DEGREE PROJECT IN INFORMATION AND COMMUNICATION TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

A Machine Learning Approach to the analysis of mortality in patients with cardiovascular diseases

JUAN MIGUEL ALDAMIZ ORCAJO

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

Cardiovascular diseases (CVDs) are the main cause of mortality worldwide, counting for a third of world demises. Consequently, early detection and underlying factors of these pathologies can play a critical role in successful treatments. Many researchers have applied machine learning (ML) for mortality risk estimation in CVDs. However, this is difficult due to their complex and multifactorial nature and the lack of large, unbiased data collections. This thesis holds statistical analysis results and a binary classification model for CVDs mortality prediction based on the ESCARVAL-RISK study, a large cohort study (54,678 patients) running from January 2008 until December 2012. This study faces highly imbalanced classes that may lead to classification models with low specificity and sensitivity. This work proposes several ways to balance classes, including hyperparameter optimization and sample techniques tested over 15 different classification algorithms to overcome the problem. While the specificity is low, the proposed approach using SHapley Additive exPlanations (SHAP) identifies factors that may be optimal targets for intensified preventive interventions.

Keywords

machine learning; supervised learning; graphical models; mortality analysis; cardiovascular diseases.

Sammanfattning

Kardiovaskulära sjukdomar är den främsta dödsorsaken i världen och står för en tredjedel av alla dödsfall i världen. Därför kan tidig upptäckt och underliggande faktorer för dessa sjukdomar spela en avgörande roll för framgångsrika behandlingar. Många forskare har tillämpat maskininlärning (ML) för uppskattning av dödlighetsrisker vid hjärt- och kärlsjukdomar. Detta är dock svårt på grund av deras komplexa och multifaktoriella natur och bristen på stora, opartiska datainsamlingar. Denna avhandling innehåller statistiska analysresultat och en binär klassificeringsmodell för att förutsäga dödligheten i hjärt- och kärlsjukdomar baserat på ESCARVAL-RISK-studien, en stor kohortstudie (54 678 patienter) som pågick från januari 2008 till december 2012. I studien finns mycket obalanserade klasser som kan leda till klassificeringsmodeller med låg specificitet och känslighet. I detta arbete föreslås flera sätt att balansera klasserna, inklusive optimering av hyperparametrar och provtagningstekniker som testats över 15 olika klassificeringsalgoritmer för att lösa problemet. Även om specificiteten är låg identifierar den föreslagna metoden med hjälp av SHapley Additive exPlanations(SHAP) faktorer som kan vara optimala mål för intensifierade förebyggande insatser.

Nyckelord

dödlighetsanalys; hjärt-kärlsjukdomar; maskininlärning; funktionens betydelse; grafiska modeller.

Acknowledgments

I want to thank Professor Emilio Soria Olivas for supervising this project and playing the role of Virgil, showing me all the circles of the machine learning infra-world. Thanks must also go to Hao Hu for his support and advice. Lastly, I want to thank my family for their constant support throughout the whole process.

Stockholm, September 2021
Juan Aldamiz

Contents

List of acronyms and abbreviations

BMI body mass index

DBP diastolic blood preassure

DLP dyslipidemia

DM diabetes mellitus type 2

HDL high-density lipoproteins

HTA hypertention

LDL low-density lipoproteins

SBP systolic blood preassure

TC Total Cholesterol

$TreDLP$ dyslipidemia treatment

$TreDM$ diabetes mellitus type 2 treatment

$TreHTA$ hypertention treatment

age age

$gender$ gender

$hypchol$ hypercholesterolemia

$mortality$ mortality

$obesity$ obesity

$smoking$ tobacco consumption

AdaBoost Adaptive Boost Classifier

AUC Area Under the Curve

CA cardiac arrhythmia

CAD coronary artery Disease

CatBoost CatBoost Classifier

CVD cardiovascular disease

DT Decision Tree Classifier

EDA exploratory data analysis

ET Extra Trees Classifier

false negative A false negative is an outcome where the model incorrectly predicts the negative class.

false positive A false positive is an outcome where the model incorrectly predicts the positive class.

GBC Gradient Boosting Classifier

HF heart failure

IDAL Intelligent Data Analysis Laboratory

KNN K-Nearest Neighbors Classifier

LDA Linear Discriminant Analysis

LightGBM Light Gradient Boosting Machine Classifier

LIME Local Interpretable Model Agnostic Explanations

LR Logistic Regression

ML machine learning

NB Naive Bayes

PCA principal component analysis

PD prediction stroke

QDA Quadratic Discriminant Analysis

RF Random Forest Classifier

Ridge Ridge Classifier

SHAP SHapley Additive exPlanations

SMOTE Synthetic Minority Oversampling Technique

SVM support vector machine

true negative A true negative is an outcome where the model correctly predicts the negative class.

true positive A true positive is an outcome where the model correctly predicts the positive class.

WHO World Health Organization

XGBoost Extreme Gradient Boosting

Chapter 1

Introduction

This research project builds a risk-based prediction system for mortality, providing a practical example of ML techniques and methods applied to health care studies. Accurate prediction of CVDs risk can play an important role in implementing more effective preventive strategies identifying susceptible individuals of CVDs. Additionally, it proposes interpretability machine learning libraries to understand how risk factors influence model predictions. Challenges motivating this research are listed in Section 1.1, pursued objectives are in Section 1.2, benefits and ethics applied are then enumerated in Section 1.3, current existing research in ML and CVDs is presented as background for this project in Section 1.4, an introduction of the proposed research methodology in Section 1.5 and the chapter concludes with an outline of the thesis structure in Section 1.6.

1.1 Problem

According to the World Health Organization (WHO), CVDs are the first cause of mortality worldwide[1], accounting for 31% of deaths globally in 2016, 85% of them by heart attack or stroke. In addition, CVDs cause 37% of premature death of people under 70. A multifactorial etiology characterizes cardiovascular epidemiology as a risk factor associated with hypertension, tobacco consumption, hyperglycemia closely related and mutually potentiated [2]. The relative risk of each condition is not well established yet. The WHO urges early detection and management of people with CVDs. However, the large volume of the vulnerable population makes the execution of effective preventive strategies difficult. Therefore, reliable tools to identify individuals with high risk may have a significant impact on preventive strategies.

1.2 Purpose and Goals

This research develops a statistical learning model for predicting mortality of CVDs using clinical data, collected by a prospective cohort study of 54,978 hospitalized patients, between 20 and 97 years old during 2008 and 2012 provided by the Intelligent Data Analysis Laboratory (IDAL). It is also intended to gain relevant insights into which factors (personal or treatment variables) are good predictors for CVDs deaths in patients. Based on a multivariate analysis of the personal, metabolic, cardiovascular, CVDs diagnostics and treatments predictors. It seeks to identify relationships, dependencies, and influence between them and CVDs mortality. One challenge in leveraging a predictive system of these characteristics is to treat the imbalanced nature of mortality which accounts for 1.7% of the cases. In this research, several resampling techniques are applied and tested to overcome class imbalances. The two following goals are considered and identified in the content of this thesis:

1. Create a binary classification model for mortality prediction in five years.
2. Obtain insights on how the features relate and impact mortality.

1.3 Benefit and Ethics

This work uses valuable clinical data collected by the ESCARVAL-RISK project and generously provided by the IDAL at the University of Valencia to promote advanced ML tasks in medical research [3]. The study sample decreased from 72,673 adults to 54,678 participants after excluding patients with a history of a previous CVD event, myocardial infarction, angina, stroke, or transitory ischemic attack. Also, patients who were already participating in clinical trials or missing variables of interest were excluded. The research results of this work are public and openly distributed, but not the facilitated study data to comply with confidentiality agreements subject to it. All patients were informed and agreed to be part of the study.

1.4 Background

The research on ML methods is growing at an extremely rapid pace over the last 10 years, and their applications expand through a plethora of different areas, being Medicine, not an exception. Figure 1.1 illustrates the exponential

growth of research publication results matching the search terms ML and Cardiovascular from 2010 to 2020, the last complete year of research publications in PubMed [4]. The increase in ML research applied to CVDs

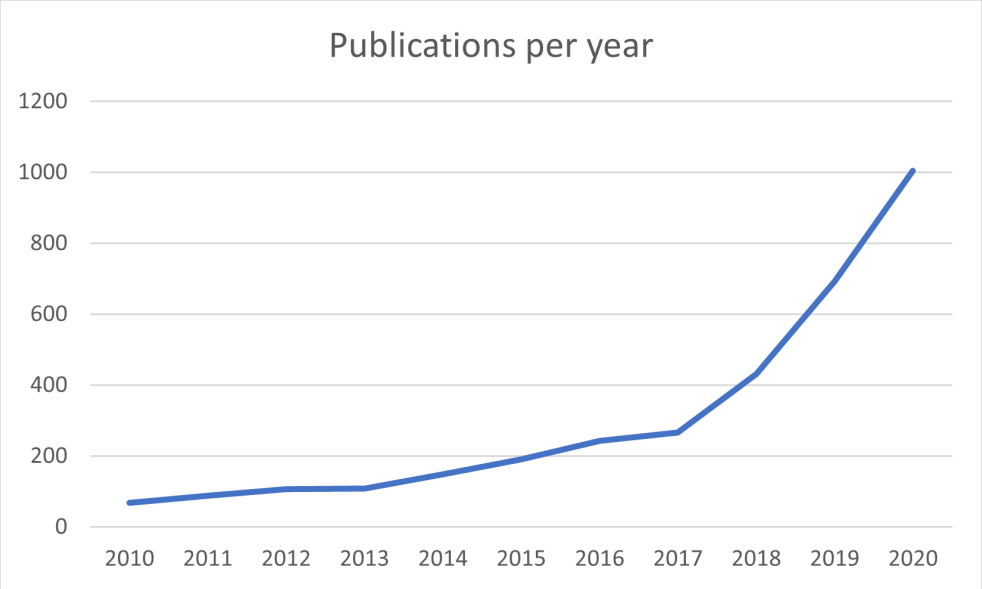


Figure 1.1 – Search results containing the search query; ((machine learning) OR (ML)) AND (Cardiovascular) or (CDV) in PubMed per year from 2010 to 2020. Source PubMed [4]

can be seen as a consequence of its promising predictive capabilities. Table 1.1 lists the main common algorithms and their metrics for two types of cardio pathologies; coronary artery Disease (CAD) and prediction stroke (PD). The other two cardio pathologies, heart failure (HF) and cardiac arrhythmia (CA) do not show results with less than 5 papers, with too few studies per model [5].

Table 1.1 – Area Under the Curve (AUC), Recall (R), Precision (P), with 95% Confidence Interval results of ML models in CVDs per pathology from 55 selected studies. Prediction of HF and CA not included due to too few studies (≤ 5) for each model [5].

Coronary Artery Diseases (CAD)						
Model	**AUC**	**95% CI**	**R**	**95% CI**	**P**	**95% CI**
Boosting	0.88	0.84-0.91	0.86	0.77-0.92	0.70	0.51-0.84
Custom Build	0.93	0.85-0.97	0.87	0.74-0.94	0.86	0.73-0.93
Prediction Stroke (PS)						
Model	**AUC**	**95% CI**	**R**	**95% CI**	**P**	**95% CI**
SVM	0.92	0.81-0.97	0.57	0.26-0.96	0.93	0.71-0.99
Boosting	0.91	0.81-0.96	0.85	0.66-0.94	0.85	0.67-0.94
CNN	0.90	0.83-0.95	0.80	0.70-0.87	0.91	0.77-0.97

This research has its foundations in two papers [6] [7] from IDAL and the Institute for Biomedical Research Hospital Clinic de Valencia (INCLIVA). The papers present research performed in cardiovascular risk patients, proposing four classification models for cardiovascular risk prediction and multiple sampling techniques. However, there are some unique obstacles responsible for the proportionally small footprint of ML in Medicine comparing to other areas. More relevant ones are cited below [8]:

1. Lack of information and reproducibility.
2. Models seen as a black box.
3. Lack of large, unbiased sources of phenotype data with enough information to characterize disease process.

This work addresses the lack of information obstacle by giving a statistical description of the study data, features, ML processes, models, and results, as well as the reasoning behind them. To mitigate the black box effect, it uses SHAP libraries to provide the interpretability of the best performance models.

1.5 Research Methodology

The ML experiments in this work use an adaptation of the supervised ML framework proposed in *Practical Machine Learning for Data Analysis* [9] where the following activities are executed:

Data Collection and Preparation Data collection and preparation were facilitated by the IDAL. Two datasets were provided by the IDAL, an original raw and a preprocessed dataset with a preselection of features based on medical domain knowledge and previous research [7], variable encoding, removal of incomplete records ,and elimination of outliers. Nevertheless, several data processing techniques including several transformers, scalers and data imputations are discussed in Section 2.2 and implemented in Section 3.4.

Exploratory Data Analysis The objective of the exploratory data analysis (EDA) is twofold. On the one hand, data quality is assessed, and potential issues like missing values or class imbalances are identified. On the other hand, it helps to understand the structure and distribution of data. It finds possible relationships between our input variables and output variable using a set of non-parametric statistical tests.

Clustering and Principal Component Analysis A principal component analysis (PCA) is a multivariate statistical technique used to deal with highly correlated variables, helping to detect linear combinations of the independent variables that can best capture the variance in the entire dataset. The components are orthogonal and not correlated with each other. Although the PCA may be useful to reduce the dimensionality of a dataset, its usage makes the interpretability of the model more difficult, conflicting with one of the main goals of this project.

Clustering is an unsupervised learning technique aiming to find a structure in unlabeled data. As the name suggests, it consists of dividing the data points into groups in which data points have similar properties, while data points in different groups should have dissimilar properties. It helps to identify patterns and structures that can improve classification performance. This work follows the hybrid 'cluster and then predict' approach, which combines unsupervised learning and then performing supervised learning classification methods [10]. PCA and Clustering are further presented in Sections 2.5 and 2.6

Classification Applying a holdout sample, a unique common test dataset is used across all experiments. The rest of the data is used as a training dataset. The proposed four classification experiments define two main processes: model selection in Figure 1.2 and model evaluation in Figure 1.3. Model Selection aims to train and optimize fifteen supervised models, and Model Evaluation seeks to evaluate the previous process and analyze the best-performing one. A total of four experiments have been designed to test different optimization

and sampling techniques. The first one serves as a baseline, and the other three propose undersampling, oversampling, and SMOTE techniques to mitigate the negative performance effects of class imbalance in classification models.

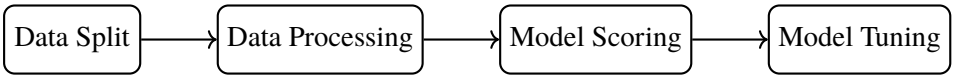


Figure 1.2 – The chronological sequence of the four activities in Model Selection

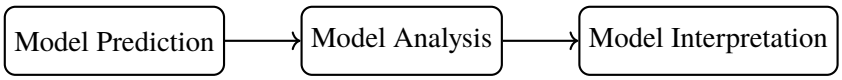


Figure 1.3 – The chronological sequence of the three activities of Model Evaluation.

Extracting valuable insights Out of the EDA and from the selected statistical learning models, SHAP plots will be summarized and contextualized, presenting the research findings and the applicable limitations.

1.6 Structure of the thesis

Chapter 2 describes the dataset and data processing techniques. It presents multivariate techniques, the PCA for dimensionality reduction, and clustering techniques. It also provides the fundamentals of the statistical learning models and ensemble techniques used for binary classification and finishes with the evaluation framework. In Chapter 3, an analysis of the premature mortality due to CVDs is done, starting with an EDA and unsupervised clustering techniques for feature extraction and selection. Then a set of binary classification models are trained and evaluated for the prediction of mortality. Next, the results of the evaluation metrics defined in the previous chapter are displayed on the evaluation framework. Finally, Chapter 4 summarizes results and conclusions, discusses applicable limitations, and outlines future work and reflections.

Chapter 2

Data and Methods

This chapter aims to provide an overview of the data and research methods used in this thesis. Section 2.1 presents statistical data classification and data analysis. Section 2.2 introduces data preparation methods used in statistical learning and modeling. Section 2.4 comments on variable reduction and PCA Section 2.5 discusses the benefits and drawbacks of principal component analysis and introduces its algorithm. Section 2.6 argues the usage of unsupervised clustering techniques for pattern recognition. Section 2.7 describes the main aspects of binary classification and their statistical models. Section 2.8 introduces the main principles of ensemble methods and the related statistical models. Section 2.9 presents the activities to train statistical models and to optimize their hyperparameters. Section 2.10 introduces a set of activities to evaluate and analyze best performing statistical models. Section 2.3 presents the software tools and frameworks in this work.

2.1 Data

The IDAL provided the data used in this research in two CSV files. One of them contains the full tabular data used in the ESCARVAL study [3], with observations from 72,673 patients. From the original dataset, patients with a history of a previous CVD event or missing values in variables of interest were excluded. As a result, a second dataset with 54,678 observations, with 18 variables defining relevant demographic, clinical, and metabolic information, was created. Categorical and numeric variables are described separately in this section. For each variable group, an account of their values and characteristics is given. Therefore Table 2.1 describes the ten categorical variables and Table 2.2 the seven numerical ones.

Statistical data can be classified as categorical or numerical; categorical represents data types that may be divided into groups in which observations can be sorted or categorized, as illustrated in Figure 2.1. It can be further classified into ordinal or nominal, depending on the existence of a ranking or order between the elements of the variable. When it does, it is ordinal and nominal otherwise. Numerical data represent values that can be measured and put into a logical order. It can be classified into discrete, quantified within a finite number and normally represented by an integer, or continues when it can have infinite values and normally represents a float or decimal.

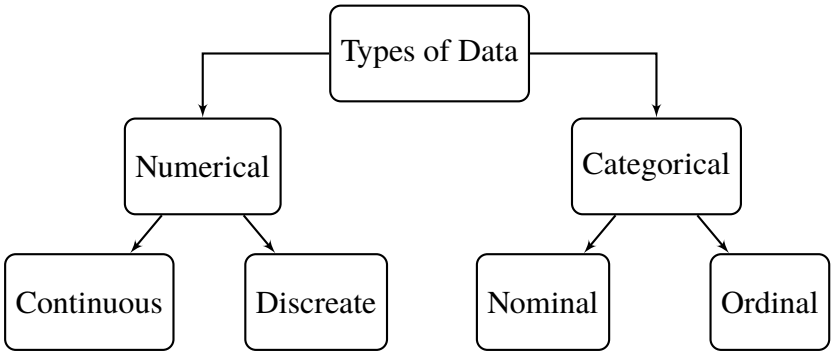


Figure 2.1 – Types of Statistical Data

2.1.1 Categorical

Categorical data provides demographic and clinical information of the patient. Table 2.1 lists a summary of the ten categorical characteristics;

Table 2.1 – Description and statistics of categorical variables

Description (Abbreviation) / Variable	**Values**	**%**
gender ($gender$)	Male = 1	53.14
Sexo	Female = 0	46.86
obesity ($obesity$)	No = 0	58.21
obestiy.cat	Yes = 1	41.79
diabetes mellitus type 2 (DM)	No = 0	64.47
DM.cat	Yes = 1	35.53
diabetes mellitus type 2 treatment ($TreDM$)	No = 0	81.51
TratDM.cat	Yes = 1	18.49
hypertention (HTA)	Yes = 1	71.27
HTA.cat	No = 0	28.73
hypertention treatment ($TreHTA$)	No = 0	58.58
TratHTA.cat	Yes = 1	41.42
hypercholesterolemia ($hypchol$)	Yes = 1	86.44
hypchol.cat	No = 0	13.56
dyslipidemia (DLP)	Yes = 1	75.63
DLP	No = 0	24.37
dyslipidemia treatment ($TreDLP$)	No = 0	71.11
TratDLP.cat	Yes = 1	28.89
tobacco consumption ($smoking$)	Smoker = 0	22.73
Tabaco.cat	Former Smoker= 1	56.71
	Non-Smoker =2	20.55

The variables are mainly Boolean and label encoded with 0/1 to indicate true/false or male/female gender. Smoking is an exception as it has three possible values and therefore is encoded with values 0/1/2. Table 2.1 shows the variable name, description, value range, observations number, and proportion ratio. The number of observations in all the variables is constant and equal to 54,678. All patients in the study show values; hence each variable sums up to 100%.

2.1.2 Numerical

Numerical data provide age, metabolic and clinical information of the patients. Table 2.2 includes the variables: Age (age), body mass index (BMI), Total Cholesterol (TC), high-density lipoproteins (HDL), low-density lipoproteins (LDL), systolic blood preassure (SBP), diastolic blood preassure (DBP).

Moreover, it also summarizes the numerical variables' main metrics listing the maximum and minimum interval, median, first, second and third quartiles, and total observations of each variable. The numbers of observations in the variables are constant and equal to 54,678, which is the study's full population size. There are no missing values. Age is the only discrete variable in the dataset; the other six are continuous.

Table 2.2 – Description and statistics of numerical variables

Description (Abbreviation)	**Mean**	**Std**	**Min**	**25%**	**50%**	**75%**	**Max**
Age age	61.67	13.01	20	53	63	71	97
Body Mass Index (BMI)	29.48	4.89	14	26	29	32	50
Cholesterol Level (TC)	210.09	41.02	68	182	209	236	400
High-Density Lipoprotein (HDL)	52.84	14	20	43	51	61	120
Low-Density Lipoproteins (LDL)	125.95	34.44	20	102	124	148	312
Systolic Blood Preassure (SBP)	135.66	18.29	60	122	135	145	240
Diastolic Blood Preassure (DBP)	79.21	10.86	30	70	80	85	148

2.1.3 Created Features

The three features below are calculated and created based on the combination of other variables;

1. $obesity$
   if BMI >= 30 **then** 1 **else** 0

2. HTA
   if HTA=1, $TreHTA$ =1, SBP > 140, DBP > 90 **then** 1 **else** 0

3. DM
   if DM=1, $TreDM$ = 1, $HemGli.MV$ > 6.5 **then** 1 **else** 0

2.2 Data Preparation

Data preparation and transformation operations for fitting statistical learning models depend on the variable's estimator and distribution characteristics. Several experiments to evaluate the best technique are normally conducted to find the best fit. In this project, data preprocessing transformers are sequentially added in a pipeline to ensure data processing integrity between test and training datasets and reduce the risk of data leaking, which consist of the use of

information in the model training process which would not be expected to be available at prediction time, causing the predictive scores to overestimate the model's utility when run in a production environment [11].

2.2.1 Missing Values

It is common to observe blank or NaN (not a number) values in a dataset for ML tasks. Unfortunately, most of the statistical learning models introduced in Section 3.4 cannot handle missing values. Here we introduce two approaches that can deal with missing values: Impute values and variable dropping. Imputations for numerical features use parameters of the variable distribution, like mean, median, or a constant, normally 0, to replace the missing values. Categorical variables use the constant 'not_available' or the mode value. Observations with NaN or blank values in meaningful variables were excluded from the clean dataset used for this project, as shown in the previous Section 2.1. The mean is applied for numerical values and 'no_available' for categorical.

2.2.2 Categorical Encoding

Categorical encoding transforms categorical variable values into numerical values. Many ML algorithms are unable to operate on categorical or labeled data directly. There are many categorical encoding techniques; this section introduces the three most commonly used ones.

One Hot Encoding Also known as dummy encoding, One Hot Encoding divides each categorical class into several Boolean variables encoded normally with 0 or 1. For categorical variables with N classes, it generates N Boolean variables. Each new variable represents a categorical class, so only one class per observation is flagged, and the others are set to 0. Therefore, not all the columns need to be generated since the entire range of values of a categorical variable can be encoded using $N - 1$ columns. To illustrate the representation of categorical variables in $N - 1$ Boolean, Table 2.3 shows an example of the One Hot Encoding applied to DLP. DLP is a categorical variable containing the values No (Not) when the patient is not diagnosed with the disease and Si (Yes) otherwise. Then two new variables are generated N_{si} and N_{no}. When the patient class is positive, N_{si} is flagged and N_{no} is set to 0. Consequently, the variable N_{no} is complementary of N_{si} and can be removed as most classification models do not provide additional information.

Table 2.3 – Value encoding DLP

DLP	DLP$_{no}$	DLP$_{si}$	Frequency
No	0	1	13,326
Yes	1	0	41,352

Pycaret [12], an auto ML library used in this project, takes the higher proportion Boolean variable of the categorical variable and discards the complementary. Considering the case of DLP, it would only use the new generated DLP$_{no}$ as it has more positive observations than N_{si}.

Ordinal Encoding Transforming the classes of the categorical value coded normally in strings into numerical ones implies an order or ranking between numerical levels. This is a major difference between One Hot and Ordinal Encoding. When applied to binary categorical features, both encoders may return the same values, i.e., column DLP_{no} in Table ,2.3 but the Ordinal Encoding sets a relationship like No < Si or vice versa.

Cardinal Encoding Similar to One Hot Encoding, Cardinal Encoding is used when a categorical variable shows higher cardinality. Instead of creating a feature per category level like One Hot Encoding, which would create noise and increase computational requirements, Cardinal Encoding can use two methods; frequency count based, which replaces the categorical value with the value's frequency; or clustering, which replaces the value with value's cluster labels.

2.2.3 Data Scaling

Normalization and standardization are two common data scalers; the former scales the value of numerical variables typically in the range of [0,1], while the latter results in a Gaussian or approximated normal distribution, normally with mean equal to zero and standard deviation equal to one (unit of variance), this means that 68% of the values will fall in the range of [1,-1] [13].

Max-Min Normalization The scale does not change the shape of the distribution but changes the range of the values. Max-Min Normalization, one of the most common normalization techniques, uses the maximum and minimum values of the distribution. Others like max-abs use the absolute value

instead of the minimum value or robust, which uses the interquartile IQL range. Equation 2.1 is applied to each variate value, where the minimum value of the distribution is subtracted and divided by the distribution range. Where max(x) is the highest distribution value, and min(x) is the lowest value [13].

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)} \tag{2.1}$$

Z-score Standardization The Z Standardization Equation 2.2 is similar to the Max-Min Normalization. The difference is that the media (μ) of the distribution is subtracted in each value and that it uses the standard deviation (σ) as the denominator.

$$x_{stand} = \frac{x - \mu}{\sigma} \tag{2.2}$$

Normalization and Standardization have a positive impact on statistical learning models based on space distance calculation like regression, K-Nearest Neighbors Classifier (KNN), etc. Those classification models are described in Section 2.7. Normalization techniques provide smaller standard deviations than standardization techniques since their values are more concentrated around the mean. On the one hand, standardization balances better the scale of magnitudes of the variables; on the other hand, outliers have a stronger impact in Normalization than on Standardization as data is scaled to smaller intervals. Data scaling depends on the variable characteristics and the method applied.

2.2.4 Transformation

Transformation seeks to stabilize variances and reduces skewness by estimating maximum likelihood. Transformation tends to perform better on models that assume normality or a Gaussian distribution in the residuals like Logistic Regression (LR), Linear Discriminant Analysis (LDA), and Naive Bayes (NB). Those models are described in Section 2.7.

Discretization Also known as Binning, discretization is an opposite transformation to reduce the cardinality of numerical variables into a finite set of range values called bins by the aggregation of numerical range values. Table 2.4 illustrates the transformation of BMI into the categorical variable $obesity$, using asymmetric range values. However, equal size bins are normally a good alternative as well.

Table 2.4 – Cardinality comparison of $obesity$ and BMI

$obesity$	BMI
0	BMI <= 30
1	BMI > 30

2.2.5 Balancing Classes

Conceptually, a dataset for categorical variables is imbalanced when the proportion ratio of one class is far higher than the others. When the minority class is in the output variable, it brings new challenges to the models. Classification problems that look to optimize Accuracy on imbalanced classes often ignore the observation of the minor class, giving an overweight to the major ones. As a result, even if the model may bring high accurate scores, as introduced in the Accuracy paradox, their prediction value may not be as good as expected [14]. To illustrate that problem for our case, the survival ratio is around 98% against less than two percent mortality ratio. This means that a model that completely ignores the death observations and classifies all cases as survival would achieve a 98% Accuracy rate, which is useless to be applied in practice. This work proposes sampling techniques to reduce class proportion differences in the models by giving more relevance to mortality observations.

Undersampling The idea is to reduce the number of observations of the majority classes so that the model will not ignore small classes. The main problem of this technique is the loss of information that is produced, removing majority class observations.

Oversampling Oversampling works opposite to undersampling. Instead of removing observations of the majority class, we increase the number of observations of the lower classes by artificially replicating records of the lower classes. However, this technique does not bring new insights to the model as they are simple repetitions. A particular type of oversampling is the Synthetic Minority Oversampling Technique (SMOTE) which uses a KNN algorithm to create synthetic data. The approach is more effective than simple oversampling because synthetic observations of the smaller classes are closed in the feature space to existing observations from the minority classes [14]. A drawback of SMOTE is that synthetic observations can be ambiguous in case of class overlaps [15].

2.3 Tools Applied

The programming code is based on several Jupyter notebooks. Python is used as the programming language in the Exploratory Data Analysis, the Principal Component Analysis, clustering, creation, and evaluation of the models.

For this purpose, several Python libraries are used. Namely, *Numpy* [16] for numerical computation. *Pandas* [17] for data analysis and transformation, *SciPy* [18] for statistical modeling, *Matplotlib* [19] and *Seaborn* [20] for data visualization, *SciKit-learn* [21] for ML, and *Pycaret* [12] for the automation of pipelines and ML.

2.4 Variable Selection

Feature selection methods aim to select which variables contribute the most to discriminate a target class, suggesting the best features for a model and reducing the number of variables required for statistical modeling. Variable selection improves training convergence and helps to reduce overfitting by improving the general prediction capabilities of the model. The classification system is represented in Figure 2.2. Below is a classification of different selection methods based on their interaction with the classification algorithm.

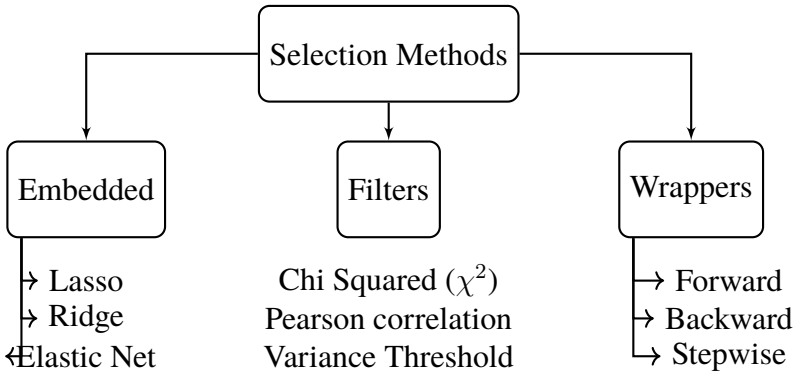


Figure 2.2 – Method Types for Variable selection

- Embedded methods are part of the classification algorithm. The examples are regularization, like Lasso and Ridge, which add a penalty to the cost function to generalize a model. Elastic Net combines both with the α parameter that can be tuned to account for class imbalance.
- Filter methods are model agnostic, meaning that they use a set of statistical tests for the variable selection regarding the target variable.

- Wrappers select a subset of features, fit a classification model and calculate the function error iteratively.

2.5 Principal Component Analysis (PCA)

PCA is a feature extraction technique that aims to reduce the model dimensionality by linearly transforming input variables into new features. The least important features can be eliminated while the most valuable parts remain in the new features. The other two important benefits to consider are that the new features after PCA are independent of each other, and PCA helps to reduce overfitting. One important drawback is that new features are not easy to interpret as there is no one-to-one relationship with the original features. Although interpretability is a decisive factor in this project, PCA will be tested. Based on the results, it will be decided whether it is applied or not to the models.

2.6 Clustering

Clustering is an unsupervised learning technique that aims to find a structure in unlabeled data. By dividing data points into groups in which data points have similar properties, while data points in different groups have dissimilar properties, it helps to identify patterns and structures. The usage of clustering in this work aims to gain insights into the data structure before designing a classifier. Clustering models can be classified on parametric or non-parametric; the former assumes that sample data comes from a population that follows a probability distribution based on a fixed set of parameters, mainly mean and standard deviation. In case it makes an assumption, it is a parametric model; otherwise, a non-parametric model.

2.6.1 K-Means

The 'K' in K-Means stands for the number of clusters. In the first step, the number of clusters 'K' and the 'K' centroids are chosen. The K-Means associates each point to the nearest centroid and calculates a central point of each cluster to reposition the centroids. The data points are then rearranged and associated with the newly defined centroids. These steps are repeated until the centroids stop moving from their positions.

Proximity measures To calculate the number of clusters in K-Means, the concept of inertia is used. Inertia calculates the sum of distances of all points within a cluster from the centroid of that cluster. The lower the inertia value, the better is the clusters. In clustering, the proximity of two data points refers to how similar or different the observations are concerning each other. Two common distance measures between data points are presented in Equations 2.3 and 2.4

$$EuclideanDistance : (x, y), (a, b) = \sqrt{(x - a)^2 + (y - b)^2} \tag{2.3}$$

$$ManhattanDistance : (x, y), (a, b) = |x - a| + |y - b| \tag{2.4}$$

2.6.2 Hierarchical clustering

It is a category of cluster algorithms that build a hierarchy of clusters. Hierarchical clusters normally can be classified as divisive or agglomerative, depending on the approach and starting point. Divisive methods seek to build a hierarchy of clusters following a top-down approach, and all data points belong to one cluster. Further points are separated into new clusters in every iteration until all data points belong to different clusters. Agglomerative methods use a bottom-up approach and work oppositely. In the beginning, all data points belong to a separate cluster, and with each iteration, closer points are joined together until only one cluster remains.

Dendrogram Agglomerative hierarchical clustering can be visualized using a dendrogram. This tree-like diagram records the sequences of merges or splits where distances between two points are expressed through vertical lines.

2.6.3 Density-Based Spatial Clustering Of Application with Noise (DBSCAN)

DBSCAN is a density-based cluster model, and models search for the data space for areas of different densities of data points. They separate different density regions and assign data points within these regions in the same cluster. The two main benefits of DBSCAN models are that they do not need to be told the number of clusters upfront and robust against outliers.

Parameters DBSCAN only requires two parameters: Epsilon and minPoints. To check the density, a circle around each data point is generated. Epsilon defines the radius of the circle. The number of data points needed inside that circle for that data point to be classified as a Core point is defined by minPoints.

2.7 Classification Models

This section enumerates a set of prediction models, techniques, and key theoretical concepts used in statistical learning, which will be applied later in Section 3.4 for binomial classification. There are two types of models: parametric and non-parametric. Parametric models involve a two-step model-based approach, where an assumption of the functional form or shape is made and then a procedure to fit or train the model data. Examples of these models are Ridge Classifier, Logistic Regression, Linear Discriminant Analysis, and Quadratic Discriminant Analysis.

For non-parametric models, there are no explicit assumptions about the functional form of f, avoiding the risk derived from selecting an incorrect functional form that does not fit the data observations. However, the trade-off of not defining a functional form is that the simplification of estimating f for a small number of parameters is no longer possible. Hence, this requires a larger number of observations to obtain an accurate estimate for f. Parametric models like Decision Trees, Naive Bayes, K-Nearest Neighbours, and Support Vector Machines assume an expected function $f(X)$, representing the relationship between the dependent and independent variables. This applies to Equation 2.5 where Y is the vector of responses in the dataset, $f(X)$ is the unknown function, and ε represents the error term produced by the variance of the data. Having the real model expressed mathematically in Equation 2.5, the next step is to find an estimated function $\hat{f}(X)$, see equation 2.6, which is used for expected response prediction.

$$Y = f(X) + \varepsilon \tag{2.5}$$

$$\hat{Y} = \hat{f}(X) \tag{2.6}$$

2.7.1 Logistic Regression

LR is a generalized linear model, and hence a parametric model, where the expected function is assumed to be linear, and the response variable is encoded,

using 0 and 1. To avoid getting responses outside of the [0,1] interval, a Sigmoid function, also known as logistic or logit function 2.7 is applied to the linear regression estimates to convert them into probabilities of classified class [0,1] [22].

$$P(Class|x_1...x_N) = \frac{1}{1 + e^{-(w_0 \sum_{n=1}^{10} w_k \times x_k)}} \tag{2.7}$$

A popular cost function for LR is cross-entropy. Since it is a convex function, the gradient descendent algorithm can find the minimal cost.

2.7.2 Discriminant Analysis (DA)

DA is commonly used for dimensional reduction, similar to the PCA commented in Section 3.2 and the binary or multi-class classification tasks. DA tries to apply allocation rules like maximum likelihood or Bayesian to identify class-specific probability densities and allocate a random variable to each class. Using Bayesian rules, data is classified into one of the output variable classes with the highest likelihood. Considering the likelihood of the discriminant function, the output considers how likely data x falls into each class. However, when data likelihood falls in the decision boundary separating the classes, it is impossible to decide which one it falls into.

Linear Discriminant Analysis LDA assumes that the data comes from multivariate Gaussian distributions, and therefore the parameter distributions are mean (μ) and covariance (Σ). The equal covariance among K classes is derived from the previous assumption obtaining a linear discriminant function (see Equation 2.8). In LDA, the decision boundary between any pair of classes is a linear function.

$$\delta_k(X) = X^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_K + \log \pi_k \tag{2.8}$$

Quadratic Discriminant Analysis Quadratic Discriminant Analysis (QDA) does not apply the assumption of equal covariance. Therefore, the quadratic term in the likelihood cannot be eliminated from the discriminant function. The resulting discriminant function is presented in Equation 2.9. Hence, the decision boundary is no longer a linear but a quadratic function.

$$\delta_k(X) = -\frac{1}{2} \log |\Sigma_K| - \frac{1}{2} (X - \mu_K)^T \Sigma_K^{-1} (X - \mu_K) + \log \pi_k \tag{2.9}$$

2.7.3 Naive Bayes

NB is a specific type of Bayes Networks (BN) that assumes the predictors are conditionally independent regarding the outcome variable. BNs are probabilistic graphical models based on a Directed Acyclic Graph (DAG) representing the relationship structure between variables and the established probabilities between those variables [23]. Bayes theorem calculates the posterior probability P(c|x) of a given class predictor as shown in Equation 2.10,

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{2.10}$$

where:
$P(x|c)$ is the likelihood of the dependent variable given the output class.
$P(c)$ is the class prior probability.
$P(x)$ is the dependent prior probability.

Assuming predictors are class-conditionally independent, NB factorizes their conditional probabilities, simplifying the calculation and analyzing the conditional probability from experimental data as formulated in Equation 2.11 and the inference process from new data [24].

$$P(c|x) = \prod_{i=1}^{n} P(x_i|c) \tag{2.11}$$

2.7.4 K-Nearest Neighbours

KNN is a non-parametric model used for classification or regression. It first calculates a similarity measure function. This is normally a Euclidean or Manhattan distance function for regression or Hamming distance function. Each observation uses the majority vote of their k number of neighbors to assign a class. As it is based on spatial distance, it requires its dependent variables to be scaled. In this thesis, normalization z_score is used for the purpose, when k is the parameter to define the number of neighbors used in the classification vote (see Equation 2.12).

$$\hat{Y} = \sum_{i=1}^{k} |x_i - y_i| \tag{2.12}$$

2.7.5 Decision Trees

Decision Tree Classifier (DT) is also a non-parametric method that can be applied for supervised classification or regression tasks. It creates a hierarchical model in which the root represents the entire population. The data is split through the decision nodes, representing the dependent variables' values into more homogeneous and smaller regions. Information gain based on the split is calculated using measures of distribution dispersion. *SciKit-learn* [21] allows choosing between Gini index and entropy using the 'criterion' parameter.

2.7.6 Support Vector Machine

As one of the most common and popular classifiers, Support vector machine (SVM), which is an extension of the Support Vector Classifier SVC, allows non-linear class boundaries. SVC is also a generalization of the Maximal Margin Classifier [25]. It uses a hyperplane as a decision boundary to divide the observation according to their classes. In a p-dimensional space, a hyperplane is a flat affine subspace of p-1 dimensions. For example, a hyperplane would be a line; in a two-dimension subspace, it would divide the plane into two. Since an infinite number of hyperplanes could exist, the objective is to find the one equidistant to the observed classes, meaning as far from the data points of both classes as possible.

2.8 Ensemble methods

Ensemble methods are an ML technique to generate better models through the aggregation of simple learners, which are estimators that perform slightly better than random chance. Although it is possible to ensemble different types of estimators, DT is the commonly chosen estimator for the following reasons:

1. They are easy to understand and interpret.
2. They are fast and have low computational requirements.
3. No data transformation or scaling is needed since they are non-parametric, and therefore, no specific distribution is assumed.
4. Categorical variables do not need to be encoded as they can handle mixed data types.
5. Multi-collinearity of features does not affect the Accuracy and prediction performance of the model.

Ensemble methods show three major advantages over simple learners:

1. They reduce overfitting since they use many weak learners that underfit (high bias) and combine those predictions into a stronger learner.
2. They reduce the overfitting (variance) and/or bias of the model
3. They are relatively robust against outliers and noise.

The following two families of ensemble methods are usually distinguished:

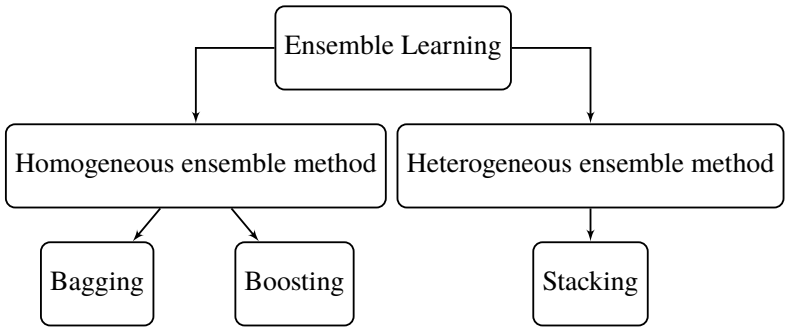


Figure 2.3 – Ensemble Learning

2.8.1 Bagging

Bagging, also known as Bootstrap aggregation, is a general-purpose procedure for reducing the variance of a statistical learning method used in DT. Basically, it uses simple learners in parallel. Bagging uses voting to obtain the aggregation of the output of each independent learner. The most common bagging models are Random Forest Classifier (RF) and Extra Trees Classifier (ET)

2.8.2 Boosting

Boosting is an ensemble technique similar to Bagging. Instead of being trained in parallel, learners are trained sequentially, aiming to reduce the bias due to the dependencies of the simple learners and increasing the weight from accumulated misclassification errors from previous learners. The two main types of boosting models are Gradient Boosting and Adaptive Boost Classifier (AdaBoost); In the former group, the models proposed in this thesis are Light Gradient Boosting Machine Classifier (LightGBM), Extreme Gradient Boosting (XGBoost) and CatBoost Classifier (CatBoost). In the former group is AdaBoost

2.9 Model Selection

In this section, classification models described in the Classification Methods Section 2.7 and Ensemble Methods presented in Section 2.8 are built and optimized as illustrated in Figure 2.4 using the training data, a subset of the full data for training models only. At the same time, the test dataset, which has no overlap with the training data, is also generated during the Data Split and is reserved for model evaluation only.

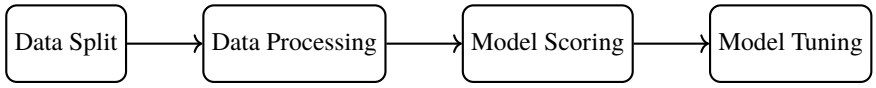


Figure 2.4 – The chronological sequence of the four activities in Model Selection

2.9.1 Data Split

This first part of the process is based on data split, processing, scoring, tuning, prediction, and ranking. The first activity, data split, includes the division of the data in two different sets, training and test, using a 70/30 ratio as shown in Figure 2.5. In addition, it includes the sampling techniques used to balance the training set in all of the experiments, except for the first one, which is used to set the baseline. Different sampling techniques are then applied in the seek for higher F1 metrics.

Figure 2.5 – Data split with a 30% holdout sample for testing

.

In supervised ML problems, models are only useful if they are performing well on different data that are unseen during the training. Therefore a portion of the dataset is reserved for testing using stratification during the data split to maintain the mortality ($mortality$) class ratio in both datasets. It is important to highlight that after splitting, the test dataset is segregated. It is not overlapped with the training dataset to avoid the problem known as data leaking. This segregated data is only used as unseen data for model evaluation.

Table 2.5 – Frequency and data percentage of the original, training, and test datasets.

	Original Set	Train Set	Test Set
Class 0	53,718	37,602	16,116
Class 1	960	672	288
Records	54,678	38,274	16,404
Size	100%	70%	30%

2.9.2 Data Processing

The next activity is data processing, where data techniques discussed in Section 2.2, like data scaling, encoding, and imputation, occurred. Since those transformations are equally applied in all the experiments in this project, they are only commented on in detail on the first experiment.

2.9.3 Model Scoring

In Model Scoring, estimators are trained using four-fold cross-validation based on a random grid search of predefined parameters. Cross-Validation (CV) is a resampling technique; it divides the training set into n number of folds of equal size and then fits the model n number of times, keeping k-1 folds for training and 1 for validation. All data points are used for training purposes after the iterations by making the validation fold different in each iteration, as displayed in Figure 2.6. A factor to consider during CV optimization is the increase in computational resources as the model needs to be training k number of times instead of just one.

2.9.4 Model Tuning

While in model scoring, parameters are tuned based on a random grid search. The optimization techniques are shifted to a Bayesian approach in model tuning, using *Optuna* [26] algorithm, implemented through the *SciKit-learn* [21] framework. Also, it is relevant to mention that the random grid search is optimized for Accuracy during model scoring. This is implemented by default in *Pycaret* [12] and cannot be changed. Nevertheless, the optimization metric shifts to F1 during model tuning, a calculated metric based on Recall and Precision and further explained in Section 2.10, improving the model's score.

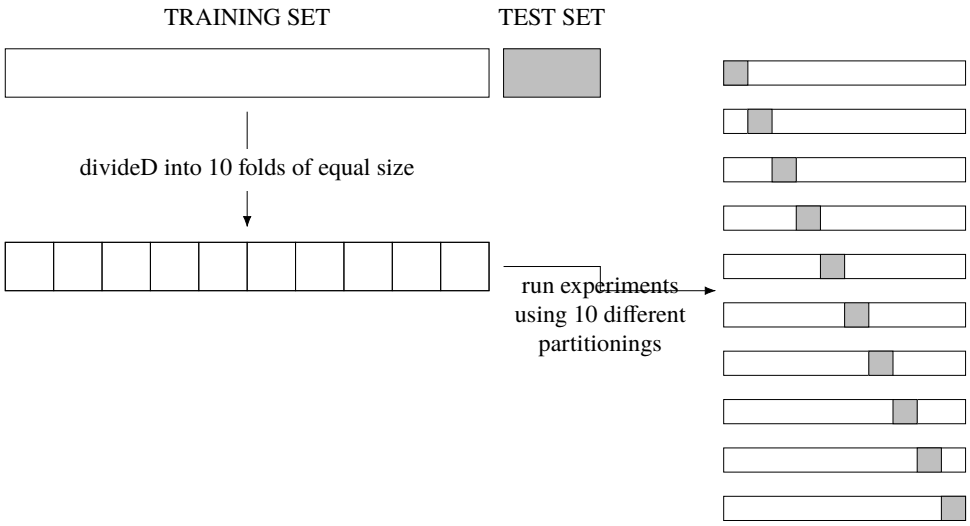


Figure 2.6 – Train/Test Split vs. Cross-Validation.

2.10 Model Evaluation

After training and optimizing the models, the next step is to test the model predictions. Therefore similar steps are repeated but using the test data instead. The test dataset is created in the data split and reserved only for prediction evaluation. In this case, the set contains 30% of the original data. Using the test predictions to create a final rank sorted by F1 scores. Once that, the model with a higher F1 score is produced and identified. The next steps are analysis and evaluation of the selected model, as shown in Figure 2.7.

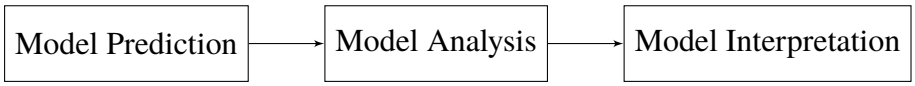


Figure 2.7 – The chronological sequence of the three activities of Model Evaluation.

2.10.1 Model Analysis

Following the most commonly used Model Analysis techniques are introduced, which were applied in this study.

Confusion Matrix

Confusion Matrix is an evaluation measure used for binary or multi-class classification ML problems. The confusion matrix is a two-by-two contingency table where expected predicted classes are represented as columns, and actual classes are represented as rows, as displayed in Table 2.6.

Table 2.6 – Confusion Matrix displaying its four elements in its respective quadrants; True positive, True negative, False positive and False negative

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

The confusion matrix is the main evaluation tool for classification models providing a contingency table. In binary classifications, observations are discriminated into quadrants showing the relationship between real values and predicted ones. Real values are split into true and false values and predicted ones in positive and negative. Consequently, the quadrants take the form of True positive (TP), True negative (TN), False positive (FP), False negative (FN). They are the building blocks for more sophisticated evaluation metrics, which are presented next.

Accuracy

Accuracy represents the correct classified observations out of all classes, the ratio of TP and TN out of all observations as shown in Equation 2.13 (see Equation 2.13).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.13}$$

It is one of the most commonly used metrics in classification problems. However, it is important to highlight that the more imbalance the target variable is, the more unreliable it becomes, as the Accuracy paradox for predictive analytics states that "Predictive Models with a given level of accuracy may have greater predictive power than models with higher Accuracy" [27]. The Accuracy paradox is clearly visible in our classification problem, as survivable observations represent 98.3% of all observations. Therefore a model tends to classify all cases as negative. As a result, it would obtain a similar ratio. However, its reliability would be more than questioned. That is why further evaluation measures are required, i.e., F1, Precision, and Recall.

Precision

Precision represents the ratio of TP, correctly predicted positive cases between all positive predicted cases, TP and FN. In other words, it shows the percentage of correctly predicted observations out of the positive predicted observations (see Equation 2.14).

$$Precision = \frac{TP}{TP + FP} \tag{2.14}$$

Recall

The recall is also known as Sensitivity or True Positive Rate (TPR). It is the ratio of TP between all positive class cases. The percentage of correct predictions out of the positive class is calculated by dividing the number of observations classified as true positive between the number of real, actual values that are positive observations. This measure is used later to plot the Receiver Operating Characteristics (ROC) (see Equation 2.15).

$$Recall = \frac{TP}{TP + FN} \tag{2.15}$$

F1 Score

F1 score is a calculated metric based on the Recall-Precision ratio excluding TN as shown in Equation 2.16. When the output class is imbalanced and the proportion of TN is very high, the sensibility of the other measures can be significantly reduced. F1 is an optimal evaluation measure in unbalancing classes, and a high number of TN, being a good evaluation measure for the analysis of high imbalances classification models' performance. Hence F1 is

the reference measure for classification in this work.

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall} \tag{2.16}$$

False Positive Rate (FPR)

FPR is the percentage of false-positive between total observation of all negative class cases. (see Equation 2.17). Together with the TPR, this metric is used to plot the (ROC) graph.

$$F1 = \frac{FP}{TN + FP} \tag{2.17}$$

Specificity

Specificity is complementary to FPR; it shows the proportion of truly not having the condition and test negative for the condition. (See Equation 2.18).

$$F1 = \frac{TN}{TN + FP} \tag{2.18}$$

Measure Relationships

An increased Threshold, used for discretizing the output of the classification models, produces an increased negative value, which also increases specificity and decreases sensitivity and vice versa for a reduction on the Threshold.

Area Under the Receiver Operating Characteristics (ROC-AUC)

ROC-AUC is a graph that shows the performance of a model in classification problems at various Thresholds, based on two components, receiver operating characteristics (ROC) and area under the curve (AUC). ROC provides the probability curve information, while AUC represents a model's capability to distinguish between the classes. ROC shows the relationship between TPR and FPR at different Threshold levels by representing the former as the Y-axis and the latter as the X-axis.

2.10.2 Model Interpretation

Currently, there are two main interpretability frameworks at the time of writing, namely Local Interpretable Model Agnostic Explanations (LIME) and (SHAP).

LIME is model-agnostic, which means it approximates a black-box model by a simple linear surrogate model locally. It learns from perturbations of the original instance features and how these affect the model scoring identifying the most relevant features. SHAP unifies various model explanation methods, which are mainly model-agnostic or model-specific approximations. It is based on the concept of Shapley Values defined in the game theory [28], in which a Shapley Value is the average contribution of features that are predicting in a different situation. One of its libraries, SHAP TreeExplainer supports interpretability of XGBoost, LightGBM, CatBoost, and DT like models and it is integrated out of the box into *Pycaret* [12]. The selection of the SHAP model over LIME for this study is based on pragmatic development reasons.

In the best performing models compatible with SHAP TreeExplainer, the following three SHAP plots are generated aiming to obtain valuable knowledge.

SHAP Dependence Contribution Plots While SHAP Summary Plots provide an interpretation of the model's feature relevance and their impact, they lack two important characteristics. Firstly, the quantification of the relationship between SHAP values and feature values, and secondly is the dependencies between features obtained in SHAP Dependence Contribution Plots.

SHAP interpretation Reasons Plot These reasons analyze feature impacts based on a particular prediction. The plot categorizes two types of features. One feature increasing the prediction value, marked in pink, and others decreasing the prediction marked in blue. Increasing values include a base value which accounts for the average model output and the particular prediction value, calculated as the difference between total feature sharp values for that particular observation and the base value.

Chapter 3

Results

3.1 Exploratory Data Analysis (EDA)

In this section, an EDA is performed to understand the data shape and potential data issues, like missing values that need to be corrected before further analysis. The analysis is split into Univariate and Bivariate Analysis:

Univariate Analysis By displaying the proportions in categorical groups and central tendency and variability measures for the numerical variables, Univariate Analysis allows understanding the shape of the distribution.

Bivariate Analysis Two categories of statistical tests; Parametric and non-parametric, can be used to assess the dataset variables' independence and dependencies. This project implements the latter as they are distribution-free, hence do not rely on any distribution, assuming [29]:

1. The level of measurement of all variables is nominal or ordinal.
2. The sample sizes of the study groups are unequal.
3. The original data were measured at the same interval or ratio level but violated one of the following assumptions of a parametric test:
   (a) The distribution of the data was seriously skewed or kurtotic (parametric tests assume an approximately normal distribution of the dependent variable). Thus the researcher must use a distribution-free statistic rather than a parametric statistic.
   (b) The data violates the assumptions of equal variance between samples or homoscedasticity.

(c) The continuous data were collapsed into a small number of categories, and thus the data no longer interval or ratio.

3.1.1 Bivariate Analysis

Looking for independence between two categorical variables, e.g., gender and tobacco, the Chi-square test of independence assesses whether there is an association between the variables although no causation inference from the test. Three nonparametric statistical tests have been done to analyze the relationship between the dataset variables.

Mann-Whitney U test

Also known as Wilcoxon Rank Sum Test. The Mann-Whitney U test is a non-parametric test to compare numerical outcomes between two independent groups. It tests whether two numerical variables are derived from the same population. The hypotheses are:

- H_0: The two populations are equal.
- H_1: The two populations are not equal.

After running the Mann-Whitney U test on the numerical variables described in Section 2.1 using $mortality$ as a categorical group, the following variables age, BMI, SBP, DBP, TC, HDL and LDL provided a P< 0.05. Therefore, all the numerical variables fall in the α region. Hence, the null hypothesis can be rejected, and the alternative hypothesis, the two populations are not equal, can be assumed. This means that all of the observed variables are qualified for the discrimination of mortality.

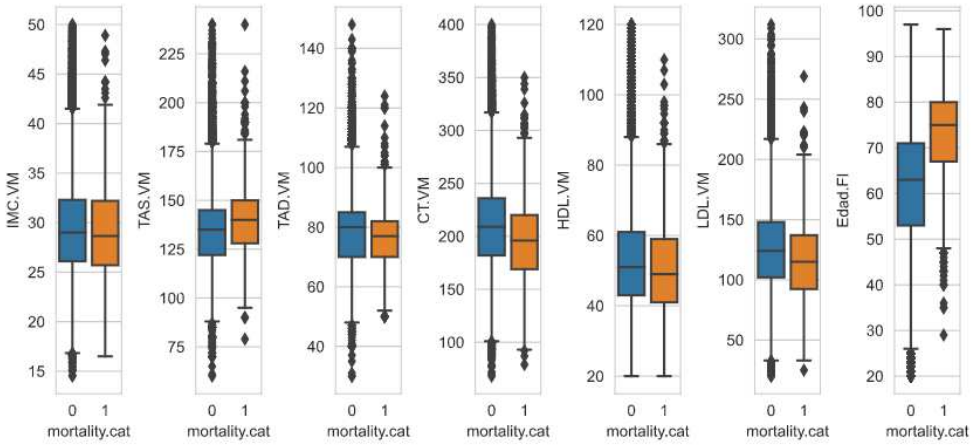


Figure 3.1 – Plot Box Numerical Variables by $mortality$

Figure 3.1 illustrates box graphs of the numeric distributions split by $mortality$. All box graphs show mortality in line with the input variable. Only age shows a significant difference between their classes.

Chi-Squared (χ^2) Test

Pearson's Chi-Squared (χ^2) is a non-parametric test also known as a distribution-free test. It is used in statistics for hypothesis testing to compare two variables in a contingency table to see if they are related. In a more general sense, it tests whether distributions of categorical variables differ from each other. The Chi-squared test is applied twice in this project. The first step is to identify dependable relationships between input variables and $mortality$, and the second is to identify dependencies between the input variables. The hypothesis are:

- H_0: The two variables are independent.
- H_1: The two are not independent.

As shown in Section 2.1, all categorical variables, except $obesity$, have provided a P-Value lower than 0.05, a significant level threshold. Therefore, we can reject the null hypothesis and assume a dependency between the following categorical variable and $mortality$: $gender$, $smoking$, DM, $TreDM$, HTA, $TreHTA$, DLP, $TreDLP$.

The scope of the second Chi-Squared Test is the input categorical variables. Each variable is tested against the rest of the categorical inputs set to identify

dependencies between them. Most of the categorical variables are independent of each other. Out of 45 relationships, there are six where the P-Value is higher than α significance level of 5%. The null hypothesis of the independence relationship is rejected, accepting the alternative hypothesis.

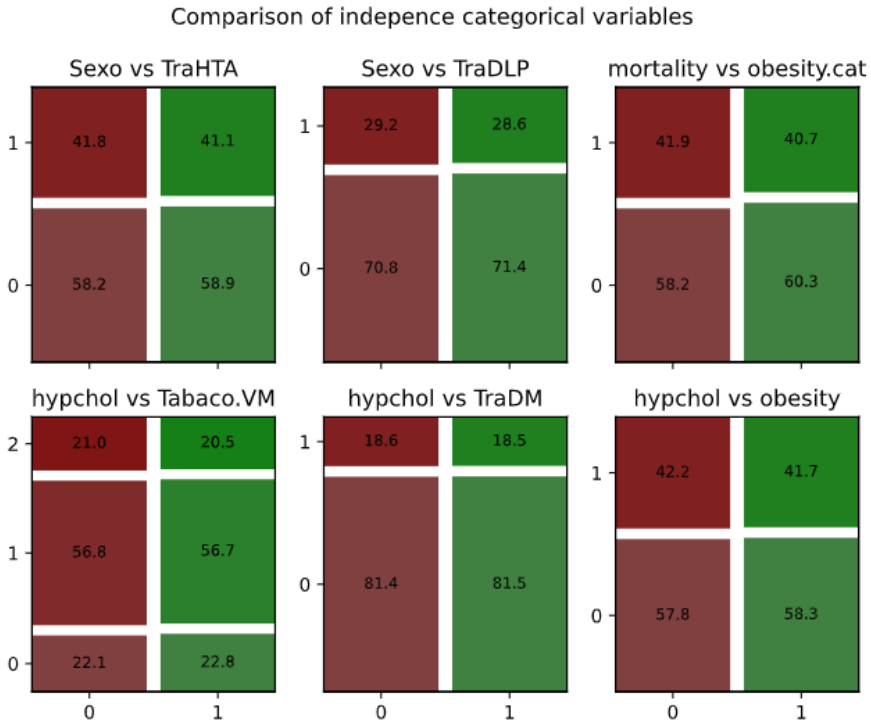
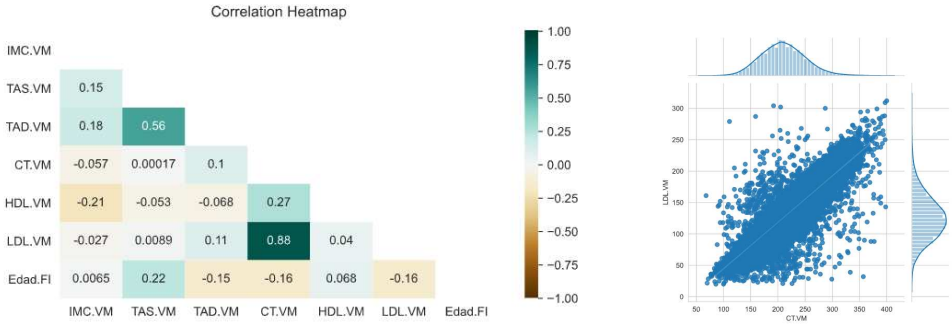


Figure 3.2 – Mosaic diagram displaying a comparison of dependent categorical variables

Only the relationships shown in Figure 3.2 have a dependency. That is applicable for $hypchol$ with $obesity$, $smoking$, and $TreDM$. $gender$ with $TreHTA$ and $TreDLP$ and $obesity$ with $mortality$.

Spearman's rho ρ test

This is the last non-parametric test, and it is used in this thesis to measure the correlations between numerical variables. As shown in Figure 3.3 (a), most variables are only weakly correlated. However, as shown in Figure 3.3 (b), it is not applicable for TC and LDL because a correlation value of 0.88 and R square of 0.77 shows a clear correlation that causes 77% of the total variance.



(a) Heat map showing the Spearman's rho correlations between numerical predictors. Only LDL.VM and CT.VM are strongly correlated

(b) A strong positive linear correlation between LDL and TC

Figure 3.3 – Spearman's rho correlation heatmap

3.2 Principal Component Analysis

The effort to reduce the dimensionality of the dataset by using PCA was not possible since it required to maintain five out of seven PCA to get over 90% of the dataset to explain the variance as listed in Table 3.1 and in Figure 3.4. In other words, the cost of reducing from seven variables to five dimensions is not worth the usage of PCs instead of features, especially when producing meaningful results is a goal.

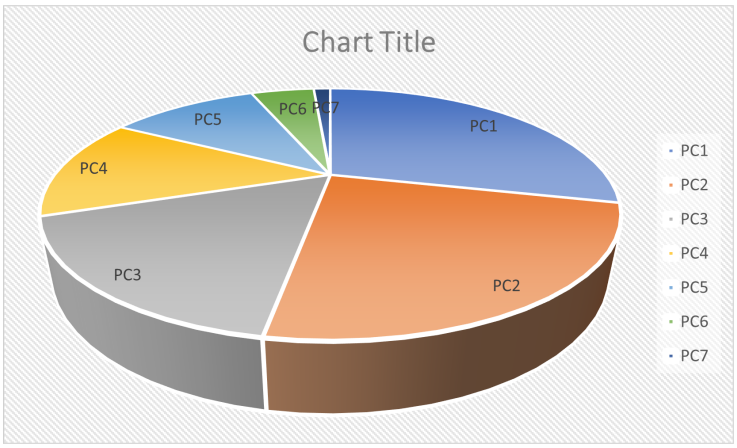


Figure 3.4 – Explained Variance of Principal Components

Table 3.1 – Individual and total explained variance per Principal Component.

PCA	Explained Variance (%)	Accumulative EV(%)
PC1	28.68%	28.68%
PC2	24.02%	52.70%
PC3	16.97%	69.68%
PC4	13.36%	83.04%
PC5	10.96%	93.99%
PC6	4.75%	98.74%
PC7	1.26%	100.00%

3.3 Clustering

Clustering is an unsupervised ML technique where groups of data with similar characteristics are created during the modeling training. They are not based on beforehand defined classes but rather based on the similarity of the observation features. Three different clustering algorithms, K-means, hierarchical agglomeration, and DBSCAN, have been used in this study to group the patients based on the numeric features described in Table 2.2. Once the cluster is created and patients are assigned to them, an analysis of the cluster's mortality ratio is derived from considering whether the cluster adds useful predicting information.

3.3.1 K means

A K means model requires only one parameter, k, the number of clusters for labeling the observations. Therefore, once k is calculated, the model can be trained. To set the optimal value of k, the inertia per number of clusters is calculated as displayed in Figure 3.5. Two techniques can then be used for asserting the value selection of parameter k. The first technique, the elbow technique, facilitates selecting clusters based on the inertia values per cluster. Figure 3.5 selects the number of clusters where the inflection curve is more pronounced.

The second technique looks at how many clusters are required to produce accumulative inertia immediately higher than a preset Threshold, for example, 75%. Using this technique and according to the results display in in Figure 3.5, the minimum number of clusters required to account for at least 75% of the total inertial would be k =5. This means that the model will label all the observations using 5 clusters and hold 76% of the total inertia. In contrast, using k=4, which

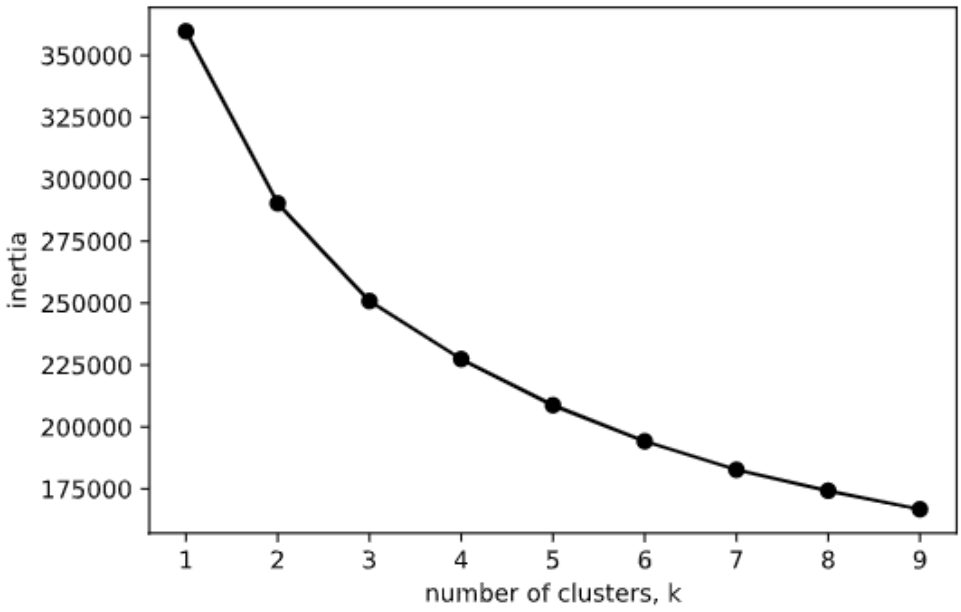


Figure 3.5 – Inertia per number of clusters

is the result of the elbow technique, would retain 68% of the total inertia using 4 clusters. Based on the elbow method, a KNN model with k=4 has been trained to group all observations into one of the four clusters. Table 3.2 shows the *mortality* proportion of each of those clusters. The new aggrupation is discarded without a significant difference in the mortality rate between the suggested clusters as no added value for the analysis is created.

Table 3.2 – Individual and accumulative inertia per number of clusters with K=4

k	**death rate (%)**	**survive rate (%)**
0	1.9	98.1
1	1.2	98.8
2	2.6	97.4
3	1.2	98.8

3.3.2 Hierarchical Clustering

In hierarchical clustering or hierarchical clustering analysis, there are two types of algorithms, agglomerative and divisive. Agglomerative clustering is

chosen for this study because divisive is rarely used in practice. Agglomerative clustering is a bottom-up approach that does not assume beforehand the number of clustering. Instead, the number of clusters is calculated based on the euclidean distance and displayed in dendrogram graphs like in Figure 3.6.

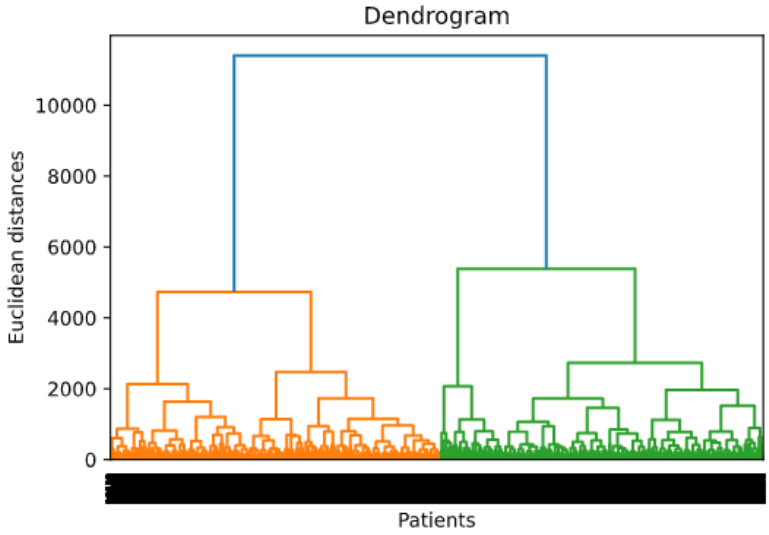


Figure 3.6 – Dendrogram showing cluster's euclidean distances

Hierarchical clusters are more computing and memory-demanding than other clustering algorithms, as KNN or DBSCAN. As a result of this high computational demand using the full dataset of 54k observations, the system runs memory after several hours of computing the model. The highest number of observations, which can be input in this model, is 40k. Hence it is not possible to analyze the mortality of each cluster. Still, it is possible to create its dendrogram (see Figure 3.6), which shows that the optimal number of clusters in this model is two since they cover the biggest segments of patients.

3.3.3 Density-based Spatial Clustering of Applications with Noise (DBSCAN)

Applying DBSCAN in this study and using the elbow method, the results show that Epsilon's optimal value is around 0.9 (see Figure 3.7). For Epsilon, the following values are tested 0.8, 0.9, and 1. For the minimum points, it is using a range of values from 7 to 14. The higher value for the Silhouette Coefficient

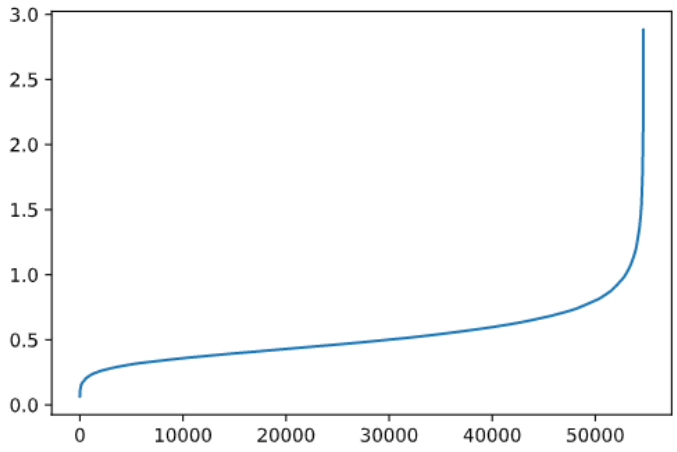


Figure 3.7 – Epsilon and distances between a 4 KNN Clusters in DBSCAN

is 0.24, which is achieved with the combination of 0.8 epsilon and 14 minimum points.

Table 3.3 – Mortality Per Cluster in DBSCAN

k	Live frequency	Death frequency
0	6,954	162
1	46,764	798

After using three different clustering methods, k-means, agglomerative hierarchical, and DBSCAN, the number of clusters ranges from one (with DBSCAN) to four clusters (with the K-means). The Agglomerative hierarchical results are in the middle with two clusters. Thus, clustering does not provide groups of data with a significant difference in the mortality rate.

3.4 Classification

In the first experiment, the model baselines for data processing and model optimization techniques are created. The rest of the experiments build upon this baseline with additional sampling and methods for handling imbalances. Hence the second experiment aims to overcome the negative effects of class imbalance by undersampling the majority class, making the number of observations of this class even with the minority class. The third experiment proposes an opposite approach, augmenting data observations from the minority class so that the

majority and minority classes have the same records. The fourth experiment proposes to use SMOTE to solve the problem of class imbalance.

3.4.1 Baseline Experiment

The Baseline Experiment aims to discover the best performing data transformation, scaling, and optimization techniques.

Baseline Data Split

In this step, the original data is divided into two datasets, 70% for training data to fit the model and tune its hyperparameters and 30% to test and evaluate model predictions. A stratified random sampling strategy ensures that the original 98:2 data split of the mortality class is maintained in the training and test datasets. The class proportions are listed across the datasets in Table 3.4.

Table 3.4 – Frequencies of output variable across the different datasets used in Undersampling Experiment.

mortality	**Full Data**	**Training Data**	**Test Data**
0	53,718	37,602	16,116
1	960	672	288

Baseline Data Processing

Pycaret [12] encodes categorical variables using One Hot Encoding by default, creating a new variable for every class of the input variable. In the case of Boolean variables where there are only two classes, the variable representing the minority class is not needed and is automatically removed, leaving the majority class to represent the feature. Taking the variable *gender* as an example, label encoding gives 0 for males and 1 for females. Still, when the variable is changed to One Hot Encoding, a new variable sexo_0, representing the majority class, is created. The minority class variable is not needed because it is represented as complementary in the majority class variable.

In this experiment, data encoding has a significant impact on the QDA model when *smoking* is set as ordinal instead of categorical, F1 is increasing from 3% to 18% with insignificant changes in other models. Standardization has a positive effect in some models like QDA, improving its F1 around 2%, but

Table 3.5 – Label encoding to One Hot transformation values. Variables with prefix _0 use their complementary values, e.g., in a variable with prefix _0 value, one represents the contrary than before the transformation.

Variable	**Transformation**	**Value**
Gender ($gender$)	Sexo_0	Male=1; Female=0
Diabetes mellitus treat. ($TreDM$)	TraDM.cat_0	No=1; Yes=0
Dyslipidemia (DLP)	DLP_0	No=1; Yes=0

present negative effects in other models like NB. Normalization shows more balance results through the models; hence, this project uses the data scaling method.

Baseline Model Scoring

The steps above add transformer functions to the pipeline and pass them to the statistical models, also known as estimators. Where score metrics, defined in the model evaluation framework in Section 2.10, are collected for each evaluated model. Accuracy, AUC, Recall, Precision, and F1 metrics are calculated and displayed in Table 3.6 for each model, ranking models based on their F1 scores.

Table 3.6 – Model Scoring Results using the training dataset in Baseline Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
QDA	**0.1878**	**0.1994**	0.1783	0.9699	0.8088
NB	0.1308	0.1220	0.1419	0.9713	0.7903
DT	0.0811	0.0952	0.0707	0.9621	0.5364
LightGBM	0.0510	0.0283	0.2738	0.9816	0.8135
Gradient Boosting Classifier (GBC)	0.0309	0.0164	0.3257	0.9820	**0.8361**
XGBoost	0.0300	0.0164	0.1798	0.9814	0.7938
CatBoost	0.0173	0.0089	0.3000	0.9822	0.8234
AdaBoost	0.0169	0.0089	0.1636	0.9823	0.8231
LDA	0.0117	0.0060	**0.5208**	0.9824	0.8284
RF	0.0089	0.0045	0.6250	**0.9825**	0.7867
ET	0.0058	0.0030	0.1667	0.9823	0.7740
KNN	0.0057	0.0030	0.0857	0.9820	0.5805
LR	0.0000	0.0000	0.0000	0.9824	0.8330
SVM	0.0000	0.0000	0.0000	0.9824	0.0000
Ridge Classifier (Ridge)	0.0000	0.0000	0.0000	0.9824	0.0000

Table 3.6 lists QDA as the best scoring model, followed by NB and DT. With the high Accuracy scores, all models show higher than 96% Accuracy and low F1. In some cases, 0% indicates that the model ignores the smaller classes, classifying by default the observations of the majority class and falling into the Accuracy paradox. F1 is focused on correcting this situation and balancing the model, and the model's parameters need to be tuned for this purpose.

Baseline Model Tuning

During model selection, *Pycaret* [12] applies a random search during the cross-validation using default defined parameters from *SciKit-learn* [21] library and Accuracy as driving metric. Model tuning not only allows a high-level optimization of default parameters based on score metrics like, for example, F1, Recall, or AUC, but it also opens up to different optimization frameworks which use different approaches. The Hypertuning parameter technique, based on the random search, is efficient but has a clear limitation as the optimal combination may be untested. Another hyperparameter tuning option is the search grid, which requires a specified subset of the hyperparameter space of the learning algorithm. However, this may require more computation power and

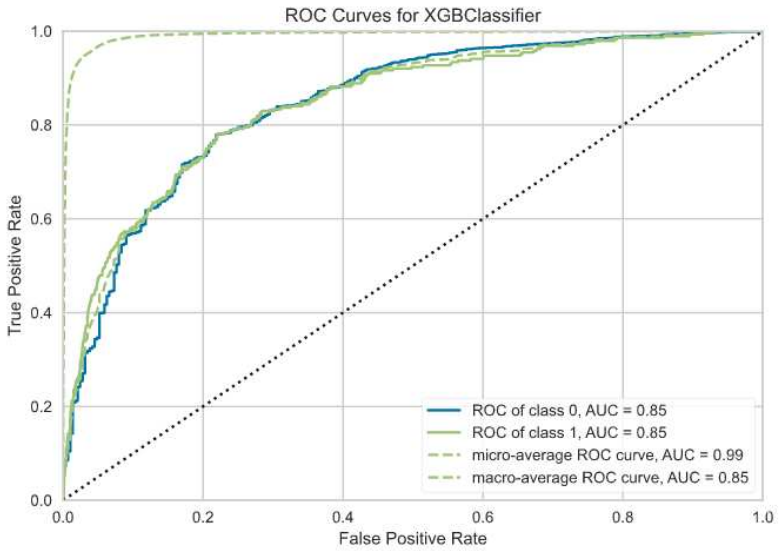
lead to an optimal untested combination of parameters. The one showing better results in our first experiment is *Optuna* [26]. Finally, Bayesian optimization frameworks have been integrated into the recent version of *Pycaret* [12]. After using the Bayesian optimization, the models significantly differ in the first scoring displayed in Table 3.6 because the models were optimized based on F1 scores.

Table 3.7 – Model Tuning Results using the training dataset in the Baseline Experiment * Models with class_weight parameter, set to balance.
** Model Tuning using *Optuna* [26] library was not possible in the CatBoost model due to implementation issues. This model has been optimized using random search from *SciKit-learn* [21] instead.

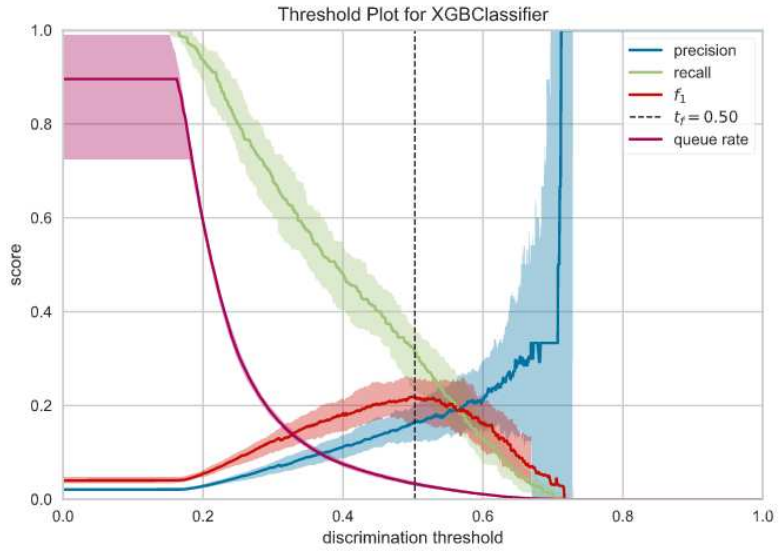
Model	F1	Recall	Prec.	Accuracy	AUC
XGBoost	**0.2163**	0.2440	0.1945	0.9688	**0.8363**
LightGBM*	0.2064	0.3348	0.1492	0.9549	0.8273
QDA	0.1924	0.1890	0.1966	0.9723	0.8114
RF*	0.1851	0.3571	0.1251	0.9448	0.8247
ET*	0.1670	0.4598	0.1022	0.9190	0.8122
NB	0.1321	0.1205	0.1470	0.9719	0.7909
SVM	0.1266	0.6027	0.0722	0.8445	0.0000
DT*	0.1248	0.3854	0.0745	0.9053	0.6515
LR	0.0995	0.7664	0.0532	0.7564	0.8332
Ridge*	0.0987	0.7693	0.0528	0.7531	0.0000
GBC	0.0976	**0.0804**	0.1242	0.9740	0.7113
LDA	0.0880	0.0595	0.1748	0.9783	0.7740
KNN	0.0704	0.0655	0.0762	0.9698	0.5257
AdaBoost	0.0116	0.0060	**0.3083**	**0.9824**	0.8135
CatBoost**	0.0226	0.0119	0.2326	0.9819	0.7852

Baseline Model Analysis

Although XGBoost, LightGBM, and QDA scores are very similar, XGBoost is selected for analysis for three reasons: the model with the highest F1 and a good Precision-Recall ratio is a DT based model enabling SHAP interpretations. Figure 3.8b shows the Precision, Recall, and F1 of the XGBoost model and the 0.5 Thresholds selected on the F1 peak at 23%. The AUC-ROC displayed in Figure 3.8a shows that the Relationship between TPR and FPR is displayed in the ROC with a 0.5 Threshold.



(a) AUC of XGBoost.



(b) Threshold of XGBoost.

Figure 3.8 – Roc and Precision-Recall curves in XGBoost in the first experiment on unseen data.

Baseline Model Predictions

Model selection and model tuning use training data to fit the models and optimize them using four-fold cross-validation. To determine the real efficiency requires unseen data by the models. Therefore, the test dataset, created at the

beginning of the process and reserved for prediction, is used to generate the model scoring listed in Table 3.8.

Table 3.8 – Model Prediction Results using the test dataset in Baseline Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
XGBoost	**0.2328**	0.2639	0.2082	0.9695	0.8408
LightGBM*	0.2144	0.4653	0.1393	0.9401	0.8405
RF*	0.2037	0.4201	0.1344	0.9423	0.8402
QDA	0.1667	0.1597	0.1742	0.9720	0.8043
ET*	0.1667	0.5104	0.0996	0.9104	0.8252
NB	0.1295	0.1181	0.1435	0.9721	0.7972
DT*	0.1288	0.4514	0.0751	0.8928	0.6815
GBC	0.1062	0.0868	0.1366	0.9743	0.7143
KNN	0.0995	**0.7664**	0.0532	0.7563	0.8332
Ridge	0.0984	0.7639	0.0526	0.7543	0.7590
LR	0.0982	0.7535	0.0525	0.7571	**0.8422**
LDA	0.0979	0.0660	0.1900	0.9787	0.7873
SVM	0.0975	0.7083	0.0524	0.7698	0.7396
AdaBoost	0.0270	0.0139	**0.5000**	**0.9824**	0.8220
CatBoost	0.0131	0.0069	0.1111	0.9816	0.8108

XGBoost, LightGBM, and QDA models show a slight increase of one, 2% over-optimized results when using the test dataset scoring from 20 to 23% F1. In the rest of the models, there is no significant difference to the previous results on training data. The score similarity displayed on training and unseen data indicates that models are neither Over nor Underfitting. Another observation is that the Precision-Recall proportion maintenance by the XGBoost maintains almost constant between the two datasets. That does not happen with the LightGBM and QDA, which increases their Recall considerably at the cost of reducing their Precision by getting a trivial F1 improvement. The top three models, XGBoost, LightGBM, and QDA obtain a close F1 score ranking in the range of 20% to 23%, as shown in Table 3.8. However, the similar F1 scores of these models can not be extrapolated to the F1 metric components, Recall and Precision, where the value range grows significantly. When looking at the confusion matrix of XGBoost results illustrated in Figure 3.9, out of 16,404 observations, true negatives cases represent a clear majority group accounting for 15,623 observations. Next are the false positives accounting for 493 cases,

followed by false negatives with 187 observations leaving the true positives with 101 observations.

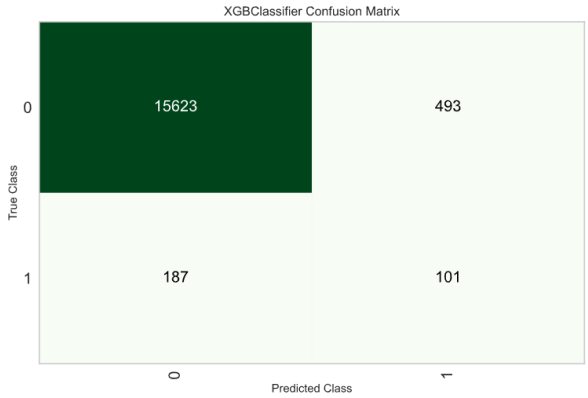


Figure 3.9 – Confusion matrix of XGBoost in the first experiment on unseen data.

Overall, XGBoost is considered the best model for three reasons. It has the highest F1 score with a 23.2% in the test dataset. It has the most balanced Precision-Recall proportion, and it is interpretable by SHARP due to its DT nature.

Baseline Model Interpretation

XGBoost feature importance is analyzed firstly based on the feature importance plot provided by *Pycaret* [12] and secondly based on SHAP plots. Feature importance is demonstrated in Figure 3.10, which shows a ranking of the model's top ten most relevant features. Again, three groups can be distinguished. The first one is made by age and high blood pressure treatment, the second one having high blood pressure, $gender$, DLP, and $TreDM$ and DM, and the third group conformed by TC, BMI and HDL, all of them with the same importance level. It is also interesting to see in this graph that the top five measures account for around 50% of the total importance of the model, and if they increased to the top seven measures, which account for the two first groups of variables, this percentage increase to around 65%.

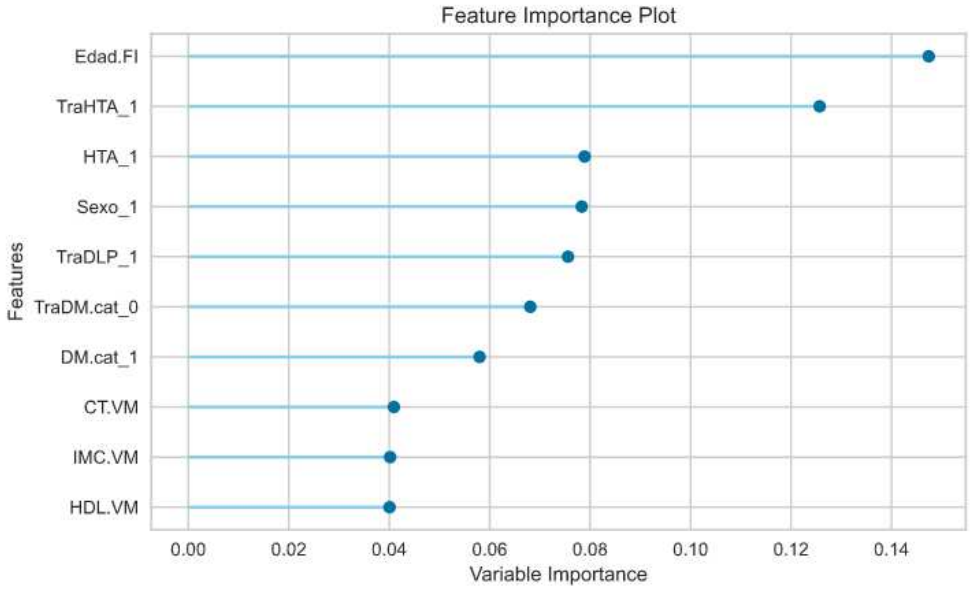


Figure 3.10 – Features of XGBoost in the first experiment on unseen data.

SHAP summary plot Because Figure 3.11, provides more holistic information than Figure 3.10, the SHAP value plot gives the information about overall feature importance in a more granular level by adding the relation between feature value and the SHAP values (impact) on the model. The top four features are share by the feature importance figure compared with the one provided by the model in Figure 3.10, although their position differs. By observing the SHAP summary plot in Figure 3.11, additional information about the features can be gathered. For example, age is the feature with the highest impact on the model, with the major SHAP value ranging from around -0.70 to 1.20, directly correlated with feature values and SHAP values. For categorical variables, values are divided into two clusters based on the category of the feature. For example, disease treatments reduce mortality prediction while CVDs increase it. Other variables like $smoking$, $hypchol$, $obesity$ do not play a significant role in the model. Also, there is no clear correlation between variable values and SHAP values in the case of $smoking$. In Hypertension treatment $TreHTA$, one cluster is slightly longer in range than the other one. For $gender$, clusters are almost identical, and for patients under dyslipidemia treatment $TreDLP$, this difference between clusters is obvious.

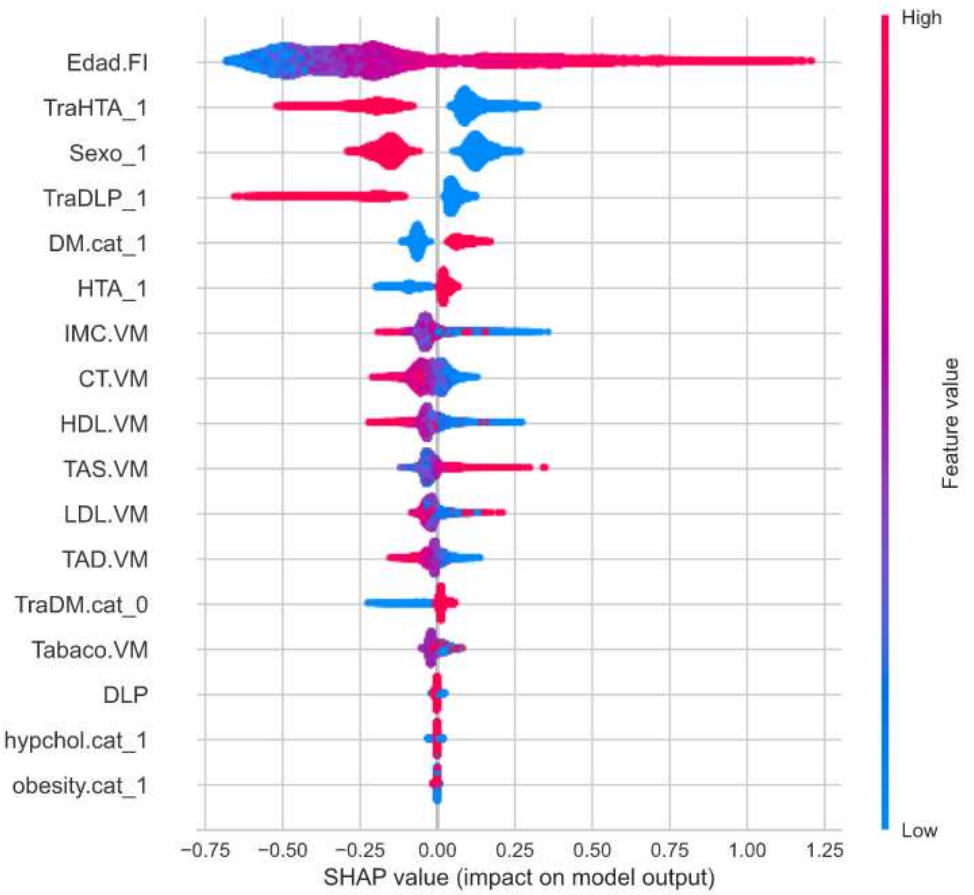


Figure 3.11 – SHAP interpretation of XGBoost in the first experiment on unseen data.

Dependence Contribution Plots Figure 3.12 shows the relation between the two most important features in the model, $mortality$, and $TreHTA$, as well as their feature values, their SHAP values, and their dependencies.

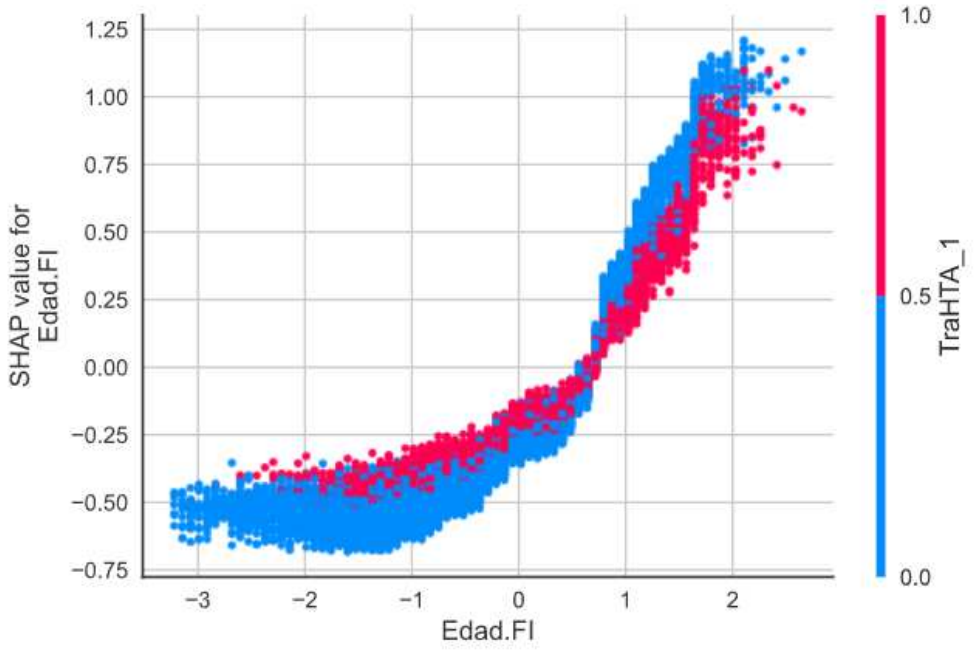


Figure 3.12 – SHAP correlation interpretation of XGBoost in 1st Experiment - test dataset.

The first insight of the plot is an almost exponential increase of the SHAP values based on age. A second insight is an inverted relationship between age and hypertension treatment. For example, in patients with normalized age between (-3.2 and 0.75), $TreHTA$ is an increasing factor, while in patients with a normalized age higher than 0.75 is a decrease factor.

Reasons Our predictions in Figure 3.13 show that being under high blood pressure treatment ($TreHTA$) and with an age = 0.3 very close to the mean are the major factors driving down the prediction of mortality in this particular observation. Other factors, which are decreasing the prediction are $DM.cat\_1$ = 0 and moderate blood pressure values when DBP = 0.5 and SBP = 0.5 numerical features are normalized. Factors that are increasing mortality prediction in this particular instance are; being male with a low HDL value of -1.6, not receiving dyslipidemia treatment, and being a $smoking$. Also, an interesting fact is that the model considers low TC = -1.7 as increasing factors.

Figure 3.13 – SHAP interpretation reasons of XGBoost in the first experiment on unseen data.

3.4.2 Undersampling Experiment

Undersampling Data Split

A 70/30 ratio is preserved, dedicating 70% of the data for model training and parameter optimization and 30% for testing. The frequency of the major class is reduced from 37,602 to 672, as shown in Table 3.9 to equalize the frequency with the minor class. Consequently, the $mortality$ changed from 1.7% to 50% in the training dataset. Since equally 672 survive and die, the test dataset remains unmodified using a stratified strategy in the data split, conserving the original 98%/2% ratio in the mortality class, as shown in Table 3.9. Although it is not necessary to reduce the number of observations in the majority class until it matches the observations of the minority class, it is usually done this way to observe the effects that undersampling has in the models clearly.

Table 3.9 – Frequencies of output variable across the different datasets used in Undersampling Experiment.

$mortality$	**Original Data**	**Training Data (before & after)**	**Test Data**
0	53,718	37,602 -> 672	16,116
1	960	672	288

Undersampling Data Processing

Due to the data split, the proportions of the input variables changed, and the variable coding is altered as listed in Table 3.10.

Table 3.10 – Label encoding to One Hot transformation values. Variables with prefix _0 use their complementary values, e.g., in a variable with prefix _0 value, representing the contrary than before the transformation.

Variable	Transformation	Values
Obesity ($obesity$)	obesity.cat_0	No=1;0=Yes
Hypertention treat. ($TreHTA$)	TraHTA_0	No=1;0=Yes
Hypercholesterolemia ($hypchol$)	hypchol.cat_0	No=1;0=Yes
Dyslipidemia treatment ($TreDLP$)	TraDLP_0	No=1;0=Yes

Undersampling Model Scoring

Undersampling produces an increase in Recall, Precision, and F1 and adds a reduction in Accuracy. It also observes a major homogenization between model metrics, i.e., CatBoost scores between 75% and 83% in its metrics. This is a significant small difference from the CatBoost results in the baseline experiment. This study Continues with the CatBoost model because it is the model with the highest F1 and Accuracy. It increases F1 from 1.73% to 75%, while Accuracy only reduces from 98% to 76%.

Table 3.11 – Model Selection Results using the training dataset in the Undersampling Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
CatBoost	**0.7589**	0.7515	0.7669	**0.7612**	**0.8384**
Ridge	0.7576	**0.7589**	0.7569	0.7574	0.0000
LDA	0.7561	0.7574	0.7554	0.7560	0.8365
GBC	0.7558	0.7470	0.7661	0.7589	0.8322
AdaBoost	0.7544	0.7574	0.7538	0.7545	0.8131
LR	0.7529	0.7515	0.7551	0.7537	0.8382
RF	0.7515	0.7292	**0.7759**	0.7589	0.8250
LightGBM	0.7441	0.7277	0.7620	0.7500	0.8171
XGBoost	0.7390	0.7307	0.7481	0.7418	0.8124
ET	0.7377	0.7188	0.7581	0.7448	0.8084
QDA	0.7279	0.7143	0.7427	0.7336	0.8047
NB	0.7232	0.7202	0.7266	0.7247	0.7945
KNN	0.7034	0.7054	0.7016	0.7024	0.7516
SVM	0.6553	0.6488	0.7400	0.6905	0.0000
DT	0.6511	0.6503	0.6525	0.6518	0.6518

Undersampling Model Tuning

The undersampling experiment uses the same optimization parameters as the previous baseline experiment, where the *Optuna* [26] library replaces *SciKit-learn* [21], which is the one used by default in *Pycarets* [12]. Models are also optimized towards F1 using 100 iterations. The first observation is that the optimization of undersampling models has far less significant effects than the optimization on the baseline models. Table 3.12 shows model scores after the tuning activity. The minor improvement effect may be explained by the significant improvement already achieved in the model scoring. Therefore, model tuning does not play a significant role in undersampling model optimization. A similar lack of improvement is shown in the usage of weight_class parameter over non weight_class parameter optimization as listed in Tables 3.12 and 3.13.

Table 3.12 – Model Tuning Results using the training dataset in the Undersampling Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
AdaBoost	**0.7669**	0.7649	0.7703	0.7679	0.8215
SVM	**0.7669**	**0.7976**	0.7390	0.7574	0.0000
CatBoost	0.7659	0.7485	0.7857	0.7716	0.8340
GBC	0.7656	0.7679	0.7643	0.7656	0.8280
XGBoost	0.7632	**0.7976**	0.7324	0.7530	0.8289
ET	0.7610	0.7307	0.7954	**0.7708**	0.8290
Ridge	0.7607	0.7604	0.7616	0.7612	0.0000
LDA	0.7579	0.7664	0.7497	0.7552	**0.8343**
LightGBM	0.7579	0.7574	0.7595	0.7582	0.8276
LR	0.7559	0.7634	0.7489	0.7537	0.8334
QDA	0.7525	0.7589	0.7467	0.7507	0.8236
RF	0.7524	0.7247	**0.7861**	0.7626	0.8273
KNN	0.7467	0.7470	0.7470	0.7463	0.8077
NB	0.7443	0.7812	0.7108	0.7314	0.8065
DT	0.7246	0.7262	0.7230	0.7240	0.7865

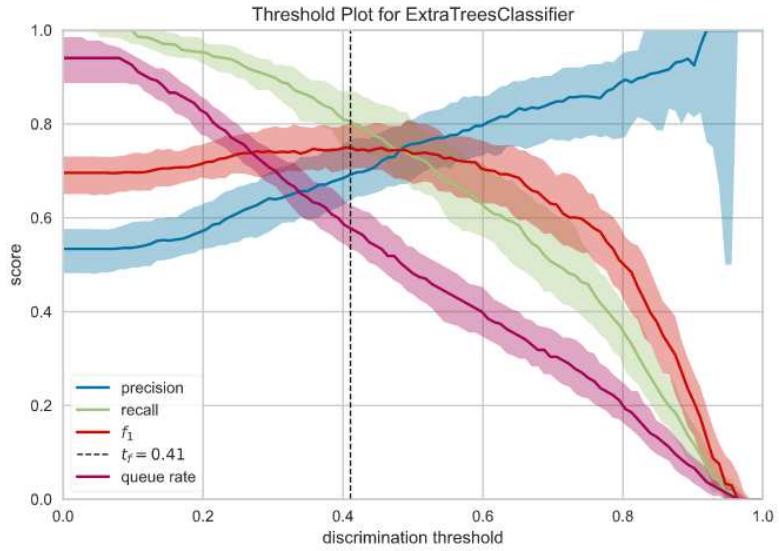
Table 3.13 – Model Tuning Result setting class_weight parameter to balance training data for the Undersampling Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
LightGBM	**0.7651**	**0.7589**	0.7728	0.7671	**0.8332**
RF	0.7581	0.7292	0.7901	0.7671	0.8312
ET	0.7637	0.7396	**0.7904**	**0.7716**	0.8267
DT	0.7321	0.7485	0.7189	0.7262	0.7922
Ridge	0.7576	**0.7589**	0.7569	0.7574	0.0000

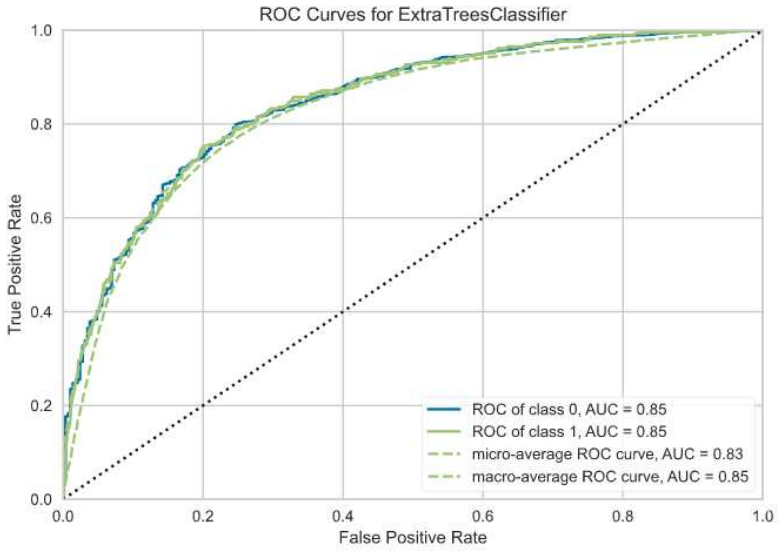
Undersampling Model Analysis

LightGBM is selected for model analysis and model interpretation due to its similar scoring to the best performing models. AdaBoost and SVM as well as its DT nature allows SHAP interpretability. Figure 3.14 shows the relationship between TPR/Recall, FPR, Precision, F1, and Threshold. This is illustrated

using the ROC-AUC in Figure 3.14b and the Threshold in Figure 3.14a. The former displays relations between Precision, Recall, and F1, which reached their peak with a 0.4 Threshold.



(a) Threshold of LightGBM in the Undersampling Experiment on training data.



(b) ROC_AUC of LightGBM in the Undersampling Experiment on training data.

Figure 3.14 – ROC_AUC and Precision-Recall curves of LightGBM in Undersampling Experiment on training data.

Undersampling Model Predictions

The confusion matrix of the LightGBM model described in Figure 3.15 illustrates how out of a total of 16,404 observations, true negatives represent a clear majority group, accounting for 14,720 observations. Next are the false positives accounting for 1,396 cases, followed by false negatives with 136 observations leaving the true positives with 152 observations.

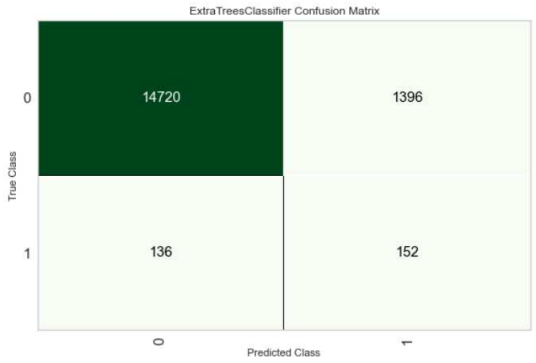


Figure 3.15 – Confusion Matrix and Class Report of ET in Undersampling Experiment

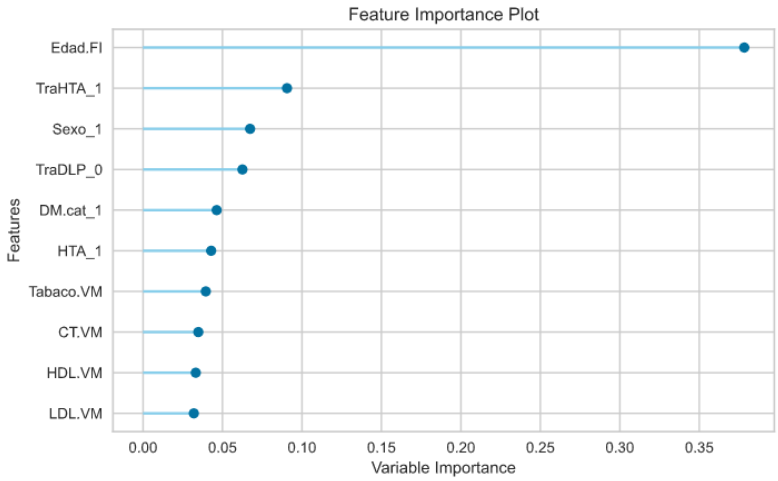
Table 3.14 list LightGBM predictions on testing data. General conclusions derived from these results are that Precision suffers a significant reduction to values between 4% and 6%, bringing down F1, while Recall and Accuracy values remain similar but still inferior compared to baseline values. In conclusion, undersampling brings homogeneity to the model, reducing the range of their measures, increase Recall and reduce Accuracy slightly and very significantly Precision and, therefore, F1.

Table 3.14 – Model Tuning Results using the test dataset in Undersampling Experiment.

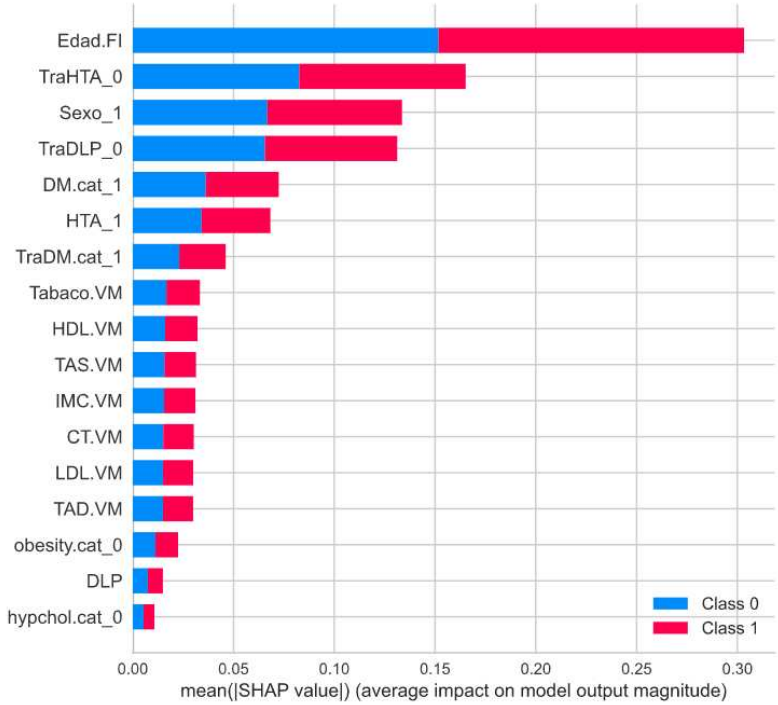
Model	F1	Recall	Prec.	Accuracy	AUC
SVM	**0.1213**	0.6840	**0.0665**	**0.8260**	0.7563
ET	0.1087	0.7396	0.0587	0.7871	**0.8448**
RF	0.1036	0.7118	0.0559	0.7838	0.8324
CatBoost	0.1008	0.7396	0.0541	0.7683	0.8338
LightGBM	0.0992	0.7535	0.0531	0.7599	0.8327
Ridge	0.0971	0.7778	0.0518	0.7460	0.7616
LR	0.0935	0.7708	0.0498	0.7376	0.8284
GBC	0.0927	0.7465	0.0494	0.7433	0.8191
AdaBoost	0.0927	0.7361	0.0495	0.7470	0.8156
KNN	0.0927	0.7708	0.0493	0.7351	0.8256
LDA	0.0914	0.7674	0.0486	0.7321	0.8281
QDA	0.0906	0.7708	0.0481	0.7282	0.8223
DT	0.0852	0.7083	0.0453	0.7331	0.7784
XGBoost	0.0846	**0.8090**	0.0446	0.6926	0.8308
NB	0.0804	0.7812	0.0424	0.6862	0.8059

Undersampling Model Interpretation

Feature importance of LightGBM is analyzed in Figure 3.16 using *Pycaret* [12] and SHAP graphics. The first Figure (3.16a) shows the feature importance plot provided by *Pycaret* [12] based on the coefficient value. The second Figure (3.17) is obtained from SHAP. Feature Importance shows a ranking of the top ten more relevant features of the model. Three groups are distinguished; the first one is made up by age and High blood pressure treatment, the second one including High blood pressure, $gender$, DLP, and diabetes mellitus treatments and DM, and the third group conformed by TC, BMI and HDL, all of them with the same importance level. In this graph, the top five measures account for around 50% of the total importance of the model.



(a) Features of LightGBM in the Undersampling Experiment on training data.



(b) SHAP Summary Plot of LightGBM in the Undersampling Experiment on training data.

Figure 3.16 – Comparison of feature importance of LightGBM and SHAP in the Undersampling Experiment on training data.

SHAP Summary Plot For the LightGBM displayed in Figure 3.17, more holistic information is provided than for the one previously in Figure 3.16a since the SHAP value plot does not only give information about the overall feature importance but also more granular information of the impact it has on the model. For example, it shows the range of SHAP Values that a feature may have. The top six features in the SHAP Summary plot are the same and in the same ranking as in the model feature importance but do not have the same importance. Edad is the feature with the highest impact on the model, and with the biggest SHAP value range, from around -0.70 to 1.20, directly correlated with feature values and SHAP values. For categorical variables, values are divided into two clusters based on the category of the features. For example, disease treatments reduce mortality prediction, while CVDs increase it. Other variables like $smoking$, DLP, $hypchol$, $obesity$ do not significantly affect the model. Also, there is no clear correlation between variable values and SHAP values in the case of $smoking$.

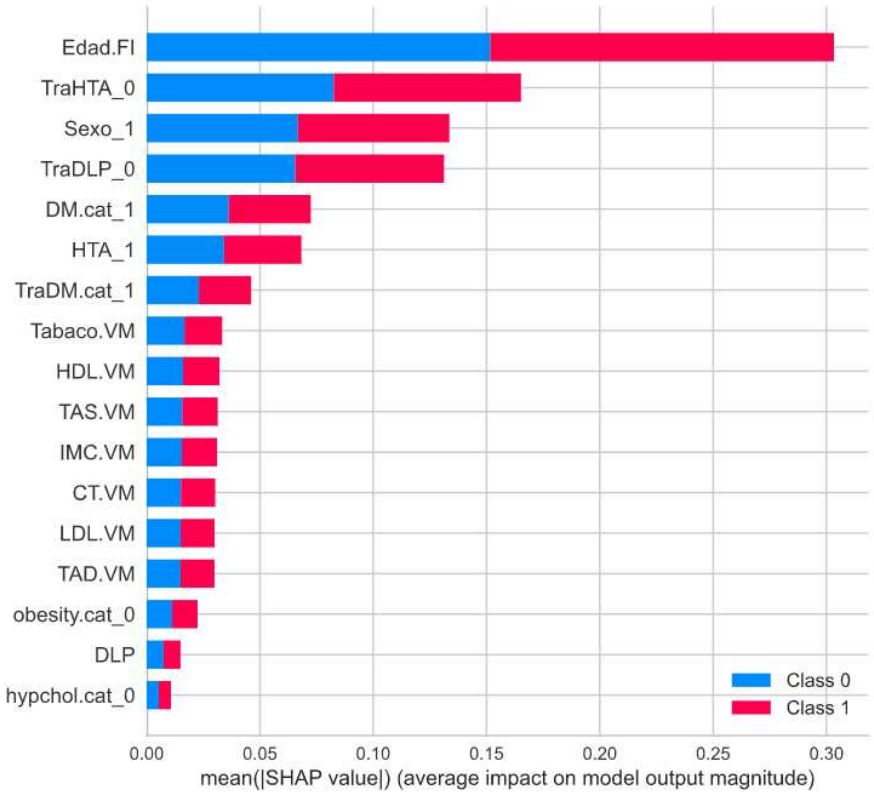


Figure 3.17 – SHAP Summary Plot of ET in Undersampling Experiment on training data.

Dependence Contribution Plots Figure 3.18 shows the relation between the two most important features in the model, age and $TreHTA$, their feature values, including their SHAP values, and their dependencies.

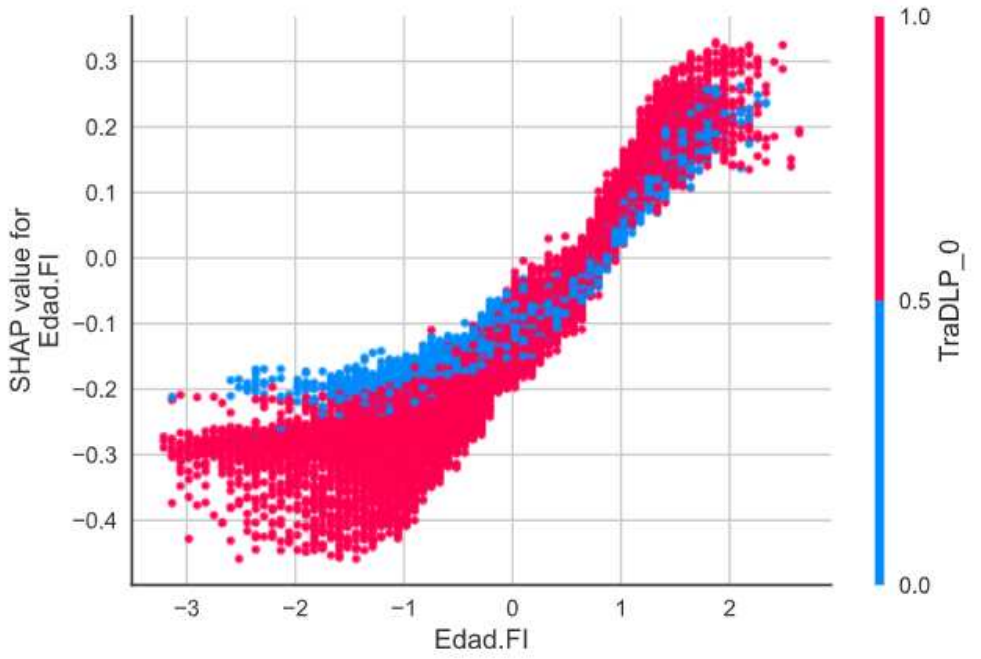


Figure 3.18 – SHAP Correlation Plot of ET in the Undersampling Experiment on training data.

The same insight as for the baseline model is obtained, where age produces an almost exponential increase of the SHAP values and the inverted relationship between age and $TreHTA$. For example, in patients with normalized age between (-3.2 and 0.75), $TreHTA$ is an increasing factor, while in patients with age higher than 0.75 is a decrease factor.

Reasons When looking at our particular prediction on Figure 3.19, having a normalized age of age = -0.98 and being under $TreDM$ are the major factors driving down mortality prediction in this particular observation. Other factors decreasing the prediction are female gender and moderate blood pressure (HTA) in the mean value (0) numerical features are normalized).

Figure 3.19 – SHAP Reasons Plot of ET in Undersampling Experiment on training data.

On the other hand, factors increasing mortality prediction in this particular instance are; being diagnosed DM and not receiving neither dyslipidemia nor high blood pressure treatment (TraDLP_0 = 1).

3.4.3 Oversampling Experiment

This experiment aims to deal with the problems associated with class imbalance by oversampling the minority class. Hence, instead of reducing observations of the majority class like in the previous experiment, observations of the minority class are augmented to balance class proportion.

Oversampling Data Split

A 70/30 ratio is preserved, dedicating 70% of the data for model training and parameter optimization and 30% for testing. The frequency of the minority class is augmented from 672 to 37,602, as shown in Table 3.15 to equalize its frequency with the majority class. Consequently, the mortality class proportion changed from 1.7% to 50% in the training dataset. The test dataset remains unmodified by using a stratified strategy in the data split, conserving the original 98%/2% ratio in the mortality class, as shown in Table 3.15.Although it is not necessary to augment the number of observations in the minority class until it matches the observations of the majority class, it is done in this study this way to clearly observed the effects that Oversampling has on the models.

Table 3.15 – Frequencies of output variables across different datasets in the Oversampling Experiment.

$mortality$	**Original Data**	**Training Data (before & after)**	**Test Data**
0	53,718	37,602	16,116
1	960	672->37,602	288

Oversampling Data Processing

Due to the data split, the proportions of the input variables changed, and the variable coding is altered as listed in Table 3.16.

Table 3.16 – Label encoding to One Hot transformation values. Variables with prefix _0 use their complementary values, e.g., in a variable with prefix _0 value, one represents the contrary than before the transformation.

Variable	Transformation	Values
gender (*gender*)	Sexo_0	Female=1;0=Male
obesity (*obesity*)	obesity.cat_0	No=1;0=Yes

Oversampling Models Scoring

In models that use oversampling, all metrics improve the results concerning undersampling and baseline models, with top models reaching almost 100% Precision.

Table 3.17 – Model Performance Results using the test dataset in the Oversampling Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
ET	**0.9998**	**1.0000**	**0.9996**	**0.9998**	**1.0000**
RF	0.9996	**1.0000**	0.9991	0.9996	**1.0000**
DT	0.9892	**1.0000**	0.9786	0.9891	0.9891
CatBoost	0.9827	**1.0000**	0.9660	0.9824	0.9989
XGBoost	0.9778	**1.0000**	0.9565	0.9772	0.9976
KNN	0.9734	1.0000	0.9481	0.9726	0.9901
LightGBM	0.9409	0.9846	0.9010	0.9382	0.9803
GBC	0.8202	0.8303	0.8103	0.8179	0.8947
AdaBoost	0.7736	0.7700	0.7772	0.7746	0.8603
Ridge	0.7585	0.7630	0.7540	0.7570	0.0000
LDA	0.7585	0.7630	0.7540	0.7570	0.8410
LR	0.7548	0.7546	0.7551	0.7549	0.8415
SVM	0.7475	0.7304	0.7679	0.7539	0.0000
QA	0.7446	0.7310	0.7587	0.7492	0.8327
NB	0.7253	0.7180	0.7327	0.7280	0.7964

Oversampling Model Tuning

Results from the classifiers by using the training dataset are collected in Table 3.18. These results are almost identical to the ones shown in the previous section on Oversampling model scoring. Therefore, it can be concluded that there is no significant effect on the optimization of oversampling models from Model Tuning.

Table 3.18 – Model Tuning Result setting class_weight parameters to balance using training data in the Oversampling experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
GBC	**0.9986**	**1.0000**	**0.9972**	**0.9986**	**1.0000**
LightGBM	0.9982	**1.0000**	0.9964	0.9982	**1.0000**
XGBoost	0.9944	**1.0000**	0.9888	0.9943	**1.0000**
KNN	0.9908	**1.0000**	0.9819	0.9908	0.9908
CatBoost	0.9684	0.9997	0.9389	0.9673	0.9922
DT	0.9631	0.9945	0.9336	0.9619	0.9813
RF	0.9600	0.9949	0.9274	0.9585	0.9900
ET	0.9328	0.9716	0.8970	0.9300	0.9740
AdaBoost	0.7890	0.7890	0.7891	0.7891	0.8723
QDA	0.7610	0.7684	0.7538	0.7587	0.8421
SVM	0.7603	0.7699	0.7511	0.7574	0.0000
Ridge	0.7585	0.7630	0.7540	0.7570	0.0000
LDA	0.7585	0.7630	0.7540	0.7570	0.8410
LR	0.7548	0.7546	0.7551	0.7549	0.8415
NB	0.7454	0.7876	0.7075	0.7310	0.8073

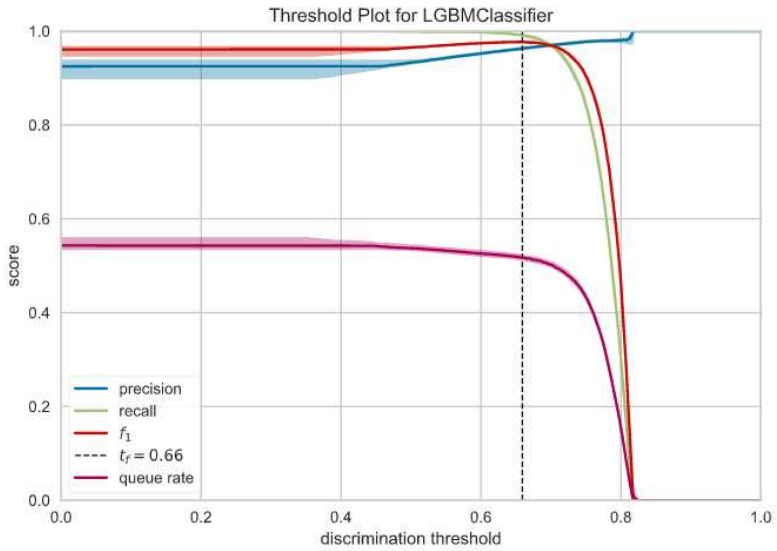
The classifiers in Table 3.19 share the optimization parameters settings except for class_weight, which is set to balance. The setting of the class_weight parameter as "balanced" produces a slight penalty on the metrics of the models trained with oversampling data.

Table 3.19 – Model Tuning Result setting class_weight parameter to balanced using training data for the Oversampling experiment.

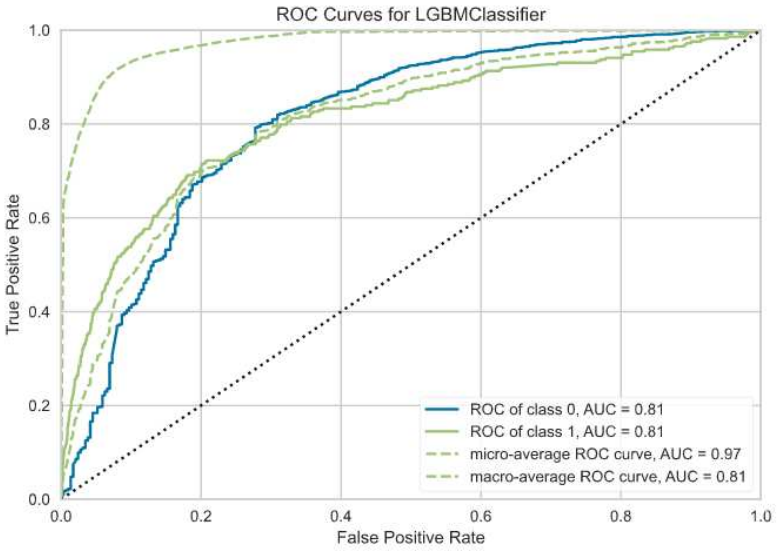
Model	F1	Recall	Prec.	Accuracy	AUC
DT	**0.9628**	0.9968	**0.9311**	**0.9615**	0.9799
LightGBM	0.9624	**0.9988**	0.9285	0.9610	**0.9918**
RF	0.9421	0.9785	0.9083	0.9398	0.9779
ET	0.9120	0.9486	0.8782	0.9085	0.9587
Ridge	0.7585	0.7630	0.7540	0.7570	0.0000

Oversampling Model Analysis

Model LightGBM is the model selected for analysis as it was previously analyzed in the undersampling experiment, and using the same model across different sampling techniques provides the opportunity of getting a better understanding of their effects. Figure 3.20 shows the relationship between Recall, FPR, Precision, F1, and Threshold. This is illustrated in Figures 3.20 by showing the ROC-AUC and Threshold. Relationships between Precision, Recall, and F1 are displayed in Figure 3.20a. It shows that F1 reaches its peak with a 0.62 Threshold. After that point, Recall falls drastically while Precision maintains a mild increase and the value of F1 drops.



(a) Threshold of RF in the Oversampling Experiment on training data.



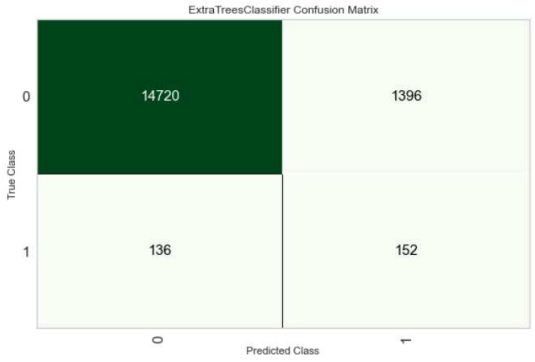
(b) ROC_AUC of RF in the Oversampling Experiment on training data.

Figure 3.20 – ROC_AUC and Precision-Recall curves of RF in Oversampling Experiment on training data.

Oversampling Model Prediction

The confusion matrix of the LightGBM model in Figure 3.21 shows that out of 16,404 observations, true negatives represent a clear majority group accounting for 14,720 observations. Next are the false positives accounting for 13,96 cases,

followed by the false negatives with 136 observations leaving the true positives with 152 observations.



(a)

Figure 3.21 – Confusion Matrix and Class Report of LightGBM in Oversampling Experiment

The prediction results from the classifiers using the test dataset are displayed in Table 3.20 and Table 3.21 shows the prediction results of classifiers with class_weight parameter with value 'balance'.

Table 3.20 – Model Prediction Results using the test dataset in the Oversampling Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
RF	0.1686	0.4861	0.1020	0.9158	0.8285
CatBoost	0.1567	0.3958	0.0977	0.9252	0.8172
ET	0.1479	0.5903	0.0845	0.8806	0.8325
DT	0.1154	0.2847	0.0724	0.9234	0.6080
XGBoost	0.1079	0.0833	0.1529	0.9758	0.7672
AdaBoost	0.1046	0.7222	0.0564	0.7829	0.8199
SVM	0.1024	0.7604	0.0549	0.7659	0.7632
LR	0.0990	0.7778	0.0529	0.7515	0.8363
Ridge	0.0987	0.7882	0.0527	0.7473	0.7674
LDA	0.0987	0.7882	0.0527	0.7473	0.8369
QDA	0.0944	0.7674	0.0503	0.7415	0.8240
KNN	0.0882	0.0868	0.0896	0.9685	0.5355
NB	0.0759	0.7708	0.0399	0.6703	0.8037
LightGBM	0.0601	0.0347	0.2222	0.9809	0.7796
GBC	0.0500	0.0278	0.2500	0.9815	0.7247

While the class_weight parameter does not provide a significant improvement in four out of the five classifiers that have been oversampled, it does for LightGBM, increasing its F1 in prediction data from 6% to 17%.

Table 3.21 – Model Prediction Results for classifiers with weight_class parameter set to balance on the test dataset in the Oversampling Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
LightGBM	0.1762	0.5208	0.106	0.9145	0.847
RF	0.1509	0.5382	0.0878	0.8937	0.8201
ET	0.1384	0.6319	0.0777	0.8619	0.839
DT	0.1106	0.2882	0.0684	0.9186	0.6102
Ridge	0.0987	0.7882	0.0527	0.7473	0.7674

Oversampling Model Interpretation

The feature importance of LightGBM is analyzed using *Pycaret* [12] and SHAP graphics. Figure 3.22 shows the feature importance plot provided by *Pycaret* [12] based on the coefficient values, while Figure 3.23 is obtained from SHAP. Feature Importance shows a ranking of the top ten most relevant features of

the model. The ranking of feature importance provided by LightGBM, when the class_weight parameter is set as the balance, is radically different from the ones seen in previous experiments. Numerical variables scale to the top of the table and the categorical variables are placed at the bottom. Also, *age*, which is normally set as the main feature with a clear difference, is now placed only on the seventh position of the table and did not make it in the top three.

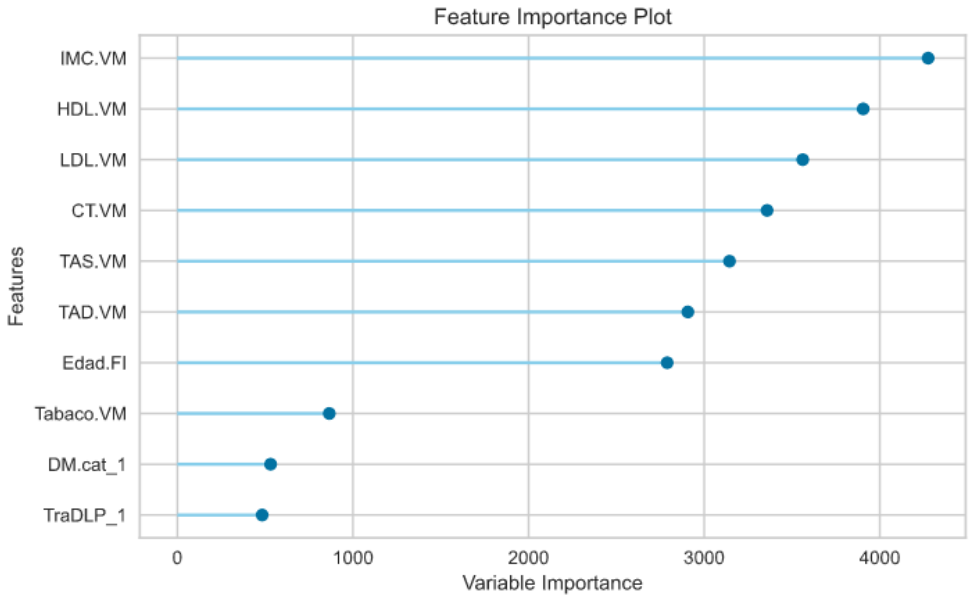


Figure 3.22 – Features of LightGBM in the Oversampling Experiment on training data.

SHAP Summary Plot The SHAP summary plot for the XGBoost model is displayed in Figure 3.23. Although not identical to the ones provided in the previous experiment, it maintains a similar feature ranking where the four features are the same as in undersampling and baseline models. Additional feature positions show small variations.

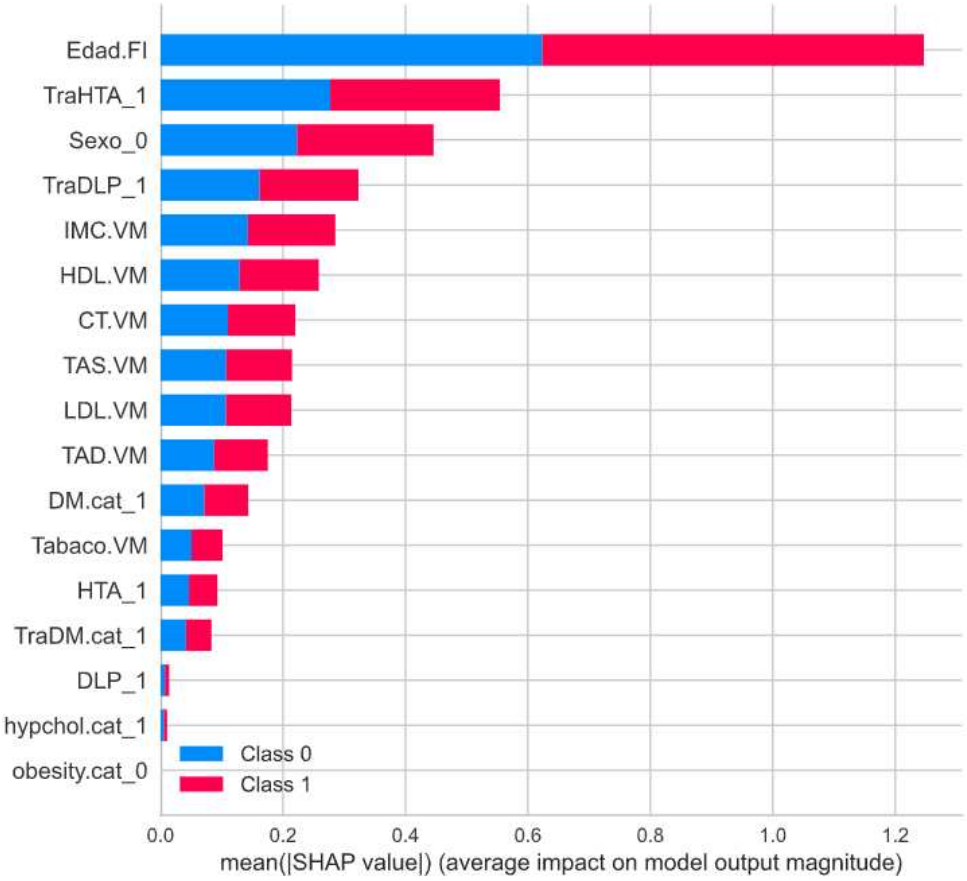


Figure 3.23 – SHAP Summary Plot of LightGBM in the Oversampling Experiment on training data.

Dependence Contribution Plots Figure 3.24 shows the relation between the two more important features in the model, age and $TreHTA$, including their feature values, their SHAP values, and their dependencies. It shows a positive correlation between SHAP values and age and no relationship between age and $TreHTA$

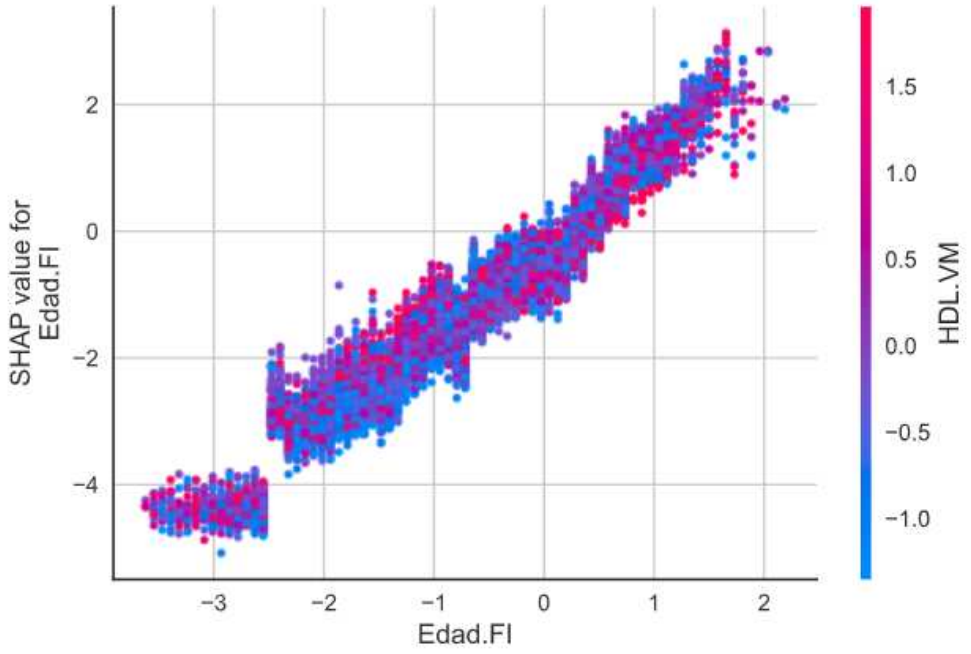


Figure 3.24 – SHAP Correlation Plot of ET in the Oversampling Experiment on training data.

Reasons An observation with a considerable low prediction is evaluated in Figure 3.25, where about half of the negative SHAP Values are driven by a normalized age of 0.3 and being under high blood pressure treatment, additional decreasing factors are not being diagnosed with DM and low blood pressure (SBP and SBP = -0.5).

Figure 3.25 – SHAP Reasons Plot of ET in the Oversampling Experiment on training data.

On the other hand, the main increasing factors are; being male with very low cholesterol levels (HDL and TC equal to -1.8), not being under a dyslipidemia treatment, and being a smoker.

3.4.4 SMOTE Experiment

This experiment aims to understand and take advantage of reducing the class imbalance of the dataset by using SMOTE, a more advanced oversampling technique than the one used in the oversampling experiment.

SMOTE Data Split

A 70/30 ratio is preserved, dedicating 70% of the data for model training and parameter optimization and 30% for testing. The frequency of the minority class is augmented from 672 to 37,602, as shown in Table 3.22, to equalize its frequency with the majority class. Consequently, the mortality class proportion changed from 1.7% to 50% in the training dataset. Test dataset remains unmodified using a stratified strategy in the data split, conserving the original 98%/2% ratio in the mortality class, as shown in Table 3.22. Although it is not necessary to augment the number of observations in the minority class until it matches the observations of the majority class, it is done this way to observe the effects that SMOTE has on the models clearly.

Table 3.22 – Frequencies of output variable across the different datasets in the SMOTE Experiment.

mortality	**Original Data**	**Training Data**	**Test Data**
0	53,718	37,602	16,116
1	960	672 -> 37,602	288

SMOTE Data Processing

Due to the data split, the proportions of the input variables changed and the variable coding. Altered Table 3.23 shows the categorical variables which are used in the SMOTE experiment.

Table 3.23 – Label encoding to One Hot transformation values. Variables with prefix _0 use their complementary values, e.g., in a variable with prefix _0 value, representing the contrary than before the transformation.

Variable	Transformation	Values
Obesity (*obesity*)	obesity.cat_0	No=1;0=Yes
Hypertention treat ($TreHTA$)	TraHTA_0	No=1;0=Yes
Hypercholesterolemia (*hypchol*)	hypchol.cat_0	No=1;0=Yes

SMOTE Models Scoring

Table 3.24 shows scoring results from classifiers, using random search from the *Scikit-Learn* [21] library, optimized towards Accuracy. The results are in general aligned with the ones reported by using simple oversampling but are slightly lower. They are showing almost perfect scores.

Table 3.24 – Model Scoring Results using the training dataset in the SMOTE Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
CatBoost	**0.9843**	0.9826	**0.9865**	**0.9845**	0.9972
XGBoost	0.9837	0.9829	0.9851	0.9839	0.9974
ET	0.9795	0.9906	0.9687	0.9793	0.9976
RF	0.9788	0.9890	0.9688	0.9786	**0.9977**
LightGBM	0.9699	0.9738	0.9664	0.9700	0.9952
DT	0.9526	0.9678	0.9381	0.9519	0.9519
KNN	0.4079	**0.9970**	0.8903	0.9371	0.9770
GBC	0.9074	0.9197	0.8956	0.9062	0.9679
AdaBoost	0.8854	0.8970	0.8743	0.8840	0.9548
SVM	0.8747	0.9010	0.8501	0.8710	0.0000
LR	0.8745	0.8886	0.8610	0.8726	0.9430
Ridge	0.8742	0.9076	0.8433	0.8694	0.0000
LDA	0.8742	0.9075	0.8433	0.8694	0.9422
QDA	0.8572	0.9081	0.8118	0.8488	0.9262
NB	0.8459	0.9034	0.7952	0.8354	0.9111

SMOTE Model Tuning

The model optimization using *Optuna* [26] produces almost identical results. Therefore, it shows no significant difference from the previous scoring activity.

Table 3.25 – Model Tuning Results using the test dataset in the SMOTE Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
GBC	**0.9901**	0.9888	0.9917	**0.9903**	**0.9990**
LightGBM	0.9882	0.9845	**0.9923**	0.9884	0.9984
CatBoost	0.9863	0.9866	0.9862	0.9864	0.9982
XGBoost	0.9848	0.9837	0.9864	0.9850	0.9977
KNN	0.9704	**0.9971**	0.9450	0.9695	0.9704
DT	0.9394	0.9602	0.9196	0.9381	0.9678
RF	0.9378	0.9688	0.9089	0.9358	0.9773
ET	0.9204	0.9521	0.8908	0.9177	0.9659
AdaBoost	0.8994	0.9076	0.8916	0.8986	0.9646
SVM	0.8764	0.8939	0.8595	0.8739	0.0000
LR	0.8746	0.8885	0.8611	0.8726	0.9430
Ridge	0.8742	0.9076	0.8433	0.8694	0.0000
LDA	0.8742	0.9076	0.8433	0.8694	0.9422
QDA	0.8700	0.9180	0.8268	0.8628	0.9372
NB	0.8490	0.8981	0.8050	0.8403	0.9162

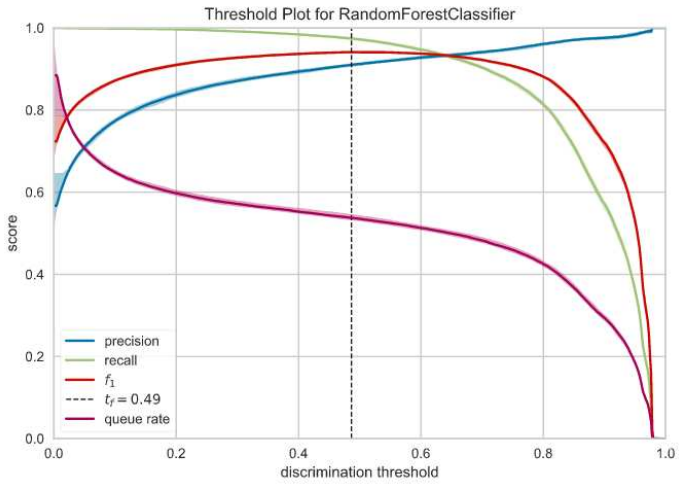
The tuning of the class_weight parameter in the SMOTE training dataset does not have any visible effect in the models.

Table 3.26 – Model Tuning Results for classifiers with weight_class parameter set to balance on training dataset in the SMOTE Experiment.

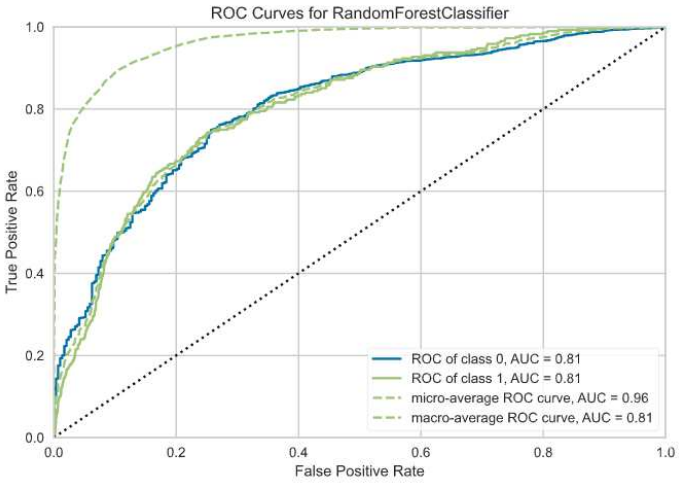
Model	F1	Recall	Prec.	Accuracy	AUC
LightGBM	**0.9902**	**0.9878**	**0.9929**	**0.9903**	**0.9990**
RF	0.9424	0.9638	0.9220	0.9411	0.9626
ET	0.9402	0.9717	0.9108	0.9383	0.9791
DT	0.9211	0.9534	0.8910	0.9184	0.9660
Ridge	0.8742	0.9076	0.8433	0.8694	0.0000

SMOTE Model Analysis

The model selected for analysis in this experiment is the RF. Figure 3.26 shows the relationship between Recall, FPR, Precision, F1, and Threshold. This is illustrated by using two graphics: Threshold plot and ROC-AUC. Relationships between Precision, Recall, F1 are displayed in Figure 3.26, which shows that F1 reaches its peak with almost a 0.5 Threshold.



(a) Threshold of RF in the SMOTE Experiment on training data.



(b) ROC_AUC of RF in the SMOTE Experiment on training data.

Figure 3.26 – ROC_AUC and Precision-Recall curves of RF in the SMOTE Experiment on training data.

SMOTE Model Prediction

The confusion matrix of the RF model in Figure 3.27 illustrated that out of 16,404 observations, the number of true negatives cases represent a clear majority group accounting for 14,917 observations. Next are the false positives accounting for 1,199 cases, followed by false negatives with 147 observations leaving the true positives with 141 observations.

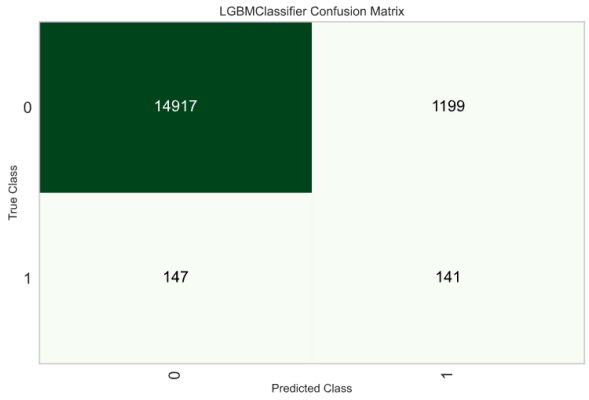


Figure 3.27 – Confusion Matrix and Class Report of RF in the SMOTE Experiment

Table 3.27 shows prediction results from the classifier using the test dataset. Classifiers in Table 3.28 have the same optimization parameters settings except for class_weight, which is set to balance. However, there are no significant differences in the results.

Table 3.27 – Model Prediction Results using the test dataset in the SMOTE Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
AdaBoost	0.1348	0.5278	0.0773	0.8811	0.7893
RF	0.1308	0.4618	0.0762	0.8923	0.8012
ET	0.1267	0.5382	0.0718	0.8697	0.7955
SVM	0.1204	0.5972	0.0670	0.8469	0.7243
LR	0.1201	0.5868	0.0669	0.8490	0.7881
Ridge	0.1130	**0.6424**	0.0619	0.8229	0.7342
LDA	0.1130	**0.6424**	0.0619	0.8229	0.7965
KNN	0.1066	0.2118	0.0713	0.9377	0.6393
DT	0.1029	0.3021	0.0620	0.9075	0.6906
QDA	0.0996	0.6389	0.0540	0.7971	0.7656
LightGBM	0.0952	0.0660	**0.1712**	0.9780	**0.8038**
NB	0.0910	**0.6424**	0.0490	0.7748	0.7443
XGBoost	0.0889	0.0764	0.1063	0.9725	0.7821
GBC	0.0821	0.0590	0.1349	**0.9768**	0.7781
CatBoost	0.0646	0.0556	0.0773	0.9718	0.7746

Table 3.28 – Model Prediction Results for classifiers with weight_class parameter set to balance on the test dataset in the SMOTE Experiment.

Model	F1	Recall	Prec.	Accuracy	AUC
RF	0.1365	0.4757	0.0797	0.8943	0.8053
ET	0.1273	0.5312	0.0723	0.8721	0.7950
Ridge	0.1130	0.6424	0.0619	0.8229	0.7342
DT	0.1072	0.3194	0.0644	0.9065	0.6533
LightGBM	0.0755	0.0486	0.1687	0.9791	0.7817

SMOTE Model Interpretation

Feature importance of RF is analyzed in Figures 3.28 and 3.29 using *Pycaret* [12] and SHAP graphics. Feature Importance shows a ranking of the top ten most relevant features of the model. Like in the previous experiments, the four first features, age, $TreHTA$, $gender$, and $TreDLP$ are constant, and

the other six show very similar importance, which justifies a ranking change with the previous experiments.

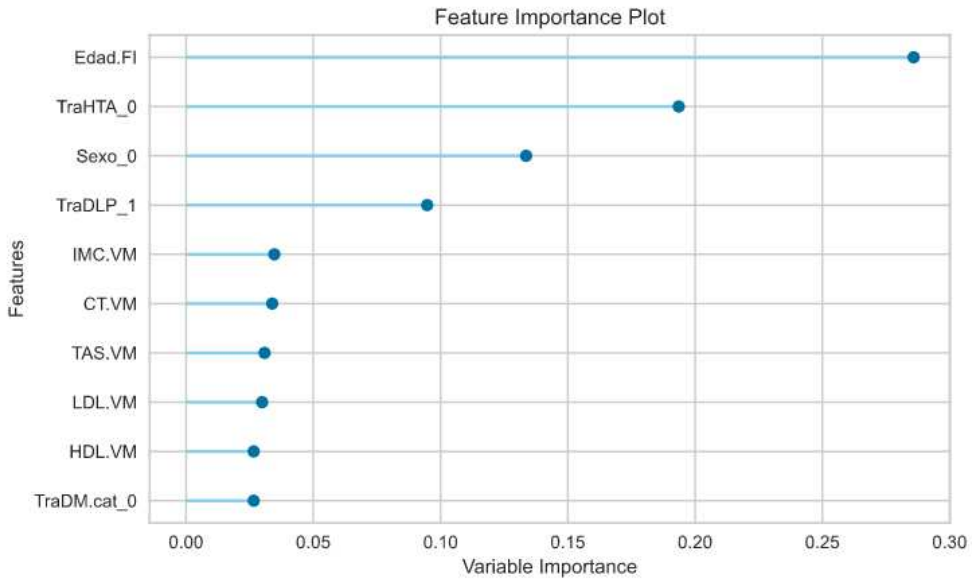


Figure 3.28 – Features of RF in the SMOTE Experiment on training data.

SHAP Summary Plot The SHAP summary plot for the RF model, which is displayed in Figure 3.29 shows the same top four features as the ones provided by the model in Figure 3.28. Also, these four features represent most of the SHAP values.

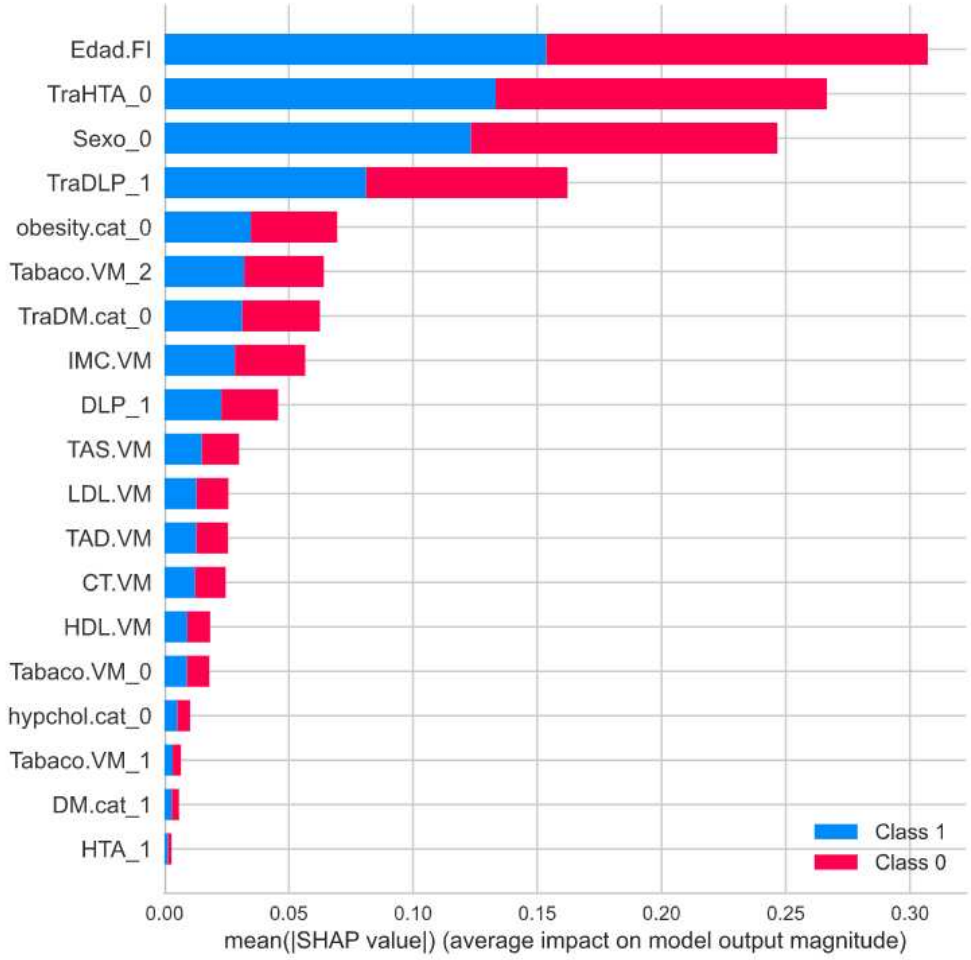


Figure 3.29 – SHAP Summary Plot of RF in the SMOTE Experiment on training data.

Dependence Contribution Plots Figure 3.30 shows the relation between the two most important features in the model, age and $TreHTA$, including their feature values, their SHAP values, and their dependencies.

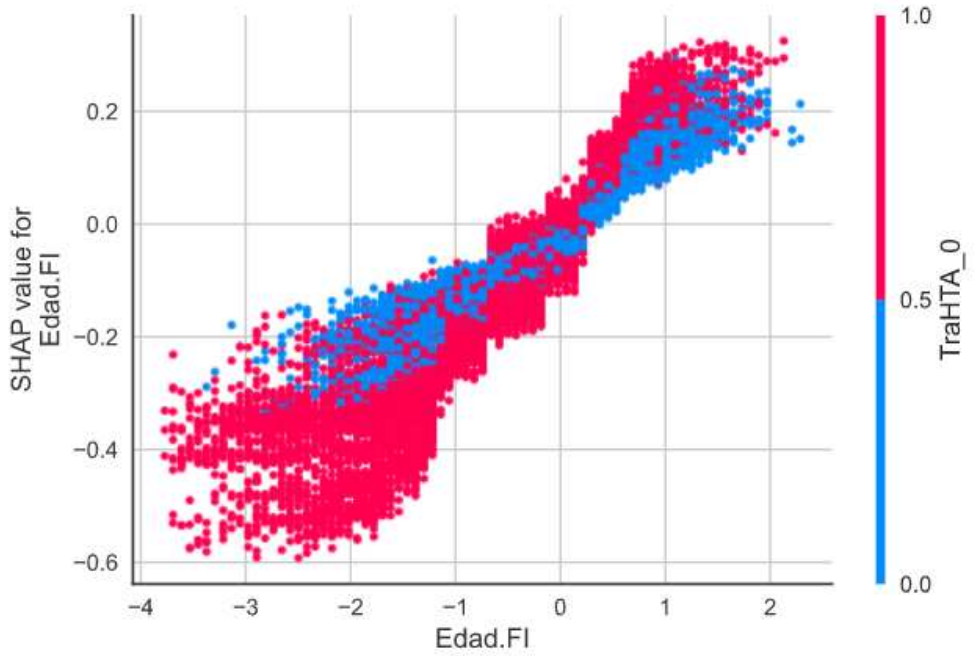


Figure 3.30 – SHAP Correlation Plot of RF in the SMOTE Experiment on training data.

The first insight of the plot is an almost exponential increase of the SHAP values based on age. A second insight is an inverted relationship between age and hypertension treatment. For example, in patients with a normalized age between (-3 and 0), $TreHTA$ is an increasing factor, while in patients with age higher than 0.25, it is a decreasing factor.

Reasons When looking at our prediction in Figure 3.31, the base value 0.5 is decreased to a function value of 0.01 mainly due to two features; $TreDLP$ and $TreHTA$. Only the male gender provides a low positive SHAP value.

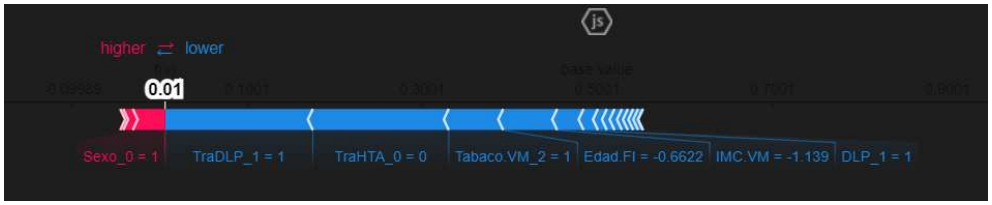


Figure 3.31 – SHAP Reasons Plot of RF in the SMOTE Experiment on training data.

Chapter 4

Conclusions and Future work

This last chapter summarizes the research conclusion, providing a summary of the most relevant experiment and cross experimental conclusions in Section 4.1. A discussion of constraints applied to this research is in Section 4.2. Lastly, the areas uncover and left for future work are listed in Section 4.3.

4.1 Conclusions

In this study, a predictive binary classification model for cardiovascular diseases with a highly imbalanced clinical dataset has been proposed focusing on the mortality, the minority class aiming to work towards an early detection system of CDVs. To avoid falling into the Accuracy paradox produced by the extreme class imbalance, evaluation measures like F1 or Recall were used instead of Accuracy. Four common balancing sampling practices were applied to solve unbalancing class problems and different data preprocessing techniques to improve model performance. Traditional statistic techniques were applied to obtain information regarding the dataset. We find dependencies between mortality and the following key categorical variables: $gender$, $smoking$, DM, $TreDM$, HTA, $TreHTA$, DLP, $TreDLP$. In addition, when the Mann-Whitney U test was applied, relationships between mortality and the following continuous variables: age, BMI, SBP, DBP, TC, HDL, LDL were shown in the next step. Dimensionality reduction techniques, PCA and clustering techniques were applied to create possible groups in the data although none of these techniques produced valid additional knowledge. In addition, classification linear models (Logistic regression, Ridge a regularization version of Logistic Regression, Linear Discriminant Analysis, and Quadratic Discriminant Analysis) and non-linear models (Machine learning models, including ensemble methods:

Bagging and Boosting) were applied. A hyper tuning parameter protocol was established, to find the best possible model, by comparing two different libraries, *Ski-learn* and *Optuna* which showed significant improvements, especially in models without sampling techniques. The best four performing model for the test dataset were: XGBoost, SVM, LightGBM and AdaBoost based on the performed experiments. Their measures are illustrated in Figure 4.1.

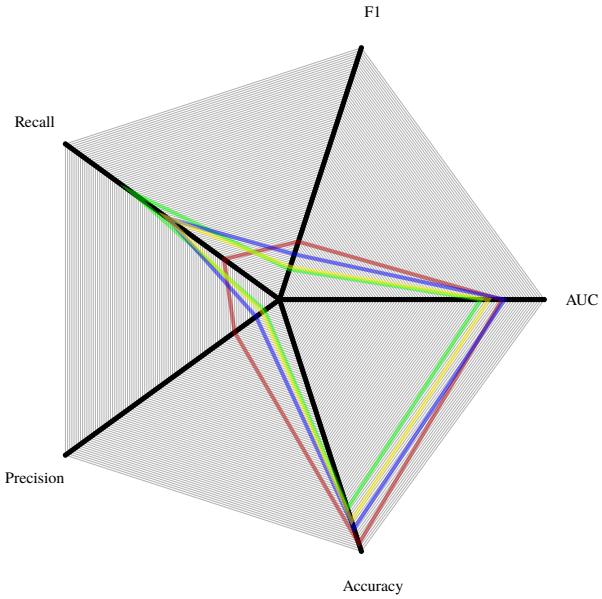


Figure 4.1 – Spider diagram of the Top 3 models scoring in unseen prediction data. XGBoost represented in red, LightGBM in green and QDA in violet.

These results reaffirm the conclusions provided by the literature [5] that situated boosting and SVM are the best performing models for CVDs. This work takes advantage of Interpretability libraries (SHAP) to get knowledge from the best performing models and features. Age, HTA treatment, gender, and DLP treatment are the four more important features as shown in this thesis's experiments. The proposed experimental setup focused on producing different approaches for balancing classes and models for mortality prediction with fairly comparable results. The study includes fifteen classifiers including boosting, SVM, DTs, and linear models and three different methods for balancing classes to improve the performance of the classification models.

4.2 Limitations

The main limitation of the study and improvement opportunity, is the low Precision and Recall obtained in the validation dataset. Although Recall is more relevant than Precision in this framework, it is essential to have a high precision to avoid producing false risk alerts. It has been assumed that the selected models cover other alternative models like Neural Networks. However, the test of alternative models could be a good candidate for further work. This research was limited to the usage of the proposed tabular data due to limited domain knowledge assuming that the proposed variables contain the most relevant features. Also there is a strong limitation found by the scarcity of minority class observation numbers, mortality due to CVDs. A full data analysis combined with a multi modal approach using additional media types like Electrocardiograms or scanner images could produce interesting results.

4.3 Future work

Many different models, techniques, and experiments have been left for the experiments with the full dataset provided are usually very time consuming, requiring days to finish a single experiment. Some activities which were not implemented in this research and could potentially bring additional insights are:

1. An evaluation of the effects of the Usage of regularization techniques in models trained with sampling techniques. Those models show improvements in Recall and Precision, but a low F1 since they intent to overfitting. Regularization techniques could improve model generalization.

2. In this work, bagging and boosting techniques have shown positive results, since three out of the four best performing models are using these techniques. However, another ensemble method that has not been tested and could bring significant improvements is stacking methods. This is the combination of several heterogeneous models, i.e. stacking together the top three best performing models.

3. The approach of this research for the prediction of mortality has been the application of supervised binary classification models, however due to the extreme imbalance of the mortality class, it could be possible to

treat this class as an anomaly and use unsupervised or semi-unsupervised anomaly detection models.

4. This work has proposed using unsupervised clustering through the application of KNN, DBSCAN and agglomerative hierarchical clusters to identify potential non-defined structures of data. No further research has been done because the mortality proportion did not show differences in the discovered clusters. However the application of classification models per local clusters could produce interesting results.

References

[1] W. H. Organization, "Cardiovascular diseases (cvds)," https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds), 2017, [Online: accessed 31-8-2021].

[2] N. Poulter, "Coronary heart disease is a multifactorial disease," *American Journal of Hypertension*, vol. 12, no. S6, pp. 92S–95S, 10 1999. doi: 10.1016/S0895-7061(99)00163-6. [Online]. Available: https://doi.org/10.1016/S0895-7061(99)00163-6

[3] J. Navarro-Pérez, D. Orozco-Beltran, V. Gil-Guillen, F. Pallares, V.and Valls, A. Fernandez, A. M. Perez-Navarro, C. Sanchis, A. Dominguez-Lucas, J. M. Martin-Moreno, J. Redon, and M. Tellez-Plaza, "Mortality and cardiovascular disease burden of uncontrolled diabetes in a registry-based cohort: the escarval-risk study," in *BMC cardiovascular disorders*, vol. 18(1), no. 180, 2018. doi: 10.1186/s12872-018-0914-1

[4] PubMed, https://pubmed.ncbi.nlm.nih.gov/, 2021, [Online: accessed 31-8-2021].

[5] C. Krittanawong, H. U. Virk, S. Bangalore, Z. Wang, W. J. Kipp, R. Pinotti, H. Zhang, S. Kaplin, B. Narasimhan, T. Kitai, U. Baber, J. L. Halperin, and W. H. W. Tang, "Machine learning prediction in cardiovascular diseases: a meta-analysis," *Scientific Reports*, 2020. doi: 10.1038/s41598-020-72685-1. [Online]. Available: https://doi.org/10.1038/s41598-020-72685-1

[6] E. Soria-Olivas, J. D. Martín-Guerrero, J. Redón, M. Tellez-Plaza, and J. Vila-Francés, "Improving mortality prediction in cardiovascular risk patients by balancing classes," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015. doi: 10.1109/ICDMW.2015.76 pp. 480–484.

[7] F. Mateo, E. Soria-Olivas, M. Martínez-Sober, M. Tellez-Plaza, J. Gómez-Sanchís, and J. Redon, "Multi-step strategy for mortality assessment in cardiovascular risk patients with imbalanced data," in *24th European Symposium on Artificial Neural Networks, ESANN 2016, Bruges, Belgium, April 27-29, 2016*, 2016. [Online]. Available: http://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-60.pdf

[8] R. C. Deo, "Machine learning in medicine," *Circulation, PMC*, vol. 20, pp. 1920–1930, 2018.

[9] A. Subasi, "Practical machine learning for data analysis using python," in *Practical Machine Learning for Data Analysis Using Python*. Academic Press, 2020. ISBN 978-0-12-821379-7 pp. 8–13.

[10] R. Soni and K. Mathai, "Improved twitter sentiment prediction through cluster-then-predict model," *ArXiv*, vol. abs/1509.02437, 2015.

[11] S. Kaufman, S. Rosset, and C. Perlich, "Leakage in data mining: Formulation, detection, and avoidance," vol. 6, 01 2011. doi: 10.1145/2020408.2020496 pp. 556–563.

[12] PyCaret, https://pycaret.org/, 2021, [Online: accessed 31-8-2021].

[13] t. PyCaret, "Pycaret, transformations," https://pycaret.org/transformation/, 2021, [Online: accessed 31-8-2021].

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," in *Artificial Intelligence (cs.AI)*, vol. 16, 2002. doi: 10.1613/jair.953 pp. 321–357.

[15] J. Brownlee, "Smote for imbalanced classification with python," https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/, 2021, [Online: accessed 31-8-2021].

[16] Numpy, https://numpy.org/, 2021, [Online: accessed 31-8-2021].

[17] Pandas, "Pandas," https://pandas.pydata.org/, 2017, [Online: accessed 31-8-2021].

[18] SciPy, https://www.scipy.org/, 2021, [Online: accessed 31-8-2021].

[19] V. w. P. Matplotlib, https://matplotlib.org/, 2021, [Online: accessed 31-8-2021].

[20] s. d. v. Seaborn, https://seaborn.pydata.org/, 2021, [Online: accessed 31-8-2021].

[21] M. L. in Python, "Machine learning in python," https://scikit-learn.org/stable/, 2021, [Online: accessed 31-8-2021].

[22] S. Weisberg, *Applied Linear Regression*. John Wiley & Sons, Inc., 2005. ISBN 9780471704096

[23] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques; Adaptive computation and machine learning*. MIT Press, 2009. ISBN 9780262013192

[24] M. Sánchez-Montañés, P. Rodríguez-Belenguer, A. J. Serrano-López, E. Soria-Olivas, and Y. Alakhdar-Mohmara, "Machine learning for mortality analysis in patients with covid-19," *Internatinal Journal of Environmental Research and Public Health*, vol. 17, 2020.

[25] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction To Statistical Learning*. Springer, 2013. ISBN 9781461471387

[26] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[27] T. Afonja, "Accuracy paradox," https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b, 2017, [Online: accessed 31-8-2021].

[28] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," vol. 41, 2014. doi: 10.1007/s10115-013-0679-x p. 647–665.

[29] M. L. McHugh, "The chi-square test of independence," *Biochemia Medica*, vol. 23, pp. 143–149, 2013.

[30] A. Schneider, G. Hommel, and M. Blettner, "Linear regression analysis," *Evaluation of Scientific Publications*, vol. 107, pp. 776—-782, 2010.

Appendix A

Results overview

Table A.1 – XGBoost Results through different experiments in test data.

Activity	Model	F1	Recall	Prec.	Accuracy	AUC
Baseline	XGBoost	0.2328	0.2639	0.2082	0.9695	0.8408
Undersampling	XGBoost	0.0846	0.8090	0.0446	0.6926	0.8308
Oversampling	XGBoost	0.1079	0.0833	0.1529	0.9758	0.7672
SMOTE	XGBoost	0.0889	0.0764	0.1063	0.9725	0.7821

Table A.2 – SVM Results through different experiments in test data.

Activity	Model	F1	Recall	Prec.	Accuracy	AUC
Baseline	SVM	0.0975	0.7083	0.0524	0.7698	0.7396
Undersampling	SVM	0.1213	0.6840	0.0665	0.8260	0.7563
Oversampling	SVM	0.1024	0.7604	0.0549	0.7659	0.7632
SMOTE	SVM	0.1024	0.7604	0.0549	0.7659	0.7632

Table A.3 – LightGBM Results through different experiments in test data.

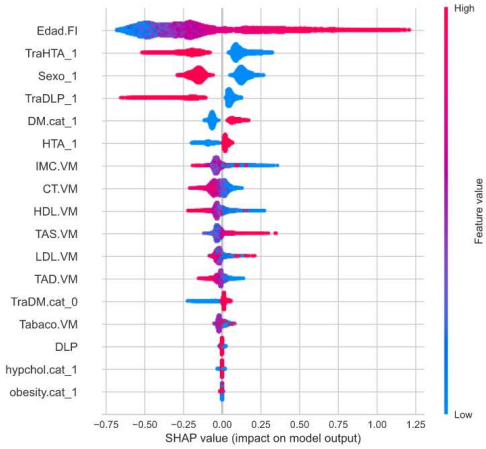
Activity	Model	F1	Recall	Prec.	Accuracy	AUC
Baseline	LightGBM	0.2144	0.4653	0.1393	0.9401	0.8405
Undersampling	LightGBM	0.0992	0.7535	0.0531	0.7599	0.8327
Oversampling	LightGBM	0.0601	0.0347	0.2222	0.9809	0.7796
SMOTE	LightGBM	0.0952	0.0660	0.1712	0.9780	0.8038

Table A.4 – AdaBoost Results through different experiments in test data.

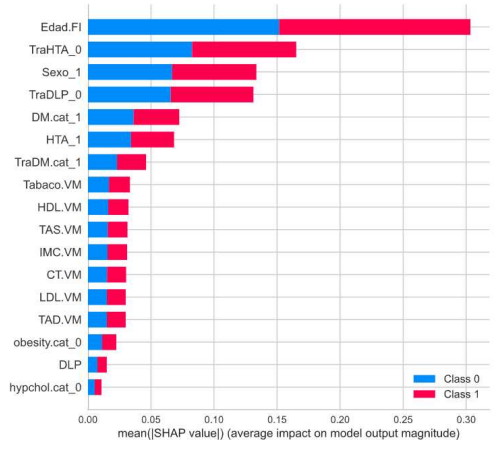
Activity	Model	F1	Recall	Prec.	Accuracy	AUC
Baseline	AdaBoost	0.0270	0.0139	0.5000	0.9824	0.8220
Undersampling	AdaBoost	0.0927	0.7361	0.0495	0.7470	0.8156
Oversampling	AdaBoost	0.1046	0.7222	0.0564	0.7829	0.8199
SMOTE	AdaBoost	0.1348	0.5278	0.0773	0.8811	0.7893

Table A.5 – XGBoost Results through their different activities.

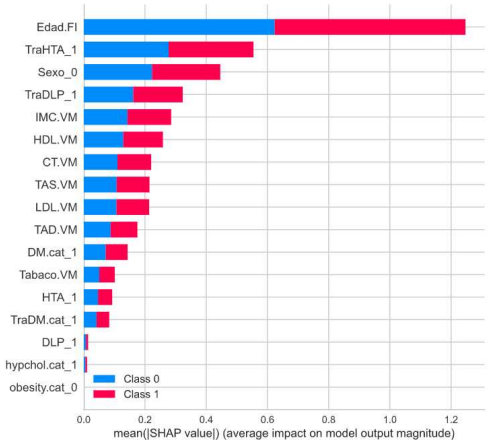
Activity	Model	F1	Recall	Prec.	Accuracy	AUC
Model Scoring	XGBoost	0.0300	0.0164	0.1798	0.9814	0.7938
Model Tuning	XGBoost	0.2163	0.2440	0.1945	0.9688	0.8363
Model Prediction	XGBoost	0.2328	0.2639	0.2082	0.9695	0.8408



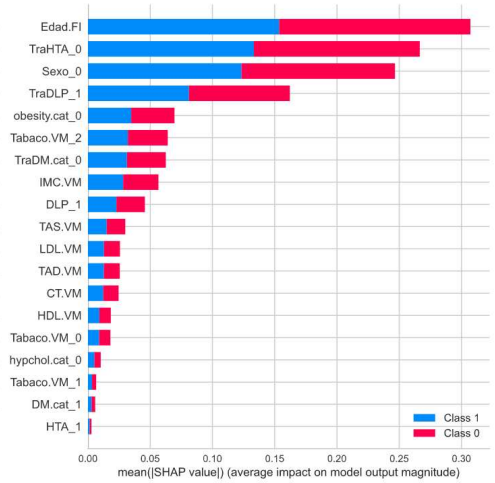
(a) XGBoost in Baseline Experiment



(b) LightGBM in Undersampling Experiment



(c) LightGBM in Oversampling Experiment



(d) RF in SMOTE Experiment

Figure A.1 – SHAP Summary Plots of the best performing model across experiments on training data

Appendix B

Other Statistical Learning Models

B.1 Linear Regression Model

This is a parametric model which assumes the expected function to be linear [30]. This assumption simplifies enormously the problem of estimating f to estimate a p-dimensional function by calculating the $p + 1$ coefficients represented by β vector.

$$Y = \beta_0 + \beta_1 X_1 + \beta_1 X_1 + ... + \beta_n X_p \tag{B.1}$$

By using a fitting model approach, loss or cost function like (ordinary) least squares (OLS) B.2 (A.k.a. residual squared sum RSS, the sum of squared residuals SSR or sum of the squared estimate of errors SSE) to estimate the β, minimizing the total Euclidean distance between a line and observations get minimized. Although linear regression is intended to predict quantitative response variables, it is possible to use linear regression and least squares for binary classification tasks by applying a dummy encoding to the response variable, using 0/1 values. However, responses outside of [0,1] interval could exist, making them difficult to be interpreted as probability estimates. Nevertheless, it turns out that the classifications obtained using linear regression to predict a binary response match with the ones using LDA [25].

$$OLS/RSS = \sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}(x_{ij}\beta_j))^2 \tag{B.2}$$

B.2 Regularization or Penalties

Regularization add a penalty term to the cost function B.3. For example, by adding an intended bias that penalizes the fitting of the model, making it less flexible, but reducing RSS variances by shrinking parameters and making prediction values less sensitive to them.

1. L1: Lasso

   In Lasso, the penalty term is defined as in Equation B.3. The penalty is modulated by parameterλ. In its minimum value, $\lambda = 0$ producing no effect leaving the original cost function. As $\lambda -> \infty$ Increasing the values of λ, the slope gets flattered, being the response less sensitive to the parameters.

   $$L1 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{B.3}$$

2. L2: Ridge

   Ridge also uses the parameter λ to adjust the sensitivity of the coefficients, and its penalty equation is almost identical to Lasso, but using the product of λ with the absolute value of the coefficients instead of their squares.

   $$L2 = RSS + \lambda \sum_{j=1}^{p} |\beta_j| \tag{B.4}$$

3. Elastic-net

   Elastic-net is the combination of the previous two. So, for example, using λ_1 and λ_2 it combines the effect of both regularizations, l1 and l2, normally applying to datasets with high dimensionality.

   $$Elastic - net = RSS + \lambda \sum_{j=1}^{p} |\beta_j| + \lambda \sum_{j=1}^{p} |\beta_j| \tag{B.5}$$

   Note: In *SciKit-learn* [21], the parameter passed is C, define as $C = \frac{1}{\lambda}$ and l1_ratio in elastic-net.

Appendix C

Results, Statistical Test

Table C.1 – χ^2 test of numerical variables vs. $mortality$

Variable/$mortality$	Statistic	P-value	Cramer's ϕ
$obesity$	1.7759	**0.1827**	0.0057
Sexo	142.7651	0.0000	0.0511
$smoking$	29.9081	0.0000	0.0234
DM	124.3360	0.0000	0.0477
$TreDM$	22.4861	0.0000	0.0203
HTA	91.3539	0.0000	0.0409
$TreHTA$	158.7569	0.0000	0.0539
$hypchol$	65.1315	0.0000	0.0345
DLP	47.6682	0.0000	0.0295
$TreDLP$	129.4639	0.0000	0.0487

To evaluate the Accuracy of a binary classification model, a prediction score of the output variable is compared with a boundary Threshold, so when it is lower, it predicts class 0 or 1 when it is higher. Binary Cross-Entropy represented in Equation C.1is used for many of the classifiers described before and applied in this project.

$$H_p(q) = -\frac{1}{n}\sum_{i=1}^{n} y_i . \log(p(y_i)) + (1 - y_i) . \log(1 - p(y_i)) \tag{C.1}$$

where:
y is the output variable, 1or 0.

Table C.2 – P-values. of χ^2 test in categorical variables

Variable	*obesity*	*smoking*	*DM*	*TreDM*	*HTA*	*TreHTA*	*hypchol*	*TreDLP*
smoking	0.000							
DM	0.000	0.000						
TreDM	0.000	0.000	0.000					
HTA	0.000	0.000	0.000	0.000				
TreHTA	0.000	0.000	0.000	0.000	0.000			
hypchol	**0.431**	**0.335**	0.002	**0.795**	0.000	0.002		
TreDLP	0.000	0.000	0.000	0.000	0.000	**0.094**	0.000	
Sexo	0.000	0.000	0.000	0.000	0.000	**0.094**	0.000	**0.108**
obesity	0.000	0.000	0.000	0.000	0.000	0.000	**0.431**	0.000

p(y) is the probability of y = 1.
log(p(y)) the log probability of being 1.
log(1-p(y)) the log probability of being 0.

C.1 Results, First Experiment

Table C.3 – NB

Parameters	**Values**
priors	None
var_smoothing	1e-09

CatBoost
Parameters of the first experiment

Table C.4 – Discriminant Analysis

Parameters	**LDAt**	**Quadratic Discriminant**
n_components	None	
priors	None	None
reg_param		0.0
shrinkage	None	
solver	'svd'	
store_covariance	False	False
tol	0.0001	0.0001

Table C.5 – SVM

Parameters	**Values**
alpha	0.0001
average	False
class_weight	None
early_stopping	False
epsilon	0.1
eta0	0.001
fit_intercept	True
l1_ratio	0.15
learning_rate	'optimal'
loss	'hinge'
max_iter	1000
n_iter_no_change	5
n_jobs	-1
penalty	'l2'
power_t	0.5
random_state	123
shuffle	True
tol	0.001
validation_fraction	0.1
verbose	0
warm_start	False

Table C.6 – XGBoost

Parameters	**Values**
base_score	0.5
booster	'gbtree'
colsample_bylevel	1
colsample_bynode	1
colsample_bytree	1
gamma	0
gpu_id	-1
importance_type	'gain'
interaction_constraints	''
learning_rate	0.300000012
max_delta_step	0
max_depth	6,
min_child_weight	1
missing	nan
monotone_constraints	'()'
n_estimators	100
n_jobs	-1
num_parallel_tree	1
objective	'binary:logistic'
random_state	123
reg_alpha	0
reg_lambda	1
scale_pos_weight	1
subsample	1
tree_method	'auto'
validate_parameters	1
verbosity	0

Table C.7 – DT algorithoms

Parameters	**XGBoost**	**ET**	**RF**	**DT**
bootstrap		False	True	
ccp_alpha	0.0	0.0	0.0	0.0
class_weight		None	None	None
criterion	'friedman_mse'	'gini'	'gini'	'gini'
init	None			
learning_rate	0.1			
loss	'deviance'			
max_depth	3	None	None	None
max_features	None	'auto'	'auto'	None
max_leaf_nodes	None	None	None	None
max_samples		None	None	
min_impurity_decrease	0.0	0.0	0.0	0.0
min_impurity_split	None	None	None	None
min_samples_leaf	1	1	1	1
min_samples_split	2	2	2	2
min_weight_fraction_leaf	0.0	0.0	0.0	0.0
n_estimators	100	100	100	
n_iter_no_change	None			
n_jobs		-1	-1	
oob_score		False	False	
presort	'deprecated'			'deprecated'
random_state	123	123	123	123
subsample	1.0			
tol	0.0001			
validation_fraction	0.1			
verbose	0	0	0	
warm_start	False	False	False	
splitter				'best'

Table C.8 – LightGBM

Parameters	**Values**
boosting_type	'gbdt'
class_weight	None
colsample_bytree	1.0
importance_type	'split'
learning_rate	0.1
max_depth	-1
min_child_samples	20
min_child_weight	0.001
min_split_gain	0.0
n_estimators	100
n_jobs	-1
num_leaves	31
objective	None
random_state	123
reg_alpha	0.0
reg_lambda	0.0
silent	True
subsample	1.0
subsample_for_bin	200000
subsample_freq	0

Table C.9 – Ada Boost

Parameters	**Values**
algorithm	'SAMME.R'
base_estimator	None
learning_rate	1.0
n_estimators	50
random_state	123

Table C.10 – KNeighbors

Parameters	**Values**
algorithm	'auto'
leaf_size	30
metric	'minkowski'
metric_params	None
n_jobs	-1
n_neighbors	5
p	2
weights	'uniform'

Table C.11 – LR

Parameters	**Values**
C	1.0
class_weight	None
dual	False
fit_intercept	True
intercept_scaling	1
l1_ratio	None
max_iter	1000
multi_class	'auto'
n_jobs	None
penalty	'l2'
random_state	123
solver	'lbfgs'
tol	0.0001
verbose	0
warm_start	False

Table C.12 – Ridge

Parameters	**Values**
alpha	1.0
class_weight	None
copy_X	True
fit_intercept	True,
max_iter	None
normalize	False
random_state	123
solver	'auto'
tol	0.001

Table C.13 – XGBoost parameters 1st experiment

XGBoost	Parameters
objective	binary:logistic
base_score	0.5
booster	gbtree
colsample_bylevel	1
colsample_bynode	1
colsample_bytree	0.7782209230420265
gamma	0
gpu_id	-1
importance_type	gain
interaction_constraints	
learning_rate	0.007437023541483795
max_delta_step	0
max_depth	5
min_child_weight	2
missing	nan
monotone_constraints	()
n_estimators	174
n_jobs	-1
num_parallel_tree	1
random_state	4277
reg_alpha	2.8640166419668526e-05
reg_lambda	5.262116255851336e-05
scale_pos_weight	16.424966424967035
subsample	0.2894780426312537
tree_method	auto
validate_parameters	1
verbosity	0

For DIVA

{
"Author1": { "name": "Juan Aldamiz"},
"Degree": {"Educational program": "Master's Programme, Information and Communication Technology"},
"Title": {
"Main title": "A Machine Learning Approach to the analysis of mortality in patients with cardiovascular diseases",
"Language": "eng" },
"Alternative title": {
"Main title": "",
"Language": "swe"
},
"External Supervisor": { "name": "Emilio Soria Olivas" },
"KTH Supervisor": { "name": "Hao Hu1" },
"Examiner": {
"name": "Magnus Boman",
"organisation": {"L1": "School of Electrical Engineering and Computer Science" }
},
"Other information": {
"Year": "2021", "Number of pages": "x,93"}
}

TRITA-EECS-EX-2021:710