Universidade do Minho
Escola de Engenharia

Marco André Lopes Faria

Automation of Ophthalmic Diagnosis: Glaucoma as a Case Study

October 2022

Automation of Ophthalmic Diagnosis: Glaucoma as a Case Study

Marco André Lopes Faria

UMinho | 2022

Universidade do Minho
Escola de Engenharia

Marco André Lopes Faria
A80719

Automation of Ophthalmic Diagnosis: Glaucoma as a Case Study

MSc Dissertation
Integrated Master's in Engineering and Management of Information Systems

Thesis performed under supervision of
Professor José Luís Mota Pereira

October 2022

COPYRIGHT

Third parties can use this academic work as long as the internationally accepted rules and good practices are respected, with regard to copyright and related rights.

Thus, the present work may be used under the terms set out in the license below. If the user needs permission to be able to use the work under conditions not foreseen in the above-mentioned licensing, he/she should contact the author, through the RepositóriUM of the University of Minho.

Licença concedida aos utilizadores deste trabalho

Atribuição
CC BY
https://creativecommons.org/licenses/by/4.0/

ACKNOWLEDGEMENTS

First, I would like to thank my family for the support and motivation that was given to me throughout all this time that made it possible for me to finish my education and achieve all my goals.

I would like to thank Luís Cortesão and all the Altice Labs team for the tireless support that was given to me throughout these 11 months of internship.

Finally, I would like to thank Professor José Luís Mota Pereira for his availability and support whenever it was needed.

DECLARATION OF INTEGRITY

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

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

ABSTRACT

Automation of Ophthalmic Diagnosis: Glaucoma as a Case Study

With ocular diseases increasing at an alarming rate due to population aging, it is relevant to explore new technologies to meet patients' needs. As we witness the exponential evolution of the huge field that is Artificial Intelligence (AI), new methodologies and cognitive models have been developed for sustainability and optimization in several sectors of our society, namely in healthcare data analysis. By incorporating intelligent systems in clinical diagnostics, it is possible to achieve a high degree of information integration, data processing, and a theoretical improvement in the speed of diagnosis. With the presented project we had the goal of training and testing supervised models with ophthalmic data, to predict and diagnose eye diseases, such as Glaucoma.

To this effect, it was carried out a Machine Learning (ML) project following a typical ML methodology. The methodology is composed of stages such as Business Understanding, Data Acquisition and Transformation, Modeling and Deployment. The document is structured according to these stages and in each of them is presented all the work that was done.

The main objective of this research was to test several features and algorithms (LGBM, Random Forest, KNN, SVM and Logistic Regression) to see which set would yield the best results. For that, activities like data collection and transformation, model training and testing, and evaluation of the final outcomes were performed. Random Forest was the best ranked algorithm according to the chosen metric, F1 Score. LGBM was also considered due to its good results in Recall, one of the most important metrics in the healthcare area.

This dissertation was carried out during an academic internship at Altice Labs, which provided all the necessary material, from access to technological tools and data. In turn, Altice Labs is working with the Centro Cirúrgico de Coimbra (CCC) that provided the necessary data for the study of automation of ophthalmologic diagnosis.

Keywords: Artificial Intelligence, Glaucoma, Machine Learning, Ophthalmology

RESUMO

Automação de Diagnósticos Oftalmológicos: Glaucoma como Caso de Estudo

Com o número de doenças oftalmológicas a aumentar a um ritmo alarmante devido ao envelhecimento da população, é importante explorar novas tecnologias com o objetivo de satisfazer as necessidades dos pacientes. Com a evolução exponencial da Inteligência Artificial, novas metodologias e modelos cognitivos têm sido desenvolvidos de forma a otimizar vários setores na nossa sociedade, entre eles a análise de dados no ramo da saúde. Ao incorporar sistemas inteligentes nos diagnósticos clínicos, é possível atingir um alto nível de integração da informação, processamento de dados e uma melhoria teórica na rapidez do diagnóstico. O objetivo do projeto apresentado é treinar e testar modelos supervisionados com dados oftalmológicos, de modo a prever e diagnosticar doenças oculares como o Glaucoma.

Para esse efeito, foi realizado um projeto de *Machine Learning*, seguindo uma metodologia típica. A metodologia é composta por fases como *Business Understanding, Data Acquisition and Transformation, Modeling e Deployment.* O documento está estruturado de acordo com essas fases e em cada uma delas será apresentado todo o trabalho realizado.

O principal objetivo desta investigação foi testar vários conjuntos de características e algoritmos (LGBM, Random Forest, KNN, SVM e Logistic Regression) para ver quais produziriam os melhores resultados. Para isso, foram realizadas atividades como a recolha e transformação de dados, treino e teste de modelos e avaliação dos resultados. Random Forest foi o algoritmo mais bem classificado de acordo com a métrica escolhida, F1 Score. LGBM foi também considerado devido aos seus bons resultados em Recall, uma das métricas mais importantes na área da saúde.

Esta dissertação foi realizada no contexto de um estágio curricular na Altice Labs, que forneceu todo o material necessário, desde acesso às ferramentas tecnológicas e dados. Por sua vez, a Altice Labs está a trabalhar com o Centro Cirúrgico de Coimbra que forneceu os dados necessários para o estudo da automação de diagnósticos oftalmológicos.

Palavras-chave: Glaucoma, Inteligência Artificial, Machine Learning, Oftalmologia

INDEX

List of abbreviations/Acronyms

AI	Artificial Intelligence
ML	Machine Learning
RGC	Retinal Ganglion Cells
ONH	Optic Nerve Head
RNFL	Retinal Nerve Fiber Layer
VF	Visual Field
DR	Diabetic Retinopathy
OCT	Optical Coherence Tomography
DL	Deep Learning
CML	Conventional Machine Learning
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
RF	Random Forest
LGBM	Light Gradient Boosting Machine
CNN	Convolutional Neural Networks
ANN	Artificial Neural Network
CCC	Centro Cirúrgico de Coimbra
IMRaD	Introduction, Methods, Results and Discussion
OCR	Optical Character Recognition
EDA	Exploratory Data Analysis

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Artificial Intelligence is taking over the world in nearly every field and has been dramatically changing our lifestyle over the past years. The health sector, as one of the most important fields, cannot be left behind. Many reasons can explain the need for technological advances in ophthalmology. With population aging becoming a demographic trend worldwide, ocular diseases are increasing at an alarming rate. Existing conventional diagnosis methods are heavily dependent on professional's experience, which can result in higher misdiagnosis. And with the amount of image data generated rapidly rising, it is more important than ever to have techniques that can analyze and process this data (Lu et al., 2018). On the other hand, the exponential growth of computing power and a better quality of ophthalmic imaging allows for better results. With the usage of ML techniques in medical diagnoses, the detection, surveillance, and treatment of ocular diseases can be improved. These techniques will also aid ophthalmologists in the detection and classification of pathological features of eye diseases. Recent research indicates that AI/ML has the potential to outperform humans in image recognition tasks (Lu et al., 2018). Incorporating AI/ML systems into clinical practice can boost productivity, aid in decision-making, and allow for image analysis automation (Valikodath, 2020).

This work emerged in the context of an existing exploratory project at Altice Labs, in collaboration with a clinical entity. Altice Labs is a company part of the Altice Group, located in Aveiro, that develops innovative products and services for the telecommunications and information technology market. In addition to having several offices in Portugal, Altice Labs has a subsidiary company operating in Brazil. For being part of a multinational corporation and due to its credibility based upon 65 years of technological experience in telecommunications, Altice Labs was a suitable option for providing me the necessary support in writing my dissertation. Altice Labs is involved in several projects, and for this project in specific, it is working with Centro Cirúrgico de Coimbra (CCC) that provided the necessary data for the study of the automation of ophthalmologic diagnosis. The data available for the project consists of medical examinations that were turned into useful data through techniques of text extraction, prior to my inclusion in the team. The existent project had the goal of applying ML models to identify which patients have characteristics that reveal glaucoma. With this, it is expected that, at a later stage, we will be able to predict that a patient will be diagnosed with Glaucoma. For that the Altice Labs team conducted an ML project where different features and different algorithms were tested to evaluate which would produce better results. Initially, there were 3 different analyses to try to distinguish the

patients: Sick/Not sick, Glaucoma/Other pathologies, Glaucoma/Not glaucoma. The team decided to proceed with Glaucoma/Not glaucoma as it was the one with better results.

My work followed on from the work of Altice Labs, I only focused on the analysis of Glaucoma/Not glaucoma. In my work, initially, it was performed research on the state of the art of ocular imaging analysis and interpretation technologies, as well as of AI/ML processes applied to the diagnosis of ocular diseases. This first phase of the project was mainly of literature review to prepare for the project's development. In a second instance, a specific scenario, among all the eye diseases mentioned in the first phase, was chosen. As we saw above it was Glaucoma and all the work done was focused on this disease. Being an ML project, it followed an appropriate methodology composed of steps such as Business Understanding, Data Acquisition and Transformation, Modeling and Deployment. All the steps of this methodology will be thoroughly analyzed during this document. The main goal of my work was to test different sets of algorithms and features to determine which would produce better results and present these results to Altice Labs. For that it was performed tasks such as Data Collection, Data Preparation and Model Training, where the model was trained and tested using the data collected from the medical examinations. With this research we hope that the diagnosis and treatment of ocular diseases can continue to improve with the help of Machine Learning techniques.

This document follows an IMRaD (Introduction, Methods, Results, and Discussion) structure, one of the most prominent formats for the structure of a scientific article. Due to the complexity of a dissertation project, other chapters were added in addition to the ones mentioned above. The document is divided into 9 sections that will be described below. The first section corresponds to the Introduction where it is presented the background and motivations for the project, its objectives, and a presentation of the company. In the second section, it was performed a Literature Review on the two main topics of the dissertation - Ophthalmology and Machine Learning. The Literature Review explores and synthetizes the key concepts of these two topics. In the third section is presented the methodology followed, the development environment and the procedure used to carry out this project. In the fourth section is described the implementation of the methodology presented in the third section. Thus, this section is sub-divided in the steps of the methodology. In the fifth section, are presented the results obtained in the fourth section. The results are presented according to the different evaluation metrics chosen during the Business Understanding stage. In the sixth section, the results obtained are discussed and analyzed. In the seventh section, are presented the conclusions obtained from this work. The eight section contains the references used for the development of the theoretical part of this document.

Lastly, the ninth section corresponds to the Appendix, where is presented artifacts that were too long to be included in the main sections, such as tables or prints from the development environment.

2. LITERATURE REVIEW

In this section, it was performed a Literature Review on the two main topics of the dissertation - Ophthalmology and Machine Learning. The Literature Review explores and synthetizes the key concepts of these two topics. Being a work of medical nature, it is important to explore the medical background first. After that, the topic of Machine Learning will be discussed.

2.1 Medical Background

In the next paragraphs will be provided an overview about the pathologies and medical examinations addressed in this study. Starting with pathologies, it will be focused on Glaucoma, Diabetic Retinopathy and Cataracts.

Glaucoma: Glaucoma is the most common cause of irreversible blindness in the world (Akkara, 2021). It is characterized by dysfunction of Retinal Ganglion Cells (RGCs), causing gradual damage to the Optic Nerve Head (ONH) and the Retinal Nerve Fiber Layer (RNFL) thickness, and a resultant loss of the Visual Field (VF). This eye condition, referred as the "silent thief of sight", is normally asymptomatic in early stages of the disease which makes it harder to be diagnosed. However, with an early diagnosis and optimal treatment it can effectively be slowed down, minimizing the risk of visual loss, hence the importance of an early diagnosis. There is no known cure for Glaucoma, but it can be controlled through eye drops, pills and, in more severe situations, with laser procedures and surgical operations. This will help to prevent or slow further damage from occurring.

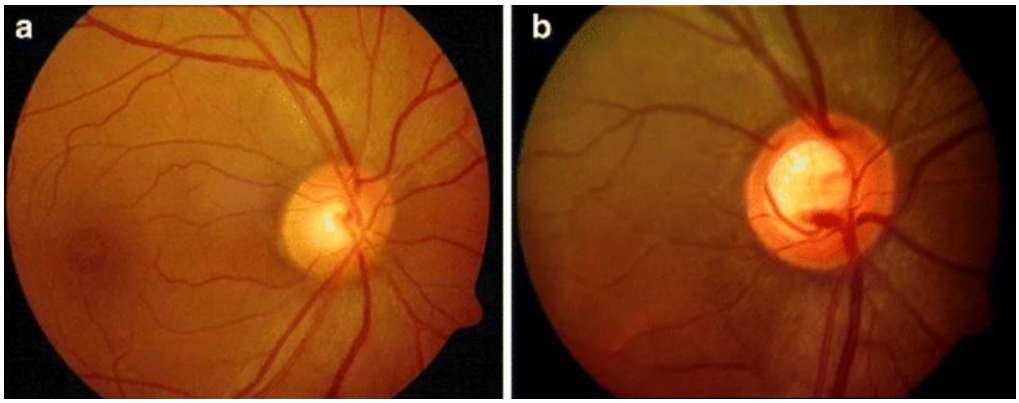


Figure 1 - Difference between a Normal Eye (left) and a Glaucomatous Eye (right) utilizing fundus image, retrieved from (Nayak et al., 2009)

Diabetic Retinopathy: Diabetic Retinopathy (DR), a complication caused by high blood sugar levels damaging the retina, is the most common and insidious complication of Diabetes Mellitus (Abramoff, 2010). Diabetes Mellitus develops when the pancreas does not produce enough insulin or the body cannot effectively process it, affecting the circulatory system and therefore the retina. When the glucose levels in the blood increase continuously, the blood vessels get severely damaged resulting in blood leak in the eyes which reduces the vision. Normally, DR takes several years to progress to the point where it threatens the vision however, if not diagnosed and treated early it can cause blindness which is incurable at advanced stages. To the present day, a cure to this pathology has not been found. Nevertheless, there are advanced treatments focused on slowing or stopping the progression of the disease that work very well. The ability to treat retinopathy symptoms relies only on early detection and early intervention. The earlier the condition is found, the easier it is to cure and better the chances that the vision will be saved. The most common treatments for retinopathy are managing the blood sugar levels, intravitreal injections and laser surgery.

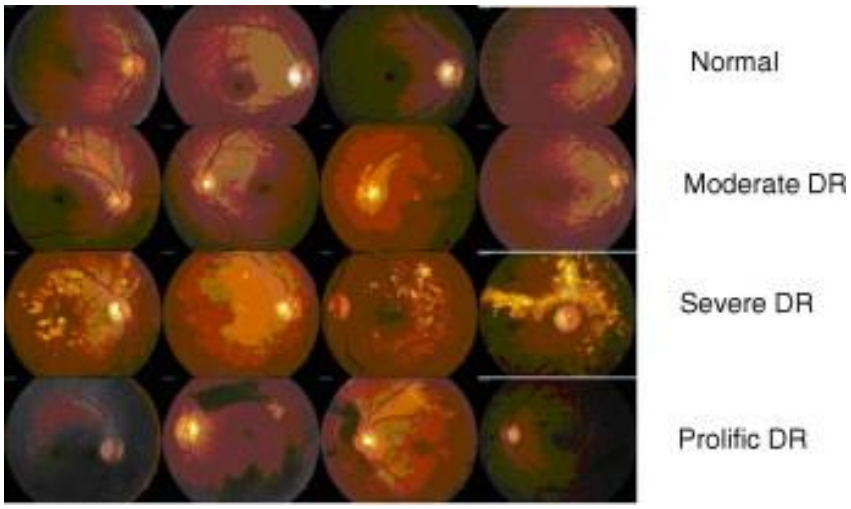


Figure 2 - Detection of DR progression utilizing fundus image, retrieved from (Yun, W. et al., 2008)

Cataracts: Cataracts is a dulling or clouding of the lens inside the eye, which is normally clear, causing symptoms such as blurry vision. For the human eye to see, light must pass through a clear lens, located behind the iris. The lens focuses the light so that the brain and the eyes can work together and process the information. If a person suffers from Cataracts, it clouds over the lens preventing the eye from focusing the light and resulting in vision loss. Vision changes depend on the Cataracts location and size. There are several factors that can cause Cataracts such as living in an area with bad air pollution or alcohol and cigarette addiction, however most Cataracts are related to age and systemic

diseases. As the world's population grows older, Cataracts are projected to become even more common. Early detection and treatment are essential for increasing patient's quality of life and lowering healthcare costs. There is no natural cure for Cataracts. Usually, the recommended medical treatments are regular eye examinations to monitor the progression of the disease, lifestyle changes, and ultimately, surgery to remove the diseased lens.

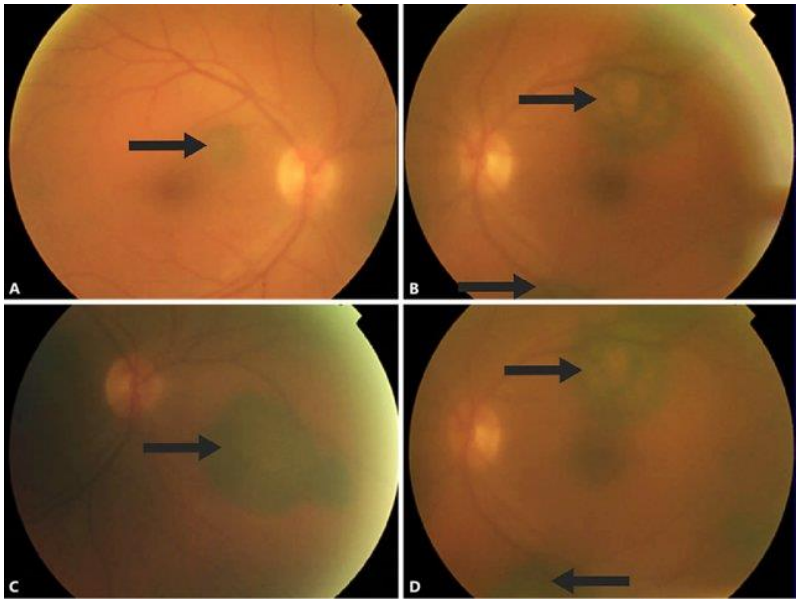


Figure 3 - Fundoscopic image of the lesions before cataract extraction in the right eye (A, C) and left eye (B, D), retrieved from (Van Noort, B. et al., 2019)

Regarding medical examinations, the ones that will be addressed on this work are Visual Field Tests, Optical Biometry, Fundus Imaging and RNFL Analysis.

Visual Field Test: Visual Field Test is an ophthalmological examination performed with a device called Campimeter. It allows to evaluate central and peripheral visual perception, identifying any change or reduction of the visual field, and quantify these same changes by comparing them in future evaluations. This test also helps in the differential diagnosis, to verify the effectiveness of some treatments and to evaluate follow-up visits, verifying the stability or the progression of the lesions. It is indicated in glaucomatous pathologies, neuro-ophthalmology, and retinal degenerations. The sequence of images presented below demonstrates the glaucoma progression through a VF test.

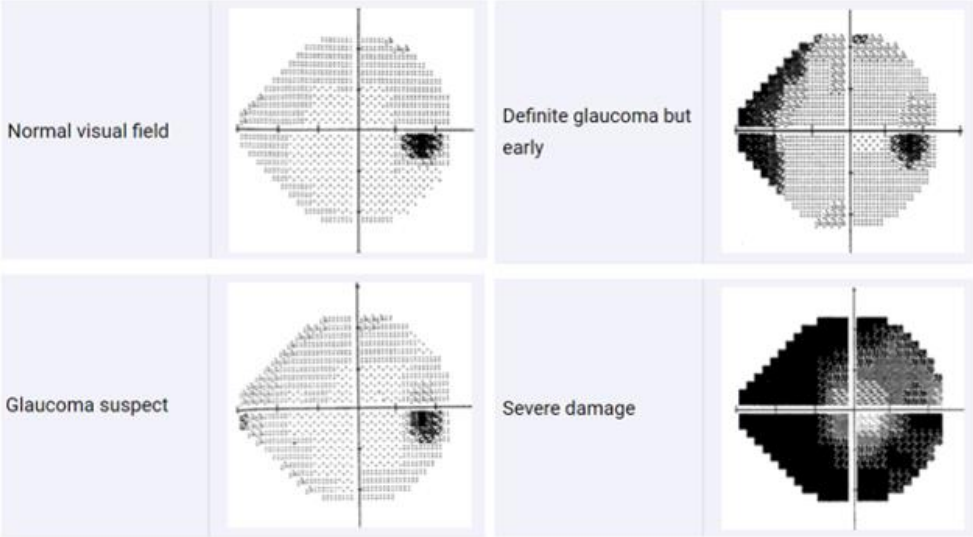


Figure 4 - Glaucoma Progression on a VF Test, retrieved from (https://www.glaucoma.org.il/glaucoma/diagnosis/vf-damage)

Optical Biometry: Optical Biometry is an ophthalmological examination performed by a technician using a device called Biometer. It is used to take measurements of the eyeball to provide data that is used for the preoperative study of various eye surgeries. It can be performed on people of all age groups, without any contraindication and it only requires the instillation of one drop of anesthetic eye drops. The patient then fixes his eyes on a point determined by the technician so that measurements can be taken with an ultrasonic probe. The probe contacts the cornea, which is anesthetized, making it a painless procedure. The most frequent situation in which biometry is used is in calculating the intraocular lens that will be implanted during Cataract surgery.

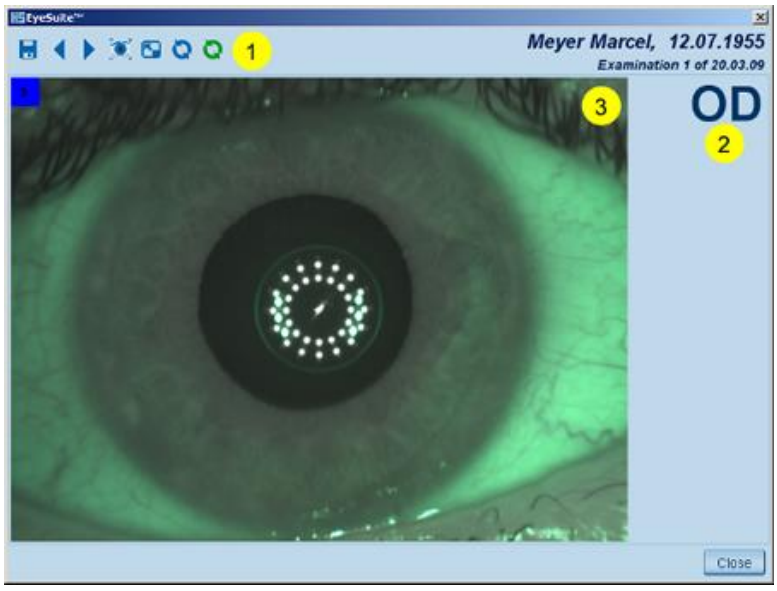


Figure 5 - Example of an Optical Biometry, retrieved from (https://www.doctor-hill.com/lenstar_haag_streit/lenstar_manual_6c.htm)

Fundus imaging: Fundus imaging is an ophthalmological examination performed with a device called fundus camera, which is a specialized low-power microscope attached to a camera used to examine structures such as the optic disc, retina, and lens. It is defined as the process of taking serial photographs of the interior of the eye through the pupil. To perform this procedure, the ophthalmologist will have to administer an eye drop to increase the size of the pupil, allowing for a more thorough examination of the eye's internal surface. Then the camera will focus and align on the pupil and create a photograph of the interior surface of the eye. This examination is most used for detection of diabetic retinopathy, glaucoma, and age-related macular degeneration on a large scale. The image below shows the typical steps required for fundus image analysis, in this case for early diabetic retinopathy.

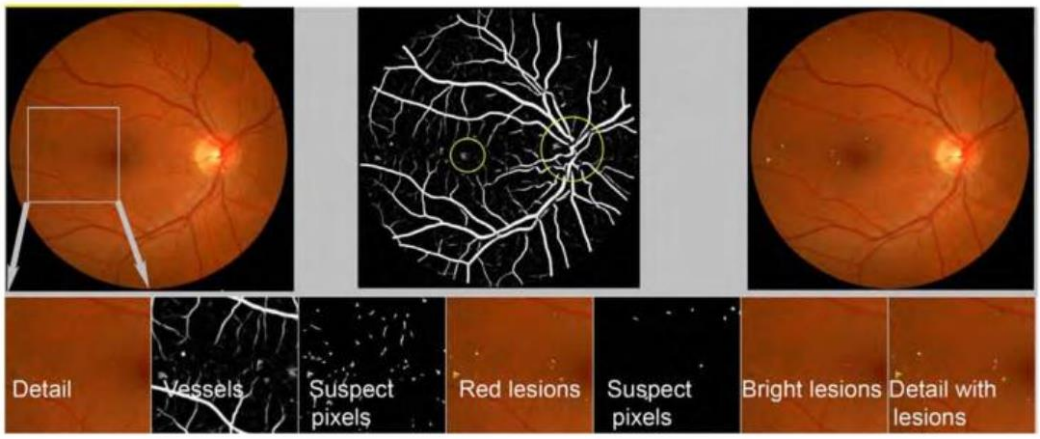


Figure 6 - Example of a Fundus Imaging, retrieved from (Abramoff et al., 2010)

RNFL Analysis: Patient's overall RNFL health is determined by the RNFL thickness. The lower the RNFL thickness value, the more it implies the patient has glaucoma. While a normal eye, without glaucoma, has a RNFL thickness of 80 microns or greater, an eye with a RNFL thickness lower than 79 is suspicious for glaucoma (Ahmed et al., 2008).

Before functional deficits in the visual fields appear, ganglion cell loss and RNFL anomalies occur. Therefore, methods to diagnose and monitor progression of optic nerve diseases are essential. RNFL analysis is a very useful tool in the treatment of glaucoma patients, having a great ability to detect early glaucoma. It can be used in combination with other medical examinations to distinguish between glaucoma and other conditions such as myopia and physiologic cupping. Optical Coherence Tomography (OCT) is a diagnostic imaging technology that provides in vivo images of RNFL thickness. It is a computer-aided precision optical tool that analyzes the retinal structure using high-resolution tomographic cross-sections of the retina. This technique allows the visualization of cross-sections of the

retina and has already demonstrated the ability to detect changes in tissue thickness with large sensitivity.

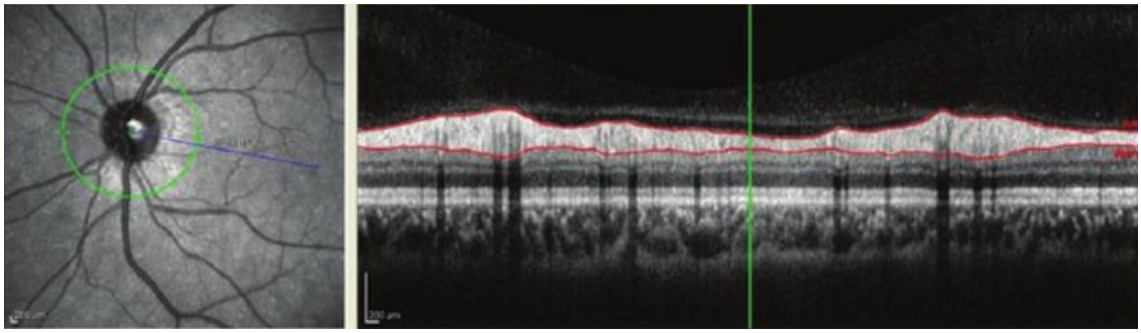


Figure 7 - Example of a RNFL Analysis, retrieved from (Ataş et al., 2014)

2.2 Machine Learning

In this dissertation two main topics are addressed - Machine Learning and Ophthalmology. These two topics have not always been related. It was not so long ago that ML techniques were integrated into medicine with the goal of improving it. This has only been possible due to the technological advances over the last years, and we are now able to apply the advantages of AI/ML to the field of healthcare which has become one of the forefronts of the AI application.

In the first place, before approaching Machine Learning, we must discuss Artificial Intelligence first. **Artificial Intelligence** is an umbrella term that refers to the field of computer science that can mimic cognitive functions such as learning and problem solving. It includes technologies like Computer Vision, Language Processing and Machine Learning.

In turn, **Machine Learning** is a subfield of Artificial Intelligence that aims to automatically detect patterns in data and then incorporate this information to predict future data. Data is used as an input to predict new output values. Machine Learning includes Conventional Machine Learning (CML) and Deep Learning (DL).

Conventional Machine Learning can obtain good performance with small datasets however, due to the manual features selection process, in bigger datasets these techniques tend to overfit, that is, when the model is good at classifying the samples in the training dataset but does not get satisfactory results when it tries to classify new data, limiting their application. Existing CML algorithms used in AI diagnosis include Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Linear Regression,

Logistic Regression and Random Forest (RF). Among them, Random Forest and Support Vector Machine are the most used CML technologies in the ophthalmology field (Lu et al., 2018).

Deep Learning is a ML technology that is rapidly gaining popularity. It simulates the activity of the human brain by using multiple layers of artificial neural networks (ANNs) capable of producing automated predictions based on input data. DL has the ability to discover complicated structures in datasets without having to explicitly describe the rules. One of the main differences between DL and CML is how their algorithms select and process image features. DL algorithms have the advantage of being more efficient because the features of input data are automatically learned in an unsupervised way without the need to manually select them, as demonstrated in the figure below. However, they have a significant disadvantage, especially in the medical context. DL algorithms are called "black boxes" because the networks generate comprehensive features that are too complex to be interpreted by the human mind. These algorithms can achieve better performances but the way they analyze patterns and make decisions is unexplainable, which is not feasible in the healthcare field. Convolutional Neural Networks (CNNs) is the most used Deep Learning method in image recognition (Lu et al., 2020).

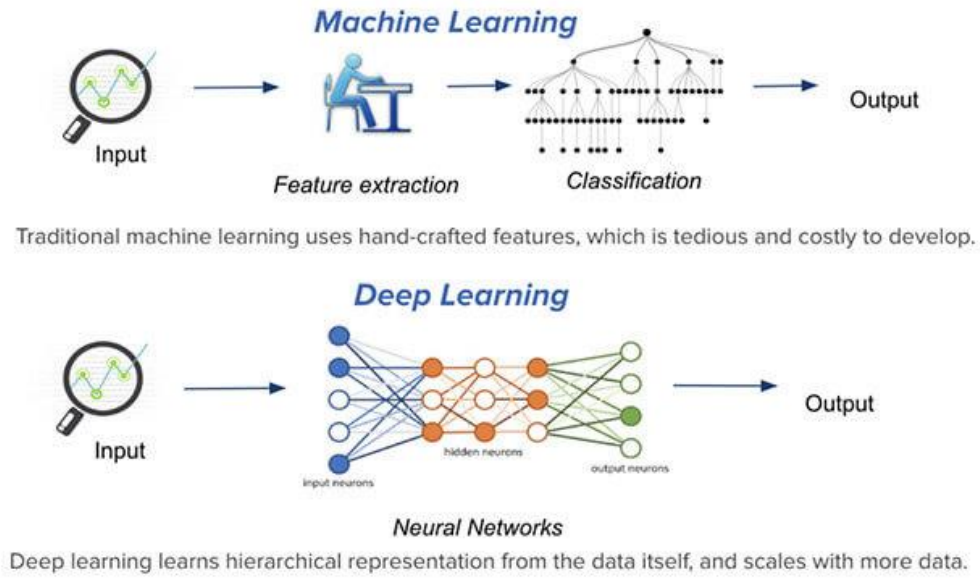


Figure 8 - Differences between CML and DL, retrieved from (https://www.merkleinc.com/blog/dispelling-myths-deep-learning-vs-machine-learning)

A machine learning model is a file that has been trained to identify specific sorts of patterns. All Machine Learning models can be classified as either supervised or unsupervised.

Supervised Learning involves training a model with input data previously labeled by experts to predict the desired outcome, thus requiring a training dataset. The experts go through the dataset and label the data with the correct information, known as ground truth. The labeled data is given to the machine which uses a learning structure to identify the correct prediction. Supervised learning is used to solve Classification and Regression problems and can be useful for discriminating clinical outcomes. One of the disadvantages of this approach is that it is time-consuming since it involves manually labeling a large volume of data. It is most used with the algorithms Random Forest, Support Vector Machine and Convolutional Neural Networks.

On the other hand, **Unsupervised Learning** is training a model with unlabeled data, that is, data which is not labeled manually by humans. A function is then inferred to discover hidden structures that normally are not detected by humans. This approach is useful because it can lead to new discoveries. Unsupervised Learning is used for tasks of Clustering and anomaly detection and is most used with algorithms like Autoencoders. A hybrid approach between Supervised and Unsupervised Learning is also possible called Semi Supervised Learning.

According to most research studies, supervised methods are more used because they can achieve better results (Lu et al., 2018).

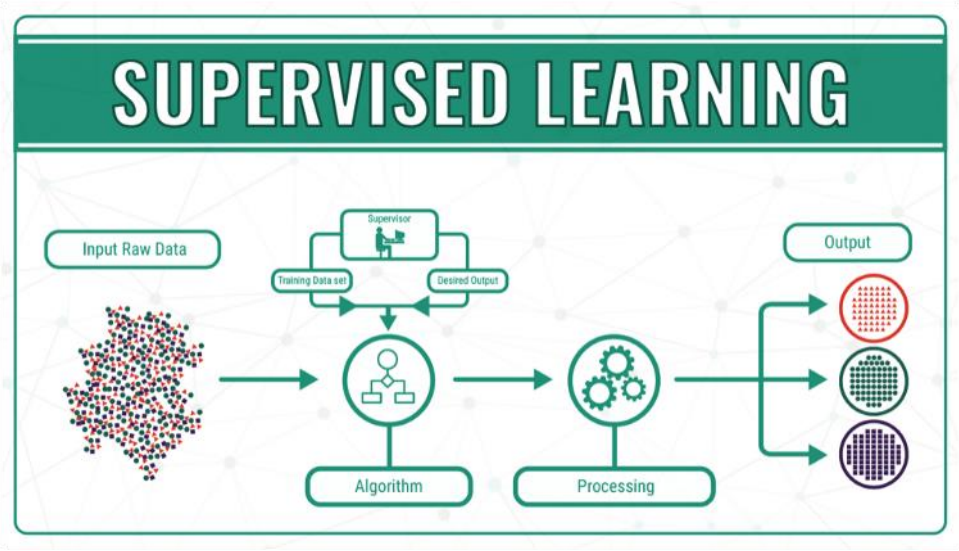


Figure 9 - Workflow of a Supervised Learning Model, retrieved from (https://becominghuman.ai/supervised-learning-vs-unsupervised-learning-8af1bc803210)

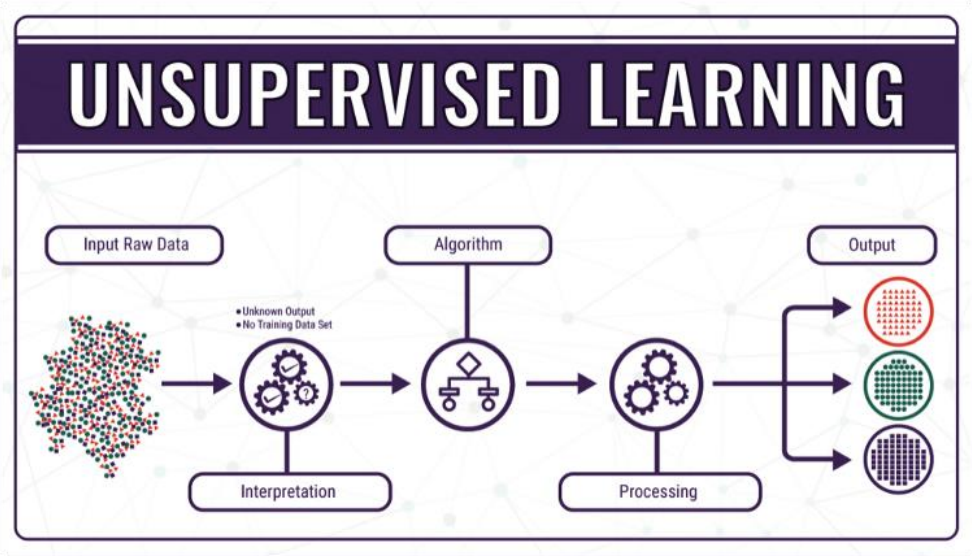


Figure 10 - Workflow of an Unsupervised Learning Model, retrieved from (https://becominghuman.ai/supervised-learning-vs-unsupervised-learning-8af1bc803210)

After the machine learning models have been classified as supervised or unsupervised, they can also be subclassified as either a Classification or Regression model if they are supervised or a Clustering model if they are unsupervised.

Classification is the process where a machine utilizes an algorithm to generate conclusions from existing data, and then uses these conclusions to predict what the new data will be. The data is categorized and previously labeled and then the labels are predicted for that data. One example of classification is spam detection in email. A classifier utilizes training data, in this case emails, to describe if they are spam or not. The classifier algorithm learns from this information, and creates a model based on what has been learnt. Then, the model can be used to detect if an unknown email is spam or not. Classification is widely used in medical diagnosis, target marketing and credit approval.

Just like Classification, **Regression** is also a supervised learning technique, but while Classification is about predicting a label, Regression is about predicting continuous values. Regression analysis consists of a set of Machine Learning methods to model the relationship between dependent and independent variables. Dependent variables are also known as targets and are the ones we want to predict. Independent variables are also known as predictors and are the factors which affect the dependent variables. This approach is mainly used for predictions and forecasting. A well-known example is the prediction of housing prices, where some features about the house are known, like size and location, and then the algorithms predict the price of the property.

As opposed to the two techniques previously reviewed, **Clustering** is an unsupervised learning technique. Unlike supervised methods, Clustering works with unlabeled data and it involves automatically discovering natural grouping in data, which is accomplished by partitioning the dataset into groups called clusters. A cluster is a group of objects belonging to the same class. Those with similar characteristics are grouped together in one cluster, while objects with distinct characteristics are grouped in another. Clustering algorithms are useful to find patterns in data that initially are undetectable to the human eye. This approach is mostly used in market and customer segmentation, social network analysis and recommendation engines.

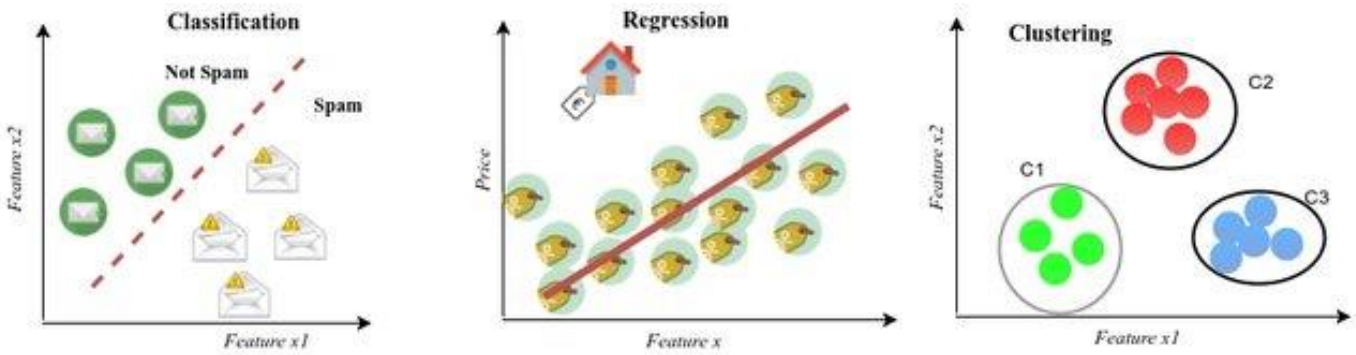


Figure 11 - Different types of Machine Learning Models, retrieved from (Kaplan, 2017)

The development of a Machine Learning model includes multiple steps such as Data Preprocessing, Train, Validate and Test the model and finally Evaluate the trained model's performance.

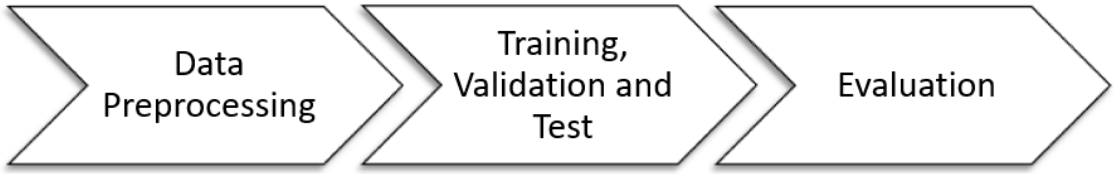


Figure 12 - Steps for building a ML model

Data Preprocessing: Data Preprocessing is a fundamental step in Machine Learning because raw data must be preprocessed to improve AI prediction efficiency. Several preprocessing steps can be performed, such as data cleaning, data integration and normalization, and feature selection. Data cleaning is the process of reviewing the data to remove duplicates, fill missing values and resolve

inconsistencies that may exist. In data integration and normalization, original data collected from diverse sources should be integrated and adapted to a common scale appropriate for thorough comparison evaluation. In feature selection, to optimize the learning process performance, the most important features are chosen. Due to their heterogeneous origin, most data is inconsistent and noisy, therefore it is crucial that data is preprocessed before being fed to the model. The quality of data directly affects the ability of the model to learn.

Training, Validation, and Test: To build a reliable ML model and to achieve the best performance possible, the base dataset must be randomly partitioned into two subsets, one to build the model and the other to test the model's performance. The first dataset is further split into training dataset and validation dataset.

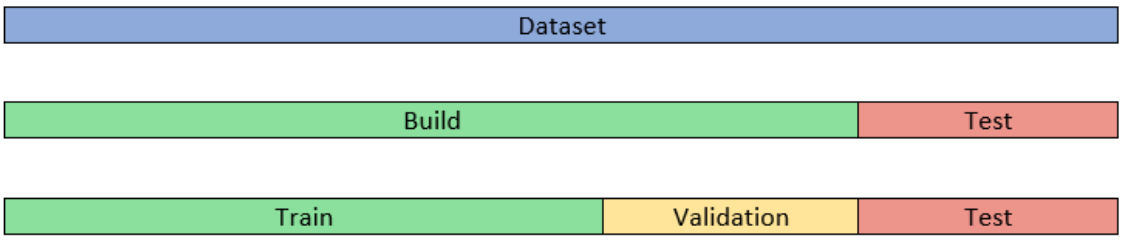


Figure 13 - Partitions of the original Dataset into Training, Validation and Test subsets

The training dataset is used to fit the parameters of the model and make it learn the hidden features in the data. This set should be as much diversified as possible so that the model is trained to predict any given scenario. The validation dataset is used for parameter selection and to tune the model's configurations according to those parameters. After the model had been trained in the previous step, the model's performance is validated to define how well the model was trained. The goal of this step is to prevent the model from overfitting. Several validation techniques can help to estimate unbiased generalized model performances and give a better understanding of how the model was trained. It makes sure that the model is accurately trained and that it outputs the right data.

There are several validation methods such as, Train and Test Split, and K-Fold Cross-Validation. In Train and Test Split, the data is split into training data and testing data, holding back the testing data to not expose the model to it, until it is time to test the model. The percentage of data intended to the training data and testing data can be decided at the time of splitting. K-Fold Cross-Validation is similar to the Train and Test Split, except that it splits the data into more than two groups. In this validation method, "K" is used as a placeholder for the number of groups the data will be split into. For example, in the

figure below, the data was split into 3 groups. One group is left out of the training data and the model is validated using the group that was left out of the training data. Then, the model is cross validated. Each of the two other groups used as training data are then also used to test the model. Each test and score can give new information about what is working and what is not in your machine learning model.

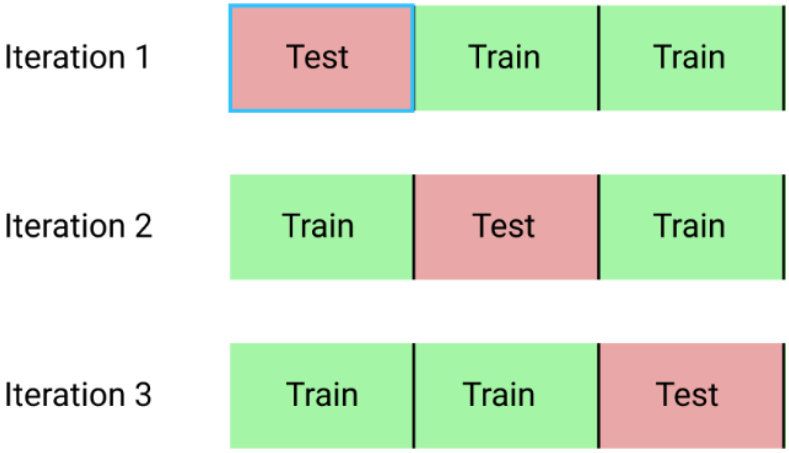


Figure 14 - Example of K-Fold Cross-Validation

Finally, the test dataset is a separate set of data from training and validation, that is used to evaluate the performance of the trained model. Since it is an independent set, it provides an unbiased final model performance.

Evaluation: After building the best learning model possible, it is required to perform an evaluation of how well the model is performing and if it is good enough to move forward. When the model performs classification predictions, there are four possible outcomes:

- True Positives (TP): the total number of correct predictions when the actual class was positive.
- True Negatives (TN): the total number of correct predictions when the actual class was negative.
- False Positives (FP): the total number of wrong predictions when the actual class was negative.
- False Negatives (FN): the total number of wrong predictions when the actual class was positive.

For better organization and interpretability of the possible outcomes, a Confusion Matrix can be created. Confusion Matrix is a table that allows visualization of the performance of an algorithm. Each row in the matrix represents an actual class, while each column represents a predicted class. It compares the actual target values with those predicted by the Machine Learning model. The number of correct and incorrect predictions are summarized and divided by each class.

Predicted class / Actual class	P	N
P	TP	FN
N	FP	TN

Figure 15 - Confusion Matrix

From this outcome, it is possible to evaluate the model using several performance metrics:

Accuracy: the proportion of positives and negatives that are correctly identified, it represents the ability of the model to correctly identify patients who have the disease and those who do not. The higher the accuracy, the better the classifier.

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

Sensitivity/Recall: the proportion of positives that are correctly identified, also known as the True Positive Rate. It represents the ability of the model to correctly identify patients who have the disease.

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity: the proportion of negatives that are correctly identified, also known as the True Negative Rate. It represents the ability of the model to correctly identify patients who do not have the disease.

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

Precision: the number of true positives divided by the total number of positive predictions. It represents the proportion of positives that are correctly identified among all positive identified samples. The higher the precision, the lower the false positive rates.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

F1 score: the harmonic average of the precision and sensitivity, it represents how precise the classifier is, as well as how robust it is. The range for F1 Score is [0, 1] where it reaches its best value at 1 and worst at 0. The higher the F1 Score, the better the performance of the model.

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Likelihood ratio for positive test results: the probability that a positive test would be expected in a patient divided by the probability that a positive test would be expected in a patient without a disease, it represents the probability of a positive test result occurring compared to a negative one.

$$LR+ = \frac{Sensitivity}{1 - Specificity}$$

Likelihood ratio for negative test results: the probability of a patient testing negative who has a disease divided by the probability of a patient testing negative who does not have a disease, it represents the probability of a negative test result occurring compared to a positive one.

$$LR- = \frac{1 - Sensitivity}{Specificity}$$

Receiver Operating Characteristic (ROC) curve: ROC curve is a graph that indicates the performance of a classification model at all classification thresholds. The graph plots two parameters, the True Positive Rate (TPR) and the False Positive Rate (FPR). It shows the trade-off between sensitivity (TPR) and specificity (1 – FPR). The closer the ROC curve is to the upper left corner, the higher the AUC value and the better the performance of the model.

Area under the curve (AUC): AUC measures the entire area underneath the ROC curve. It can have any value between 0 and 1 and it is a good indicator of the model's quality. The AUC indicator measures the accuracy of positive and negative samples at the same time. It tells how much the model is capable of distinguishing between classes, which means the higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

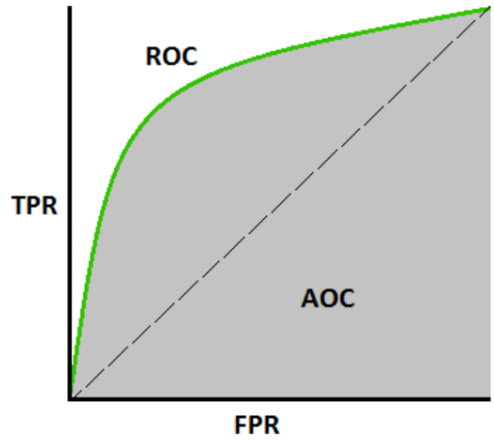


Figure 16 - Example of the ROC curve and AUC, retrieved from (https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5)

Machine Learning Algorithms

Decision Tree: Decision tree is a type of supervised learning algorithm that is mostly used for classification problems. It can be subclassified as Classification Trees, when the variables are categorical, and Regression Trees, when the variables are continuous. It is a tree-like model in which each node represents a test on an attribute and each branch represents the outcome of the test. The root node represents the entire population and is further split into two or more homogeneous sets, based on the most important attributes. Decision trees do not require any statistical knowledge and are very intuitive to interpret, making it one of the most used algorithms. On the other hand, it is an algorithm with a high tendency to overfit and it is not fit for continuous variables.

Random Forest: Random Forest is a supervised machine learning algorithm that is constructed from many individual decision trees. It utilizes ensemble learning, a technique that combines multiple learning algorithms to achieve better predictive performance than could have obtained from any of the individual learning algorithms alone. Each individual tree in the random forest makes its own prediction and the final prediction is done by taking the average of the output from the various trees. The benefit of this algorithm is that although some of the trees may be wrong, many other trees will be right, so together the trees have more chances of being accurate. The precision of the outcome improves as the number of trees grows. Random Forest algorithms solve some of the limitations of decision trees. It reduces the overfitting of datasets and increases precision.

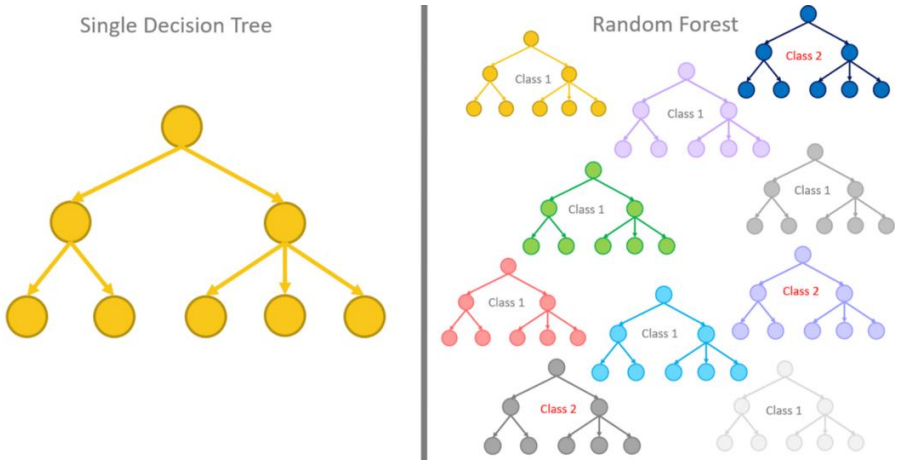


Figure 17 - Single Decision Tree vs Random Forest, retrieved from (https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147)

Support Vector Machine: Support Vector Machine is a supervised learning algorithm that can be used for regression and classification but is mostly used in classification problems. In this algorithm, each data item is plotted as a point in an N-dimensional space, where N is the number of features you have, with the value of each feature being the value of a particular coordinate. Then, the objective is to find a hyperplane that best differentiates the classes. Support Vectors are the coordinates of each individual observation, and the best hyperplane is the one whose distance to the nearest element of each class is the largest, as demonstrated in the figure below. SVMs algorithms can achieve good values of accuracy with less computation power.

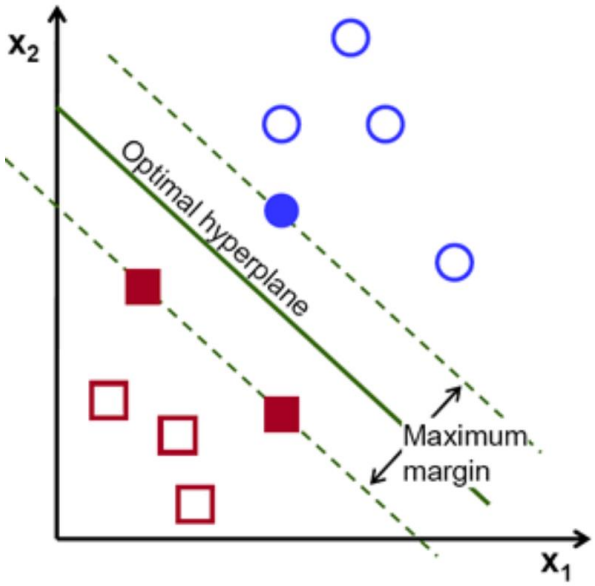


Figure 18 - The Optimal Hyperplane in a SVM Algorithm, retrieved from (https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47)

K-Nearest Neighbor: K-Nearest Neighbor is a supervised learning technique that can be used for both classification and regression but is mostly used in classification tasks. KNN works by finding the Euclidean distance between a new data point and all its K neighbors. Given a new data point, the first step of this algorithm is to define the number K of the neighbors we intend to use. Then we calculate the distance between the new data point and its K number of neighbors. We count the number of data points in each category among these K neighbors and designate the new data point to the category with the highest number of neighbors. To identify the best K number, the algorithm is executed several times with different values and the K that reduces the greatest number of errors while maintaining the ability to accurately make predictions is selected. This algorithm is simple to understand and easy to implement. It is also robust when the training data is noisy and large. In return, the computation cost is high due to calculating the distance between the data points for all the training samples and the need to always determine the value of K might sometimes be complex.

To more easily understand this algorithm an example is shown below:

As illustrated in the image below, assume we have a new data point, and we want to put it in the required category.

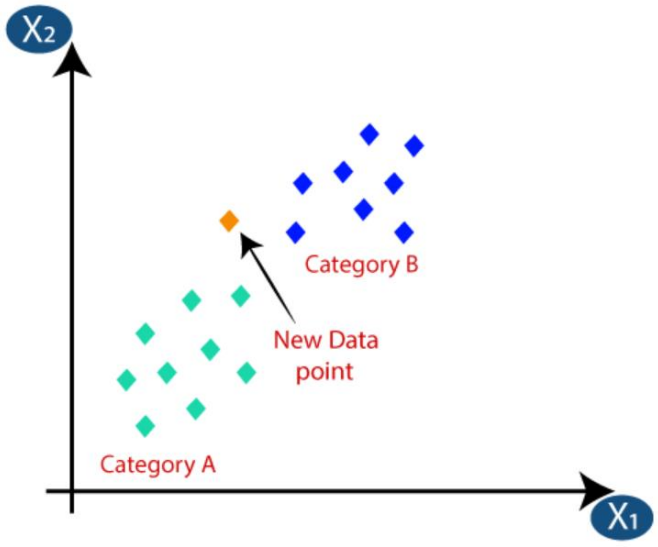


Figure 19 - New Data Point in a KNN Algorithm, retrieved from (https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

As said before, the first step is to choose the number of K neighbors. For this example, we will choose the k=5. Next, it is calculated the Euclidean distance between the data points, using the formula:

$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$$

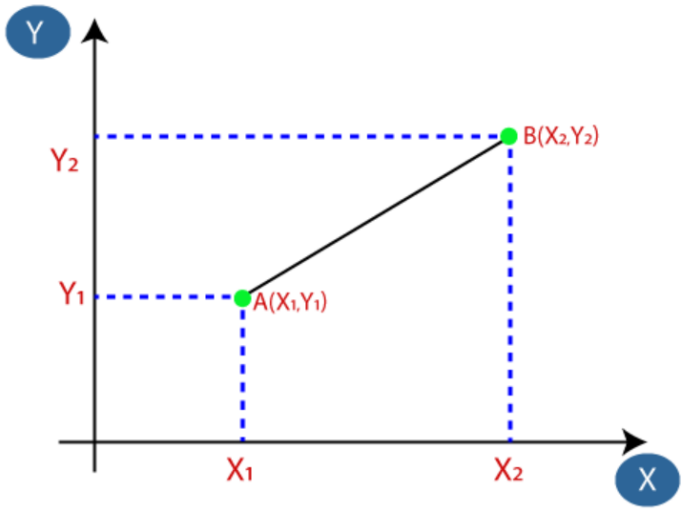


Figure 20 - Euclidean Distance between two points in a KNN Algorithm, retrieved from (https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

After calculating the Euclidean distance, we now know that we have three nearest neighbors in category A and two nearest neighbors in category B, as illustrated in the image below.

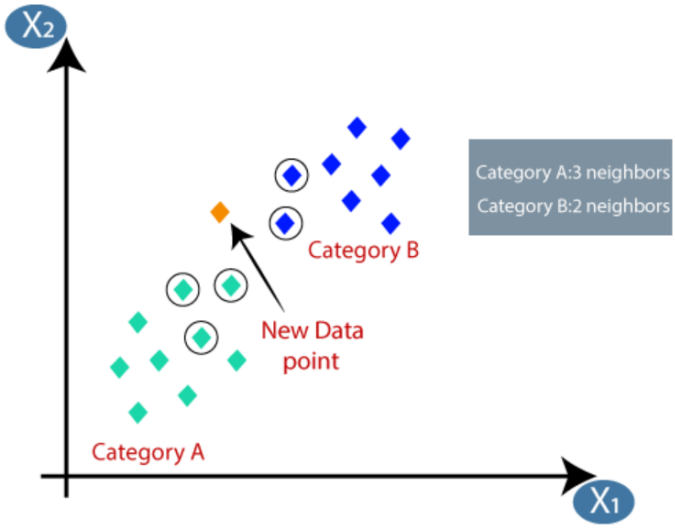


Figure 21 - Results of a KNN Algorithm, retrieved from (https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

Therefore, it is safe to assume that the new data point must belong to category A.

Linear Regression: Linear Regression is a simple supervised learning algorithm used for regression tasks. It predicts values within a continuous range rather than to classifying them into categories. It establishes a relationship between independent and dependent variables by fitting a best line, also

known as regression line. If there is only one independent variable it is called simple linear regression, and if there is more than one independent variable it is called multiple linear regression. Linear regression is a linear model that describes the relationship between the independent variables (input) and the dependent variables (output). It is an algorithm very easy to interpret, to implement and to train but, on the other hand, it is quite sensitive to outliers.

Logistic Regression: Despite having "regression" in the name, Logistic Regression is a classification algorithm. It is used to predict the probability of a target variable, thus having its output values between 0 and 1. Generally, the nature of the target is binary, which means the dependent variable will have only two possible types. But Logistic Regression can also be Multinomial, where the dependent variable can have three or more possible unordered types, and Ordinal, where the dependent variable can have three or more possible ordered types. Although it is easy to implement and to interpret, the major limitation of this algorithm is the assumption of linearity between the dependent and the independent variables.

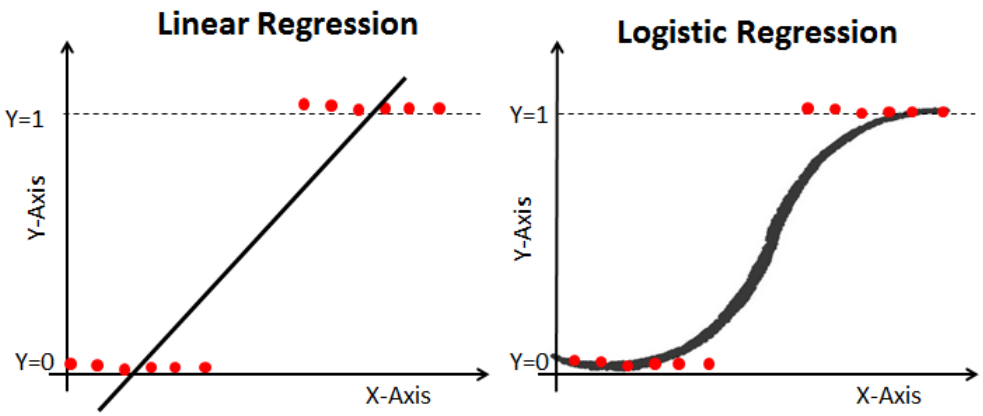


Figure 22 - Linear Regression vs Logistic Regression, retrieved from (https://medium.com/@mvanshika25/logistic-regression-ee47cc89345f)

Light Gradient Boosting Machine: Light Gradient Boosting Machine (LightGBM or LGBM) is a gradient boosting framework based on decision trees and is used for ranking, classification, and many other Machine Learning tasks. Despite being based on tree-based algorithms, LGBM grows trees vertically while other algorithms grow trees horizontally, meaning that LGBM grows tree leaf-wise while other algorithms grow level-wise. Leaf-wise algorithms are capable of reduce more loss than level-wise algorithms. These two approaches can be visualized in the figure below.

Due to the ongoing increase of data size, it is becoming harder for traditional ML algorithms to give faster results. LightGBM has advantage over these algorithms due to its high speed, can handle larger sizes of data and takes lower memory to run. Despite all these advantages, it is not advisable to be used on small datasets, since LGBM is sensitive to overfitting and can easily overfit small data.

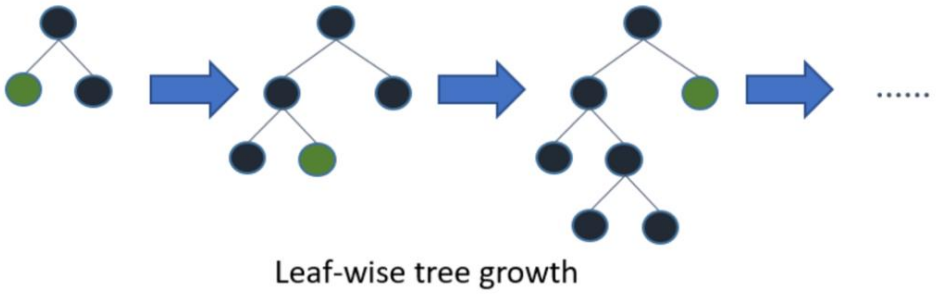


Figure 23 - How LGBM trees work, retrieved from (https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc)

2.3 Machine Learning Application in Ophthalmology

After addressing these two topics individually, it is now necessary to discuss them as a whole. As discussed in the medical background, most ocular diseases can be prevented if diagnosed on time. However, conventional diagnostic methods are failing to meet patients' needs, hence the need for other alternatives. The exponential rise of technologies that we have witnessed in recent years has led to a growth in computational power, resulting in advances in Machine Learning. The goal of this dissertation is to apply these technological advances to ophthalmology so that we can improve the diagnosis and treatment of ocular diseases.

Incorporating AI systems into clinical practice can bring many advantages such as enhancing workplace productivity, as well as aid in the clinical decision-making process (Topol, 2019). When comparing the performance between AI systems and human ophthalmologists doing a medical evaluation, AI systems have the potential to achieve much better information integration, data processing, and diagnostic speed. However, these intelligent systems should not be seen as replacements of the human diagnostic, but rather as a tool to help to improve them. There is a general fear that the evolution of AI

will lead to increases in the unemployment rate. Despite that might happen in other fields of study, in healthcare these systems should always be seen as an addition. It can even be argued that the introduction of AI can increase the work of physicians by serving as a diagnostic tool generating predictions that can positively affect patient management (Roach, 2017). In a study done by Ting, it was designed an AI-integrated platform to screen patients with cataracts. In addition to showing good results of diagnostic performance, the platform improved physician efficiency by allowing them to evaluate ten times as many patients a year (Ting et al., 2019). Ophthalmologists and doctors, in general, need to work with engineers and developers to gain a greater understanding of the abilities of technology and embrace Artificial Intelligence in their diagnoses to improve them. With this cross-disciplinary collaboration, in the future, the inclusion of intelligence within ophthalmic devices may improve the quality of patient care, for example, by embedding AI systems within ophthalmic imaging devices for real-time image diagnosis, like portable fundus cameras or smartphones, with little operator expertise. This will allow for greater scrutiny of diagnostics since patients will be self-screened before an ophthalmologist appointment and patients in remote areas could receive routine eye examinations and undergo monitoring of disease progression without the intervention of highly skilled operators (Lu, et al., 2020).

Although AI techniques have proven to be an effective diagnostic tool and the accuracy of the ML models is very promising, we need to remain prudent when considering how to deploy these models to the real world. Most ML models are developed to focus on binary classification problems, which means it can only distinguish if a patient has a determined disease or not, having difficulties when considering other diseases. In a study done by Choi to automatically detect multiple different retinal diseases with fundus photographs, when only normal and Diabetic Retinopathy fundus images were used, the accuracy of the model was 87.4%. However, when all 10 categories were included, it dropped to 30.5% (Choi et al., 2017). The need for a massive volume of data is still one of the most fundamental issues. Although many data sets are available, they only cover a small portion of the diseases. To optimally train the models, larger data sets from larger patient cohorts under different settings and conditions are needed. The “black box” problem, already mentioned above, is another issue that has been identified as an obstacle to the application of Deep Learning in healthcare. Although DL algorithms can achieve better performances, the way they analyze patterns and make decisions is unexplainable, which is not feasible in the healthcare field. This way, increasing the interpretability of networks will be another important research direction.

Case Study: Glaucoma

The prevalence of glaucoma is projected to increase by almost 50% over the next 20 years as the global population lives longer (Tham et al., 2014). Although the number of ophthalmologists is increasing, the population aged over 60 years is growing at almost twice the rate (Resnikoff et al., 2012). As a result, the number of ophthalmologists will not be enough to address this gap between supply and demand, hence the need to look for other techniques that are more innovative, technological, and capable. AI systems might represent a potential solution to this growing demand and has already demonstrated that has the potential to develop effective glaucoma screening to identify undiagnosed glaucoma as well as detect glaucoma progression in existing patients.

Glaucoma patients suffer from high intraocular pressure, damage of the optic nerve head, RNFL defect, and gradual vision loss, so automatically detect and monitor features related to glaucoma has major importance for a proper workup. In this case study, we will go through literature and analyze some of the state-of-the-art technologies regarding these glaucoma characteristics.

Optic Disc Photography: For fundus photographs, Li evaluated a DL algorithm that showed a sensitivity of 95.6% and a specificity of 92% to detect Glaucomatous Optic Neuropathy. However, high myopia caused false negatives and physiological cupping caused false positives (Li et al., 2018). Al-Aswad also evaluated a DL system to detect Glaucomatous Optic Neuropathy from color fundus photographs. His system outperformed 5 out of 6 ophthalmologists in the study (Al-Aswad et al., 2019). Cerentini developed an automatic classification method for detecting glaucoma in fundus images using GoogLeNet (Cerentini et al., 2017). Haleem utilized a novel technique for automatic boundary detection of the optic disc and cup to aid in automatic glaucoma diagnosis from fundus images (Haleem et al., 2017).

Optical Coherence Tomography (OCT): Asaoka evaluated a DL algorithm with pre-training that diagnosed glaucoma based on macular OCT for RNFL and RGC (Asaoka et al., 2019). Muhammad showed a hybrid Deep Learning method that had 93.1% sensitivity in detecting glaucoma suspects (Muhammad et al., 2017). A study done by Christopher evaluated unsupervised machine learning, artificial neural networks, and support vector machines for glaucoma OCT (Christopher et al., 2018).

Visual Fields: Li evaluated a Convolutional Neural Network to automatically differentiate Glaucoma VF from non-Glaucoma VF (Li et al., 2018). Although there are still few studies using DL methods to diagnose glaucoma, Asaoka used Visual Fields to construct DL-based glaucomatous diagnostic models.

In the same study it was stated that pre-perimetric open-angle glaucoma eyes can be detected through DL with better performance than those got from CML techniques (Asaoka et al., 2016).

Machine Learning algorithms are progressively becoming better, and in the future, it is expected to obtain models with both good sensitivity and specificity, being able to outperform human ophthalmologists. Combining the results of the several glaucoma evaluation tests will be the way to go to achieve a perfect diagnosis (Akkara, 2022).

3. MATERIALS AND METHODS

In this section is presented the methodology followed, the development environment and the procedure used to carry out this project. As I was integrated in the Altice Labs team for the realization of this project, all the tools used in the development environment, as well as the methodology, were the same tools and methodology used for the realization of their projects.

3.1 Methodology

Machine Learning projects are inherently agile and flexible, requiring constant back and forth interactions across multiple stages. A value chain with many diverse stakeholders is required for a successful project implementation. In this agile environment, leveraging business value from AI/ML technologies requires the specification of a cognitive process methodology, a set of reference processes across the organization for enabling the successful implementation of cognitive projects.

The following methodology provides an overview of the lifecycle of a cognitive project. It contains the stages of the project, their respective tasks, and relationships between these tasks. Below, all the tasks are addressed merely to explain the methodology, however, some of them are not covered in the Implementation stage as they are not necessary for the project. The main stages that are typically executed in a cognitive project are Business Understanding stage, Data Acquisition and Transformation stage, Modeling stage and Deployment stage. In Figure 2, an illustrative representation of the cognitive process methodology is presented.

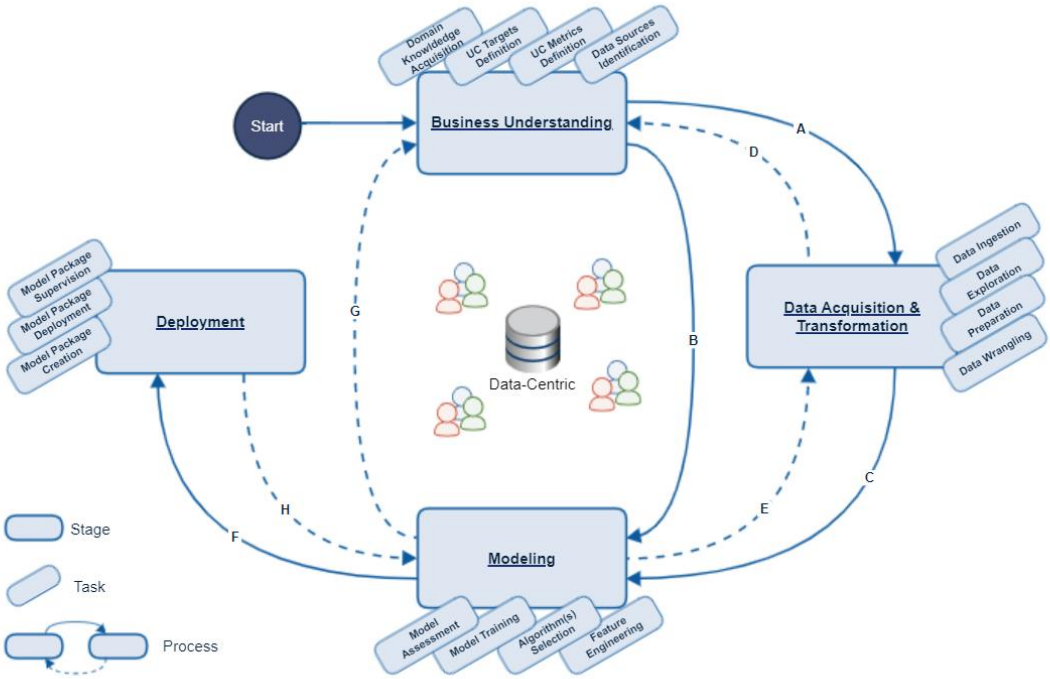


Figure 24 - Stages, Tasks and Processes of the methodology used, retrieved from Altice Labs internal website

Stage 1: Business Understanding

Objective:

- Acquire knowledge about the problem domain and specify the cognitive use-case in its several dimensions

Tasks:

- Domain Knowledge Acquisition Task: Understand the use-case objectives and requirements from a business perspective
- UC Targets Definition Task: Convert the acquired knowledge into a cognition problem definition
- UC Metrics Definition Task: Define the metrics to evaluate the cognition problem results, as well as the functional and non-functional requirements
- Data Sources Identification Task: Identify and describe the required data sources to execute the use-case

Stage 2: Data Acquisition and Transformation

Objective:

- Manage the complete data lifecycle (ingest, explore, prepare, transform/wrangle)

Tasks:

- Data Ingestion Task: Data sources integration and collection
- Data Exploration Task: Data understanding during the whole life of the stage
- Data Preparation Task: Attributes selection, cleaning and restructuring towards the final dataset format
- Data Wrangling Task: Data encoding, normalization, and transformation towards the final dataset format

Stage 3: Modeling

Objective:

- Select, apply, and calibrate several modeling techniques to achieve optimal values

Tasks:

- Features Engineering Task: Use-case features identification
- Algorithm Selection Task: Select the most appropriated algorithms for training
- Model Training Task: Apply selected algorithms for training the model
- Model Assessment Task: Evaluate and calibrate the algorithm to achieve optimal values

Stage 4: Deployment

Objective:

- Deploy the model in a production environment

Main tasks:

- Model Package Creation Task: Create the model package artifacts
- Model Package Deployment Task: Deploy the model
- Model Package Supervision Task: Supervise the model

3.2 Development Environment

In this section will be presented, with a brief description, the development environment with the tools used during the project. All the tools were suggested and approved by the Altice Labs team.

Microsoft Teams

Microsoft Teams is a collaboration app built for hybrid work that provides a platform for setting up group conversations, video calls and file sharing. It is the communication tool used in Altice Labs.

Jupyter Lab

Jupyter Lab is a web-based interactive development environment for notebooks, code, and data. Its flexible interface and modular design allows users to configure and arrange workflows in Data Science and Machine Learning, with extensions to expand functionality. All the programming in this dissertation, since Data Acquisition and Transformation to Modeling, was done through Jupyter Notebooks in Jupyter Lab.

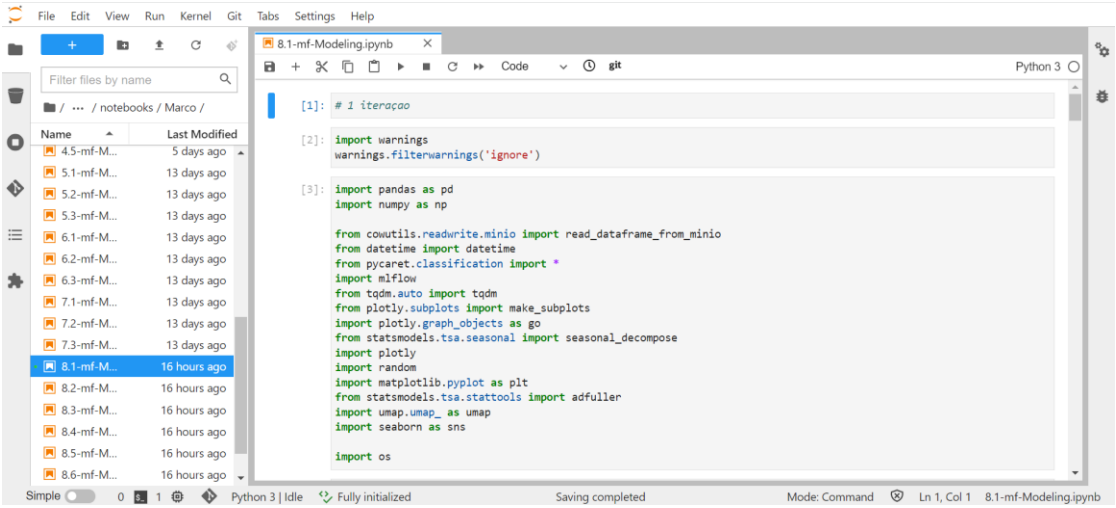


Figure 25 - Interface of a notebook on Jupyter Lab

MLflow

MLflow is a platform to streamline Machine Learning development, including tracking experiments, packaging code into reproducible runs, and sharing and deploying. As it can be seen in the image below, MLflow shows some information like start time, duration and run name, as well as the metrics needed to evaluate the model. It was used to get a more organized view of the results by aggregating them all in one place.

				Metrics					
	↑ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2_	recall_score	roc_auc_score
	1 month ago	15.1s	1_LGBM_gla...	0.841	0.668		0.761		0.778	0.826
	1 month ago	15.6s	1_LGBM_gla...	0.871	0.771		0.872		0.788	0.857
	1 month ago	11.9s	1_LGBM_gla...	0.848	0.627		0.757		0.718	0.811
	1 month ago	13.2s	1_LGBM_gla...	0.841	0.644		0.733		0.786	0.826
	1 month ago	12.8s	1_LGBM_gla...	0.826	0.595		0.653		0.842	0.831
	1 month ago	12.8s	1_LGBM_gla...	0.848	0.664		0.78		0.744	0.822
	1 month ago	12.2s	1_LGBM_gla...	0.886	0.751		0.826		0.844	0.876
	1 month ago	12.5s	1_LGBM_gla...	0.833	0.595		0.69		0.763	0.812
	1 month ago	12.0s	1_LGBM_gla...	0.826	0.614		0.732		0.714	0.796
	1 month ago	11.9s	1_LGBM_gla...	0.833	0.585		0.683		0.757	0.81

Figure 26 - Interface of MLflow

MinIO

MinIO is a High-Performance Object Storage that can handle unstructured data such as photos, videos, log files, backups, and containers. In this project it was used to store the final dataset after its creation in the Jupyter Notebooks.

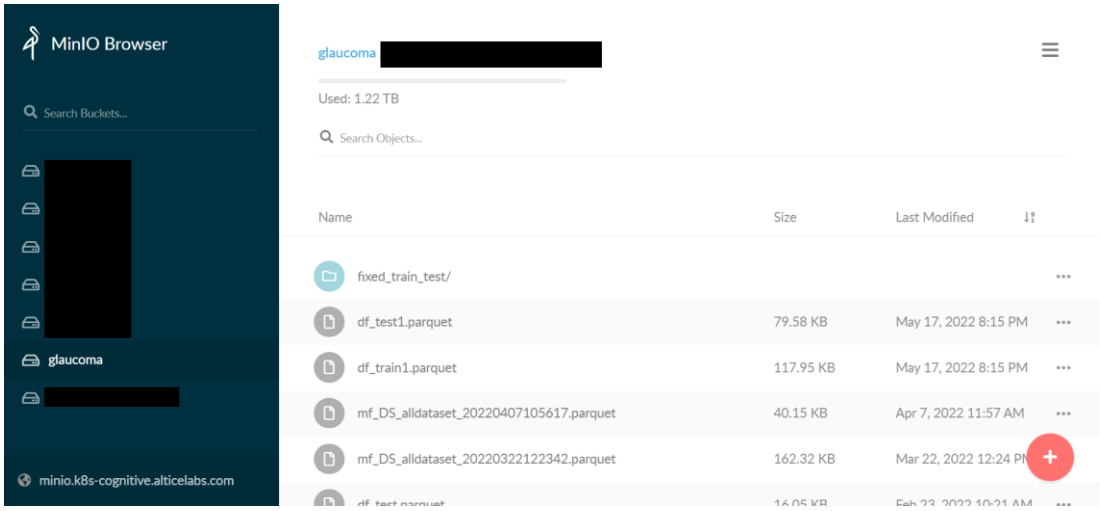


Figure 27 - Interface of MinIO

Python

Python is a high-level, interpreted, general-purpose programming language with a design philosophy that emphasizes code readability. One of the reasons for its popularity in Machine Learning projects is the large collection of libraries that users can work with, such as Scikit-Learn, PyCaret and Numpy. All the programming in this project was done using this language.

Scikit-learn

Scikit-learn is a Machine Learning library for the Python programming language. It features various classification, regression and clustering algorithms and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It can also be used in preprocessing, model selection and dimensionality reduction tasks.

PyCaret

PyCaret is an open-source, low-code and end-to-end library in Python that automates Machine Learning workflows. It provides a simple way to do ML tasks such as Exploratory Data Analysis, Data Preprocessing and Model Training. It was introduced with the purpose of validating the results obtained through the Scikit-learn library.

3.3 Procedure

The procedure followed to carry out this project will be explained in this section. To begin with, the steps of the methodology that were addressed above theoretically, will be addressed in a practical way in the Implementation section. In there, these steps will be thoroughly analyzed, and all the work done in each one of them will be presented.

Business Understanding was a critical stage that provided the necessary information to make project-related decisions. Following a review of the literature, it was decided to use Conventional Machine Learning algorithms rather than Deep Learning. The learning of the algorithms can be classified as Supervised Learning and further subclassified as a Classification task. The selection of algorithms (SVM, KNN, Random Forest, LGBM, and Logistic Regression) was also supported by the literature and the Altice Labs team.

For each one of the algorithms, it was performed 6 different iterations to evaluate which algorithm and set of features would produce better results. The different iterations are explained in the Model Training section. For every iteration the label used was *glaucoma_flag* as the goal of this work is to determine whether a patient has glaucoma.

All the programming of this project was done in the Python language using Jupyter Notebooks. Python is a trending language in Machine Learning projects due to its large collection of libraries. In that way, the Scikit-learn library was used to perform the project. On a later stage of the project, the library

PyCaret was introduced to validate the work done. All the different iterations were performed using the two libraries to have a more reliable result.

Bellow it will be presented the different Jupyter Notebooks used during the project, more specifically during the stages of Data Acquisition and Transformation and Modeling. There are a total of 31 notebooks and a brief explanation of each one will be provided.

Notebooks from 0.0 to 2.4 correspond to the Data Acquisition and Transformation stage. In these notebooks, tasks such as Dataset Creation and Exploratory Data Analysis were performed.

- **Notebook 0.0:** the first notebook was used to start exploring the tool and its features. It was not produced anything relevant to the project.

- **Notebook 1.0:** this is the notebook where the dataset was created, since reading the tables from the database, apply the transformations needed and exporting the dataset to MinIO.

- **Notebook 2.1:** in this notebook was performed an EDA where we could learn more about the number of nulls in the dataset, its shape, and the distribution of data.
- **Notebook 2.2:** in this notebook was also performed an EDA but with more complex techniques such as PCA and t-SNE. Some of these techniques were just tested but not implemented in the final work.
- **Notebook 2.3:** this notebook was similar to notebook 2.2 but were tested Feature Reduction techniques.
- **Notebook 2.4:** this notebook was similar to notebook 2.3 but it was tested the label eye_sick_label instead of the glaucoma_flag.

From the notebook 3.1 until the last one, the notebooks correspond to the Modeling stage. To be better organized, notebooks with the prefix 3 correspond to the first iteration using scikit-learn, notebooks with the prefix 4 correspond to the second iteration using scikit-learn, notebooks with the prefix 5 correspond to the third iteration using scikit-learn, notebooks with the prefix 6 correspond to the fourth iteration scikit-learn, notebooks with the prefix 7 correspond to the fifth iteration scikit-learn, and finally notebooks with the prefix 8 correspond to all the six iterations using the PyCaret library.

- **Notebook 3.1:** This notebook concerns the first iteration where the algorithm LGBM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 3.2:** This notebook concerns the first iteration where the algorithm Logistic Regression was applied. This iteration was performed using the scikit-learn library.
- **Notebook 3.3:** This notebook concerns the first iteration where the algorithm KNN was applied. This iteration was performed using the scikit-learn library.
- **Notebook 3.4:** This notebook concerns the first iteration where the algorithm SVM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 3.5:** This notebook concerns the first iteration where the algorithm Random Forest was applied. This iteration was performed using the scikit-learn library.

- **Notebook 4.1:** This notebook concerns the second iteration where the algorithm LGBM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 4.2:** This notebook concerns the second iteration where the algorithm Logistic Regression was applied. This iteration was performed using the scikit-learn library.
- **Notebook 4.3:** This notebook concerns the second iteration where the algorithm KNN was applied. This iteration was performed using the scikit-learn library.
- **Notebook 4.4:** This notebook concerns the second iteration where the algorithm SVM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 4.5:** This notebook concerns the second iteration where the algorithm Random Forest was applied. This iteration was performed using the scikit-learn library.

- **Notebook 5.1:** This notebook concerns the third iteration where the algorithm LGBM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 5.2:** This notebook concerns the third iteration where the algorithm SVM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 5.3:** This notebook concerns the third iteration where the algorithm Random Forest was applied. This iteration was performed using the scikit-learn library.

- **Notebook 6.1:** This notebook concerns the fourth iteration where the algorithm LGBM was applied. This iteration was performed using the scikit-learn library.

- **Notebook 6.2:** This notebook concerns the fourth iteration where the algorithm SVM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 6.3:** This notebook concerns the fourth iteration where the algorithm Random Forest was applied. This iteration was performed using the scikit-learn library.

- **Notebook 7.1:** This notebook concerns the fifth iteration where the algorithm LGBM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 7.2:** This notebook concerns the fifth iteration where the algorithm SVM was applied. This iteration was performed using the scikit-learn library.
- **Notebook 7.3:** This notebook concerns the fifth iteration where the algorithm Random Forest was applied. This iteration was performed using the scikit-learn library.

- **Notebook 8.1:** This notebook concerns the first iteration using the PyCaret library.
- **Notebook 8.2:** This notebook concerns the second iteration using the PyCaret library.
- **Notebook 8.3:** This notebook concerns the third iteration using the PyCaret library.
- **Notebook 8.4:** This notebook concerns the fourth iteration using the PyCaret library.
- **Notebook 8.5:** This notebook concerns the fifth iteration using the PyCaret library.
- **Notebook 8.6:** This notebook concerns the sixth iteration using the PyCaret library.

4. IMPLEMENTATION

This section will put into practice the methodology discussed above. All the stages covered theoretically in the Methodology section, such as Business Understanding, Data Acquisition and Transformation, Modeling and Deployment, will now be addressed practically. Each stage of the methodology will have its own sub-sections where all the work done will be explained.

4.1 Business Understanding

A correct solution to the problem involves a correct understanding of the problem. Business Understanding stage consists of a precise specification of the problem together with methods of evaluating the achievement of the goal. Before starting a ML project, all team members should have the domain knowledge to formulate the problem correctly. For example, this project in specific is about ophthalmic diseases, a subject that was completely unknown for me until then. Therefore, it is of utmost importance a stage where we try to understand the problem before starting to implement the solution itself. To have a better understanding of the subjects discussed, it was performed a Literature Review, which can be found above in its own section. This stage was extremally important to provide the necessary information to make decisions about important issues of the project. Bellow will be presented some of the most important conclusions of the Literature Review that were used to support the decisions made throughout the project.

- "However, they have a significant disadvantage, especially in the medical context. DL algorithms are called black boxes because the networks generate comprehensive features that are too complex to be interpreted by the human mind. These algorithms can achieve better performances but the way they analyze patterns and make decisions is unexplainable, which is not feasible in the healthcare field (Lu et al., 2020)." - This quote was the reason it was decided to use CML algorithms rather than DL algorithms. Despite being able to get better results, in the healthcare field it is given more relevance to interpretability of the results.
- "According to most research studies, supervised methods are more used because they can achieve better results (Lu et al., 2018)." - This quote was the reason it was decided to use Supervised Learning rather than Unsupervised Learning.
- "Classification is widely used in medical diagnosis, target marketing and credit approval." - This quote was the reason it was decided to use Classification rather than Regression or Clustering.

- "Random Forest and Support Vector Machine are the most used CML technologies in the ophthalmology field (Lu et al., 2018)" - Algorithm selection was done based on the literature and by suggestion of the Altice Labs team.

4.2 Data Acquisition and Transformation

Data Acquisition and Transformation has the objective of managing the complete data lifecycle, including ingesting, exploring, preparing, and transforming the data. Therefore, this step is composed of another sub-steps such as Data Ingestion, Data Description, Data Exploration and Data Preparation. Each one of these sub-steps has its own section bellow where they will be better described, and the work done in each one of them will be presented.

The creation of a properly clean, transformed, and consolidated dataset is the main goal of this step and its intended output. To this end, it is necessary to go through this entire data lifecycle where any possible inconsistency can be resolved.

After the dataset creation, a process of feature selection to improve the model's performance is also conducted. Large datasets are increasingly common and are often difficult to interpret. To interpret such datasets and prevent them from overfitting, methods are required to reduce their dimensionality by transforming a large set of variables into a smaller one that still preserves most of the information.

In the same way, training and testing a very large dataset can be a challenging task. As the performance of a ML algorithm depends heavily on the features contained in the dataset, we should only retain features that actually help our model to learn something. Unnecessary and redundant features slow down the training time of the algorithm and should be removed.

Reducing the number of variables of a dataset works as a trade-off between accuracy and interpretability. The goal in dimensionality reduction is to trade a little accuracy for simplicity, because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for ML algorithms.

4.2.1 Data Ingestion

Data sources used in this project are images from real medical examinations, made available by the Centro Cirúrgico de Coimbra. There are 4 different medical examinations: Ocular Tension, Visual Field Test (also referred as PEC due to its Portuguese translation), RNFL Analysis and Optical Biometry.

From the beginning of the project to the current date, CCC has delivered medical examinations to Altice Labs in 7 different moments, having in total data from 218 patients, 428 eyes and 479 diagnostics by eye. The data was then stored in a Postgres SQL Database, the database used internally by the Altice Labs team.

Table 1 - Deliveries of medical examinations from CCC

Delivery	Date	Description
Delivery 0	December 2020	Sample of 6 patients + January 2021 - sample of 141 patients (including 6 previous patients)
Delivery 1	July 2021	Sample of 15 patients
Delivery 2	August 2021	Sample of 19 patients (one already existing in delivery 1 with equal exams)
Delivery 3	October 2021	Sample of 51 patients (three already existing with new exams)
Delivery 4	November 2021	Sample of 24 patients
Delivery 5	December 2021	Sample of 76 patients
Delivery 6	February 2022	Following a request for clarification, missing diagnoses from deliveries 4 and 5 were provided

As said before, CCC (Centro Cirúrgico de Coimbra) delivered the data in the form of medical examinations. This raw format has low business value to a Machine Learning project. To transform it into useful data it was needed to extract the text from the images of the exams. I did not participate in

this task, as it was already in progress when I joined the Altice Labs team, but a brief explanation will still be provided.

The Altice Labs team used mainly two software tools for the text extraction. Opencv for Image Preprocessing and Tesseract for Optical Character Recognition (OCR). These two steps are sequential, Image Preprocessing must be done before Optical Character Recognition to improve the quality of this last step. Some of the tasks of Image Preprocessing include rescaling, binarization, noise removal and thresholding.

4.2.2 Data Description

In this section, it is demonstrated an ER Diagram that was designed by the Altice Labs team to illustrate how the entities relate to each other. For further details, in the Appendix it can be found a set of tables containing an individually description of all the entities used in the diagram. The tables are composed of Field, Type, Description, Example and Possible Values.

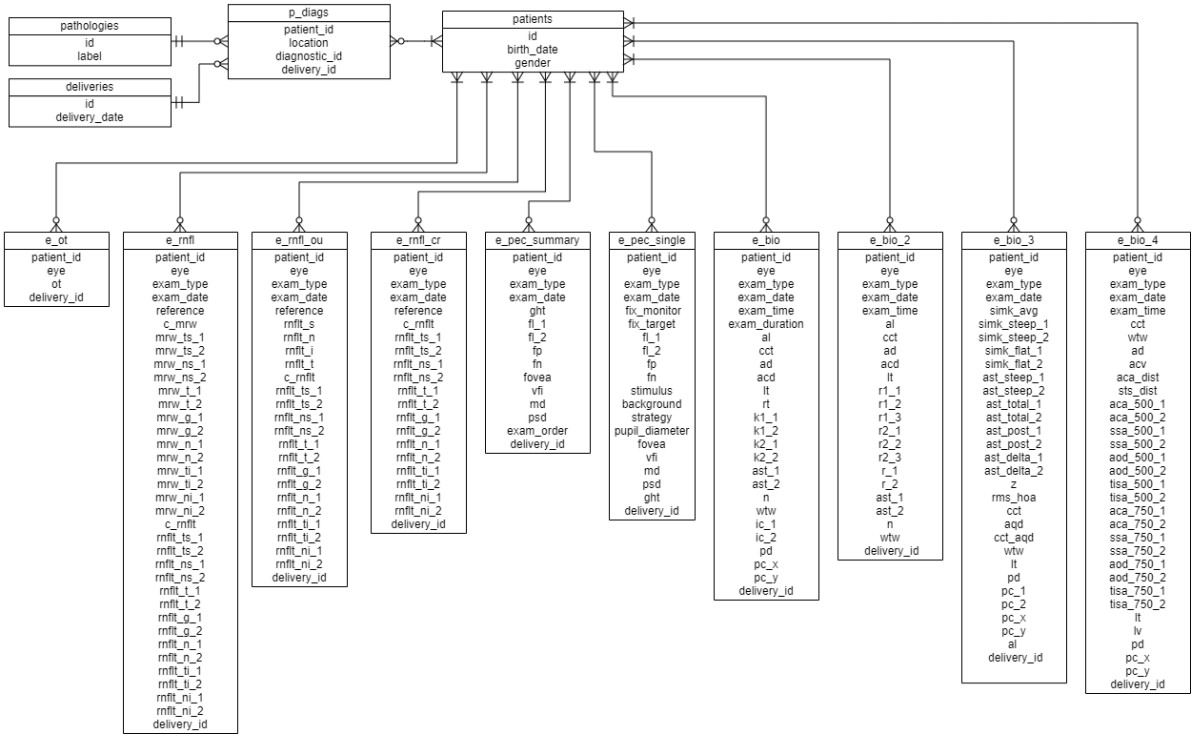


Figure 28 - ER Diagram

4.2.3 Data Exploration

After the data ingestion, we need to explore and understand the data we have in our hands. With that purpose, an Exploratory Data Analysis (EDA) was performed. EDA refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies and check assumptions with the help of summary statistics and graphical representations. It is a difficult task to look at a dataset as a whole and try to determine important characteristics of the data. Therefore, EDA techniques are a useful tool for seeing what the data can tell us before the modeling task.
In this section will be presented the main results obtained from the EDA. The steps performed in the Jupyter Notebooks to realize this analysis can be found in the Appendix.

Dataset Composition

- The created dataset is composed of 623 rows and 84 columns.

- In total the dataset has 1545 nulls, and their distribution and percentage can be seen in a table in the Appendix. In the table below, it is presented the top ten fields with most nulls.

Table 2 – Fields with most nulls on the dataset

Field	Nulls	Null %
al	39	6.3
acd	144	23.1
lt	161	25.8
k1_value	34	5.5
k1_angle	44	7.1
k2_value	34	5.5
k2_angle	44	7.1
ast_value	38	6.1
ast_angle	48	7.7
pd	159	25.5

- The feature "Label" has the possible values of: "Glaucoma" (29.9%), "Normal" (32.6%), "Outras Patologias" (14.6%), "Suspeito de Glaucoma" (23.0%). On the original data there were more pathologies other than glaucoma. Those values were replaced by "Outras Patologias"

referring to other pathologies, since the use case focus on the pathology of glaucoma. The transformations performed on the dataset will be addressed further in the document in its respective section.

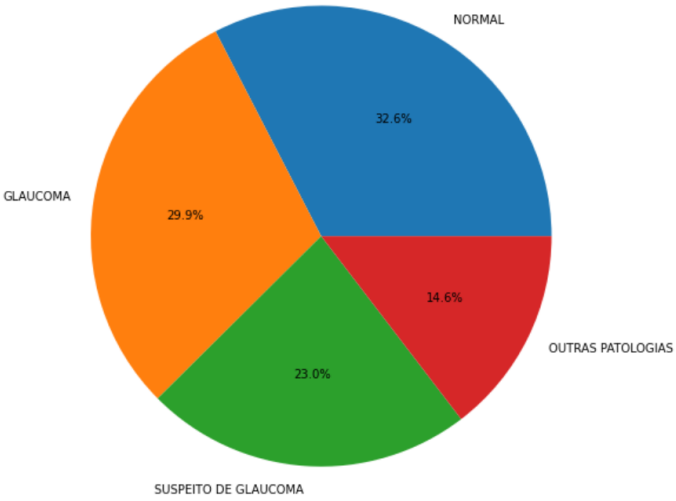


Figure 29 - Different values of the feature label

- The feature "Gender" has the possible values of "M" (49.9%) referring to Male patients and "F" (50.1%) referring to Female patients.

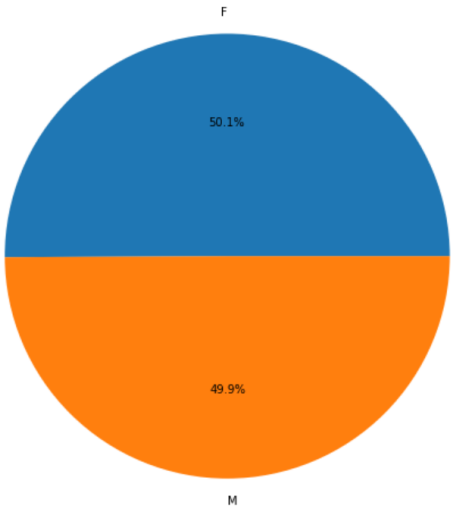


Figure 30 - Different values of the feature gender

Patients Analysis

- In the following image is presented the distribution of different pathologies by the patient's ages. It is noticeable that patients between 60 and 70 years represent most of the sample, possibly indicating that this age gap is the most affected by these pathologies.

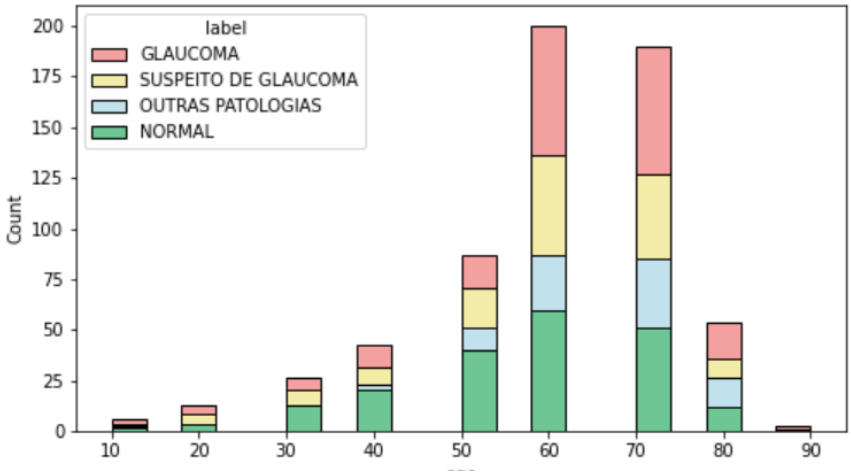


Figure 31 - Distribution of different pathologies by the patient's ages

- The following pie charts show the distribution of the patients by Label, Eye Sick Label, Glaucoma Flag and Gender, respectively.

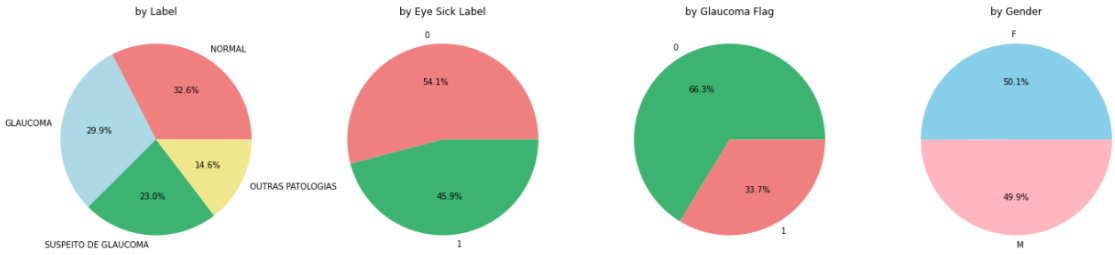


Figure 32 - Distribution of the patients by different labels

- In the following image we can see the distribution of the different pathologies by gender. There is no significant evidence that one of the genders is more affected by glaucoma or other pathologies than the other.

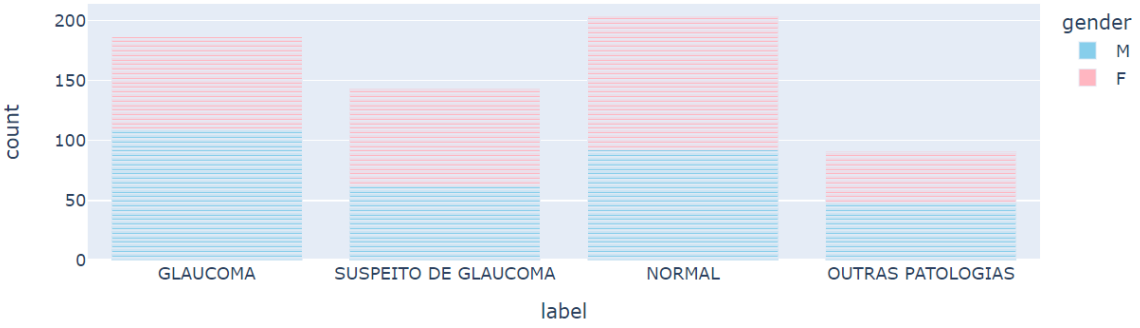


Figure 33 - Distribution of the different pathologies by gender

- The graph below represents the distribution of Ocular Tension and Pupil Diameter by Gender.

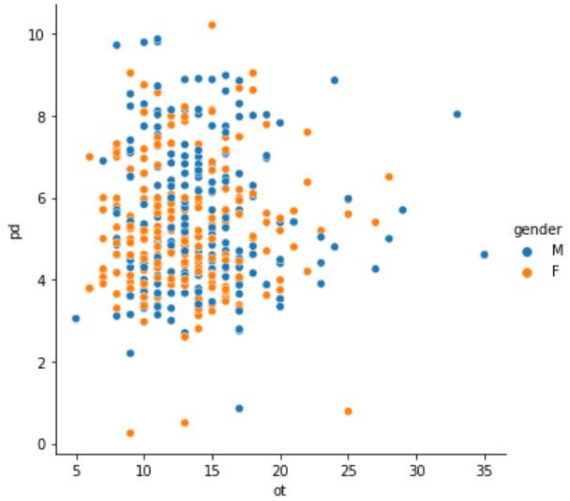


Figure 34 - Distribution of Ocular Tension and Pupil Diameter by Gender

- The graph below represents the distribution of Ocular Tension and Pupil Diameter by Age.

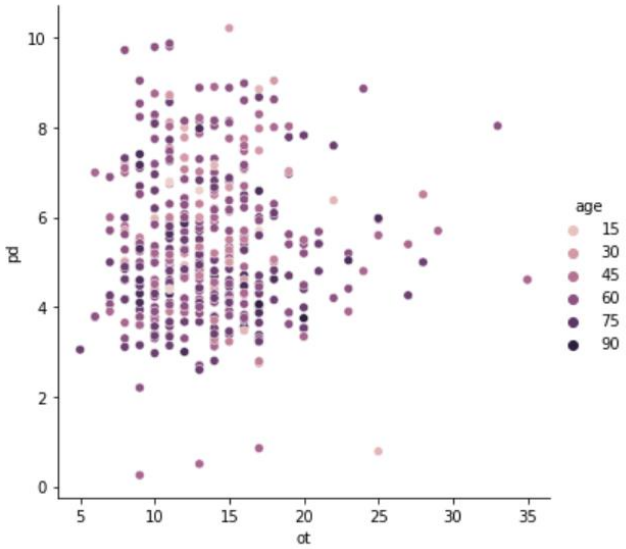


Figure 35 - Distribution of Ocular Tension and Pupil Diameter by Age

4.2.4 Data Preparation

Data preparation is the process of cleaning and transforming raw data prior to processing and analysis. It is an essential prerequisite to put data in context to turn it into insights and eliminate bias resulting from poor data quality. The components of Data Preparation include data preprocessing, cleansing and transformation. The goal of Data Preparation is to improve analysis and limit errors and inaccuracies that can occur to data during processing.

All Machine Learning algorithms have better performances with clean data, but besides that, some of the algorithms do not even work with raw data, such as null fields and categorical or numerical values. Also, when training a model sometimes it is necessary to bring the range of the numerical variables to a common scale. This helps to give equal importance to all the variables at the time of training.

In this section are presented the transformations performed on the original data. The steps performed in the Jupyter Notebooks to transform the data can be found in the Appendix.

Patients Data

The following modifications are applied to the patient's data:

- Creates the field age_10 (age truncated down to the 10's)
- Uppercases all the string values so that they are uniform

Diagnostic Data

The following modifications are applied to the diagnostics data:

- Uppercases all the string values so that they are uniform
- Redefines the label by turning the values that are not "Normal", "Glaucoma" or "Suspeito de Glaucoma" into "OUTRAS PATOLOGIAS"
- Drops duplications originated from changing any label other than the previous ones to " OUTRAS PATOLOGIAS ". An eye that had more than one pathology would be repeated with the same label " OUTRAS PATOLOGIAS"

Biometry Data

The following modifications are applied to the biometry data:

- Uppercases all the string values so that they are uniform
- Processes exam_duration field so that it contains the numeric values for the exam duration in minutes

PEC Exams Data

The following modifications are applied to the PEC data:

- Discards the columns exam_time, exam_duration, fix_monitor, fix_target, stimulus, background, strategy and pd, as these fields tend to have the same values in all samples
- Uppercases all the string values so that they are uniform
- Computes fl (Fixation Losses) ratio by dividing fl_1 by fl_2. Null results are converted to zero
- Replaces null values in fp and fn by zero

Pupil Diameter Data

The following processes are done to obtain the pupil diameter data:

- Concatenates PEC data and Biometry data where the pd value is not Null, keeping only the patient_id, eye, exam_type, exam_date, exam_time and pd
- Uppercases all the string values so that they are uniform

RNFL Data

The following modifications are applied to the RNFL data:

- Uppercases all the string values so that they are uniform

The transformations mentioned above were the ones made to each table individually. To the dataset as a whole, were also applied transformations such as dropping nulls and standardization of values.

4.2.5 Resulting Dataset

After all the steps performed during the Data Acquisition and Transformation stage, the resulting dataset consists of the following features:

Table 3 - Features of the resulting dataset and their origin

Features	Origin	Description
patient_id	Relation field	Patient identifier
age	Patient Data	Age of the patient at the last exam
gender	Patient Data	Gender of the patient
label	Diagnostics Data	Name associated with the pathology
other_eye_glau_flag	Diagnostics Data	Flag (1 or 0) indicating that the other eye has glaucoma
glaucoma_flag	Diagnostics Data	Flag (1 or 0) indicating that the eye has glaucoma
eye_sick_label	Diagnostics Data	Flag (1 or 0) indicating that the eye is affected by a sickness
eye	Diagnostics Data	Eye (left or right)
vfi_last	PEC Data	Last VFI (Visual Field Index) determined for the eye
vfi_min	PEC Data	Lowest VFI determined for the eye
vfi_max	PEC Data	Largest VFI determined for the eye

vfi_mean	PEC Data	Average of the various VFIs determined for the eye
vfi_std	PEC Data	Standard deviation of the various VFIs determined for the eye
md_last	PEC Data	Last MD (Mean deviation) determined for the eye
md_min	PEC Data	Lowest MD determined for the eye
md_max	PEC Data	Largest MD determined for the eye
md_mean	PEC Data	Average of the various MDs determined for the eye
md_std	PEC Data	Standard deviation of the various MDs determined for the eye
psd_last	PEC Data	Last PSD (Pattern standard deviation) determined for the eye
psd_min	PEC Data	Lowest PSD determined for the eye
psd_max	PEC Data	Largest PSD determined for the eye
psd_mean	PEC Data	Average of the various PSDs determined for the eye
psd_std	PEC Data	Standard deviation of the various PSDs determined

		for the eye
last_fl	PEC Data	Last FL (Fixation Losses) determined for the eye
last_ght	PEC Data	Last GHT (Glaucoma Hemifield Test) determined for the eye
pec_qtd	PEC Data	Quantity of PEC exams in an eye
al	Biometry Data	Axial length (mm)
cct	Biometry Data	Cornea thickness (μm)
ad	Biometry Data	Aqueous depth (mm)
acd	Biometry Data	Anterior chamber depth (mm)
lt	Biometry Data	Lens thickness (mm)
k1_value	Biometry Data	Flat meridian value (D)
k1_angle	Biometry Data	Flat meridian angle (°)
k2_value	Biometry Data	Steep meridian value (D)
k2_angle	Biometry Data	Steep meridian angle (°)
ast_value	Biometry Data	Astigmatism value (D)
ast_angle	Biometry Data	Astigmatism angle (°)
wtw	Biometry Data	White to White (mm)
ic_x	Biometry Data	Iris barycenter x-axis (mm)
ic_y	Biometry Data	Iris barycenter y-axis (mm)
pc_x	Biometry Data	Pupil barycenter x-axis (mm)
pc_y	Biometry Data	Pupil barycenter y-axis (mm)
pd	Pupil Diameter Data	Pupil diameter (mm)
ot	Optical Tension Data	Optical Tension measurement
mrw_result	RNFL Data	Minimum Rim Width

mrw_g_value	RNFL Data	Absolute value for MRW Global
mrw_g_perc	RNFL Data	Percentile ratio of the MRW Global
mrw_ts_value	RNFL Data	Absolute value for MRW Temporal Superior sector
mrw_ns_value	RNFL Data	Absolute value for MRW Nasal Superior sector
mrw_t_value	RNFL Data	Absolute value for MRW Temporal sector
mrw_n_value	RNFL Data	Absolute value for MRW Nasal sector
mrw_ti_value	RNFL Data	Absolute value for MRW Temporal Inferior sector
mrw_ni_value	RNFL Data	Absolute value for MRW Nasal Inferior sector
mrw_ts_perc	RNFL Data	Percentile ratio of the MRW Temporal Superior sector
mrw_ns_perc	RNFL Data	Percentile ratio of the MRW Nasal Superior sector
mrw_t_perc	RNFL Data	Percentile ratio of the MRW Temporal sector
mrw_n_perc	RNFL Data	Percentile ratio of the MRW Nasal sector
mrw_ti_perc	RNFL Data	Percentile ratio of the MRW Temporal Inferior sector
mrw_ni_perc	RNFL Data	Percentile ratio of the MRW Nasal Inferior sector
rnflt_result	RNFL Data	Retinal Nerve Fiber Layer
rnflt_g_value	RNFL Data	Absolute value for RNFLT Global

rnflt_g_perc	RNFL Data	Percentile ratio of the RNFLT Global
rnflt_ts_value	RNFL Data	Absolute value for RNFLT Temporal Superior sector
rnflt_ns_value	RNFL Data	Absolute value for RNFLT Nasal Superior sector
rnflt_t_value	RNFL Data	Absolute value for RNFLT Temporal sector
rnflt_n_value	RNFL Data	Absolute value for RNFLT Nasal sector
rnflt_ti_value	RNFL Data	Absolute value for RNFLT Temporal Inferior sector
rnflt_ni_value	RNFL Data	Absolute value for RNFLT Nasal Inferior sector
rnflt_ts_perc	RNFL Data	Percentile ratio of the RNFLT Temporal Superior sector
rnflt_ns_perc	RNFL Data	Percentile ratio of the RNFLT Nasal Superior sector
rnflt_t_perc	RNFL Data	Percentile ratio of the RNFLT Temporal sector
rnflt_n_perc	RNFL Data	Percentile ratio of the RNFLT Nasal sector
rnflt_ti_perc	RNFL Data	Percentile ratio of the RNFLT Temporal Inferior sector
rnflt_ni_perc	RNFL Data	Percentile ratio of the RNFLT Nasal Inferior sector
mrw_value_min	RNFL Data	Minimum Absolute Value of MRW

mrw_value_max	RNFL Data	Maximum Absolute Value of MRW
mrw_perc_min	RNFL Data	Minimum Percentile Value of MRW
mrw_perc_max	RNFL Data	Maximum Percentile Value of MRW
rnflt_value_min	RNFL Data	Minimum Absolute Value of RNFLT
rnflt_value_max	RNFL Data	Maximum Absolute Value of RNFLT
rnflt_perc_min	RNFL Data	Minimum Percentile Value of RNFLT
rnflt_perc_max	RNFL Data	Maximum Percentile Value of RNFLT
rnflt_outsides	RNFL Data	Quantity of sectors of the eye that had a qualitative evaluation of outside normal limits
rnflt_borderlines	RNFL Data	Quantity of eye sectors that had a qualitative borderline assessment

4.3 Modeling

In this section it will be covered the training of the Machine Learning model and all the steps that led to it, from algorithm selection to validation. To train a ML algorithm able to make predictions based on the provided data it is necessary to apply mathematical, computer science, and business knowledge. It is a crucial stage that will determine the quality and accuracy of future predictions in new situations.

The process of modeling involves training a ML algorithm to predict the labels from the features, tuning it for the business needs, and validating it. Essentially, the objective of this step is to select, apply, and calibrate several modeling techniques to achieve optimal values.

This section is divided in Algorithm Selection, Model Training and Validation.

4.3.1 Algorithm Selection

In this step the most appropriated algorithms for training are selected. There are several trade-off points in selecting the best algorithm like model complexity, interpretability, performance, and computer requirements. After a consideration of the different trade-offs and by suggestion of the Altice Labs team, the selected algorithms were KNN, LGBM, Random Forest, Logistic Regression and SVM. The first iterations of Model Training included all the selected algorithms to identify which produces better results. Following iterations only included the best suited models to simplify the work done. The choice for the best suited models were based on the Literature Review and on results obtained in the previous iterations. According to the Literature Review, Random Forest and Support Vector Machine are the most used CML technologies in the ophthalmology field (Lu et al., 2018). Thus, these two algorithms, and LGBM which was obtaining good results, were the algorithms selected for the rest of the iterations.

4.3.2 Model Training

Model Training is the phase of the project lifecycle in which training data is provided to a Machine Learning algorithm to learn from. The goal of model training is to create the most accurate mathematical model of the correlation between data features and a target label. The model's performance dictates how well the applications created with it will perform. After completing all the preparatory steps, such as data ingestion, dataset creation, and data transformation, and after selecting the desired algorithms, it is time to apply them and train the models.

This step was divided in 6 different iterations that will now be explained. To all these 6 iterations, the label used was *glaucoma_flag* as the objective of the work is to determine if a patient has glaucoma or not.

In this section, it will only be explained theoretically the different iterations. The procedure followed to perform these iterations, since splitting the data into test and train sets, creating the model and evaluate metrics, is described in the Appendix using prints from the Jupyter Notebooks.

Iteration 1

In the first iteration of Model Training, all the selected algorithms were applied to a dataset containing all the existent features, except for the labels and the patient ID because they do not have business value and even affect the model's performance. This first iteration, the one with the greatest number of features, will serve as a baseline for the next iterations where different combinations of features will be tested. In the table below it can be found the list of features used in this iteration.

Table 4 - Features used in Iteration 1

Feature	Origin
age	Patient Data
gender	Patient Data
eye	Diagnostics Data
vfi_last	PEC Data
vfi_min	PEC Data
vfi_max	PEC Data
vfi_mean	PEC Data
vfi_std	PEC Data
md_last	PEC Data
md_min	PEC Data
md_max	PEC Data
md_mean	PEC Data
md_std	PEC Data
psd_last	PEC Data
psd_min	PEC Data
psd_max	PEC Data
psd_mean	PEC Data
psd_std	PEC Data
last_fl	PEC Data
last_ght	PEC Data
pec_qtd	PEC Data
al	Biometry Data
cct	Biometry Data
ad	Biometry Data

acd	Biometry Data
lt	Biometry Data
k1_value	Biometry Data
k1_angle	Biometry Data
k2_value	Biometry Data
k2_angle	Biometry Data
ast_value	Biometry Data
ast_angle	Biometry Data
wtw	Biometry Data
ic_x	Biometry Data
ic_y	Biometry Data
pc_x	Biometry Data
pc_y	Biometry Data
pd	Pupil Diameter Data
ot	Optical Tension Data
mrw_result	RNFL Data
mrw_g_value	RNFL Data
mrw_g_perc	RNFL Data
mrw_ts_value	RNFL Data
mrw_ns_value	RNFL Data
mrw_t_value	RNFL Data
mrw_n_value	RNFL Data
mrw_ti_value	RNFL Data
mrw_ni_value	RNFL Data
mrw_ts_perc	RNFL Data
mrw_ns_perc	RNFL Data
mrw_t_perc	RNFL Data
mrw_n_perc	RNFL Data
mrw_ti_perc	RNFL Data
mrw_ni_perc	RNFL Data
rnflt_result	RNFL Data

rnflt_g_value	RNFL Data
rnflt_g_perc	RNFL Data
rnflt_ts_value	RNFL Data
rnflt_ns_value	RNFL Data
rnflt_t_value	RNFL Data
rnflt_n_value	RNFL Data
rnflt_ti_value	RNFL Data
rnflt_ni_value	RNFL Data
rnflt_ts_perc	RNFL Data
rnflt_ns_perc	RNFL Data
rnflt_t_perc	RNFL Data
rnflt_n_perc	RNFL Data
rnflt_ti_perc	RNFL Data
rnflt_ni_perc	RNFL Data
mrw_value_min	RNFL Data
mrw_value_max	RNFL Data
mrw_perc_min	RNFL Data
mrw_perc_max	RNFL Data
rnflt_value_min	RNFL Data
rnflt_value_max	RNFL Data
rnflt_perc_min	RNFL Data
rnflt_perc_max	RNFL Data
rnflt_outsides	RNFL Data
rnflt_borderlines	RNFL Data

Iteration 2

In the second iteration of Model Training, features that were previously identified, in the Data Exploration phase, as features with a big percentage of nulls were removed. All the algorithms selected were applied in this iteration. In the table below it can be found the list of features used in this iteration.

Table 5 - Features used in Iteration 2

Field	Origin
age	Patient Data
gender	Patient Data
glaucoma_flag	Diagnostics Data
eye	Diagnostics Data
vfi_last	PEC Data
md_last	PEC Data
psd_last	PEC Data
last_ght	PEC Data
al	Biometry Data
cct	Biometry Data
ot	Optical Tension Data
mrw_result	RNFL Data
mrw_ts_perc	RNFL Data
mrw_ns_perc	RNFL Data
mrw_t_perc	RNFL Data
mrw_g_perc	RNFL Data
mrw_n_perc	RNFL Data
mrw_ti_perc	RNFL Data
mrw_ni_perc	RNFL Data
mrw_perc_min	RNFL Data
mrw_perc_max	RNFL Data
rnflt_result	RNFL Data
rnflt_ts_perc	RNFL Data
rnflt_ns_perc	RNFL Data
rnflt_t_perc	RNFL Data
rnflt_g_perc	RNFL Data
rnflt_n_perc	RNFL Data
rnflt_ti_perc	RNFL Data
rnflt_ni_perc	RNFL Data

rnflt_perc_min	RNFL Data
rnflt_perc_max	RNFL Data
rnflt_outsides	RNFL Data
rnflt_borderlines	RNFL Data
mrw_g_value	RNFL Data
rnflt_g_value	RNFL Data
mrw_value_min	RNFL Data
rnflt_value_min	RNFL Data

Iteration 3

In the third iteration, and from here on to simplify the work done, only three of the selected algorithms were applied: LGBM, SVM and Random Forest. For this iteration, it was performed Feature Selection and it were selected features that were already identified to produce better results. This set of features was achieved mainly with help of the medical staff at CCC (Centro Cirúrgico de Coimbra), which informed the Altice Labs team, the features that normally are used to identify if a patient has glaucoma. In the table below it can be found the list of features used in this iteration.

Table 6 - Features used in Iteration 3

Field	Origin
age	Patient Data
gender	Patient Data
glaucoma_flag	Diagnostics Data
eye	Diagnostics Data
vfi_last	PEC Data
md_last	PEC Data
psd_last	PEC Data
last_ght	PEC Data
al	Biometry Data
cct	Biometry Data
ot	Optical Tension Data
mrw_g_value	RNFL Data

rnflt_g_value	RNFL Data

Iteration 4 and 5

In the fourth iteration, a set containing features with only absolute values was used. In turn, in the fifth iteration, it was used a set containing features with only percentile values. The objective of these two iterations was to assess which of the sets produced better results. In the tables below it can be found the list of features used in these iterations.

Iteration 4:

Table 7 - Features used in Iteration 4

Field	Origin
age	Patient Data
gender	Patient Data
glaucoma_flag	Diagnostics Data
eye	Diagnostics Data
vfi_last	PEC Data
md_last	PEC Data
psd_last	PEC Data
last_ght	PEC Data
al	Biometry Data
cct	Biometry Data
ot	Optical Tension Data
mrw_g_value	RNFL Data
rnflt_g_value	RNFL Data
mrw_ts_value	RNFL Data
mrw_ns_value	RNFL Data
mrw_t_value	RNFL Data
mrw_n_value	RNFL Data
mrw_ti_value	RNFL Data
mrw_ni_value	RNFL Data

rnflt_ts_value	RNFL Data
rnflt_ns_value	RNFL Data
rnflt_t_value	RNFL Data
rnflt_n_value	RNFL Data
rnflt_ti_value	RNFL Data
rnflt_ni_value	RNFL Data
mrw_value_min	RNFL Data
mrw_value_max	RNFL Data
rnflt_value_min	RNFL Data
rnflt_value_max	RNFL Data

Iteration 5:

Table 8 - Features used in Iteration 5

Field	Origin
age	Patient Data
gender	Patient Data
glaucoma_flag	Diagnostics Data
eye	Diagnostics Data
vfi_last	PEC Data
md_last	PEC Data
psd_last	PEC Data
last_ght	PEC Data
al	Biometry Data
cct	Biometry Data
ot	Optical Tension Data
mrw_g_perc	RNFL Data
rnflt_g_ perc	RNFL Data
mrw_ts_ perc	RNFL Data
mrw_ns_ perc	RNFL Data
mrw_t_ perc	RNFL Data

mrw_n_ perc	RNFL Data
mrw_ti_ perc	RNFL Data
mrw_ni_ perc	RNFL Data
rnflt_ts_ perc	RNFL Data
rnflt_ns_ perc	RNFL Data
rnflt_t_ perc	RNFL Data
rnflt_n_ perc	RNFL Data
rnflt_ti_ perc	RNFL Data
rnflt_ni_ perc	RNFL Data
mrw_ perc _min	RNFL Data
mrw_ perc _max	RNFL Data
rnflt_ perc _min	RNFL Data
rnflt_ perc _max	RNFL Data

Iteration 6

In the sixth iteration, it was used a set of features identified as the features that were more important to glaucoma identification. To create this dataset, techniques of Feature Importance were applied to the iterations 1 and 2 and the best features were selected. This iteration was different since it was only done in PyCaret, as opposed to the other 5 iterations that were done both using the PyCaret and Scikit Learn library. In the table below it can be found the list of features used to this iteration.

Table 9 - Features used in Iteration 6

Field	Origin
age	Patient Data
gender	Patient Data
glaucoma_flag	Diagnostics Data
eye	Diagnostics Data
mrw_g_value	PEC Data
mrw_value_max	PEC Data
rnflt_g_value	PEC Data
mrw_ni_value	PEC Data

mrw_n_value	Biometry Data
mrw_ts_value	Biometry Data
mrw_n_perc	Optical Tension Data
mrw_g_perc	RNFL Data
mrw_value_min	RNFL Data
mrw_perc_min	RNFL Data
mrw_ni_perc	RNFL Data
mrw_ti_perc	RNFL Data
rnflt_g_perc	RNFL Data
mrw_ns_perc	RNFL Data
rnflt_ti_perc	RNFL Data
mrw_ns_value	RNFL Data

4.3.3 Validation

With the goal of validating the work done in the previous step, it was used the tool PyCaret, a low-code python library that allows to easily replicate all the work done and validate it. PyCaret works with functions that perform tasks in a workflow. The functions used in this work will be explained below.

Setup

This function must be called before executing any other function, it initializes the experiment and prepares it based on all the parameters passed in the function. The parameters used in this experiment and its respective values are:

- Data: This parameter corresponds to the data used in the experiment. As it was explained in the previous section, the model was trained in 6 different iterations, so 6 different data frames in 6 different Jupyter Notebooks were used.
- Target: Name of the target column to be passed in as a string. The target column is the column whose values are modeled and predicted by other variables. For all the iterations, the target column was *glaucoma_flag*.

- Ignore Features: It is used to ignore features during model training. For example, one of the features ignored in all iterations was patient ID since it does not represent value to the model training.
- Train Size: Proportion of the dataset to be used for training and validation. In this case the proportion used was 0.75 meaning that 75% of the dataset will be used for training and validation. The other 25% of the dataset will be used for testing.
- Normalize: It transforms the numeric features by scaling them to a given range.
- Fold: Number of folds to be used in cross validation. In this case it was used 3 folds.
- Data Split Stratify: Controls stratification during Train Test Split. Stratification ensures that training and test sets have the same proportion of the feature of interest as in the original dataset.
- Fix Imbalance: It is used to balance the unequal distribution of target class when training a dataset.

Compare Models

This function trains and evaluates the performance of the chosen models using cross-validation. The function had 2 parameters. The parameter "sort" had the value of "F1", meaning that F1 Score would be the metric used to rank the models results, and the parameter "include" where the value was the models we intended to use. For the iteration 1 and 2 all the selected models were used (LGBM, Random Forest, KNN, Logistic Regression and SVM) and for the remaining iterations the models LGBM, Random Forest and SVM were used. The output of this function is a scoring grid with average cross-validated scores.

Tune Model

This function tunes the hyperparameters of a given model with the goal of optimizing it. The function was used with 3 Folds and 100 iterations as parameters. Normally, increasing the number of iterations improves the model performance but also increases the training time. The function also had the parameter "optimize" with the value of "F1" since this was the metric we wanted to be optimized

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In Machine Learning, this is accomplished by selecting appropriate hyperparameters.

Tuning is usually a trial-and-error process by which you change some hyperparameters, run the algorithm on the data again, then compare its performance on your validation set to determine which set of hyperparameters results in the most accurate model.

Evaluate Model

This function displays a basic user interface for analyzing the performance of a trained model. It shows several types of plots including AUC, Confusion Matrix, and Feature Importance.

4.3.4 Model Assessment

Model Assessment is the process of using different evaluation metrics to understand a ML model's performance. While evaluating models, selecting the right metric is a critical step of the task. Different applications of the model demand the analysis of different metrics. For some applications it can be more important to analyze the True Positive Rate, while for others is the True Negative Rate. For other applications, it may even be necessary to use a set of metrics to get the whole picture of the problem.

From the existent metrics, the ones that were considered in this work were: Accuracy, Precision, Recall, AUC, F1 Score and Confusion Matrix. The metric used to rank the models was F1 Score since it is one of the most complete metrics. Recall was also considered due to its importance in the medical field. This metric is regarded as being among the most important for medical studies, since it is desired to miss as few positive instances as possible (Hicks et al., 2022).

Both Recall and Specificity are important metrics in this field of study as they complement each other and are both needed to fully understand a model's strengths and weaknesses. The ideal situation would be to have both high Recall and high Specificity, but it is important to recognize that these two metrics exist in a state of balance. Increased Recall normally comes at the expense of reduced Specificity. In a situation where a trade-off is needed, we should choose the metric that is more suitable for the problem at hand. As in the medical field the goal is to minimize false-negative results, we prioritize Recall since it measures how often a test correctly generates a positive result for people who have the condition that is being tested for. A test that is highly sensitive (high Recall) will flag almost everyone who has the disease and not generate many false-negative results.

4.4 Deployment

Deployment is the process of integrating a Machine Learning model in a production environment where it can be used for its intended purpose. Models can be deployed in a wide range of environments, and it involves extensive planning, documentation, and a variety of different tools to complete this process. It also requires coordination between IT teams, software developers, and business professionals to ensure the model works reliably, since there is often a discrepancy between the programming language in which a model is written and the languages of the production system.

Being this an academic work, this phase will not be addressed practically, as the model will not be integrated in the Altice Labs production environment.

5. RESULTS

In this section will be presented the results obtained across the different iterations. These results were evaluated through different metrics that were already described.

For each iteration it will be presented both ways that were used to perform it. The first one was using the Scikit Learn library through Jupyter Notebooks and the second one was using the PyCaret library, also through Jupyter Notebooks.

5.1 Scikit-learn

To get a more organized view, the results were grouped on MLflow. For each algorithm of each iteration, 10 runs were performed to obtain a more reliable result. The image below shows, as an example, the 10 runs performed for the algorithm LGBM in the Iteration 1 (prints for the other algorithms and iterations can be found in the Appendix). The results of the metrics presented in the tables below are the average of those 10 runs.

				Metrics					
	↑ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2_	recall_score	roc_auc_score
	1 month ago	15.1s	1_LGBM_gla...	0.841	0.668		0.761		0.778	0.826
	1 month ago	15.6s	1_LGBM_gla...	0.871	0.771		0.872		0.788	0.857
	1 month ago	11.9s	1_LGBM_gla...	0.848	0.627		0.757		0.718	0.811
	1 month ago	13.2s	1_LGBM_gla...	0.841	0.644		0.733		0.786	0.826
	1 month ago	12.8s	1_LGBM_gla...	0.826	0.595		0.653		0.842	0.831
	1 month ago	12.8s	1_LGBM_gla...	0.848	0.664		0.78		0.744	0.822
	1 month ago	12.2s	1_LGBM_gla...	0.886	0.751		0.826		0.844	0.876
	1 month ago	12.5s	1_LGBM_gla...	0.833	0.595		0.69		0.763	0.812
	1 month ago	12.0s	1_LGBM_gla...	0.826	0.614		0.732		0.714	0.796
	1 month ago	11.9s	1_LGBM_gla...	0.833	0.585		0.683		0.757	0.81

Figure 36 – Example of the 10 runs performed for the algorithm LGBM in the Iteration 1

Iteration 1

- **LGBM**

Table 10 - Results of the algorithm LGBM in Iteration 1, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.845	0.749	**0.773**	**0.827**	0.776

Table 11 - Confusion Matrix of the algorithm LGBM in Iteration 1, using the Scikit-learn library

46 (TN)	6 (FP)
5 (FN)	14 (TP)

- **Logistic Regression**

Table 12 - Results of the algorithm LR in Iteration 1, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.815	0.689	0.646	0.765	0.824

Table 13 - Confusion Matrix of the algorithm LR in Iteration 1, using the Scikit-learn library

45 (TN)	6 (FP)
8 (FN)	12 (TP)

- **KNN**

Table 14 - Results of the algorithm KNN in Iteration 1, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.817	0.750	0.598	0.756	0.804

Table 15 - Confusion Matrix of the algorithm KNN in Iteration 1, using the Scikit-learn library

46 (TN)	4 (FP)
9 (FN)	12 (TP)

- **SVM**

Table 16 - Results of the algorithm SVM in Iteration 1, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.849	**0.892**	0.502	0.737	0.816

Table 17 - Confusion Matrix of the algorithm SVM in Iteration 1, using the Scikit-learn library

51 (TN)	1 (FP)
9 (FN)	10 (TP)

- **Random Forest**

Table 18 - Results of the algorithm RF in Iteration 1, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.869	0.826	0.660	0.803	**0.892**

Table 19 - Confusion Matrix of the algorithm RF in Iteration 1, using the Scikit-learn library

49 (TN)	3 (FP)
6 (FN)	13 (TP)

Iteration 2

- **LGBM**

Table 20 - Results of the algorithm LGBM in Iteration 2, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.820	0.701	**0.796**	0.813	0.744

Table 21 - Confusion Matrix of the algorithm LGBM in Iteration 2, using the Scikit-learn library

42 (TN)	11 (FP)
3 (FN)	15 (TP)

- **Logistic Regression**

Table 22 - Results of the algorithm LR in Iteration 2, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.839	0.779	0.699	0.802	0.838

Table 23 - Confusion Matrix of the algorithm LR in Iteration 2, using the Scikit-learn library

50 (TN)	3 (FP)
8 (FN)	10 (TP)

- **KNN**

Table 24 - Results of the algorithm KNN in Iteration 2, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.823	0.777	0.639	0.776	0.816

Table 25 - Confusion Matrix of the algorithm KNN in Iteration 2, using the Scikit-learn library

47 (TN)	5 (FP)
8 (FN)	11 (TP)

- **SVM**

Table 26 - Results of the algorithm SVM in Iteration 2, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.809	0.783	0.585	0.752	0.840

Table 27 - Confusion Matrix of the algorithm SVM in Iteration 2, using the Scikit-learn library

51 (TN)	2 (FP)
9 (FN)	9 (TP)

- **Random Forest**

Table 28 - Results of the algorithm RF in Iteration 2, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.873	**0.854**	0.740	**0.839**	**0.864**

Table 29 - Confusion Matrix of the algorithm RF in Iteration 2, using the Scikit-learn library

47 (TN)	2 (FP)
6 (FN)	16 (TP)

Iteration 3

- **LGBM**

Table 30 - Results of the algorithm LGBM in Iteration 3, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.835	0.765	**0.772**	**0.820**	0.736

Table 31 - Confusion Matrix of the algorithm LGBM in Iteration 3, using the Scikit-learn library

39 (TN)	10 (FP)
4 (FN)	18 (TP)

- **SVM**

Table 32 - Results of the algorithm SVM in Iteration 3, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.795	**0.884**	0.473	0.719	0.786

Table 33 - Confusion Matrix of the algorithm SVM in Iteration 3, using the Scikit-learn library

49 (TN)	2 (FP)
12 (FN)	8 (TP)

- **Random Forest**

Table 34 - Results of the algorithm RF in Iteration 3, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.853	0.851	0.667	0.805	**0.886**

Table 35 - Confusion Matrix of the algorithm RF in Iteration 3, using the Scikit-learn library

48 (TN)	2 (FP)
6 (FN)	15 (TP)

Iteration 4

- **LGBM**

Table 36 - Results of the algorithm LGBM in Iteration 4, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.810	0.701	**0.740**	0.792	0.767

Table 37 - Confusion Matrix of the algorithm LGBM in Iteration 4, using the Scikit-learn library

43 (TN)	9 (FP)
4 (FN)	15 (TP)

- **SVM**

Table 38 - Results of the algorithm SVM in Iteration 4, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.812	**0.899**	0.530	0.749	0.828

Table 39 - Confusion Matrix of the algorithm SVM in Iteration 4, using the Scikit-learn library

52 (TN)	1 (FP)
9 (FN)	9 (TP)

- **Random Forest**

Table 40 - Results of the algorithm RF in Iteration 4, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.860	0.826	0.723	**0.825**	**0.852**

Table 41 - Confusion Matrix of the algorithm RF in Iteration 4, using the Scikit-learn library

49 (TN)	3 (FP)
5 (FN)	14 (TP)

Iteration 5

- **LGBM**

Table 42 - Results of the algorithm LGBM in Iteration 5, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.801	0.682	**0.794**	0.801	0.726

Table 43 - Confusion Matrix of the algorithm LGBM in Iteration 5, using the Scikit-learn library

41 (TN)	10 (FP)
5 (FN)	15 (TP)

- **SVM**

Table 44 - Results of the algorithm SVM in Iteration 5, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.827	**0.834**	0.599	0.760	0.828

Table 45 - Confusion Matrix of the algorithm SVM in Iteration 5, using the Scikit-learn library

50 (TN)	3 (FP)

8 (FN)	10 (TP)

- **Random Forest**

Table 46 - Results of the algorithm RF in Iteration 5, using the Scikit-learn library

Accuracy	Precision	Recall	AUC	F1 Score
0.843	0.813	0.697	**0.808**	**0.842**

Table 47 - Confusion Matrix of the algorithm RF in Iteration 5, using the Scikit-learn library

45 (TN)	4 (FP)
7 (FN)	15 (TP)

5.2 PyCaret

Using the PyCaret library, the functions compare_models(), tune_models() and evaluate_models() were implemented.

Similar to the previous step in Scikit-learn, it was performed 10 runs for each iteration to obtain a more reliable result. The image below shows, as an example, one of the 10 runs performed for the Iteration 1 (prints for the other iterations can be found in the Appendix). The results of the metrics presented in the tables below are the average of those 10 runs.

	Model	Accuracy	AUC	Recall	Prec.	F1
lightgbm	Light Gradient Boosting Machine	0.8629	0.9111	0.7511	0.8236	0.7855
rf	Random Forest Classifier	0.8586	0.9210	0.7702	0.8010	0.7846
knn	K Neighbors Classifier	0.7772	0.8818	0.8532	0.6239	0.7202
lr	Logistic Regression	0.7772	0.8452	0.7000	0.6595	0.6778
svm	SVM - Linear Kernel	0.7472	0.0000	0.6489	0.6189	0.6304

Figure 37 – Example of one of the 10 runs performed for the Iteration 1 using the function compare_models()

5.2.1 Compare Models

Iteration 1

- **LGBM**

Table 48 - Results of the algorithm LGBM in Iteration 1, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.837	0.778	0.729	0.889	0.750

- **Logistic Regression**

Table 49 - Results of the algorithm LR in Iteration 1, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.774	0.654	0.726	0.836	0.683

- **KNN**

Table 50 - Results of the algorithm KNN in Iteration 1, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.754	0.604	**0.839**	0.854	0.698

- **SVM**

Table 51 - Results of the algorithm SVM in Iteration 1, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.746	0.612	0.683	0	0.642

- **Random Forest**

Table 52 - Results of the algorithm RF in Iteration 1, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.858	**0.812**	0.777	**0.910**	**0.790**

Iteration 2

- **LGBM**

Table 53 - Results of the algorithm LGBM in Iteration 2, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.824	0.757	0.715	0.880	0.732

- **Logistic Regression**

Table 54 - Results of the algorithm LR in Iteration 2, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.790	0.669	0.756	0.861	0.707

- **KNN**

Table 55 - Results of the algorithm KNN in Iteration 2, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.758	0.613	**0.804**	0.855	0.692

- **SVM**

Table 56 - Results of the algorithm SVM in Iteration 2, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.761	0.636	0.729	0	0.668

- **Random Forest**

Table 57 - Results of the algorithm RF in Iteration 2, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.841	**0.787**	0.732	**0.899**	**0.756**

Iteration 3

- **LGBM**

Table 58 - Results of the algorithm LGBM in Iteration 3, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.844	0.767	**0.777**	0.895	**0.771**

- **SVM**

Table 59 - Results of the algorithm SVM in Iteration 3, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.773	0.640	0.721	0	0.670

- **Random Forest**

Table 60 - Results of the algorithm RF in Iteration 3, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.847	**0.785**	0.753	**0.900**	0.768

Iteration 4

- **LGBM**

Table 61 - Results of the algorithm LGBM in Iteration 4, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.844	0.773	0.777	0.899	0.772

- **SVM**

Table 62 - Results of the algorithm SVM in Iteration 4, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.751	0.632	0.701	0	0.657

- **Random Forest**

Table 63 - Results of the algorithm RF in Iteration 4, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.854	**0.781**	**0.796**	**0.908**	**0.787**

Iteration 5

- **LGBM**

Table 64 - Results of the algorithm LGBM in Iteration 5, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.841	0.772	0.758	0.910	0.763

- **SVM**

Table 65 - Results of the algorithm SVM in Iteration 5, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.792	0.659	**0.790**	0	0.718

- **Random Forest**

Table 66 - Results of the algorithm RF in Iteration 5, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.850	**0.800**	0.759	**0.928**	**0.771**

Iteration 6

- **LGBM**

Table 67 - Results of the algorithm LGBM in Iteration 6, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.839	0.761	0.765	0.897	0.762

- **SVM**

Table 68 - Results of the algorithm SVM in Iteration 6, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.782	0.678	0.700	0	0.679

- **Random Forest**

Table 69 - Results of the algorithm RF in Iteration 6, using the PyCaret library

Accuracy	Precision	Recall	AUC	F1 Score
0.855	**0.786**	**0.785**	**0.912**	**0.784**

5.2.2 Tune Models

In PyCaret, after performing the previous iterations to compare models and features, the function tune_model() was used. For each iteration, the input of this function is the model with the best F1 Score of the respective previous iteration. This function had the goal of tuning the hyperparameters to try to obtain better results. It was also performed 10 runs for each iteration to get a more reliable result.

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8526	0.9136	0.8269	0.7544	0.7890
1	0.8654	0.9240	0.7925	0.8077	0.8000
2	0.8645	0.8895	0.7500	0.8298	0.7879
Mean	0.8608	0.9090	0.7898	0.7973	0.7923
Std	0.0058	0.0145	0.0315	0.0316	0.0055

Figure 38 - Output of the function tune_model()

Iteration 1

Table 70 - Results of model tuning in Iteration 1

Accuracy	Precision	Recall	AUC	F1 Score
0.857	0.792	0.780	0.902	**0.793**

Iteration 2

Table 71 - Results of model tuning in Iteration 2

Accuracy	Precision	Recall	AUC	F1 Score
0.858	**0.795**	0.777	0.905	0.784

Iteration 3

Table 72 - Results of model tuning in Iteration 3

Accuracy	Precision	Recall	AUC	F1 Score
0.848	0.774	0.791	0.888	0.777

Iteration 4

Table 73 - Results of model tuning in Iteration 4

Accuracy	Precision	Recall	AUC	F1 Score
0.849	0.778	0.782	0.906	0.777

Iteration 5

Table 74 - Results of model tuning in Iteration 5

Accuracy	Precision	Recall	AUC	F1 Score
0.839	0.761	0.761	0.898	0.760

Iteration 6

Table 75 - Results of model tuning in Iteration 6

Accuracy	Precision	Recall	AUC	F1 Score
0.855	0.780	**0.801**	**0.907**	0.788

5.2.3 Evaluate Models

After tuning the model's hyperparameters and having the best possible model, some graphs were plotted using the function evaluate_model(). The graphs plotted were AUC, Confusion Matrix, and Feature Importance, for each one of the iterations.

Iteration 1

- AUC

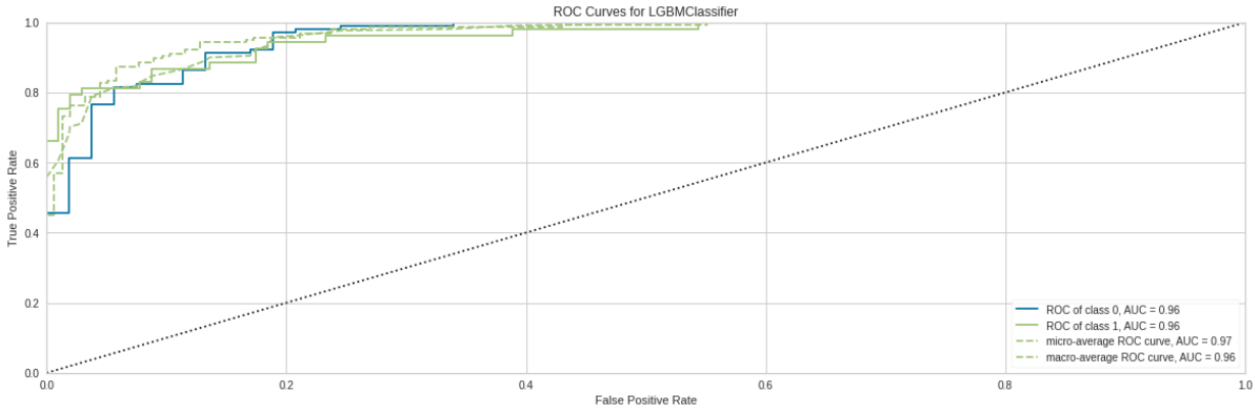


Figure 39 - AUC Plot for Iteration 1

- Confusion Matrix

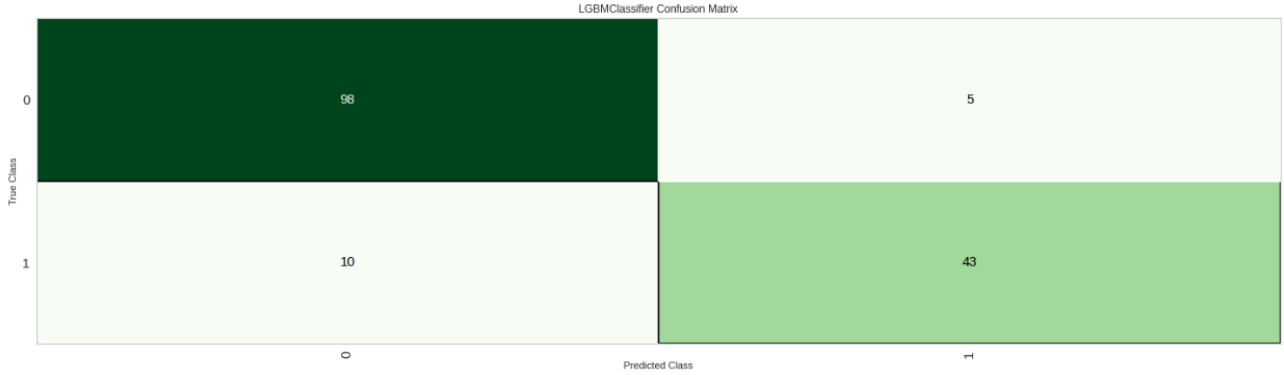


Figure 40 - Confusion Matrix for Iteration 1

- Feature Importance

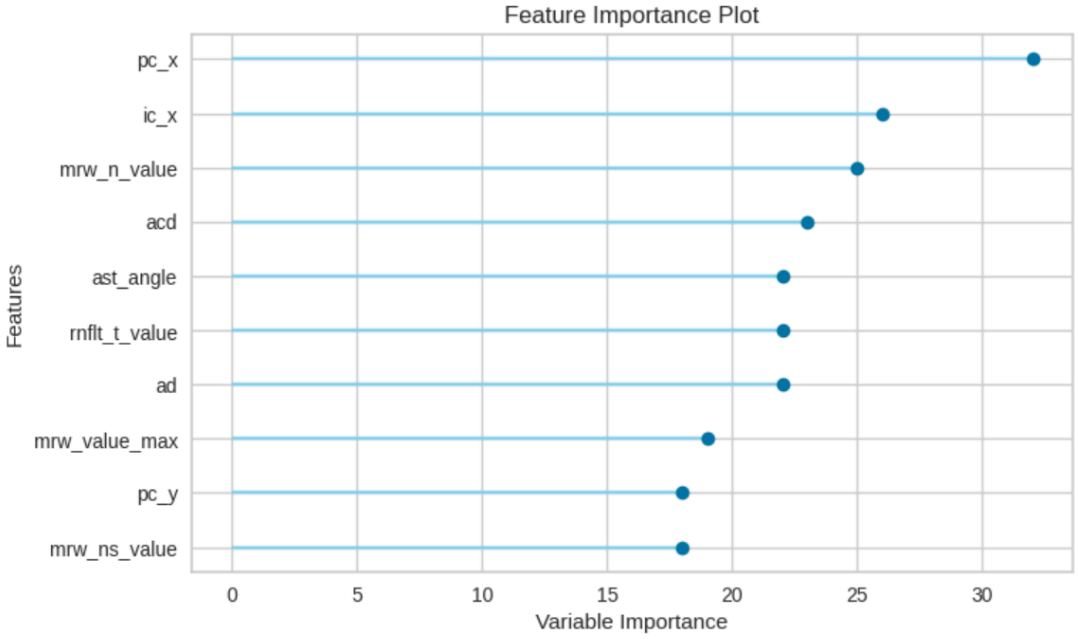


Figure 41 - Feature Importance Plot for Iteration 1

Iteration 2

- AUC

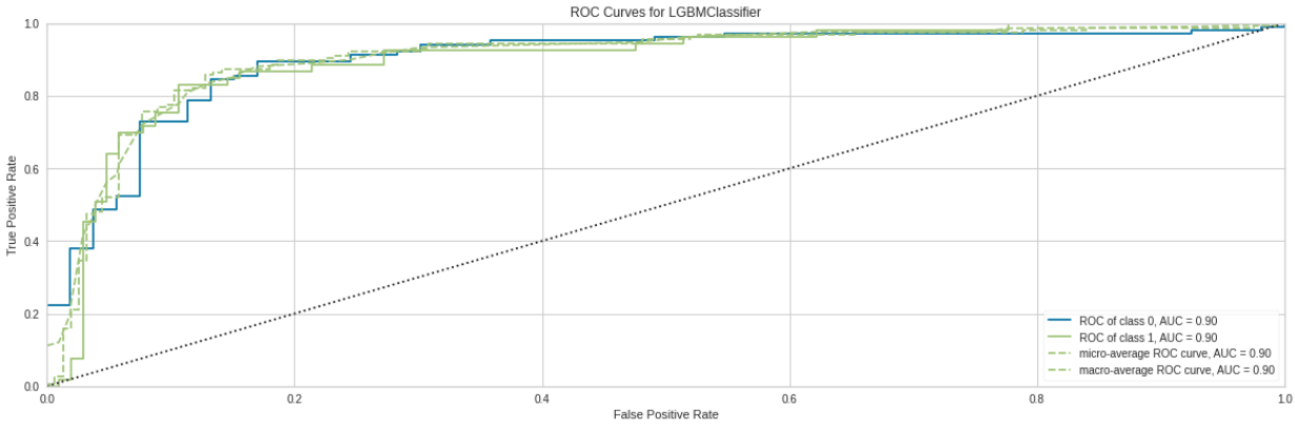


Figure 42 - AUC Plot for Iteration 2

- Confusion Matrix

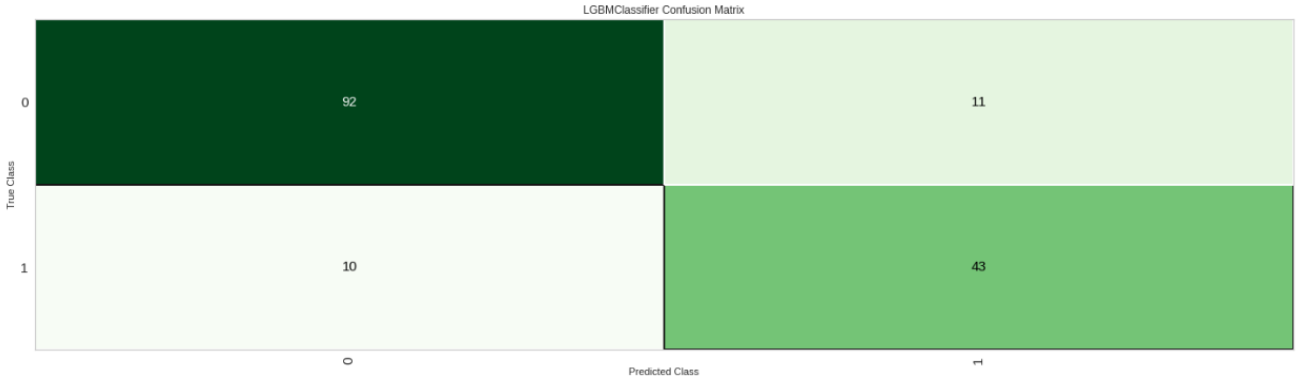


Figure 43 - Confusion Matrix for Iteration 2

- Feature Importance

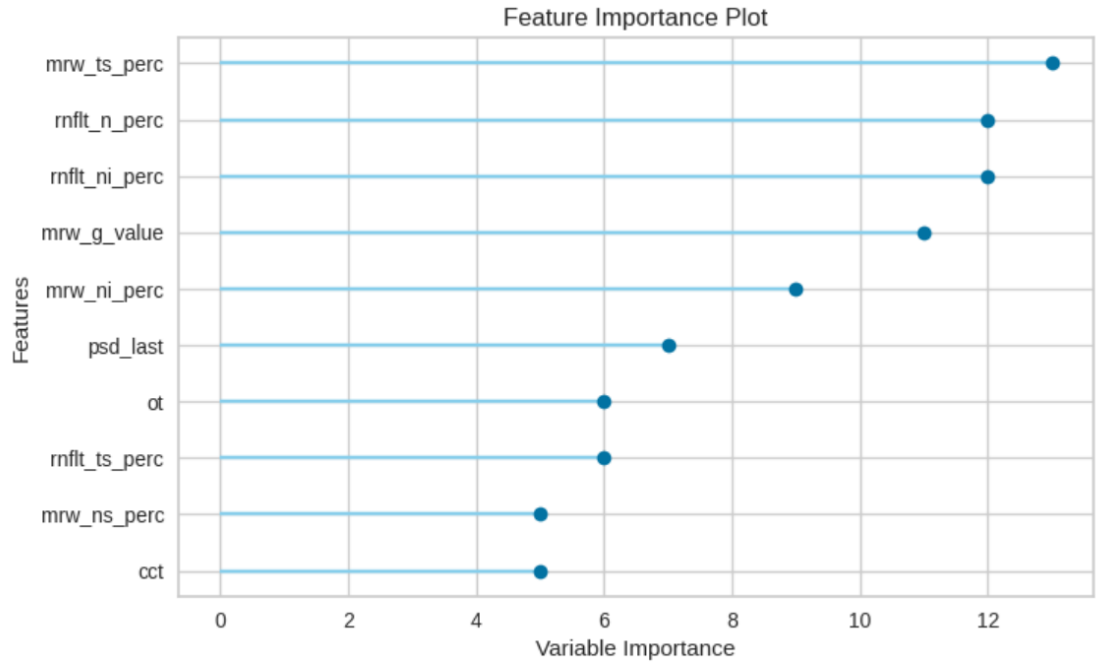


Figure 44 - Feature Importance Plot for Iteration 2

Iteration 3

- AUC

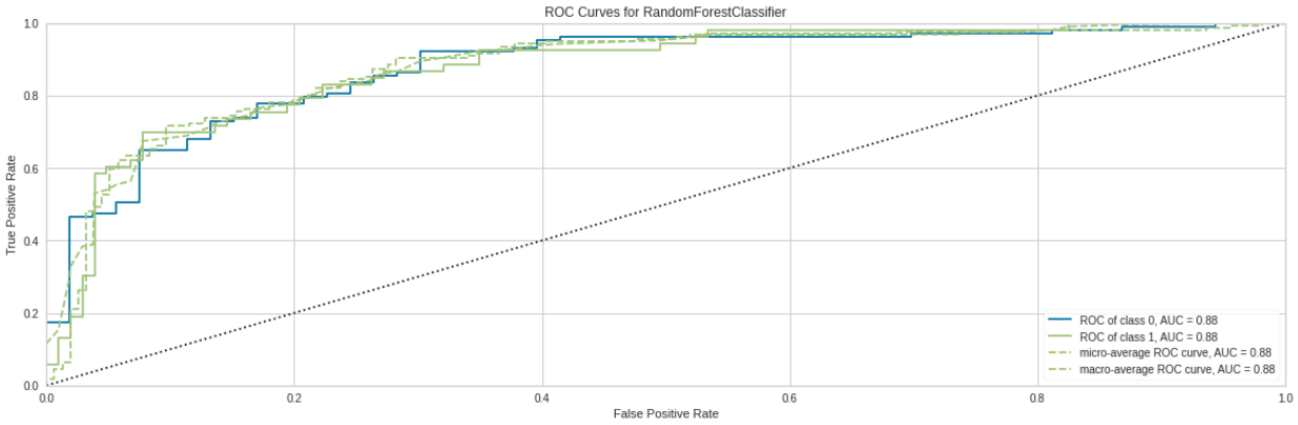


Figure 45 - AUC Plot for Iteration 3

- Confusion Matrix

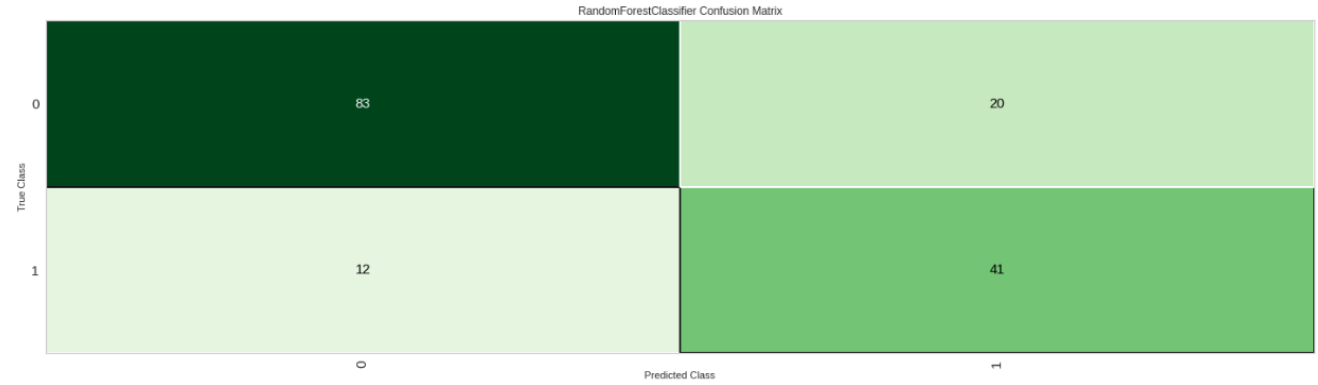


Figure 46 - Confusion Matrix for Iteration 3

- Feature Importance

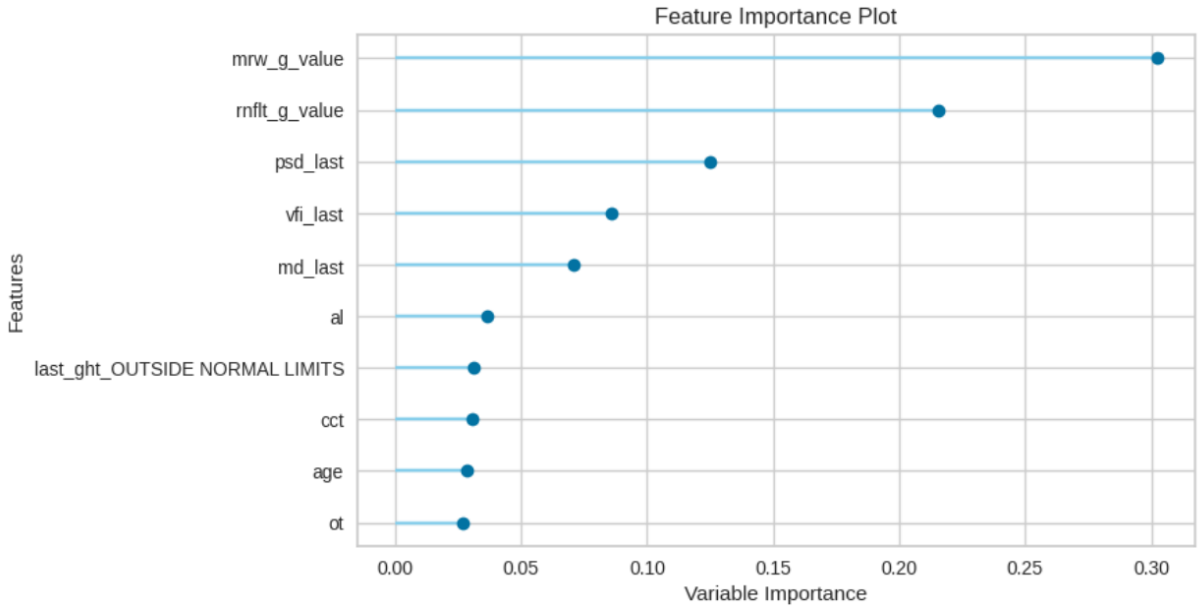


Figure 47 - Feature Importance Plot for Iteration 3

Iteration 4

- AUC

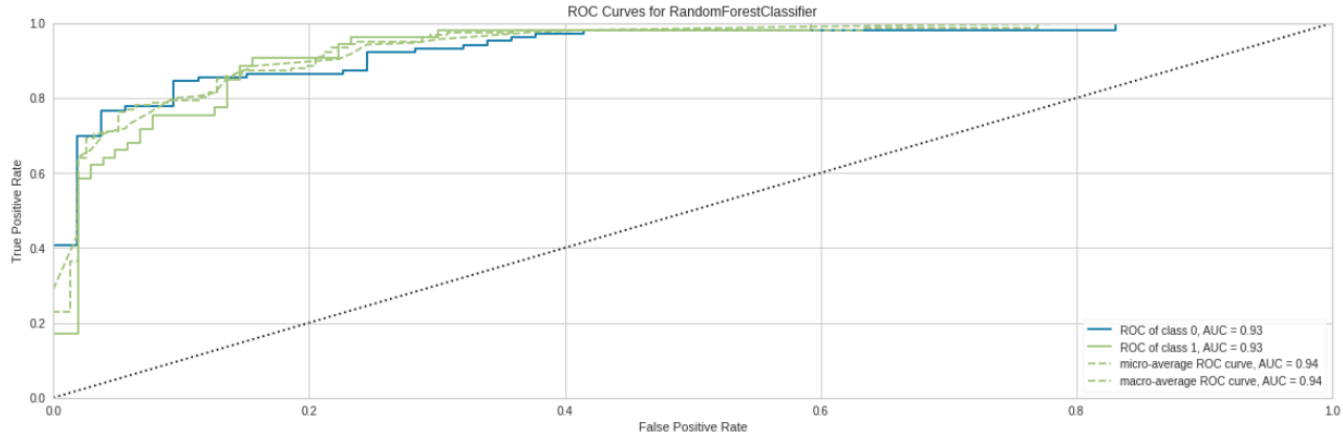


Figure 48 - AUC Plot for Iteration 4

- Confusion Matrix

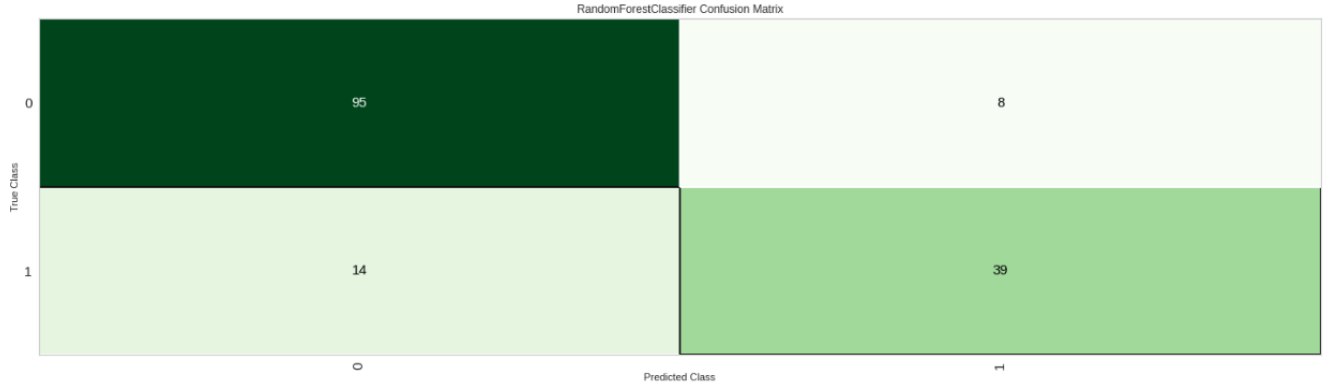


Figure 49 - Confusion Matrix for Iteration 4

- Feature Importance

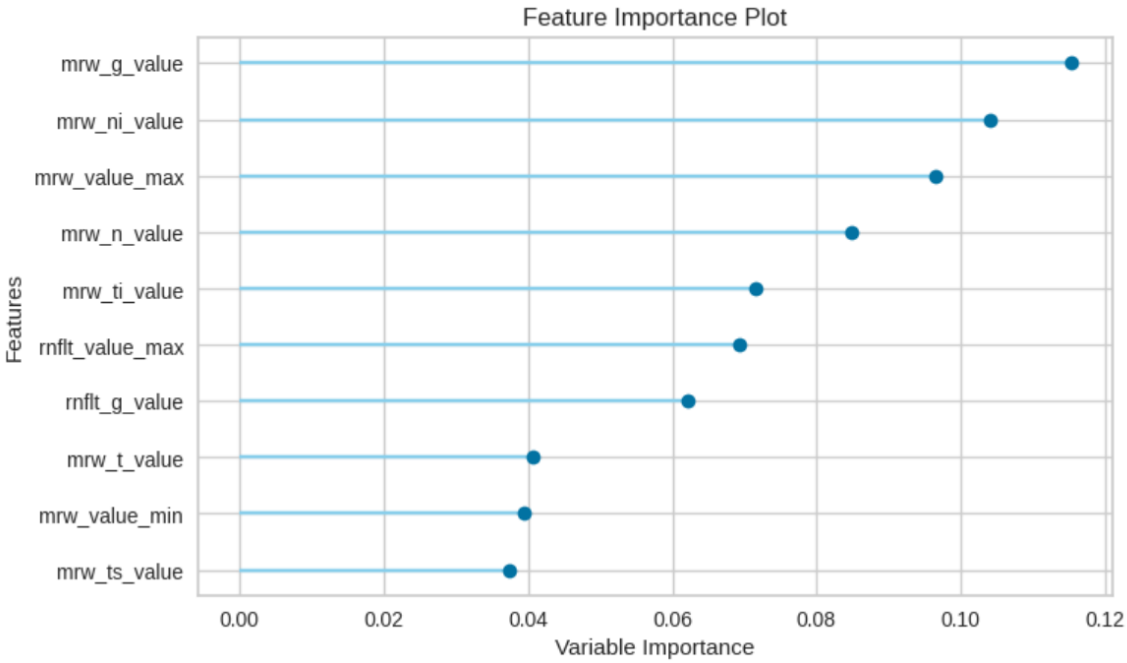


Figure 50 - Feature Importance Plot for Iteration 4

Iteration 5

- AUC

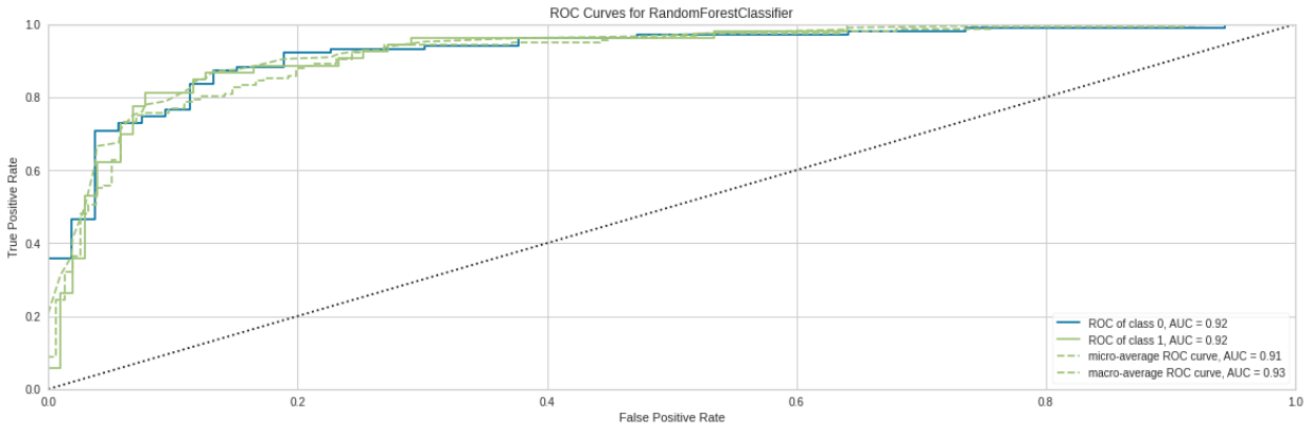


Figure 51 - AUC Plot for Iteration 5

- Confusion Matrix

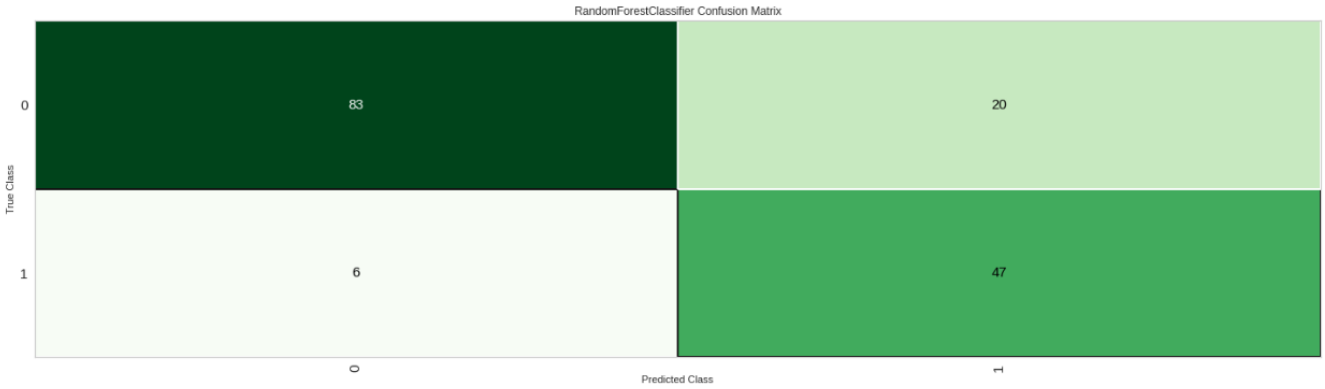


Figure 52 - Confusion Matrix for Iteration 5

- Feature Importance

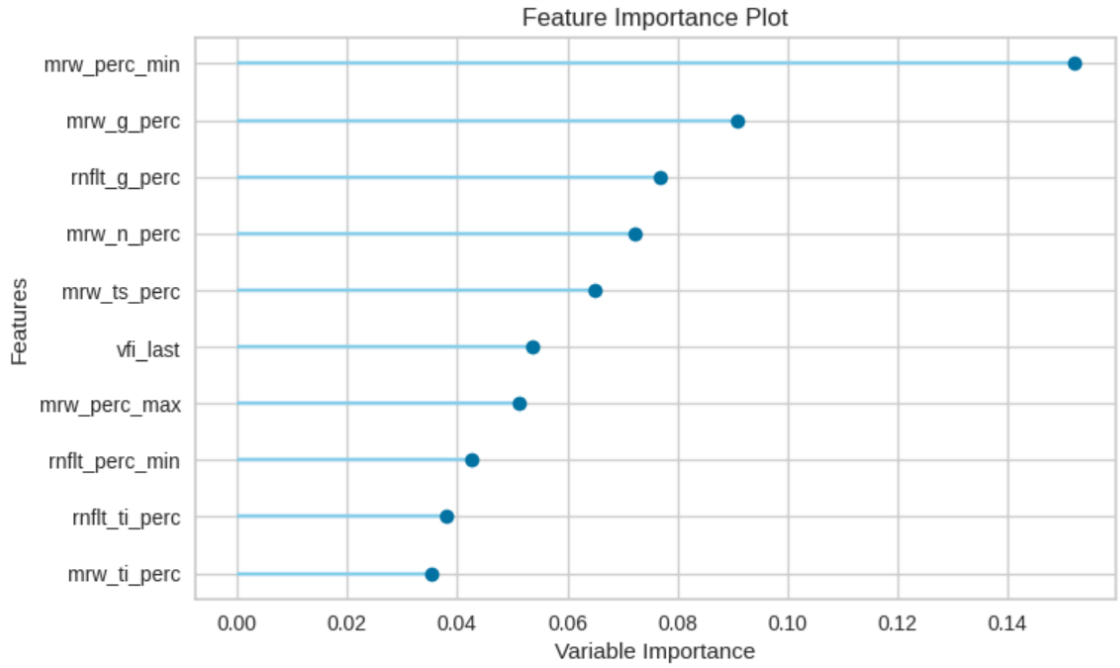


Figure 53 - Feature Importance Plot for Iteration 5

Iteration 6

- AUC

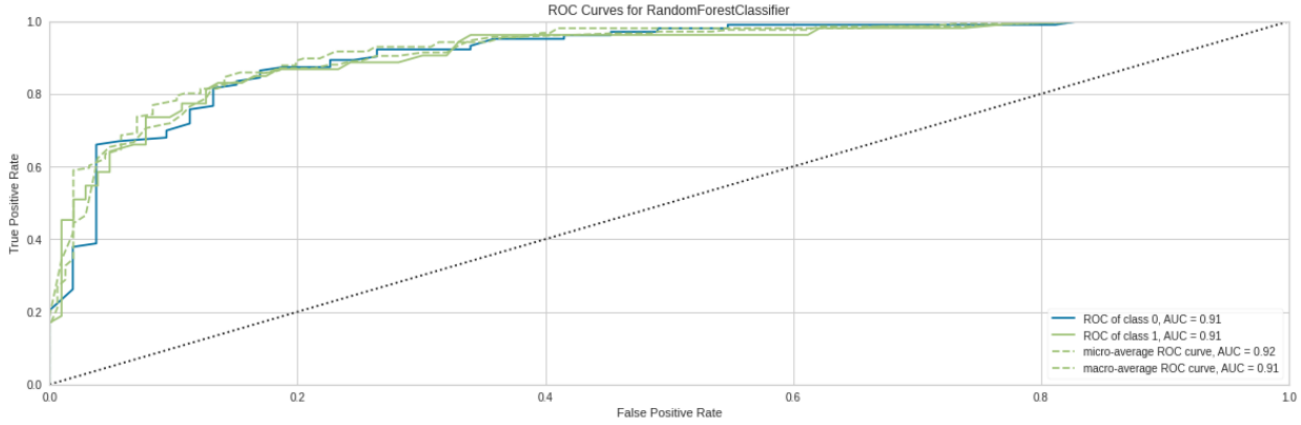


Figure 54 - AUC Plot for Iteration 6

- Confusion Matrix

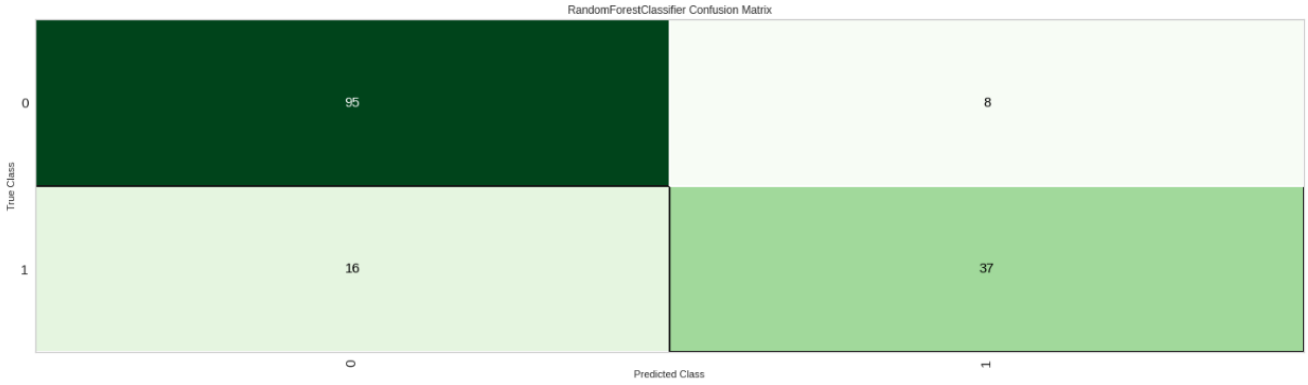


Figure 55 - Confusion Matrix for Iteration 6

- Feature Importance

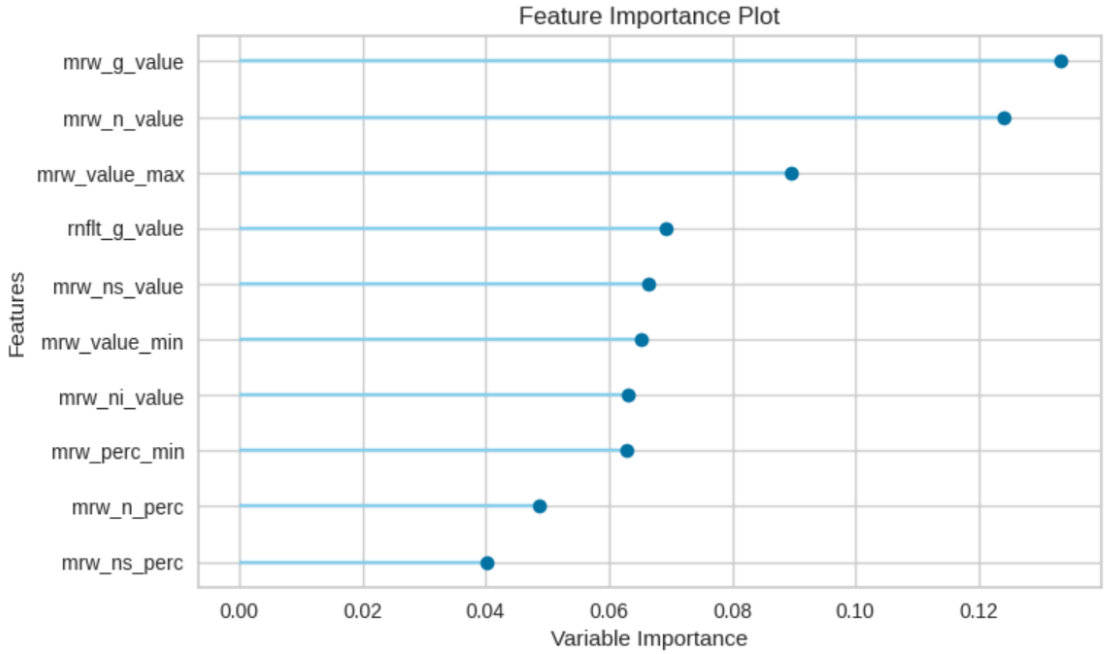


Figure 56 - Feature Importance Plot for Iteration 6

After a careful analysis of the results, it is possible to draw some conclusions:

- Random Forest is the best classified algorithm, according to the metric F1 Score. However, when it comes to Recall, it doesn't get as good results as when compared to other algorithms.
- LGBM algorithm obtained good results in Recall.
- LGBM was the algorithm with less false-negative results.
- According to the Feature Importance Plot, Mrw_g_value is the most important feature.
- Methods of Feature Importance and Feature Selection applied to the iterations 1, 2, 3 and 6 did not turn into better results.
- The comparison of the iterations 4 and 5 showed that a set of features containing absolute values obtained better results than a set of features containing percentile values, however the difference was not that significative.
- Tuning the model normally improves the results, however in some of the iterations that did not happen.

6. Discussion

In this section it will be discussed the results presented in the previous section.

One of the main goals of this work was to test different sets of features in Model Training to evaluate which would produce better results. In that sense, iterations 1, 2, 3 and 6 were analyzed. The first iteration, containing all the available features, served as a baseline for the other iterations. In the second iteration it was performed Feature Selection and some of the features containing an excessive number of nulls were removed. In the third iteration Feature Selection was also performed and features that normally are used by the staff of CCC to identify Glaucoma were selected. Finally, in the sixth iteration, techniques of Feature Importance were applied and the most important features to identify Glaucoma according to those techniques were selected. Comparing the iterations 2, 3 and 6 with the iteration 1, it would be expected to obtain better results, due to the application of Feature Selection and Feature Importance techniques. However, that was not the case. After an individual consideration and a discussion with the Altice Labs team, it was agreed that the data available for this study was not enough. In a future work it would be necessary to gather more medical examinations to draw this type of conclusions. The low amount of data was identified as the main obstacle for the realization of this project.

With the remaining iterations that were not included in the previous discussion, iterations 4 and 5, the goal of its analysis was also to evaluate which one would produce better results. Iteration 4 utilized a dataset with features that had only absolute values, whereas iteration 5 used a dataset with features that had only percentile values. Comparing the results of the two iterations, it can be concluded that iteration 4 produced better results, indicating that absolute values are more crucial for the early detection of glaucoma than percentile values. The analysis of Feature Importance graphs also reveals that these graphs contain more features with absolute value than those with percentile value. It is crucial to note that the results from these two iterations did not differ that significantly from one another, and it would be wise to collect more information before coming to any more detailed inferences.

Another major goal of this project was to test various algorithms to see which produced the best results. In this regard, five different algorithms were chosen, and after comparing their results, it was determined that the algorithm Random Forest was the best ranked. Although F1 Score was chosen as the evaluation metric to rank these models, Random Forest consistently outperformed the other algorithms in other metrics. Despite being the best ranked algorithm in almost all the metrics, it didn't

get as good results in Recall, one of the most important metrics regarding the medical field. As it was discussed in the Model Assessment section, it was prioritized to minimize false-negative results to do not let any positive cases go unidentified. That being said, the algorithm LGBM should also be considered, since it was the best ranked in terms of Recall.

As previously said, understanding the topic at hand is critical for analyzing and sharing Machine Learning outcomes. In the medical industry, it is emphasized to reduce false-negative findings since the main aim is to not miss any positive instances and notice them as soon as feasible for treatment. Early identification and treatment of Glaucoma are critical for reducing the risk of vision loss. Following an examination of the Confusion Matrixes of each iteration, it was determined that LGBM was the algorithm with the fewest false-negative outcomes. At all the reasons stated in this section, it was determined that the algorithms Random Forest and LGBM were the best for detecting glaucoma.

7. CONCLUSION

In this section the conclusion of this dissertation will be presented, where it will be reflected all the work done and address future work and difficulties experienced.

During this document, new technologies were explored in order to meet patients' needs. With population aging becoming a demographic trend worldwide, with conventional diagnosis methods depending on professional's experience, and with the amount of image data generated rapidly rising, it is more important than ever to have techniques that can analyze and process this data.

Fortunately, the exponential progress of Artificial Intelligence over the previous few years allows us to remain optimistic about this situation. The purpose of this effort was to use AI technology to ophthalmology to enhance the detection and treatment of ocular illnesses.

To achieve this goal, a project was carried out, where a typical ML methodology was followed and tasks such as collecting and transforming the data, training and testing the model, and evaluating the achieved results, were performed. The document was organized according to the methodology in discussion and in each section the work done was addressed. After that, the results were presented and discussed.

With the findings acquired, various conclusions could be drawn, such as which methods and features to utilize. However, additional data would be required in a future effort to reach more developed findings. One of the most significant challenges in Machine Learning projects is lack of data, which causes issues such as sampling bias, missing values, and even overfitting, in which the training model memorizes the patterns. A specific quantity of data is necessary for efficient training and performance of an ML model. A project cannot be completed successfully if the data is insufficient. In this research, the absence of data was evident when iterations 2, 3, and 6 did not produce better outcomes than iteration 1 despite what would be predicted conceptually.

Early identification and intervention are critical in the treatment of ocular illnesses in ophthalmology. The earlier a condition is detected, the higher the odds of saving the vision. Medical professionals must collaborate with engineers and developers to obtain a better knowledge of technology's capabilities and use Artificial Intelligence into their diagnostics to enhance them. With this cross-disciplinary collaboration, the addition of intelligence into ophthalmic equipment may improve patient care in the future. The work done in this dissertation aimed to bring closer these two distinct fields.

Finally, choosing to carry out the dissertation as a trainee in a company was the best decision. By working as a trainee in Altice Labs I had the chance to work with a team of experienced members and I

had the opportunity to get to know and test new technological tools. Consequently, I was able to improve both my soft and hard skills in a way that would not be possible without the internship.

BIBLIOGRAPHY

Abramoff M., Garvin M. & Sonka M. (2010). Retinal Imaging and Image Analysis. IEEE reviews in biomedical engineering, 3, pp. 169. https://doi.org/10.1109/RBME.2010.2084567

Al-Aswad, L. A., Kapoor, R., Chu, C. K. (2019). Evaluation of a Deep Learning System for Identifying Glaucomatous Optic Neuropathy Based on Color Fundus Photographs. J Glaucoma. https://doi.org/10.1097/IJG.0000000000001319

Artificial Intelligence in Glaucoma. (2022). Retrieved February 9, 2022, from https://eyewiki.aao.org/Artificial_Intelligence_in_Glaucoma

Artificial Intelligence. (2017). Retrieved November 16, 2021, from https://www.aao.org/eyenet/article/artificial-intelligence

Asaoka, R., Murata, H., Hirasawa, K. (2019). Using Deep Learning and Transfer Learning to Accurately Diagnose Early-Onset Glaucoma From Macular Optical Coherence Tomography Images. Am J Ophthalmol. pp.136-145. https://doi.org/10.1016/j.ajo.2018.10.007

Asaoka, R., Murata, H., Iwase, A., Araie, M. (2016). Detecting pre-perimetric glaucoma with standard automated perimetry using a deep learning classifier. Ophthalmology, 123(9), pp. 1974–1980. https://doi.org/10.1016/J.OPHTHA.2016.05.029

Cerentini, A., Welfer, D., Cordeiro, M., Pereira, C., Dotto, G. (2017). Automatic Identification of Glaucoma Using Deep Learning Methods. Stud Health Technol Inform, 245, pp. 318-321.

Choi, J. Y., Yoo, T. K., Seo, J. G., Kwak, J., Um, T. T., Rim, T. H. (2017). Multi-categorical deep learning neural network to classify retinal images. PLoS One, 12(11). https://doi.org/10.1371/JOURNAL.PONE.0187336

Christopher, M., Belghith, A., Weinreb, R. (2018). Retinal Nerve Fiber Layer Features Identified by Unsupervised Machine Learning on Optical Coherence Tomography Scans Predict Glaucoma Progression. Invest Ophthalmol Vis Sci, 59(7), pp. 2748-2756. https://doi.org/10.1167/iovs.17-23387

Confirming a Questionable Glaucoma Diagnosis. (2008). Retrieved November 3, 2021, from https://www.ophthalmologymanagement.com/supplements/2008/october-2008/experiencing-high-definition-spectral-domain-optic/confirming-a-questionable-glaucoma-diagnosis

Haleem, M., Han, L., Hemert, J. (2017). A Novel Adaptive Deformable Model for Automated Optic Disc and Cup Segmentation to Aid Glaucoma Diagnosis. J Med Syst. 42(1) https://doi.org/10.1007/s10916-017-0859-4

Hicks, S., Strümke, I., Thambawita, V., Hammou, M., Riegler, M., Halvorsen, P., Parasa, S. (2022). On evaluation metrics for medical applications of artificial intelligence. Scientific Reports, 12(1), pp. 1-9. https://doi.org/10.1038/s41598-022-09954-8

Introduction to Artificial Intelligence in Ophthalmology. (2020). Retrieved November 16, 2021, from https://eyewiki.aao.org/Introduction_to_Artificial_Intelligence_in_Ophthalmology

Li, F., Wang, Z., Qu, G. (2018). Automatic differentiation of Glaucoma visual field from non-glaucoma visual filed using deep convolutional neural network. BMC Med Imaging, 18(1), pp. 35. https://doi.org/10.1186/s12880-018-0273-5

Lu, W., Tong, Y., Yu, Y. & Shen, Y. (2020). Application of machine learning in ophthalmic imaging modalities. Eye and Vision, 7(1), pp. 1-15. https://doi.org/10.1186/S40662-020-00183-6

Lu, W., Tong, Y., Yu, Y., Xing, Y., Chen, Changzheng & Shen, Y. (2018). Applications of Artificial Intelligence in Ophthalmology: General Overview. Journal of Ophthalmology. https://doi.org/10.1155/2018/5278196

Muhammad, H., Fuchs, T., De Cuir N. (2017). Hybrid Deep Learning on Single Wide-field Optical Coherence tomography Scans Accurately Classifies Glaucoma Suspects. J Glaucoma, 26(12), pp.1086-1094. https://doi.org/10.1097/IJG.0000000000000765

Nriagu, J. (2019). Encyclopedia of Environmental Health, 6(2).

Resnikoff, S., Felch, W., Gauthier, T., Spivey, B. (2012). The number of ophthalmologists in practice and training worldwide: a growing gap despite more than 200,000 practitioners. Br J Ophthalmol, 96, pp. 783-787. https://doi.org/10.1136/BJOPHTHALMOL-2011-301378

Tham, Y., Li, X., Wong, T., Quigley, H., Aung, T., Cheng, C. (2014). Global prevalence of glaucoma and projections of glaucoma burden through 2040: a systematic review and meta-analysis. Ophthalmology, 121, pp. 2081-2090. https://doi.org/10.1016/J.OPHTHA.2014.05.013

Ting, D., Ang, M., Mehta, J. (2019). Artificial intelligence-assisted telemedicine platform for cataract screening and management: a potential model of care for global eye health. Br J Ophthalmol, 103(11), pp. 1537-1538. https://doi.org/10.1136/bjophthalmol-2019-315025

Topol, E. J. (2019). High-performance medicine: the convergence of human and artificial intelligence. Nature Medicine, 25(1), pp. 44-56. https://doi.org/10.1038/s41591-018-0300-7

APPENDIX

Entities

Here can be found a set of tables containing an individually description of all the entities used in the diagram. The tables are composed of Field, Type, Description, Example and Possible Values.

patients

List of all patients with demographic information

Field	Type	Description	Example	Possible Values
id	Bigint	Patient's ID	2134087	
birth_date	Date	Birthday date	08-07-1954	
gender	Text	Patient's gender	M	M F

pathologies

List of possible pathologies

Field	Type	Description	Example	Possible Values
id	Bigint	Diagnostic identifier	0	0 to 30
label	Text	Diagnostic name	Normal	"Normal" "Outras Patologias" "Glaucoma" "Suspeito de Glaucoma" "Edema" "Edema NO" "Edema

				Macular'' ''Descolamento'' ''Descolamento Seroso'' ''Pos Descolamento'' ''Buraco Lamelar'' ''Buraco Macular'' ''Pos Buraco Macular'' ''Membrana epi'' ''Pos Membrana'' ''Patologia NO'' ''Maculopatia'' ''Miope'' ''Miope Retinopatia'' ''Alteraçoes Perifericas Retinogra'' ''RD'' ''DMRI'' ''Miopia elevada'' ''Catarata'' ''Atrofiamento Peripapilar'' ''Opacidade Corneana''

				"Pós Vitrectomia" "Queratocone" "Benson Disease" "Opacificação Capsular" "Hemorragia Subhialoideia Macular"

p_diags

Diagnostic information by patient eye

Field	Type	Description	Example	Possible Values
patient_id	Bigint	Patient's Identifier	2134087	
location	Text	Associated eye	Left	Left Right
diagnostic_id	Bigint	Diagnostic ID	2	0 to 30
delivery_id	Bigint	Diagnosis/exam delivery ID	3	0 to 5

deliveries

Delivery dates, for data versioning control

Field	Type	Description	Example	Possible Values
id	Int	Delivery Identifier	2	0 to 5
delivery_date	Date	Delivery date	2021-07-27	

e_bio

This table contains the data from biometry exams, in the most common format. One record per patient/eye.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's ID	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry	Biometry
exam_date	Date	Exam date	17-10-2014	
exam_time	String	Exam time	13:10	
exam_duration	String	Exam duration in minutes	1 Min	
al	Double	Axial length (mm) - measured from the corneal tear film to the inner limiting membrane	22,46	
cct	Double	Cornea thickness (µm) - measured form the corneal tear film to the corneal endothelium	538	
ad	Double	Aqueous depth (mm) - measured form the corneal	2,05	

		endothelium to the anterior surface of the lens		
acd	Double	Anterior chamber depth (mm) - measured from cornea posterior to the iris (anterior surface of the lens). Is important in estimating the risk of angle closure glaucoma.	2,59	
lt	Double	Lens thickness (mm) - measured form the anterior to the posterior surface of the lens. Specifically in pseudo phakic condition, measurement of implanted lens is not always achievable (e.g., due to light lens	4,50	

		tilt and the non-light scattering properties of an implant).		
rt	Double	Retina thickness (µm) (System constant) - measured only manually by adjusting the measurement gate in the A-scan from the inner limiting membrane to the retinal pigment epithelium.	200	
K1_1	Double	Flat meridian 1 (D) - K1 and K2 are the two major meridians, determined using the 3mm ring, are 90 degrees from each other.	45,04	
K1_2	Double	Flat meridian 2 (°) - the degree where the flat meridian was	173	

		determined. It differs from k2° in 90 degrees.		
k2_1	Double	Steep meridian 1 (D) - K1 and K2 are the two major meridians, determined using the 3mm ring	46,26	
k2_2	Double	Steep meridian 2 (°) - the degree where the flat meridian was determined. It differs from k1° in 90 degrees.	83	
ast_1	Double	Astigmatism 1 (D) - displayed in Diopters either in plus or minus cylinder notation dependent on the display option chosen in the Biometry preferences	1,23	
ast_2	Double	Astigmatism 2 (°)	83	
n	Double	Keratometric index - displays	1,3375	

		the keratometric index chosen for this measurement to convert corneal curvature into corneal power		
wtw	Double	White to White (mm) - measured as the horizontal diameter of a best fit circle to the iris boarder	11,31	
Ic_1	Double	Iris barycenter 1 (mm) - provides information of the shift of the iris center relative to the apex in X and Y coordinates	-0,27	
Ic_2	Double	Iris barycenter 2 (mm) - provides information of the shift of the iris center relative to the apex in X and Y coordinates	-0,25	
pd	Double	Pupil diameter	3,62	

		(mm) - measured at the diameter of a best fit circle to the pupil boarder		
pc_x	Double	Pupil barycenter X (mm) - provides information of the shift of the pupil center relative to the apex in X coordinate	-0,16	
pc_y	Double	Pupil barycenter Y (mm) - provides information of the shift of the pupil center relative to the apex in Y coordinate	-0,18	
delivery_date	Int	Delivery date	3	0 to 5

e_bio_2

This table contains the data from the biometry exams, in IOL format. One record per patient/eye.

Field	Type	Description	Example	Possible Values
Patient_id	Int	Patient's	2134087	

		Identifier		
Eye	String	Patient´s eye	Right	Left Right
Exam_type	String	Exam type	Biometry_2	Biometry_2
Exam_date	Date	Exam date	17-10-2014	
Exam_time	String	Exam time	13:10	
Exam_duration	String	Exam duration	1 Min	
al	Double	Axial length (mm) - measured from the corneal tear film to the inner limiting membrane	22,46	
cct	Double	Cornea thickness (µm) - measured form the corneal tear film to the corneal endothelium	538	
ad	Double	Aqueous depth (mm) - measured form the corneal endothelium to the anterior surface of the lens	2,05	
acd	Double	Anterior chamber depth	2,59	

		(mm) - measured from cornea posterior to the iris (anterior surface of the lens). Is important in estimating the risk of angle closure glaucoma.		
lt	Double	Lens thickness (mm) - measured from the anterior to the posterior surface of the lens. Specifically in pseudo phakic condition, measurement of implanted lens is not always achievable	4,50	
r1_1	Double	mm	8,30	
r1_2	Double	D	40,67	
r1_3	Double	°	179	
r2_1	Double	mm	8,21	
r2_2	Double	D	41,11	
r2_3	Double	°	89	
r_1	Double	mm	8,25	

r_2	Double	D	40,89	
ast_1	Double	Astigmatism 1 (D) - displayed in Diopters either in plus or minus cylinder notation dependent on the display option chosen in the Biometry Preferences	1,23	
ast_2	Double	Astigmatism 2 (°)	83	
n	Double	Keratometric index - displays the keratometric index chosen for this measurement to convert corneal curvature into corneal power	1,3375	
wtw	Double	White to White (mm) - measured as the horizontal diameter of a best fit circle to the iris boarder	11,31	
delivery_date	Int	Delivery date	3	0 to 5

e_bio_3

This table contains the biometry exam data, in Cataract Premium IOL format. One record per patient/eye.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_3	Biometry_3
exam_date	Date	Exam date	17-10-2014	
simk_avg	Double	D	45,25	
simk_steep_1	Double	Steep meridian value (D)	43,08	
simk_steep_2	Double	Steep meridian angle (°)	14	
simk_flat_1	Double	Flat meridian value (D)	41,46	
simk_flat_2	Double	Flat meridian angle (°)	104	
ast_steep_1	Double	Astigmatism value (D)	1,62	
ast_steep_2	Double	Astigmatism angle (°)	14	
Ast_total_1	Double	D	2,00	
ast_total_2	Double	°	14	
ast_post_1	Double	D	-0,14	
ast_post_2	Double	°	94	
ast_delta_1	Double	D	-0,37	
ast_delta_2	Double	°	0	
z	Double		0,02	
rms_hoa	Double		0,09	

cct	Double	Cornea thickness (µm)	529	
aqd	Double	mm	4,04	
cct_aqd	Double	mm	3,55	
wtw	Double	White to White (mm) - measured as the horizontal diameter of a best fit circle to the iris boarder	11,35	
lt	Double	Lens thickness (mm) - measured form the anterior to the posterior surface of the lens. Specifically, in pseudophakic condition, measurement of implanted lens is not always achievable	2,29	
pd	Double	Pupil diameter (mm) - measured at the diameter of a best fit circle to the pupil	4,1	

		boarder		
pc_1	Double	mm	0,56	
pc_2	Double	mm	154	
pc_x	Double	Pupil barycenter x-axis (mm)	-0,29	
pc_y	Double	Pupil barycenter y-axis (mm)	-0,08	
al	Double	Axial length (mm)	25,00	
delivery_id	Int	Delivery identifier	3	0 to 5

e_bio_4

This table contains the data from the biometry exams, in Bio Basic format. One record per patient/eye.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_2	Biometry_2
exam_date	Date	Exam date	17-10-2014	
exam_time	String	Exam time	13:10	
cct	Double	Cornea thickness (µm) - measured from the corneal tear film to the corneal endothelium	538	
wtw	Double	White to White	11,31	

		(mm) - measured as the horizontal diameter of a best fit circle to the iris boarder		
ad	Double	Aqueous depth (mm) - measured from the corneal endothelium to the anterior surface of the lens	2,05	
acv	Double	mm^3	150.42	
aca_dist	Double	mm	12.42	
sts_dist	Double	mm	12.27	
aca_500_1	Double	º	10	
aca_500_2	Double	º	16	
ssa_500_1	Double	º	16	
ssa_500_2	Double	º	23	
aod_500_1	Double	mm	0.15	
aod_500_2	Double	mm	0.21	
tisa_500_1	Double	mm^2	0.052	
tisa_500_2	Double	mm^2	0.095	
aca_750_1	Double	º	16	
aca_750_2	Double	º	21	
ssa_750_1	Double	º	22	
ssa_750_2	Double	º	27	
aod_750_1	Double	mm	0.31	
aod_750_2	Double	mm	0.37	

tisa_750_1	Double	mm^2	0.108	
tisa_750_2	Double	mm^2	0.164	
lt	Double	Lens thickness (mm) - measured form the anterior to the posterior surface of the lens. Specifically, in pseudophakic condition, measurement of implanted lens is not always achievable	4.44	
lv	Double	mm	0.35	
pd	Double	Pupil diameter (mm) - measured at the diameter of a best fit circle to the pupil boarder	4.9	
pc_x	Double	Pupil barycenter X (mm) - provides information of the shift of the pupil center relative to the	-0.41	

		apex in X coordinate		
pc_y	Double	Pupil barycenter Y (mm) - provides information of the shift of the pupil center relative to the apex in Y coordinate	0.09	
delivery_id	Int	Delivery Identifier	3	0 to 5

e_pec_single

This table contains the data from the PEC exams.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_2	PEC_single
exam_date	Date	Exam date	17-10-2014	
exam_time	String	Exam time	13:10	
exam_duration	String	Exam duration	1 Min	
fix_monitor	String	Fixation monitor	Gaze/Blind Spot	Blind Spot Gaze/Blind Spot Gaze Monitor Gaze Track

				OFF
fix_target	String	Fixation target	Central	
fl_1	Int	Fixation losses 1	1	
fl_2	Int	Fixation losses 2	11	
fp	Double	False positives (%)	0	
Fn	Double	False negatives (%)	2	
stimulus	String	Stimulus	III, White	
background	Double	Background (ASB)	31.5	
strategy	String	Strategy	SITA-Fast	
pupil_diameter	Double	Pupil diameter (mm)	4.2	
fovea	String	Fovea	OFF	
vfi	Double	Visual Field Index (%)	99	0 a 100
md	Double	Mean deviation (dB)	-0.71	-30,91 a 3,11
psd	Double	Pattern standard deviation (dB)	1.66	0,86 a 15,07
ght	String	Glaucoma Hemifield Test result for the	Borderline	Within Normal Limits Borderline Borderline/General

		exam		Reduction Abnormally High Sensitivity Within Normal Limits *** Low Test Reliability *** General Reduction of Sensitivity Outside Normal Limits Outside Normal Limits *** Low Test Reliability ***
Delivery_id	Int	Delivery Identifier	3	0 to 5

e_pec_summary

This table contains the data from the PEC exams.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_2	PEC_summary
exam_date	Date	Exam date	17-10-2014	
exam_order	Int	Exam order among the various exams in the file	1	1 2 3
ght	String	Glaucoma Hemifield Test result for the exam	Borderline	Within Normal Limits Borderline Borderline/General Reduction

				Abnormally High Sensitivity Within Normal Limits *** Low Test Reliability *** General Reduction of Sensitivity Outside Normal Limits Outside Normal Limits *** Low Test Reliability ***
fl_1	Int	Fixation losses 1	1	
fl_2	Int	Fixation losses 2	11	
fp	Double	False positives (%)	0	
fn	Double	False negatives (%)	2	
fovea	String	Fovea	OFF	
vfi	Double	Visual Field Index (%)	99	0 a 100
md	Double	Mean deviation (dB)	-0.71	-30,91 a 3,11
psd	Double	Pattern standard deviation (dB)	1.66	0,86 a 15,07
delivery_id	Date	Delivery_id	3	0 to 5

e_rnfl

Field	Type	Description	Example	Possible

				Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_2	RNFL
exam_date	Date	Exam date	17-10-2014	
reference	String	Reference database version	European Descent (2014)	
c_mrw	String	Classification MRM	Outside Normal Limits	Within Normal Limits Borderline Outside Normal Limits
mrw_ts_1	Double	MRW TS 1	286	
mrw_ts_2	Double	MRW TS 2 (%)	63	
mrw_ns_1	Double	MRW NS 1	354	
mrw_ns_2	Double	MRW NS 2 (%)	72	
mrw_t_1	Double	MRW T 1	204	
mrw_t_2	Double	MRW T 2 (%)	59	
mrw_g_1	Double	MRW G 1	313	
mrw_g_2	Double	MRW G 2 (%)	76	

mrw_n_1	Double	MRW N 1	345	
mrw_n_2	Double	MRW N 2 (%)	79	
mrw_ti_1	Double	MRW TI 1	365	
mrw_ti_2	Double	MRW TI 2 (%)	82	
mrw_ni_1	Double	MRW NI 1	402	
mrw_ni_2	Double	MRW NI 2 (%)	77	
c_rnflt	String	Classification RNFLT	Within Normal Limits	Within Normal Limits Borderline Outside Normal Limits No Classification Possible
rnflt_ts_1	Double	RNFLT TS 1	120	
rnflt_ts_2	Double	RNFLT TS 2 (%)	34	
rnflt_ns_1	Double	RNFLT NS 1	79	
rnflt_ns_2	Double	RNFLT NS 2 (%)	23	
rnflt_t_1	Double	RNFLT T 1	62	
rnflt_t_2	Double	RNFLT T 2 (%)	35	

rnflt_g_1	Double	RNFLT G 1	86	
rnflt_g_2	Double	RNFLT G 2 (%)	47	
rnflt_n_1	Double	RNFLT N 1	65	
rnflt_n_2	Double	RNFLT N 2 (%)	33	
rnflt_ti_1	Double	RNFLT TI 1	148	
rnflt_ti_2	Double	RNFLT TI 2 (%)	75	
rnflt_ni_1	Double	RNFLT NI 1	112	
rnflt_ni_2	Double	RNFLT NI 2 (%)	92	
delivery_id	Int	Delivery Identifier	3	0 to 5

e_rnfl_ou

This table contains the data from the RNFL exams.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_2	RNFL
exam_date	Date	Exam date	17-10-2014	
reference	String	Reference database version	European Descent (2014)	

rnflt_s	Double	RNFLT sector Superior	116	
rnflt_n	Double	RNFLT sector Nasal	77	
rnflt_i	Double	RNFLT sector Inferior	153	
rnflt_t	Double	RNFLT sector Temporal	61	
c_rnflt	String	Classification RNFLT	Borderline	Within Normal Limits Borderline Outside Normal Limits No Classification Possible
rnflt_ts_1	Double	RNFLT TS 1	120	
rnflt_ts_2	Double	RNFLT TS 2 (%)	34	
rnflt_ns_1	Double	RNFLT NS 1	79	
rnflt_ns_2	Double	RNFLT NS 2 (%)	23	
rnflt_t_1	Double	RNFLT T 1	62	
rnflt_t_2	Double	RNFLT T 2 (%)	35	

rnflt_g_1	Double	RNFLT G 1	86	
rnflt_g_2	Double	RNFLT G 2 (%)	47	
rnflt_n_1	Double	RNFLT N 1	65	
rnflt_n_2	Double	RNFLT N 2 (%)	33	
rnflt_ti_1	Double	RNFLT TI 1	148	
rnflt_ti_2	Double	RNFLT TI 2 (%)	75	
rnflt_ni_1	Double	RNFLT NI 1	112	
rnflt_ni_2	Double	RNFLT NI 2 (%)	92	
delivery_id	Int	Delivery Identifier	3	0 to 5

e_rnfl_cr

RNFL CR (Change Report - Follow ups)

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's ID	2134087	
eye	String	Patient´s eye	Right	Left Right
exam_type	String	Exam type	Biometry_2	RNFL
exam_date	Date	Exam date	17-10-2014	
reference	String	Reference database version	European Descent (2014)	

c_rnflt	String	Classification RNFLT	Borderline	Within Normal Limits Borderline Outside Normal Limits No Classification Possible
rnflt_ts_1	Double	RNFLT TS 1	120	
rnflt_ts_2	Double	RNFLT TS 2 (%)	34	
rnflt_ns_1	Double	RNFLT NS 1	79	
rnflt_ns_2	Double	RNFLT NS 2 (%)	23	
rnflt_t_1	Double	RNFLT T 1	62	
rnflt_t_2	Double	RNFLT T 2 (%)	35	
rnflt_g_1	Double	RNFLT G 1	86	
rnflt_g_2	Double	RNFLT G 2 (%)	47	
rnflt_n_1	Double	RNFLT N 1	65	
rnflt_n_2	Double	RNFLT N 2 (%)	33	
rnflt_ti_1	Double	RNFLT TI 1	148	
rnflt_ti_2	Double	RNFLT TI 2 (%)	75	

rnflt_ni_1	Double	RNFLT NI 1	112	
rnflt_ni_2	Double	RNFLT NI 2 (%)	92	
delivery_id	Int	Delivery Identifier	3	0 to 5

e_ot

This table contains the data from the Ocular Tension exams.

Field	Type	Description	Example	Possible Values
patient_id	Int	Patient's Identifier	2134087	
eye	String	Patient´s eye	Right	Left Right
ot	Int	Ocular tension	10	
delivery_id	Int	Delivery Identifier	3	0 to 5

Dataset Creation

The data used to create the dataset was sourced from the Postgres SQL Database, and a single record was generated for each eye of the patient. The resulting dataset was stored in MinIO. Both MinIO and Postgres SQL Database are the tools used internally by the Altice Labs team.

It is provided a brief description of the steps used to create the dataset, as well as tables indicating the fields, their source, and their description.

Patient Data

Read from the patients table in the database. Date of last exam is obtained by joining the last exam_date in the tables e_bio, e_bio_2, e_bio_3, e_bio_4, e_pec_single, e_pec_summary, e_rnfl, e_rnfl_cr, e_rnfl_ou.

Fields	Source	Description
patient_id	p_diags	Patient identifier
birth_date	p_diags	Date of patient birth
gender	p_diags	Gender of the patient
exam_date	mapped	Last exam of the patient
age	mapped	Age of the patient at the last exam

Optical Tension Data

Read from the e_ot table in the database.

Fields	Source	Description
patient_id	e_ot	Patient identifier
eye	e_ot	Eye (left or right)
ot	e_ot	Optical Tension measurement

Pathologies Diagnostics Data

Read from the p_diags and pathologies tables in the database. Joined with deliveries table to consider that when a patient-eye pair have diagnostics on several deliveries, only the diagnostics of the last is kept.

From the diagnosis provided by the CCC it was created 3 distinct labels:

- label - label containing the diagnosis, with the possible values of: “NORMAL”, “SUSPEITO DE GLAUCOMA”, “GLAUCOMA”, “OUTRAS PATOLOGIAS”
- eye_sick_label - label indicating whether the eye has any diagnosed disease, regardless of the disease, with the possible values of “0” and “1” (0 – Eye Not Sick and 1 – Eye Sick)
- glaucoma_flag - label indicating whether the eye has diagnosed Glaucoma, with the possible values of “0” and “1” (0 – Eye Without Glaucoma and 1 – Eye With Glaucoma)

A patient-eye pair can have several pathologies associated.

Fields	Source	Description
patient_id	p_diags.patient_id	Patient identifier
eye	p_diags.location	Eye (left or right)
label	pathologies.label	Name associated with the pathology
other_eye_glau_flag	mapped	Flag (1 or 0) indicating that the other eye has glaucoma
glaucoma_flag	mapped	Flag (1 or 0) indicating that the eye has glaucoma
eye_sick_label	mapped	Flag (1 or 0) indicating that the eye is affected by a sickness

Biometries Data

Read from the e_bio, e_bio_2, e_bio_3 and e_bio_4 tables in the database. The resulting data is a union of the data in the tables. From Biometry exam, we pick the following features:

- 'al', 'cct', 'ad', 'acd', 'lt', 'k1_value', 'k1_angle', 'k2_value', 'k2_angle', 'ast_value', 'ast_angle', 'wtw', 'ic_x', 'ic_y', 'pc_x', 'pc_y'

There is one Biometry exam per patient.

Field	e_bio	e_bio_2	e_bio_3	e_bio_4	Description
patient_id	patient_id	patient_id	patient_id	patient_id	Patient identifier
eye	eye	eye	eye	eye	Eye (left or right)

exam_type	exam_type	exam_type	exam_type	exam_type	Type of exam
exam_date	exam_date	exam_date	exam_date	exam_date	Date of the exam
exam_time	exam_time	exam_time		exam_time	Hour of the exam
exam_duration	exam_duration				Duration of the exam in minutes
al	al	al	al		Axial length (mm)
cct	cct	cct	cct	cct	Cornea thickness (µm)
ad	ad	ad		ad	Aqueous depth (mm)
acd	acd	acd			Anterior chamber depth (mm)
lt	lt	lt	lt	lt	Lens thickness (mm)
rt	rt				Retina thickness (µm)
k1_value	k1_1	r1_2	simk_flat_1		Flat meridian value (D)
k1_angle	k1_2	r1_3	simk_flat_2		Flat meridian angle (°)
k2_value	k2_1	r2_2	simk_steep_1		Steep meridian value (D)

k2_angle	k2_2	r2_3	simk_steep_2		Steep meridian angle (°)
ast_value	ast_1	ast_1	ast_steep_1		Astigmatism value (D)
ast_angle	ast_2	ast_2	ast_steep_2		Astigmatism angle (°)
n	n	n			Keratometric index
wtw	wtw	wtw	wtw	wtw	White to White (mm)
ic_x	ic_1				Iris barycenter x-axis (mm)
ic_y	ic_2				Iris barycenter y-axis (mm)
pd	pd		pd	pd	Pupil diameter (mm)
pc_x	pc_x		pc_x	pc_x	Pupil barycenter x-axis (mm)
pc_y	pc_y		pc_y	pc_y	Pupil barycenter y-axis (mm)
r1_1		r1_1			No Description Available.
r2_1		r2_1			No Description

					Available.
r_1		r_1			No Description Available.
r_2		r_2			No Description Available.
simk_avg			simk_avg		No Description Available.
ast_total_value			ast_total_1		No Description Available.
ast_total_angle			ast_total_2		No Description Available.
ast_post_value			ast_post_1		No Description Available.
ast_post_angle			ast_post_2		No Description Available.
ast_delta_valu			ast_delta_1		No Description Available.
ast_delta_angle			ast_delta_2		No Description Available.
z			z		No Description Available.

rms_hoa			rms_hoa		No Description Available.
aqd			aqd		No Description Available.
cct_aqd			cct_aqd		No Description Available.
pc_value			pc_1		No Description Available.
pc_angle			pc_2		No Description Available.
acv				acv	No Description Available.
aca_dist				aca_dist	No Description Available.
sts_dist				sts_dist	No Description Available.
aca_500_angle9h				aca_500_1	No Description Available.
aca_500_angle3h				aca_500_2	No Description Available.
ssa_500_angle9h				ssa_500_1	No

					Description Available.
ssa_500_angle3h				ssa_500_2	No Description Available.
aod_500_angle9h				aod_500_1	No Description Available.
aod_500_angle3h				aod_500_2	No Description Available.
tisa_500_angle9h				tisa_500_1	No Description Available.
tisa_500_angle3h				tisa_500_2	No Description Available.
aca_750_angle9h				aca_750_1	No Description Available.
aca_750_angle3h				aca_750_2	No Description Available.
ssa_750_angle9h				ssa_750_1	No Description Available.
ssa_750_angle3h				ssa_750_2	No Description Available.
aod_750_angle9h				aod_750_1	No Description

					Available.
aod_750_angle3h				aod_750_2	No Description Available.
tisa_750_angle9h				tisa_750_1	No Description Available.
tisa_750_angle3h				tisa_750_2	No Description Available.
lv				lv	No Description Available.

PEC Data

Read from e_pec_single and e_pec_summary tables in the database. The resulting data is a union of the data in the tables. We have 1 to 3 PEC exams per patient. We consolidated the following features:

- 'vfi_last', 'vfi_min', 'vfi_max', 'vfi_mean', 'vfi_std'
- 'md_last', 'md_min', 'md_max', 'md_mean', 'md_std'
- 'psd_last', 'psd_min', 'psd_max', 'psd_mean', 'psd_std'
- 'last_fl', 'last_ght', 'pec_qtd'

Field	e_pec_single	e_pec_summary	Description
patient_id	patient_id	patient_id	Patient identifier
eye	eye	eye	Eye (left or right)
exam_type	exam_type	exam_type	Type of exam
exam_date	exam_date	exam_date	Date of the exam
exam_time	exam_time		Time of the exam
exam_duration	exam_duration		Duration of the exam
fix_monitor	fix_monitor		Type of fix monitor (Blind Spot, Gaze

			Track, etc)
fix_target	fix_target		Type of fix target (Central)
fl_1	fl_1	fl_1	Fixation Losses numerator
fl_2	fl_2	fl_2	Fixation Losses denominator
fp	fp	fp	False positives (%)
fn	fn	fn	False negatives (%)
stimulus	stimulus		Type of stimulus (III, White)
background	background		Background (ASB)
strategy	strategy		Strategy used (SITA-Fast)
pd	pupil_diameter		Pupil Diameter (mm)
fovea	fovea	fovea	Fovea
vfi	vfi	vfi	Visual Field Index (%)
md	md	md	Mean deviation (dB)
psd	psd	psd	Pattern standard deviation (dB)
ght	ght	ght	Glaucoma Hemifield Test result for the exam
exam_order	'1'	exam_order	Exam order in the several exams in the file

RNFL Data

Read from e_rnfl, e_rnfl_cr and e_rnfl_ou tables in the database. The resulting data is a union of the data in the tables. We have 1 to 3 RNFL exams per patient, although most patients only have 1 exam.

Original MRW:

- 'mrw_result', 'mrw_g_value', 'mrw_g_perc'
- 'mrw_ts_value', 'mrw_ns_value', 'mrw_t_value', 'mrw_n_value', 'mrw_ti_value', 'mrw_ni_value'
- 'mrw_ts_perc', 'mrw_ns_perc', 'mrw_t_perc', 'mrw_n_perc', 'mrw_ti_perc', 'mrw_ni_perc'

MRW summary:

- 'mrw_value_min', 'mrw_value_max', 'mrw_perc_min', 'mrw_perc_max'

Original RNFLT:

- 'rnflt_result', 'rnflt_g_value', 'rnflt_g_perc'
- 'rnflt_ts_value', 'rnflt_ns_value', 'rnflt_t_value', 'rnflt_n_value', 'rnflt_ti_value', 'rnflt_ni_value'
- 'rnflt_ts_perc', 'rnflt_ns_perc', 'rnflt_t_perc', 'rnflt_n_perc', 'rnflt_ti_perc', 'rnflt_ni_perc'

RNFLT summary:

- 'rnflt_value_min', 'rnflt_value_max', 'rnflt,_value_min', 'rnflt_value_max'
- 'rnflt_outsides', 'rnflt_borderlines'

Field	e_rnfl	e_rnfl_cr	e_rnfl_ou	Description
patient_id	patient_id	patient_id	patient_id	Patient identifier
eye	eye	eye	eye	Eye (left or right)
exam_type	exam_type	exam_type	exam_type	Type of exam
exam_date	exam_date	exam_date	exam_date	Date of the exam
reference	reference	reference	reference	Reference database used for percentiles and status classification
c_mrw	c_mrw			MRW Classification
mrw_ts_value	mrw_ts_1			Absolute value for MRW

				Temporal Superior sector
mrw_ns_value	mrw_ns_1			Absolute value for MRW Nasal Superior sector
mrw_t_value	mrw_t_1			Absolute value for MRW Temporal sector
mrw_g_value	mrw_g_1			Absolute value for MRW Global
mrw_n_value	mrw_n_1			Absolute value for MRW Nasal sector
mrw_ti_value	mrw_ti_1			Absolute value for MRW Temporal Inferior sector
mrw_ni_value	mrw_ni_1			Absolute value for MRW Nasal Inferior sector
mrw_ts_perc	mrw_ts_2			Percentile ratio of the MRW Temporal Superior sector
mrw_ns_perc	mrw_ns_2			Percentile ratio of the MRW Nasal Superior sector
mrw_t_perc	mrw_t_2			Percentile ratio of the MRW Temporal sector

mrw_g_perc	mrw_g_2			Percentile ratio of the MRW Global
mrw_n_perc	mrw_n_2			Percentile ratio of the MRW Nasal sector
mrw_ti_perc	mrw_ti_2			Percentile ratio of the MRW Temporal Inferior sector
mrw_ni_perc	mrw_ni_2			Percentile ratio of the MRW Nasal Inferior sector
c_rnflt	c_rnflt	c_rnflt	c_rnflt	RNFLT Classification
rnflt_ts_value	rnflt_ts_1	rnflt_ts_1	rnflt_ts_1	Absolute value for RNFLT Temporal Superior sector
rnflt_ns_value	rnflt_ns_1	rnflt_ns_1	rnflt_ns_1	Absolute value for RNFLT Nasal Superior sector
rnflt_t_value	rnflt_t_1	rnflt_t_1	rnflt_t_1	Absolute value for RNFLT Temporal sector
rnflt_g_value	rnflt_g_1	rnflt_g_1	rnflt_g_1	Absolute value for RNFLT Global
rnflt_n_value	rnflt_n_1	rnflt_n_1	rnflt_n_1	Absolute value for RNFLT Nasal

				sector
rnflt_ti_value	rnflt_ti_1	rnflt_ti_1	rnflt_ti_1	Absolute value for RNFLT Temporal Inferior sector
rnflt_ni_value	rnflt_ni_1	rnflt_ni_1	rnflt_ni_1	Absolute value for RNFLT Nasal Inferior sector
rnflt_ts_perc	rnflt_ts_2			Percentile ratio of the RNFLT Temporal Superior sector
rnflt_ns_perc	rnflt_ns_2			Percentile ratio of the RNFLT Nasal Superior sector
rnflt_t_perc	rnflt_t_2			Percentile ratio of the RNFLT Temporal sector
rnflt_g_perc	rnflt_g_2			Percentile ratio of the RNFLT Global
rnflt_n_perc	rnflt_n_2			Percentile ratio of the RNFLT Nasal sector
rnflt_ti_perc	rnflt_ti_2			Percentile ratio of the RNFLT Temporal Inferior sector
rnflt_ni_perc	rnflt_ni_2			Percentile ratio of the RNFLT

				Nasal Inferior sector
rnflt_ts_p50		rnflt_ts_2	rnflt_ts_2	Median of the RNFLT Temporal Superior sector
rnflt_ns_p50		rnflt_ns_2	rnflt_ns_2	Median of the RNFLT Nasal Superior sector
rnflt_t_p50		rnflt_t_2	rnflt_t_2	Median of the RNFLT Temporal sector
rnflt_g_p50		rnflt_g_2	rnflt_g_2	Median of the RNFLT Global
rnflt_n_p50		rnflt_n_2	rnflt_n_2	Median of the RNFLT Nasal sector
rnflt_ti_p50		rnflt_ti_2	rnflt_ti_2	Median of the RNFLT Temporal Inferior sector
rnflt_ni_p50		rnflt_ni_2	rnflt_ni_2	Median of the RNFLT Nasal Inferior sector
rnflt_4s_value			rnflt_s	Absolute value for RNFLT Superior sector
rnflt_4n_value			rnflt_n	Absolute value for RNFLT Nasal sector
rnflt_4i_value			rnflt_i	Absolute value

				for RNFLT Inferior sector
rnflt_4t_value			rnflt_t	Absolute value for RNFLT Temporal sector

Dataset Creation in JupyterLab

All the steps for creating the dataset in the JupiterLab, such as Data Ingestion and Data Transformations are presented in the images below. The prints shown below correspond to the Notebook 1.0.

Create File Name

```
import datetime

prefix = "DS_"
study = 'alldataset_'
timestamp = ""+'{:%Y%m%d%H%M%S}'.format(datetime.datetime.now())
extension = '.parquet'
file_name = author+prefix+study+timestamp+extension
file_name
```

```
'mf_DS_alldataset_20220322122342.parquet'
```

```
bucket_name = "glaucoma"
bucket_object = "data/processed/datasets/"+file_name
```

Imports

```
#System Libs
import os
import datetime
from dotenv import load_dotenv, find_dotenv

# S3/SQL Libs
import psycopg2
from minio import Minio
from minio.error import MinioException
import pandas.io.sql as psql

#DS Libs
import pandas as pd
import numpy as np
import math
import scipy.stats as st

#Visualization Libs
from tqdm.notebook import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

Setup I/O Access

```
# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()
# Load up the entries as environment variables
load_dotenv(dotenv_path)
```

```
True
```

```
s3_minio_kwargs = {
    "key": os.environ.get("MINIO_ACCESS_KEY"),
    "secret": os.environ.get("MINIO_SECRET_KEY"),
    "client_kwargs": {
        'endpoint_url': 'http://' + os.environ.get("MINIO_ENDPOINT")
    }
}
s3_minio_kwargs
```

```
# DB
param_db = {
    "host"     : os.environ.get("POSTGRES_ENDPOINT"),
    "port"     : os.environ.get("POSTGRES_PORT"),
    "database" : os.environ.get("POSTGRES_DATABASE"),
    "user"     : os.environ.get("POSTGRES_USERNAME"),
    "password" : os.environ.get("POSTGRES_PASSWORD")
}
param_db
```

PostgreSQL Connection

```
param_db = {
    "host"     : [redacted]
    "port"     :
    "database" :
    "user"     :
    "password" :
}

def connect(param_db):
    """Connect to the PostgresSQL database server"""
    conn = None
    try:
        # connect to the PostgresSQL server
        print('Connecting to the POstgresSQL database...')
        conn = psycopg2.connect(**param_db)
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
        sys.exit(1)
    print("Connection successful")
    return conn

#Connect to the database
conn =connect(param_db)
```

```
Connecting to the POstgresSQL database...
Connection successful
```

Query Functions

```
def postgresql_to_dataframe(conn, select_query, column_names = None):
    """
    Tranform a SELECT query into a pandas dataframe
    """
    with conn.cursor() as cursor:
        cursor.execute(select_query)
        if column_names is None:
            column_names = [desc[0] for desc in cursor.description]
        tuples = cursor.fetchall()
        return pd.DataFrame(tuples, columns=column_names)
```

Create Patient Summary Dataframe

```python
#Creates a summary by patient eye

# GLOBAL: patient_id | age | gender | label (GLAUCOMA/NORMAL/OUTRO) | eye (RIGHT/LEFT)
# PEC: vfi_last, vfi_min, vfi_max, vfi_mean, vfi_std, md_last, md_min, md_max, md_mean, md_std, psd_last, psd_min, p
# BIO: al, cct, ad, acd, lt, k1_value, k1_degree, k2_value, k2_degree, ast_value, ast_degree, wtw, ic_x, ic_y, pd, p
# MRW: mrw_result, mrw_ts_value, mrw_ts_perc, mrw_ns_value, mrw_ns_perc, mrw_t_value, mrw_t_perc, mrw_g_value, mrw_g
# RNFLT: rfnlt_result, rnflt_ts_value, rnflt_ts_perc, rnflt_ns_value, rnflt_ns_perc, rnflt_t_value, rnflt_t_perc, rn

def createPatientSummary(_patient_id):

    output = []
    _df_patient = df_patientsByPatient.get_group(_patient_id)
    _output_by_patient = [_patient_id, _df_patient['age'].iloc[0], _df_patient['gender'].iloc[0]]

    # Scroll through the eyes for which we have diagnostics
    # Set because for the same eye we can have more than one diagnosis

    for _eye in set(df_diagnostic[df_diagnostic['patient_id']==_patient_id]['eye']): #['RIGHT','LEFT']
        _df_pec = []
        _df_bio = []
        _df_rnfl = []
        _df_pd = []
        _df_ot = []

        # filter the exams by patient and eye
        try:
            _df_pec   = df_pec_examsByPatientEye.get_group((_patient_id, _eye)).sort_values('exam_date', ascending=Fal
            _pec_flag = 1
        except:
            _pec_flag = 0
        try:
            _df_bio   = df_biometry_examsByPatientEye.get_group((_patient_id, _eye)).sort_values('exam_date', ascendin
            _bio_flag = 1
        except:
            _bio_flag = 0
        try:
            _df_pd = df_pd_ByPatientEye.get_group((_patient_id, _eye)).sort_values('exam_date', ascending=False)
            _pd_flag = 1
        except:
            _pd_flag = 0
        try:
            _df_ot =  df_ot_ByPatientEye.get_group((_patient_id, _eye))['ot'].values[0]
            _ot_flag = 1
        except:
            _ot_flag = 0
        try:
            _df_rnfl = df_rnfl_examsByPatientEye.get_group((_patient_id, _eye)).sort_values('exam_date', ascending=Fa
            _rnfl_flag = 1
        except:
            _rnfl_flag = 0

        # Validate that I have at least 1 exam for that patient/eye
        if ((_pec_flag+_bio_flag+_rnfl_flag)>0):

            if(_pec_flag ==1):
                # PEC
                # If there is more than one PEC for the same day, the most credible remains.
                _df_pec     = _df_pec[_df_pec.groupby('exam_date')['fl'].rank(method='first')==1]


                # Creating statistics of the indicators
                _pec_vfi = statistic(_df_pec, 'vfi', 'exam_date') #vfi_last, vfi_min, vfi_max, vfi_mean, vfi_std
                _pec_md  = statistic(_df_pec, 'md', 'exam_date')  #md_last, md_min, md_max, md_mean, md_std
                _pec_psd = statistic(_df_pec, 'psd', 'exam_date') #psd_last, psd_min, psd_max, psd_mean, psd_std
                _last_fl    = _df_pec[~(_df_pec['fl'].isna())].sort_values('exam_date',ascending=False).iloc[0]['fl'
                _last_ght   = _df_pec[~(_df_pec['ght'].isna())].sort_values('exam_date',ascending=False).iloc[0]['gh
                _pec_qtd    = _df_pec.shape[0]

                _pec_summary = _pec_vfi + _pec_md + _pec_psd + [_last_fl, _last_ght, _pec_qtd]
            else:
                _pec_summary = [None,None,None,None,None,None,None,None,None,None,None,None,None,None,None,None,None

            if(_bio_flag == 1):
                # Biometrics
                # For representativeness we restrict to the last exam
                # ignore the fields patient_id, eye, exam_type, exam_date, exam_time, exam_duration
```

```
                # ignore the rt and n fields, since they are always constant (rt=200 and n=1.3375)
                _bio_summary = list(_df_bio.head(1)[['al', 'cct', 'ad', 'acd', 'lt', 'k1_value',\
                                                     'k1_angle', 'k2_value', 'k2_angle', 'ast_value', 'ast_angle', \
                                                     'wtw', 'ic_x', 'ic_y', 'pc_x', 'pc_y']].iloc[0].array)
            else:
                _bio_summary = list([None,None,None,None,None,None,None,None,None,None,None,None,None,None,None,None
----------
            if(_pd_flag ==1):
                # [Pupil Diameter]
---------
                # Selection of the last known DP
                _last_pd    = _df_pd[~(_df_pd['pd'].isna())].sort_values(['exam_date','exam_type'],ascending=[False,
                _pd_summary = list([_last_pd])
            else:
                _pd_summary = list([None])
----------
            if(_ot_flag ==1):
                # Ocular Tension
                _ot = _df_ot
                _ot_summary = list([_ot])
            else:
                _ot_summary = list([None])
-------------
            if(_rnfl_flag ==1):
                # RNFL
                # In the case of the RNFL we have almost for every patient a single exam per eye. Only for 10 eyes (
                # Due to representativeness we restrict to the last exam
```

```
                # ignore the fields patient_id, eye, exam_type, exam_date
----------
                # RNFL: mrw_result, mrw_ts_value, mrw_ts_perc, mrw_ns_value, mrw_ns_perc, mrw_t_value, mrw_t_perc, r
                #       rfnlt_result, rnflt_ts_value, rnflt_ts_perc, rnflt_ns_value, rnflt_ns_perc, rnflt_t_value, r
----------
                _rnfl_summary =[]
----------
                _c_mrw        = _df_rnfl.head(1)['c_mrw'].values[0]
                _mrw_g_value  = _df_rnfl.head(1)['mrw_g_value'].values[0]
                _mrw_g_perc   = _df_rnfl.head(1)['mrw_g_perc'].values[0]
                _mrw_ts_value = _df_rnfl.head(1)['mrw_ts_value'].values[0]
                _mrw_ns_value = _df_rnfl.head(1)['mrw_ns_value'].values[0]
                _mrw_t_value  = _df_rnfl.head(1)['mrw_t_value'].values[0]
                _mrw_n_value  = _df_rnfl.head(1)['mrw_n_value'].values[0]
                _mrw_ti_value = _df_rnfl.head(1)['mrw_ti_value'].values[0]
                _mrw_ni_value = _df_rnfl.head(1)['mrw_ni_value'].values[0]
                _mrw_ts_perc  = _df_rnfl.head(1)['mrw_ts_perc'].values[0]
                _mrw_ns_perc  = _df_rnfl.head(1)['mrw_ns_perc'].values[0]
                _mrw_t_perc   = _df_rnfl.head(1)['mrw_t_perc'].values[0]
                _mrw_n_perc   = _df_rnfl.head(1)['mrw_n_perc'].values[0]
                _mrw_ti_perc  = _df_rnfl.head(1)['mrw_ti_perc'].values[0]
                _mrw_ni_perc  = _df_rnfl.head(1)['mrw_ni_perc'].values[0]
                _c_rnflt      = _df_rnfl.head(1)['c_rnflt'].values[0]
                _rnflt_g_value = _df_rnfl.head(1)['rnflt_g_value'].values[0]
                _rnflt_g_perc  = _df_rnfl.head(1)['rnflt_g_perc'].values[0]
                _rnflt_ts_value= _df_rnfl.head(1)['rnflt_ts_value'].values[0]
                _rnflt_ns_value= _df_rnfl.head(1)['rnflt_ns_value'].values[0]
```

```
_rnflt_t_value = _df_rnfl.head(1)['rnflt_t_value'].values[0]
_rnflt_n_value = _df_rnfl.head(1)['rnflt_n_value'].values[0]
_rnflt_ti_value= _df_rnfl.head(1)['rnflt_ti_value'].values[0]
_rnflt_ni_value= _df_rnfl.head(1)['rnflt_ni_value'].values[0]
_rnflt_ts_perc = _df_rnfl.head(1)['rnflt_ts_perc'].values[0]
_rnflt_ns_perc = _df_rnfl.head(1)['rnflt_ns_perc'].values[0]
_rnflt_t_perc  = _df_rnfl.head(1)['rnflt_t_perc'].values[0]
_rnflt_n_perc  = _df_rnfl.head(1)['rnflt_n_perc'].values[0]
_rnflt_ti_perc = _df_rnfl.head(1)['rnflt_ti_perc'].values[0]
_rnflt_ni_perc = _df_rnfl.head(1)['rnflt_ni_perc'].values[0]

_rnflt_g_p50 = _df_rnfl.head(1)['rnflt_g_p50'].values[0]
_rnflt_ts_p50 = _df_rnfl.head(1)['rnflt_ts_p50'].values[0]
_rnflt_ns_p50 = _df_rnfl.head(1)['rnflt_ns_p50'].values[0]
_rnflt_t_p50  = _df_rnfl.head(1)['rnflt_t_p50'].values[0]
_rnflt_n_p50  = _df_rnfl.head(1)['rnflt_n_p50'].values[0]
_rnflt_ti_p50 = _df_rnfl.head(1)['rnflt_ti_p50'].values[0]
_rnflt_ni_p50 = _df_rnfl.head(1)['rnflt_ni_p50'].values[0]


------------------
 # Calculation of rnflt_<sector>_perc when we don't have this, but have rnflt_<sector>_p50
if( math.isnan(_rnflt_g_perc) and not(math.isnan(_rnflt_g_p50))):
    _rnflt_g_perc  = calc_perc('g',_rnflt_g_value ,_rnflt_g_p50 )
if( math.isnan(_rnflt_ts_perc) and not(math.isnan(_rnflt_ts_p50))):
    _rnflt_ts_perc  = calc_perc('ts',_rnflt_ts_value ,_rnflt_ts_p50 )
if( math.isnan(_rnflt_ns_perc) and not(math.isnan(_rnflt_ns_p50))):
    _rnflt_ns_perc  = calc_perc('ns',_rnflt_ns_value ,_rnflt_ns_p50 )
```

```
if( math.isnan(_rnflt_t_perc) and not(math.isnan(_rnflt_t_p50))):
    _rnflt_t_perc  = calc_perc('t',_rnflt_t_value ,_rnflt_t_p50 )
if( math.isnan(_rnflt_n_perc) and not(math.isnan(_rnflt_n_p50))):
    _rnflt_n_perc  = calc_perc('n',_rnflt_n_value ,_rnflt_n_p50 )
if( math.isnan(_rnflt_ti_perc) and not(math.isnan(_rnflt_ti_p50))):
    _rnflt_ti_perc  = calc_perc('ti',_rnflt_ti_value ,_rnflt_ti_p50 )
if( math.isnan(_rnflt_ni_perc) and not(math.isnan(_rnflt_ni_p50))):
    _rnflt_ni_perc  = calc_perc('ni',_rnflt_ni_value ,_rnflt_ni_p50 )

_rnfl_summary = [_c_mrw, _mrw_g_value, _mrw_g_perc\
            ,_mrw_ts_value, _mrw_ns_value, _mrw_t_value, _mrw_n_value, _mrw_ti_value, _mrw_ni_value\
            ,_mrw_ts_perc, _mrw_ns_perc, _mrw_t_perc, _mrw_n_perc, _mrw_ti_perc, _mrw_ni_perc\
            ,_c_rnflt, _rnflt_g_value, _rnflt_g_perc\
            ,_rnflt_ts_value, _rnflt_ns_value, _rnflt_t_value, _rnflt_n_value, _rnflt_ti_value, _rnf
            ,_rnflt_ts_perc, _rnflt_ns_perc, _rnflt_t_perc, _rnflt_n_perc, _rnflt_ti_perc, _rnflt_ni
------------------
#In the next summaries of MRW and RNFL the MRW_G and RFLT_G value are not included since it is alrea
------------------
#mrw_value_min, mrw_value_max
#Let's compare the 3rd and 4th results of the statistic function. The 3rd is the mean which can be s
#the 4th is the standard deviation, as we have few samples, we'll ignore

_df_rnfl_mrw_value = _df_rnfl[['mrw_ts_value', 'mrw_ns_value', 'mrw_t_value', 'mrw_n_value', 'mrw_ti
_df_rnfl_mrw_value.rename(columns={_df_rnfl_mrw_value.columns[0]: "mrw_value" }, inplace = True)
_rnfl_mrw_value = statistic(_df_rnfl_mrw_value, 'mrw_value')[0:2]
------------------
#mrw_perc_min, mrw_perc_max
```

```
                _df_rnfl_mrw_perc = _df_rnfl[['mrw_ts_perc', 'mrw_ns_perc', 'mrw_t_perc', 'mrw_n_perc', 'mrw_ti_perc
                _df_rnfl_mrw_perc.rename(columns={_df_rnfl_mrw_perc.columns[0]: "mrw_perc" }, inplace = True)
                _rnfl_mrw_perc = statistic(_df_rnfl_mrw_perc, 'mrw_perc')[0:2]
------------------
                #rnflt_value_min, rnflt_value_max
--------------
                _df_rnfl_rnflt_value = _df_rnfl[['rnflt_ts_value', 'rnflt_ns_value', 'rnflt_t_value', 'rnflt_n_value
                #_df_rnfl_rnflt_value = _df_rnfl[['rnflt_ts_value', 'rnflt_ns_value', 'rnflt_t_value', 'rnflt_g_valu
                _df_rnfl_rnflt_value.rename(columns={_df_rnfl_rnflt_value.columns[0]: "rnflt_value" }, inplace = Tr
                _rnfl_rnflt_value = statistic(_df_rnfl_rnflt_value, 'rnflt_value')[0:2]
------------------
                #rnflt_perc_min, rnflt_perc_max
--------------
                # In the case of RNFL PERC because for some this value was only calculated in this function so I hav
                # this because some exams don't have the perc for perc_50
                _df_rnfl_rnflt_perc = pd.DataFrame({'rnflt_perc': [_rnflt_ts_perc,_rnflt_ns_perc,_rnflt_t_perc,_rnfl
                        columns = ['rnflt_perc'],
                        index=['rnflt_ts_perc','rnflt_ns_perc','rnflt_t_perc','rnflt_n_perc','rnflt_ti_perc','rnflt
                _rnfl_rnflt_perc = statistic(_df_rnfl_rnflt_perc, 'rnflt_perc')[0:2]


------------------
                # Need to analyze the RNFLT indicators in more detail
                # Percentils
                    # > 5% - Within Normal Limits
                    # < 5% - Borderline
                    # < 1% - Outside Normal Limits
```

```
                #rnflt_outsides, rnflt_borderlines
                _df_rnfl_perc_aux = _df_rnfl[['rnflt_ts_perc', 'rnflt_ns_perc', 'rnflt_t_perc', 'rnflt_g_perc', 'rnf
                _rnfl_outsides = _df_rnfl_perc_aux[(_df_rnfl_perc_aux <=1)].count(1).values[0]
                _rnfl_borderlines = _df_rnfl_perc_aux[(_df_rnfl_perc_aux > 1) & (_df_rnfl_perc_aux <= 5)].count(1).v
-----------------------------------
                _rnfl_summary = _rnfl_summary + _rnfl_mrw_value + _rnfl_mrw_perc + _rnfl_rnflt_value + _rnfl_rnflt_p
            else:
                _rnfl_summary = list([None,None,None,None,None,None,None,None,None,None,None,None,None,None,Nor
-------------
            # The same eye may have more than one diagnosis. E.g. you may have glaucoma, and other pathologies
            _df_diagnostic = df_diagnosticByPatientEye.get_group((_patient_id,_eye))
            _other_eye_glau_flag = _df_diagnostic['other_eye_glau_flag'].iloc[0]
            _glaucoma_flag = _df_diagnostic['glaucoma_flag'].iloc[0]
            _eye_sick_label = _df_diagnostic['eye_sick_label'].iloc[0]
            for diag in _df_diagnostic['label']:
                _output_by_eye = _output_by_patient + [diag] + [_other_eye_glau_flag]+ [_glaucoma_flag]+ [_eye_sick_
                output.append(_output_by_eye)
---------------------------
    return output
```

Read Tables

Patients

```
# Get Age from Birth Date
# Creates the field age
# The most recent date of diagnosis/examination is considered. It would be closer to the age the doctor knew
# when he made the diagnosis to the target patient

query = '''
SELECT
    id as "patient_id",
    birth_date,
    gender,
    exam_date,
    CASE
        WHEN DATE_PART('month', exam_date::date)*31 + DATE_PART('day', exam_date::date) > DATE_PART('month', birth_d
            THEN DATE_PART('year', exam_date::date) - DATE_PART('year', birth_date::date)
        ELSE DATE_PART('year', exam_date::date) - DATE_PART('year', birth_date::date) - 1
    END as "age"
FROM patients
LEFT OUTER JOIN (
    SELECT patient_id, max(exam_date) as "exam_date", max(delivery_id) as "delivery_id"
    FROM (
        SELECT patient_id, exam_date, delivery_id FROM e_bio
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_bio_2
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_bio_3
    UNION ALL
```

```
        SELECT patient_id, exam_date, delivery_id FROM e_bio_4
    --UNION ALL
    --    SELECT patient_id, delivery_date, delivery_id as exam_date FROM e_ot
    --    INNER JOIN deliveries ON e_ot.delivery_id = deliveries.id
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_pec_single
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_pec_summary
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_rnfl
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_rnfl_cr
    UNION ALL
        SELECT patient_id, exam_date, delivery_id FROM e_rnfl_ou
    ) AS T1
    WHERE delivery_id <= {0:}
    GROUP BY patient_id
) AS T2 ON patients.id = T2.patient_id
WHERE delivery_id <= {0:};
'''.format(delivery_thrsh_global)
column_names = ['patient_id','birth_date','gender','exam_date','age']
df_patients = postgresql_to_dataframe(conn, query, column_names)
df_patients.head(3)
```

	patient_id	birth_date	gender	exam_date	age
0	160	1949-04-14	F	2020-08-31	71.0
1	485	1953-12-10	F	2020-11-13	66.0
2	659	1952-09-28	M	2020-08-28	67.0

Patients Preprocessing

```
# Creates the field age_10 (age truncated down to the 10's)
# Uppercases all the string values

df_patients = df_patients.applymap(lambda s:s.upper() if type(s) == str else s)
df_patients['age_10'] = df_patients['age']//10*10
df_patients.head(3)
```

	patient_id	birth_date	gender	exam_date	age	age_10
0	160	1949-04-14	F	2020-08-31	71.0	70.0
1	485	1953-12-10	F	2020-11-13	66.0	60.0
2	659	1952-09-28	M	2020-08-28	67.0	60.0

Ocular Tension

```
query = '''SELECT patient_id, eye, ot FROM public.e_ot where ot > 0 and delivery_id <= {}'''.format(delivery_thrsh_g

column_names = ["patient_id", "eye", "ot"]
df_ot = postgresql_to_dataframe(conn, query, column_names)
df_ot = df_ot.applymap(lambda s:s.upper() if type(s) == str else s)
df_ot.head(3)
```

	patient_id	eye	ot
0	2210366	RIGHT	15
1	2210366	LEFT	18
2	105418	RIGHT	10

Pathologies

```
query = ''' SELECT * FROM pathologies'''

column_names = ["id","label"]
df_pathologies = postgresql_to_dataframe(conn, query, column_names)
df_pathologies.head(10)
```

	id	label
0	0	Normal
1	1	Outras Patologias
2	2	Glaucoma
3	3	Suspeito de Glaucoma
4	4	Edema
5	5	Edema NO
6	6	Edema Macular
7	7	Descolamento
8	8	Descolamento Seroso
9	10	Buraco Lamelar

Diagnostics

```
# Loads the patient pathologies diagnosis with the flags:
    #other_eye_glau_flag -> The other eye of the patient has glaucoma
    #glaucoma_flag -> This eye has glaucoma
    #eye_sick_label -> This eye has a diagnosed pathology (it is NOT Normal AND NOT Suspeito de Glaucoma)

query = '''
WITH
  last_delivery AS (
    SELECT patient_id, location, max(delivery_id) as "delivery_id"
    FROM p_diags where delivery_id <= {}
    GROUP BY patient_id, location
  ),
  left_glaucoma_flag AS (
    SELECT p_diags.patient_id, 1 as "left_glaucoma_flag"
    FROM p_diags
    INNER JOIN last_delivery ON p_diags.patient_id = last_delivery.patient_id and p_diags.location = last_delivery.l
    INNER JOIN pathologies ON p_diags.diagnostic_id = pathologies.id
    WHERE label = 'Glaucoma' and p_diags.location = 'left'
    GROUP BY p_diags.patient_id
  ),
  right_glaucoma_flag AS (
    SELECT p_diags.patient_id, 1 as "right_glaucoma_flag"
    FROM p_diags
    INNER JOIN last_delivery ON p_diags.patient_id = last_delivery.patient_id and p_diags.location = last_delivery.l
    INNER JOIN pathologies ON p_diags.diagnostic_id = pathologies.id
    WHERE label = 'Glaucoma' and p_diags.location = 'right'
```

```
    GROUP BY p_diags.patient_id
  ),
  left_eye_sick_label AS (
    SELECT p_diags.patient_id, 1 as "left_eye_sick_label"
    FROM p_diags
    INNER JOIN last_delivery ON p_diags.patient_id = last_delivery.patient_id and p_diags.location = last_delivery.l
    INNER JOIN pathologies ON p_diags.diagnostic_id = pathologies.id
    WHERE label not in('Normal', 'Suspeito de Glaucoma')  and p_diags.location = 'left'
    GROUP BY p_diags.patient_id
  ),
  right_eye_sick_label AS (
    SELECT p_diags.patient_id, 1 as "right_eye_sick_label"
    FROM p_diags
    INNER JOIN last_delivery ON p_diags.patient_id = last_delivery.patient_id and p_diags.location = last_delivery.l
    INNER JOIN pathologies ON p_diags.diagnostic_id = pathologies.id
    WHERE label not in('Normal', 'Suspeito de Glaucoma')  and p_diags.location = 'right'
    GROUP BY p_diags.patient_id
  )
SELECT
    p_diags.patient_id AS "patient_id",
    p_diags.location AS "eye",
    pathologies.label AS "label",
    CASE p_diags.location
        WHEN 'right' THEN COALESCE(left_glaucoma_flag, 0)
        WHEN 'left'  THEN COALESCE(right_glaucoma_flag, 0)
        ELSE NULL
    END AS "other_eye_glau_flag",
```

```
    CASE p_diags.location
        WHEN 'right' THEN COALESCE(right_glaucoma_flag, 0)
        WHEN 'left'  THEN COALESCE(left_glaucoma_flag, 0)
        ELSE NULL
    END AS "glaucoma_flag",
    CASE p_diags.location
        WHEN 'right' THEN COALESCE(right_eye_sick_label, 0)
        WHEN 'left'  THEN COALESCE(left_eye_sick_label, 0)
        ELSE NULL
    END AS "eye_sick_label"
FROM p_diags
INNER JOIN last_delivery ON p_diags.patient_id = last_delivery.patient_id and p_diags.location = last_delivery.locat
INNER JOIN pathologies ON p_diags.diagnostic_id = pathologies.id
LEFT OUTER JOIN left_glaucoma_flag    ON p_diags.patient_id = left_glaucoma_flag.patient_id
LEFT OUTER JOIN right_glaucoma_flag   ON p_diags.patient_id = right_glaucoma_flag.patient_id
LEFT OUTER JOIN left_eye_sick_label   ON p_diags.patient_id = left_eye_sick_label.patient_id
LEFT OUTER JOIN right_eye_sick_label  ON p_diags.patient_id = right_eye_sick_label.patient_id
ORDER BY p_diags.patient_id, p_diags.location, pathologies.label;
'''.format(delivery_thrsh_global)
column_names = ["patient_id", "eye",'label','other_eye_glau_flag','glaucoma_flag','eye_sick_label']
df_diagnostic = postgresql_to_dataframe(conn, query, column_names)
df_diagnostic.head(3)
```

	patient_id	eye	label	other_eye_glau_flag	glaucoma_flag	eye_sick_label
0	18	right	Normal	0	0	0
1	100	right	Normal	0	0	0
2	160	left	Membrana epi	0	0	1

Diagnostics Preprocessing

```
#Uppercases all string values
#Redefines the label by turning the values that are not Normal, Glaucoma and Suspeito de Glaucoma into OUTRAS PATOLO
#Drops duplicate

df_diagnostic = df_diagnostic.applymap(lambda s:s.upper() if type(s) == str else s)

df_diagnostic.loc[~(df_diagnostic['label'].isin(['NORMAL','GLAUCOMA','SUSPEITO DE GLAUCOMA'])),'label']='OUTRAS PATO
df_diagnostic = df_diagnostic.drop_duplicates()
df_diagnostic.head(3)
```

	patient_id	eye	label	other_eye_glau_flag	glaucoma_flag	eye_sick_label
0	18	RIGHT	NORMAL	0	0	0
1	100	RIGHT	NORMAL	0	0	0
2	160	LEFT	OUTRAS PATOLOGIAS	0	0	1

Biometries

```
df_bio = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"exam_time" AS "exam_time",
"exam_duration" AS "exam_duration",
"al" AS "al",
"cct" AS "cct",
"ad" AS "ad",
"acd" AS "acd",
"lt" AS "lt",
"rt" AS "rt",
"k1_1" AS "k1_value",
"k1_2" AS "k1_angle",
"k2_1" AS "k2_value",
"k2_2" AS "k2_angle",
"ast_1" AS "ast_value",
"ast_2" AS "ast_angle",
"n" AS "n",
"wtw" AS "wtw",
"ic_1" AS "ic_x",
"ic_2" AS "ic_y",
"pd" AS "pd",
"pc_x" AS "pc_x",
```

```
"pc_y" AS "pc_y"
FROM e_bio
WHERE delivery_id <= {}'''.format(delivery_thrsh_global),None)

df_bio_2 = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"exam_time" AS "exam_time",
"al" AS "al",
"cct" AS "cct",
"ad" AS "ad",
"acd" AS "acd",
"lt" AS "lt",
"r1_2" AS "k1_value",
"r1_3" AS "k1_angle",
"r2_2" AS "k2_value",
"r2_3" AS "k2_angle",
"ast_1" AS "ast_value",
"ast_2" AS "ast_angle",
"n" AS "n",
"wtw" AS "wtw",
"r1_1" AS "r1_1",
"r2_1" AS "r2_1",
"r_1" AS "r_1",
```

```
"r_2" AS "r_2"
FROM e_bio_2
WHERE delivery_id <= {}'''.format(delivery_thrsh_global),None)

df_bio_3 = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"al" AS "al",
"cct" AS "cct",
"lt" AS "lt",
"simk_flat_1" AS "k1_value",
"simk_flat_2" AS "k1_angle",
"simk_steep_1" AS "k2_value",
"simk_steep_2" AS "k2_angle",
"ast_steep_1" AS "ast_value",
"ast_steep_2" AS "ast_angle",
"wtw" AS "wtw",
"pd" AS "pd",
"pc_x" AS "pc_x",
"pc_y" AS "pc_y",
"simk_avg" AS "simk_avg",
"ast_total_1" AS "ast_total_value",
"ast_total_2" AS "ast_total_angle",
"ast_post_1" AS "ast_post_value",
```

```
"ast_post_2" AS "ast_post_angle",
"ast_delta_1" AS "ast_delta_value",
"ast_delta_2" AS "ast_delta_angle",
"z" AS "z",
"rms_hoa" AS "rms_hoa",
"aqd" AS "aqd",
"cct_aqd" AS "cct_aqd",
"pc_1" AS "pc_value",
"pc_2" AS "pc_angle"
FROM e_bio_3
WHERE delivery_id <= {}'''.format(delivery_thrsh_global),None)

df_bio_4 = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"exam_time" AS "exam_time",
"cct" AS "cct",
"ad" AS "ad",
"lt" AS "lt",
"wtw" AS "wtw",
"pd" AS "pd",
"pc_x" AS "pc_x",
"pc_y" AS "pc_y",
"acv" AS "acv",
```

```
"aca_dist" AS "aca_dist",
"sts_dist" AS "sts_dist",
"aca_500_1" AS "aca_500_angle9h",
"aca_500_2" AS "aca_500_angle3h",
"ssa_500_1" AS "ssa_500_angle9h",
"ssa_500_2" AS "ssa_500_angle3h",
"aod_500_1" AS "aod_500_angle9h",
"aod_500_2" AS "aod_500_angle3h",
"tisa_500_1" AS "tisa_500_angle9h",
"tisa_500_2" AS "tisa_500_angle3h",
"aca_750_1" AS "aca_750_angle9h",
"aca_750_2" AS "aca_750_angle3h",
"ssa_750_1" AS "ssa_750_angle9h",
"ssa_750_2" AS "ssa_750_angle3h",
"aod_750_1" AS "aod_750_angle9h",
"aod_750_2" AS "aod_750_angle3h",
"tisa_750_1" AS "tisa_750_angle9h",
"tisa_750_2" AS "tisa_750_angle3h",
"lv" AS "lv"
FROM e_bio_4
WHERE delivery_id <= {}'''.format(delivery_thrsh_global),None)

df_biometry = pd.concat([df_bio, df_bio_2, df_bio_3, df_bio_4], axis=0)

for col in df_biometry:
    if pd.api.types.is_object_dtype(df_biometry[col].dtype):
```

```
        df_biometry[col] = df_biometry[col].apply(lambda x: None if type(x) == float and np.isnan(x) else x)
df_biometry.head(3)
```

	patient_id	eye	exam_type	exam_date	exam_time	exam_duration	al	cct	ad	acd	...	tisa_500_angle3h	aca_750_angle9h	
0	2184117	left	Biometry	2014-04-08	17:55	3	23.15	525.0	3.15	3.68	...	NaN	NaN	
1	2184117	right	Biometry	2014-04-08	17:55	3	23.28	530.0	2.61	3.14	...	NaN	NaN	
2	100	right	Biometry	2014-09-15	11:37	1	28.09	533.0	3.30	3.84	...	NaN	NaN	

3 rows × 62 columns

Biometries Preprocessing

```
#All string fileds in upper. Eg. in the exam_duration field it will make the uniformization of Min, min
#In exam_duration filed stay only the numeric part

df_biometry_exams = df_biometry.copy()

df_biometry_exams = df_biometry_exams.applymap(lambda s:s.upper() if type(s) == str else s)

df_biometry_exams['exam_duration'] = df_biometry_exams['exam_duration'].str.replace(' MIN','')
df_biometry_exams['exam_duration'] = pd.to_numeric(df_biometry_exams['exam_duration'])
df_biometry_exams.head(3)
```

	patient_id	eye	exam_type	exam_date	exam_time	exam_duration	al	cct	ad	acd	...	tisa_500_angle3h	aca_750_angle9h
0	2184117	LEFT	BIOMETRY	2014-04-08	17:55	3.0	23.15	525.0	3.15	3.68	...	NaN	NaN
1	2184117	RIGHT	BIOMETRY	2014-04-08	17:55	3.0	23.28	530.0	2.61	3.14	...	NaN	NaN
2	100	RIGHT	BIOMETRY	2014-09-15	11:37	1.0	28.09	533.0	3.30	3.84	...	NaN	NaN

3 rows × 62 columns

PECs

```
query = '''
    SELECT patient_id, eye, exam_type, exam_date, exam_time, exam_duration,
        fix_monitor, fix_target, fl_1, fl_2, fp, fn,
        stimulus, background, strategy, pupil_diameter,
        fovea, vfi, md, psd, ght,
        1 as exam_order
    from public.e_pec_single where delivery_id <= {0:}
    union all
    SELECT patient_id, eye, exam_type, exam_date, null as exam_time, null as exam_duration,
        null as fix_monitor, null as fix_target, fl_1, fl_2, fp, fn,
        null as stimulus, null as background, null as strategy,
        null as pupil_diameter,
        fovea, vfi, md, psd, ght,
        exam_order
    from public.e_pec_summary where delivery_id <= {0:}
    '''.format(delivery_thrsh_global)

column_names = ["patient_id", "eye", "exam_type", "exam_date", "exam_time", "exam_duration", \
                "fix_monitor", "fix_target", "fl_1", "fl_2", "fp", "fn", \
                "stimulus", "background", "strategy", "pd", \
                "fovea", "vfi", "md", "psd", "ght", "exam_order"]

df_pec = postgresql_to_dataframe(conn, query, column_names)
df_pec.head(3)
```

	patient_id	eye	exam_type	exam_date	exam_time	exam_duration	fix_monitor	fix_target	fl_1	fl_2	...	stimulus	background	stra
0	100	right	PEC_single	2021-07-27	11:12	02:31	Gaze Monitor	Central	0.0	0.0	...	III, White	31.5	
1	24698	left	PEC_single	2021-07-28	09:12	03:02	OFF	Central	0.0	0.0	...	III, White	31.5	
2	2291532	left	PEC_single	2021-07-26	12:30	02:21	Gaze Track	Central	0.0	0.0	...	III, White	31.5	

3 rows × 22 columns

PECs Preprocessing

```
#Discards columns exam_time, exam_duration, fix_monitor, fix_target, stimulus, background, strategy and pd
#Uppercases all string values
#Computes fl (Fixation Losses) ratio by dividing fl_1 by fl_2. Null results are converted to zero
#Replaces null values in fp and fn by zero

# join the 2 tables, keeping the columns in common
df_pec_exams = df_pec.copy()
df_pec_exams = df_pec_exams.drop(columns=['exam_time', 'exam_duration','fix_monitor','fix_target','stimulus','backgr
df_pec_exams = df_pec_exams.applymap(lambda s:s.upper() if type(s) == str else s)

# Fixation Losses: fl_1/fl_2
# the fixation losse must be less than 2 in 10
df_pec_exams['fl'] = df_pec_exams['fl_1'].div(df_pec_exams['fl_2']).replace(np.NaN, 0)

# If the FP and FN is null it means that during the exam no False True or a False Positive occurred
df_pec_exams['fp'].fillna(0, inplace = True)
df_pec_exams['fn'].fillna(0, inplace = True)
df_pec_exams.head(3)
```

	patient_id	eye	exam_type	exam_date	fl_1	fl_2	fp	fn	fovea	vfi	md	psd	ght	exam_order	fl
0	100	RIGHT	PEC_SINGLE	2021-07-27	0.0	0.0	0.0	0.0	OFF	93.0	-4.54	4.90	OUTSIDE NORMAL LIMITS	1	0.0
1	24698	LEFT	PEC_SINGLE	2021-07-28	0.0	0.0	0.0	5.0	OFF	99.0	-0.40	1.63	BORDERLINE	1	0.0
2	2291532	LEFT	PEC_SINGLE	2021-07-26	0.0	0.0	0.0	0.0	OFF	99.0	-0.27	1.89	OUTSIDE NORMAL LIMITS	1	0.0

Biometry and PEC - Pupil Diameter

```
#The Pupil dilatation (pd) is present on the pec_single exams and also on biometries
#Concatenates PEC data and Biometry data where the pd value is not Null, keeping only the patient_id, eye, exam_type

df_pd_pec = df_pec[df_pec['pd'].notnull()][['patient_id','eye','exam_type','exam_date','exam_time','pd']]
df_pd_pec = df_pd_pec.applymap(lambda s:s.upper() if type(s) == str else s)

df_pd_bio = df_biometry_exams[df_biometry_exams['pd'].notnull()][['patient_id','eye','exam_type','exam_date','exam_t

df_pd = pd.concat([df_pd_pec, df_pd_bio], ignore_index=True)
df_pd.head(3)
```

	patient_id	eye	exam_type	exam_date	exam_time	pd
0	100	RIGHT	PEC_SINGLE	2021-07-27	11:12	5.9
1	24698	LEFT	PEC_SINGLE	2021-07-28	09:12	4.7
2	2291532	LEFT	PEC_SINGLE	2021-07-26	12:30	5.9

RNFLs

```
df_rnfl = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"reference" AS "reference",
"c_mrw" AS "c_mrw",
"mrw_ts_1" AS "mrw_ts_value",
"mrw_ns_1" AS "mrw_ns_value",
"mrw_t_1" AS "mrw_t_value",
"mrw_g_1" AS "mrw_g_value",
"mrw_n_1" AS "mrw_n_value",
"mrw_ti_1" AS "mrw_ti_value",
"mrw_ni_1" AS "mrw_ni_value",
"mrw_ts_2" AS "mrw_ts_perc",
"mrw_ns_2" AS "mrw_ns_perc",
"mrw_t_2" AS "mrw_t_perc",
"mrw_g_2" AS "mrw_g_perc",
"mrw_n_2" AS "mrw_n_perc",
"mrw_ti_2" AS "mrw_ti_perc",
"mrw_ni_2" AS "mrw_ni_perc",
"c_rnflt" AS "c_rnflt",
"rnflt_ts_1" AS "rnflt_ts_value",
"rnflt_ns_1" AS "rnflt_ns_value",
"rnflt_t_1" AS "rnflt_t_value",
```

```
"rnflt_g_1" AS "rnflt_g_value",
"rnflt_n_1" AS "rnflt_n_value",
"rnflt_ti_1" AS "rnflt_ti_value",
"rnflt_ni_1" AS "rnflt_ni_value",
"rnflt_ts_2" AS "rnflt_ts_perc",
"rnflt_ns_2" AS "rnflt_ns_perc",
"rnflt_t_2" AS "rnflt_t_perc",
"rnflt_g_2" AS "rnflt_g_perc",
"rnflt_n_2" AS "rnflt_n_perc",
"rnflt_ti_2" AS "rnflt_ti_perc",
"rnflt_ni_2" AS "rnflt_ni_perc"
FROM e_rnfl
WHERE delivery_id <= {}'''.format(delivery_thrsh_global))



df_rnfl_cr = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"reference" AS "reference",
"c_rnflt" AS "c_rnflt",
"rnflt_ts_1" AS "rnflt_ts_value",
"rnflt_ns_1" AS "rnflt_ns_value",
"rnflt_t_1" AS "rnflt_t_value",
```

```
"rnflt_g_1" AS "rnflt_g_value",
"rnflt_n_1" AS "rnflt_n_value",
"rnflt_ti_1" AS "rnflt_ti_value",
"rnflt_ni_1" AS "rnflt_ni_value",
"rnflt_ts_2" AS "rnflt_ts_p50",
"rnflt_ns_2" AS "rnflt_ns_p50",
"rnflt_t_2" AS "rnflt_t_p50",
"rnflt_g_2" AS "rnflt_g_p50",
"rnflt_n_2" AS "rnflt_n_p50",
"rnflt_ti_2" AS "rnflt_ti_p50",
"rnflt_ni_2" AS "rnflt_ni_p50"
FROM e_rnfl_cr
WHERE delivery_id <= {}'''.format(delivery_thrsh_global))


df_rnfl_ou = postgresql_to_dataframe(
    conn,
    '''SELECT "patient_id" AS "patient_id",
"eye" AS "eye",
"exam_type" AS "exam_type",
"exam_date" AS "exam_date",
"reference" AS "reference",
"c_rnflt" AS "c_rnflt",
"rnflt_ts_1" AS "rnflt_ts_value",
"rnflt_ns_1" AS "rnflt_ns_value",
"rnflt_t_1" AS "rnflt_t_value",
"rnflt_g_1" AS "rnflt_g_value",
```

```
"rnflt_n_1" AS "rnflt_n_value",
"rnflt_ti_1" AS "rnflt_ti_value",
"rnflt_ni_1" AS "rnflt_ni_value",
"rnflt_ts_2" AS "rnflt_ts_p50",
"rnflt_ns_2" AS "rnflt_ns_p50",
"rnflt_t_2" AS "rnflt_t_p50",
"rnflt_g_2" AS "rnflt_g_p50",
"rnflt_n_2" AS "rnflt_n_p50",
"rnflt_ti_2" AS "rnflt_ti_p50",
"rnflt_ni_2" AS "rnflt_ni_p50",
"rnflt_s" AS "rnflt_4s_value",
"rnflt_n" AS "rnflt_4n_value",
"rnflt_i" AS "rnflt_4i_value",
"rnflt_t" AS "rnflt_4t_value"
FROM e_rnfl_ou
WHERE delivery_id <= {}'''.format(delivery_thrsh_global))


df_rnfl = pd.concat([df_rnfl, df_rnfl_cr, df_rnfl_ou], axis=0)

for col in df_rnfl:
    if pd.api.types.is_object_dtype(df_rnfl[col].dtype):
        df_rnfl[col] = df_rnfl[col].apply(lambda x: None if type(x) == float and np.isnan(x) else x)
df_rnfl.head(3)
```

	patient_id	eye	exam_type	exam_date	reference	c_mrw	mrw_ts_value	mrw_ns_value	mrw_t_value	mrw_g_value	...	rnflt_ns_p5
0	465	right	RNFL	2021-07-20	European Descent (2014)	Borderline	194.0	224.0	170.0	238.0	...	Na
1	2180966	right	RNFL	2021-07-22	European Descent (2014)	Outside Normal Limits	145.0	171.0	152.0	181.0	...	Na
2	2241844	right	RNFL	2021-07-21	European Descent (2014)	Borderline	214.0	214.0	177.0	226.0	...	Na

3 rows × 46 columns

RNFLs Preprocessing

```
# Change the fields names
df_rnfl_exams= df_rnfl.copy()

# All string fileds in upper.
df_rnfl_exams = df_rnfl_exams.applymap(lambda s:s.upper() if type(s) == str else s)
df_rnfl_exams.head(3)
```

	patient_id	eye	exam_type	exam_date	reference	c_mrw	mrw_ts_value	mrw_ns_value	mrw_t_value	mrw_g_value	...	rnflt_
0	465	RIGHT	RNFL	2021-07-20	EUROPEAN DESCENT (2014)	BORDERLINE	194.0	224.0	170.0	238.0	...	
1	2180966	RIGHT	RNFL	2021-07-22	EUROPEAN DESCENT (2014)	OUTSIDE NORMAL LIMITS	145.0	171.0	152.0	181.0	...	
2	2241844	RIGHT	RNFL	2021-07-21	EUROPEAN DESCENT (2014)	BORDERLINE	214.0	214.0	177.0	226.0	...	

3 rows × 46 columns

Dataset Creation

Group all exams by the pair (patient/eye)

```
df_patientsByPatient = df_patients.groupby(['patient_id'])
df_diagnosticByPatientEye = df_diagnostic.groupby(['patient_id','eye'])
df_pec_examsByPatientEye = df_pec_exams.groupby(['patient_id','eye'])
df_biometry_examsByPatientEye = df_biometry_exams.groupby(['patient_id','eye'])
df_rnfl_examsByPatientEye = df_rnfl_exams.groupby(['patient_id','eye'])
df_pd_ByPatientEye = df_pd.groupby(['patient_id','eye'])
df_ot_ByPatientEye = df_ot.groupby(['patient_id','eye'])
```

```
df_diagnostic.groupby(['patient_id','eye'])
```

Set Patient List

```
patientList = list(set(df_patients.patient_id))
```

Generate Patient Summary

```
eye_evals = []
for patient in tqdm(patientList):
    if patient != 2277199:
        eye_evals.append(createPatientSummary(patient))
```

Eye Evaluation DataFrame (Final Dataset)

```
#Dataframe with summary of the evaluation of each patient's eye

#GLOBAL: patient_id, age, gender, label (GLAUCOMA/NORMAL/OUTRO), other_eye_glau_flag, glaucoma_flag, eye_sick_label,
#PEC: vfi_last, vfi_min, vfi_max, vfi_mean, vfi_std, md_last, md_min, md_max, md_mean, md_std, psd_last, psd_min, ps
#BIO: al, cct, ad, acd, lt, k1_value, k1_degree, k2_value, k2_degree, at_value, ast_degree, wtw, ic_x, ic_y, pd, pc_
#MRW: mrw_result, mrw_ts_value, mrw_ts_perc, mrw_ns_value, mrw_ns_perc, mrw_t_value, mrw_t_perc, mrw_g_value, mrw_g_
#RNFLT: rnflt_result, rnflt_ts_value, rnflt_ts_perc, rnflt_ns_value, rnflt_ns_perc, rnflt_t_value, rnflt_t_perc, rnf

eye_evaluation = [item for line in eye_evals for item in line]
```

```
df_eye = pd.DataFrame(eye_evaluation, columns=['patient_id', 'age', 'gender', 'label', 'other_eye_glau_flag','glaucc
                      ,'vfi_last', 'vfi_min', 'vfi_max', 'vfi_mean', 'vfi_std', 'md_last', 'md_min', 'md_max', 'md_mea
                      ,'psd_last', 'psd_min', 'psd_max', 'psd_mean', 'psd_std', 'last_fl', 'last_ght', 'pec_qtd'\
                      ,'al', 'cct', 'ad', 'acd', 'lt', 'k1_value', 'k1_angle', 'k2_value', 'k2_angle', 'ast_value', 'a
                      ,'pc_x', 'pc_y'\
                      ,'pd'\
                      ,'ot'\
                      ,'mrw_result', 'mrw_g_value', 'mrw_g_perc'\
                      ,'mrw_ts_value', 'mrw_ns_value', 'mrw_t_value', 'mrw_n_value', 'mrw_ti_value', 'mrw_ni_value'\
                      ,'mrw_ts_perc', 'mrw_ns_perc', 'mrw_t_perc', 'mrw_n_perc', 'mrw_ti_perc', 'mrw_ni_perc'\
                      ,'rnflt_result', 'rnflt_g_value', 'rnflt_g_perc'\
                      ,'rnflt_ts_value', 'rnflt_ns_value', 'rnflt_t_value', 'rnflt_n_value', 'rnflt_ti_value', 'rnflt_
                      ,'rnflt_ts_perc', 'rnflt_ns_perc', 'rnflt_t_perc', 'rnflt_n_perc', 'rnflt_ti_perc', 'rnflt_ni_pe
                      ,'mrw_value_min', 'mrw_value_max', 'mrw_perc_min', 'mrw_perc_max'\
                      ,'rnflt_value_min', 'rnflt_value_max', 'rnflt_perc_min', 'rnflt_perc_max'\
                      ,'rnflt_outsides', 'rnflt_borderlines'
                      ])
df_eye
```

	patient_id	age	gender	label	other_eye_glau_flag	glaucoma_flag	eye_sick_label	eye	vfi_last	vfi_min	...	mrw_value_mi
0	2287616	67.0	M	SUSPEITO DE GLAUCOMA	1	0	0	LEFT	99.0	99.0	...	171.
1	2287616	67.0	M	GLAUCOMA	0	1	1	RIGHT	94.0	94.0	...	28.
2	2289665	79.0	F	OUTRAS PATOLOGIAS	0	0	1	LEFT	99.0	92.0	...	255.
3	2289665	79.0	F	NORMAL	0	0	0	RIGHT	100.0	97.0	...	204.
4	2183174	60.0	F	SUSPEITO DE GLAUCOMA	1	0	0	LEFT	99.0	97.0	...	224.
...	...	...	...	...	...	...	...	...	...	...	...	
618	2179051	57.0	F	OUTRAS PATOLOGIAS	0	0	1	LEFT	97.0	83.0	...	180.
619	2179051	57.0	F	NORMAL	0	0	0	RIGHT	100.0	99.0	...	173.
620	2138093	69.0	M	SUSPEITO DE GLAUCOMA	0	0	0	LEFT	95.0	95.0	...	165.
621	2138093	69.0	M	SUSPEITO DE GLAUCOMA	0	0	0	RIGHT	86.0	86.0	...	149.
622	116733	72.0	F	NORMAL	0	0	0	RIGHT	98.0	96.0	...	183.

623 rows × 84 columns

Export Dataset to MinIO

```
df_eye.to_parquet('s3://'+bucket_name+"/"+bucket_object, storage_options=s3_minio_kwargs)
```

EDA

All the steps of performing the Exploratory Data Analysis, in the Jupiter Notebooks, are presented in the images below. The goal was to understand the dataset and the type of data we had. The prints shown below correspond to the notebook 2.1.

Imports

```
import pandas as pd
from minio import Minio
from minio.error import MinioException
import pandas.io.sql as psql
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.stats as st
from tqdm.notebook import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

Setup I/O Access

```
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

print(dotenv_path)
```

```
/home/jovyan/.env
```

```
s3_minio_kwargs = {
    "key": os.environ['MINIO_ACCESS_KEY'],
    "secret": os.environ['MINIO_SECRET_KEY'],
    "client_kwargs": {
        'endpoint_url': "http://" + os.environ['MINIO_ENDPOINT']
    }
}

minioClient = Minio(os.environ['MINIO_ENDPOINT'],
                    access_key=os.environ['MINIO_ACCESS_KEY'],
                    secret_key=os.environ['MINIO_SECRET_KEY'],
                    secure=False)
```

Load Dataset from MinIO

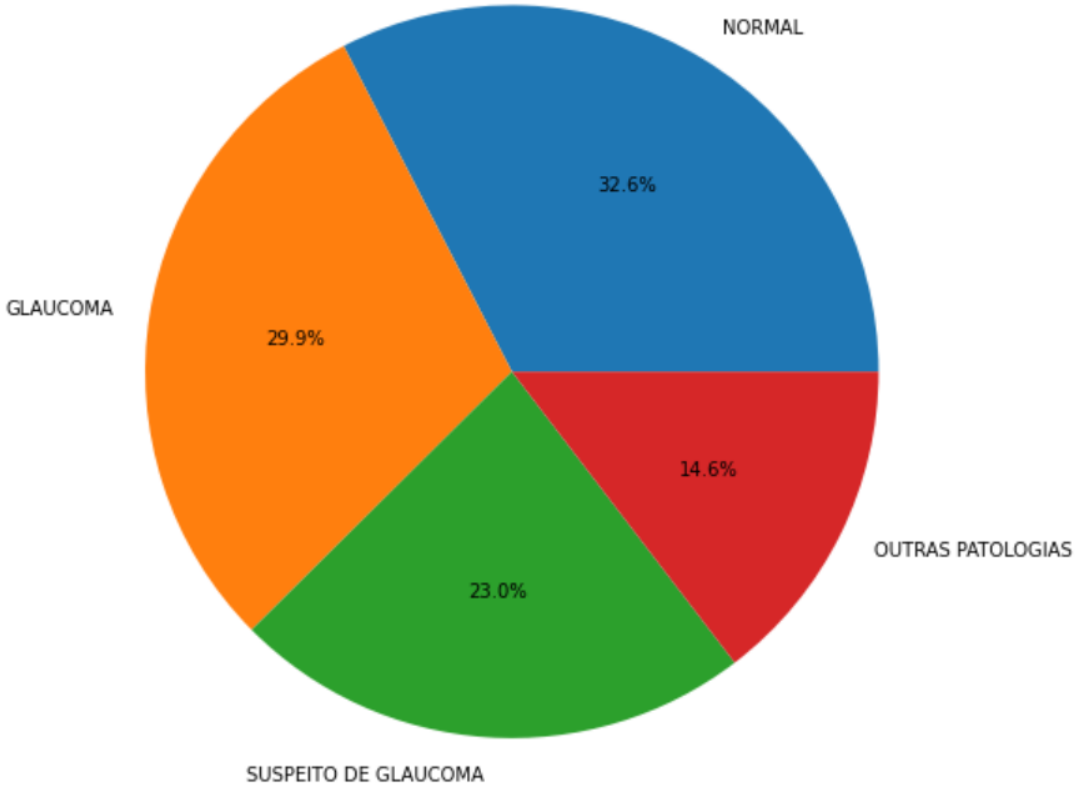
```
df = pd.read_parquet("s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet",
                     storage_options=s3_minio_kwargs)
```

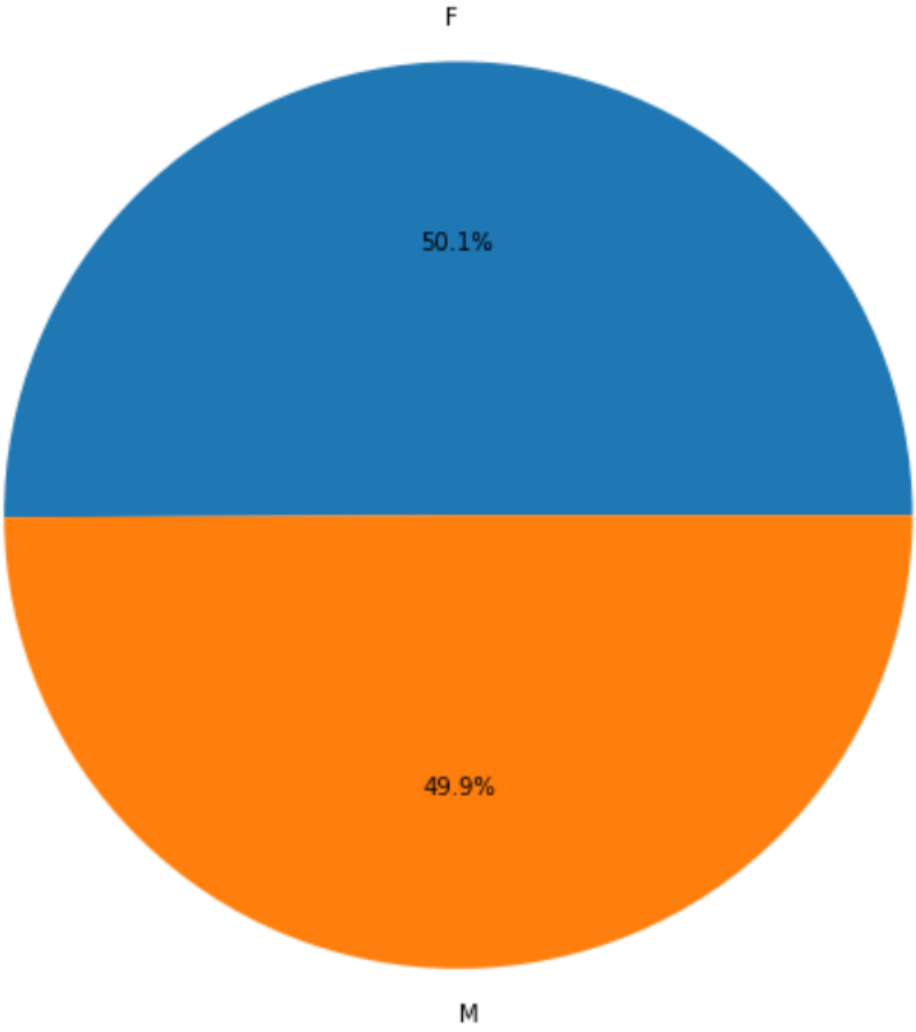
Dataset Composition

```
df.shape
```

```
(623, 84)
```

```
fig, (ax1) = plt.subplots(1,1,figsize=(10,10)) #ax1 refer to your two pies
ax1.pie(df['label'].value_counts(), labels=df['label'].value_counts().index, autopct = '%1.1f%%')
```


NORMAL
32.6%
GLAUCOMA
29.9%
14.6%
OUTRAS PATOLOGIAS
23.0%
SUSPEITO DE GLAUCOMA



Checking Nulls

```
df.isnull().sum()
```

```
patient_id             0
age                    0
gender                 0
label                  0
other_eye_glau_flag    0
                      ..
rnflt_value_max        6
rnflt_perc_min         6
rnflt_perc_max         6
rnflt_outsides         6
rnflt_borderlines      6
Length: 84, dtype: int64
```

```
df.isnull().sum().sort_values(ascending = False)
```

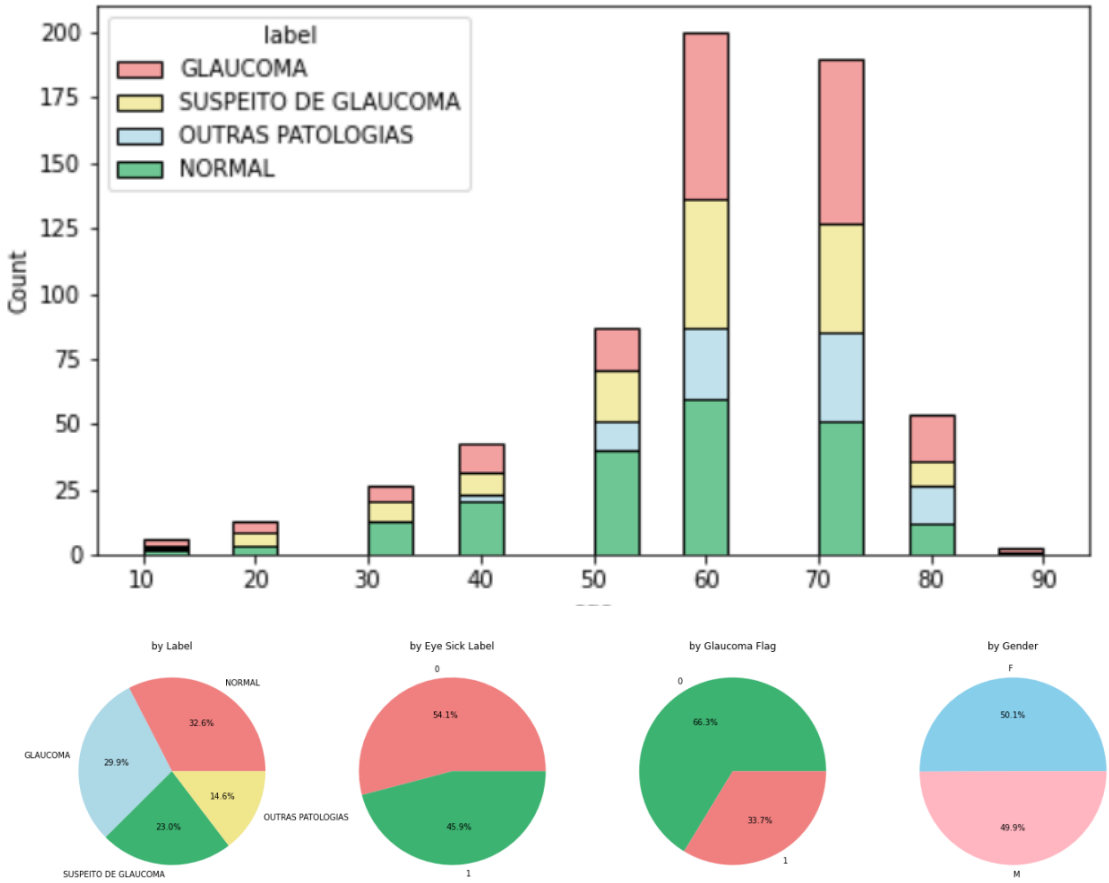
```
pc_y                  215
pc_x                  215
lt                    161
pd                    159
ic_y                  151
                     ...
glaucoma_flag           0
other_eye_glau_flag     0
label                   0
gender                  0
patient_id              0
Length: 84, dtype: int64
```

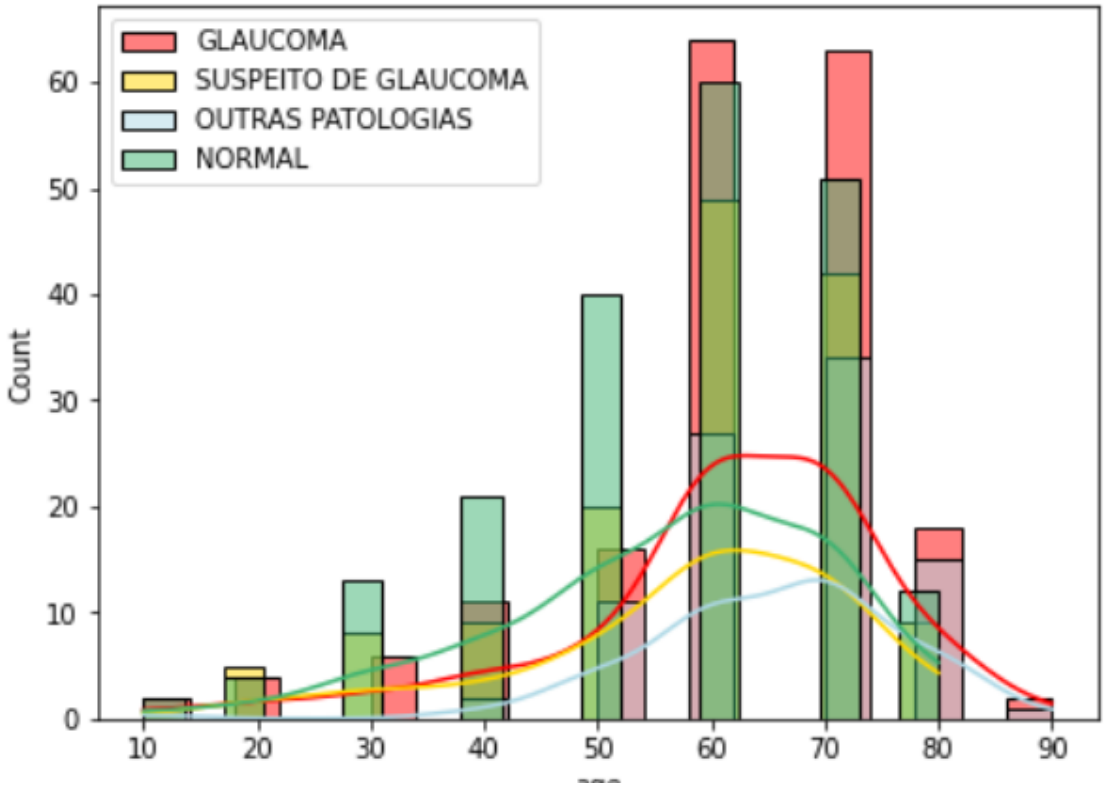
Field	Nulls	Null %
age	0	0.0
other_eye_glau_flag	0	0.0
glaucoma_flag	0	0.0
eye_sick_label	0	0.0
vfi_last	9	1.4
md_last	9	1.4
psd_last	9	1.4
last_fl	9	1.4
last_ght	9	1.4
pec_qtd	9	1.4
al	39	6.3
cct	30	4.8
acd	144	23.1
lt	161	25.8
k1_value	34	5.5
k1_angle	44	7.1
k2_value	34	5.5
k2_angle	44	7.1
ast_value	38	6.1
ast_angle	48	7.7
pd	159	25.5
ot	24	3.9
mrw_result	30	4.8

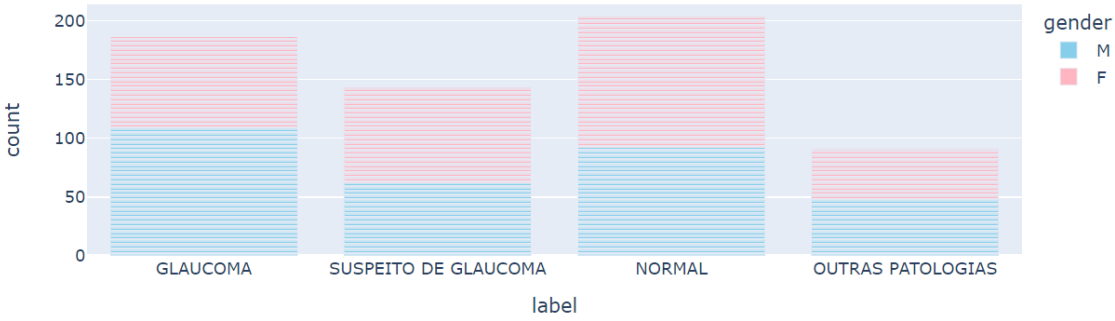
mrw_g_value	30	4.8
mrw_g_perc	30	4.8
mrw_ts_value	30	4.8
mrw_ns_value	30	4.8
mrw_t_value	30	4.8
mrw_n_value	30	4.8
mrw_ti_value	30	4.8
mrw_ni_value	30	4.8
mrw_ts_perc	30	4.8
mrw_ns_perc	30	4.8
mrw_t_perc	30	4.8
mrw_n_perc	30	4.8
mrw_ti_perc	30	4.8
mrw_ni_perc	30	4.8
rnflt_result	6	1.0
rnflt_g_value	8	1.3
rnflt_g_perc	8	1.3
rnflt_ts_value	6	1.0
rnflt_ns_value	6	1.0
rnflt_t_value	8	1.3
rnflt_n_value	6	1.0
rnflt_ti_value	6	1.0
rnflt_ni_value	6	1.0
rnflt_ts_perc	6	1.0
rnflt_ns_perc	6	1.0
rnflt_t_perc	8	1.3
rnflt_n_perc	6	1.0
rnflt_ti_perc	6	1.0
rnflt_ni_perc	6	1.0
mrw_value_min	30	4.8
mrw_value_max	30	4.8

mrw_perc_min	30	4.8
mrw_perc_max	30	4.8
rnflt_value_min	6	1.0
rnflt_value_max	6	1.0
rnflt_perc_min	6	1.0
rnflt_perc_max	6	1.0

Patients Analysis







Data Transformations

The steps performed in the Jupyter Notebooks to transform the data can be found in the images below.

Drop Nulls

```
df = df.dropna()
```

Transforming Numerical Columns in Categorical

```
def transf_perc_to_classification (perc):
    if perc < 1:
        result = 'Outside'
    elif (perc >= 1 and perc < 5):
        result = 'Borderline'
    elif perc >= 5:
        result = 'Normal'
    else:
        result = "NAN"
    return result
```

Getting Numerical Columns

```
numeric_columns = [col for col in df.columns if pd.api.types.is_numeric_dtype(df[col])]
print(len(numeric_columns))
numeric_columns
```

Standardization of values

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[numeric_columns[0:]])
```

```
scaled_data
```

```
array([[ 0.42303194,  0.2429896 , -0.61533794, ..., -0.83275136,
         0.87995205, -0.6867566 ],
       [ 0.42303194,  0.2429896 ,  1.62512327, ...,  0.02377964,
        -0.62853718,  0.65833908],
       [ 0.27366781,  0.2429896 ,  1.62512327, ...,  0.55087564,
        -0.62853718, -0.6867566 ],
       ...,
       [ 0.26777144, -0.45407148, -0.61533794, ...,  0.32027114,
        -0.62853718, -0.6867566 ],
       [ 0.26777144, -0.45407148, -0.61533794, ...,  0.38615814,
         0.37712231,  0.65833908],
       [-2.68158172,  0.94005069, -0.61533794, ...,  0.68264963,
        -0.62853718, -0.6867566 ]])
```

Model Training

The procedure followed to perform the different iterations of model training, such as splitting the data into test and train sets, creating the model, and evaluate metrics, is going to be described below using prints from the Jupyter Notebooks. As it was said above, each iteration is repeated for each of the chosen algorithms, so it was decided to demonstrate only one of the algorithms for each iteration. This will allow the section to become more readable and simpler. There will be no loss of information since the pipeline for the different algorithms is almost identical and demonstrating one of them for each iteration is enough to understand the work done.

Iteration 1

KNN

Imports

```
import pandas as pd
from minio import Minio
from minio.error import MinioException
import pandas.io.sql as psql
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.stats as st
from tqdm.notebook import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

```
# Data Access
from minio import Minio

# Data handling
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# System
import os

# MLFLow
import mlflow
from mlflow.tracking import MlflowClient

# Models
from sklearn.neighbors import KNeighborsClassifier
# Modeling utils
import sklearn

# Explainability
import shap
from sklearn.inspection import permutation_importance

# Serialization
import pickle
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
```

```
# Load utils module
from src.models.model_utils import evaluate_results, display_evaluation, cross_validation_training, get_results
```

```
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

print(dotenv_path)
```

```
/home/jovyan/.env
```

MinIO Setup

```
s3_minio_kwargs = {
    "key": os.environ['MINIO_ACCESS_KEY'],
    "secret": os.environ['MINIO_SECRET_KEY'],
    "client_kwargs": {
        'endpoint_url': "http://" + os.environ['MINIO_ENDPOINT']
    }
}

minioClient = Minio(os.environ['MINIO_ENDPOINT'],
                    access_key=os.environ['MINIO_ACCESS_KEY'],
                    secret_key=os.environ['MINIO_SECRET_KEY'],
                    secure=False)
```

MLFlow Log Function

```
user = 'root'
baseline = 'baseline_2022_2'

def mlf_log(a):
    mlflow.log_param('user',user)
    mlflow.log_param('dataset',baseline)
    mlflow.log_param('algorithm',a)
```

Auxiliar Functions

```
def log_model_metrics(y_test, y_pred):
    mlflow.log_metric('accuracy_score', sklearn.metrics.accuracy_score(y_test, y_pred))
    mlflow.log_metric('balanced_accuracy_score', sklearn.metrics.balanced_accuracy_score(y_test, y_pred))
    mlflow.log_metric('average_precision_score', sklearn.metrics.average_precision_score(y_test, y_pred))
    mlflow.log_metric('precision_score', sklearn.metrics.precision_score(y_test, y_pred))
    mlflow.log_metric('recall_score', sklearn.metrics.recall_score(y_test, y_pred))
    mlflow.log_metric('roc_auc_score',sklearn.metrics.roc_auc_score(y_test, y_pred))
```

```
def compute_feature_importances(reg, X_test, y_test):
    # Mean Decrease in Impurity - MDI
    importances = reg.feature_importances_
    std = np.std([tree.feature_importances_ for tree in reg.estimators_], axis=0)
    # Mean Accuracy Decrease - MAD
    result = permutation_importance(
        reg, X_test, y_test, n_repeats=10, random_state=42, n_jobs=8
    )

    df_forest_importances = pd.DataFrame({"importances_mdi": importances, "std_mdi": std, "importances_mad": result.importances_

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mdi"])["importances_mdi"].plot.bar(yerr=df_forest_importances.sort_values(["i
    ax.set_title("Feature importances using MDI")
    ax.set_ylabel("Mean decrease in impurity")
    fig.tight_layout()
    fig.savefig("artifacts/feature_importance_mdi.png")
    plt.close("all")

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mad"])["importances_mad"].plot.bar(yerr=df_forest_importances.sort_values(["i
    ax.set_title("Feature importances using permutation on full model")
    ax.set_ylabel("Mean accuracy decrease")
    fig.tight_layout()

    fig.savefig("artifacts/feature_importance_mad.png")
    #plt.show()
    plt.close("all")

    mlflow.log_artifact('artifacts/feature_importance_mdi.png', 'explainability')
    mlflow.log_artifact('artifacts/feature_importance_mad.png', 'explainability')

    return df_forest_importances
```

Load Dataset

```
df = pd.read_parquet("s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet",
                     storage_options=s3_minio_kwargs)
```

Preprocessing

Drop columns and registers with nulls

```
df = df.drop(['label', 'other_eye_glau_flag', 'eye_sick_label'], axis=1)

df["patient_id"] = df["patient_id"].astype("category")
df=df.dropna()
```

Train-Test split

```
df_train, df_test = train_test_split(df, test_size=0.25)
```

Input data and labels

```
X_train, y_train = df_train.drop(columns='glaucoma_flag'), df_train.glaucoma_flag
X_test, y_test = df_test.drop(columns='glaucoma_flag'), df_test.glaucoma_flag
```

Numeric columns and Scaler

```
numeric_columns = [col for col in X_train.columns if pd.api.types.is_numeric_dtype(df[col])]
print(len(numeric_columns))

X_train = X_train[numeric_columns[0:]]
X_test = X_test[numeric_columns[0:]]
```

```
74
```

```
# Fit scaler
sc = StandardScaler().fit(X_train)
# Apply
X_train_sc = pd.DataFrame(sc.transform(X_train), columns=X_train.columns)
X_test_sc = pd.DataFrame(sc.transform(X_test), columns=X_test.columns)
```

MLFlow Setup

```
# Set MLFlow Client URL
mlflow.set_tracking_uri(os.environ['MLFLOW_URL'])
display(os.environ['MLFLOW_URL'])
```

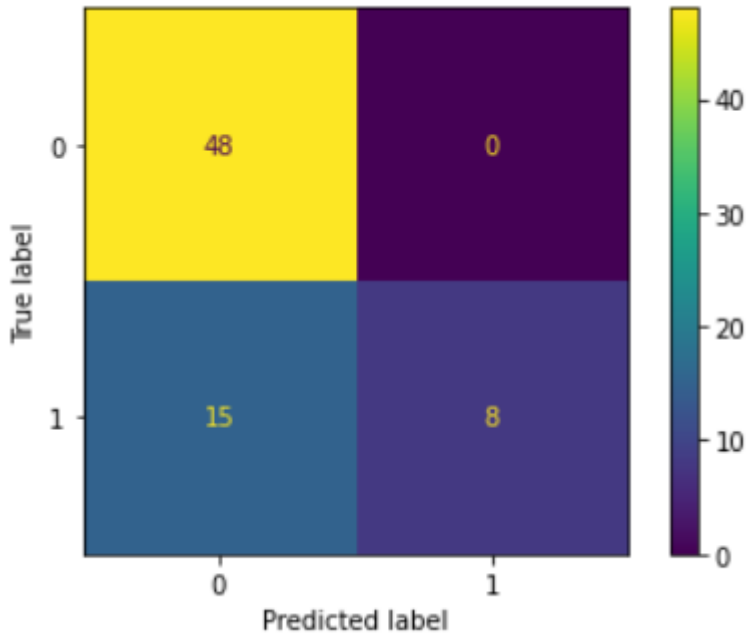
```
'http://mlflow.k8s-cognitive.alticelabs.com/'
```

```
# Set experiment
mlflow.set_experiment('Glaucoma_test_MF')
```

```
<Experiment: artifact_location='s3://mlflow/mlflow/artifacts/27', experiment_id='27', lifecycle_stage='active', name='Glaucoma_
test_MF', tags={}>
```

Modeling

```
# Activate autolog
mlflow.sklearn.autolog()

# Start run
with mlflow.start_run(run_name='1_KNN_glaucoma_flag') as run:

    # Log params
    mlf_log('KNN')

    # Train the model
    reg = KNeighborsClassifier().fit(X_train_sc, y_train)


    #show metrics on mlflow
    log_model_metrics(y_test, reg.predict(X_test_sc))


    metrics.plot_confusion_matrix(reg, X_test_sc, y_test)
    mlflow.log_artifact('artifacts/validation_cm.png', 'validation')
```



Iteration 2

Logistic Regression

Imports

```
import pandas as pd
from minio import Minio
from minio.error import MinioException
import pandas.io.sql as psql
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.stats as st
from tqdm.notebook import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

```
# Data Access
from minio import Minio

# Data handling
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# System
import os

# MLFLow
import mlflow
from mlflow.tracking import MlflowClient

# Models
from sklearn.neighbors import KNeighborsClassifier
# Modeling utils
import sklearn

# Explainability
import shap
from sklearn.inspection import permutation_importance

# Modeling metrics
from sklearn import metrics
```

```
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

print(dotenv_path)
```

MinIO Setup

```
s3_minio_kwargs = {
    "key": os.environ['MINIO_ACCESS_KEY'],
    "secret": os.environ['MINIO_SECRET_KEY'],
    "client_kwargs": {
        'endpoint_url': "http://" + os.environ['MINIO_ENDPOINT']
    }
}

minioClient = Minio(os.environ['MINIO_ENDPOINT'],
                    access_key=os.environ['MINIO_ACCESS_KEY'],
                    secret_key=os.environ['MINIO_SECRET_KEY'],
                    secure=False)
```

MLFlow log function

```
user = 'root'
baseline = 'baseline_2022_2'

def mlf_log(a):
    mlflow.log_param('user',user)
    mlflow.log_param('dataset',baseline)
    mlflow.log_param('algorithm',a)
```

Auxiliar functions

```
def log_model_metrics(y_test, y_pred):
    mlflow.log_metric('accuracy_score', sklearn.metrics.accuracy_score(y_test, y_pred))
    mlflow.log_metric('balanced_accuracy_score', sklearn.metrics.balanced_accuracy_score(y_test, y_pred))
    mlflow.log_metric('average_precision_score', sklearn.metrics.average_precision_score(y_test, y_pred))
    mlflow.log_metric('precision_score', sklearn.metrics.precision_score(y_test, y_pred))
    mlflow.log_metric('recall_score', sklearn.metrics.recall_score(y_test, y_pred))
    mlflow.log_metric('roc_auc_score',sklearn.metrics.roc_auc_score(y_test, y_pred))
```

```
def compute_feature_importances(reg, X_test, y_test):
    # Mean Decrease in Impurity - MDI
    importances = reg.feature_importances_
    std = np.std([tree.feature_importances_ for tree in reg.estimators_], axis=0)
    # Mean Accuracy Decrease - MAD
    result = permutation_importance(
        reg, X_test, y_test, n_repeats=10, random_state=42, n_jobs=8
    )

    df_forest_importances = pd.DataFrame({"importances_mdi": importances, "std_mdi": std, "importances_mad": result.importances_r

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mdi"])["importances_mdi"].plot.bar(yerr=df_forest_importances.sort_values(["i
    ax.set_title("Feature importances using MDI")
    ax.set_ylabel("Mean decrease in impurity")
    fig.tight_layout()
    fig.savefig("artifacts/feature_importance_mdi.png")
    plt.close("all")

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mad"])["importances_mad"].plot.bar(yerr=df_forest_importances.sort_values(["i
    ax.set_title("Feature importances using permutation on full model")
    ax.set_ylabel("Mean accuracy decrease")
    fig.tight_layout()

    fig.savefig("artifacts/feature_importance_mad.png")
    #plt.show()
    plt.close("all")

    mlflow.log_artifact('artifacts/feature_importance_mdi.png', 'explainability')
    mlflow.log_artifact('artifacts/feature_importance_mad.png', 'explainability')

    return df_forest_importances
```

Load dataset

```
df = pd.read_parquet("s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet",
                     storage_options=s3_minio_kwargs)
```

Preprocessing

Drop registers with nulls and choose columns to dataframe

```
df=df.dropna()
```

```
df = df[['patient_id', 'age', 'gender', 'glaucoma_flag', 'eye', 'vfi_last', 'md_last', 'psd_last', 'last_ght', 'al', 'cct', \
       'ot',\
       'mrw_result', 'mrw_ts_perc', 'mrw_ns_perc', 'mrw_t_perc', 'mrw_g_perc', 'mrw_n_perc', 'mrw_ti_perc', 'mrw_ni_perc',\
       'mrw_perc_min', 'mrw_perc_max', \
       'rnflt_result', 'rnflt_ts_perc', 'rnflt_ns_perc', 'rnflt_t_perc', 'rnflt_g_perc', 'rnflt_n_perc', 'rnflt_ti_perc',
         'rnflt_ni_perc',\
       'rnflt_perc_min', 'rnflt_perc_max', \
       'rnflt_outsides', 'rnflt_borderlines', 'mrw_g_value', 'rnflt_g_value', 'mrw_value_min', 'rnflt_value_min']]

df["patient_id"] = df["patient_id"].astype("category")
```

Train-Test split

```
df_train, df_test = train_test_split(df, test_size=0.25)
```

Input data and labels

```
X_train, y_train = df_train.drop(columns='glaucoma_flag'), df_train.glaucoma_flag
X_test, y_test = df_test.drop(columns='glaucoma_flag'), df_test.glaucoma_flag
```

Numeric columns and Scaler

```
numeric_columns = [col for col in X_train.columns if pd.api.types.is_numeric_dtype(df[col])]
print(len(numeric_columns))

X_train = X_train[numeric_columns[0:]]
X_test = X_test[numeric_columns[0:]]
```

```
31
```

```
# Fit scaler
sc = StandardScaler().fit(X_train)
# Apply
X_train_sc = pd.DataFrame(sc.transform(X_train), columns=X_train.columns)
X_test_sc = pd.DataFrame(sc.transform(X_test), columns=X_test.columns)
```

MLFlow Setup

```
# Set MLFlow Client URL
mlflow.set_tracking_uri(os.environ['MLFLOW_URL'])
display(os.environ['MLFLOW_URL'])
```

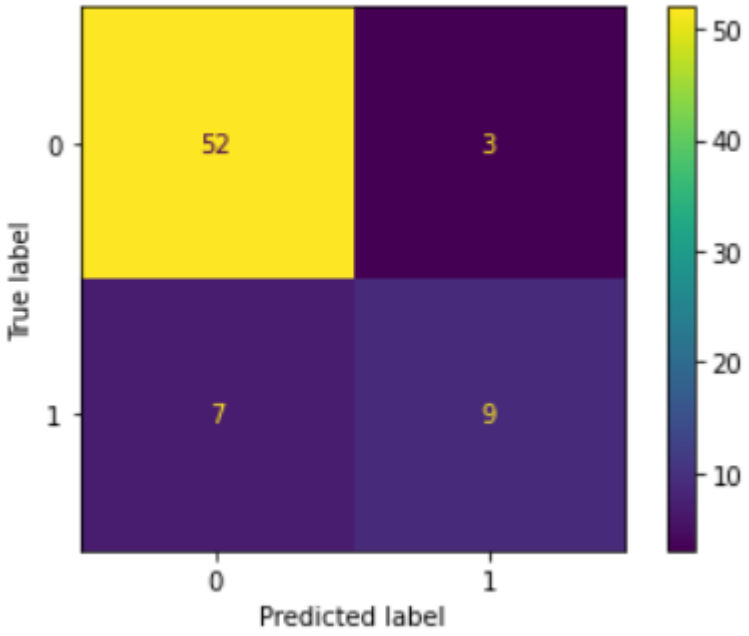
```
'http://mlflow.k8s-cognitive.alticelabs.com/'
```

```
# Set experiment
mlflow.set_experiment('Glaucoma_test_MF')
```

```
<Experiment: artifact_location='s3://mlflow/mlflow/artifacts/27', experiment_id='27', lifecycle_stage='active', name='Glaucoma_
test_MF', tags={}>
```

Modeling

```
# Activate autolog
mlflow.sklearn.autolog()

# Start run
with mlflow.start_run(run_name='2_lr_glaucoma_flag') as run:

    # Log params
    mlf_log('Logistic Regression')

    # Train the model
    reg = LogisticRegression().fit(X_train_sc, y_train)

    #show metrics on mlflow
    log_model_metrics(y_test, reg.predict(X_test_sc))

    # Log metrics
    mlflow.log_metric('r2_score', metrics.r2_score(y_test, reg.predict(X_test_sc)))

    metrics.plot_confusion_matrix(reg, X_test_sc, y_test)
    mlflow.log_artifact('artifacts/validation_cm.png', 'validation')
```



Iteration 3

LGBM

Imports

```
# Data Access
from minio import Minio

# Data handling
import pandas as pd
import numpy as np

# Visualization
import matplotlib as mpl
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm

# Preprocessing
from sklearn.model_selection import train_test_split
from src.cowutils.dataprocessing.preproc import Transformer

# restore the used matplotlib backend
mpl.use(mpl_backend)

# System
import os

# MLFLow
import mlflow
from mlflow.tracking import MlflowClient

# Models
import lightgbm

# Modeling utils
import sklearn
```

```
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

print(dotenv_path)
```

```
/home/jovyan/.env
```

```
# Load utils module
from src.models.model_utils import evaluate_results, display_evaluation, cross_validation_training, get_results,
generate_tree_shap_summary, print_logged_info, get_roc_auc, print_model_io_metadata
```

MinIO Setup

```
s3_minio_kwargs = {
    "key": os.environ['MINIO_ACCESS_KEY'],
    "secret": os.environ['MINIO_SECRET_KEY'],
    "client_kwargs": {
        'endpoint_url': "http://" + os.environ['MINIO_ENDPOINT']
    }
}

minioClient = Minio(os.environ['MINIO_ENDPOINT'],
                    access_key=os.environ['MINIO_ACCESS_KEY'],
                    secret_key=os.environ['MINIO_SECRET_KEY'],
                    secure=False)
```

MLFlow Setup

```
mlf_dataset = "s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet"
mlf_experiment = 'Glaucoma_test_MF'
run_name = '3_LGBM_glaucoma_flag'
```

```
import string

formatter = string.Formatter()

parsed = list(formatter.parse(mlf_dataset))
if parsed[0][1] == "latest":

    prot, path = parsed[0][0].split("s3://")
    s = path.split("/")
    bucket_name = s[0]
    object_path = "/".join(s[1:])

    mlf_dataset_file = None
    mlf_dataset_file_modified = None
    for file in minioClient.list_objects(bucket_name, object_path):
        if not file.is_dir:
            if mlf_dataset_file_modified is None or mlf_dataset_file_modified < file.last_modified:
                mlf_dataset_file_modified = file.last_modified
                mlf_dataset_file = file.object_name

    mlf_dataset_file = "s3://" + bucket_name + "/" + mlf_dataset_file
else:
    mlf_dataset_file = mlf_dataset

print(mlf_dataset_file)
```

```
s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet
```

```
mlflow.set_tracking_uri(os.environ['MLFLOW_URL'])

tracking_uri = mlflow.get_tracking_uri()
print("Current tracking uri: {}".format(tracking_uri))

mlflow.set_experiment(mlf_experiment)
```

```
Current tracking uri: http://mlflow.k8s-cognitive.alticelabs.com/

<Experiment: artifact_location='s3://mlflow/mlflow/artifacts/27', experiment_id='27', lifecycle_stage='active', name='Glaucoma_
test_MF', tags={}>
```

Load Dataset

```
df_ds = pd.read_parquet(mlf_dataset_file, storage_options=s3_minio_kwargs)
```

Preprocessing

```
df_ds=df_ds.dropna()
```

```
t = Transformer()

columns_to_use = ['patient_id','age', 'gender', 'eye', 'vfi_last', 'md_last', 'psd_last', 'last_ght', 'al', 'cct',
                  'ot', 'mrw_g_value', 'rnflt_g_value', 'glaucoma_flag']
df_ds, dropped_columns = t.drop_columns(df_ds, [col for col in df_ds if col not in columns_to_use])

df_ds = t.drop_na_rows(df_ds, 1, [])

df_ds, category_indexes = t.get_and_transform_data_types(df_ds)
```

Train-Test split

```
df_train, df_test = train_test_split(df_ds.sample(frac=1), test_size=0.25)
```

Auxiliar Functions

```
def log_model_metrics(y_test, y_pred):
    mlflow.log_metric('accuracy_score', sklearn.metrics.accuracy_score(y_test, y_pred))
    mlflow.log_metric('balanced_accuracy_score', sklearn.metrics.balanced_accuracy_score(y_test, y_pred))
    mlflow.log_metric('average_precision_score', sklearn.metrics.average_precision_score(y_test, y_pred))
    mlflow.log_metric('precision_score', sklearn.metrics.precision_score(y_test, y_pred))
    mlflow.log_metric('recall_score', sklearn.metrics.recall_score(y_test, y_pred))
    mlflow.log_metric('roc_auc_score',sklearn.metrics.roc_auc_score(y_test, y_pred))
```

```
def log_features_importances(features, booster):
    fig, ax = plt.subplots(figsize=(12,8))
    pd.DataFrame(list(zip(features, booster.feature_importance(importance_type='split'))), columns=("feature", "importance")).sor
    fig.tight_layout()
    fig.savefig("artifacts/feature_importance_split.png")
    #plt.show()
    plt.close("all")

    fig, ax = plt.subplots(figsize=(12,8))
    pd.DataFrame(list(zip(features, booster.feature_importance(importance_type='gain'))), columns=("feature", "importance")).sort
    fig.tight_layout()
    fig.savefig("artifacts/feature_importance_gain.png")
    #plt.show()
    plt.close("all")

    mlflow.log_artifact('artifacts/feature_importance_split.png', 'explainability')
    mlflow.log_artifact('artifacts/feature_importance_gain.png', 'explainability')
```

```
oversample = True  # Oversample data before training
num_boost_rounds=100
lgbm_params = {
    'boosting': "rf",
    'objective': 'binary',
    'metric': 'binary_logloss',
    'is_unbalance': not oversample,
    "num_leaves": None,
    "max_depth": None,
    "min_data_in_leaf" : None,
    "bagging_freq": 10,
    "bagging_fraction": 0.9,
}
```

Input data and labels

```
label_column = 'glaucoma_flag'
```

```
ds_valid = df_test.drop(columns=['patient_id'])
ds = df_train.drop(columns=['patient_id'])
```

Modeling

```
# Train the model
mlflow.autolog()
with mlflow.start_run(run_name=run_name) as run:
    mlflow.log_param('dataset', mlf_dataset_file)
    mlflow.log_artifact("artifacts/preprocessing.pickle")

    booster, results = cross_validation_training(ds, features, label_column, n_folds=3, oversample=oversample,
                                                 lgbm_params=lgbm_params, num_boost_rounds=num_boost_rounds)

    # Log metrics for threshold 0.5
    log_model_metrics(ds_valid[label_column], (booster.predict(ds_valid.drop(columns=[label_column]))>0.5).astype(int))

    log_model_artifacts(booster, results, ds_valid, features, label_column)

    log_features_importances(features, booster)

    explainer, shap_values = generate_tree_shap_summary(booster, ds_valid.drop(columns=[label_column]),
                                                         fig_path="artifacts/shap_summary_plot.png", mlflow_path="explainability")

    print_logged_info(mlflow.get_run(run_id=run.info.run_id))

    print_model_io_metadata(run.info.run_id, "model")
```

Iteration 4

Random Forest

Imports

```
import pandas as pd
from minio import Minio
from minio.error import MinioException
import pandas.io.sql as psql
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.stats as st
from tqdm.notebook import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

```
# Data Access
from minio import Minio

# Data handling
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# System
import os

# MLFLow
import mlflow
from mlflow.tracking import MlflowClient

# Models
from sklearn.ensemble import RandomForestClassifier

# Modeling utils
import sklearn

# Modeling metrics
from sklearn import metrics
```

```
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

print(dotenv_path)
```

```
/home/jovyan/.env
```

MinIO Setup

```
s3_minio_kwargs = {
    "key": os.environ['MINIO_ACCESS_KEY'],
    "secret": os.environ['MINIO_SECRET_KEY'],
    "client_kwargs": {
        'endpoint_url': "http://" + os.environ['MINIO_ENDPOINT']
    }
}

minioClient = Minio(os.environ['MINIO_ENDPOINT'],
                    access_key=os.environ['MINIO_ACCESS_KEY'],
                    secret_key=os.environ['MINIO_SECRET_KEY'],
                    secure=False)
```

MLFlow log function

```
user = 'root'
baseline = 'baseline_2022_2'

def mlf_log(a):
    mlflow.log_param('user',user)
    mlflow.log_param('dataset',baseline)
    mlflow.log_param('algorithm',a)
```

Auxiliar Functions

```
def log_model_metrics(y_test, y_pred):
    mlflow.log_metric('accuracy_score', sklearn.metrics.accuracy_score(y_test, y_pred))
    mlflow.log_metric('balanced_accuracy_score', sklearn.metrics.balanced_accuracy_score(y_test, y_pred))
    mlflow.log_metric('average_precision_score', sklearn.metrics.average_precision_score(y_test, y_pred))
    mlflow.log_metric('precision_score', sklearn.metrics.precision_score(y_test, y_pred))
    mlflow.log_metric('recall_score', sklearn.metrics.recall_score(y_test, y_pred))
    mlflow.log_metric('roc_auc_score',sklearn.metrics.roc_auc_score(y_test, y_pred))
```

```
def compute_feature_importances(reg, X_test, y_test):
    # Mean Decrease in Impurity - MDI
    importances = reg.feature_importances_
    std = np.std([tree.feature_importances_ for tree in reg.estimators_], axis=0)
    # Mean Accuracy Decrease - MAD
    result = permutation_importance(
        reg, X_test, y_test, n_repeats=10, random_state=42, n_jobs=8
    )

    df_forest_importances = pd.DataFrame({"importances_mdi": importances, "std_mdi": std, "importances_mad": result.importances_r

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mdi"])["importances_mdi"].plot.bar(yerr=df_forest_importances.sort_values(["i
    ax.set_title("Feature importances using MDI")
    ax.set_ylabel("Mean decrease in impurity")
    fig.tight_layout()
    fig.savefig("artifacts/feature_importance_mdi.png")
    plt.close("all")

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mad"])["importances_mad"].plot.bar(yerr=df_forest_importances.sort_values(["i
    ax.set_title("Feature importances using permutation on full model")
    ax.set_ylabel("Mean accuracy decrease")
    fig.tight_layout()

    fig.savefig("artifacts/feature_importance_mad.png")
    #plt.show()
    plt.close("all")

    mlflow.log_artifact('artifacts/feature_importance_mdi.png', 'explainability')
    mlflow.log_artifact('artifacts/feature_importance_mad.png', 'explainability')

    return df_forest_importances
```

Load dataset

```
df = pd.read_parquet("s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet",
                     storage_options=s3_minio_kwargs)
```

Preprocessing

Drop registers with nulls and choose columns to dataframe

```
df=df.dropna()
```

```
df = df[['patient_id','age', 'gender', 'eye', 'vfi_last', 'md_last', 'psd_last', 'last_ght', 'al', 'cct', 'ot',
         'mrw_g_value', 'rnflt_g_value', 'mrw_ts_value', 'mrw_ns_value', 'mrw_t_value', 'mrw_n_value', 'mrw_ti_value',
         'mrw_ni_value', 'rnflt_ts_value', 'rnflt_ns_value', 'rnflt_t_value', 'rnflt_n_value', 'rnflt_ti_value',
         'rnflt_ni_value', 'mrw_value_min', 'mrw_value_max', 'rnflt_value_min', 'rnflt_value_max', 'glaucoma_flag']]

df["patient_id"] = df["patient_id"].astype("category")
```

Train-Test split

```
df_train, df_test = train_test_split(df, test_size=0.25)
```

Input data and labels

```
X_train, y_train = df_train.drop(columns='glaucoma_flag'), df_train.glaucoma_flag
X_test, y_test = df_test.drop(columns='glaucoma_flag'), df_test.glaucoma_flag
```

Numeric columns and Scaler

```
numeric_columns = [col for col in X_train.columns if pd.api.types.is_numeric_dtype(df[col])]
print(len(numeric_columns))

X_train = X_train[numeric_columns[0:]]
X_test = X_test[numeric_columns[0:]]
```

```
25
```

```
# Fit scaler
sc = StandardScaler().fit(X_train)
# Apply
X_train_sc = pd.DataFrame(sc.transform(X_train), columns=X_train.columns)
X_test_sc = pd.DataFrame(sc.transform(X_test), columns=X_test.columns)
```

MLFlow Setup

```
# Set MLFlow Client URL
mlflow.set_tracking_uri(os.environ['MLFLOW_URL'])
display(os.environ['MLFLOW_URL'])
```

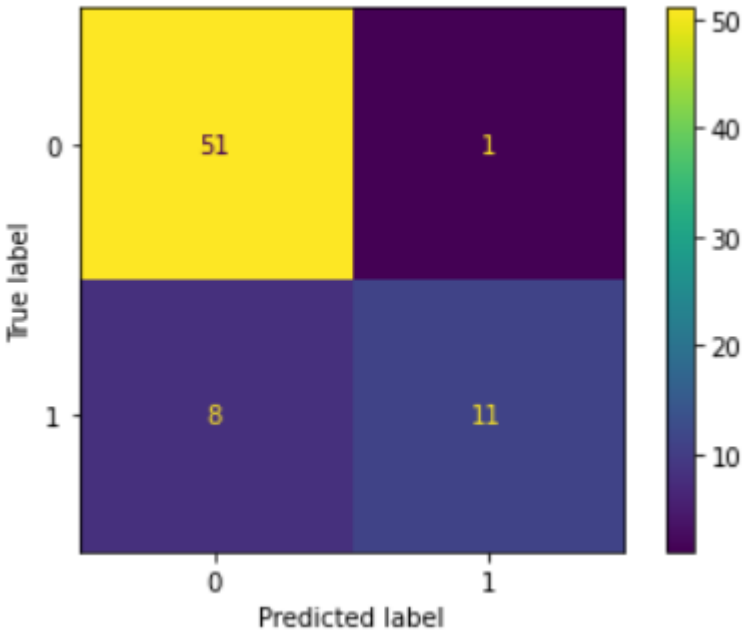
```
'http://mlflow.k8s-cognitive.alticelabs.com/'
```

```
# Set experiment
mlflow.set_experiment('Glaucoma_test_MF')
```

```
<Experiment: artifact_location='s3://mlflow/mlflow/artifacts/27', experiment_id='27', lifecycle_stage='active', name='Glaucoma_
test_MF', tags={}>
```

Modeling

```
# Activate autolog
mlflow.sklearn.autolog()

# Start run
with mlflow.start_run(run_name='4_rf_glaucoma_flag') as run:

    # Log params
    mlf_log('Random Forest')

    # Train the model
    reg = RandomForestClassifier().fit(X_train_sc, y_train)

    #show metrics on mlflow
    log_model_metrics(y_test, reg.predict(X_test_sc))

    # Log metrics
    mlflow.log_metric('r2_score', metrics.r2_score(y_test, reg.predict(X_test_sc)))

    # Confusion matrix
    metrics.plot_confusion_matrix(reg, X_test_sc, y_test)
    mlflow.log_artifact('artifacts/validation_cm.png', 'validation')
```



Iteration 5

SVM

Imports

```
import pandas as pd
from minio import Minio
from minio.error import MinioException
import pandas.io.sql as psql
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import math
import scipy.stats as st
from tqdm.notebook import tqdm
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

```
# Data Access
from minio import Minio

# Data handling
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt

# Preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# System
import os

# MLFLow
import mlflow
from mlflow.tracking import MlflowClient

# Models
from imblearn.over_sampling import SMOTE
from sklearn.svm import SVC

# Modeling utils
import sklearn

# Modeling metrics
from sklearn import metrics
```

```
# Load utils module
from src.models.model_utils import evaluate_results, display_evaluation, cross_validation_training, get_results
```

```
import os
from dotenv import load_dotenv, find_dotenv

# find .env automagically by walking up directories until it's found
dotenv_path = find_dotenv()

# load up the entries as environment variables
load_dotenv(dotenv_path)

print(dotenv_path)
```

```
/home/jovyan/.env
```

MinIO Setup

```
s3_minio_kwargs = {
    "key": os.environ['MINIO_ACCESS_KEY'],
    "secret": os.environ['MINIO_SECRET_KEY'],
    "client_kwargs": {
        'endpoint_url': "http://" + os.environ['MINIO_ENDPOINT']
    }
}

minioClient = Minio(os.environ['MINIO_ENDPOINT'],
                    access_key=os.environ['MINIO_ACCESS_KEY'],
                    secret_key=os.environ['MINIO_SECRET_KEY'],
                    secure=False)
```

MLFlow log function

```
user = 'root'
baseline = 'baseline_2022_2'

def mlf_log(a):
    mlflow.log_param('user',user)
    mlflow.log_param('dataset',baseline)
    mlflow.log_param('algorithm',a)
```

Auxiliar Functions

```
def log_model_metrics(y_test, y_pred):
    mlflow.log_metric('accuracy_score', sklearn.metrics.accuracy_score(y_test, y_pred))
    mlflow.log_metric('balanced_accuracy_score', sklearn.metrics.balanced_accuracy_score(y_test, y_pred))
    mlflow.log_metric('average_precision_score', sklearn.metrics.average_precision_score(y_test, y_pred))
    mlflow.log_metric('precision_score', sklearn.metrics.precision_score(y_test, y_pred))
    mlflow.log_metric('recall_score', sklearn.metrics.recall_score(y_test, y_pred))
    mlflow.log_metric('roc_auc_score',sklearn.metrics.roc_auc_score(y_test, y_pred))
```

```
def compute_feature_importances(reg, X_test, y_test):
    # Mean Decrease in Impurity - MDI
    importances = reg.feature_importances_
    std = np.std([tree.feature_importances_ for tree in reg.estimators_], axis=0)
    # Mean Accuracy Decrease - MAD
    result = permutation_importance(
        reg, X_test, y_test, n_repeats=10, random_state=42, n_jobs=8
    )

    df_forest_importances = pd.DataFrame({"importances_mdi": importances, "std_mdi": std, "importances_mad": result.importances_

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mdi"])["importances_mdi"].plot.bar(yerr=df_forest_importances.sort_values(["
    ax.set_title("Feature importances using MDI")
    ax.set_ylabel("Mean decrease in impurity")
    fig.tight_layout()
    fig.savefig("artifacts/feature_importance_mdi.png")
    plt.close("all")

    fig, ax = plt.subplots(figsize=(12,8))
    df_forest_importances.sort_values(["importances_mad"])["importances_mad"].plot.bar(yerr=df_forest_importances.sort_values(["
    ax.set_title("Feature importances using permutation on full model")
    ax.set_ylabel("Mean accuracy decrease")
    fig.tight_layout()

    fig.savefig("artifacts/feature_importance_mad.png")
    #plt.show()
    plt.close("all")

    mlflow.log_artifact('artifacts/feature_importance_mdi.png', 'explainability')
    mlflow.log_artifact('artifacts/feature_importance_mad.png', 'explainability')

    return df_forest_importances
```

Load dataset

```
df = pd.read_parquet("s3://glaucoma/data/processed/baselines/baseline_2022_2/ev_DS_alldataset_20220322164932.parquet",
                     storage_options=s3_minio_kwargs)
```

Preprocessing

Drop registers with nulls and choose columns to dataframe

```
df=df.dropna()
```

```
df = df[['patient_id','age', 'gender', 'eye', 'vfi_last', 'md_last', 'psd_last', 'last_ght', 'al', 'cct', 'ot',
         'mrw_g_perc', 'rnflt_g_perc', 'mrw_ts_perc', 'mrw_ns_perc', 'mrw_t_perc', 'mrw_n_perc', 'mrw_ti_perc',
         'mrw_ni_perc', 'rnflt_ts_perc', 'rnflt_ns_perc', 'rnflt_t_perc', 'rnflt_n_perc', 'rnflt_ti_perc',
         'rnflt_ni_perc', 'mrw_perc_min', 'mrw_perc_max', 'rnflt_perc_min', 'rnflt_perc_max', 'glaucoma_flag']]

df["patient_id"] = df["patient_id"].astype("category")
```

Train-Test split

```
df_train, df_test = train_test_split(df, test_size=0.25)
```

Input data and labels

```
X_train, y_train = df_train.drop(columns='glaucoma_flag'), df_train.glaucoma_flag
X_test, y_test = df_test.drop(columns='glaucoma_flag'), df_test.glaucoma_flag
```

Numeric columns and Scaler

```
numeric_columns = [col for col in X_train.columns if pd.api.types.is_numeric_dtype(df[col])]
print(len(numeric_columns))

X_train = X_train[numeric_columns[0:]]
X_test = X_test[numeric_columns[0:]]
```

```
25
```

```
# Fit scaler
sc = StandardScaler().fit(X_train)
# Apply
X_train_sc = pd.DataFrame(sc.transform(X_train), columns=X_train.columns)
X_test_sc = pd.DataFrame(sc.transform(X_test), columns=X_test.columns)
```

Oversampling

```
# unbalanced data: 1/3 Glaucoma cases (glaucoma_flag=1)
smote_sampler = SMOTE(sampling_strategy=0.7, random_state=42)
X_smote, y_smote = smote_sampler.fit_resample(X_train_sc, y_train)
```

MLFlow Setup

```
# Set MLFlow Client URL
mlflow.set_tracking_uri(os.environ['MLFLOW_URL'])
display(os.environ['MLFLOW_URL'])
```

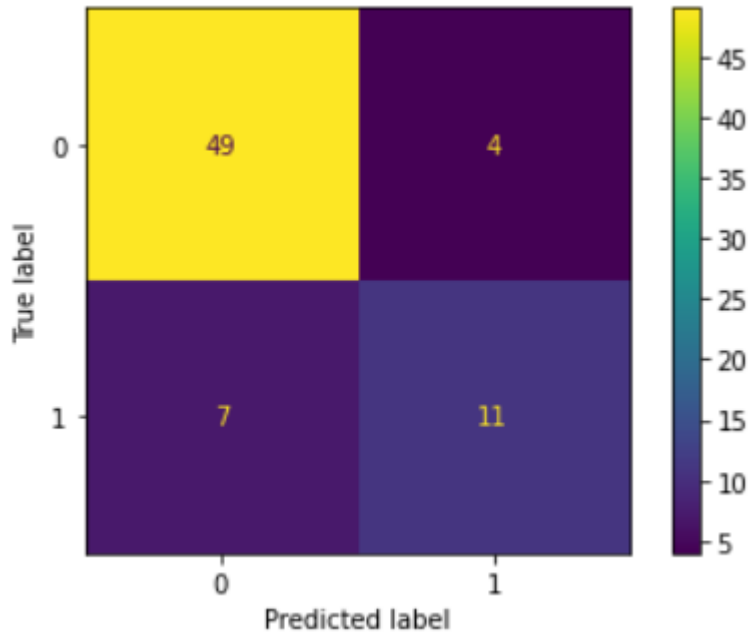
```
'http://mlflow.k8s-cognitive.alticelabs.com/'
```

```
# Set experiment
mlflow.set_experiment('Glaucoma_test_MF')
```

```
<Experiment: artifact_location='s3://mlflow/mlflow/artifacts/27', experiment_id='27', lifecycle_stage='active', name='Glaucoma_
test_MF', tags={}>
```

Modeling

```
# Activate autolog
mlflow.sklearn.autolog()

# Start run
with mlflow.start_run(run_name='5_SVM_glaucoma_flag') as run:

    # Log params
    mlf_log('SVM')

    # Train the model
    reg = SVC(kernel="poly", degree=3, probability=True).fit(X_smote, y_smote)

    log_model_metrics(y_test, reg.predict(X_test_sc))

    metrics.plot_confusion_matrix(reg, X_test_sc, y_test)

    mlflow.log_artifact('artifacts/validation_cm.png', 'validation')
```



Results

Sciki-learn

The images below demonstrate on MLflow the results of the 10 runs performed for each algorithm in all the five iterations.

Iteration 1

LGBM

				Metrics					
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	balanced_accura	precision_score	recall_score	roc_auc_
	17 seconds ago	11.9s	1_LGBM_gla...	0.833	0.585	0.81	0.683	0.757	0.81
	45 seconds ago	12.0s	1_LGBM_gla...	0.826	0.614	0.796	0.732	0.714	0.796
	1 minute ago	12.5s	1_LGBM_gla...	0.833	0.595	0.812	0.69	0.763	0.812
	1 minute ago	12.2s	1_LGBM_gla...	0.886	0.751	0.876	0.826	0.844	0.876
	2 minutes ago	12.8s	1_LGBM_gla...	0.848	0.664	0.822	0.78	0.744	0.822
	2 minutes ago	12.8s	1_LGBM_gla...	0.826	0.595	0.831	0.653	0.842	0.831
	3 minutes ago	13.2s	1_LGBM_gla...	0.841	0.644	0.826	0.733	0.786	0.826
	45 minutes ago	11.9s	1_LGBM_gla...	0.848	0.627	0.811	0.757	0.718	0.811
	45 minutes ago	15.6s	1_LGBM_gla...	0.871	0.771	0.857	0.872	0.788	0.857
	46 minutes ago	15.1s	1_LGBM_gla...	0.841	0.668	0.826	0.761	0.778	0.826

Logistic Regression

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	1 minute ago	1.3s	1_lr_glauco...	0.831	0.608		0.778		0.636	0.777
	6 minutes ago	1.4s	1_lr_glauco...	0.803	0.589		0.633		0.864	0.82
	7 minutes ago	1.3s	1_lr_glauco...	0.887	0.629		0.722		0.813	0.861
	7 minutes ago	1.3s	1_lr_glauco...	0.789	0.592		0.813		0.52	0.727
	7 minutes ago	1.4s	1_lr_glauco...	0.817	0.494		0.647		0.611	0.749
	7 minutes ago	1.2s	1_lr_glauco...	0.859	0.61		0.682		0.833	0.851
	8 minutes ago	1.4s	1_lr_glauco...	0.746	0.468		0.6		0.545	0.691
	8 minutes ago	1.1s	1_lr_glauco...	0.845	0.555		0.706		0.667	0.786
	8 minutes ago	1.3s	1_lr_glauco...	0.789	0.495		0.714		0.476	0.698
	9 minutes ago	1.2s	1_lr_glauco...	0.789	0.427		0.6		0.5	0.693

KNN

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	2 minutes ago	1.7s	1_KNN_glau...	0.775	0.499		0.688		0.5	0.699
	2 minutes ago	2.0s	1_KNN_glau...	0.845	0.617		0.778		0.667	0.793
	3 minutes ago	1.8s	1_KNN_glau...	0.831	0.51		0.714		0.556	0.74
	3 minutes ago	1.8s	1_KNN_glau...	0.845	0.637		0.789		0.682	0.8
	3 minutes ago	1.8s	1_KNN_glau...	0.859	0.546		0.818		0.529	0.746
	3 minutes ago	2.0s	1_KNN_glau...	0.873	0.682		0.875		0.667	0.813
	4 minutes ago	2.0s	1_KNN_glau...	0.732	0.545		0.75		0.444	0.677
	4 minutes ago	1.9s	1_KNN_glau...	0.803	0.521		0.65		0.65	0.756
	4 minutes ago	1.9s	1_KNN_glau...	0.845	0.54		0.591		0.867	0.853
	5 minutes ago	1.9s	1_KNN_glau...	0.761	0.569		0.846		0.423	0.689

SVM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	6 minutes ago	1.0s	1_SVM_glau...	0.775	0.395		0.571		0.444	0.666
	7 minutes ago	1.1s	1_SVM_glau...	0.845	0.631		1		0.476	0.738
	7 minutes ago	1.1s	1_SVM_glau...	0.873	0.564		1		0.438	0.719
	8 minutes ago	1.3s	1_SVM_glau...	0.817	0.499		1		0.316	0.658
	8 minutes ago	1.1s	1_SVM_glau...	0.887	0.659		0.917		0.611	0.796
	8 minutes ago	1.1s	1_SVM_glau...	0.873	0.559		0.818		0.563	0.763
	9 minutes ago	1.1s	1_SVM_glau...	0.817	0.552		0.786		0.524	0.732
	9 minutes ago	1.2s	1_SVM_glau...	0.859	0.615		1		0.474	0.737
	10 minutes ago	1.2s	1_SVM_glau...	0.845	0.703		0.938		0.6	0.789
	12 minutes ago	1.2s	1_SVM_glau...	0.901	0.592		0.889		0.571	0.777

Random Forest

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	43 seconds ago	1.6s	1_rf_glauco...	0.873	0.747		0.9		0.72	0.838
	1 minute ago	1.6s	1_rf_glauco...	0.901	0.675		0.857		0.706	0.834
	1 minute ago	1.8s	1_rf_glauco...	0.901	0.696		0.867		0.722	0.842
	2 minutes ago	1.6s	1_rf_glauco...	0.859	0.649		0.867		0.619	0.79
	2 minutes ago	1.7s	1_rf_glauco...	0.873	0.559		0.818		0.563	0.763
	3 minutes ago	1.6s	1_rf_glauco...	0.845	0.671		0.783		0.75	0.822
	3 minutes ago	1.7s	1_rf_glauco...	0.845	0.659		0.875		0.609	0.784
	3 minutes ago	1.8s	1_rf_glauco...	0.887	0.567		0.714		0.714	0.822
	4 minutes ago	1.6s	1_rf_glauco...	0.859	0.559		0.733		0.647	0.786
	5 minutes ago	1.7s	1_rf_glauco...	0.845	0.592		0.846		0.55	0.755

Iteration 2

LGBM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	bal	precision_score	r2_	recall_score	roc_auc_score
	7 minutes ago	10.0s	2_LGBM_gla...	0.837	0.635	(	0.7	-	0.833	0.836
	8 minutes ago	9.7s	2_LGBM_gla...	0.815	0.604	(	0.68	-	0.791	0.808
	8 minutes ago	10.3s	2_LGBM_gla...	0.807	0.665	(	0.737	-	0.792	0.805
	9 minutes ago	10.4s	2_LGBM_gla...	0.859	0.678	(	0.773	-	0.791	0.841
	9 minutes ago	10.4s	2_LGBM_gla...	0.844	0.645	(	0.723	-	0.81	0.835
	10 minutes ago	10.0s	2_LGBM_gla...	0.793	0.582	(	0.648	-	0.795	0.793
	10 minutes ago	10.0s	2_LGBM_gla...	0.815	0.617	(	0.708	-	0.756	0.8
	11 minutes ago	9.5s	2_LGBM_gla...	0.793	0.562	(	0.64	-	0.762	0.784
	12 minutes ago	9.7s	2_LGBM_gla...	0.844	0.662	(	0.725	-	0.841	0.844
	13 minutes ago	11.1s	2_LGBM_gla...	0.793	0.596	(	0.667	-	0.783	0.79

Logistic Regression

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	bala	precision_score	r2	recall_score	roc_auc_score
	3 minutes ago	1.5s	2_lr_glauco...	0.859	0.63	0..	0.771		0.711	0.814
	3 minutes ago	1.5s	2_lr_glauco...	0.822	0.643	0..	0.767		0.702	0.794
	3 minutes ago	1.4s	2_lr_glauco...	0.852	0.58	0..	0.727		0.686	0.798
	4 minutes ago	1.4s	2_lr_glauco...	0.859	0.685	0..	0.791		0.773	0.837
	4 minutes ago	1.5s	2_lr_glauco...	0.822	0.643	0..	0.811		0.638	0.779
	4 minutes ago	1.5s	2_lr_glauco...	0.815	0.678	0..	0.78		0.736	0.801
	4 minutes ago	1.5s	2_lr_glauco...	0.822	0.633	0..	0.775		0.674	0.786
	5 minutes ago	1.4s	2_lr_glauco...	0.867	0.684	0..	0.816		0.738	0.831
	5 minutes ago	1.7s	2_lr_glauco...	0.844	0.714	0..	0.841		0.725	0.821
	6 minutes ago	1.5s	2_lr_glauco...	0.83	0.537	0..	0.71		0.611	0.76

KNN

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	bala	precision_score	r2	recall_score	roc_auc_score
	6 minutes ago	2.0s	2_KNN_glau...	0.807	0.495	0..	0.704		0.514	0.716
	6 minutes ago	1.9s	2_KNN_glau...	0.807	0.629	0..	0.824		0.583	0.757
	7 minutes ago	1.9s	2_KNN_glau...	0.822	0.603	0..	0.771		0.628	0.77
	7 minutes ago	2.0s	2_KNN_glau...	0.83	0.633	0..	0.723		0.773	0.815
	7 minutes ago	1.9s	2_KNN_glau...	0.867	0.658	0..	0.769		0.769	0.838
	7 minutes ago	2.0s	2_KNN_glau...	0.77	0.601	0..	0.833		0.49	0.715
	8 minutes ago	2.0s	2_KNN_glau...	0.852	0.68	0..	0.821		0.711	0.817
	8 minutes ago	1.9s	2_KNN_glau...	0.793	0.534	0..	0.719		0.548	0.725
	9 minutes ago	2.2s	2_KNN_glau...	0.83	0.664	0..	0.805		0.688	0.798
	9 minutes ago	2.0s	2_KNN_glau...	0.852	0.653	0..	0.806		0.69	0.808

SVM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	bala	precision_score	r2	recall_score	roc_auc_score
	2 minutes ago	1.2s	2_SVM_glau...	0.837	0.652	0..	0.829		0.644	0.789
	3 minutes ago	1.0s	2_SVM_glau...	0.785	0.567	0..	0.758		0.543	0.727
	3 minutes ago	1.2s	2_SVM_glau...	0.815	0.61	0..	0.778		0.622	0.767
	3 minutes ago	1.1s	2_SVM_glau...	0.815	0.575	0..	0.793		0.548	0.742
	3 minutes ago	1.1s	2_SVM_glau...	0.785	0.556	0..	0.75		0.533	0.722
	4 minutes ago	1.2s	2_SVM_glau...	0.8	0.537	0..	0.719		0.561	0.733
	4 minutes ago	1.1s	2_SVM_glau...	0.822	0.58	0..	0.758		0.61	0.762
	4 minutes ago	1.2s	2_SVM_glau...	0.807	0.607	0..	0.778		0.609	0.759
	4 minutes ago	1.2s	2_SVM_glau...	0.852	0.657	0..	0.893		0.595	0.781
	5 minutes ago	1.1s	2_SVM_glau...	0.778	0.608	0..	0.769		0.588	0.741

Random Forest

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	bala	precision_score	r2	recall_score	roc_auc_score
	2 minutes ago	1.6s	2_rf_glauco...	0.904	0.793	0..	0.902		0.804	0.88
	2 minutes ago	1.6s	2_rf_glauco...	0.844	0.659	0..	0.745		0.795	0.832
	3 minutes ago	1.6s	2_rf_glauco...	0.859	0.66	0..	0.775		0.756	0.83
	3 minutes ago	1.8s	2_rf_glauco...	0.852	0.778	0..	0.95		0.679	0.827
	3 minutes ago	1.7s	2_rf_glauco...	0.874	0.736	0..	0.938		0.667	0.822
	3 minutes ago	1.7s	2_rf_glauco...	0.919	0.777	0..	0.909		0.789	0.879
	4 minutes ago	1.8s	2_rf_glauco...	0.874	0.714	0..	0.814		0.795	0.854
	4 minutes ago	1.9s	2_rf_glauco...	0.859	0.649	0..	0.8		0.7	0.813
	4 minutes ago	1.7s	2_rf_glauco...	0.844	0.668	0..	0.853		0.644	0.794
	4 minutes ago	1.8s	2_rf_glauco...	0.896	0.726	0..	0.857		0.769	0.859

Iteration 3

LGBM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	13 minutes ago	10.7s	3_LGBM_gla...	0.822	0.643	(	0.795	·	0.66	0.784
	14 minutes ago	10.7s	3_LGBM_gla...	0.844	0.658	(	0.756	·	0.773	0.826
	14 minutes ago	11.0s	3_LGBM_gla...	0.874	0.733	(	0.8	·	0.851	0.869
	15 minutes ago	10.1s	3_LGBM_gla...	0.778	0.585	(	0.649	·	0.787	0.78
	15 minutes ago	11.2s	3_LGBM_gla...	0.837	0.707	(	0.763	·	0.849	0.839
	15 minutes ago	9.4s	3_LGBM_gla...	0.793	0.564	(	0.7	·	0.636	0.752
	16 minutes ago	10.6s	3_LGBM_gla...	0.874	0.745	(	0.82	·	0.837	0.866
	16 minutes ago	9.6s	3_LGBM_gla...	0.83	0.672	(	0.795	·	0.714	0.805
	17 minutes ago	10.4s	3_LGBM_gla...	0.859	0.739	(	0.824	·	0.808	0.85
	17 minutes ago	9.0s	3_LGBM_gla...	0.837	0.669	(	0.745	·	0.809	0.83

SVM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2_	recall_score	roc_auc_score
	2 minutes ago	1.1s	3_SVM_glau...	0.741	0.59	(	0.828	·	0.444	0.691
	2 minutes ago	1.1s	3_SVM_glau...	0.83	0.621	(	0.885	·	0.535	0.751
	2 minutes ago	1.2s	3_SVM_glau...	0.807	0.588	(	1	·	0.395	0.698
	2 minutes ago	1.2s	3_SVM_glau...	0.807	0.546	(	0.8	·	0.488	0.717
	3 minutes ago	1.1s	3_SVM_glau...	0.793	0.639	(	0.871	·	0.529	0.741
	3 minutes ago	1.2s	3_SVM_glau...	0.77	0.625	(	0.867	·	0.491	0.721
	3 minutes ago	1.1s	3_SVM_glau...	0.844	0.649	(	0.957	·	0.524	0.757
	4 minutes ago	1.1s	3_SVM_glau...	0.785	0.58	(	0.821	·	0.489	0.716
	4 minutes ago	1.1s	3_SVM_glau...	0.793	0.601	(	0.913	·	0.447	0.712
	4 minutes ago	1.2s	3_SVM_glau...	0.778	0.56	(	0.9	·	0.391	0.684

Random Forest

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2_	recall_score	roc_auc_score
	2 minutes ago	1.6s	3_rf_glauco...	0.889	0.771	(	0.9	(	0.766	0.86
	2 minutes ago	1.7s	3_rf_glauco...	0.844	0.668	(	0.853	(	0.644	0.794
	2 minutes ago	1.8s	3_rf_glauco...	0.807	0.56	(	0.786	(	0.524	0.73
	3 minutes ago	1.6s	3_rf_glauco...	0.867	0.709	(	0.846	(	0.733	0.833
	3 minutes ago	1.7s	3_rf_glauco...	0.83	0.658	(	0.833	(	0.638	0.785
	3 minutes ago	2.0s	3_rf_glauco...	0.867	0.709	(	0.846	(	0.733	0.833
	3 minutes ago	1.6s	3_rf_glauco...	0.837	0.673	(	0.857	(	0.638	0.791
	4 minutes ago	1.5s	3_rf_glauco...	0.837	0.61	(	0.828	(	0.585	0.766
	4 minutes ago	1.6s	3_rf_glauco...	0.896	0.778	(	0.9	(	0.783	0.869
	6 minutes ago	1.7s	3_rf_glauco...	0.859	0.65	(	0.862	(	0.625	0.791

Iteration 4

LGBM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2_	recall_score	roc_auc_score
	5 minutes ago	10.4s	4_LGBM_gla...	0.822	0.592	(	0.698	·	0.732	0.797
	6 minutes ago	9.7s	4_LGBM_gla...	0.837	0.653	(	0.745	·	0.778	0.822
	6 minutes ago	9.2s	4_LGBM_gla...	0.763	0.491	(	0.639	·	0.548	0.704
	7 minutes ago	9.8s	4_LGBM_gla...	0.822	0.652	(	0.74	·	0.771	0.811
	7 minutes ago	10.0s	4_LGBM_gla...	0.785	0.503	(	0.605	·	0.684	0.754
	7 minutes ago	9.5s	4_LGBM_gla...	0.8	0.602	(	0.694	·	0.739	0.785
	8 minutes ago	8.9s	4_LGBM_gla...	0.8	0.581	(	0.66	·	0.767	0.791
	8 minutes ago	9.7s	4_LGBM_gla...	0.844	0.725	(	0.789	·	0.833	0.843
	9 minutes ago	10.2s	4_LGBM_gla...	0.822	0.657	(	0.7	·	0.875	0.834
	9 minutes ago	9.6s	4_LGBM_gla...	0.807	0.609	(	0.738	·	0.674	0.775

SVM

				Metrics						
	Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	50 seconds ago	1.0s	4_SVM_glau...	0.778	0.595		0.88		0.449	0.707
	1 minute ago	1.0s	4_SVM_glau...	0.8	0.58		0.947		0.409	0.699
	1 minute ago	1.0s	4_SVM_glau...	0.844	0.641		0.889		0.571	0.77
	1 minute ago	1.0s	4_SVM_glau...	0.837	0.64		0.92		0.535	0.757
	1 minute ago	1.1s	4_SVM_glau...	0.8	0.716		0.919		0.586	0.774
	1 minute ago	1.0s	4_SVM_glau...	0.83	0.621		0.885		0.535	0.751
	2 minutes ago	1.0s	4_SVM_glau...	0.8	0.659		0.929		0.51	0.743
	2 minutes ago	1.0s	4_SVM_glau...	0.815	0.549		0.778		0.525	0.731
	2 minutes ago	1.0s	4_SVM_glau...	0.837	0.652		0.923		0.545	0.762
	3 minutes ago	1.1s	4_SVM_glau...	0.837	0.722		0.917		0.635	0.799

Random Forest

				Metrics						
	Start Time	Duration	Run Name	accuracy_score	average_precisic	ba	precision_score	r2	recall_score	roc_auc_score
	1 minute ago	1.7s	4_rf_glauco...	0.837	0.622		0.778		0.667	0.79
	2 minutes ago	1.7s	4_rf_glauco...	0.859	0.701		0.814		0.761	0.835
	2 minutes ago	1.6s	4_rf_glauco...	0.867	0.665		0.824		0.7	0.818
	2 minutes ago	1.6s	4_rf_glauco...	0.874	0.707		0.81		0.791	0.852
	3 minutes ago	1.7s	4_rf_glauco...	0.837	0.736		0.884		0.691	0.814
	3 minutes ago	1.8s	4_rf_glauco...	0.852	0.706		0.909		0.638	0.802
	3 minutes ago	1.6s	4_rf_glauco...	0.859	0.688		0.857		0.682	0.813
	3 minutes ago	1.7s	4_rf_glauco...	0.822	0.607		0.732		0.698	0.789
	4 minutes ago	1.7s	4_rf_glauco...	0.896	0.702		0.789		0.833	0.876
	4 minutes ago	1.7s	4_rf_glauco...	0.896	0.734		0.861		0.775	0.861

Iteration 5

LGBM

				Metrics						
	Start Time	Duration	Run Name	accuracy_score	average_precisic	balanc	precision_score	r2	recall_score	roc_auc_score
	6 minutes ago	9.7s	5_LGBM_gla...	0.852	0.69	0.8...	0.842		0.696	0.814
	7 minutes ago	9.9s	5_LGBM_gla...	0.77	0.586	0.7...	0.621		0.872	0.794
	7 minutes ago	9.6s	5_LGBM_gla...	0.807	0.605	0.81	0.667		0.818	0.81
	7 minutes ago	9.6s	5_LGBM_gla...	0.77	0.582	0.77	0.649		0.771	0.77
	8 minutes ago	10.3s	5_LGBM_gla...	0.8	0.641	0.7...	0.773		0.667	0.774
	9 minutes ago	10.2s	5_LGBM_gla...	0.8	0.611	0.8...	0.645		0.889	0.822
	10 minutes ago	10.3s	5_LGBM_gla...	0.83	0.637	0.8...	0.706		0.818	0.827
	10 minutes ago	9.6s	5_LGBM_gla...	0.748	0.496	0.7...	0.566		0.732	0.744
	11 minutes ago	10.3s	5_LGBM_gla...	0.793	0.577	0.7...	0.636		0.814	0.798
	11 minutes ago	10.4s	5_LGBM_gla...	0.844	0.664	0.8...	0.717		0.864	0.849

SVM

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	balanc	precision_score	r2	recall_score	roc_auc_score
	2 minutes ago	1.3s	5_SVM_glau...	0.852	0.725	0.7...	0.967		0.604	0.796
	2 minutes ago	1.1s	5_SVM_glau...	0.8	0.537	0.7...	0.719		0.561	0.733
	2 minutes ago	1.1s	5_SVM_glau...	0.815	0.638	0.7...	0.893		0.532	0.749
	3 minutes ago	1.1s	5_SVM_glau...	0.852	0.663	0.8...	0.829		0.674	0.805
	3 minutes ago	1.0s	5_SVM_glau...	0.859	0.659	0.81	0.824		0.683	0.81
	3 minutes ago	1.1s	5_SVM_glau...	0.822	0.712	0.79	0.895		0.63	0.79
	4 minutes ago	1.0s	5_SVM_glau...	0.778	0.566	0.7...	0.774		0.511	0.716
	4 minutes ago	1.1s	5_SVM_glau...	0.83	0.568	0.7...	0.808		0.538	0.743
	4 minutes ago	1.0s	5_SVM_glau...	0.8	0.614	0.7...	0.769		0.625	0.761
	1 hour ago	1.1s	5_SVM_glau...	0.867	0.656	0.8	0.862		0.641	0.8

Random Forest

				Metrics						
	↓ Start Time	Duration	Run Name	accuracy_score	average_precisic	balanc	precision_score	r2	recall_score	roc_auc_score
	2 minutes ago	1.8s	5_rf_glauco...	0.859	0.621	0.8...	0.75		0.73	0.819
	2 minutes ago	1.7s	5_rf_glauco...	0.859	0.705	0.82	0.865		0.696	0.82
	2 minutes ago	1.6s	5_rf_glauco...	0.83	0.729	0.81	0.867		0.696	0.81
	2 minutes ago	1.6s	5_rf_glauco...	0.859	0.69	0.8...	0.879		0.659	0.808
	3 minutes ago	1.6s	5_rf_glauco...	0.837	0.656	0.7...	0.871		0.6	0.778
	3 minutes ago	1.8s	5_rf_glauco...	0.822	0.614	0.78	0.763		0.659	0.78
	3 minutes ago	1.8s	5_rf_glauco...	0.83	0.647	0.8...	0.756		0.739	0.808
	5 minutes ago	1.7s	5_rf_glauco...	0.867	0.685	0.8...	0.773		0.81	0.851
	5 minutes ago	1.6s	5_rf_glauco...	0.844	0.665	0.8...	0.786		0.733	0.817
	5 minutes ago	1.6s	5_rf_glauco...	0.822	0.662	0.7...	0.821		0.653	0.786

PyCaret

compare_model()

The image below shows for each one of the six iterations, one of the 10 runs performed. These iterations were performed in Pycaret using the function compare_model().

Iteration 1

	Model	Accuracy	AUC	Recall	Prec.	F1
lightgbm	Light Gradient Boosting Machine	0.8629	0.9111	0.7511	0.8236	0.7855
rf	Random Forest Classifier	0.8586	0.9210	0.7702	0.8010	0.7846
knn	K Neighbors Classifier	0.7772	0.8818	0.8532	0.6239	0.7202
lr	Logistic Regression	0.7772	0.8452	0.7000	0.6595	0.6778
svm	SVM - Linear Kernel	0.7472	0.0000	0.6489	0.6189	0.6304

Iteration 2

	Model	Accuracy	AUC	Recall	Prec.	F1
lightgbm	Light Gradient Boosting Machine	0.8437	0.8843	0.7126	0.8030	0.7513
rf	Random Forest Classifier	0.8287	0.9059	0.7189	0.7610	0.7353
svm	SVM - Linear Kernel	0.7923	0.0000	0.7444	0.6742	0.7045
knn	K Neighbors Classifier	0.7558	0.8436	0.8148	0.6007	0.6910
lr	Logistic Regression	0.7666	0.8533	0.7252	0.6339	0.6716

Iteration 3

	Model	Accuracy	AUC	Recall	Prec.	F1
rf	Random Forest Classifier	0.8501	0.8977	0.7390	0.8010	0.7684
lightgbm	Light Gradient Boosting Machine	0.8394	0.8707	0.7519	0.7660	0.7586
svm	SVM - Linear Kernel	0.7580	0.0000	0.7523	0.6404	0.6787

Iteration 4

	Model	Accuracy	AUC	Recall	Prec.	F1
rf	Random Forest Classifier	0.8630	0.9219	0.7835	0.8120	0.7942
lightgbm	Light Gradient Boosting Machine	0.8501	0.8979	0.7964	0.7716	0.7821
svm	SVM - Linear Kernel	0.7988	0.0000	0.7967	0.6725	0.7250

Iteration 5

	Model	Accuracy	AUC	Recall	Prec.	F1
rf	Random Forest Classifier	0.8245	0.8997	0.7265	0.7451	0.7349
lightgbm	Light Gradient Boosting Machine	0.8053	0.8780	0.6948	0.7184	0.7050
svm	SVM - Linear Kernel	0.7794	0.0000	0.6821	0.6719	0.6731

Iteration 6

	Model	Accuracy	AUC	Recall	Prec.	F1
rf	Random Forest Classifier	0.8372	0.9029	0.7897	0.7467	0.7650
lightgbm	Light Gradient Boosting Machine	0.8115	0.8703	0.7132	0.7246	0.7151
svm	SVM - Linear Kernel	0.7538	0.0000	0.7264	0.6215	0.6636

tune_model()

The images below show the results after tuning the model using the function tune_model().

Iteration 1

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8526	0.9136	0.8269	0.7544	0.7890
1	0.8654	0.9240	0.7925	0.8077	0.8000
2	0.8645	0.8895	0.7500	0.8298	0.7879
Mean	0.8608	0.9090	0.7898	0.7973	0.7923
Std	0.0058	0.0145	0.0315	0.0316	0.0055

Iteration 2

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8654	0.8943	0.8269	0.7818	0.8037
1	0.8718	0.9063	0.8113	0.8113	0.8113
2	0.8710	0.9080	0.7692	0.8333	0.8000
Mean	0.8694	0.9029	0.8025	0.8088	0.8050
Std	0.0028	0.0061	0.0244	0.0211	0.0047

Iteration 3

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8526	0.8941	0.8491	0.7500	0.7965
1	0.8590	0.8680	0.7500	0.8125	0.7800
2	0.8323	0.9027	0.9231	0.6857	0.7869
Mean	0.8479	0.8883	0.8407	0.7494	0.7878
Std	0.0114	0.0148	0.0709	0.0518	0.0067

Iteration 4

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8141	0.8835	0.7358	0.7222	0.7290
1	0.8205	0.9231	0.8269	0.6935	0.7544
2	0.8774	0.9298	0.7692	0.8511	0.8081
Mean	0.8373	0.9121	0.7773	0.7556	0.7638
Std	0.0285	0.0204	0.0376	0.0685	0.0330

Iteration 5

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8654	0.9298	0.7736	0.8200	0.7961
1	0.8333	0.9035	0.8269	0.7167	0.7679
2	0.8323	0.8906	0.6923	0.7826	0.7347
Mean	0.8437	0.9080	0.7643	0.7731	0.7662
Std	0.0154	0.0163	0.0553	0.0427	0.0251

Iteration 6

Fold	Accuracy	AUC	Recall	Prec.	F1
0	0.8718	0.9328	0.8302	0.8000	0.8148
1	0.8333	0.9210	0.8654	0.7031	0.7759
2	0.8387	0.8672	0.6731	0.8140	0.7368
Mean	0.8479	0.9070	0.7896	0.7724	0.7758
Std	0.0170	0.0286	0.0836	0.0493	0.0318