

2.1 VIRAL FUSION AND VIRAL FUSION PROTEIN'S ROLE

Enveloped virus

According to the presence or absence of an outer membrane, viruses can be grouped into two distinct groups: enveloped or nonenveloped. Nonenveloped viruses use a protein shell in order to protect their nucleic material. By contrast, enveloped viruses have a lipid membrane bilayer that protects both its capsid and its genome. This lipid bilayer, known as viral envelope, is formed from their host's membranes during the mechanisms of virus assembly and budding (Marsh, 2005, preface I) (Marsh and Helenius, 2006) (Weissenhorn et al., 2007). Enveloped viruses insert part of their viral proteins into the host membrane, fuse their envelope with the membrane of the host cell, insert their genetic material into the host cell and replicate inside their host cell. Enveloped viruses then exit the now infected host cell by budding and secretion (Marsh and Helenius, 2006) (Wolf et al., 2010).

Membrane Fusion

In order for enveloped viruses to insert their genetic material into a host cell, it is required to occur membrane fusion between the viral envelope and a host cellular membrane, bringing those two initially separate membrane bilayers into close contact so that they can merge themselves into a single one (Baquero et al., 2013) (White et al., 2008). An enveloped virus has several copies of one or more VFPs on its membrane, which are in their native pre-fusion conformation, that needs to convert into a fusogenic conformation to initiate the fusion process.

Two events must take place for the fusogenic conformational transition to occur: the first makes that conversion possible, usually by proteolytic cleavage; the second begins the conformational transition, in a process that requires triggering by one of the following: ligand binding (by a coreceptor on the host membrane or in of its internal compartments), proton-binding by pH decrease (usually on the host's endosome, in case of viruses that enter it through endocytosis) or by binding of another protein on the virus outer membrane (Harrison, 2015) (Hughson, 1995) (Teissier and Pécheur, 2007). VFPs are necessary for

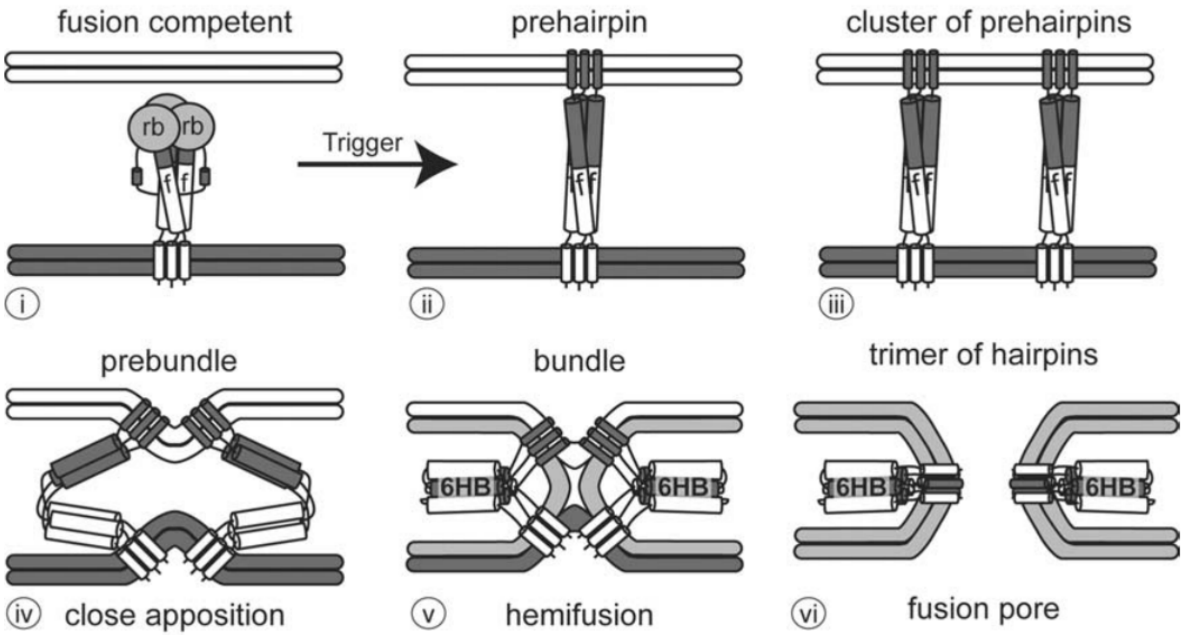


Figure 1: Generic steps of the membrane fusion process, including the different conformations that the fusion protein present on the virus membrane assumes during those steps: (i) the fusion proteins in a fusion competent form in the virus' membrane, upon a trigger, (ii,iii) links to the host's membrane in their prehairpin conformation, (iv) bringing the two membranes into a close contact, (v) eventually forming a hemi-fused state, when the outer layers of the membrane fuse with one another but the inners ones still are not in contact; (vi) finally a pore starts to form when those layers merge; figure from (White et al., 2008).

membrane fusion to occur due to the presence of a high kinetic barrier that the fusion of two membranes must overcome, caused by a repulsive hydration force, which increases massively as the distance between the two membrane surfaces shrinks below 20 Å. Thus, VFPs act as catalyst that make this reaction possible by enabling the virus to overcome this energetic barrier (Harrison, 2015).

According to the currently most accepted model, after the initial apposition step, membrane fusion passes through a "hemi-fusion intermediate", when the outer leaflets of the two membranes merge together, but with the distal ones still not in contact. The hemi-fused state forms a stalk-like structure, shrinking the contact area to its possible minimum so that it can minimize the work needed to overcome hydration force repulsion. The final step of this reaction corresponds to the fusion of the still unattached inner leaflets, starting to form a hydrophilic pore that eventually enlarges, connecting the intermembrane volumes originally separated by the 2 membranes, allowing the viral genome to transferred to the host cell's cytoplasm, enabling the virus to start its replication process (Apellániz et al., 2014) (Harrison, 2015) (Hughson, 1995) (Teissier and Pécheur, 2007). All the steps and the fusion process mentioned above can be observed on the figure 1.

Classes of Fusion Proteins

All VFPs can be described by some common characteristics they all share, such as the fact they are trimers-of-hairpins in their post-fusion form. This structure has a central N-terminal trimeric core, with C-terminal regions packed outside it, allowing to bring the two key hydrophobic components of any viral VFP, the FPep and transmembrane domain, into close functional proximity (White et al., 2008). However, VFPs differ substantially between themselves, which led to the necessity to group them into classes. At the time of writing, the consensus is that there are 3 known distinct classes of VFPs: I, II and III (Apellániz et al., 2014) (Harrison, 2015) (White et al., 2008). The following descriptions may not be applied to all the proteins of each class, since there are excepts for some of the referred characteristics, but they are true for at least most of the class members.

Class I proteins, which include proteins found in Influenza, HIV and SARS-CoV-2 viruses, form spike-shaped structures in perpendicular orientation with the viral membrane. Their main native fusion secondary structure consists in a α-helix. The FPep (structure present in all VFPs, corresponding to the segment that can insert itself into the host's membrane, even if detached from the rest of the protein) (Marsh, 2005, pp. 42) is found buried in subunit interface, in the pre-fusion structure, and is close to the N-terminal on the protein primary sequence. Both their native VFP and their fusion-active structures are trimeric, being that the post-fusion form is a trimer-of-hairpins, and this class is characterized by a 6HB (six-helical bundle) structure, which is a central N-terminal trimeric α-helical coiled coil with three C-terminal helices. The pre-fusion form of the protein is metastable, and the protein needs proteolytic processing to maturate into a fusion competent form. This class contains proteins that can be activated to their fusogenic form by low pH, receptor binding (with or without being followed by low pH) or enzyme-induced covalent modifications (Baquero et al., 2013) (Modis et al., 2004) (Teissier and Pécheur, 2007) (White et al., 2008).

Class II proteins (a class that contains proteins found in Flaviviridae and Togaviridae viruses) lie parallel (near flat) to the virus membrane in the pre-fusion state, and its main secondary structure is a β-sheet. They contain internal FPeps connected to the protein by both ends and located at the tip of elongated β-strands. Unlike Class I proteins, this class' proteins are dimeric in its native structure, albeit also trimeric in its fusion-active form. Their post-fusion conformation is also a trimer-of-hairpins, usually β-structures. Like the previous class, these proteins are metastable and require proteolytic processing, but unlike Class I, this processing occurs on a complementary peptide. The only known activation mechanism for this class is low pH (Baquero et al., 2013) (Teissier and Pécheur, 2007) (White et al., 2008).

Class III was found more recently, therefore information regarding this type of proteins may not be as complete as the former two classes. This class, present in proteins of the Rhabdoviridae and Herpesviridae viruses, is somewhat a hybrid between Class I and Class II features. Proteins from this class orient perpendicularly in relation to the virus membrane

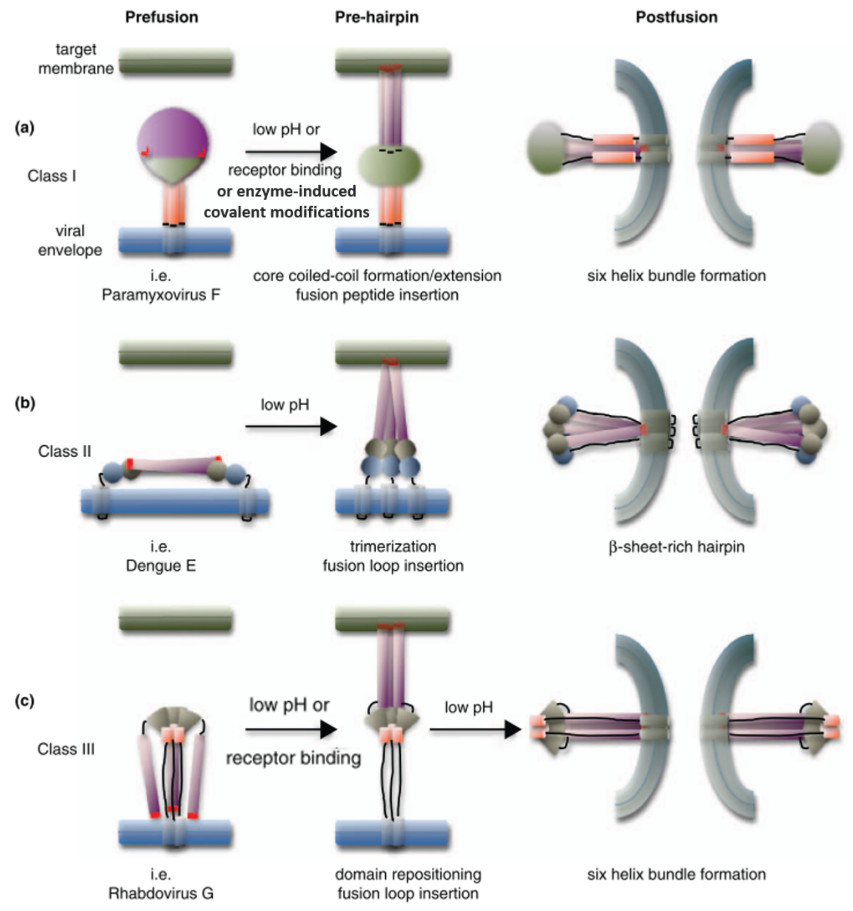


Figure 2: Examples of the differences between proteins of different classes, regarding its structures in prefusion, pre-hairpin and post-fusion conformations, orientation with the outer membrane in the pre-fusion state and the types of possible triggering; figure from (Plemper, 2011), complemented with (White et al., 2008) information.

and can contain both α-helix and β-sheet secondary structures. Like class II, their FPep is internal (in two fusion loops) within the primary sequence and is located at the tip of elongated β-strands, but unlike Class II that have them masked, Class III protein tend to have them exposed. Like Class I, they are trimeric in both native and fusion-active forms, and its post-fusion form is also a trimer-of-hairpins, with a structure having features from the other two classes. However, contrasting both previous classes, these proteins are not metastable and do not require proteolytic processing. This class of protein can be activated by low pH or by receptors (Baquero et al., 2013) (Teissier and Pécheur, 2007) (White et al., 2008).

Some of the differences between the referred classes can be seen in the figure 2, mainly differences in the prefusion, pre-hairpin and post-fusion conformations.

Triggering Mechanisms

As referred above, for the fusion reaction to occur, the VFP needs to be converted into a membrane embedded pre-hairpin and then to the final fusion-competent conformation, which is known as a trimer-of-hairpins. So that this conversion can happen, the protein needs to be activated by a trigger in the form of an environmental stimulus (Marsh, 2005, pp. 27-28) (White et al., 2008).

The current consensus is that there are 4 distinct known triggering mechanisms: exposure to low pH, receptor binding, receptor binding followed by low pH and enzyme-induced covalent modifications. Low pH (at the endosome, after the virus is endocytosed) promotes reactions that lead to structural rearrangements of the protein, upon which the FPep position is shifted, allowing to insert itself into the host's membrane. This trigger mechanism is common in viruses such as orthomyxoviruses, togaviruses and flaviviruses (Marsh, 2005, pp. 28) (White et al., 2008).

Receptor binding (at neutral pH) occurs at cell membrane or at intercellular compartments and it can occur in different conditions, depending of the virus: binding of a host cell receptor, receptor binding to a receptor-binding subunit of the VFP, receptor binding to a separate viral receptor binding protein or binding to several receptors; in this trigger the pH is a non-relevant environment factor, and it is found in paramyxoviruses, herpesviruses and coronaviruses (White et al., 2008) (Marsh, 2005, pp. 29).

Receptor binding that requires to be followed by low pH is a two-step reaction: a receptor binds with the metastable fusion competent protein at physiological temperature (at neutral pH), allowing the protein to perform structural modifications, enabling it to assume its membrane-embedded pre-hairpin state; after that the contact of the protein with low pH converts it into its final state, as a trimer-of-hairpins. This trigger was found in avian retroviruses (Marsh, 2005, pp. 29) (White et al., 2008).

The last known process, which was found in the Ebola virus, also requires low pH, but in order to active the protein into its fusion capable state this mechanism needs the action of endosomal proteases, like cathepsins (B and L) and thermolysin for the case of Ebola (White et al., 2008).

Fusion Peptides

As already mentioned, the FPep is the part of the VFP that can insert into the host's membrane during the fusion process. It can also promote lipid-mixing even if detached from the rest of the protein. These characteristics make it a crucial part of the VFP. This structure has numerous possible forms, but it tends to always follow at least most of the following features (note that the next characteristics are not global to all FPeps, since there are exceptions for most of them): FPeps tend to be hydrophobic structures, with abundant glycine and alanine residues in its sequence, as well as common aromatic residues and finally FPeps tend to be a conserved region (this means that mutations within the FPep decrease or even abolish the activity of the peptide) (Apellániz et al., 2014) (Epand, 2003).

Fusion Proteins' Importance as Therapeutic Targets and in Therapy Clinical Trials

The importance of VFPs in the fusion process makes them interesting therapeutic targets for the diseases associated with those viruses. This is especially true for their FPeps, since, as already seen, they tend be a conserved region. In fact, these conserved elements are known drug targets for development of antivirals and vaccines (Apellániz et al., 2014). Alongside the FPep, another interesting structure for the development of therapies is the final common trimer-of-hairpins, which corresponds to the fusion capable state of the protein (White et al., 2008).

Examples of antivirals that have been researched in recent years include peptides composed of Human Pegivirus' E2 and E1 envelope proteins domains, which target both loop region and the FPep of the HIV-1 gp41 (Gomara et al., 2019), bnAb's that can be used to target Influenza's hemaglutinin (Wu and Wilson, 2018), and compounds derived from heptad repeat regions of the fusion envelope glycoproteins that can inhibit Measles virus fusion mechanism (Figueira et al., 2017).

VFP's importance can also be exemplified by the defining pandemic of 2020: severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). As mentioned before, this is also an enveloped virus, and one of the main therapeutic targets that were studied was its VFP, which in this virus is called spike (S)-protein. Several therapeutic monoclonal antibodies have been experimented that specifically targeted the SARS-CoV-2, naturally with variable degrees of efficacy and toxicity, to become potential epitope-specific therapeutics (Li et al., 2020).

Several of the most promising vaccines being developed for this virus are based on the spike (S)-protein's mRNA. This genetic information is inserted into a lipid nanoparticle, that will enter our cells. Those cells will then produce the protein in its prefusion conformation, which will induce an immune response on the organism, by producing antibodies capable of targeting those proteins and later, if in case of a SARS-COV-2 infection, the virus' VFP. Examples of this type of vaccine are the Moderna's mRNA-1273 (Jackson et al., 2020) and the Pfizer-BioNTech's BNT162b2 mRNA (Polack et al., 2020).

Another example of the importance of VFP is within research for gene therapy clinical trials, like exemplified by the paper by Tomás et al (Tomás et al., 2019), on which the authors describe how VFPs from some enveloped virus, like rhabdovirus vesicular stomatitis virus and γ-retrovirus, are used to pseudotype lentiviral vectors, which are well-known tools capable of gene transfer into mammalian cells. In this context, the VFPs are used to enable these viral vectors to transfer the genes of interest into the target cell, which is crucial in gene therapy strategies.

2.2 RELATIONAL DATABASES

A relational database is defined by its logical structure being solely made up by relations. Mathematically speaking, a relation is a declaration of a table containing attributes (columns) and tuples (rows). This declaration does not define the table's data, but what each column will store. Each row of data is an instance of the declared relation (Harrington, 2016, pp. 89-90).

Each table's column has its own unique name and a domain (normally a data type associated with that attribute). A table's columns must be able to be observed in every possible order without affecting the meaning of its data (Harrington, 2016, pp. 90-91).

Each row must be unique in its values and, for each row, there is a single value for a specific column and a presence of a primary key (a column or a set of columns that can uniquely identify a given row). Like in the case of the columns, the rows must be able to be observed in every possible order without affecting the meaning of its data (Harrington, 2016, pp. 91).

So, in a relational database, the user must be able to access a given piece of data by specifying 3 things: the table's name, that table's columns and the row's primary key. Different tables can have related data using foreign keys. A foreign key is a column of a table's row that match a primary key of another table's row, thus relating the data of those rows (Harrington, 2016, pp. 92, 93 and 98).

So, in relational databases, the data are stored fragmented in multiple tables, which allows to access only the relevant data when a search is made, and to store the data into multiple smaller files, instead of a larger one (Oracle, 2021).

2.3 WEB SERVER

The main objective of this thesis is to develop a web server and, thus, it is important to make some relevant definitions. Indeed, a way to store, retrieve and display a great amount of data is through a web server coupled with a relational database. A web server is a software that receives HTTP (Hypertext Transfer Protocol) requests from a web browser whenever it requires access to a certain file; when this request reaches the correct server, it will send the required document, also through HTTP, like schematically shown in the figure 3 (MDN Web Docs, 2019).

HTTP is a protocol, which means that it is a list of conditions that determine communication between two computers (the client and the server), that must be both textual and stateless. Being textual implies that HTTP's commands ought to be plain-text and human-readable, whilst to be stateless implies previous communications cannot be remembered by either the server or by the client. These conditions include, but are not limited to: HTTP

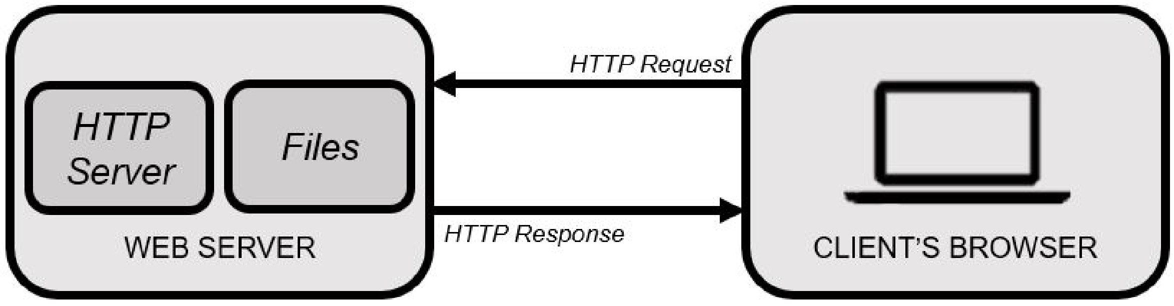


Figure 3: Basic functioning of the HTTP communications between the user's browser and the web server: HTTP request from the client (Browser) leads to the server send back the access to the requested files also trough HTTP; figure inspired by (MDN Web Docs, 2019).

requests can only be sent by clients, they can only be sent to servers and they must always contain files' URLs; the servers must always answer to all requests, even if only with an error message (like the "404 not found" warning).

There are two main types of web servers: static servers, which send the files as they were created and stored to the browser, and dynamic web servers that have the basic structure of a static web server, albeit with additional software, such as an application server and a database, being the main difference that the application server updates the stored files before sending them to the browser (MDN Web Docs, 2019).

A web server has basically 2 fundamental parts: a back-end and a front-end. The back-end serves a multitude of tasks, such as connecting the application to the required databases, assuring correct connection between components and engendering back-end functionality. There are several frameworks available and some of the most popular include ExpressJS (that uses JavaScript), CakePHP and Laravel (both based on PHP), Flask and Django (both using Python). All of them have their strengths and weaknesses, depending on the type of application for which they will be used. The front-end is what the browser user will see, for which the developer will require to create a user-friendly interface to allow an easy utilization of the web server. The front-end is developed using HTML (the content that will show on the browser), CSS (the layout of the browser) and finally JavaScript or TypeScript, both of which describe the behaviour of the web page (Williams, 2019).

To understand several of these frameworks, some other definitions must be understood, like the concepts of API and REST.

An API (Application Programming Interface) functions like a contract that ensures that whenever a client makes a request, the server will always send back a response in a pre-determined format or provide a defined action. APIs allow developers to access, update or delete information and other resources remotely without the need to fully comprehend all the technical details associated with that particular server (Braunstein, 2018, pp. 9).

REST stands for "**RE**presentational **S**tate **T**ransfer". REST guarantees that the server remains stateless (the client and server are independent from each other). This allows the request to contain all the required information to handle the request itself, which reveals extremely important in managing an application with non-periodical requests and with queries that result in different possible outputs (Braunstein, 2018, pp. 9).

2.4 MACHINE LEARNING

Machine Learning is a field within artificial intelligence that consists in algorithms whose purpose is to optimize a certain performance criterion using experience gathered from training datasets. Computationally, this type of algorithms is built by generalizing the problem from learnt experience regarding a certain class of tasks (Alpaydın, 2010, pp. 1-4) (Awad and Khanna, 2015, pp. 1-2).

2.4.1 *Classes of Machine Learning Algorithm*

An algorithm can belong to one of several classes. Supervised learning creates a relationship between the input data and its label (target variable), using its labelled training dataset to build the model function that tries to generalize the relation between the inputs and the desired outputs (figure 4). Unsupervised learning aims to hypothesize representations of input data to obtain predictions of hidden structures in unlabelled datasets; examples of this type of algorithm include clustering and dimensionality reduction. Semi-supervised learning uses features of the previous two types of algorithms, matching a small number of labelled data with a larger number of unlabelled information to create a model; this type of algorithm is promising in human learning, such as speech and vision. There are a few other well-known types of ML, such as reinforcement learning, transductive learning and inductive inference (Alpaydın, 2010, pp. 11) (Awad and Khanna, 2015, pp. 6-9). Here we will focus on supervised ML.

2.4.2 *Pipeline to Develop a Machine Learning Model*

Before starting to develop a supervised ML algorithm, it is important to understand that it has also its shortcomings, mainly that ML will not be able to attain perfect accuracy in most cases. This type of algorithm is most useful when the problem in question requires a large amount of data with unknown patterns. In order to develop this type of algorithm one should usually follow these steps (Awad and Khanna, 2015, pp. 5-6):

1. Collect all the data that can be used to build the model;
2. Preprocess the data so that it can later be used by a ML model; this step does not have a universal pipeline to follow, but may include tasks such as putting the data in an

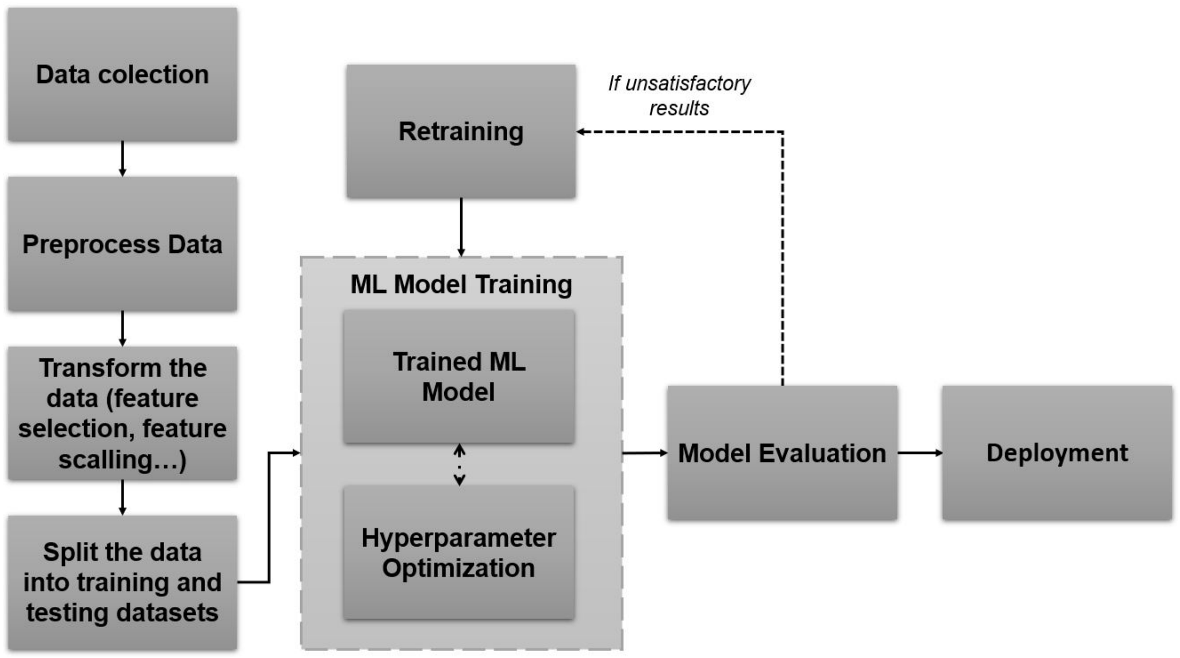

Data colection
Preprocess Data
Transform the data (feature selection, feature scalling...)
Split the data into training and testing datasets
Retraining
If unsatisfactory results
ML Model Training
Trained ML Model
Hyperparameter Optimization
Model Evaluation
Deployment

Figure 4: General procedure of supervisioned learning.

appropriate format, removing or treating missing / incorrect data, sampling it into intervals (so to it can minimize redundancy) and normalizing;

3. Transform the data to attain features adequate to be the input of the training algorithm (this can be performed in the form of feature scaling, decomposition into features, aggregation of several instances into one feature or feature selection);

4. Split the dataset into training and testing datasets, either by defining them manually or through methods like *k*-fold cross-validation, boosting or bagging (these last two methodologies are based on bootstrap);

5. Select one class of ML models and train it: use the training dataset to be the input to training algorithm which will output the trained model; this step will possibly include processes such as hyperparameter optimization to try to improve the overall capabilities of the model;

6. Test the algorithm to validate its effectiveness and performance in a test dataset not used in the previous step; in case of non-desirable results, it might be need to perform additional hyperparameter optimization in the training step;

7. If the last step ended with good results, some other methods can be applied to try to improve the output, like combining different created models to obtain a better overall one;

8. Finally, apply the final model to make predictions in new unlabelled datasets.

2.4.3 *Types of Machine Learning Algorithms*

There are several classes of ML algorithms, each with its methodology and applications. Some of the most relevant and well-known will be reviewed below.

Support Vector Machines

A Support Vector Machine (SVM) model will use its training dataset observations as points in a multiple dimension space, typically applying a transformation to the original points through a kernel function which increases the dimensionality of the space. The algorithm will build a map (hyperplane), which consists in the descriptive surface of the model, that is optimized itself by finding the hyperplane that confers the maximal margin (or functional margin) that can best separate the partitions implied by the class labels. The model represents this optimal hyperplane as a group of support vectors. The label of an unknown observation is predicted as the class corresponding to the partition it will be part of. This type of model has the capability of generating both simpler (linear) or highly complex decision boundaries through the use of kernel functions, allowing them to be applied in several types of problem. They are useful in such tasks such as classification, regression and novelty detection (Awad and Khanna, 2015, pp. 11, 67-68) (Ben-Hur and Weston, 2010) (Noble, 2006).

K-Nearest Neighbours

A K-Nearest Neighbours (KNN) model classifies an unlabelled example by selecting the group of the k examples in the training dataset that are most similar to the example, and labels it based on the most common class label within that group of nearest neighbours. This type of algorithm requires a labelled training dataset, a distance metric (such as Euclidean or Manhattan) to determine the similarity between pairs of examples and the definition of a parameter k. This parameter must be chosen wisely, since different values may lead to very distinct neighbourhoods. This algorithm tends to be computationally less demanding than many other algorithms of ML, which allows to apply it to datasets with multiple dimensions. However, the quality of the training dataset will be a key factor to create an accurate model and large training and testing datasets (or with a high number of dimensions) may lead to models that are too complex (Awad and Khanna, 2015, pp. 11) (Kuang and Zhao, 2009).

Naïve Bayes

Naïve Bayes (NB) is a probabilistic classifier that estimates probabilities for class labels by calculating the frequency of each possible combination of feature and label values in the training dataset. This method uses, as its name implies, the Bayes' Theorem for conditional probabilities to infer class probabilities with a strong assumption of independence, which explains the "naïve" in the algorithm's name. This means that it assumes all attributes are independent from one another (conditional independence), which, even if not completely true in most situations, shortens the algorithm learning time, maintaining still a reasonable approximation in many problems. It is one of the simplest ML algorithms to build and use, since it does not require extensive parameter estimation methods, and it is especially effective in certain applications such as text classification and spam filtering (Awad and Khanna, 2015, pp. 15) (Patil and Sherekar, 2013) (Zhang, 2005).

Classification and Regression Tress

Decision / regression trees are composed by nodes (each will make a test regarding a single feature of the sample) and leaves (the terminal nodes that provide an output). Nodes are connected to the following nodes by edges, that lead a sample to the next test (or, if a leaf, provide an output). To build a model, this algorithm starts with the entirety of the training dataset in the root node, analyses all possible tests (attributes) and selects the one which is the most promising to accurately predict the intended target, thus splitting the dataset into child nodes (according to the possible values of the attribute). This recursive partitioning will repeat for each node until each terminal node (leaf) is associated to only one type of output (when this occurs the leaf is called pure), which may not be possible for all models. To reduce the complexity of the model, so that it can become less prone to overfit, the algorithm can either stop creating branches early (pre-pruning) or try to remove some of the less useful branches (pruning). The prediction of an unlabelled sample corresponds to verify on which partition of the feature space it belongs, providing the output of the leaf reached when navigating the tree based on the feature values of the example (Awad and Khanna, 2015, pp. 15) (Loh, 2011) (Müller and Guido, 2016, pp. 70-74).

Linear Regression

For regression, a linear function is described by the following equation:

$$\hat{y} = \sum (w[i] * x[i]) + b, i \epsilon [0, p] \tag{1}$$

where $\hat{y}$ is the predicted output, $x[i]$ corresponds to the value of the i-th feature for a given example, p is the number of features present in the function, and w and b the parameters that describe the model.

This model tries to find the w and b parameters by minimizing the mean squared error. i.e. the sum of the squared differences between the correct outputs (y) of the training dataset and the predictions ($\hat{y}$). By not requiring to define hyperparameters, this model is simple to build and train, but does not have ways to control its own model complexity.

One alternative to address this issue is using regularization methods, which aim to restrict a model to prevent overfitting. One example of regularization is Ridge regression (also called L2 Regularization), which follows the same basic principles of linear regression, but also tries to keep the w (coefficients) as close to zero as possible, thus minimizing the importance of each individual feature in the model. Another common method is Lasso (or L1 Regularization), which is similar to Rigde, but penalizing weight values by its absolute value (L1-norm). This typically leads some of the w values to be exactly zero, removing all the importance given to some of the features present in the dataset (which in essence is also a form of feature selection) (Müller and Guido, 2016, pp. 45-53).

Logistic Regression

A Logistic Regression model corresponds to a probabilistic statistical classification algorithm. This model forms multivariate regression relations between a dependent outcome (Y) and several independent attributes (X). The logistic function (P(Y|X), as seen in the figure 5, has all its possible outputs contained between 0 and 1, and the model will use this function to select the predicted output of an unknown observation. In binary classification problems, values of P(Y|X) under 0.5 will lead the model to assume a 0 (zero) label, and an outcome of 1 will be assumed if the value is bigger than that threshold (Awad and Khanna, 2015, pp. 22-23) (Lee, 2005).

Artificial Neural Networks

As its name implies, this type of model is inspired in the structure of a biological neuron. In this structure, a given number of inputs, each with its own weight, enter the neuron, which sums those inputs and passes the value of that operation to an activation function, giving the corresponding output. An artificial neural network (ANN) uses several layers of these neurons (or nodes). It has one input layer, one or several hidden ones and an output one. The hidden layers, as well as the output one, are characterized by having its specific weight matrix and bias vector (structure that allows to shift the flow of data), and output vector, allowing to construct complex structures.

The optimal number of hidden layers, and how many neurons each layer should have, will depend greatly on the problem at hands. Each output for a neuron corresponds to a non-linear transformation of the inputs. Increasing the number of neurons increases the risk of overfitting of the model, and it will struggle to generalize new data, which will worsen the overall performance of the model. So, during the training stage of this model, it must

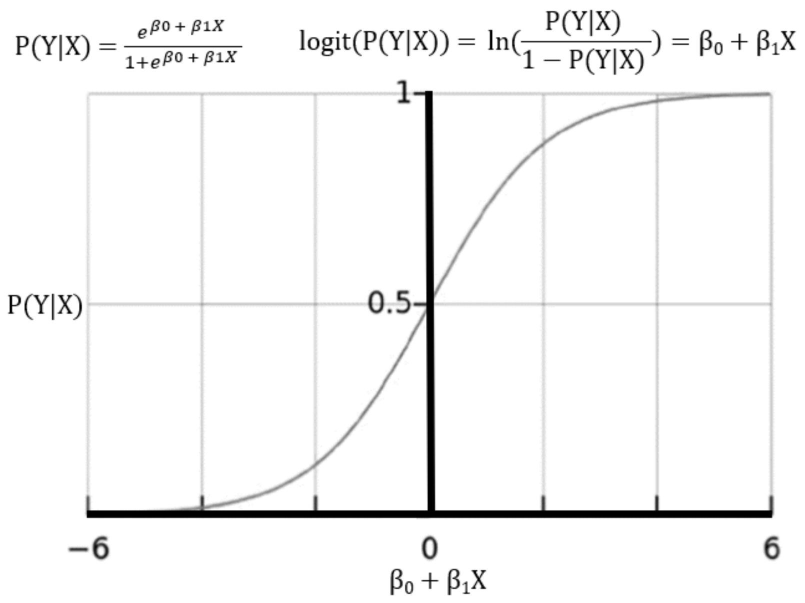


Figure 5: The equation on the top left expresses the logistic function and the one on the top right the one to obtain the coefficients of the logistic regression; the graph shows this relation where all the values of the P(Y|X) are comprised between 0 and 1. Figure and equations from (Awad and Khanna, 2015, pp. 22-23).

learn from a significant amount of data to create a model complex enough that can be useful for the problem, but not too much to not lose its ability to generalize. Because this type of model can be used on very complex problems, it is a very popular ML algorithm in areas such as biology and chemistry (Amato et al., 2013) (Awad and Khanna, 2015, pp. 129-133).

Hidden Markov Model

An Hidden Markov Model (HMM) model can be described as a hidden Markov chain with a given number of states, that can be characterized by a number of possible (hidden) states and by a number of observed states (or symbols). There are two types of transitions, each with its associated probability: state transition and observation emission (the latter describes each state), and finally the model can also be described by its initial state. This type of model has several advantages that makes it an interesting algorithm for certain applications, such as handling new data robustly and having a relative short model development time. Even though HMM are not usually classified as a supervised learning model, they can perform a similar purpose if they use sequential data. HMMs are extensively used in temporal pattern recognition, such as speech and calligraphy recognition, and DNA sequence analysis (Awad and Khanna, 2015, pp. 24-25) (Hassan and Nath, 2005).

2.4.4 *Ensemble Models*

It is common to ensemble different ML models to create functions capable of correctly predicting outputs with better accuracy. For that, an algorithm can be built with several types of ML models, on which a combination function is used so that the ensembled model can provide an overall output. To work properly, the ensemble model requires its individual models to be at least "weak learners" (better than random) and those models to differ between them in the examples they get wrong. One of the ways to achieve this is to train different models with distinct datasets through cross-validation, bagging or boosting, or to inject randomness into the algorithm by turning deterministic decisions into stochastic ones.

Combination functions can be implemented through model votes or confidence metrics (or some combination of both), in classification problems; in regression ones it is common to use simple or weighted average of the individual outputs (Awad and Khanna, 2015, pp. 3 & 13) (Dietterich, 2000) (Whitehead and Yaeger, 2010).

Random Forest

A Random Forest (RF) is a type of ensemble model that is composed by several decision trees to provide an overall prediction. In order to form this combination of trees, it needs to use bagging, which is a method that creates a set of training sets from the original one, by sampling in each iteration N examples with replacement (N being the original size of the training sample). In this type of model, each individual decision tree is built on a subset of the original features, being typically a tree with low complexity. Each tree makes its own output predictions, and since those individual trees are weak learners, these predictions tend to have high variance between them. However, by having several trees in the "forest", it allows the model to provide averaged outputs based on those individual models, thus becoming a strong learner, usually with better performance and less variance when compared with a single deep tree. RF can also help to determine the relative importance of each variable (Awad and Khanna, 2015, pp. 24) (Pal, 2005).

2.4.5 *Evaluating a machine learning model*

Useful metrics have been developed to determine the accuracy of the created model, i.e. the frequency of correct predictions over a dataset, and other relevant statistics. In binary classification models, a confusion matrix can be built (table 1), which is a table where the model's predictions produced are shown as true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN), when taking into account their output and actual result (Alpaydın, 2010, pp. 489) (Awad and Khanna, 2015, pp. 2-3).

Confusion Matrix		*Predicted Value*	
		Positive	*Negative*
Real Value	*Positive*	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 1: Confusion Matrix.

With those values, useful metrics can be computed to describe the performance of the model (in a classification model) (Alpaydın, 2010, pp. 490) (Awad and Khanna, 2015, pp. 3, 52-53) (Boughorbel et al., 2017):

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F - Measure = \frac{(\beta^2 + 1) * (Precision * Recall)}{(\beta^2) * Precision + Recall}, \beta \in]0, \infty[\tag{6}$$

$$MatthewsCorrelationCoefficient = \frac{TP * TN - FP * FN}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FP))}} \tag{7}$$

For regression models we need to define other metrics applicable to continuous variables, such as RMSE (root mean squared error) and MAPE (mean absolute percentage error), as shown in the following equations, being e the differences between the real and output values, Y the real value and n the number of observations (Cho, 2003):

$$RMSE = \sqrt[2]{\frac{\sum e^2}{n}} \tag{8}$$

$$MAPE = \frac{\sum |e| \, / Y}{n} \tag{9}$$

All these metrics help to select the most appropriate model for a given problem, and to compare the performance of different models. More sophisticated approaches have been developed to better determine the model's performance. One of the most well-known is k-fold cross-validation, in which the dataset is randomly split into mutually exclusive k sub-datasets of roughly the same dimensions; the model trains those k times, within each cycle using one of sub-datasets as its validation dataset, and the rest as its training, allowing to have k sets of values of performance for the model, in turn having a better perspective of its accuracy metrics. One special case within k-fold cross-validation is leave-one-out cross-validation, when k is equal to the size of the dataset, and it is especially useful for small samples of data.

Another important method is called bootstrap, which creates a bootstrap sample from n instances sampled (with substitution) from the original dataset, and that will become the model's train dataset, with the same size of initial dataset; the rest of the data will be used for testing the model (Alpaydın, 2010, pp. 486-489) (Kohavi, 1995).

2.4.6 *Feature Selection*

A frequent necessity is to select the features that generate better and simpler ML models, with shorter training cycles, in a process referred as feature selection, on which the most irrelevant and redundant features are removed. This procedure is usually performed by fixing a maximal number of features and determine how many and which combination of them leads to models that have the minimal generalization error.

For the first one there are 3 distinct methods: filter, wrapper and embedded. Filter methods rank each feature by a given metric (such as Pearson correlation or Fisher score), and then sort all attributes by that property, enabling to remove the less significant ones; this methodology is simple and computationally light.

Wrapper methods perform searches through the feature space, creating subsets of features, and analysing each subset by training a specific model to evaluate its predictive power (through an error metric as accuracy or f-measure), and then decide which subset leads to the model with the best performance. The output of this type of algorithm is highly related with the chosen learning algorithm, and it is also a more expensive method in computational terms.

Embedded methods combine feature selection and model fitting, creating a single optimization methodology; there are several algorithms within this class, each with their characteristics (Peralta and Soto, 2014) (Weston et al., 2001).

2.4.7 *Application of Machine Learning in Peptide Prediction*

Frameworks for Machine Learning Development Applied to Proteins

Several ML tools are being developed and published that help to augment our understanding of proteins. Those models are usually applied to predict a protein's subcellular location, structural characteristics and function sites, fold recognition, or protein–protein and protein–ligand interactions (Chen et al., 2019). Even though there are several frameworks available to help in such tasks, most of them either focus on just one (or small group of) step(s) necessary to the development of a ML model, or, if they are more advanced, they tend to lack some of the necessary tasks to build a model with good performance (such as feature selection functions) (Chen et al., 2019).

Two of the most promising frameworks within the category of web servers that allow the development of ML models applied for proteins are BioSeq-Analysis (Liu et al., 2019) and iLearn (Chen et al., 2019). Both contain functions for feature extraction and model construction and evaluation. iLearn is slightly more complete than BioSeq-Analysis, since it contains more feature extraction and selection functions, and it allows the training of more types of models, and finally, it can also use ensemble methods to combine models, but both can be very useful when building models for proteins (Chen et al., 2019).

Cell-Penetrating Peptides Prediction

One of the most promising areas where ML models are being applied is in cell-penetrating peptides prediction. These peptides (as its name implies) can insert themselves into cells, being interesting molecular tools in multiple biomedical applications (Manavalan et al., 2018). With this type of task in mind, Manavalan et al developed a web server called MLCPP, that contains a two-layer prediction framework: the first one predicts if a specific peptide can be classified as a cell penetrating peptide or not (build using an ERT model, which is a model similar to RF) with an MCC of 0.768; and the second layer predicts its uptake efficiency (a RF model with an MCC of 0.445) (Manavalan et al., 2018). Another example of such application is proposed by Kumar et al, that also launched a web server, called CellPPDMod, that uses a RF model with data from peptide's 2D and 3D structure, as well as fingerprints, and achieved accuracy of 95.1% and an MCC value of 0.90 (Kumar et al., 2018).

Antimicrobial Peptides Prediction

Another field that has gained a lot of attention in the past few years addresses the prediction of antimicrobial peptides. Examples of web servers with this purpose include AntiBP2 that uses peptide's amino acid composition (AAC) based features to train SVM models, which achieved an MCC value of 0.843 (Lata et al., 2010). Another example is iAMP-2L which uses a two-layer classifier: the first layer to detect if a peptide is antimicrobial or not, and the second to identify the functional type or types. It uses PAAC (pseudo amino acid composition) information as inputs to KNN models, achieving an MCC value of 0.726 (Xiao et al., 2013). Another tool called iAMPpred used SVM models that learned from compositional, physicochemical and structural features, achieving MCC values between 0.74 and 0.91 (depending of the test dataset and combination of used features, which included

AAC, PAAC, physicochemical and structural features) (Meher et al., 2017). AmPEP used also RF models trained with information based of distribution patterns of each peptide's amino acid properties along its sequence, with the best model achieving an MCC performance of 0.900 (Bhadra et al., 2018).

Anticancer Peptides Prediction

Several papers describe the development of anticancer peptides prediction models. For example, Anticp used SVM models which received as input AAC and binary profiles of SwissProt entries of anticancer and random peptides (or antimicrobial peptides, in their alternative dataset); the model trained with the binary profiles achieved MCC of 0.83 (Tyagi et al., 2013). Another method named MLACP trained SVM and RF models to classify anticancer peptides using features from the amino acid sequences, including AAC, dipeptide and atomic composition, and physicochemical properties, obtaining MCC value of 0.78 (Manavalan et al., 2017). A related predictor called ACPred-FL applied SVM models that used the features listed in MLACP, with the addition of a few more, like CTD (Composition, Transition, Distribution), and after feature selection, trained models achieved MCC performance of 0.813 and 0.814 in the different tested feature descriptors (Wei et al., 2018). Hajisharifi et al proposed SVM models that were trained with Chou's PAAC and local alignment kernel data; the former achieved a MCC value of 0.66 and the latter of 0.784 (Hajisharifi et al., 2014).

Machine Learning applied to Fusion Proteins / Fusion Peptides

Even with a lot of effort put in applying ML to proteins, there are not many examples of ML models specifically developed for VFPs. The ones that can be found are usually aimed to predict where the FPep locates within the VFP, given its crucial role in the fusion process. Moreover, they tend to focus on a single viral family, which is a much simpler problem than building a general model that can be applied to VFPs from different viral families.

One of the models developed for this task is proposed by Wu et al (2016), in which they developed HMM models with similarity comparison. Their models were trained with NCBI retroviruses' FPep entries (with as complete as possible annotations). This model predicts roughly the location of new FPeps, which through a similarity comparison is more accurately located. This last step determines the "refined" FPep position by maximizing the amount of matches, as well as the Smith-Waterman local alignment score, between the initial predicted FPep and an annotated FPep, minimizing also the absolute difference between the initial and ending positions of the model's predicted location and the "refined one". This overall procedure achieves an accuracy of 92.0% (10-fold cross-validation) and 92.3% (leave-one-out cross-validation) (Wu et al., 2016).

Wu et al, in 2019, suggested a different computational model to approach similar problems. They used the same data as mentioned on the paper above (also for retroviruses' FPeps), but also included entries from UniProtKB and UCSC genome browser databases (including annotations when available and removing entries deemed redundancies or with inaccurate annotations). For their training dataset, from each annotated FPep they retrieved 2 types of sample, one containing the start position of the peptide and the other containing its end position. All samples consist in arrays with a random size that will contain several amino acids before and after the true position of the start / end amino acid. They extracted features from each sample in the form of a vector containing normalized position weight matrix method values. From those features they trained SVM models capable of predicting each of those positions in a peptide sequence (if the model is not able to predict both of those positions for a particular sequence, the algorithm would consider that there is not a FPep domain on it). This methodology achieved accuracies of 91.2% and 89.1% in 10-fold cross-validation and 5-fold cross-validation respectively (Wu et al., 2019).

To address the same issue, Pereira et al started by building 3 different datasets from a total of 107 PDB and 640 UniProt entries from 255 different enveloped viruses, where FPeps were detected in 617 of those sequences. The first dataset allowed to detect whether the model could identify FPeps from other random sequences within the VFP; the second one to determine if it could distinguish between FPeps and transmembrane domains; and the third negative dataset contained both random VFP segments and transmembrane domains to access the models ability to discriminate FPeps from non-FPeps sequences. Using SVM it was performed a feature generation for each dataset. Features were generated by calculating different physicochemical descriptors using available packages and the best ones were selected using an iterative scheme for feature elimination based on linear SVM model. These features included single amino acid and dipeptide compositions, hydrophobicity, polarity, secondary structure, charge and solvent accessibility, and PAAC. After data pre-processing and feature selection (using support vector classifiers), several models were tested, with varying results, but the one with the most promising performance was a weighted ensemble of KNN (with a weight of 3), SVM (4), neural network (2), Logistic Regression (2) and Naïve-Bayes (3) models. This model achieved accuracies of 0.94, 0.99 and 0.81, and F1 Scores of 0.93, 0.98 and 0.83, for the first, second and third datasets, respectively (Pereira et al., 2019).

Sequeira et al built ProPythia, a general automated ML platform for protein classification, and applied it to the same problem described above (Sequeira et al., 2019). They took the 3 original datasets developed by Pereira et al and for each of those built several new datasets: 4 datasets resulting of feature selection of the generated continuous features (one using support vector classifiers, a second using a tree classifier and the other two were the result of applying univariate feature selection on the first two), one with AAC, another with Composition, Transition and Distribution (CTD) information, and the latter containing PAAC features. Several models were trained, including SVM, RF, stochastic gradient descent, Gradient Boosting, KNN, ANN and NB models. For the first dataset (of which the aim was to verify the model's capability of distinguish FPeps from other random sequences of VFPs) the best models were SVM's and ANN with the datasets created by feature selection (with MCC and accuracy values greater than 0.94); the second (to verify the model's capability to distinguish FPeps from transmembrane domains) achieved the best results using SVM, stochastic gradient descent and ANN, with the dataset built using support vector classifiers for feature selection, achieving performances near 1; the third (to distinguish FPeps from random sequences from VFPs and transmembrane domains) had naturally lower performance metrics (due to the more complex nature of the problem), but still achieved MCC values around 0.8 and accuracy of 0.9 when using SVM or stochastic gradient descent, with the support vector classifier dataset for feature selection (Sequeira et al., 2019).

Besides prediction of the location of FPep in a VFP sequence, ML models are also being applied to predict possible antiviral peptides for enveloped viruses, by targeting fusion processes (on which VFPs and their FPeps are fundamental, as referred before). One such work focused in this type of prediction is reported by Xu et al, that built SVM models using 5 physicochemical properties of peptides (isoelectric point, molecular weight, grand average of hydropathicity, solvent accessibility and secondary structure), as well as using a statistical scoring function based on the peptide sequence, from a library of 202 known enveloped virus entry inhibitors, as the model's inputs. The model with the best performance achieved an accuracy of 92.3% (5-fold cross validation), with a MCC value of 0.84 (Xu et al., 2015).

2.4.8 *Machine Learning Packages*

Scikit-Learn

Scikit-learn is a high-level efficient package that contains several easy to use ML methods (both supervised and unsupervised) in Python language, using some of its libraries like Numpy and Scipy (it also uses some C++ libraries). It has several methods that allow to perform the ML tasks described above, such as tools to preprocess data, to create models, to determine a model's performance and perform model selection (Pedregosa et al., 2011).

3 METHODS

3.1 RELEVANT BIOLOGICAL DATABASES

There are several protein databases available, that allow to retrieve a lot of information regarding proteins, such as their sequence, structure, function, families, post-translational modifications, subcellular localization, binding and interactions locations and energetics (Xu, 2004).

The following descriptions concern some of the most relevant and widely used protein databases: UniProt, NCBI Protein and Protein Data Bank. Alongside those protein databases, NCBI Taxonomy will also be referred, which holds data regarding the taxonomy of the viruses that contain the referred proteins, as well as information regarding those viruses' host organisms.

Finally, Immune Epitope Database will be also explained, detailing their importance regarding epitope information.

3.1.1 *UniProt*

Universal Protein Resource (UniProt) is a well-known open-source protein sequence and annotation database. It is comprised by 3 distinct databases: UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef), and the UniProt Archive (UniParc). It was a result of a joint venture of the European Bioinformatics Institute (EMBL-EBI), the Swiss Institute of Bioinformatics (SIB) and the Protein Information Resource (PIR). Each protein entry (depending on the level of curation) can have information regarding sequence, function, coding gene, structure, organism taxonomy, subcellular location, processing, expression, interaction and more (The UniProt Consortium, 2019).

3.1.2 *NCBI Protein and NCBI Taxonomy*

NCBI's (National Center for Biotechnology Information) mission is to gather new information technologies (like automated systems to store biological information, interfaces to allow the easy use of that data and ways to gather new information) to help understand fundamental molecular and genetic processes that control health and disease. It has several databases like NCBI Protein, whose entries usually contain information regarding sequence, coding genes and links to other databases to complete the information about a specific protein or peptide. Another pertinent database is the NCBI Taxonomy one, since it contains all the relevant taxonomy data regarding organisms, including viruses and their hosts (NCBI Resource Coordinators, 2016).

3.1.3 *PDB*

Protein Data Bank (PDB) aims to, according to their own moto, "enable open access to the accumulating knowledge of 3D structure, function, and evolution of biological macromolecules, expanding the frontiers of fundamental biology, biomedicine, and biotechnology". It mainly provides 3D structure data of proteins (or of parts of them), DNA and RNA molecules, but can also include other information depending of its curation (Burley et al., 2019).

3.1.4 *IEDB*

Immune Epitope Database or IEDB (http://www.iedb.org/) is a repository of experimental data of immune epitopes, funded by National Institute of Allergy and Infectious Diseases. Epitopes, according to their own documentation, are "molecular targets of adaptive immune responses", which are important for researching possible therapies for specific medical conditions. This database contains epitope's information gathered mostly from literature, all of which are manually curated. They perform searches of PubMed articles related to issues such as allergies, infectious diseases and autoimmunity every two weeks to always be up-to-date with the most recent publications. Besides providing this type of information, this repository makes available several tools to allow to perform prediction of epitopes within biological sequences (Vita et al., 2014) (Vita et al., 2018).

3.2 SEQUENCE ALIGNMENT TOOLS

A sequence alignment search allows scientists to find homologous biological sequences, which helps to determine the function of an unknown sequence comparing it against similar

Type of BLAST	Type of alignment	Programs
nucleotide blast	Nucleotide sequence vs. nucleotide sequence	*Megablast*: used in intra-species comparison; *Discontiguous megablast*: for cross-species comparison; *Blastn*: for cross-species comparison also, but for shorter queries.
protein blast	Protein sequence vs. Protein sequence	*Blastp*: sequence identification and similarity searches; *DELTA-BLAST*: similar with blastp but with higher sensitivity; *PSI-BLAST*: iterative search for PSSM construction or identification of relatives for a protein family; *PHI-BLAST*: protein alignment with input pattern as constraint
blastx	nucleotide translated sequence vs protein sequence	*blastx*
tblastn	protein sequence vs nucleotide translated sequence	*tblastn*
tblastx	nucleotide translated sequence vs nucleotide translated sequence	*tblastx*

Table 2: Description of the several types of BLAST analysis (NCBI, 2016)

ones with already known functions. There are several sequence alignment tools available, but one of the most popular is BLAST (Basic Local Alignment Search Tool). It uses heuristic algorithms to produce results quicker than exact methods such as dynamic programming. It also provides an estimation, called expect value, of the number of matches that could have occurred by chance, helping to evaluate the confidence given to the alignment produced. As referred by its name, it performs local alignments which is useful to detect similarities against domains/shorter stretches of sequences, allowing to detect some sub-sequences' functionalities (Madden, 2013). There are several types of BLAST, which are described in the table 2.

NCBI has a functionality called NCBI BLAST which is one of the most popular tools to perform BLAST analysis, that allows the user to choose between all the types mentioned on the table above, and to run the analysis against all available databases (a list that includes SwissProt, RefSeq and PDB), or run it against a specific database. Besides that, it enables to select other important parameters like scoring (match/mismatch and gap costs) or filter and masking parameters, which can affect heavily the output given by the BLAST analysis (NCBI, 2016).

Another one of the most well-known sequence alignment tools is Clustal, which the latest version of it is called Clustal Omega. This package allows to perform multiple sequence alignments between three or more biological sequences in a fast and accurate manner. It retains the core of the progressive sequence alignment methodology from its predecessors

(Clustal X and Clustal W), but upgrades them by using a mBed algorithm to allow to generate guide trees of any possible size, while also using a profile–profile aligner with a high degree of accuracy, which is based on HHalign package (Sievers et al., 2011) (Sievers and Higgins, 2014).

Other significant sequence alignment tools include HMMER, that is a software that through probabilistic methods (HMM models) can detect sequence homology, being widely applied to protein sequences (Finn et al., 2011). Weblogo is also a useful tool, that, given a multiple sequence alignment, generates graphical representations of sequence patterns, giving a description of the amino acid conservation for each position of the sequence. The taller a character (in terms of bits) is one specific position, the more conserved this amino acid is within the analysed sequences (Crooks et al., 2004).

BioPython

BioPython is an open source application programming interface (API) aimed for bioinformatic or computational biology programmers to work with sequences of nucleic acids and proteins. Through its class "Seq", it allows to work with biological sequences as basically Python strings, albeit defining its alphabet (p. e. DNA's "ACGT") and some other features. The "SeqIO" class provides an interface that allows to handle biological sequence data in several formats (such as fasta, genbank and embl), or even work with sequence alignments. BioPython allows to access data from several databases, such as NCBI, ExPASy and KEGG, and it also allows to perform Blast alignments calling the NCBI Blast server, along several other useful tools for a bioinformatician (Cock et al., 2009).

3.3 MYSQL

MySQL is a popular open-source management system that uses SQL ("Structured Query Language"), which is a well-known and widely used language to access and manage databases. With the MySQL server, one can create relational databases by defining their attributes in a relational schema, and then add and modify its data using SQL statements. This is a fast, user-friendly and reliable framework that can be run from a personal computer, but also supports web servers with multiple different back-ends, client programs and tools, and APIs (Oracle, 2021).

3.4 DJANGO

Django is an open-source Python web application (one example of those is web servers) development framework that allows developers to build their applications faster, since it contains several features that help the construction of software development projects, such as built-in authentication and content administration. It is easy to learn, due to its simple syntax and since Python tends to be a simpler language to learn when compared with other languages, Django is usually attractive for beginners. This is also a secure software, and with ample maintainability, scalability and versatility, and a popular framework to utilize when it is required to retrieve data from SQL databases (like MySQL and PostgreSQL) (DjangoSoftwareFoundation, 2019) (Plekhanova, 2009).

To build a web server application in Django that can access data from a database (or multiple ones) and export these data to a second application that will act as the front-end, one possible way is through the use of the Django REST API (DjangoRESTFramework, 2020a).

This framework is a useful tool to develop Web APIs, by allowing to expand the already described capabilities that the "default" Django contains of easily creating a web server by providing, among other features, pre-built views for the data to be displayed, additional authentication features that are non-present in the "base" Django, and also enabling serialization (DjangoRESTFramework, 2020a). This latter is of great interest, since it allows complex data (like database querysets) to be converted to Python datatypes, which themselves can be converted to JSON or XML data, both of them datatypes that other frameworks can easily use. Django REST also includes features to enable deserialization, a process that converts parsed data, that can be sent from other frameworks, back to more complex types that can be used inside the Django app (DjangoRESTFramework, 2020d).

After creating the Django project (which is done automatically using the "$django - admin$ $startproject$ '$project$ $name$'" command), several files are created:

- *manage.py* allows the programmer to interact with the Django project and it is the one that needs to be launched in order to run locally the project (most of the times it does not require to be modified after the automatic creation).

- *settings.py* contains all the definitions necessary (all the installed apps, all the required middleware, templates, ...) to run the apps contained within the Django webserver. To connect Django to a database (SQLite, MySQL, PostgreSQL, Oracle,. . .) the app developer must define in this file's parameter DATABASES the database's engine, name, user, password, host and port. Most of the installed apps built for Django REST require to have several of their definitions declared here also.

- *urls.py* contains all the URL declarations contained in the project (all the possible directories contained in the URL of the app).

- *wsgi.py*; WSGI is the Django's main deployment platform; this file will be mostly relevant when the developer puts the application online, since the app should be started from this file instead of the *manage.py* during that stage (DjangoSoftwareFoundation, 2020b) (DjangoSoftwareFoundation, 2020f).

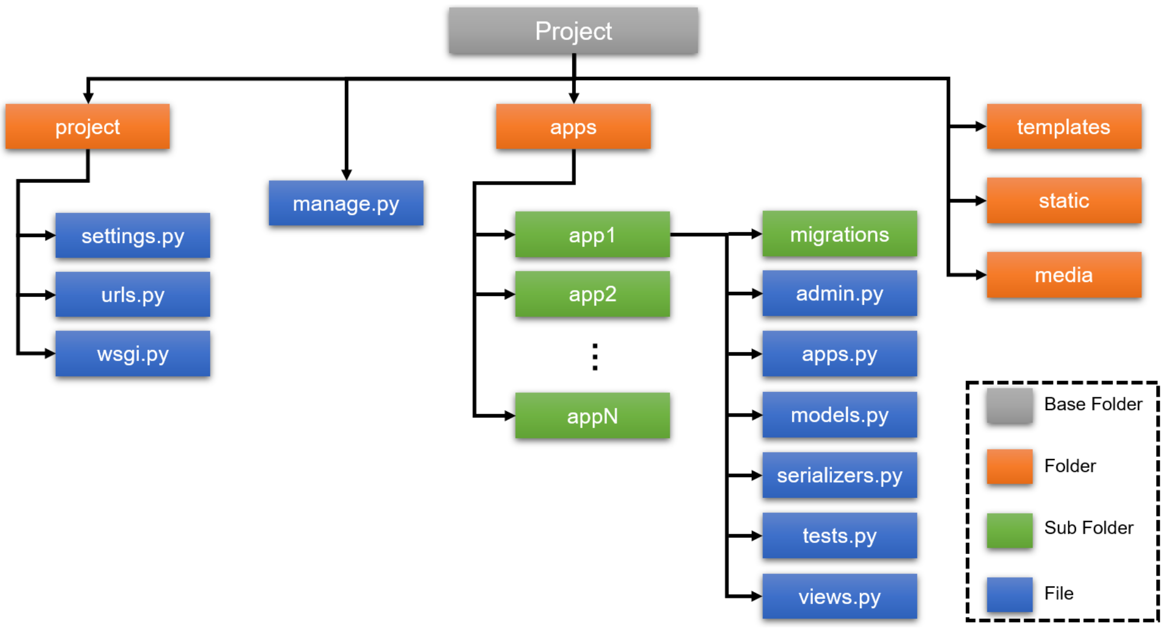


Figure 6: Structure of a Django REST project.

Next, a Django app can be created (through the python command *"python manage.py startapp 'app name'"*), with the necessity of adding the app name into *INSTALLED_APPS* parameter of *settings.py*. This process automatically generates the following files, on which most of the development will be done (the project's structure can be seen in the Figure 6):

- *admin.py* contains models and functions related to the administration of the application, including searching, creating, updating and deleting table data, as well as administering the levels of access each user can be entitled to.
- *apps.py* contains further applications contained in the "main" app.
- *models.py*; models are, according to Django documentation, "sources of truth about your data". This file contains the description of the database's tables (including all the table's fields key behaviors, like the data types of each attribute and each data table's foreign keys); this file can be built automatically using the *"inspectdb"* command, if the database is already built and if its settings are correctly described in the *settings.py*.
- *tests.py*, which contains functions that check the operation of the webserver's code; it can verify parts of the application or its overall functioning; basically, these functions will identify and prevent eventual problems on the code and its respective requests.
- *serializers.py*: contains all the functions that perform the serialization of the data from the models to the views (and vice-versa);

- *views.py*; a view consists in a function or a method that determine what is the response provided to a given request, which can or cannot be viewed in the browser whether or not there is a template associated with the view. Django will utilize a set view in function of the URL of the request, or, more accurately, the part of the URL after the site / domain name, that is requested (DjangoSoftwareFoundation, 2020c) (DjangoSoftwareFoundation, 2020a) (DjangoSoftwareFoundation, 2020d) (DjangoSoftwareFoundation, 2020e)

Django REST can also contain several other optional functions that will help filter and paginate the data shown in the browser. The following list describes some of those functionalities:

- *DjangoFilterBackend* allows to filter the shown JSON data from a particular serializer by exact matches against one or more of the object attributes. These attributes will be included in the URL of the browser, in the following format: *"urlname/-page?param1=(string)&param2=(string)&..."*. In order to use it, besides being declared in the *settings.py* (as stated above), for each view the developer must define all the possible attributes capable of restricting the data (within the *filterset_fields* command) (DjangoRESTFramework, 2020b).

- *SearchFilter* allows to perform single query parameter-based searching, and, unlike the former, it can detect partial matches. The parameters will be added into the URL, in a similar manner to the one shown in the Filter functionality. In order to incorporate this functionality into the application, for each of the views, the developer ought to define all the possible query attributes (within *search_fields* parameter) against which the search will be performed (DjangoRESTFramework, 2020b).

- *PageNumberPagination* allows to define how many objects show on a single page and create links for all possible subgroups of data. This could be done automatically by adding this package as paginator class and defining the size of the page on the *PAGE_SIZE* parameter in the *settings.py*, or one can build custom paginators as a class within the view file (DjangoRESTFramework, 2020c).

3.5 ANGULAR

Angular is a Typescript-written framework that allows the development of the single-page applications in Typescript and HTML for multiple devices. It enables to extend the capabilities of HTML vocabulary, by introducing ways to, among other features, create dynamic views, by creating data binding and updating the view whenever the model changes. It has also the capability of creating components that allow parts of the code to be reusable. This last one makes it also more efficient, since it only loads the necessary components for the page that is requested (Noring and Deeleman, 2017) (Super-powered by Google, 2018).

An Angular application has as its core building blocks the modules (in Angular also known as *NgModules*) that allow to compile components and retrieve related code to put it into functional sets. All Angular applications have at least one module, the root module, that allows to bootstrap the application, with the rest of the application's modules commonly referred as feature modules (Super-poweredbyGoogle, 2020a).

Components in Angular serve the same basic purpose than the view in Django: determining which sets of browser elements Angular can choose among and modify, according to the program logic and data. In essence, an Angular application is a tree of different components, each of them with its associated data, purpose and utility, and they are called whenever needed. An Angular component encompasses 3 types of data: HTML, CSS and JavaScript (or TypeScript, which is Javascript-based code). Components use services that contain functionalities that are not directly related to views but that can be injected into components as dependencies, allowing Angular components to be modular, which enables their code to be reusable. Services are the way parts of the Angular app are allowed to share data among themselves. This server data flow usually takes the form of a data stream, that permits to transform more easily the data that is being returned and modifying it to the desired format to be used (Super-poweredbyGoogle, 2020a) (Super-poweredbyGoogle, 2020b) (Super-poweredbyGoogle, 2020c).

The *HttpClient* method is the Angular method to fetch data from external servers and provide it to the application as a stream. It enables to perform several types of request to data sources (such as GET ones, that retrieve a given set of data; POST, that adds data into the data source, but can also be used to retrieve data which requires inputs that cannot fit the URL request; PUT, that allows to updating a given data entry and DELETE that remove an data entry) (Super-poweredbyGoogle, 2020d).

Components and services are classes which contain decorators (functions capable of modifying JavaScript classes) that define their type and associated metadata, that instruct Angular in the correct manner to use them (Super-poweredbyGoogle, 2020a).

This component's metadata essentially creates a relation between a component class and a given template. The latter ultimately defines a view, since it contains both HTML info, directives and binding markup, enabling the Angular application to change the HTML code before rendering it when the app requires that HTML info to show a view (Super-poweredbyGoogle, 2020b).

The service's metadata determines the information required to enable the availability of components by dependency injection (Super-poweredbyGoogle, 2020b).

An Angular component can define several different views, which are arranged hierarchically. In order to properly perform this type of task, Angular contains a *Router* service that determines the navigation paths among distinct views. This *Router* allows to display distinct components and data in function on which application's directory is being used at a given time, and also allows to navigate between different views when the user changes the URL in the address bar, clicks in a link of a page or selects the back and forward buttons of the web browser (Super-poweredbyGoogle, 2020e).

3.6 DOCKER

Docker is a platform that allows developers to build, run, and share applications, using a process called containerization. In essence, a container is just a running process with some encapsulation features needed to separate it from different containers and also from the host; this enables each container to interact with their own private filesystem, which in Docker is referred as an image. This image needs to contain everything required to properly run the intended application: code / binary, runtimes, dependencies, and all the other necessary files (DockerInc, 2020d).

In order to build docker containers, the developer should understand two types of file: Dockerfile and YAML. Regarding the former, Dockerfile is a text file that contains all the necessary command line commands required to the task of creating an image. Using this file, the user can use the *"docker build"* to automatically execute all the instructions contained in the file in order to form the image. Finally, it can run the container using the *"docker run"* command (DockerInc, 2020g).

If the aim is to build a multi-container Docker app, one can use an YAML file to define all the application services, and through the command *"docker-compose build"*, Docker can create all of those in a single process; in order to use it, the developer must define the app's environment through the use of a Dockerfile (or multiple ones); define the app's services in the *docker-compose.yml* so that they can be deployed into an isolated environment, and define all the relevant features to contain in those services; finally, the developer can build the project and start the app (or multiple apps) through the *"docker-compose up"* command (DockerInc, 2020a).

In order to share the project with other developers / users, the developer can add the project's containers into the Docker online repository (https://hub.docker.com/). In order to do so, it is necessary to commit a new image of the container using its container ID (visible through the command *"docker ps"*) through the docker commit command, and tag the image of the intended repository, so to finally push it, which creates an image in that repository that can be accessed by other users using the *"docker pull"* (if the repository is public). If the Dockerfiles are correctly defined, this manual process for creating images can be replaced by

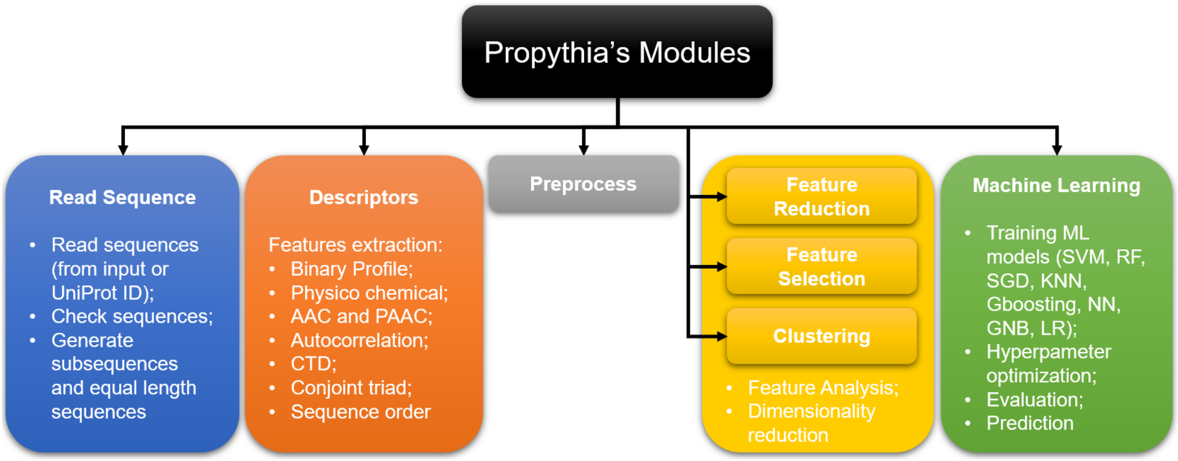


Figure 7: General description of Propythia's 7 modules (Sequeira et al., 2021)

the use of Automated Builds, which in essence connects the Docker hub repository to a code repository (like Github), and whenever the latter is updated, new images are formed using the latest changes (DockerInc, 2020f) (DockerInc, 2020e).

3.7 PROPYTHIA

This is a Python framework meant to allow developers and advanced users to write code to build ML models for classification of peptides / proteins from their physicochemical properties, developed by the host group of this work. The major aim of the package is to facilitate the development of the necessary pipelines to build ML models, by having functions to preprocess data, perform feature selection, run clustering algorithms, train and optimize ML models, as well as to deal with protein data and retrieve related descriptors from their sequences (Sequeira et al., 2019) (Sequeira et al., 2021). With that purpose in mind, this framework contains several functions lumped together into 7 distinct main modules, also illustrated in the Figure 7:

1. *sequence.py*, contains functions that allow to read or change sequences, and transform that information into objects; it contains functions that enable to retrieve sequences through UniProt ID's, to verify if a given sequence is a valid protein sequence and from a given sequence obtain all its possible subsequences, among other functions;

2. *descriptors.py* allows to obtain all the relevant descriptors that describe a given peptide. This list of features can be lumped together into several distinct groups, as defined by the authors: binary profiles (of AAC and of residues for 25 phychem feature); physicochemical features of the sequence (length, charge, charge density, number of C, H, N, O and S atoms within the amino acids, sum of the each possible type of bond

composition, molecular weight, gravy, aromacity, isolectric point, instability index, proportion of aminoacids that most likely are in helix, turn or sheet, molar extinction coefficient, flexibility, aliphatic index, Boman index and hydrophobic index); Aminoacid Compositions (AAC, Dipeptide and Tripeptide compositions), PAAC (Type I PAAC, Type I PAAC for a given property and Type II PAAC – Amphiphilic); auto correlations (normalized Moreau-Broto, Moran and Geary); CTD; Conjoint Triad, Sequence Order (sequence order coupling numbers and quasi sequence order) and base class peptide features (moment of sequence, global or window averaging descriptor, hydrophobicity / hydrophobic moment profile, arcs, autocorrelation and cross correlation of amino acid values for a certain descriptor scale);

3. *preprocess.py* contains all the typical functions to preprocess the data intended for the training of ML models, including functions to check for missing data and remove columns with only zero values, duplicate columns, or low variance of those values;

4. *feature_reduction.py* that, like its name implies, has functions meant to reduce the number of features present on the data, through both unsupervised methods, such as PCA, and supervised methods. Other complementary methods allow to graphically represent those results;

5. *feature_selection.py* contains functions to select the features that are more likely to produce better ML models, through supervised methods, like univariate feature selection, recursive feature elimination, and also through methods that allow to select features using specific models;

6. *clustering.py* allows to perform clustering analysis, containing functions for that purpose such as K-means and hierarchical clustering;

7. *machine_learning.py* contains functions that allow to train and test ML models; the function *train_best_model* allows to retrieve the best classifier from a list of models that include RF, KNN, SGD, LR, GNB, ANN and Gradient Boosting, each of those have their hyperparameters optimized by grid search cross validation; it contains methods to inform about the quality of a given model by retrieving performance measures such as accuracy, precision, recall, sensitivity, specificity, MCC and ROC when predicting a certain test dataset; this class has functions that allow to provide a prediction for a given peptide, and also to predict all possible subsequences within a protein through a sliding window (*predict_window* function); finally, it also contains graphical interfaces to represent the validation curve for a given classifier, through the plot of learning curves (Sequeira et al., 2019) (Sequeira et al., 2021).

4 DEVELOPMENT

4.1 APP'S STRUCTURE

The overall structure of this project's webserver can be seen in the Figure 8. There will be three different components, each with its associated Docker container:

- The MySQL database (to store all the data that the server requires).
- Django back-end (to access the database data and serialize it, besides containing several views that will be called by the front-end).
- Angular front-end (the application intended for the end user).

In the following sections, the development of the components of the project will be described, as well as its deployment into a server.

4.2 MYSQL DATABASE

4.2.1 *Description of the Database*

The database that stores the gathered data on viral fusion proteins and peptides was built based on the data gathered on a thesis by Pereira (Pereira et al., 2019). Its relational structure, names of the tables and attributes, as well of the type of data stored in each attribute can be seen on the following scheme (figure 9).

The "main" table of this database is called Protein and contains the information regarding each VFP's name, class, type of activation, fusogenic peptide description (its name, location within the protein and sequence), as well as having the UniProt and NCBI Protein entries for the protein. Another main table is the Fusion Peptide, which describes the location of this peptide within the VFP sequence (attribute "Residues"), sequence and some attributes regarding annotation method and experimental evidence. Both previous tables connect to the Structure table, which mainly has their respective PDB entries (when they exist), but can have information from other repositories.

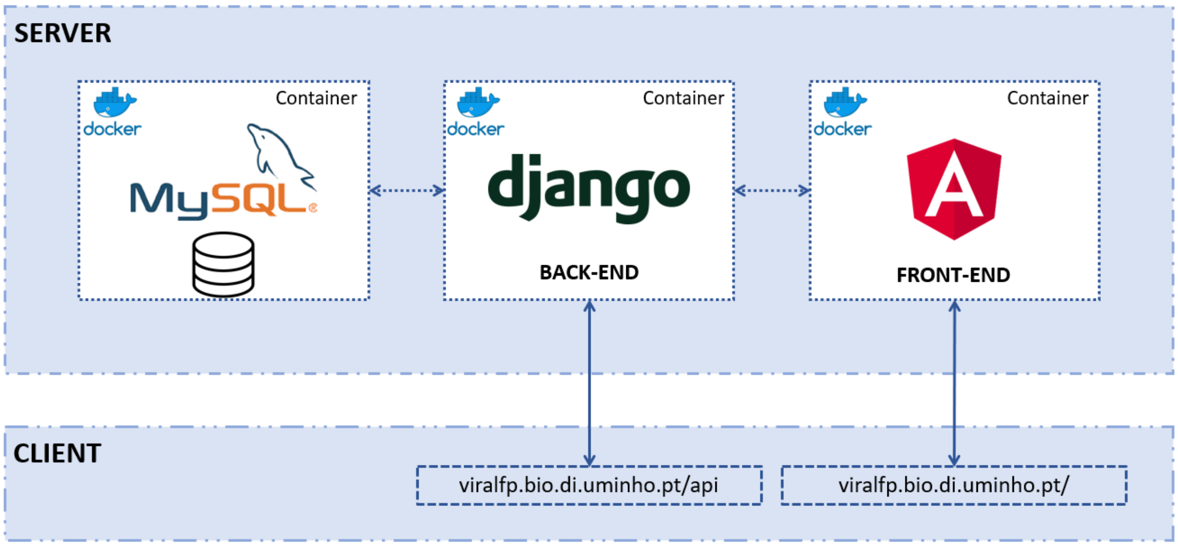


Figure 8: Structure of the app: within the server there are the 3 Docker containers: the MySQL, with the database dump; the Django (back-end) and the Angular (front-end). The latter two can be accessed from a client through those browser links, even though it is the latter that is intended as the one the client would use.

The last key table is the Taxonomy, which has all the relevant taxonomy of the protein's virus (family, genus, species and subspecies / strain), as well as having the virus' NCBI Taxonomy entry. This table connects to the Host's table, which has the common name and NCBI Taxonomy of each virus' host organism.

A complementary table present on this database is the References table, which stores the relevant bibliography regarding either each protein, FPep or structure, mainly in the form of DOI links, but also can include PMC entries or even article's references.

Finally, the inhibitor / antibody table was added, that contains info regarding those types of protein that are associated with a particular VFP. Besides that, some attributes names of the "original" tables were changed, in order to make it more clearer, as well a NCBI Taxonomy entry that was included into the Host table; finally, a primary key in the table that links the Virus and Host's tables was added. This last one was introduced to prevent some errors when performing queries.

4.2.2 *Curation of data*

The initial filling of the database used as its starting point a database built by Pereira (Pereira et al., 2019) in Excel, which contained an agglomerate of information of UniProt, NCBI Protein and PDB entries, and additional protein information retrieved from some scientific articles. This filling used the Python package mysql.connector (OracleCorporation, 2020) to automatically pick the CSV data and put it into the database.

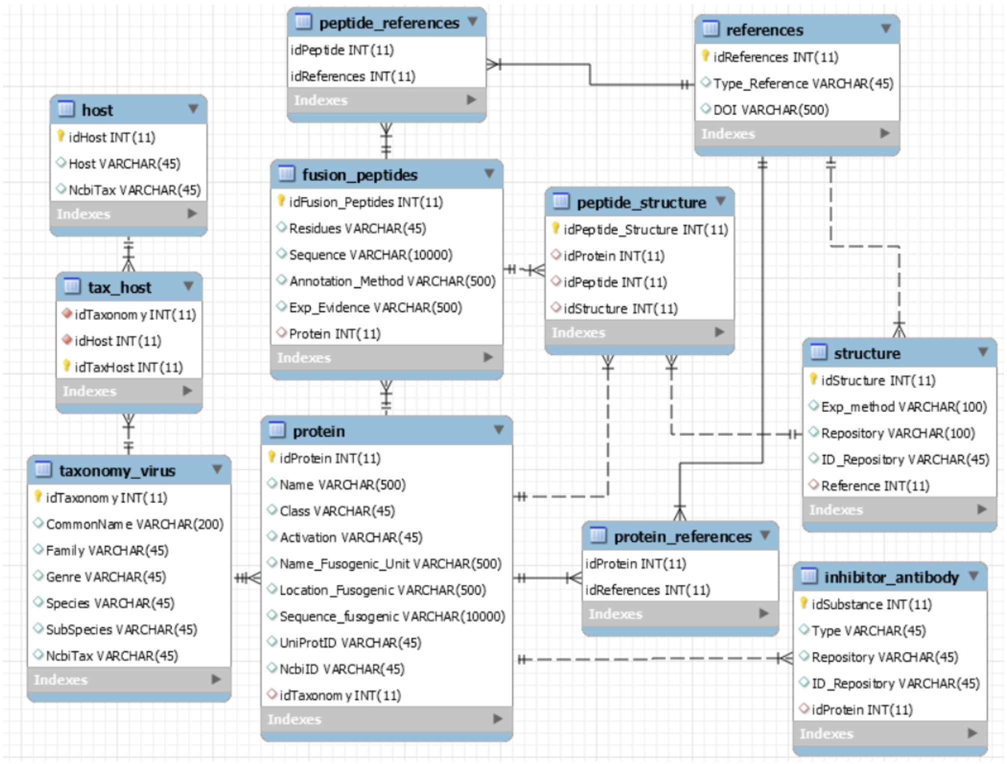


Figure 9: Structure of the database

In the filling of Taxonomy Virus table, BioPython (Cock et al., 2009) was also used to search taxonomy parameters like each virus' genus, that were not present in the original database. However, one can predict that some of the data present in the original CSV database might be outdated or could even contain some errors caused by the automatic filling if those databases entries missed some parameter or were in a non-expected format.

So, in order to limit as most as possible this issue, a manual curation was performed. In the Taxonomy Virus table, composed with data almost exclusively from NCBI Taxonomy, the most common correction was on the Species parameter since the initial automatic filling sometimes assumed the common name parameter to be the same as the species one, which often is not true. Another focal point of the curation process was to make sure if the family and genus parameters were in fact correct, because a lot of times they were filled with the sub-family or sub-genus, respectively.

The Protein table is mostly composed by UniProt's entries, but also from a sizeable number of NCBI Protein entries. This curation was more difficult because it required to interpret the description in the UniProt entry, whilst the NCBI Protein entries lack some of the key aspects of the table, such as the class or activation method. The table was corrected as best as possible with the available info from those databases, but the entries with unclear parameters were mostly left the be filled in a later time.

The FPeps data is mostly found on the same data entries as each respective protein, so its curation was done in parallel with the Protein one. In this table, the most common fix was to fill some parameters not included in the initial database. The curation of the remaining tables was mostly to verify if the references introduced in the original database were also all present in the final one, and also to verify if each virus had the expected possible hosts in the table relating the virus' taxonomy with the host one.

This curation process also involved the removal of the information associated with eukaryotic organisms present in the original Excel database, besides removing all the blank lines or data entries with no meaningful data, from the database.

A new table was created which contains some of most relevant publications and PDB entries of inhibitors and antibodies associated with each VFP. This filling was performed using Biopython for the Pubmed searches, and pypdb (Gilpin, 2015) for the PDB searches. The search was in the form as "Species" + "Strain / SubSpecies" + "Protein Name" + "Inhibitor / Antibody".

4.3 DJANGO APPLICATION

This component can be accessed through the URL https://viralfp.bio.di.uminho.pt/api.

The main purpose of this back-end is to access the data of the MySQL database, a task for which Django contains several functionalities available. This framework will serialize the query data into a JSON format, that can easily be used by client apps, which in this case will be the Angular front-end, so that it can be displayed or saved. Besides that, the back-end will allow a user of the front-end to add or modify data, through deserialization methods.

Since bioinformatics packages are more commonly written in Python that in the other programming language used in this project (Typescript), those tools are inserted here, that later will be called by the front-end. These features include Clustal and Weblogo tools, connections to IEDB, ML prediction of the FPep within the VFP sequence and conservation scores.

Table 3 contains a brief review of the most relevant views of the back-end. In the next sections, a more detailed explanation of the all the parts of this app will be performed.

4.3.1 *Table's Views*

There is one view for each MySQL database table, with the same name of the respective table. When considered relevant, some data was included in the serializers from other tables accessible through foreign keys. In the case of the fusion protein's table, some of the virus' taxonomy data was included; in the fusion peptides' table, the protein's name was added; and, finally, in the tables that link data from two or more different tables, like the one that

URL	*Purpose*
(table name)	Display 10 entries of that table data in a JSON format.
(table name)/save	Retrieve all results of a given query.
clustal_all	Retrieve Clustal Omega alignment of protein sequences.
conserv	Retrieve conservation features from a protein.
iedb	Retrieve the likelihood of subpeptides within a protein sequence being a epitope.
ml_predict	Retrieve the likelihood of subpeptides within a protein sequence being a fusion peptide.
weblogobackend	Build a weblogo from protein sequences, using the package Logomaker (Tareen and Kinney, 2019).
weblogoclustal	Clustal Omega alignment of sequences for later produce weblogos.

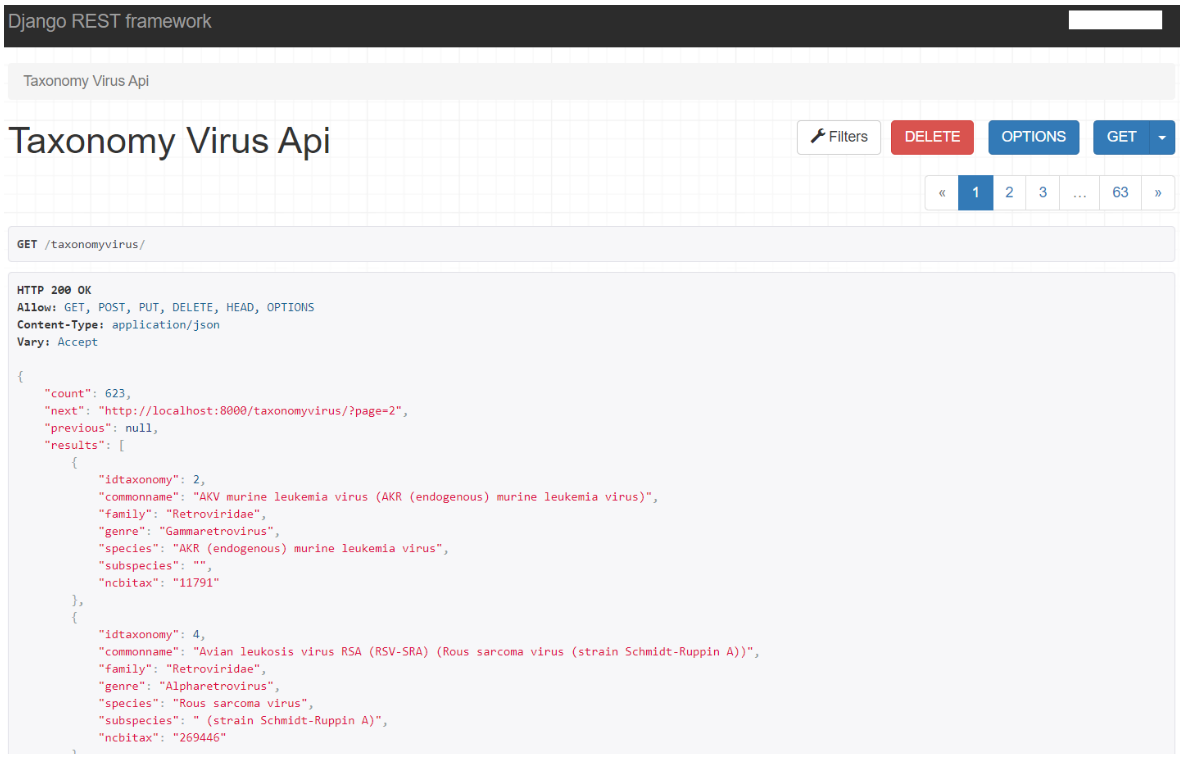
Table 3: Relevant domains of the back-end.

connects the references table with the protein and fusion peptide ones, or the table that links the virus' taxonomy table with the hosts' one, data from both tables were introduced so that the view can contain all the important information and not just the foreign keys present on them.

All data pages within the app will show something similar to the figure 10. The data will be displayed in JSON format. The JSON object displayed in this page contains the 10 first table data entries (plus any attributes related to the foreign keys that were added on the serializers) on the "results" parameter. This object also contains a "count" parameter, that displays the total number of entries on this table, the "next" parameter that shows the link to the view that provide the following 10 data entries, and the "previous" parameter which provides the link for the previous 10 ones.

This page also contains the pagination buttons (implemented with the *PageNumberPagination* package), that serve a similar purpose as the "next" and "previous" parameters. It also has a GET button, which allows to switch this API view to its corresponding raw JSON one. The OPTIONS button shows the headers of the GET request that allowed to display the page and the description of each attribute of the data (type of data, if a given variable is required or not, if it is read only or not, its label and size / range parameters). The DELETE button is necessary to later accept DELETE requests, but within this page, this button is not functional since to perform both DELETE and PUT requests, the URL request requires the presence of the primary key of the data entry to be updated / deleted.

The FILTERS button implements the filtering and searching methods described in the *DjangoFilterBackend* and *SearchFilter*. These filtering / search possible parameters are usually all the attributes present in each table with the notable exception of all foreign keys IDs in the case of searching, because search against foreign keys would generate errors, since in the Django app those foreign keys are essentially objects containing the corresponding data entry and so cannot be filtered against a string. Finally, this page accepts both POST and



Taxonomy Virus Api

```
GET /taxonomyvirus/
```

```
HTTP 200 OK
Allow: GET, POST, PUT, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 623,
    "next": "http://localhost:8000/taxonomyvirus/?page=2",
    "previous": null,
    "results": [
        {
            "idtaxonomy": 2,
            "commonname": "AKV murine leukemia virus (AKR (endogenous) murine leukemia virus)",
            "family": "Retroviridae",
            "genre": "Gammaretrovirus",
            "species": "AKR (endogenous) murine leukemia virus",
            "subspecies": "",
            "ncbitax": "11791"
        },
        {
            "idtaxonomy": 4,
            "commonname": "Avian leukosis virus RSA (RSV-SRA) (Rous sarcoma virus (strain Schmidt-Ruppin A))",
            "family": "Retroviridae",
            "genre": "Alpharetrovirus",
            "species": "Rous sarcoma virus",
            "subspecies": " (strain Schmidt-Ruppin A)",
            "ncbitax": "269446"
```

Figure 10: Appearance of the browser page of the Django application.

PUT requests, both in this page; those type of requests can be done in this page also, either through a HTML form or using raw data (in JSON) (only the POST form works within the page, as already mentioned above).

In order to add a new entry to the database (through POST requests), for each class of the serializers.py a function "create" was added that, upon receiving the JSON data from the request, will create the object in the database. This was necessary because, as mentioned before, attributes from tables related to the foreign keys were added in most of the views in order to complement the information of each entry. However, because the back-end expects these extra attributes in the request, the Django app will not allow to create new objects of the "main" table with only the "base" attributes of the table in the POST request. This added function can circumvent this issue by telling Django to create the object required with certain attributes, which obviously in this case are the table attributes that are included in the request. Also, some MySQL query code was included that ensured that the new object would get the smallest primary key available.

To perform PUT and DELETE requests, a "put" and a "delete" function were required to be added for each view. Both retrieve the request data and the primary key in the URL (this is the reason why both buttons do not work in each view's page) and update/ delete, respectively, the intended data entry.

4.3.2 *Tools' Views*

Besides those views that allow to explore the database data, several others views were added, mainly to later be called by the front-end. For that reason, most of them do not contain any HTML information, meaning their content will not show up in the browser. Most of these views are meant to be used in the Tools component of the Angular app.

IEDB

The *'iedb'* view will allow to send a POST request to IEDB (through the API http://tools-cluster-interface.iedb.org/tools_api/bcell/), to retrieve the epitope's prediction information regarding a given sequence (Vita et al., 2014). This view also requires defining a method to perform the analysis (the methods available within the IEDB API are Bepipred, Bepipred-2.0, Chou-Fasman, Emini, Kolaskar-Tongaonkar and Parker) and a window size for that sequence. The results of this API are returned in a JSON format.

Weblogo

The *'weblogoclustal'*, as its name implies, performs the Clustal Omega alignment (through a Clustal Omega console) necessary to align the sequence to later perform the Weblogo analysis. After that, the function reads the aligned sequences in the FASTA file written by the console and returns the data as a JSON object. To fully build weblogos in the back-end (in the view *'weblogobackend'*), the most obvious solution was to use the official Python API provided, through the package *'weblogo'* (https://weblogo.readthedocs.io/). This mostly functions properly for text output, and so, if the TXT output is requested, the view sends the result of that package. However, this was not possible for the PNG output, due to issues of that package.

Thus, to circumvent this issue, a package called *'logomaker'* (Tareen and Kinney, 2019) was used. This package allows to expand the *matplotlib* package (a well-known Python package to build graphs) so that one can build a sequence logo from a matrix of weights. So, to build a similar weblogo to the ones produced in that API, the weblogo matrix (obtained with the *'weblogo'* package given a set of protein sequences) was required to be split into several sub-matrices, one for each line of the weblogo. In order to use this package, special attention was needed to assure that the same y-scale was used throughout all weblogo, as well as to make sure that the last line does not change in size relative to the rest of the lines (because the package will correct the size of the columns to fill all the line), something that was achieved by adding columns of zeros in the last sub-matrix. After the graph is written into a file, the same file is read as a base64-string. This string will be the output of this view if the PNG output is requested.

Clustal

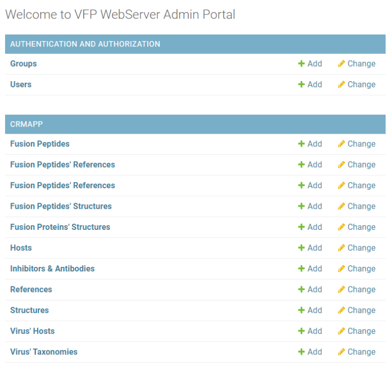
Clustal Omega analysis was performed using the same console as mentioned for the *'weblogoclustal'* view. This view, called *'clustal_all'*, will receive the sequences and the additional parameters as part of the POST request, perform the Clustal alignment, write the alignment and guide tree files, and, finally, send the alignment file content, as well as the alignment's guide tree file's content as plain text.

Machine Learning

The *'ml_predict'* view allows to perform the FPep prediction for a given sequence, using models trained using Propythia (Sequeira et al., 2019) (Sequeira et al., 2021), from a dataset kindly provided by Sequeira. This file was obtained by extracting the features available in Propythia from a dataset where the non-positive results are composed with both transmembrane domains sequences and non-FPep and non-transmembrane domains. After that, those features went through a feature reduction process using an SVC model. According to Sequeira's work, the models that are trained with data that went through this specific feature reduction method tend to have better performance. Finally, ML models were trained with the Propythia's *train_best_model* function. Those models are stored in Pickle files. If a given model is selected, the file corresponding to that model will be loaded using the *'pickle'* package and used in an instance of the Propythia's ML class (which uses the already mentioned dataset). This view also requires receiving a window size and size of the gap. Using those parameters, the *'predicted_window'* function of the Propythia is called to perform the predict the likelihood of the subpeptides of that window size to be FPeps. Those results will be returned in a JSON format.

Conservation Features

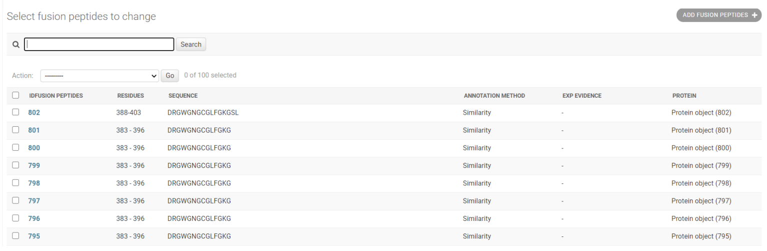
A view (*'conserv'*) was added to obtain the conservation scores of a peptide against a virus family's weblogo (the procedure to obtain these scores is fully detailed in the section "Conservation Features" of this chapter). This view receives a POST request which must contain a protein sequence, a window size and a taxonomy family, and for each possible peptide with that window size, the function obtains the conservation value for the selected alignment. For each position of the sequence, the view obtains the maximal value for all the conservation features from the peptides that contain it. A JSON containing those values is returned.



Welcome to VFP WebServer Admin Portal

AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
Users	Add	Change

CRMAPP		
Fusion Peptides	Add	Change
Fusion Peptides' References	Add	Change
Fusion Peptides' References	Add	Change
Fusion Peptides' Structures	Add	Change
Fusion Proteins' Structures	Add	Change
Hosts	Add	Change
Inhibitors & Antibodies	Add	Change
References	Add	Change
Structures	Add	Change
Virus' Hosts	Add	Change
Virus' Taxonomies	Add	Change



Select fusion peptides to change

0 of 100 selected

IDFUSION PEPTIDES	RESIDUES	SEQUENCE	ANNOTATION METHOD	EXP EVIDENCE	PROTEIN
802	388-403	DRGWGNGCGLFGKGSL	Similarity	-	Protein object (802)
801	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (801)
800	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (800)
799	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (799)
798	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (798)
797	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (797)
796	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (796)
795	383 - 396	DRGWGNGCGLFGKG	Similarity	-	Protein object (795)

(a) Application Administrator Page. (b) Example of a Django Admin Table Page (in this case the Fusion Peptides one).

Figure 11: Django Administrator Component.

4.3.3 *Saving Data Features*

There are several added views which aim to produce query results to store into CSV files. For all the pages present in the front-end, a duplicate view was included, with only the pagination definitions removed, which means that all the data for a given search is displayed. There are other views that receive POST requests (sent by the front-end) with the JSON data to save into files. There is a very similar one that does the same sort of task for the ML prediction results, but this one saves the results as TXT file.

4.3.4 *Authentication and Administrator's Features*

An authentication and registration package called $'django - rest - auth'$ was added to the application. This package contains a set of REST API endpoints to perform tasks such as login, logout, password reset or change and registration. These endpoints will enable client apps (in this case the Angular front-end) to communicate to the back-end independently through REST APIs. In order to later match the Angular authentication methods already present, which uses e-mail as mandatory (instead of the username which is the default in this package), the *settings.py* file needed to contain all the necessary declarations so to turn the username optional whilst making email mandatory (through the use of a package called $'django - allauth'$). This file was also modified so that the payload would return in a JWT format, which is the most common format for this type of data when its intend is to be shared between multiple parties (TivixInc, 2018a) (TivixInc, 2018b) (JWT, 2020).

Alongside those pages, this application also includes an Administrator Page (figure 11a), that contains functions to manage the info and access of users, as well as manage the webserver data (it allows to perform view the data and also to perform POST, PUT and

DELETE requests). Obviously, the access of each function is restricted by the type of user. For each of the tables to appear in the Admin page, the statement *"@admin.register(NAME OF DATABASE)"* was inserted within the *admin.py* file, as well as defining which fields needed to appear in the page, and finally which were the possible search fields. Ultimately, this provides a user-friendly way to explore and modify the data as the user wishes (Figure 11b).

4.3.5 *Connection to Other Frameworks*

To allow the access of this data by other frameworks in Django, django-cors-headers was added to the application. This is a Django App that adds Cross-Origin Resource Sharing (CORS) headers to HTTP responses. It allows the Django application to accept in-browser requests from other origins, enabling the application's resources to be accessed on other domains. Note that one important concern related to this package is to guarantee the future security of the application, because using CORS can unintentionally open up the web server's private data to others (Yiu and Johnson, 2019).

4.4 ANGULAR APPLICATION

This component can be accessed through the link https://viralfp.bio.di.uminho.pt/.

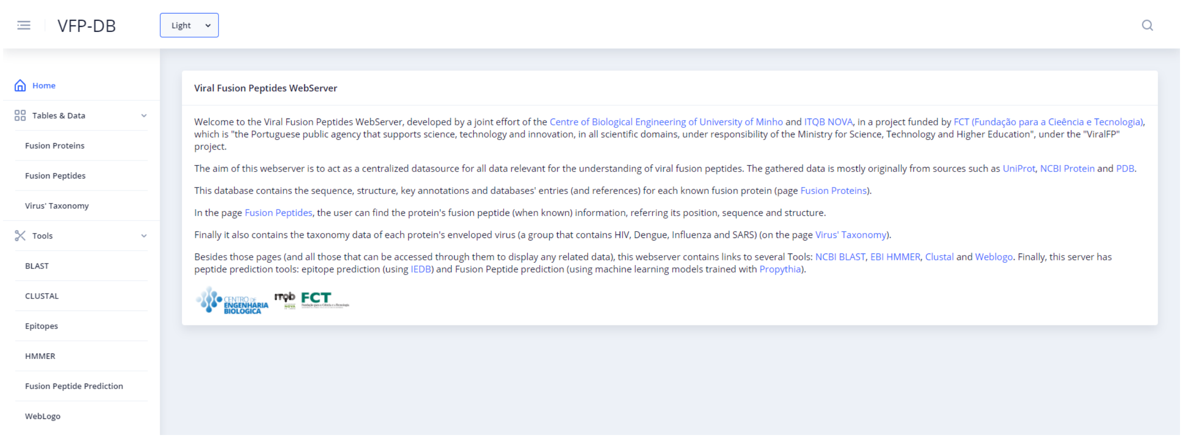
This application is aimed to be the interface where the end user can access all the data present in MySQL database in a easy-to-use interface. The app also contains links to perform BLAST and HMMER analysis, as well as having Clustal and Weblogo features, and, finally, tools to predict epitopes and FPep from VFP sequences.

In order for the app to function as it is supposed to, the end user must allow the browser to open popup pages, because several of the pages described below will require that permission.

This front-end was built using a framework called Nebular (https://akveo.github.io/nebular/, which is an open-source Angular User Interface Library that consists in a "pre-made" modular, customizable and configurable application. This enables to reduce the development time by already having several features like authentication and security methods, as well as having tables and card formats that are user-friendly and visually appealing.

4.4.1 *Front Page*

The front page contains a description of the overall purpose of the application, with a small text of the information that can be found and the authors of the application. Within this text there are relevant links for the development of this project (figure 12). This page also contains a side menu with all the functionalities present on the application, and a global



Viral Fusion Peptides WebServer

Welcome to the Viral Fusion Peptides WebServer, developed by a joint effort of the Centre of Biological Engineering of University of Minho and ITQB NOVA, in a project funded by FCT (Fundação para a Ciêência e Tecnologia), which is "the Portuguese public agency that supports science, technology and innovation, in all scientific domains, under responsibility of the Ministry for Science, Technology and Higher Education", under the "ViralFP" project.

The aim of this webserver is to act as a centralized datasource for all data relevant for the understanding of viral fusion peptides. The gathered data is mostly originally from sources such as UniProt, NCBI Protein and PDB.

This database contains the sequence, structure, key annotations and databases' entries (and references) for each known fusion protein (page Fusion Proteins).

In the page Fusion Peptides, the user can find the protein's fusion peptide (when known) information, referring its position, sequence and structure.

Finally it also contains the taxonomy data of each protein's enveloped virus (a group that contains HIV, Dengue, Influenza and SARS) (on the page Virus' Taxonomy).

Besides those pages (and all those that can be accessed through them to display any related data), this webserver contains links to several Tools: NCBI BLAST, EBI HMMER, Clustal and Weblogo. Finally, this server has peptide prediction tools: epitope prediction (using IEDB) and Fusion Peptide prediction (using machine learning models trained with Propythia).

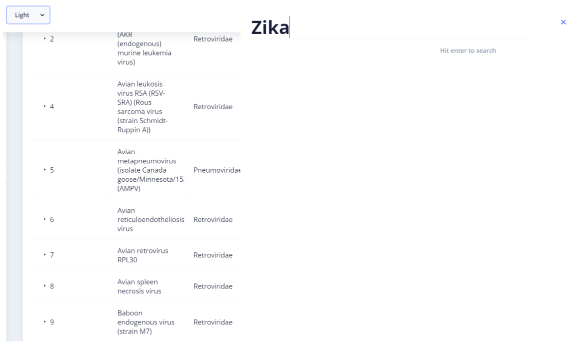
Figure 12: Application's frontpage.

search method in the page's header, that opens a small window with the pages that contain matches against the search term. This search is limited to the 3 most important pages in the application, which are the fusion proteins, the fusion peptides and the taxonomy tables, as well as providing some specific results within each of those three pages, like shown in Figs. 13a and 13b, which provides links for new windows with the results.

4.4.2 *Tables' Pages*

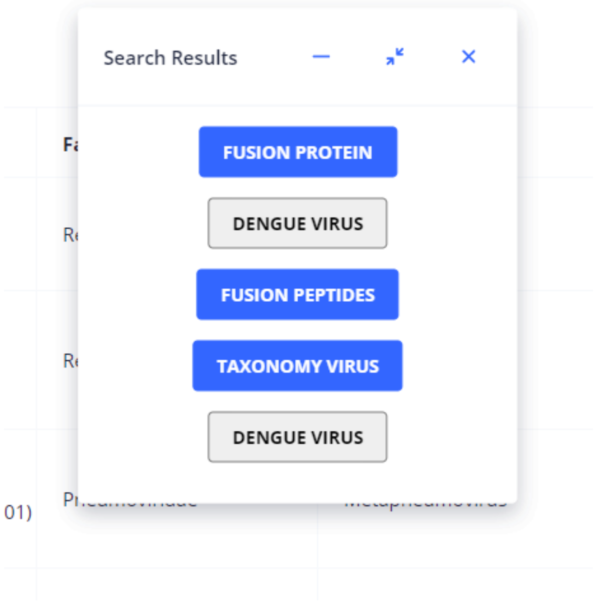
The information of the database can be found on the "Tables & Data" option of the side menu. This contains links for the Taxonomy, Fusion Protein and Fusion Peptides tables. Those tables and all the others that can be accessed through them are in a Tree Grid format (figure 14), which is a Nebular feature: each data entry will correspond to a table line; upon clicking on one line, its "children data", which in this case corresponds to buttons that connect to that line's related data, will appear below. This related data is: in the case of the Taxonomy one, the tables of their hosts, fusion proteins and fusion peptides; in the case of the Fusion Protein one, their virus' taxonomy, fusion peptides, structure, references and known inhibitors / antibodies; and, finally, in the case of the Fusion Peptide one, their fusion protein and references. These children information pages will appear in different tabs upon clicking the respective button.

All the pages contain search methods that use the Search and Filtering tools described in the Django app, as well as having pagination methods, also using the corresponding methods in the Django back-end. All these data are retrieved using services (mentioned above), where each page has its own dedicated file. Every time a search is made, the page is changed, or a ”children data” is selected, new requests are made to the services, which themselves make requests to the views from the back-end, that allows to update the table's



2	(AKR (endogenous) murine leukemia virus)	Retroviridae
4	Avian leukosis virus RSA (RSV-SRA) (Rous sarcoma virus (strain Schmidt-Ruppin A))	Retroviridae
5	Avian metapneumovirus (isolate Canada goose/Minnesota/15 (AMPV)	Pneumoviridae
6	Avian reticuloendotheliosis virus	Retroviridae
7	Avian retrovirus RPL30	Retroviridae
8	Avian spleen necrosis virus	Retroviridae
9	Baboon endogenous virus (strain M7)	Retroviridae

(a) Search Bar in the Angular App.



(b) Results from the search Bar in the Angular App.

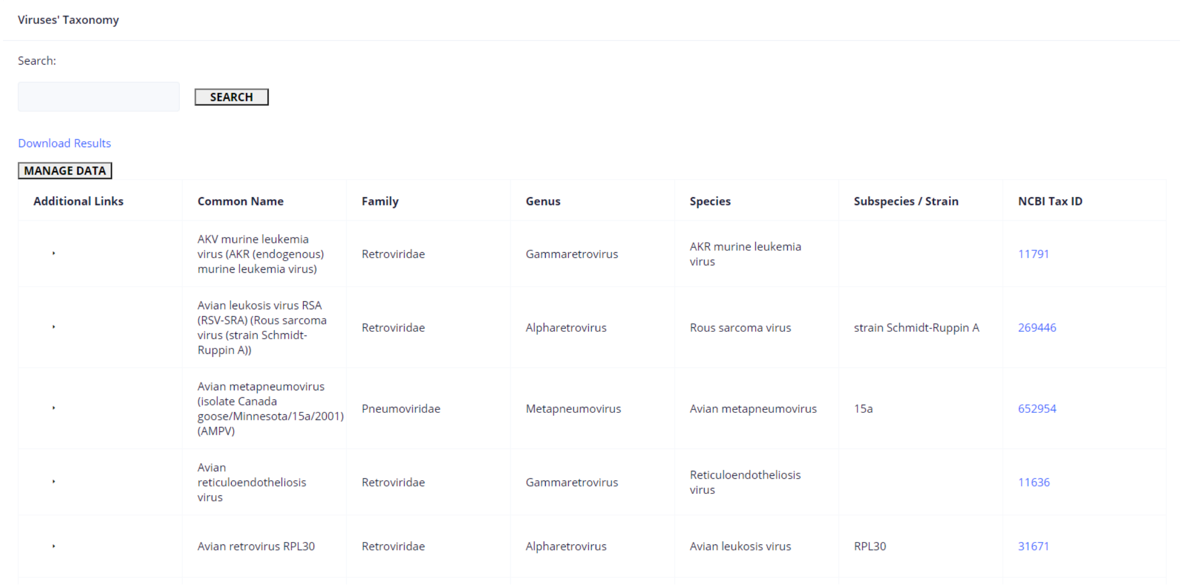
Figure 13: Search display in the front-end app.

data. The 3 main pages also contain suggestions to autocomplete search terms to match terms presents in those tables.

All these data table pages contain forms that allow to perform both POST and PUT requests to the back-end in order to add or update an entry, respectively. At the time of writing, since there is no successful authentication features inserted in the application, in order to prevent unauthenticated users to add or update data, the HTML code on each of the table pages corresponding to those features was commented, but not eliminated, so that in the future these functionalities can be added in the app.

These table pages have features that save the query results into CSV files (in the form of the ”Download Results” link). This procedure uses the views described in Django to retrieve the full data of a given query, but the actual saving component was made within the front-end, by transforming the JSON data returned by each respective subscribe into a CSV-like string. Through the use of the *”DomSanitizer”* class of the *’@angular/platform-browser’* package, the app can in essence create an URL to download the result. Note that it is important to only perform this request to the server after completely loading the page. Otherwise, the output of the file comes often as ”undefined”, since loading the data is one of the last tasks performed while loading a new page.

All table pages contain a “MANAGE DATA” button, that redirects into a new tab of the back-end admin page, that allows to observe, add, update or delete data entries. This was


Viruses' Taxonomy

Search:

Additional Links	Common Name	Family	Genus	Species	Subspecies / Strain	NCBI Tax ID
•	AKV murine leukemia virus (AKR (endogenous) murine leukemia virus)	Retroviridae	Gammaretrovirus	AKR murine leukemia virus		11791
•	Avian leukosis virus RSA (RSV-SRA) (Rous sarcoma virus (strain Schmidt-Ruppin A))	Retroviridae	Alpharetrovirus	Rous sarcoma virus	strain Schmidt-Ruppin A	269446
•	Avian metapneumovirus (isolate Canada goose/Minnesota/15a/2001) (AMPV)	Pneumoviridae	Metapneumovirus	Avian metapneumovirus	15a	652954
•	Avian reticuloendotheliosis virus	Retroviridae	Gammaretrovirus	Reticuloendotheliosis virus		11636
•	Avian retrovirus RPL30	Retroviridae	Alpharetrovirus	Avian leukosis virus	RPL30	31671

Figure 14: Virus' Taxonomy Table.

not as first envisioned, since the original plan intended to have all the major functionalities in the front-end, but due to the difficulties of implementing authentication features, and since the Django admin page contains all the data management tasks, it was decided to include it in the front-end. This page is restricted to allowed users, since it requires authentication.

Finally, in order to select specific protein sequences to be used in the Tools present in the app, in the Fusion Protein page there are checkboxes that allow to select one or multiple sequences, and one button for each of the tools that can be found within the page.

4.4.3 *Tools' Pages*

As mentioned above, the app also contains other tools, lumped together in the menu as "Tools". These pages can receive sequences inputs from the Fusion Protein page as an URL parameter. Most of them receive the full sequences encoded into the URL. However, the Clustal and Weblogo pages only receive the sequence IDs separated by commas. This was a necessary fix, since both are meant for multiple alignments, meaning they will require several sequences to be sent at the same time, so there is a higher risk of requests sent to them surpassing the URL string size limit, invalidating the request. On those two pages, for each of the IDs, a request is made to the back-end to retrieve the desired sequences, so they can be used for the subsequent requests to be performed within the pages.

These Tools pages can be classified into one of two categories: sequence alignment and sequence prediction.

Sequence Alignment' Pages

In the Sequence Alignment tools, a page that links a selected sequence to the NCBI BLAST was added; this page contains methods that allow to preselect also the type of Blast to perform (the options are Quick BLASTP, BLASTP, PSI-BLAST, PHI-BLAST and DELTA-BLAST) and select the database to run the BLAST against, where the included options are Non-redundant protein sequences, Reference proteins (RefSeq Protein), Model Organisms (landmark), UniProtKB/Swiss-Prot, Patented protein sequences (pataa), PDB and Transcriptome Shotgun Assembly proteins (Johnson et al., 2008).

In a very similar manner, the HMMER app allows to select a sequence and the database against which the search is made (in here the options are Reference Proteomes, UniProtKB, SwissProt and PDB), and then send this info into the EMBL-EBI portal, showing the results in this portal (Potter et al., 2018).

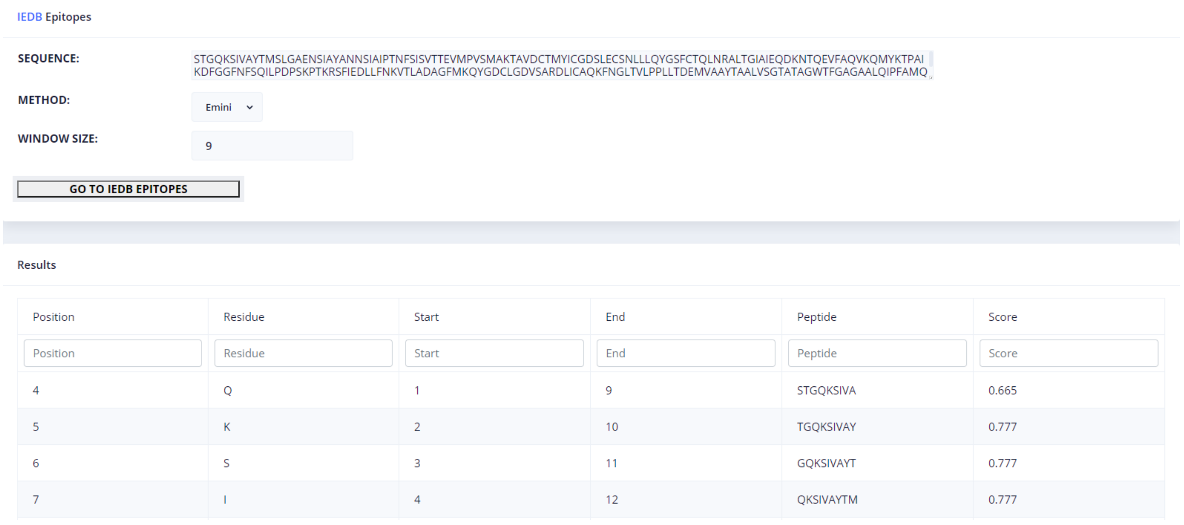
WEBLOGO contains two possible ways to perform the analysis. The first uses the weblogo.threeplusone.com portal (Crooks et al., 2004), on which the app requires at least three sequences (in a FASTA format), and from those it builds a weblogo (it can be in a PNG or raw data formats) using the rest of the standard values to perform the analysis. to use this functionality, the front-end sends the sequences to the Django back-end, that through a Clustal Omega console, performs the multiple alignment. The second method will use only the back-end; the page contains two different buttons for the purpose: one to display in the same page the PNG output of the weblogo obtained; the second one allows to open a pop-up page with the text result of the same process.

CLUSTAL also contains two possible methods. The first uses the https://www.genome.jp/ portal to perform a CLUSTAL W analysis; the app asks for the type of output of the analysis (CLUSTALW, FASTA or PHYLIP), the type of pairwise alignment (FAST/APPROXIMATE or SLOW/ACCURATE) and the weight matrix (BLOSUM or PAM) (Li, 2003). The second performs a Clustal Omega alignment from the back-end; upon clicking the button, one pop-up will open in the browser to display the alignment in the required format; a second one can appear if the the guide tree data is also requested. The original plan intended to use the CLUSTAL Omega from EBI, but since this web page does not accept query parameters and the API Python file provided by them to connect generates a package import error (from the package urllib) when it is used inside Django, that option was not used.

Sequence Prediction' Pages

There are two tools relative to sequence prediction: epitope prediction and FPep prediction (through the use of ML models).

Regarding the first, it contains access to the IEDB portal (Vita et al., 2014) in order to perform antibody epitope prediction; the front-end sends a request with all the already mentioned parameters to the back-end; the latter sends a second request to the IEDB's API


IEDB Epitopes

SEQUENCE: STGQKSIVAYTMSLGAENSIAYANNSIAIPTNFSISVTTEVMPVSMAKTAVDCTMYICGDSLECSNLLLQYGSFCTQLNRALTGIAIEQDKNTQEVFAQVKQMYKTPAI KDFGGFNFSQILPDPSKPTKRSFIEDLLFNKVTLADAGFMKQYGDCLGDVSARDLICAQKFNGLTVLPPLLTDEMVAAYTAALVSGTATAGWTFGAGAALQIPFAMQ

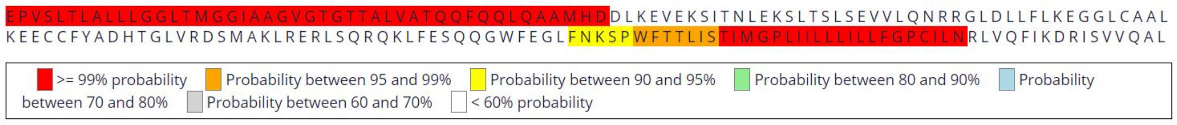
METHOD: Emini

WINDOW SIZE: 9

Results

Position	Residue	Start	End	Peptide	Score
Position	Residue	Start	End	Peptide	Score
4	Q	1	9	STGQKSIVA	0.665
5	K	2	10	TGQKSIVAY	0.777
6	S	3	11	GQKSIVAYT	0.777
7	I	4	12	QKSIVAYTM	0.777

Figure 15: Example of epitope prediction, in the Smart Table format.


E P V S L T L A L L L G G L T M G G I A A G V G T G T T A L V A T Q Q F Q Q L Q A A M H D D L K E V E K S I T N L E K S L T S L S E V V L Q N R R G L D L L F L K E G G L C A A L
K E E C C F Y A D H T G L V R D S M A K L R E R L S Q R Q K L F E S Q Q G W F E G L F N K S P W F T T L I S T I M G P L I I L L L I L L F G P C I L N R L V Q F I K D R I S V V Q A L

🟥 >= 99% probability 🟧 Probability between 95 and 99% 🟨 Probability between 90 and 95% 🟩 Probability between 80 and 90% 🟦 Probability between 70 and 80% ⬜ Probability between 60 and 70% ☐ < 60% probability

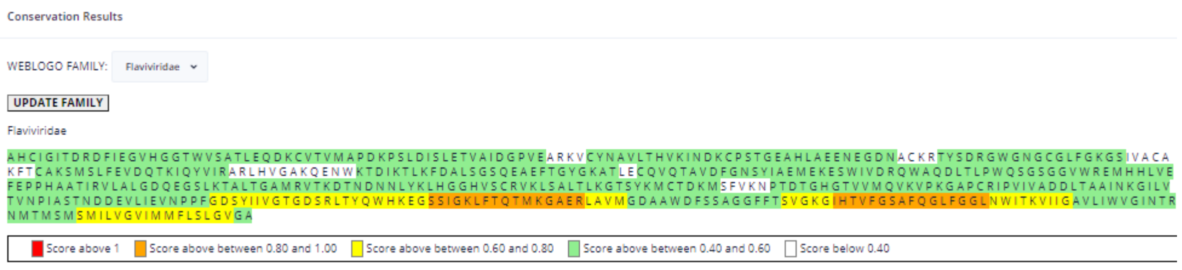
Figure 16: Machine Learning prediction of the fusion peptide location.

URL for the tool (as stated above), and sends back the results to the front-end, showing them in a Smart Table format, which is also a Nebular feature (Figure 15); both Bepipred methods provide 2 different result tables, the rest of them provide just one (Vita et al., 2015).

Regarding ML prediction of the location of FPeps within the VFP sequence, models trained by Propythia were used (Sequeira et al., 2019) as already described in the previous section. This page contains a text box where users can insert a protein sequence or select one entry from the Fusion Proteins pages. The user can also specify which model to use, from the set of available models: SVM, RF, Gradient Boosting, KNN, Linear Regression, Gaussian NB and ANN models. Also, the user can select the window and gaps sizes.

The page provides two types of output: a table one which contains every possible peptide likelihood of being a FPep; and a graphical interface that, for each amino acid displays a color that represents the likelihood of being a FPep (in this case the max probability of all peptides that contain that particular character), with the respective color legend (like shown in Figure 16); also, for each amino acid, a popup box shows its position within the VFP sequence and the probability value of belong to a FPep. Finally, the page has a save functionality that stores the results into a text file into the user desktop.

In the FPep prediction page, a tool was also included to obtain the already mentioned conservation features. This will perform in a very similar manner as the one described for the graphical representation for the FPep prediction. Upon receiving the JSON with the


Conservation Results

Flaviviridae

```
AHCIGITDRDFIEGVHGGTWVSATLEQDKCVTVMAPDKPSLDISLETVAIDGPVEARKVCYNAVLTHVKINDKCPSTGEAHLAEENEGDNACKRTYSDRGWGNGCGLFGKGSIVACA
KFTCAKSMSLFEVDQTKIQYVIRARLHVGAKQENWKTDIKTLKFDALSGSQEAEFTGYGKATLECQVQTAVDFGNSYIAEMEKESWIVDRQWAQDLTLPWQSGSGGVWREMHHLVE
FEPPHAATIRVLALGDQEGSLKTALTGAMRVTKDTNDNNLYKLHGGHVSCRVKLSALTLKGTSYKMCTDKMSFVKNPTDTGHGTVVMQVKVPKGAPCRIPVIVADDLTAAINKGILV
TVNPIASTNDDEVLIEVNPPFGDSYIIVGTGDSRLTYQWHKEGSSIGKLFTQTMKGAERLAVMGDAAWDFSSAGGFFTSVGKGIHTVFGSAFQGLFGGLNWITKVIIGAVLIWVGINTR
NMTMSMSMILVGVIMMFLSLGVGA
```

🟥 Score above 1 🟧 Score above between 0.80 and 1.00 🟨 Score above between 0.60 and 0.80 🟩 Score above between 0.40 and 0.60 ⬜ Score below 0.40

Figure 17: Example of the results of the conservation tool.

output (similar to the ML prediction, as seen in the Figure 17), for each sequence's position, a score is provided and from that value the colour corresponding to that score is attributed to it. The family can be selected from a list, which upon clicking the button, updates the schematic to the desired family.

4.5 CREATING THE DOCKER REPOSITORY

In order to create the repository necessary to use the available server services, so that this app can be used online, 3 images in a repository were created: one for the MySQL database, one for the Django back-end and one for the Angular front-end.

The first two were originally created using a docker-compose. This was useful since, as stated before, in order for the Django app work, the MySQL's one also needs to do so. To function properly, both of the images must be inserted into the same Docker Network. Besides the docker-compose, 2 Dockerfiles were also defined.

The one related with the MySQL component retrieves the official Docker's MySQL image, defines all the relevant MySQL variables (user names, passwords and the name of the intended DB) and adds the Dump of the MySQL database (this is a file that contains all the commands required to create all the DB tables and its stored data) described above into the entrypoint.

The Django's Dockerfile retrieves the official Python image, copies all the project files into the container, installs all the necessary Python packages needed within the back-end (all these packages are defined in a text file) and, finally, installs the Clustal Omega.

Within the docker-compose, in both services, the exposed ports are defined, which Dockerfile to build and the network associated. In the MySQL service, the entrypoint command *'–default-authentication-plugin=mysql_native_password'* was required, so that the Django service can access the data using the password defined in the docker-compose. The Django service required the command *python manage.py runserver* to run the server and to define the volumes (all the necessary data to be shared between local storage and the container). Finally, the DATABASES parameters of *settings.py* were required to be updated so that it could respond to the MySQL service (DockerInc, 2020b) (DockerInc, 2020c).

Docker-compose files were only used for the original images; for the ones built using automated builds (which are the ones kept updated), only the Dockerfiles were used. The Django one needed to be updated to utilize the Gunicorn command. This will start the server using the wsgi file instead of the *manage.py* one, which is more appropriate for the production stage (deploying it into a server).

For the Angular image one tutorial that can be found online was followed. This one used only a DockerFile that retrieved the Node base image, installed the browser Chrome for protractor tests, copied all the necessary data from the project, installed Angular and all the relevant packages in the container and added the command *"ng build"* required to run the server. After building the image and running it, defining the apps volumes and ports, the container could be run at the selected port (Herman, 2019).

Before the project could be inserted into the repository, some directories within the Angular application required to be fixed since initially the project included static URLs. They were replaced with URLs generated with *RouterLink*, which enables the app to create the correct links independently of the domain name. Originally, only the Django's save query features were used, but they did not work within the container. So they had to be replaced by the Angular functions. Also, the Clustal alignment tool, like already referred, needed to use the console installed in the container, which replaced the one present in the code. However, the views that used this console required an additional command to make sure a response was not sent before the completion of the multiple alignment analysis, a problem that did not occur when the code was run locally (Super-poweredbyGoogle, 2020f).

After all that, the public repository was built. In order to run it locally, one must use the command docker pull for each of the three images, create a network and run the three containers, with the caveat that both MySQL and Django must be in the same Docker network and for each container must define the port (where MySQL is expected to be 3307:3306, Django 8000:8000 and Angular 4201:4200). Finally, the MySQL one must be called *"django_db_1"*, in order to match the settings of the Django container.

Alongside that, connections to a Github repository were added in the Docker Hub repository. This will allow to perform automated builds of all the required images, and update them whenever the project's code is updated.

One can access this Docker repository at https://hub.docker.com/r/pedrodmmoreira/vfp_webserver, which contains all the instructions that allow to install all the components of the app.

The live version of the webpage can be visited at https://viralfp.bio.di.uminho.pt/. This link will display the front-end. When first deployed there was a major issue that was not expected: the app did not seem to work whenever the request was made to a sub-directory directly. This means that whenever a request was made to a URL which was not the base URL (the one indicated above) into the browser search method, or one refreshed the browser, the page would show up a "404 Not Found" error. The problem was that the routing within the Angular app was not correctly deployed from the production. One could use the *HashLocationStrategy* (instead the default *PathLocationStrategy*) to try to circumvent the issue. However, this seemed to affect some of the router procedures, mainly when a button leads to open a new tab, and so this strategy was removed (Super-poweredbyGoogle, 2020e). The used alternative was updating the Nginx definitions within the server that is running the site in order to try to find the required files that determine the routing within the app, as well as to tell the server how to handle the most common types of request errors.

The back-end can be visited at https://viralfp.bio.di.uminho.pt/api/. This one is mostly working according to the expected behaviour, except for its aesthetic look, since all the static files are not loading (even though static directories were add to the URLs of the app), meaning that the pages only shows up the raw HTML data, and so, it is not pleasing-looking for the end user. This is troubling just because the Admin page was meant to be useful for the administrator user, but it does not affect the end user, since all the tools work as intended.

4.6 CONSERVATION FEATURES

Through the Propythia framework (Sequeira et al., 2019) (Sequeira et al., 2021), models can be trained that are capable of predicting the probability of a given peptide to belong to the FPep of its VFP. The Propythia framework already extracts a lot of features that describe a given peptide (listed above). However, there is not a feature regarding the conservation of that peptide. Since FPeps tend to be a well conserved region of the VFP, specially between proteins of viruses that belong to the same taxonomic family, features specific for that regard were created.

VFP sequences present in the database were used to create weblogos. After retrieving all the sequences and the family of each respective virus from the database built for this project, only the families that contained at least four sequences in the database ended up selected. This allowed to select the following 13 families: *Arenaviridae*, that have 4 sequences in the database; *Coronaviridae* (16 sequences); *Filoviridae* (16); *Flaviviridae* (71); *Herpesviridae* (22); *Orthomyxoviridae* (170); *Paramyxoviridae* (55); *Peribunyaviridae* (4); *Phenuiviridae* (4); *Pneumoviridae* (9); *Retroviridae* (164); *Rhabdoviridae* (6) and *Togaviridae* (32).

For each of those families, a Clustal Omega (Sievers et al., 2011) alignment, required to create the weblogos, was performed. The weblogos themselves were created using the '*weblogo*' package, which is the Python API format for the http://weblogo.threeplusone.com portal. However, it was detected that performing the same weblogo analysis to the same alignment with apparently the same parameters would lead to different entropy values when compared to the results obtained from the web portal. This can be explained if the web portal performs some type of post-analysis correction, but by the time of writing of this thesis, the reasoning of this discrepancy was not fully understood.

One conservation feature was created for each of the families. For each of the weblogos, the “final” scores for each character of each position of the matrix were precalculated. This was achieved using the conservation equation described by Crooks (Crooks et al., 2004), in which this metric is obtained by the following expression:

$$R_{seq} = S_{max} - S_{obs} = log_2 N - (-\sum_{n=1}^{N} p_n * log2(p_n)) \tag{10}$$

where R_{seq} is the conservation of a given position; S_{max} is the maximal entropy possible in a position, equal to $log_2(N)$, N being the total number of possible distinct symbols, meaning that in a protein sequence this value is 20 (the number of possible amino acids); and S_{obs} is the observed entropy in a given position. Because these weblogos are result of alignments of families with different number of sequences, that number of sequences was taken into the calculations.

So, before analyzing the sequences, the weblogo matrix was preprocessed by calculating in each position the result of the following equation:

$$PositionScore = \frac{(log_2(20) - S_{obs}) * bits}{n_{seqs}} \tag{11}$$

where the *Sobs* is the value of the column ”Entropy” on that specific row, *bits* the number of bits on each position of the matrix and n_{seqs} the number of the sequences in the family.

After processing those calculations, there is one matrix saved in a Pandas dataframe with only the scores per position. So, after transforming the Pandas dataframe to Numpy arrays, it is possible to retrieve the local alignment scores for each sequence (or subsequence) by verifying the scores of the sequence against all the possible positions in the matrix and save the best of those values as a feature.

These functions were introduced into a file within the Django project that contains the content of the *descriptors.py* of the Propythia package, to allow retrieving these conservation features, so that they can be used in the page of ML prediction of the location of the FPep.

5

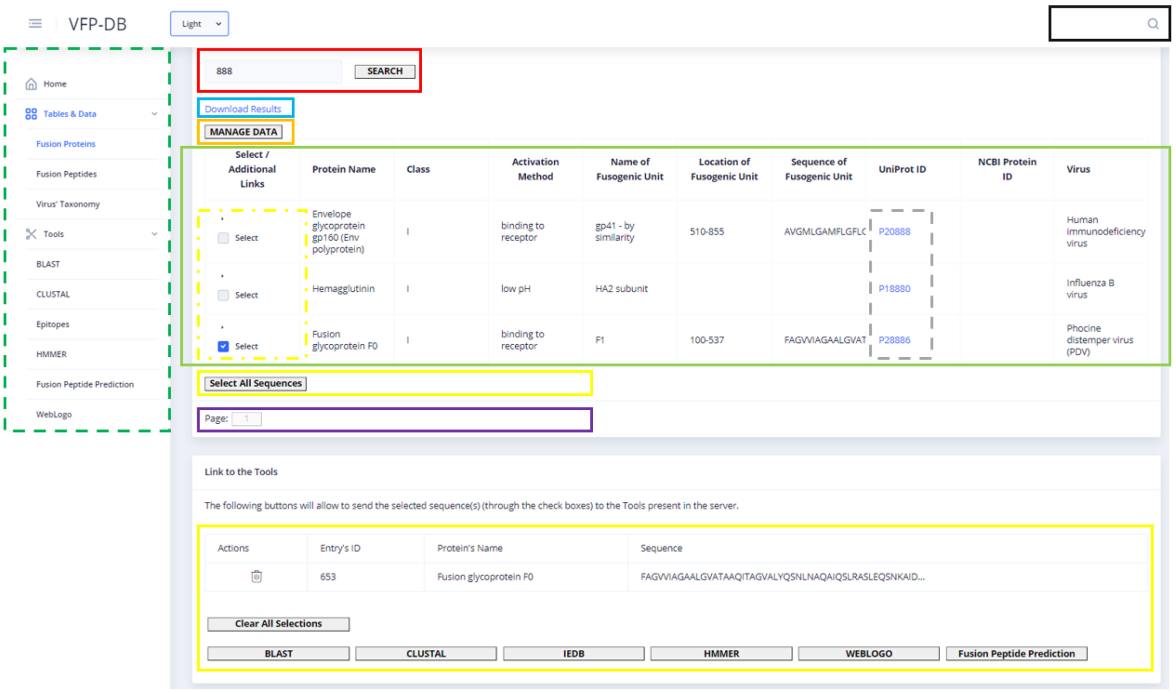
RESULTS AND DISCUSSION

5.1 BRIEF MANUAL OF THE APPLICATION

There are two major components of the front-end application (the one intended for the end users) that require an explanation: the tables and the tools.

The figure 18 shows the representation of the Fusion Protein page and will allow to explain all the tables pages' different elements. The following list describes them, from the left to the right and top to down:

1. On the dashed green rectangle on the left, this is the menu of the application, where one can access the main data tables, as well as access all the tools present in the server.
2. On the top right, in the black box, the user can find the link to the global search tool described in the Figs. 13a and 13b images in the development chapter.
3. The red rectangle contains the search methods also previously described, that will (partially) determine what is shown in the table.
4. The blue rectangle contains the link to download the query result as a CSV file.
5. The orange rectangle is the link to the back-end's administrator page.
6. The green rectangle is the table resulting from the query determined by the search performed and the current page; within each line, there are links to related data (if available).
7. The yellow rectangles contains all the tools that allow to use the selected sequences (chosen in the checkboxes on the yellow dashed rectangle, or through the "Select all Sequences button"); upon choosing at least one sequence the bottom yellow rectangle will appear, on which the user can eliminate some (in the table) or all (in the "Clear All Sequences") of the selected sequences and redirect the chosen sequences to the Tools; the links in the menu will go to the same tools, but no sequences will be sent; those links are exclusive from the Fusion Protein page.



Select / Additional Links	Protein Name	Class	Activation Method	Name of Fusogenic Unit	Location of Fusogenic Unit	Sequence of Fusogenic Unit	UniProt ID	NCBI Protein ID	Virus
☐ Select	Envelope glycoprotein gp160 (Env polyprotein)	I	binding to receptor	gp41 - by similarity	510-855	AVGMLGAMFLGFLC	P20888		Human immunodeficiency virus
☐ Select	Hemagglutinin	I	low pH	HA2 subunit			P18880		Influenza B virus
✓ Select	Fusion glycoprotein F0	I	binding to receptor	F1	100-537	FAGVIAGAALGVAT	P28886		Phocine distemper virus (PDV)

Link to the Tools

The following buttons will allow to send the selected sequence(s) (through the check boxes) to the Tools present in the server.

Actions	Entry's ID	Protein's Name	Sequence
🗑	653	Fusion glycoprotein F0	FAGVIAGAALGVATAAQITAGVALYQSNLNAQAIQSLRASLEQSNKAID...

Figure 18: Fusion Protein Page Scheme.

8. Columns such as the one in the dashed grey rectangle will redirected to the selected links; the links include UniProt, NCBI Protein, PDB and NCBI Taxonomy.

9. The purple box contains all the pagination links that will also determine the content of the table (each table contains a maximum of 10 entries).

Regarding the Tool pages, all pages contain a text box to insert the sequence input; the multiple alignment ones require at least 3 protein sequences in FASTA format to be inserted into the text box; the prediction ones just need a protein sequence. Within the text boxes, a small description for the type of expected input will appear; if those sequences come from the Fusion Protein page links, these inputs already come in the expected format. Besides that, the rest of the parameters are chosen within the pages, with clear labels to explain them, in the form of buttons, check boxes or select lists. The results of the tools can appear in new tabs to show text results, like in the Clustal and the text result of Weblogo of the back-end or to access other web domains, like BLAST or HMMER. They can also appear within the same page, like in ML prediction and the image result of Weblogo from the back-end.

5.2 CURRENT STATE OF THE APPLICATION

The application is mostly as first planned: it allows a relatively easy access to the MySQL database through a web interface that is user-friendly. However, some aspects must be taken into consideration: first, as an application, there are countless variable situations not accounted for within the development of its code that in the future might lead to errors, which only through extensive use can those problems be detected and resolved.

One other problem found is the time the web page takes to show up, which can take up to several seconds. In order to try to reduce this problem, most of the unnecessary / unused files and components, especially the ones that were automatically generated when the Nebular project was created, were removed from the project, which did help to reduce that loading time, but there is still room to improve.

Something that was not yet achieved was to automatically remove all the buttons of the "children data" that lead to empty tables. In order to tell Angular in real time that a particular entry does not require a given link to another table, multiple requests are necessary to be made, one for each type of data, through services within the "main service" (which corresponds to getting the main data for the table) to verify if the JSON responses given for that "children data" are empty. However, no solution was found to allow to send service requests within another services' request (inside the subscribe method); so, for the time being, all the links are created, even those that lead to empty tables, which affects the practicality of the app.

The most important feature lacking in the current application is the authentication. All the front-end application's table pages will contain code for both POST and PUT forms, in order to add or update an entry, respectively. This should be only performed by allowed users only, and the rest of the users to just suggest alterations to the database.

This authentication feature was already successfully performed in the back-end, through the use of JWT tokens (mentioned above), and correctly tested its performance through a program called Postman, that allows to emulate requests to a server. However, it was never correctly implemented in the Angular front-end, so those definitions were removed from the front-end in order not to interfere with the expected functioning of the web server.

One can also improve the security of the back-end, since in order to develop more quickly the application, the "*@csrf_exempt*" command was frequently applied to the Django views, which removes the necessity of providing a cookie while performing a POST request to that view, which later, if more widely used, can be a security risk for the webserver. Also, regarding the Django component of the project, the use of CORS is also a possible issue (as already mentioned). Alongside that, the functions that allow to save querysets into CSV data in the front-end used the *"bypassSecurityTrustResourceUrl"* function of the *"Sanitizer"* class, which means that no security checks are made when downloading the data.

Other aspects which require further work concern the WebLogo, Clustal, IEDB and ML Tools. Regarding the Weblogo, both methods (using http://weblogo.threeplusone.com/ webpage or using the solely the back-end) are flawed. The first one works as intended, if the number of sequences is not large. This is due to the fact that the page sends URL requests into the Weblogo Portal; if the string that compresses the URL request is too long (500 characters), the browser will show a request error. This is especially true because Weblogo requires sequence alignments, meaning that the higher the number of sequences, the longer the string will be, due to both the higher number of those sequences and also the gaps that are result of the Clustal alignment performed. Weblogo Python API was tried (https://weblogo.readthedocs.io/en/latest/intro.html) in the Django back-end, that uses the same server, but there were found two problems to properly use it: first, there is not enough information within both the API and webpage's documentations to accurately insert all the necessary parameters in the API to fully replicate the results produced to a given alignment in the Webpage (as already mentioned); the second problem was the fact that the view was unable to build PNG files using the API (even though the API works perfectly to write the Weblogo results into text files), because this Python package requires the introduction of the Python Ghostscipt package, but this package would generate several problems in the Django project, rendering the back-end virtually unusable.

Regarding the second method, the created weblogos are not exact replicas of the ones formed in the Weblogo site, mostly due to differences in the entropy values of the weblogo matrix (because this method requires the "*weblogo*" package described above). Also, the "*weblogo*" package seems to generate errors whenever a protein alphabet is selected to describe the multiple alignment needed to create the matrix. Again, this appears to be an error exclusively within a Django project, since in a stand-alone Python script this issue does not occur. So, a string alphabet containing all the aminoacids plus the gap symbol was used, and for that reason there is a column in the text matrix for the gap symbol, which under normal circumstances should not be necessary.

Like stated beforehand, regarding the Clustal analysis, the original intent was to use the EBI one. However, it was found that URL requests cannot be sent to the webpage (unlike the HMMER, which also uses a EBI Webpage), so the Python API code provided by the EBI in https://raw.githubusercontent.com/ebi-wp/webservice-clients/master/python/clustalo.py was experimented. Even though this script was perfectly usable outside the Django project, within it the API produces an error regarding the urllib package, meaning that the request needed cannot be performed, and so, no results are shown. The solution found was to use the console on the Django, but the output is less complete as if that API could be used, and there are concerns regarding the performance of this method.

Regarding the problems found in the IEDB page, some of the results of part of the possible methods return empty, mostly, but not limited to, from both Bepipred methods. This was confirmed by analysing the output of the JSON provided by the back-end, and performing the analysis on the same sequence using other methods an output was provided.

Finally, on the page for ML predictions, the ML prediction still have problems since its prediction output tends to take some time. Sometimes, this slowness causes problems within the Django container, stopping the process of that view, which results in no output for the analysis to be provided.

5.3 CURATION OF THE DATA

The data was mostly corrected as first intended. But there are a few issues that still needs future thought.

Most of the results of the inhibitors / antibodies search did not consider the strain information, which meant that the majority of the results within the same species are similar and not specific for each individual strain. Moreover, analyzing these results, one can see that the PubMed searches did not produce very useful results, so only the PDB entries (which do also have the same problem, albeit in a smaller scale) were used.

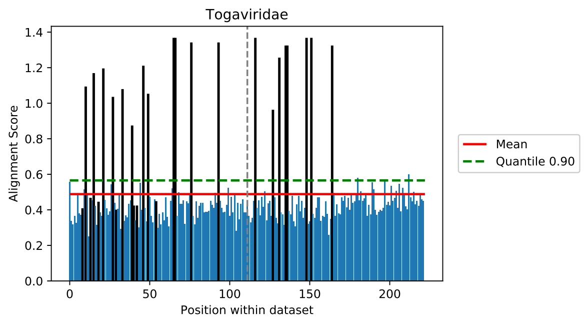
So, further work is required to gather all the known inhibitors / antibodies associated with each strain in the Database. This work might include searches through data mining in the case of the Pubmed articles search. Also, there are some reservations of the quality of the results of this type of automatic search, which likewise will require a future full manual review, or maybe to perform another type of search, since the one applied here seems to not differentiate rather well results between distinct strains of the same species.

Finally, as this is a field were info keeps being published by the community, there will be the necessity to add all relevant entries published in recent times in a future work.

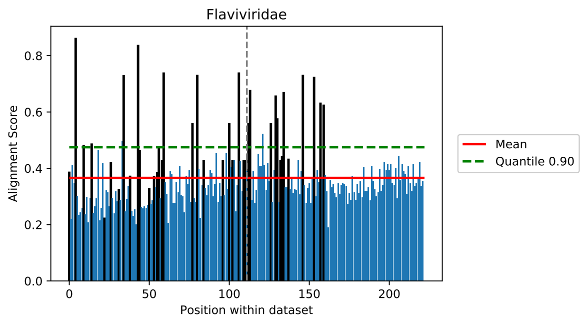
5.4 CONSERVATION FEATURES

The graphs in the figure 19 help to describe the utility of the generated conservation features. The families on those images were selected because they were the ones most common in the dataset. Analysing these six families, it is easy to see that the peptides that tend to have higher conservation scores are usually the ones belonging to the same family as the weblogo, as it was expected, even though some of the highest scores in most of these families are from peptides that do not belong the that specific family (this is especially true in the *Retroviridae* and *Herpesviridae*'s weblogos). The majority of the results for all the rest of the peptides tend to be below the average. Analyzing the scores of peptides that belong to the same family as the ones in the weblogo (figure 20), we can see that most of the higher results are from positive cases (i.e. FPep), indicating high degree of conservation, even though some of the results show high levels of conservation within non-FPep.

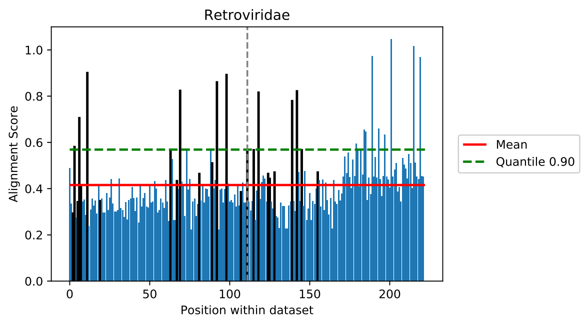
The problem of using these features in a ML process is that most of them are removed during feature selection. On the first dataset (where the negative cases are non-FPep and non-transmembrane domains sequences), only three of the 13 features are maintained after a SVC feature selection (which retains a total of 76 features); on the second dataset (negative



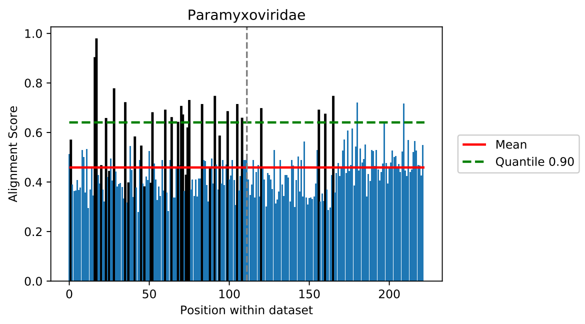
(a) Scores against the Togaviridae family weblogo.



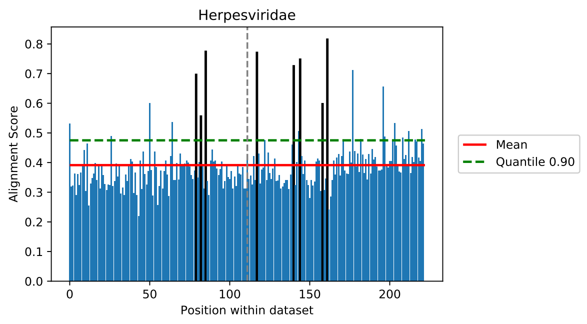
(b) Conservation scores against the Flaviviridae family weblogo.



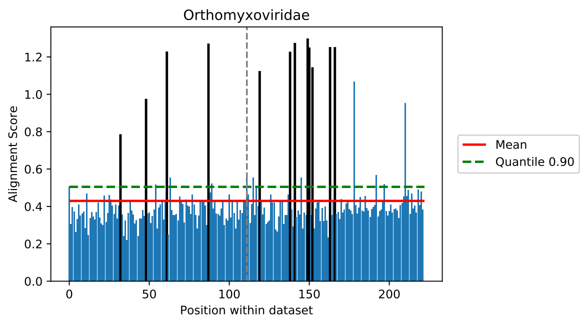
(c) Conservation scores against the Retroviridae family weblogo.



(d) Scores against the Paramyxoviridae family weblogo.



(e) Scores against the Herpesviridae family weblogo.



(f) Scores against the Orthomyxoviridae family weblogo.

Figure 19: Conservation scores against some of the weblogos built from the peptides of a dataset, where the first 111 elements are non-Fusion Peptides (on the left of the dashed line) and rest are Fusion Peptides (on the right). The peptides from viruses that belong to the same taxonomy family as the ones on the webLogo are in black.

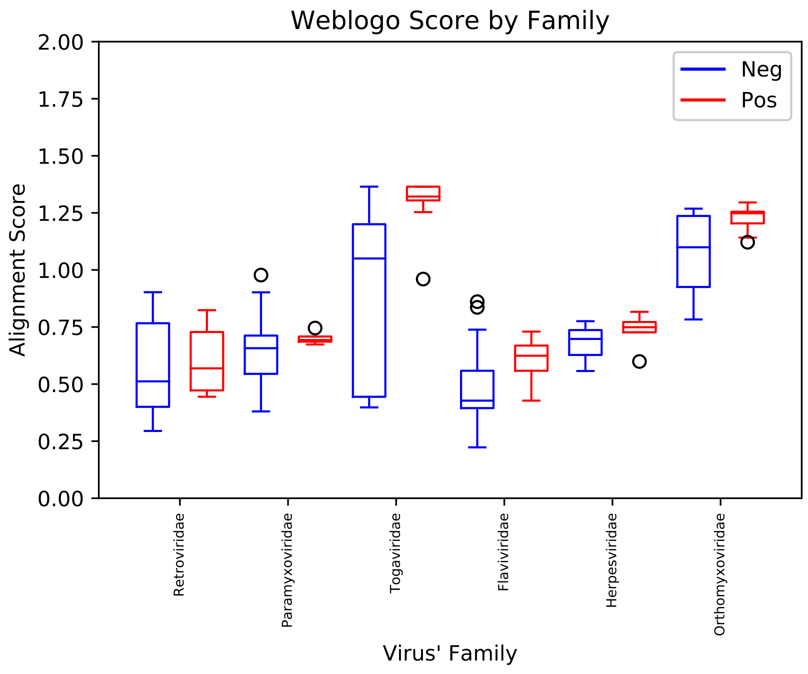


Figure 20: Boxplot of the conservation scores of some of the most common families in the dataset described in the figure 19. The scores in those boxplots are exclusively from peptides that belong to the same family as the ones in the weblogo.

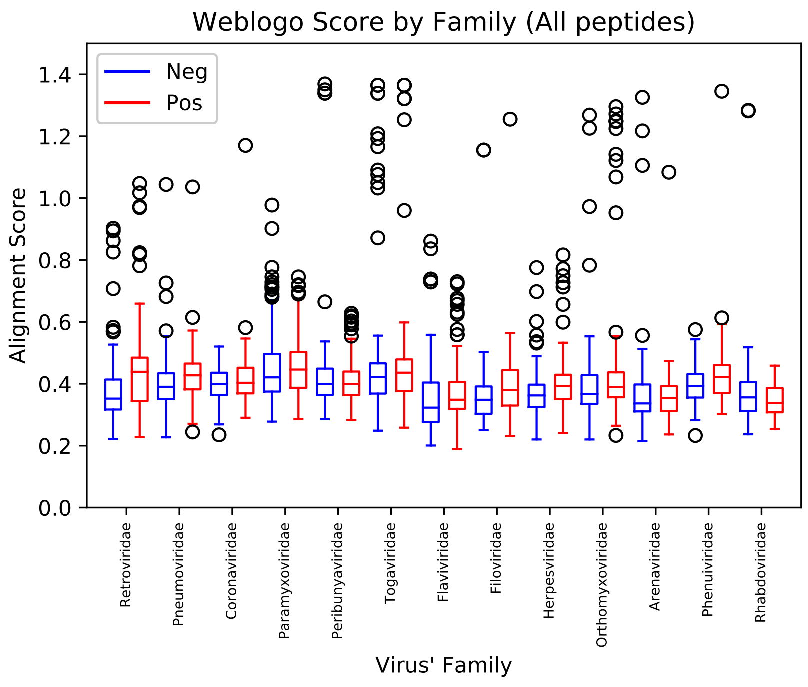


Figure 21: Boxplot of the conservation scores of all families against all peptides in the dataset described in the figure 19.

cases are transmembrane domains sequences), none of the conservation feature is retained after feature selection (which saves 58 features); in the third dataset (negative cases are a mix of both previous datasets' negative cases), only one feature was maintained (103 total features stored). For that reason, the performance scores of the models do not differ significantly from the ones presented by Sequeira (Sequeira et al., 2019).

Furthermore, a closer look to the results of the figure 21 show that between those weblogos, the maximal values differ considerably. Besides that, if extracting all these features from all the sequences, regardless of their family, the data shows that there is not a clear distinction between negative cases and positive cases. All these motives make obtaining a coherent single feature from those individual scores difficult. Other features were tested based on these conservation scores, but none of them achieved a desired degree of distinction between negative and positive cases.

For these reasons, right now the conservation scores are more useful as a complementary information to the ML prediction output and not as part of the ML pipeline.

6

CONCLUSION AND FURTHER WORK

Even though there are still a lot of aspects that require future work, as been stated in the previous topics, this work provides a significant contribution for the scientific community, since it presents a tool that can be useful to any researcher interested in obtaining several types of information regarding VFP in a user-friendly way.

In the context we currently live in 2020/2021, this sort of information is increasingly important. As we all know, this past year a lot of research was put into SARS-COV-2, with a particular emphasis for its VFP (the Spike protein) and FPep as therapeutic targets (as mentioned above) to develop potential vaccines to control this pandemic. This server is a simple way for researchers to be able to obtain information regarding these enveloped viruses (and their VFP) in the future, since researchers can find that virus' taxonomy information, its VFP and FPep, besides information regarding other similar viruses from the same family that can also provide key pieces of data, due to their similarity and because some of those viruses are more well-understood than SARS-COV-2.

Currently, the application achieves most of the proposed objectives: it allows a relatively easy access to the MySQL database through a web interface that is user-friendly. The features that allow users to alter the database were developed, but the authentication features that allow to distinguish between authorized and non-authorized users are still not fully implemented. All the desired bioinformatics tools are inserted into the application, through web interfaces, even if some of them require some work in order to improve functionalities and performance.

ML models that are capable of predicting FPeps within the VFP sequence were trained as described by Sequeira (Sequeira et al., 2019) and inserted into the server. In the same page, the end user can also obtain a conservation score of a peptide against a virus' taxonomy family multiple alignment, information that can complement the output of the ML prediction, since FPep is a conserved region within the VFP. Finally, a full manual curation of the database information was performed.

Future work within the website will include fixing bugs and unaccounted situations within the application; adding features to make the application more user-friendly; improving the initial loading time of the website; upgrading the app's aesthetic look: possible corrections may include deleting all the unnecessary options and features on some of the components of the app; removing all the weblinks present in the data tables that lead to webpages with empty tables; introducing all the necessary authentication features lacking currently on the project and asserting the cybersecurity of the webserver for the end user.

Specific tasks necessary to improve the utility of this application for the scientific community will include fixing the Weblogo, Clustal and IEDB's known problems, mostly by correcting the mentioned issues and upgrading the performance of the server, as well as to provide more options in those tools to increase their functionalities; looking into possible solutions to make getting the outputs of the ML tools faster, which would also allow to prevent problems within the server; studying and implementing alternative ways to consider conservation features and, finally, adding all the relevant entries that were published in recent times and correcting parts of the information present in the server.

BIBLIOGRAPHY

Ethem Alpaydın. *Introduction to Machine Learning Second Edition*. Massachusetts Institute of Technology, second edi edition, 2010. ISBN 978-0-262-01243-0.

Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, 11(2):47–58, 2013. ISSN 12140287. doi: 10.2478/v10136-012-0031-x.

Beatriz Apellániz, Nerea Huarte, Eneko Largo, and José L. Nieva. The three lives of viral fusion peptides. *Chemistry and Physics of Lipids*, 181:40–55, 2014. ISSN 18732941. doi: 10.1016/j.chemphyslip.2014.03.003.

Mariette Awad and Rahul Khanna. *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*. Apres Open, 2015. ISBN 9781430259909. doi: 10.1007/978-1-4302-5990-9.

Eduard Baquero, Aurélie A. Albertini, Patrice Vachette, Jean Lepault, Stéphane Bressanelli, and Yves Gaudin. Intermediate conformations during viral fusion glycoprotein structural transition. *Current Opinion in Virology*, 3(2):143–150, 2013. ISSN 18796265. doi: 10.1016/j.coviro.2013.03.006.

Asa Ben-Hur and Jason Weston. A User's Guide to Support Vector Machines. *Methods in Molecular Biology*, (January 2010):223–239, 2010. doi: 10.1007/978-1-60327-241-4_13.

Pratiti Bhadra, Jielu Yan, Jinyan Li, Simon Fong, and Shirley W.I. Siu. AmPEP: Sequence-based prediction of antimicrobial peptides using distribution patterns of amino acid properties and random forest. *Scientific Reports*, 8(1):1–10, 2018. ISSN 20452322. doi: 10.1038/s41598-018-19752-w.

Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLoS ONE*, 12(6):1–17, 2017. ISSN 19326203. doi: 10.1371/journal.pone.0177678.

Mark L Braunstein. *Mark L. Braunstein - Health Informatics on FHIR_ How HL7's New API is Transforming Healthcare-Springer International Publishing (2018)*. Springer, 2018. ISBN 9783319934136.

Stephen K. Burley, Helen M. Berman, Charmi Bhikadiya, Chunxiao Bi, Li Chen, Luigi Di Costanzo, Cole Christie, Ken Dalenberg, Jose M. Duarte, Shuchismita Dutta, Zukang

Feng, Sutapa Ghosh, David S. Goodsell, Rachel K. Green, Vladimir Guranović, Dmytro Guzenko, Brian P. Hudson, Tara Kalro, Yuhe Liang, Robert Lowe, Harry Namkoong, Ezra Peisach, Irina Periskova, Andreas Prlić, Chris Randle, Alexander Rose, Peter Rose, Raul Sala, Monica Sekharan, Chenghua Shao, Lihua Tan, Yi Ping Tao, Yana Valasatava, Maria Voigt, John Westbrook, Jesse Woo, Huanwang Yang, Jasmine Young, Marina Zhuravleva, and Christine Zardecki. RCSB Protein Data Bank: Biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Research*, 47(D1):D464–D474, 2019. ISSN 13674803. doi: 10.1093/nar/gky1004.

Zhen Chen, Pei Zhao, Fuyi Li, Tatiana T Marquez-Lago, André Leier, Jerico Revote, Yan Zhu, David R Powell, Tatsuya Akutsu, Geoffrey I Webb, Kuo-Chen Chou, A Ian Smith, Roger J Daly, Jian Li, and Jiangning Song. iLearn: an integrated platform and meta-learner for feature engineering, machine-learning analysis and modeling of DNA, RNA and protein sequence data. *Briefings in Bioinformatics*, (April), 2019. ISSN 1477-4054. doi: 10.1093/bib/bbz041.

Vincent Cho. A comparison of three different approaches to tourist arrival forecasting. *Tourism Management*, 24:323–330, 2003.

Peter J.A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J.L. De Hoon. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009. ISSN 13674803. doi: 10.1093/bioinformatics/btp163.

G Crooks, G Hon, J Chandonia, and S Brenner. WebLogo: A Sequence Logo Generator. *Genome Research*, 14:1188–1190, 2004. ISSN 1088-9051. doi: 10.1101/gr.849004.1.

Thomas G Dietterich. Ensemble Methods in Machine Learning. *International Workshop on multiple classifier systems*, pages 1–15, 2000. doi: 10.1007/3-540-45014-9_1. URL http://www.cs.orst.edu/~tgd.

DjangoRESTFramework. Django rest framework. https://www.django-rest-framework.org/, 2020a. Accessed: 2020-01-16 at 19:12.

DjangoRESTFramework. Filtering. https://www.django-rest-framework.org/api-guide/filtering/, 2020b. Accessed: 10/02/2020 at 11:29.

DjangoRESTFramework. Pagination. https://www.django-rest-framework.org/api-guide/pagination/, 2020c. Accessed: 10/02/2020 at 11:34.

DjangoRESTFramework. Serializers. https://www.django-rest-framework.org/api-guide/serializers/, 2020d. Accessed: 16/01/2020 at 19:16.

DjangoSoftwareFoundation. Django, 2019. URL https://www.djangoproject.com/start/overview/. Accessed: 03/10/2019 at 18:50.

DjangoSoftwareFoundation. Integrating django with a legacy database. https://docs.djangoproject.com/en/3.0/howto/legacy-databases/, 2020a. Accessed: 16/01/2020 at 23:31.

DjangoSoftwareFoundation. Writing your first django app, part 1. https://docs.djangoproject.com/en/3.0/intro/tutorial01/, 2020b. Accessed: 16/01/2020 at 22:49.

DjangoSoftwareFoundation. Writing your first django app, part 2. https://docs.djangoproject.com/en/3.0/intro/tutorial02/, 2020c. Accessed: 16/01/2020 at 23:29.

DjangoSoftwareFoundation. Writing your first django app, part 3. https://docs.djangoproject.com/en/3.0/intro/tutorial03/, 2020d. Accessed: 16/01/2020 at 23:37.

DjangoSoftwareFoundation. Writing your first django app, part 5. https://docs.djangoproject.com/en/3.0/intro/tutorial05/, 2020e. Accessed: 23/01/2020 at 22:42.

DjangoSoftwareFoundation. How to deploy with wsgi. https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/, 2020f. Accessed: 10/10/2020 at 15:57.

DockerInc. Overview of docker compose. https://docs.docker.com/compose/, 2020a. Accessed: 02/08/2020 at 18:22.

DockerInc. Quickstart: Compose and django. https://docs.docker.com/compose/django/, 2020b. Accessed: 11/09/2020 at 13:38.

DockerInc. mysql. https://hub.docker.com/_/mysql, 2020c. Accessed: 11/09/2020 at 13:54.

DockerInc. Orientation and setup. https://docs.docker.com/get-started/, 2020d. Accessed: 02/08/2020 at 17:28.

DockerInc. docker push. https://docs.docker.com/engine/reference/commandline/push/, 2020e. Accessed: in 11/09/2020 at 13:38.

DockerInc. Repositories. https://docs.docker.com/docker-hub/repos/, 2020f. Accessed: 11/09/2020 at 13:38.

DockerInc. Orientation and setup. https://docs.docker.com/engine/reference/builder/, 2020g. Accessed: 02/08/2020 at 18:07.

Richard M. Epand. Fusion peptides and the mechanism of viral fusion. *Biochimica et Biophysica Acta - Biomembranes*, 1614(1):116–121, 2003. ISSN 00052736. doi: 10.1016/S0005-2736(03)00169-X.

T. N. Figueira, L. M. Palermo, A. S. Veiga, D. Huey, C. A. Alabi, N. C. Santos, J. C. Welsch, C. Mathieu, B. Horvat, S. Niewiesk, A. Moscona, M. A. R. B. Castanho, and M. Porotto. In Vivo Efficacy of Measles Virus Fusion Protein-Derived Peptides Is Modulated by the Properties of Self-Assembly and Membrane Residence . *Journal of Virology*, 91(1), 2017. ISSN 0022-538X. doi: 10.1128/jvi.01554-16.

Robert D. Finn, Jody Clements, and Sean R. Eddy. HMMER web server: Interactive sequence similarity searching. *Nucleic Acids Research*, 39(SUPPL. 2):29–37, 2011. ISSN 03051048. doi: 10.1093/nar/gkr367.

William Gilpin. PyPDB: a Python API for the Protein Data Bank. *Bioinformatics*, 32(1): 159–160, 09 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv543. URL https://doi.org/10.1093/bioinformatics/btv543.

Maria J. Gomara, Yolanda Perez, Javier P. Martinez, Ramon Barnadas-Rodriguez, Anke Schultz, Hagen von Briesen, Alex Peralvarez-Marin, Andreas Meyerhans, and Isabel Haro. Peptide Assembly on the Membrane Determines the HIV-1 Inhibitory Activity of Dual-Targeting Fusion Inhibitor Peptides. *Scientific Reports*, 9(1):1–13, 2019. ISSN 20452322. doi: 10.1038/s41598-019-40125-4.

Zohre Hajisharifi, Moien Piryaiee, Majid Mohammad Beigi, Mandana Behbahani, and Hassan Mohabatkar. Predicting anticancer peptides with Chou's pseudo amino acid composition and investigating their mutagenicity via Ames test. *Journal of Theoretical Biology*, 341:34–40, 2014. ISSN 00225193. doi: 10.1016/j.jtbi.2013.08.037.

Jan L. Harrington. Chapter 5 - the relational data model. In Jan L. Harrington, editor, *Relational Database Design and Implementation (Fourth Edition)*, pages 89 – 105. Morgan Kaufmann, Boston, fourth edition edition, 2016. ISBN 978-0-12-804399-8. doi: https://doi.org/10.1016/B978-0-12-804399-8.00005-3. URL http://www.sciencedirect.com/science/article/pii/B9780128043998000053.

Stephen C. Harrison. Viral membrane fusion. *Virology*, 479-480:498–507, 2015. ISSN 10960341. doi: 10.1016/j.virol.2015.03.043.

Md Rafiul Hassan and Baikunth Nath. Stock market forecasting using Hidden Markov Model: A new approach. In *Proceedings - 5th International Conference on Intelligent Systems Design and Applications 2005, ISDA '05*, pages 192–196, 2005. ISBN 0769522866. doi: 10.1109/ISDA.2005.85.

Michael Herman. Dockerizing an angular app. https://mherman.org/blog/dockerizing-an-angular-app/, 2019. Accessed: 11/09/2020 at 14:22.

Frederick M. Hughson. Structural characterization of viral fusion proteins. *Current Biology*, 5(3):265–274, 1995. ISSN 09609822. doi: 10.1016/S0960-9822(95)00057-1.

Lisa A Jackson, Evan J Anderson, Nadine G Rouphael, Paul C Roberts, Mamodikoe Makhene, Rhea N Coler, Michele P McCullough, James D Chappell, Mark R Denison, Laura J Stevens, et al. An mrna vaccine against sars-cov-2—preliminary report. *New England Journal of Medicine*, 2020.

Mark Johnson, Irena Zaretskaya, Yan Raytselis, Yuri Merezhuk, Scott Mcginnis, and Thomas L Madden. NCBI BLAST : a better web interface. *Nucleic Acids Research*, 36(April):5–9, 2008. doi: 10.1093/nar/gkn201.

JWT. Jwt. https://jwt.io/, 2020. Accessed: 14/10/2020 at 11:39.

Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *International Joint Conference of Artificial Intelligence*, (June), 1995.

Quansheng Kuang and Lei Zhao. A practical GPU based kNN algorithm. *International symposium on computer science and computational technology (ISCSCT)*, 7(3):151–155, 2009.

Vinod Kumar, Piyush Agrawal, Rajesh Kumar, Sherry Bhalla, Salman Sadullah Usmani, Grish C. Varshney, and Gajendra P.S. Raghava. Prediction of cell-penetrating potential of modified peptides containing natural and chemically modified residues. *Frontiers in Microbiology*, 9(APR):1–10, 2018. ISSN 1664302X. doi: 10.3389/fmicb.2018.00725.

Sneh Lata, Nitish K. Mishra, and Gajendra P.S. Raghava. AntiBP2: Improved version of antibacterial peptide prediction. *BMC Bioinformatics*, 11(SUPPLL.1):1–7, 2010. ISSN 14712105. doi: 10.1186/1471-2105-11-S1-S19.

Saro Lee. Application of logistic regression model and its validation for landslide susceptibility mapping using GIS and remote sensing data. *International Journal of Remote Sensing*, 26(7):1477–1491, 2005. ISSN 13665901. doi: 10.1080/01431160412331331012.

Jing Li, Peng Zhan, and Xinyong Liu. Targeting the entry step of sars-cov-2: a promising therapeutic approach. *Signal transduction and targeted therapy*, 5(1):1–2, 2020.

Kuo-Bin Li. ClustalW-MPI: ClustalW analysis using distributed and parallel computing. *Bioinformatics*, 19(12):1585–1586, 08 2003. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg192. URL https://doi.org/10.1093/bioinformatics/btg192.

Bin Liu, Xin Gao, and Hanyu Zhang. BioSeq-Analysis2.0: an updated platform for analyzing DNA, RNA and protein sequences at sequence level and residue level based on machine learning approaches. *Nucleic acids research*, 47(20), 2019. ISSN 13624962. doi: 10.1093/nar/gkz740.

Wei-Yin Loh. Classification and regression trees. *WIREs Data Mining and Knowledge Discovery*, 1(January/February):14–23, 2011. doi: 10.1002/widm.8.

Tom Madden. The BLAST Sequence Analysis Tool. In *The NCBI Handbook [Internet]*. Bethesda (MD): National Center for Biotechnology Information (US), 2nd editio edition, 2013.

Balachandran Manavalan, Shaherin Basith, Tae Hwan Shin, Sun Choi, Myeong Ok Kim, and Gwang Lee. MLACP: Machine-learning-based prediction of anticancer peptides. *Oncotarget*, 8(44):77121–77136, 2017. ISSN 19492553. doi: 10.18632/oncotarget.20365.

Balachandran Manavalan, Sathiyamoorthy Subramaniyam, Tae Hwan Shin, Myeong Ok Kim, and Gwang Lee. Machine-Learning-Based Prediction of Cell-Penetrating Peptides and Their Uptake Efficiency with Improved Accuracy. *Journal of Proteome Research*, 17(8): 2715–2726, 2018. ISSN 15353907. doi: 10.1021/acs.jproteome.8b00148.

M. Marsh. *Current topics in microbiology and immunology*, volume 6. Springer, 2005. ISBN 3540214305. doi: 10.3109/08820137709055812.

Mark Marsh and Ari Helenius. Virus entry: Open sesame. *Cell*, 124(4):729–740, 2006. ISSN 00928674. doi: 10.1016/j.cell.2006.02.007.

MDN Web Docs. What is a web server ? , 2019. URL https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server#Deeper_dive. Accessed: 03/10/2019 at 18:33.

Prabina Kumar Meher, Tanmaya Kumar Sahu, Varsha Saini, and Atmakuri Ramakrishna Rao. Predicting antimicrobial peptides with improved accuracy by incorporating the compositional, physico-chemical and structural features into Chou's general PseAAC. *Scientific Reports*, 7(January):1–12, 2017. ISSN 20452322. doi: 10.1038/srep42362.

Yorgo Modis, Steven Ogata, David Clements, and Stephen C Harrison. Structure of the dengue virus envelope protein after membrane fusion. *Nature*, 427(February 2004):313–319, 2004. doi: 10.1038/nature02165.

Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python A guide for Data Scientists*. Number 9. O'Reilly Media, Inc, Sebastopol, CA, USA, 2016. ISBN 9788578110796.

NCBI. BLAST Homepage and Selected Search Pages. *NCBI Handout Series*, pages 1–8, 2016. URL ftp://ftp.ncbi.nlm.nih.gov/pub/factsheets/HowTo_BLASTGuide.pdf.

NCBI Resource Coordinators. Database resources of the National Center for Biotechnology Information [ftp://ftp.ncbi.nih.gov/genomes/Bacteria]. *Nucleic acids research*, 44 (November 2015):7–19, 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1290.

William S. Noble. What is a support vector machine? *Nature Biotechnology*, 24(12):1565–1567, 2006. ISSN 10870156. doi: 10.1038/nbt1206-1565.

Christoffer Noring and Pablo Deeleman. *Learning Angular 5*. Packt Publishing, Birmingham, UK, second edi edition, 2017. ISBN 978-1-78712-492-9. URL https://www.cambridge.org/core/product/identifier/CBO9781107415324A009/type/book_part.

Oracle. What is mysql? https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, 2021. Accessed: 12/01/2021 at 00:10.

OracleCorporation. Mysql connector/python developer guide. https://dev.mysql.com/doc/connector-python/en/, 2020. Accessed: 07/05/2020 at 15:20.

M. Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005. ISSN 01431161. doi: 10.1080/01431160412331269698.

Tina R. Patil and S. S. Sherekar. Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification. *International Journal of Intelligent Systems and Applications in Engineering*, 6(2):256–261, 2013. ISSN 2147-6799. doi: 10.18201/ijisae.2019252786.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn. *Journal of Machine Learning Research*, 12:2825–2830, jun 2011.

Billy Peralta and Alvaro Soto. Embedded local feature selection within mixture of experts. *Information Sciences*, 269:176–187, 2014. ISSN 00200255. doi: 10.1016/j.ins.2014.01.008.

Sara Pereira, Diana Lousa, and Isabel Rocha. Building a database and development of a Machine Learning algorithm to identify and characterize viral Fusion Peptides. Master's thesis, Universidade do Minho, 2019.

Julia Plekhanova. Evaluating web development frameworks: Django, Ruby on Rails and CakePHP. Technical Report September, Fox School of Business, Temple University, 2009. URL http://ibit.temple.edu/wp-content/uploads/2011/03/IBITWebframeworks.pdf.

Richard K. Plemper. Cell entry of enveloped viruses. *Current Opinion in Virology*, 1(2):92–100, 2011. ISSN 18796265. doi: 10.1016/j.coviro.2011.06.002.

Fernando P Polack, Stephen J Thomas, Nicholas Kitchin, Judith Absalon, Alejandra Gurtman, Stephen Lockhart, John L Perez, Gonzalo Pérez Marc, Edson D Moreira, Cristiano Zerbini, et al. Safety and efficacy of the bnt162b2 mrna covid-19 vaccine. *New England Journal of Medicine*, 383(27):2603–2615, 2020.

Simon C Potter, Sean R Eddy, Youngmi Park, Rodrigo Lopez, Robert D Finn, and The Hmmer. HMMER web server : 2018 update. *Nucleic Acids Research*, 46(June):200–204, 2018. doi: 10.1093/nar/gky448.

Ana Sequeira, Miguel Rocha, and Diana Lousa. Building an automated platform for the classification of peptides/proteins using machine learning. Master's thesis, Universidade do Minho, 2019.

Ana Marta Sequeira, Diana Lousa, and Miguel Rocha. Propythia: A python automated platform for the classification of proteins using machine learning. In Gabriella Panuccio, Miguel Rocha, Florentino Fdez-Riverola, Mohd Saberi Mohamad, and Roberto Casado-Vara, editors, *Practical Applications of Computational Biology & Bioinformatics, 14th International Conference (PACBB 2020)*, pages 32–41, Cham, 2021. Springer International Publishing. ISBN 978-3-030-54568-0.

Fabian Sievers and Desmond G. Higgins. *Multiple Sequence Alignment Methods*. Humana Press, david j. r edition, 2014. ISBN 9781627036450. doi: 10.1017/cbo9780511617829.007.

Fabian Sievers, Andreas Wilm, David Dineen, Toby J. Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D. Thompson, and Desmond G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(539), 2011. ISSN 17444292. doi: 10.1038/msb.2011.75.

Super-powered by Google. AngularJS, 2018. URL https://angularjs.org/. Accessed: 30/10/2019 at 10:21.

Super-poweredbyGoogle. Introduction to angular concepts. https://angular.io/guide/architecture, 2020a. Accessed: 04/03/2020 at 17:15.

Super-poweredbyGoogle. Introduction to components and templates. https://angular.io/guide/architecture-components, 2020b. Accessed: 07/03/2020 at 16:04.

Super-poweredbyGoogle. Managing data. https://angular.io/start/data, 2020c. Accessed: 27/01/2020 at 11:50.

Super-poweredbyGoogle. Communicating with backend services using http. https://angular.io/guide/http, 2020d. Accessed: 07/03/2020 at 16:27.

Super-poweredbyGoogle. Adding navigation. https://angular.io/start/routing, 2020e. Accessed: 23/01/2020 at 23:15.

Super-poweredbyGoogle. Routerlink. https://angular.io/api/router/RouterLink, 2020f. Accessed: 20/09/2020 at 00:28.

Ammar Tareen and Justin B. Kinney. Logomaker: Beautiful sequence logos in python. *bioRxiv*, 2019. doi: 10.1101/635029. URL https://www.biorxiv.org/content/early/2019/05/13/635029.

Élodie Teissier and Eve Isabelle Pécheur. Lipids as modulators of membrane fusion mediated by viral fusion proteins. *European Biophysics Journal*, 36(8):887–899, 2007. ISSN 01757571. doi: 10.1007/s00249-007-0201-z.

The UniProt Consortium. UniProt: A worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 2019. ISSN 13624962. doi: 10.1093/nar/gky1049.

TivixInc. Welcome to django-rest-auth's documentation! https://django-rest-auth.readthedocs.io/en/latest/index.html, 2018a. Accessed: 14/10/2020 at 11:01.

TivixInc. Installation. https://django-allauth.readthedocs.io/en/latest/installation.html, 2018b. Accessed: 14/10/2020 at 11:28.

Hélio A. Tomás, Daniel A. Mestre, Ana F. Rodrigues, Miguel R. Guerreiro, Manuel J.T. Carrondo, and Ana Sofia Coroadinha. Improved galv-tr glycoproteins to pseudotype lentiviral vectors: Impact of viral protease activity in the production of lv pseudotypes. *Molecular Therapy - Methods Clinical Development*, 15:1 – 8, 2019. ISSN 2329-0501. doi: https://doi.org/10.1016/j.omtm.2019.08.001. URL http://www.sciencedirect.com/science/article/pii/S2329050119300804.

Atul Tyagi, Pallavi Kapoor, Rahul Kumar, Kumardeep Chaudhary, Ankur Gautam, and G. P.S. Raghava. In silico models for designing and discovering novel anticancer peptides. *Scientific Reports*, 3:1–8, 2013. ISSN 20452322. doi: 10.1038/srep02984.

Randi Vita, James A. Overton, Jason A. Greenbaum, Julia Ponomarenko, Jason D. Clark, Jason R. Cantrell, Daniel K. Wheeler, Joseph L. Gabbard, Deborah Hix, Alessandro Sette, and Bjoern Peters. The immune epitope database (IEDB) 3.0. *Nucleic Acids Research*, 43(D1):D405–D412, 10 2014. ISSN 0305-1048. doi: 10.1093/nar/gku938. URL https://doi.org/10.1093/nar/gku938.

Randi Vita, James A Overton, Jason A Greenbaum, Julia Ponomarenko, D Clark, Jason R Cantrell, Daniel K Wheeler, Joseph L Gabbard, Deborah Hix, Alessandro Sette, and Bjoern Peters. The immune epitope database (IEDB) 3.0. *Nucleic Acids Research*, 43(October 2014): 405–412, 2015. doi: 10.1093/nar/gku938.

Randi Vita, Swapnil Mahajan, James A Overton, Sandeep Kumar Dhanda, Sheridan Martini, Jason R Cantrell, Daniel K Wheeler, Alessandro Sette, and Bjoern Peters. The Immune Epitope Database (IEDB): 2018 update. *Nucleic Acids Research*, 47(D1):D339–D343, 10 2018. ISSN 0305-1048. doi: 10.1093/nar/gky1006. URL https://doi.org/10.1093/nar/gky1006.

Leyi Wei, Chen Zhou, Huangrong Chen, Jiangning Song, and Ran Su. ACPred-FL: a sequence-based predictor using effective feature representation to improve the prediction of anti-cancer peptides. *Bioinformatics*, 34(23):4007–4016, 2018. ISSN 13674811. doi: 10.1093/bioinformatics/bty451.

Winfried Weissenhorn, Andreas Hinz, and Yves Gaudin. Virus membrane fusion. *FEBS Letters*, 581(11):2150–2155, 2007. ISSN 00145793. doi: 10.1016/j.febslet.2007.01.093.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. *Advances in Neural Information Processing Systems*, 2001. ISSN 10495258.

Judith M. White, Sue E. Delos, Matthew Brecher, and Kathryn Schornberg. Structures and mechanisms of viral membrane fusion proteins: Multiple variations on a common theme. *Critical Reviews in Biochemistry and Molecular Biology*, 43(3):189–219, 2008. ISSN 10409238. doi: 10.1080/10409230802058320.

Matthew Whitehead and Larry Yaeger. Sentiment mining using ensemble classification models. *Innovations and Advances in Computer Sciences and Engineering*, pages 509–514, 2010. doi: 10.1007/978-90-481-3658-2_89.

Martin Williams. Top 8 Best Backend Frameworks, 2019. URL https://www.keycdn.com/blog/best-backend-frameworks. Accessed: 03/10/2019 at 17:55.

M. C. Wolf, A. N. Freiberg, T. Zhang, Z. Akyol-Ataman, A. Grock, P. W. Hong, J. Li, N. F. Watson, A. Q. Fang, H. C. Aguilar, M. Porotto, A. N. Honko, R. Damoiseaux, J. P. Miller, S. E. Woodson, S. Chantasirivisal, V. Fontanes, O. A. Negrete, P. Krogstad, A. Dasgupta, A. Moscona, L. E. Hensley, S. P. Whelan, K. F. Faull, M. R. Holbrook, M. E. Jung, and B. Lee. A broad-spectrum antiviral targeting entry of enveloped viruses. *Proceedings of the National Academy of Sciences*, 107(7):3157–3162, 2010. ISSN 0027-8424. doi: 10.1073/pnas.0909587107.

Nicholas C Wu and Ian A Wilson. Structural insights into the design of novel anti-influenza therapies. *National Structure & Molecular Biology*, 2018. doi: 10.1038/s41594-018-0025-9.

Sijia Wu, Jiuqiang Han, Ruiling Liu, Jun Liu, and Hongqiang Lv. A computational model for predicting fusion peptide of retroviruses. *Computational Biology and Chemistry*, 61:245–250, 2016. ISSN 14769271. doi: 10.1016/j.compbiolchem.2016.02.013.

Sijia Wu, Xiaoming Wu, Jie Tian, Xiaobo Zhou, and Liyu Huang. PredictFP2: a new computational model to predict fusion peptide domain in all retroviruses. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019. ISSN 1545-5963. doi: 10.1109/tcbb.2019.2898943.

Xuan Xiao, Pu Wang, Wei Zhong Lin, Jian Hua Jia, and Kuo Chen Chou. IAMP-2L: A two-level multi-label classifier for identifying antimicrobial peptides and their functional types. *Analytical Biochemistry*, 436(2):168–177, 2013. ISSN 00032697. doi: 10.1016/j.ab.2013.01.019.

Dong Xu. Protein Databases on the Internet. *Current Protocols in Molecular Biology*, 2004. ISSN 1934-3647. doi: 10.1002/0471142727.mb1904s68.

Yongtao Xu, Shui Yu, Jian Wei Zou, Guixiang Hu, Noorsaadah A.B.D. Rahman, Rozana Binti Othman, Xia Tao, and Meilan Huang. Identification of peptide inhibitors of enveloped viruses using support vector machine. *PLoS ONE*, 10(12):1–15, 2015. ISSN 19326203. doi: 10.1371/journal.pone.0144171.

Otto Yiu and Adam Johnson. django-cors-headers 3.6.0. https://pypi.org/project/django-cors-headers/, 2019. Accessed: 28/12/2019 at 19:21.

Harry Zhang. Exploring conditions for the optimality of naïve bayes. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(2):183–198, 2005. ISSN 02180014. doi: 10.1142/S0218001405003983.