Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

Prognosis using Deep Learning in CoViD-19 patients

A thesis presented by

José Luis Guadiana Álvarez

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

In

Manufacturing Systems

Monterrey, Nuevo León, 30 th November, 2020

Dedication

A mi familia, que siempre me brindaron consejos y su apoyo incondicional cuando más lo necesitaba.
A mi padre, que es mi ejemplo a seguir desde que tengo uso de memoria. Espero que esto me haga estar más cerca de ser una fracción del hombre que tú eres.
A mi madre, que siempre estuvo al pendiente de mi salud y me llenó de fuerzas para sobrellevar cualquier obstáculo y reto.
A mi hermana, que siempre me brindó su oído para escuchar mis problemas y me alentó a seguir adelante.
A mis amigos, que me apoyaron en todo momento y me ayudaron a distraerme y a no perder mi vida social.

Acknowledgements

I would like to express my deepest gratitude to my thesis advisor, Dr. Rubén Morales Menéndez and co-advisor M.Sc. Etna Aurora Rojas Flores for their important suggestions, guidance and patience. I also want to thank the committee members Dra. Adriana Vargas Martínez, and Dr. Ricardo Ambrocio Ramírez Mendoza for their support and their valuable contributions and suggestions. Their support has been very important to achieve the goals of this work.

I thank Tecnológico de Monterrey and the Automotive Consortium for giving me a scholarship during my studies and CONACyT for the financial support provided.

Prognosis using Deep Learning in CoViD-19 patients

By
José Luis Guadiana Álvarez

Abstract

Prognostics study the prediction of an event before it happens, to enable efficient critical decision making. Over the past few years, it has gained a lot of research attention in many fields, i.e. manufacture, economics, and medicine. Particularly in medicine, prognostics are very useful for front line physicians to predict how a disease may affect a patient and react accordingly to save as many lives as possible. One clear example is the recently discovered *Coronavirus Disease 2019 (CoViD-19)*.

Because of its novelty, not nearly enough is known about the virus' behaviour and *Key Performance Indicators (KPIs)* to asses a mortality prediction. However, using a lot of complex and expensive medical biomarkers could be impossible for many low budget hospitals. This motivates the development of a prediction model that not only maximizes performance, but does so using the least amount of biomarkers possible. For mortality risk prediction, falsely assuming that a patient has a low mortality risk is far more critical than the opposite. Therefore, false negative predictions should be prioritized over false positive ones.

This research project proposes a *CoViD-19* mortality risk calculator based on a *Deep Learning* model trained on a data set provided by the *HM Hospitales* from Madrid, Spain. A pre-processing strategy for unbalanced classes and feature selection is proposed. Benefit of using over-sampling and imputation techniques is evaluated. Also, an imputation method based on the *K-Nearest Neighbor (KNN)* algorithm for biomarker data is is proposed and its efficiency is evaluated. Results are compared against a *Random Forest (RF)* model while showing the trade-off between feature input space and the number of samples available. Results on the *MPCD* score show the proposed *DL* outperforms the proposed *RF* on every data set when evaluating even with an over-sampling technique. Finally, the proposed *KNN* method proves beneficial for data imputation, improving the model's *Recall* score from 0.87 to 0.90.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Prognostics is defined as "Predict the progression of an event based on current and future operational and environmental conditions to estimate the time at which a system no longer fulfils its function within desired specifications, i.e. *Remaining Useful Life (RUL)* [Goebel *et al.*, 2012]. To predict is to know of the occurrence of an event, most of the times defined as a fault, on a given system before its parameters go out of specification. To predict an event is very useful and it is a topic of interest in many different areas:

- Medicine: Patient's accurate and early diagnosis and prognosis of diseases, X-ray and CT-Scan image interpretation [Panwar *et al.*, 2020b]
- Manufacture: Downtime reduction in production lines by making machine health diagnosis and prognosis for optimal maintenance.
- Finance: Financial credit risk prediction [Ma and Lv, 2019], Stock market price prediction [Ferdiansyah *et al.*, 2019] and Risk of bankruptcy prediction.

Predictive analytics (prognosis) is one of the main research topics nowadays, forecasting a 20% to 50% market penetration in less than 2 years, according to Gartner Inc., [S. *et al.*, 2020]. Fig. 1.1 shows the evolution of expectations for new technologies through time. At the begging of any new technology, the expectations for its productivity applications grow exponentially (*Innovation Trigger*) until it reaches a peak of hype (*Peak of Inflated Expectation*). However, most of the expected applications are proven inefficient or impossible, so the hype around said technology is lowered (*Through of Disillusionment*). Given time, real productive applications for problem solving will be materialized (*Slope of Enlightenment*) until the technology reaches its limits for new applications and stabilizes (*Plateau of Productivity*).

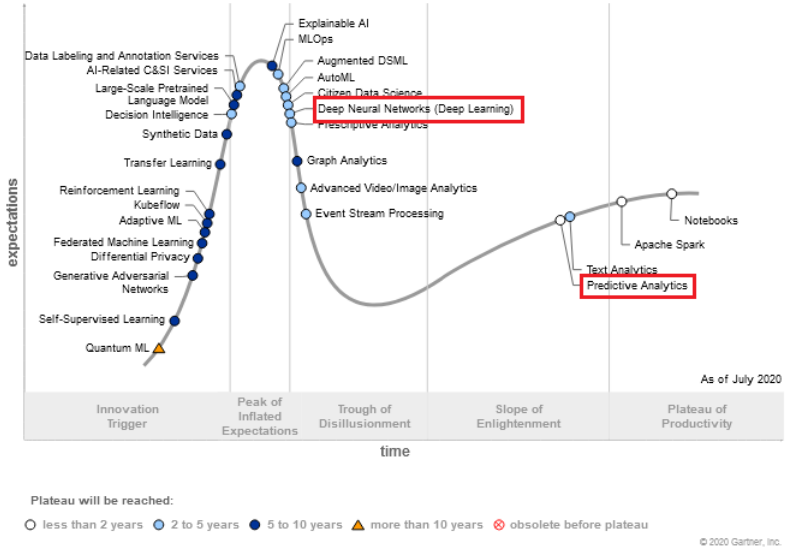

Figure 1.1: Hype Cycle for Data Science and Machine Learning, 2020

1.1 Motivation

In the medical field, the ability to correctly predict the evolution of a disease in patients enables doctors and hospitals to make critical treatment decisions earlier, which ultimately saves more lives. One clear example is the recently discovered *Coronavirus Disease 2019 (CoViD-19)*.

CoViD-19 is a respiratory disease caused by the *Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2)* [Wu *et al.*, 2020] which was declared as a global pandemic on March 11, 2020 by the [Cucinotta and Vanelli, 2020]. Up to date, more than 62 million people have been infected and more than 1.49 million have died of *CoViD-19* [The Johns Hopkins University, 2020]. Most cities around the world have implemented isolation and social distancing as a preventive measure, which has greatly impacted their economy, and will continue to do so, by completely stopping most ”non-essential” economical activities. According to [Fernandes, 2020] *Gross Domestic Product (GDP)* growth would take a hit ranging from 3% - 5% depending on the country. In other scenarios, *GDP* can fall as much as 10%. On average, each additional month of crisis costs 2% - 2.5% of global *GDP*. The need for an efficient model to accurately distinguish critical patients from others, becomes clear.

1.2 Problem Description

In Mexico, there have been 1,113,543 confirmed infected people and 105,940 deceases due to *CoViD-19* (up until December 1^{st}, 2020) [CONACYT and DataLab, 2020]. For *CoViD-19*, identifying severe patients at their acute phase is clinically trivial. Early identification of those at risk to quickly deteriorate, and those with very low risk for critical disease, are the measures required

for efficient triage [Assaf *et al.*, 2020]. Most of the times it is not the fact that the disease itself is very life threatening, but rather the lack of proper critical patient management. Because of the high reproduction number of the *SARS-CoV-2* virus, studies estimated a median R_0 value of 5.7 [Read, 2020], the amount of infected patients grows in an exponential manner and hospitals cannot keep up with the sudden amount of people who need critical attention, facing shortages of beds, mechanical ventilators and medical drugs. The reproduction number R_0 of an infection can be thought of as the expected number of cases directly generated by one case in a population where all individuals are susceptible to infection [Fraser *et al.*, 2009].

Moreover, not every hospital has the resources, i.e. budget, time, staff, equipment, etc., to conduct many complicated tests before needing to make a decision. A mortality risk calculator for *CoViD-19* must not only be as accurate as possible, but also use only the minimum amount of features in order to produce an acceptable prediction. That means, the trade-off between the number of input variables and model performance must be practically optimized.

1.3 Research Question

Because of its novelty, there is still much we don't know about the *CoViD-19* disease and not nearly as much data as we would want to make proper *data-driven* models. Even though there are already a lot of *Machine Learning (ML)* algorithms for *CoViD-19* prediction, most of them have not reached optimal results, because of the lack of useful data, or because they are highly biased to only a certain population. The lack of an accurate model to predict risk of mortality in *CoViD-19* patients is evident and would be greatly beneficial for hospitals to properly manage critical patients (efficient triage). This research project attempts to answer the following questions:

- Which are the basic features needed for an acceptable mortality risk prediction *DL* model?
- How does the prediction performance of *DL* models change when using basic against specialized features?
- How does the *DL* model compare against other *ML* algorithms?
- How does oversampling and data augmentation techniques affect the *DL* model?

1.4 Solution Overview

Develop a *Deep Learning (DL)* model to predict the mortality risk of *CoViD-19* patients. Feature selection and pre-processing, as well as hyper-parameter tuning, is done to optimize the algorithm's

convergence. The model is a *Feed-Forward Neural Network (FFNN)* with one-dimensional features, where sequential data is transformed into stationary data by getting its statistical attributes (maximum and minimum values).

1.5 Main Contribution

The main contribution of this research project is a *DL* model that can be used in a mortality risk prediction web application. The application was developed through *Amazon Web Services (AWS)* and is intended to help frontline physicians in clinical decision making under time-sensitive and resource-constrained conditions for *CoViD-19* patients. The proposed model achieves an average *Maximum Probability of Correct Decision (MPCD)* score of 0.84 using a 10-fold *Cross-validation (CV)* evaluation method on a data set using 26 input features., which represents a 10% improvement when compared against other a *Random Forest (RF)* algorithm.

1.6 Organization

This research work is organized as follows:

- Chapter 2 presents the state of the art methods and approaches for Prognosis using *DL*, along with a basic theoretical background.
- Chapter 3 describes the experimental database, the description of the used database and the structure of the *Design of Experiments (DoE)* used.
- Chapter 4 describes the proposed methodology
- Chapter 5 presents the main results.
- Chapter 6 presents the conclusion, the contributions and future work of this research.
- Bibliography
- Appendices

Chapter 2

State of the Art

First, the necessary theoretical background information to comprehend the methodology and performance metrics used in this research project are described. Then, a literature review of relevant papers in the prognosis of *CoViD-19* is presented. Table 2.1 shows a summary of the literature review. Finally, an explanation of key concepts such as *ML*, *DL*, *Artificial Neural Networks (ANN)* and different feature selection techniques is presented.

2.1 Theoretical Background

Over the years, many different fault prognosis methods have been introduced. However, [Pattipati *et al.*, 2009] states that they are all framed within the same three categories: *Model-based*, *Knowledge-based*, and *Data-driven based* methods. *Model-based* approaches require an accurate mathematical model to be developed and use residuals as features, where residuals are the outcomes of consistency checks between the sensed measurements of a real system and the outputs of a mathematical model, [Vachtsevanos, 2006]. *Knowledge-based* approaches use symbolic representations to solve problems, which can be very difficult when dealing with complex systems, time consuming, and completely rely on the availability of expert knowledge. *Data-driven based* approaches are used when monitoring systems have big amounts of historical data available, [Pattipati *et al.*, 2009].

In recent times, *data-driven* methods have produced state of the art results in many research fields, mainly due to an increase in computational power, and both quantity and quality of data. New technological trends, i.e. *Industry 4.0 (I4.0)*, encourage companies and systems to have access to a lot of real-time process variables to analyze and support strategic decision making. The concept of *I4.0* was born to enable industrial processes to move from a physical process with IT support, to an integrated cyber-physical system of production [Henning, 2013]. However, it has quickly being adopted by different areas to use *data-driven* approaches for complex problem solv-

ing without needing expert knowledge. For example, *data-driven* methods allow hospitals and medical staff to not rely on specialized doctors to correctly diagnose and treat certain diseases using different patient's attributes or even images, i.e. X-ray or CT-Scans. As the world moves towards a new era in which huge amounts of data are available, known as *Big Data*, *Data-driven methods* such as *AI*, *ML* and *DL*, have gotten a lot of attention.

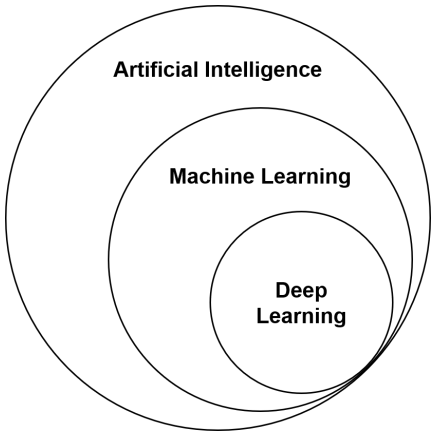

Figure 2.1: Representation of *DL* inside of *AI*

AI is a subset of computer science, concerned with how to give computers the sophistication to act intelligently [Nilsson, 2014]. Its goal is to enable computers to do complicated abstract tasks that usually require human intervention. *ML* is a field from *AI* which is focused on developing algorithms that enable computers and systems to imitate the human learning process by extracting useful knowledge and making appropriate decisions from big data [Zhao *et al.*, 2019]. The learning process can either be supervised or un-supervised. In Supervised learning the user gives the system the outcome it expects for each data sample. In un-supervised learning the model must first generate an outcome from the given data samples and then adjust itself to learn from these self-generated labels.

2.2 Literature Review

Prognosis has been a very popular research topic in recent years. In the medical field, [Jiang *et al.*, 2020] developed a risk of developing *Acute Respiratory Distress Syndrome (ARDS)* for *CoViD-19* patients predictive model. The database consisted in 53 confirmed *CoViD-19* patients from the Wenzhou Central Hospital and Cangnan People's Hospital in Wenzhou, China. The median age was 43 years and 62.2% were men. Common symptoms included fever (in 47 patients, 88.7%) and cough (in 32, 60.4%). Median days from symptom onset to hospitalization was 3 days, no patients presented more than once and none were pregnant. Various lab tests were taken into account, although these varied between hospital according to availability. Median white blood

cell count ($x10^9/L$) was reported to be 4.8, and median Lymphocyte count ($x10^9/L$) was of 1.2. Out of the 53 patients 9.4% developed ARDS, 1.9% was taken into the *Intensive Care Unit (ICU)* and 47.5% required supplemental oxygen. All 53 patients took lopinavir and litonavir tablets; dose was 200 mg twice a day of lopinavir. 29 patients took umifenovir. 43 patients took rectal suppositories of recombinant human interferon-2a. All 53 patients have now been discharged. The median length of stay was 27 days (interquartile range 23-31.5, 9-45). Discharge required normal temperature for over three days, no respiratory and gastrointestinal symptoms, *Polymerase Chain Reaction (PCR)* swab negative twice over at least 2 days, and *PCR* stool sample negative as well. They used filter and wrapper methods for feature selection. For the filter methods they adopted entropy, information gain, Gini index and Chi-Squared statistics. Finally, a greedy forward selection algorithm was implemented to leave only the most relevant features. The final list of features were: ALT, Myalgias, Hemoglobin, Gender, Temperature, Na+, K+, Lymphocyte count, Creatinine, Age and White blood cell count. They compared various *ML* algorithms using a 10-fold cross validation accuracy. The top accuracy achieved was 80% from a *Support Vector Machine (SVM)* and a *K-Nearest Neighbor (KNN)* (k=5) algorithm. Authors contemplate the idea of trying out *DL* methods next, once they have the necessary amount of sample data.

Authors in [Pourhomayoun and Shakibi, 2020] proposed a *ML* algorithm to accurately predict the mortality risk of *CoViD-19* patients. They used a dataset with more than 117,000 laboratory-confirmed *CoViD-19* patients from 76 countries with a average age of 56.6, from which 74.4% recovered. Data imputation techniques were used for missing values and a balanced dataset was created for training and testing the model. 112 features were available from symptoms and doctor's medical notes, and patient's demographic and physiological data. After applying different filter and wrapper methods, the feature space was reduced to 42 features. The tested *ML* algorithms included: *SVM*, *ANNs*, *RF*, *Decision Tree*, *Logistic Regression*, and *KNN*. The evaluation technique was 10-fold cross-validation and the performance metrics were accuracy, *Area Under the Curve of Receiver Operating Characteristics curve (AUC-ROC)* and *Confusion Matrix (CM)*. The best performance accuracy achieved was 93.75% by the *ANN* algorithm. Hyper-parameters were tuned using grid search and the final architecture had two hidden layers with 10 neurons in the first layer and 3 neurons in the second layer. Sigmoid function was used as the hidden layer activation function and stochastic gradient as the optimizer with constant learning rate and a regularization rate of alpha = 0.01.

In [Khan *et al.*, 2020] a *Gaussian Process Regression (GPR)* model with optimized hyper-parameters is used to predict the mortality rate in five different countries (Turkey, Spain, Sweden, France and Pakistan). Its results were compared against a polynomial regression model using the Root Mean Square Error as a performance metric. The data was obtained from [Max Roser and Hasell, 2020] using data from March 21, 2020 to May 10, 2020, which represent 51 days. 36 days were used for training and 15 days for testing. It was shown that, even though the *GPR* model

takes longer to train, mainly because of the hyper-parameter optimization, it outperformed the Polynomial Regressor in every country.

In [Burdick *et al.*, 2020] an *Extreme Gradient Boosting (XGBoost)* Classifier is used to model the probability of requiring mechanical ventilation within the next 24 hours, using data from the first 2 hours after admission. The model is compared against the *Modified Early Warning Score (MEWS)* score, which is commonly used to identify likely patient deterioration and mortality, using sensitivity, specificity, positive likelihood ratio, negative likelihood ratio, and diagnostic odds ratio as performance metrics. Data for 197 patients with confirmed *CoViD-19* was obtained from five US health systems, being 51.3% male and the majority being between 30 and 80 years old (73%). For each patient, 12 features were used as input to the model. These included: diastolic blood pressure (DBP), systolic blood pressure (SBP), heart rate (HR), temperature, respiratory rate (RR), *Saturation of Peripheral Oxygen (SpO2)*, white blood cell (WBC), platelet count, lactate, blood urea nitrogen (BUN), creatinine, and bilirubin. Missing values were left as empty holders, which are valid inputs to the model because each node has a default direction that will be used in the event that the feature in that node is missing. Results showed that the *XGBoost* Classifier outperformed *MEWS* both in every metric, specifically sensitivity (0.90) and specificity (0.58), meaning that it is capable of detecting 16% more patients at risk while simultaneously reducing false positive alerts.

[Kim *et al.*, 2019] attempted to predict the occurrence of *Major Adverse Cardiac Events (MACE)* in *Acute Myocardial Infarction (AMI)* patients, during the 1, 6 and 12 months follow-up periods after hospital admission using a *FFNN*. The used database consists of 10,813 patients from 52 Korean hospitals, using 51 variables for prediction. This method was compared against other *ML* methods such as *Gradient Boosted Machine (GBM)*, *Generalized Linear Model (GLM)* and a commonly used regression method in the medical field: *Global Registry of Acute Coronary Event (GRACE)*. The *FFNN* greatly outperformed the commonly used *GRACE* model in every evaluated metric (Accuracy=95.98%, Sensitivity=81.25%, Specificity=96.10% and *AUC*=0.97).

In [Bertsimas *et al.*, 2020], a mortality risk calculator *XGBoost* model was developed using patients from hospitals in Spain (HM Hospitals) and Italy (ASST Cremona). The study comprised 2,831 patients, 711 (25.1%) of whom died during hospitalization while the remaining ones were discharged. Vitals signs and laboratory test values from only the first day upon admission were used. Two models were trained for the mortality risk calculator: one with lab test results and one without them. Missing values were imputed using *KNN*, features missing in more than 40% were excluded and 95% confidence intervals were calculated using bootstrapping. Performance evaluation using 40 random data partitions into training and test sets was made yielding an average *AUC* of 93.8% using laboratory values and 90.5% without laboratory test values.

[Panwar *et al.*, 2020b] developed a deep neural network transfer model based on a *Convolutional Neural Network (CNN)* called *nCOVnet* that can diagnose a patient with *CoViD-19* by analyzing their lungs' X-ray images. The evaluation metrics used were: *CM* (Sensitivity, Specificity

and Accuracy) and *AUC-ROC*. The data set consisted of 284 X-ray images from which around 142 were positive for *CoViD-19*. A 70%-30% training-testing split was made for proper model evaluation. The *nCOVnet* uses the top layers of the *VGG16* image classifier as a base model and then adds 5 layers as part of the transfer learning methodology. The proposed model achieved Sensitivity of 97.62%, Specificity of 78.57%, Accuracy of 88.10% and *AUC-ROC* of 0.88.

In [Panwar *et al.*, 2020a], authors proposed a *CNN* transfer learning model to diagnose *CoViD-19* patients using X-ray and CT-Scan images. The model was evaluated using Precision, Recall, F1-Score and accuracy, along with the corresponding *CM* on three data sets:

- [Cohen *et al.*, 2020] with 285 patients and 526 images.
- [Soares *et al.*, 2020] has 1,252 positive and 1230 negative CT-Scans for *CoViD-19* patients from a hospital in Sao Paulo, Brazil.
- [Kermany *et al.*, 2018] 5,856 X-Rays images of Pneumonia and normal patients.

The architecture used for the transfer learning part was *VGG19* which was trained on the *ImageNet* database [Deng *et al.*, 2009]. Because of the importance of explainable models in the medical field, they used the *Gradient Weighted Class Activation Mapping (GRAD-CAM)* technique [Selvaraju *et al.*, 2017] for this purpose. The results obtained were: $Precision = 0.95,\ Recall = 0.94,\ F1 - Score = 0.95\ and\ Accuracy = 95\%$.

[Yadaw *et al.*, 2020] developed a mortality prediction model using a database (3,841 patients, 8.2% deceased) with patients treated at the Mount Sinai Health System in NY, USA. The features were patient's age, SpO2 and type of patient. The selected metrics was *AUC* of *ROC* (0.91) scores. The best results were obtained with the *XGBoost* algorithm.

In [Zhao *et al.*, 2020] a risk scores based on clinical characteristics at presentation to predict *ICU* admission and mortality in *CoViD-19* patients was developed. Clinical data including demographic information, chronic comorbidities, vital signs, symptoms, laboratory tests, and outcomes were considered. The database has 641 hospitalized patients with a median age of 60 years old, 40.1% female, 62% no critical illness, 30% were admitted to the *ICU*, and 82 who expired. Five significant variables predicting *ICU* admission were lactate dehydrogenase, procalcitonin, SpO2, smoking history, and LIN. Seven significant variables predicting mortality were heart failure, procalcitonin, lactate dehydrogenase, chronic obstructive pulmonary disease, SpO2, heart rate, and age. The mortality group uniquely contained cardiopulmonary variables. The risk score model (a multivariable regression model) yielded good accuracy with an *AUC-ROC* of 0.74 for predicting *ICU* admission and 0.82 for mortality.

A predictive model of *CoViD-19* disease progression is proposed by [Ji *et al.*, 2020] using Multivariate Analysis (Cox proportional regression), and a database with 208 patients. The average age was 44, 117 of 208 patients (56.2%) were male, 31 (14.9%) were older than 60 years, 45 (21.6%)

had at least one underlying comorbidity, the average hospitalization time was 17.5 days, and in 40 (19.2%) patients, the clinical conditions deteriorated progressed during the observation period. Comorbidity was defined as having at least one of the followings: hypertension, diabetes, cardiovascular disease,liver disease, asthma, chronic lung disease, HIV infections and malignancy for at least 6 months. Analyses showed Comorbidity, older Age, Lower lymphocyte count, and higher Lactate dehydrogenase (*CALL*) were independent high-risk factors for *CoViD-19* progression. A nomogram achieved good concordance (0.86) and a good *AUC-ROC* (0.91). Using *CALL* score model, clinicians can improve the therapeutic effect and reduce the mortality with more accurate and efficient use of medical resources.

Table 2.1: Comparison between *ML* models to predict CoViD-19 patients mortality risk

References	*Model*	*Input features*	*Sample size and distribution*	*Performance evaluation*
[Jiang *et al.*, 2020]	SVM and KNN	11 input features (ALT, myalgias, hemoglobin, gender, temperature, Na+, K+, LIN, Cr, age and white blood cell count).	53 CoViD-19 patients from China. The median age was 43 years, 62.2% men, common symptoms included fever 88.7 % and cough 60.4%.	$Acc = 80\%$
[Pourhomayoun and Shakibi, 2020]	ANN	42 input features	>117,000 patients from 76 countries with an average age of 56.6 years and 74.4% recovered.	$Acc = 93.75\%$
[Khan *et al.*, 2020]	GPR	1 feature (Daily mortality rate)	51 days of mortality rate data from 5 different countries. 36 training observations and 15 for testing.	$RMSE = 13.06$
[Burdick *et al.*, 2020]	XGBoost Classifier	12 features (Diastolic blood pressure, systolic blood pressure, heart rate, temperature, respiratory rate, SpO2, WBC, platelet count, lactate, blood urea nitrogen, Cr, and bilirubin).	197 patients from 5 US hospitals. 51.3% male, 30-80 years old (73%).	$Recall = 0.90$, $TPR = 0.58$
[Kim *et al.*, 2019]	ANN	51 features (21 numerical, 26 categorical and 4 discrete data).	10,813 registries from Korea, aged between 20 and 100 years old with the 1-year follow-up *MACE* after discharge.	$Acc = 95.98$, $Recall = 81.25$, $TPR = 96.10$, $AUC = 0.97$
[Bertsimas *et al.*, 2020]	XGBoost Classifier	20 features (Demographic, comorbidities and lab values).	2,831 patients from Spain and Italy (27.2% deceased).	$AUC = 93.8\%$
[Panwar *et al.*, 2020b]	CNN and Deep Transfer Learning	224x224 RGB X-ray images	284 images (142 positive for CoViD-19)	$Recall = 97.62\%$, $TPR = 78.57\%$, $Acc = 88.10\%$, $AUC - ROC = 0.88$
[Panwar *et al.*, 2020a]	CNN and Deep Transfer Learning	X-ray and CT-Scan images	Evaluated on three different datasets. (1) 285 patients and 526 images, (2) 1252 positive and 1,230 negative CT-Scans from Brazil, and (3) 5,856 X-Rays images of Pneumonia and normal patients	$Precision = 0.95$, $Recall = 0.94$, $F1 - Score = 0.95$, $Acc = 95\%$
[Yadaw *et al.*, 2020]	XGBoost Classifier	3 features (Age, minimum oxygen saturation and type of encounter)	3,841 patients (8.2% deceased and 55% male) from USA.	$AUC - ROC = 0.91$

Table 2.1: Comparison between *ML* models to predict CoViD-19 patients mortality risk (Continued)

References	*Model*	*Input features*	*Sample size and distribution*	*Performance evaluation*
[Zhao *et al.*, 2020]	Multivariate Regression model	7 features (heart failure, procalcitonin, lactate dehydrogenase, chronic obstructive pulmonary disease, SpO2, heart rate, and age)	641 patients (median age of 60 years old, 40.1% female and 82 patients died).	$AUC - ROC = 0.82$
[Ji *et al.*, 2020]	Multivariate Analysis (Cox proportional regression)	4 features (Comorbidity, age, lymphocyte count and Lactate dehydrogenase).	208 patients (Average age of 44, 56.2% male patients, 14.9% older than 60 years old, 21.6% had at least one underlying comorbidity and the average hospitalization time was 17.5 days)	$AUC = 0.91$

2.3 Deep Learning

DL is a *ML* technique that attempts to model high level representations behind data by using a multi-layer non-linear hierarchical architecture approach [Yan *et al.*, 2019]. The basic idea behind *DL* is to stack many shallow *ML* algorithms to obtain more abstract representation of features as the network gets deeper. *DL* has recently gained popularity particularly in the field of computer vision, but is rapidly moving towards different areas, such as diagnosis and prognosis in the medical field. The quintessential *DL* algorithm is the *ANN*.

ANNs are a type of *ML* algorithm roughly based on the biological neurons of the brain and the way that they are interconnected with one another to learn complex abstract representations. Each neuron is typically a simple linear regression model that applies a non-linear activation function to its output $a = g(\sum_{i=0}^{n} x_i w_i + b)$. They have a bias b and each connection has a determined weight w_i which represents the activation of that specific connection to another neuron x_i, similar to how biological neurons work through electrical impulses. *ANNs* consist of input layers, hidden layers and output layers, where each layer has a collection of neurons that are connected to other layers in a specific way, defined by the type of *ANN* architecture that is being used.

2.3.1 Feed-Forward Neural Networks (FFNN)

Also known as *Multi-Layer Perceptron (MLP)*, these networks consist of fully connected layers where every neuron in layer l is connected to every neuron in layer $l-1$ and $l+1$. They are called Feed-Forward because the flow of information is unidirectional from inputs to outputs following eqn. 2.1.

$$\mathbb{A}_l = g_l(\mathbb{W}_l * \mathbb{A}_{l-1}^T + b_l) \tag{2.1}$$

where $\mathbb{W}_l \in \mathbb{R}^{n \times m}$ is the weight matrix of the l^{th} layer with n neurons and m inputs, $b_l \in \mathbb{R}^n$ is the bias vector of layer l, $g_l(x)$ is the non-linear activation function of layer l and $\mathbb{A}_l \in \mathbb{R}^n$ is the output activation vector of n neurons in layer l. Some of the most common activation functions include: *sigmoid*, *tanh* and *ReLU*. Figure 2.2 shows the graphical representation of a *FFNN* with a 3 neuron input layer, only 1 hidden layer using 4 neurons and a 2 dimensional output layer.

2.3.2 Binary cross-entropy loss function

ANNs use loss functions to quantify a deviance between a model's prediction and the expected (true) output. It is used by the *Gradient Descent* algorithm to know how to modify the model's parameters in order to minimize it. The binary cross-entropy loss function is used for binary classification problems where $y \in \{0, 1\}$ and $\hat{y} \in [0, 1]$, eqn. 2.2.

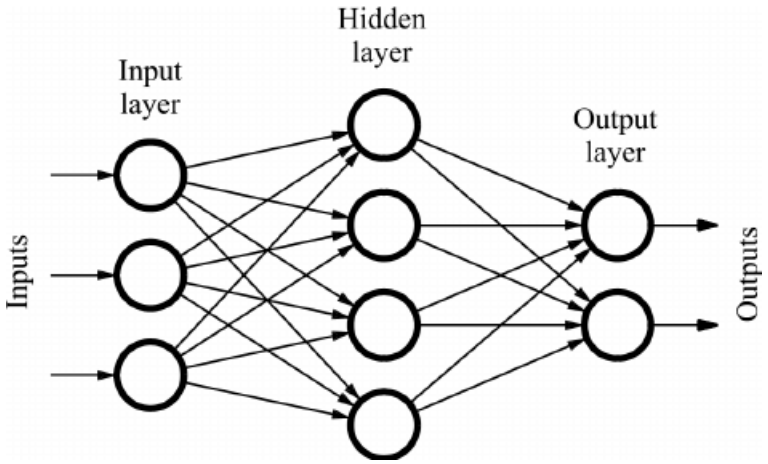

Figure 2.2: Graphical representation of a *FFNN* architecture

$$L(\hat{y}, y) = y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \tag{2.2}$$

where y is the true output of the system, $\hat{y}$ is the model's predicted output, and $L(\hat{y}, y)$ is the selected loss function between the true and predicted outputs.

2.3.3 Adam Optimizer

ANNs "learn" by minimizing a given loss function $L(\hat{y}, y)$, by modifying every parameter of the network using true historical data from the system (Supervised learning). This learning process is done using a technique called *Back Propagation* which consists on using *Stochastic Gradient Descent (SGD)* (eqn. 2.3) to adjust every θ parameter to minimize the loss function. It is called *Back Propagation* because the flow of information now goes from outputs to inputs.

$$\theta_i = \theta_i - \alpha \left(\frac{\partial L(\hat{y}, y)}{\partial \theta_i} \right) \tag{2.3}$$

where θ_i represents the i^{th} parameter from the complete parameter space Θ, i.e. $\theta_i \in \Theta$, and α is the learning rate of the *ANN*. *SGD* uses a single training sample to optimize the system's parameters. This could be beneficial for very large data sets, but inefficient for smaller ones. Most of the times we may converge to the optima solution faster by dividing our samples in batches and training using each batch of samples at a time. This process is called *Mini-batch Gradient Descent* and is currently the most commonly used implementation of Gradient Descent [Li *et al.*, 2014] and the one we will use in thie research project.

The *Adaptive Momentum Estimation (Adam)* optimizer is a variant to the regular *SGD* optimizer that computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [Kingma and Ba, 2014].

$$\theta = \theta - \alpha \left(\frac{V_{d\theta}^{corrected}}{\sqrt{S_{d\theta}^{corrected}} + \epsilon} \right) \tag{2.4}$$

$$V_{d\theta}^{corrected} = \frac{V_{d\theta}}{(1-\beta_1)^t} \quad and \quad S_{d\theta}^{corrected} = \frac{S_{d\theta}}{(1-\beta_2)^t} \tag{2.5}$$

$$V_{d\theta} = \beta_1 V_{d\theta} + (1-\beta_1) d\theta \quad and \quad S_{d\theta} = \beta_2 S_{d\theta} + (1-\beta_2) d\theta^2 \tag{2.6}$$

where $V_{d\theta}$ and $S_{d\theta}$ are moving averages at iteration t for the first and second momentum respectively, β_1 and β_2 are hyper-parameters (typically 0.9 and 0.999 respectively), $d\theta$ and $d\theta^2$ are the gradient and element-wise squared gradient for parameter θ respectively, $V_{d\theta}^{corrected}$ and $S_{d\theta}^{corrected}$ are the bias corrected moving averages, and ϵ is a hyper-parameter to ensure not dividing by 0 (typically around 10^{-8}).

2.3.4 Performance metrics

The *CM* is widely used to report results in classification problems, because it is possible to observe the relations between the classifier's predicted outputs and the system's true outputs. Table 2.2 shows an example of a *CM* for a binary classification problem. *TN* and *TP* are the elements correctly classified, and therefore the elements outside the diagonal represent the ones miss-classified. A asymmetric *CM* is related to a biased classifier.

Table 2.2: *CM* for a binary classification problem

		Prediction	
		Negative	Positive
True Label	Negative	True Negative (*TN*)	False Positive (*FP*) Type I error
	Positive	False Negative (*FN*) Type II error	True Positive (*TP*)

Accuracy is a metric used to predict the correctness of a *ML* model, eqn (2.7). *Precision* means the percentage of the results which are relevant, eqn (2.8). *Recall* refers to the percentage of total relevant results correctly classified, eqn (2.9). The *F-measure* was developed as an evaluation metric that would give a harmonic mean between precision and recall. The β parameter from eqn. (2.10) gives more weight to recall as it increases.

$$Accuracy = \frac{TP+TN}{Total} = \frac{TP+TN}{TP+FP+TN+FN} \tag{2.7}$$

$$Precision = \frac{TP}{Actual\ Results} = \frac{TP}{TP+FP} \tag{2.8}$$

$$Recall = \frac{TP}{Predicted\ Results} = \frac{TP}{TP+FN} \tag{2.9}$$

$$F_\beta = (1+\beta^2)\frac{Precision * Recall}{(\beta^2 * Precision) + Recall} \tag{2.10}$$

The *AUC-ROC* score is used to evaluate the real classification performance of a model. This curve is obtained by measuring the *True Positive Rate (TPR)* and *False Positive Rate (FPR)* using different decision thresholds. The closer the *AUC-ROC* score is to 1, the better is the prediction performance of the evaluated model.

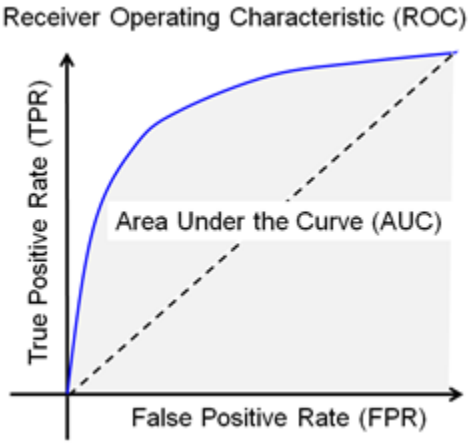

Figure 2.3: Area Under the Curve graphic

2.3.5 Learning algorithm

An overview of the *DL* algorithm can be visualized in Fig. 2.4. We start by feeding the *ANN* a batch of samples and forward propagating them trough the network (eqn. 2.1). Once we have the network's predicted output for the selected batch of samples, we compare its result with the system's true output using the defined Loss function (eqn. 2.2). Using the Loss function value we can then proceed to adjust the network's parameters using the defined Optimizer function (eqn. 2.4) through Back Propagation. We then repeat this process for every batch of samples to train on every available training sample.

The number of times to perform the learning algorithm is defined by the $epochs$ hyper-parameter. Final model's performance is calculated using the defined performance metrics.

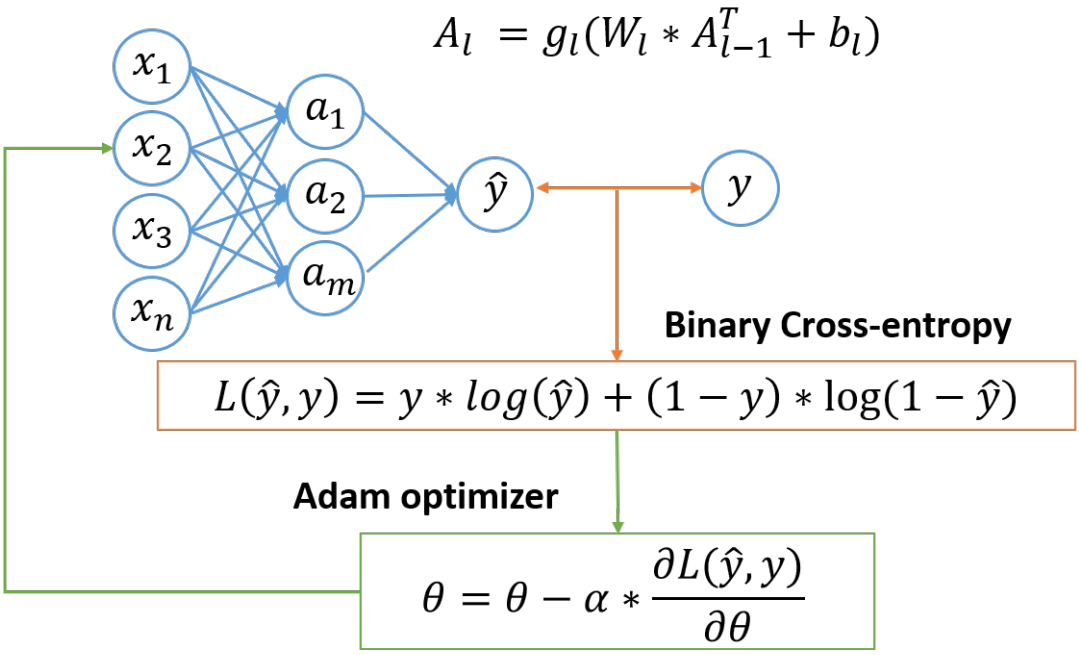

Figure 2.4: *DL* learning flow chart

This research projects proposes a *DL* model for mortality risk prediction for *CoViD-19* patients. The trade-off between feature and sample space is analyzed, along with the effects of using over-sampling in the unbalanced data set.

Chapter 3

Experimental System

3.1 Database description

Because there was no reliable database for CoViD-19 patients in Mexico, data from a different region was used to validate the proposed methodology. The database was provided by *HM Hospitales* from Madrid, Spain, under the *Covid Data Save Lives* project, [Hospitales, 2020]. It contains the anonymized records of 2,307 *CoViD-19* patients since the beginning of the epidemic to April 25, 2020. The database is divided into six different sections, each containing a different type of data for each patient. The common key among every file is the *patient_id* feature, which helps to identify patients across every section of the database. A summary of the information contained in the database is shown in Table 3.1.

Table 3.1: Complete raw database's sections description

Section	*Description*
1	Demographic data: Patient ID, Age, Gender, Diagnosis (Positive/Negative/Pending), Admission/Discharge date and motive, SpO2, Temperature, Heart rate, Blood Pressure, etc.
2	Prescribed medication: daily dose and duration
3	Evolution of vitals signs: SpO2, Heart rate, Temperature, Blood pressure and Blood glucose values
4	Laboratory tests with date, results and units
5	Comorbidities, these are coded based in [World Health Organization, 2019]

This raw database poses five main technical challenges:

1. Incomplete records. A *Not a Number (NaN)* value is placed whenever there is a missing value for a feature.

2. Features with different engineering scales.
3. Combination of categorical and numerical features.
4. Irrelevant and redundant features.
5. Highly unbalanced classes.

3.2 Data cleansing

From all of the 2,307 available records, patients were filtered according to the following criteria: Leave only patients who:

- have a *CoViD-19* positive diagnosis.
- have been discharged or confirmed deceased.
- have an age different than 0.
- have a registered value of SpO2.

After applying all of these filters, only 1,503 patients were kept. A *RF* algorithm was used to select the features with the highest predictive power to decrease the feature space, by analyzing the importance assigned to each feature by the algorithm. For this purpose, the *SHapley Additive exPlanations (SHAP)* value [Lundberg and Lee, 2017] was used to estimate the impact/weight of each input variable in prediction. The *SHAP* value graph is a graphical visualization of how much a feature contributes to a model's prediction. A large positive *SHAP* value indicates the feature is very relevant to detect positive outputs, while a large negative value is associated to a negative output. The color bar shows the feature value associated to the given *SHAP* value, while the thickness of a feature's line, indicates the amount of samples present in the data set for the given feature value. Only the 4 most important input features are shown in Fig. 3.1. It corroborates the assumptions that the older people have higher mortality risk than the younger people, and the lower SpO2, the higher the risk. The selected demographic features were: Age, Gender, Patient ID and Discharge motive (Label). The 5 most relevant comorbidities features were: kidney failure (N17), hypertension (I10), diabetes (E11), heart disease (I25) and respiratory distress (J80).

Most relevant lab tests were selected by literature review. Selected biomarkers include: Prothrombin activity, Creatinine, Dimer-D, Ferritin, Immunoglobulin G, Immunoglobulin M, Interleukin 6, Lactate, LDH, Leukocytes (Count and %), Lymphocytes (Count and %), Neutrophils (Count and %), C-reactive protein, Platelets, Prothrombin time, and Troponin. Because sampling

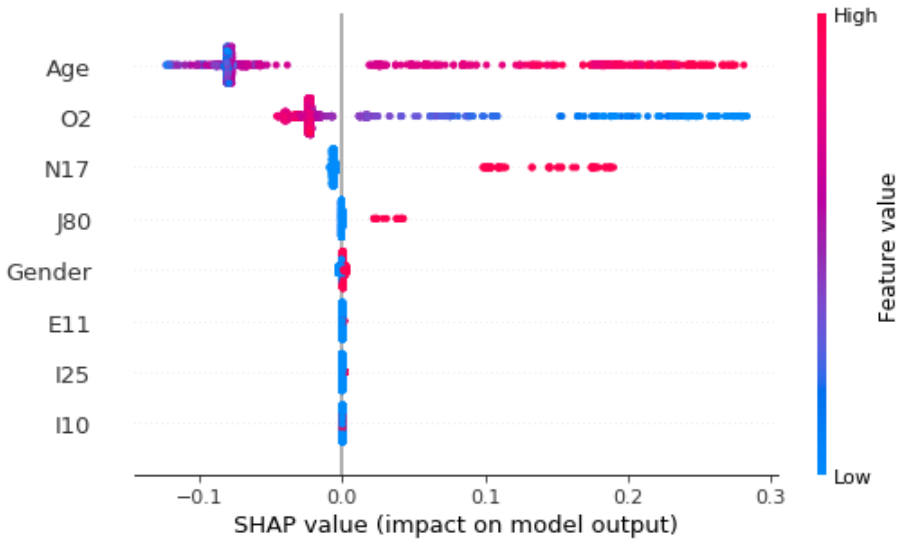

Figure 3.1: Relative importance of features according to *SHAP* value.

frequency for lab tests is inconsistent, simple time series statistical representation, such as maximum and minimum values, were chosen to represent these features. Therefore, we add two features for every biomarker, one for the maximum and another one for the minimum values.

All available features for study are presented in Table 3.2. Features 1, and 4 through 9 are categorical data $\in [True, False]$, while every other feature is numerical data $\in \mathbb{R}$.

3.3 Data distribution

Table 3.3 shows the filtered database distribution with 1,503 *CoViD-19* patients. Fig. 3.2 shows some graphical representations for the main features and Appendix C shows the distribution for every biomarker feature.

From Fig. 3.2 we can see that the patient's age distribution has a mean of about 70 years old following a normal distribution. Oxygen saturation values have a mean of 92.28, with a couple of lower outliers, which suggest a more severe disease state according to literature. A clear unbalance between the classes is clearly observed, with only 16.5% of deceased patients. As for comorbidities, there are 919 patients with none of the selected comorbidities, 398 patients with only 1 comorbidity, 148 patients with 2 of them, 35 patients with 3, 3 patients with 4 comorbidities, and no patients with every comorbidity. The most common comorbidity among patients is hypertension.

Table 3.2: (I)ndex, acronym and name of the features (clinical data)

I	*Acronym*	*Feature*	*I*	*Acronym*	*Feature*
1		Gender	25	IGM_m	IgM (Immunoglobulin M) min
2		Age	26	IGG_m	IgG (Immunoglobulin G) min
3	SpO2	Oxigen saturation	27	TNI_m	Troponin min
4		Kidney Failure	28	AP_m	Prothrombin activity min
5		Hypertension	29	TP_m	Prothrombin time min
6		Diabetes	30	$LIN\%_M$	Lymphocytes % Max
7		Heart Disease	31	LIN_M	Lymphocytes Max
8		Respiratory distress	32	$LEUC_M$	Leukocytes Max
9		Discharge motive	33	NEU_M	Neutrophils Max
10		Patient ID	34	$NEU\%_M$	Neutrophils % Max
11	$LIN\%_m$	Lymphocyte % min	35	$PLAQ_M$	Platelet count Max
12	LIN_m	Lymphocytes min	36	PCR_M	C-reactive protein Max
13	$LEUC_m$	Leukocytes min	37	DD_M	Dimer D Max
14	NEU_m	Neutrophils min	38	Cr_M	Creatinine Max
15	$NEU\%_m$	Neutrophils % min	39	LDH_M	LDH Max
16	$PLAQ_m$	Platelet count min	40	$IL6_M$	Interleukin 6 Max
17	PCR_m	C-reactive protein min	41	$LEULCR_M$	Leukocytes count Max
18	DD_m	D Dimer min	42	LAC_M	Lactate Max
19	Cr_m	Creatinine min	43	FER_M	Ferritin Max
20	LDH_m	LDH min	44	IGM_M	IgM (Immunoglobulin M) Max
21	$IL6_m$	Interleukin 6 min	45	IGG_M	IgG (Immunoglobulin G) Max
22	$LEULCR_m$	Leukocytes count min	46	TNI_M	Troponin Max
23	LAC_m	Lactate min	47	AP_M	Prothrombin activity Max
24	FER_m	Ferritin min	48	TP_M	Prothrombin time Max

Table 3.3: Filtered database distribution according to key features. Biomarkers data shows average values for maximum and minimum features

Feature	*Detail*	*Number of patients (%)*
Gender	Male	927 (61.7%)
Comorbidities	Kidney failure	81 (5.4%)
	Hypertension	446 (29.7%)
	Diabetes	194 (12.9%)
	Heart disease	69 (4.6%)
	Respiratory distress	21 (1.4%)
Discharge motive	Deceased	248 (16.5%)
		Mean value min - max
Biomarkers	LEUC ($\times 10^3/\mu$L)	6.03 - 9.85
	LIN ($\times 10^3/\mu$L)	1.00 - 1.65
	LIN%	13.90 - 25.03
	NEU ($\times 10^3/\mu$L)	4.10 - 7.84
	NEU%	63.23 - 78.66
	PLAQ ($\times 10^3/\mu$L)	206.42 - 320.81
	Cr (mg/dL)	0.84 - 1.09
	PCR (mg/L)	38.84 - 130.69
	LDH (U/L)	482.91 - 726.57
	DD (ng/mL)	1194.71 - 4509.60
	IL6 (pg/mL)	192.74 - 239.50
	LAC (mmol/L)	1.73 - 2.29
	FER (ng/mL)	1150.13 - 1526.91
	TNI (ng/L)	27.01 - 36.47
	AP (%)	70.58 - 80.93
	TP (s)	13.70 - 16.55

3.4 Preprocessing

To increase the training data availability, a preprocessing algorithm [Escobar *et al.*, 2020] is applied, it is a *Greedy-like* algorithm that at each iteration maximizes the number of samples by selecting the column (feature) with more rows (samples) available. Since the original data set contains a lot of missing cells, the sample size reduces as the number of features increases. Fig. 3.3 shows the amount of available samples per feature where every vertical line shows the partition for

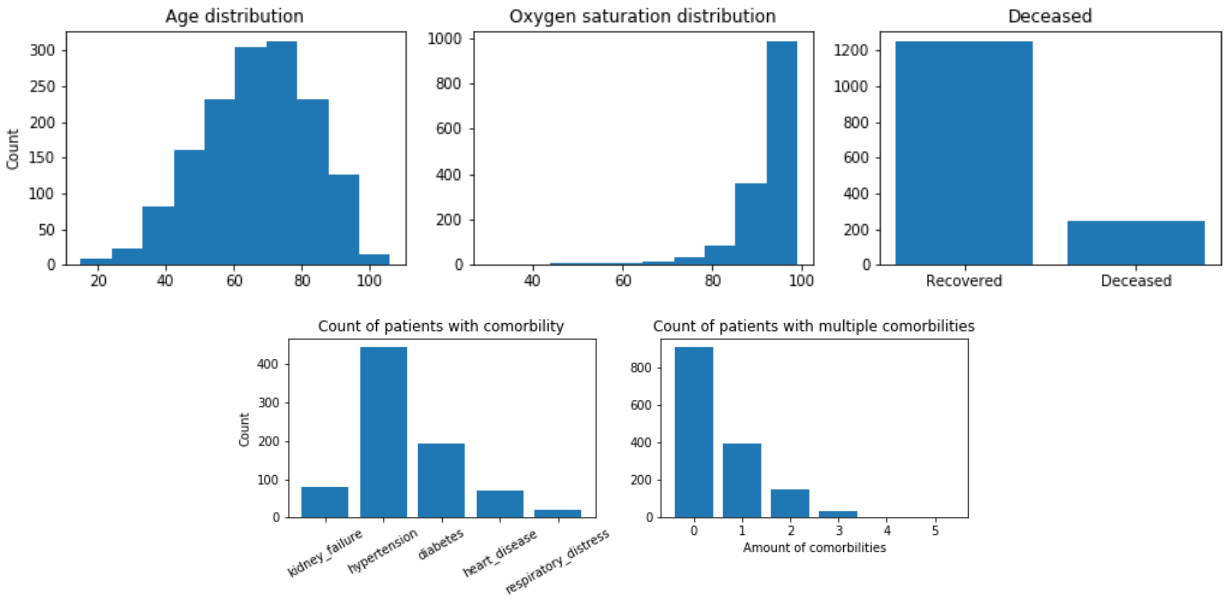

Figure 3.2: Distribution of age, SPO2, deceases and comorbidities distributions of patients in filtered database

every sub data set. By examining the plot, eight sub-data sets are heuristically created. Table 3.4 shows the number of features and samples in every sub data set. After the 8^{th} data set most of the samples are lost, so we will only analyze until this data set.

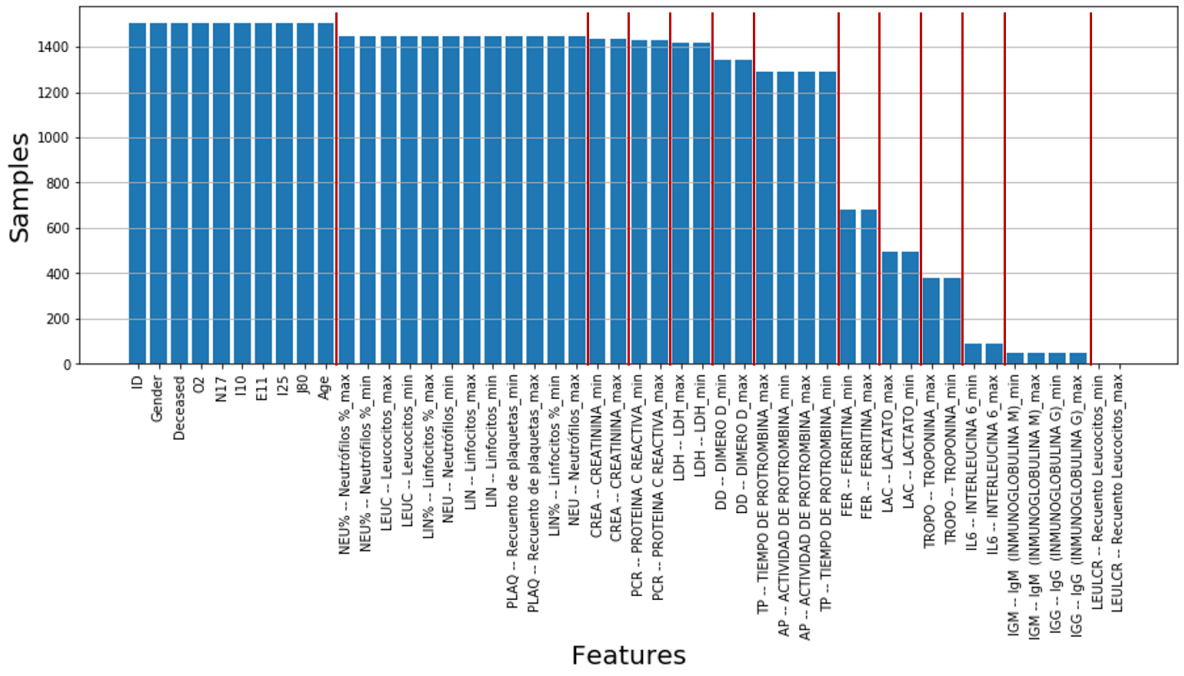

Figure 3.3: Sub-data sets divisions according to available sample number

Table 3.4: Number of features and samples in every defined sub data set

	Data sets												
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
Num. of features	10	22	24	26	28	30	34	36	38	40	42	46	48
Num. of samples	1503	1449	1434	1428	1419	1341	1291	683	496	379	90	47	2

The eight sub-data sets pose a trade off between the number of features and the number of samples, since the sub-set with the most discriminatory information cannot be determined in advanced, the learning algorithm is applied to all of them.

Since normalizing the data generally speeds up learning and leads to faster convergence [Ioffe and Szegedy, 2015], the remaining numeric features are re-scaled using the *Min-Max* normalization method (eqn. 3.1), [Mohamad and Usman, 2013].

$$X_{norm} = (x - x_{min})/(x_{max} - x_{min}) \tag{3.1}$$

Chapter 4

Proposal

Section 4.1 reviews the *Key Performance Indicator (KPI)* used to evaluate the model. Then, section 4.2 shows the training and testing strategy and the decision threshold tuning strategy is explained in section 4.3. Section 4.4 and 4.5 introduce the over-sampling and data imputation approaches respectively. Finally, the proposed *DL* model is described in section 4.6

4.1 KPI

The *MPCD* score is a probabilistic-based measure of classification performance aimed at analyzing highly unbalanced data structures. It is sensitive to the recognition rate by class; therefore, in a high conformance production rate, its score mainly describes the ability of a model to correctly classify the minority class. The *MPCD* score is calculated following eqn (4.1). The α and β errors can be calculated following eqn (4.2) from the classifier's *CM*.

$$MPCD = (1-\alpha)(1-\beta) = (TPR)(TNR) \tag{4.1}$$

where $MPCD \in [0, 1]$

$$\alpha = \frac{FP}{FP+TN}, \;\; \beta = \frac{FN}{FN+TP} \tag{4.2}$$

Table 4.1 shows the behaviour of the $MPCD$ score and accuracy given different performance scenarios using the database described in section 3 where we have 248 positive (deceased) samples and 1255 negative (recovered) samples.

Table 4.1 shows that the $MPCD$ score gives more weight to the *TP* metric, as it corresponds to the minority class, and how the $MPCD$ score is better to identify the *FN* predictions over some ordinary metric like the *Accuracy*.

Table 4.1: Examples for MPCD scores

Scenario	*TN*	*TP*	*FN*	*FP*	α-*error*	β-*error*	*Accuracy*	$MPCD$
1	1255	248	0	0	0	0	1	1
2	1254	248	0	1	7.96×10^{-4}	0	0.999	0.999
3	1255	247	1	0	0	4.03×10^{-3}	0.999	0.995
4	1155	248	0	100	0.079	0	0.933	0.92
5	1255	148	100	0	0	0.40	0.933	0.59
6	1255	0	248	0	0	1	0.835	0

4.2 Training and Testing split

To properly assess the performance of a *ML* model, we must compare its outputs to the true outputs generated by the real system. Ideally, we would generate an extra set of samples for testing purposes. Alternatively, we can create these data sets from our original data set by dividing it in training and testing samples. Testing samples are typically a small fraction of the data set, just big enough to have enough population variation in them. For every data set we will use the K-fold *CV* algorithm. In this algorithm, the data set is first randomly shuffled to avoid bias and then divided in K equally sized parts (folds). The proposed model is trained K times, where at each iteration a different fold of the data set is used as the testing set while every other fold is used for training. The final unbiased result is the average value of each evaluation metric across every fold. The proportion of the label classes distribution should be kept across every fold. This is done to avoid a fold of the model training with only positive or negative class samples, therefore getting a biased classifier. For this purpose we use the *StratifiedKFold* scikit-learn function, which keeps the proportion of the label feature across every fold. For our model we will use 90% of the samples for training and 10% for testing (i.e. $K = 10$). Fig. 4.1 shows a graphical representation of the 10-Fold *CV* algorithm.

4.3 Decision threshold

The output of the trained model is the probability of a given sample belonging to one class or another. Intuitively, we might assign an equally spaced decision threshold for every class, e.g. 0.5 for a binary classification problem. However, it is not recommended to just take the largest output value to assign a class, particularly with highly unbalanced data sets. Depending on the performance metric we would like to optimize, we might desire a higher or lower decision threshold. For example, for our mortality risk calculator, the *FNR* should be prioritized over the *FPR*. In this

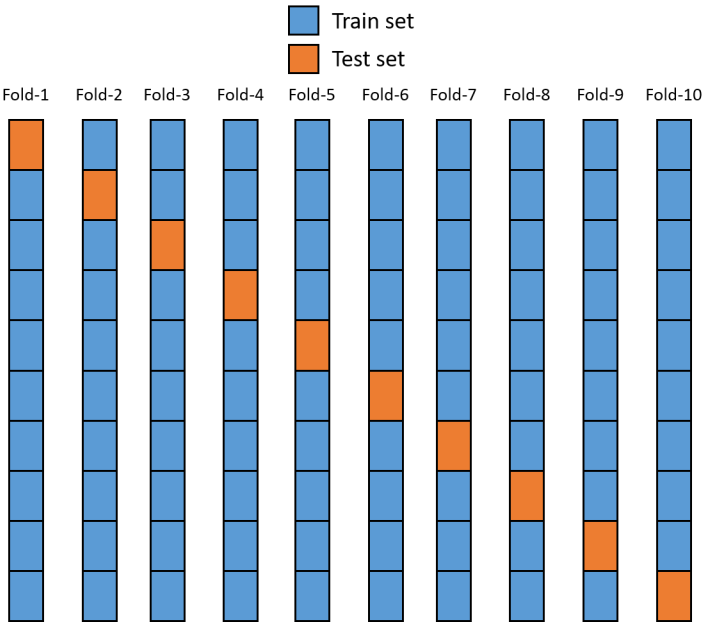

Figure 4.1: 10-Fold *CV* sets distribution

case, we might desire a lower decision threshold, to avoid miss classifying a patient with a high mortality risk. If we use 0.2 as a decision threshold to decide whether a patient has a high mortality risk or not, we would classify them as a patient with high risk even if we are only 20% sure of it. This would minimize the $Recall$ and therefore the FNR scores.

This research project will use the *Optimal Classifying Threshold Method (OCTM)* algorithm [Escobar and Morales-Menendez, 2017] to obtain the decision threshold value which optimizes the *MPCD* score (see Fig. 4.2).

Input: $CP(CP_1, CP_2, \ldots CP_m)$:
List of conditional probabilities ordered
Output: γ
Optimal classification threshold
Initialization: *set CCTL as empty*
List of MPCD values associated to each candidate classification threshold

1. *For* $i = 1$ *to* $m - 1$ *do begin*
2. $CCT_i = \frac{CP_i + CP_{i+1}}{2}$
3. *estimate* $MPCD_i$ *at each* CCT_i
4. *add* $MPCD_i$ *to CCTL*
5. *end*
6. *Find* p, *the position of the* max $(CCTL)$
7. *return* $\gamma = CCT_p$

Figure 4.2: Pseudo-code of the *OCTM* algorithm

4.4 SMOTE technique

The *Synthetic Minority Over-sampling TEchnique (SMOTE)* is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement. It generates synthetic examples in a less application specific manner, by operating in "feature space" rather than "data space". The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors [Chawla *et al.*, 2002]. It is not advisable to sample synthetic samples until both classes are completely balanced, because completely balanced classes are, in most cases, not necessary to proper model a system's behaviour. The *SMOTE* algorithm is implemented using the imbalanced-learn (version 0.7.0) [Lemaître *et al.*, 2017] from Python. For this research project, a final proportion of $sm = 0.80$ will be set to the minority class.

$$sm = \frac{Majority\ class}{Minority\ class} \tag{4.3}$$

4.5 Data imputation

The impact of using imputed data on the prediction model is evaluated for the cases where there are time or budget constraints, and obtaining complex biomarker data is impossible or unfeasible. To properly evaluate the proposed imputation method, we will only imputate available biomarker features to be able to calculate an error between imputated and real values. The methodology described in section 4.2 will be followed. Data will be split in train and test sets using a 10-Fold approach. Biomarker features will be imputed on the test set using the mean value of the K most similar patients from the real biomarker data of the train set using the *KNN* algorithm. The value of K is determined by the amount of available data. The benefit of using the imputed features will be evaluated by comparing the model's performance against the same test set with the real biomarker data. The error of the estimated imputed data is calculated using the *Root Squared Error (RSE)* (eqn. 4.4).

$$RSE = \sqrt{(x_i - \hat{x_i})^2} \tag{4.4}$$

where x is the real feature value and $\hat{x}$ is the imputed feature value.

Finally, the benefit of adding imputed biomarkers data will be measured by comparing the performance of the imputed test set against the performance of a model which only uses basic patient information, without any imputation. As we impute more features, the model's performance will have more uncertainty and therefore a higher error. This motivates us to impute only the necessary amount of features to see an improvement on our model without adding variance to the output. The *SHAP* values of the data set (see table 3.4) with the best overall performance will be

used for feature selection. This research project will only impute 2 features to keep a low variance. Fig. 4.3 shows the *SHAP* values for the features included in the 4^{th} data set. In this case we would choose to impute the 2 most relevant features for prediction: NEU$\%_m$ and LIN$\%_M$. Since this feature selection could be biased because of the size of available features in our data base, we will also run a test using features recommended in the literature for the imputation process: DD$_m$ and DD$_M$

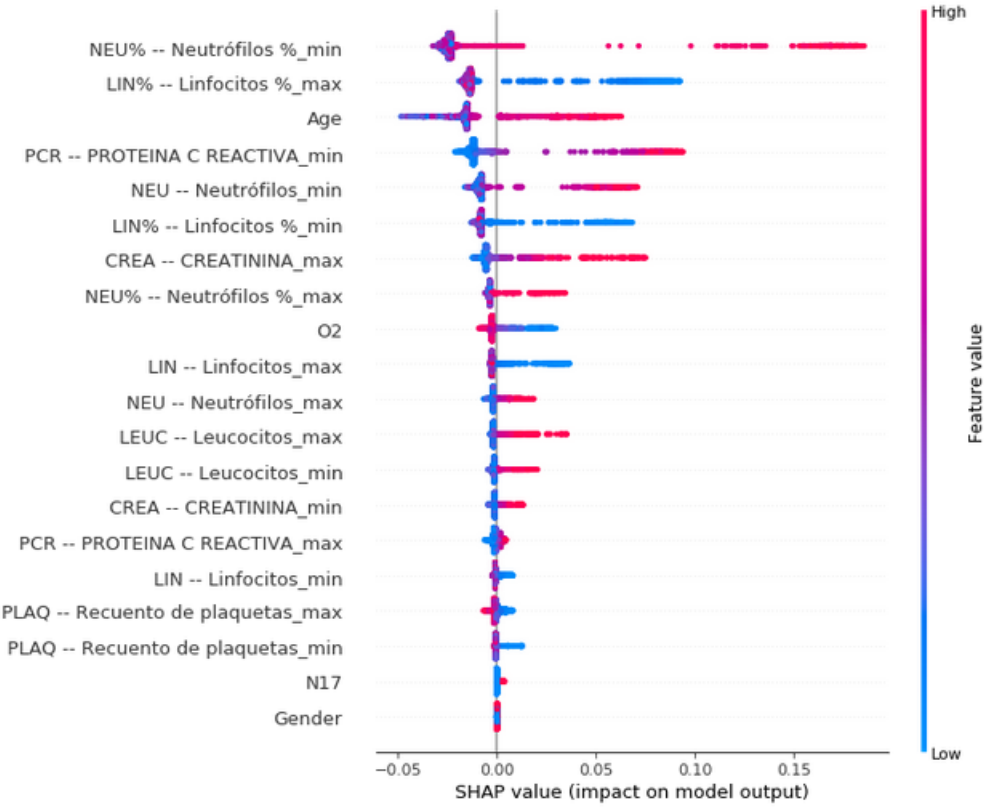

Figure 4.3: *SHAP* values for Data set 8

4.6 Deep Learning model

The *DL* model is a *FFNN* using binary cross-entropy as the loss function and the Adam algorithm as an optimizer to adjust the network's weights. Fig. 4.4 shows a graphical representation of the *FFNN* model. The model has 3 hidden layers with 17, 10 and 5 neurons respectively. The "?" sign stands for the batch size the mini-batch optimization technique. Since we are dealing with a binary classification problem we can use the Sigmoid function as the output activation, which will set our output to be $\in [0, 1]$. By tuning the decision threshold we can then decide to assign either one of the two classes to th prediction.

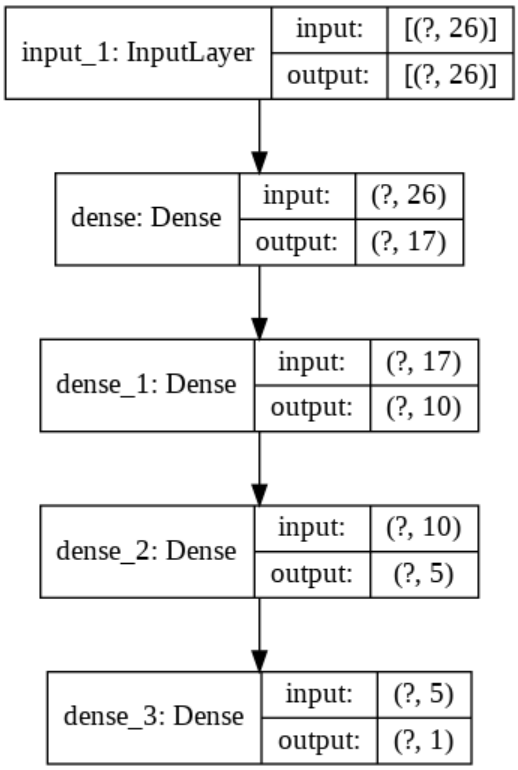

Figure 4.4: Proposed *FFNN* model plot

Hyper-parameters are summarized in table 4.2. This model was developed using the Keras framework (version 2.2.4) running on Tensorflow 2.0.0 in python 3.6

Table 4.2: Hyper-parameters used for the proposed *FFNN* model

Parameter	*Value*
Hidden layers	3
Neuron number	[17, 10, 5]
Hidden activation functions	Sigmoid
Output activation	Sigmoid
Batch size	32
Epochs	200
Learning rate	0.001
β_1	0.9
β_2	0.999
ϵ	1e-07

Chapter 5

Results

This chapter summarizes the results found for the proposed *DL* model on the data sets described in section 3.4. Section 5.1 summarizes the results for data sets with and without the *SMOTE* approach. Both approaches' results are compared against a *RF* model in section 5.2. Then, section 5.3 presents the results for the data imputation technique described in section **??**. Finally, section 5.4 presents a discussion of the obtained results.

5.1 *DL* results summary

Fig. 5.1 shows a boxplot graph of the *MPCD* score for the proposed *DL* model with and without the *SMOTE* approach on every sub data set described in section 3.4.

Results for every fold in every sub data set are presented in Appendix D following the performance indicators proposed in section 4.1, along with a boxplot graph for every *KPI* across the 10-folds. After the 4^{th} data set the *MPCD* score does not improve significantly. Therefore, we use this data set as the optimal one and as a comparison point. Fig. 5.2 shows the *MPCD* boxplot graph for every fold of the 4^{th} data set for both approaches.

5.2 Performance comparison

For comparison purposes, we will train a *RF* algorithm on the same data sets described in section 3.4. Section 5.2.1 briefly describes the *RF* algorithm.

5.2.1 The *RF* algorithm

The *RF* algorithm uses an ensemble of Decision Trees to make a prediction. A Decision Tree fits a function (typically piece-wise constant, i.e. mean) over domain $\mathcal{X}$ by recursive partitioning in a greedy way, see Fig. 5.3.

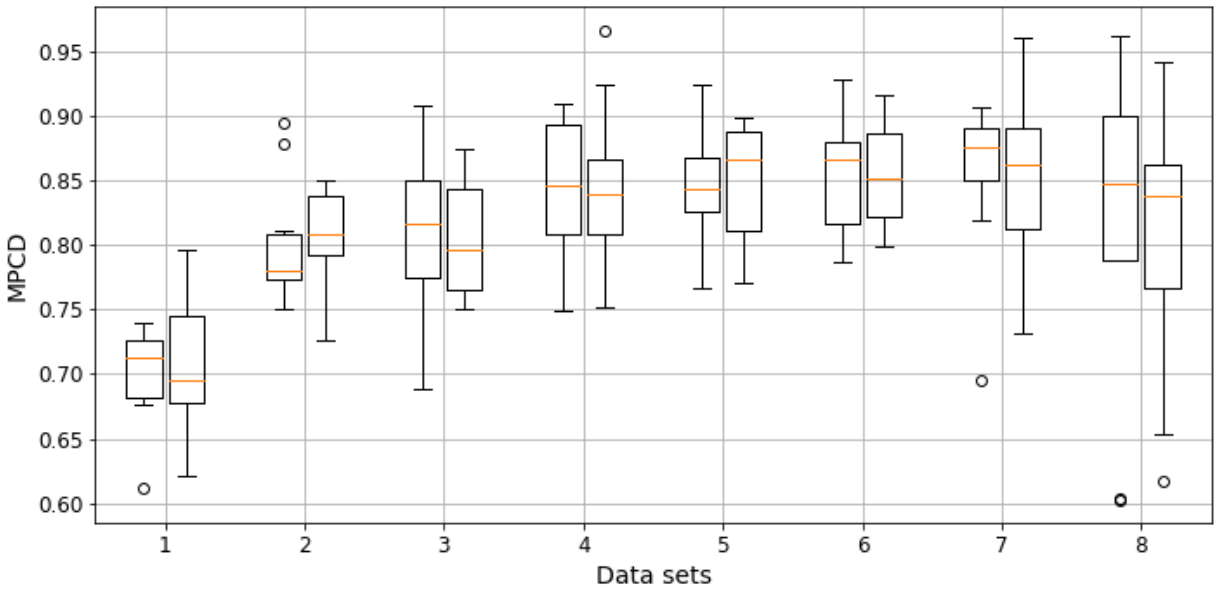

Figure 5.1: Results of the *DL* model for every data set (Left - Normal, Right - *SMOTE*)

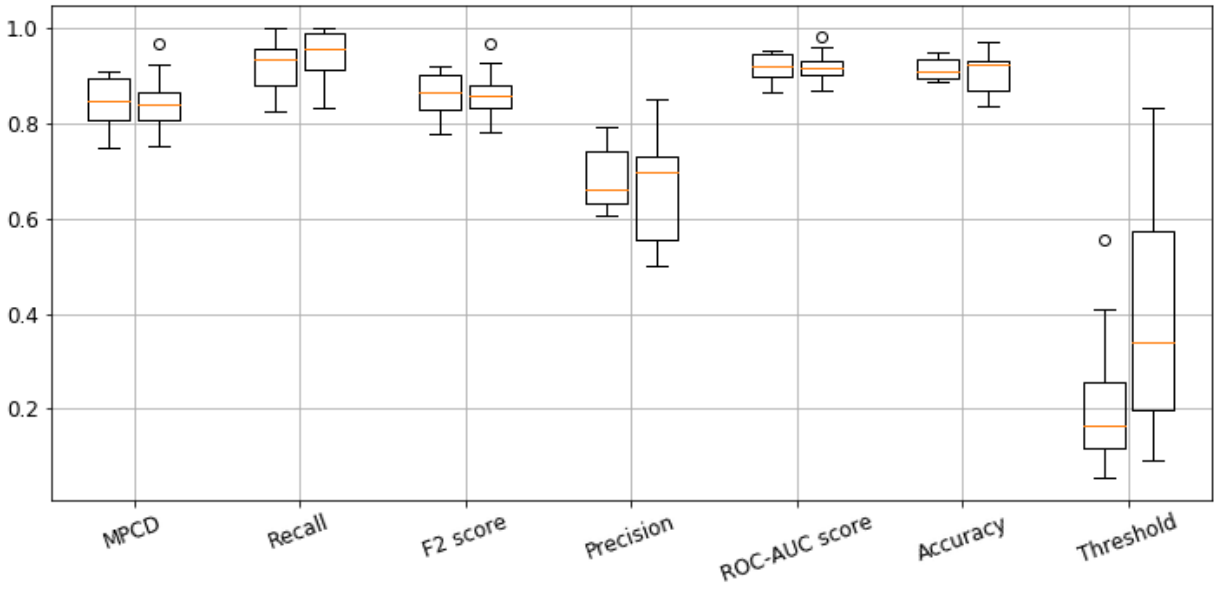

Figure 5.2: Performance metrics for sub-data set 4 (Left - Normal, Right - *SMOTE*)

Individual Decision Trees suffer from high variance, although when very bushy (very deep), they have very low bias. A *RF* is a classifier consisting of a collection of tree-structured shallow classifiers $[h(x, \Theta_k), k = 1, ...]$ where Θ_k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x [Breiman, 2001]. *RF* use a collection of shallow tree classifiers together with bootstrap and random variable subspace sampling

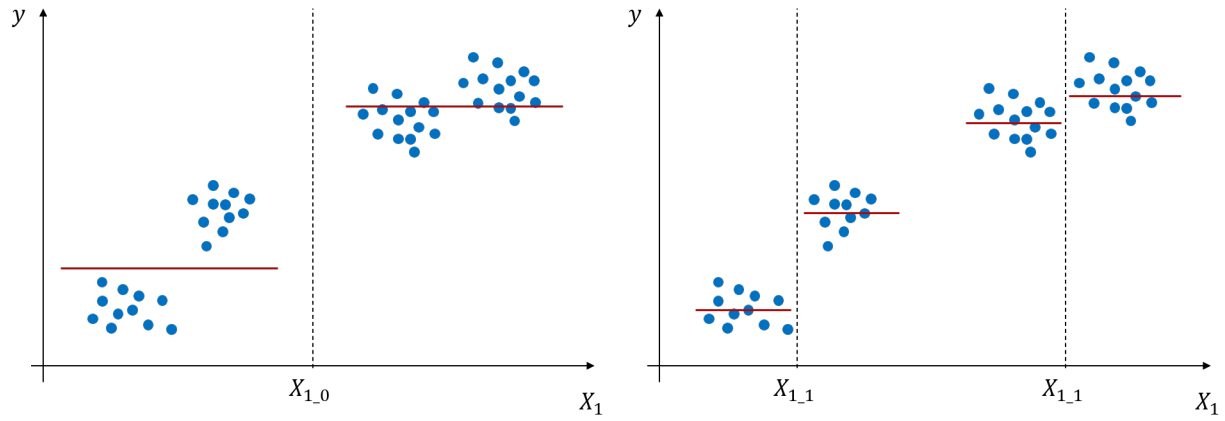

Figure 5.3: Recursive partitioning of variable X_1 in a Decision Tree

to reduce the high variance mentioned above while maintaining their low bias attribute. The final prediction of the *RF* is defined by the majority between every decision tree.

Although this is a binary classification problem, we use a *RF* Regressor in order to be able to tune the decision threshold for classification as described in section 4.3. This model was developed using the sckit-learn (version 0.23.2) library in python (3.6). Table 5.1 shows the *RF* model's hyper-parameters.

Table 5.1: Hyper-parameters used for the *RF* Regressor model

Parameter	*Value*
Number of estimators	500
Maximum depth	2
Maximum Features	5

5.2.2 *DL* vs *RF* results

A comparison of the results from the *DL* and *RF* models on every data set with and without the *SMOTE* approach are presented in Fig. 5.4 and Fig. 5.5 respectively.

KPI tables and boxplot graphs for every fold on every data set for the proposed *RF* model can be found in Appendix D.2.

5.3 Data imputation results

Results of imputating the NEU%$_m$ and LIN%$_M$ biomarkers are shown in Fig. 5.6. The boxplot graph of every performance metric is compared with and without imputation. We can see an in-

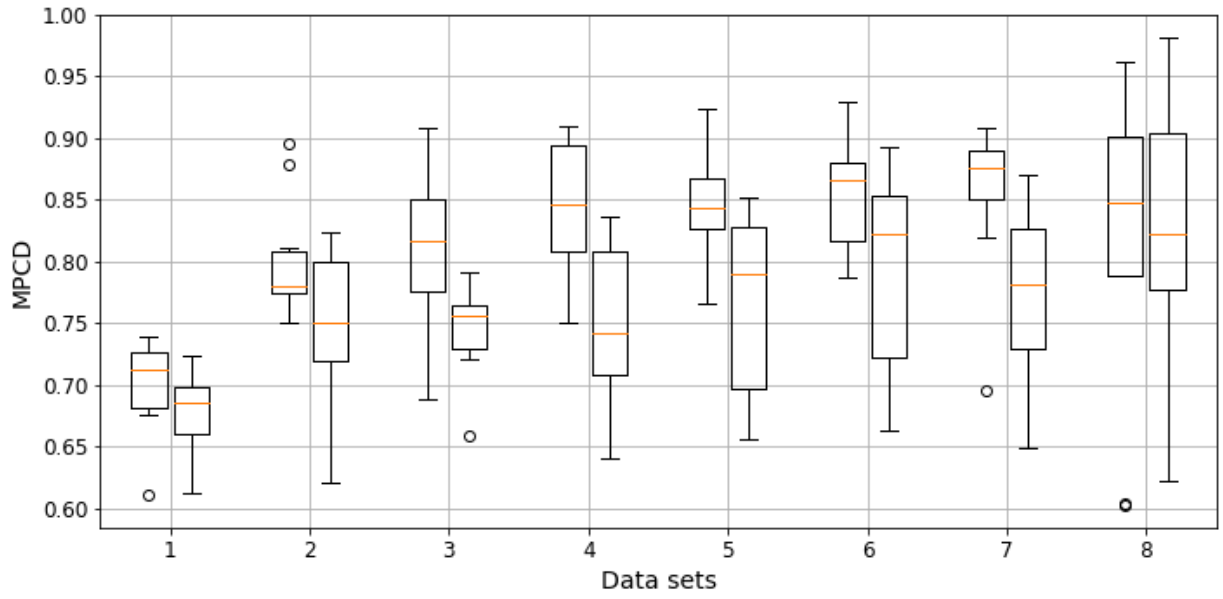

Figure 5.4: Comparison of results for the *DL* (left) and *RF* (right) models

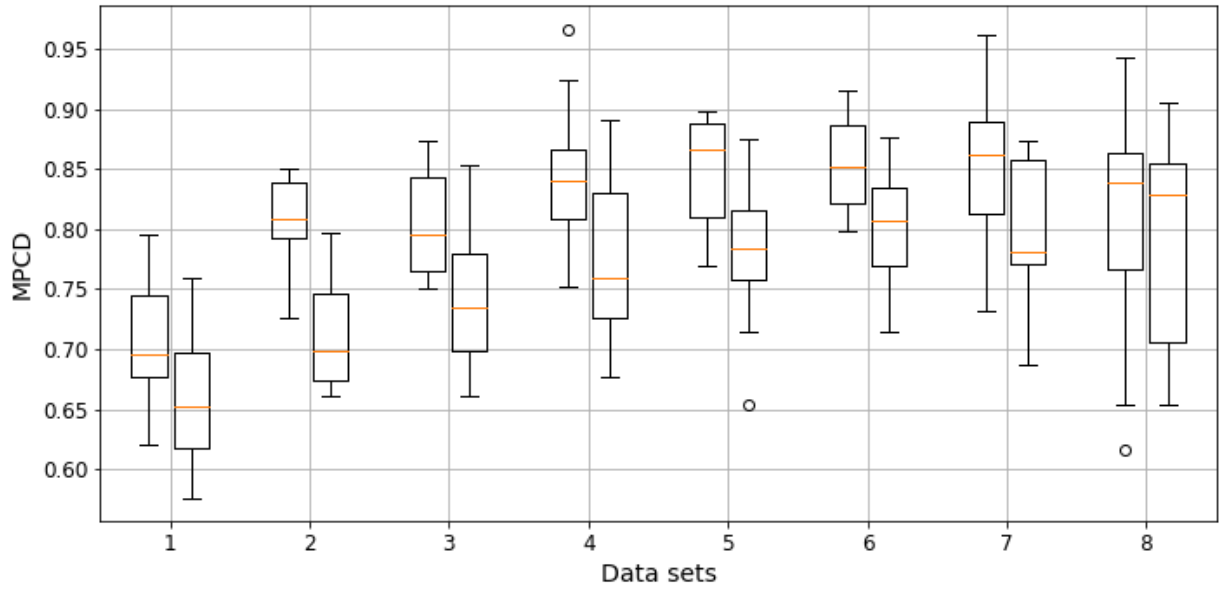

Figure 5.5: Comparison of results for the *DL* (left) and *RF* (right) models using *SMOTE*

crease in the variance of each evaluation metric, which is expected because of the error introduced by the imputation process. The mean performance value of the model was overall the same, which suggests that the imputation process did not introduce false information to the process. Results for the DD_m and DD_M imputation process are shown in Fig. 5.7. In this case we can see a similar variance when comparing the imputated and real value models, suggesting a good imputation

performance.

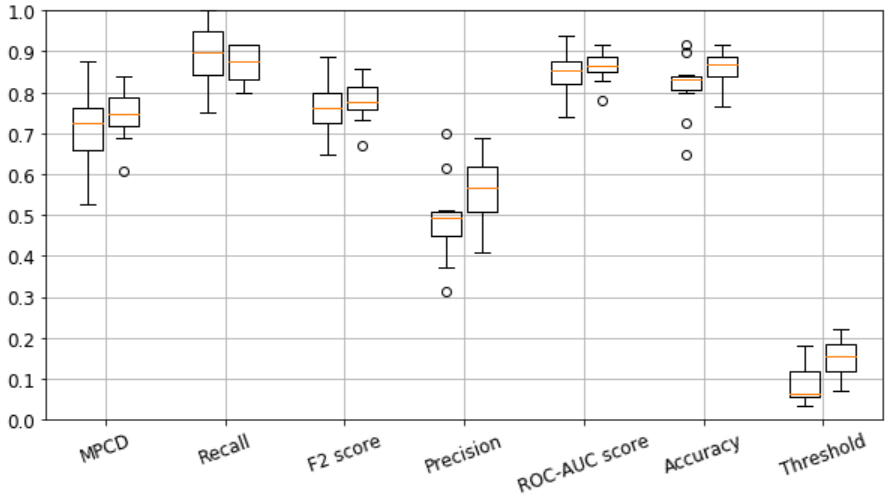

Figure 5.6: Performance metrics for the imputation of the NEU%$_m$ and LIN%$_M$ biomarkers (Left - Imputated, Right - Real values)

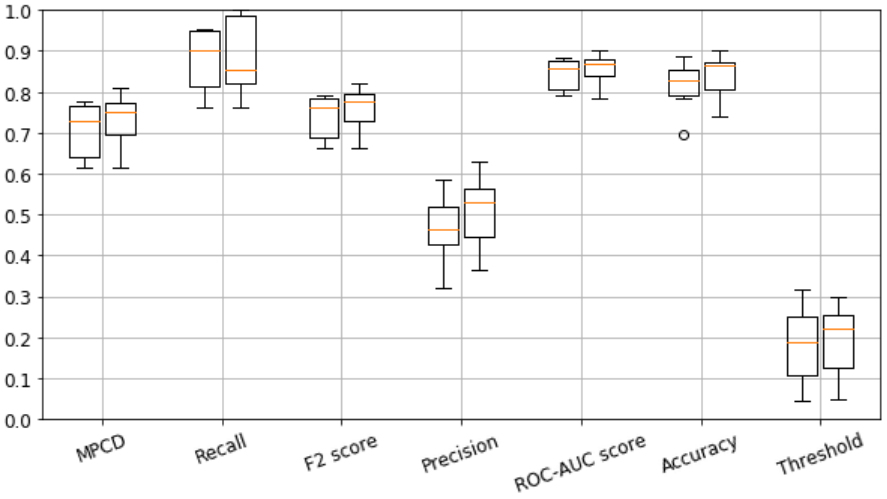

Figure 5.7: Performance metrics for the imputation of the DD$_m$ and DD$_M$ biomarkers (Left - Imputated, Right - Real values)

Fig. 5.8 and Fig, 5.9 show the boxplot graph of *RSE* values for the first of the 10 folds when imputating the NEU%$_m$ and LIN%$_M$, and DD$_m$ and DD$_M$, respectively. The red dots highlighted in the graph represent the patients who had a different classification between real and imputated data.

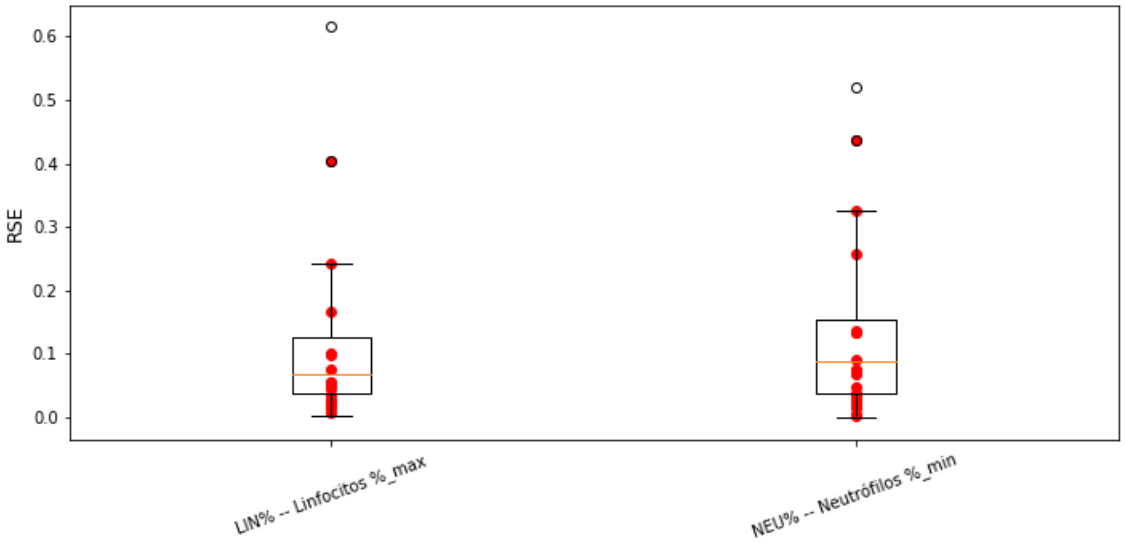

Figure 5.8: *RSE* values for imputated NEU$\%_m$ and LIN$\%_M$ features

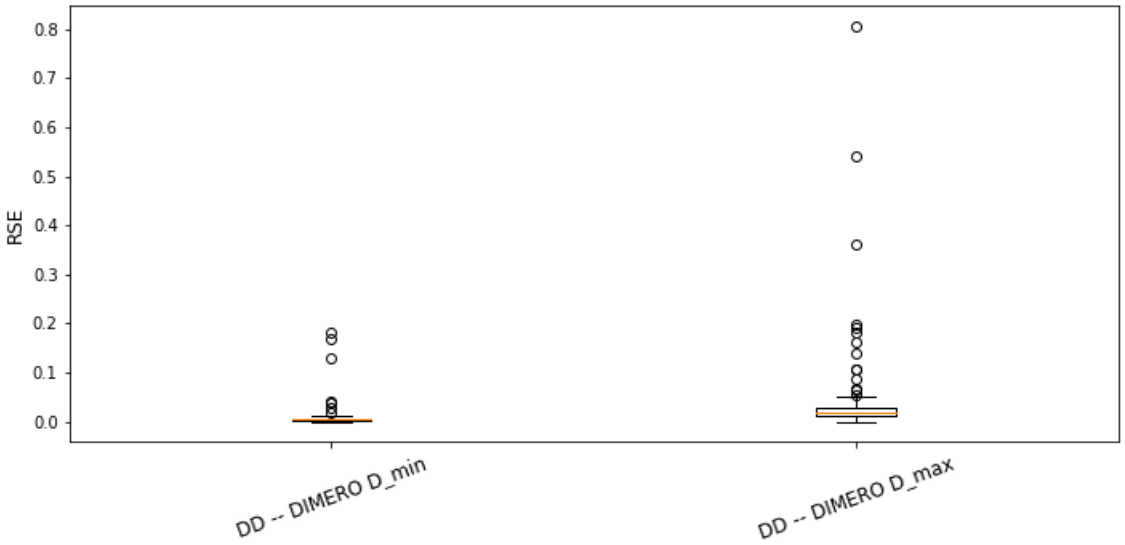

Figure 5.9: *RSE* values for imputated DD$_m$ and DD$_M$ features

5.4 Discussion

From Fig. 5.1 we can observe a clear performance upgrade as we advance from data set to data set. This is expected as the more features are used, the more information is available to better model the system's behaviour. However, an increase in the variability of the results is also observed, because more samples are dropped as more features are added, as shown in Table 3.4. Comparing both approaches on every data set shows that the *SMOTE* approach yields no significant improvement for the model. The proposed *DL* model is capable of making an accurate prediction even on the

unbalanced data set. Further, analyzing the proposed *DL* outcomes distributions, we can see that it is very close to the actual output distribution of the dataset. Fig. 5.10 shows the original database class distribution on the lehft side, while the right side shows the *DL* predicted distribution. This indicates that the proposed model successfully models the data set underlying distribution.

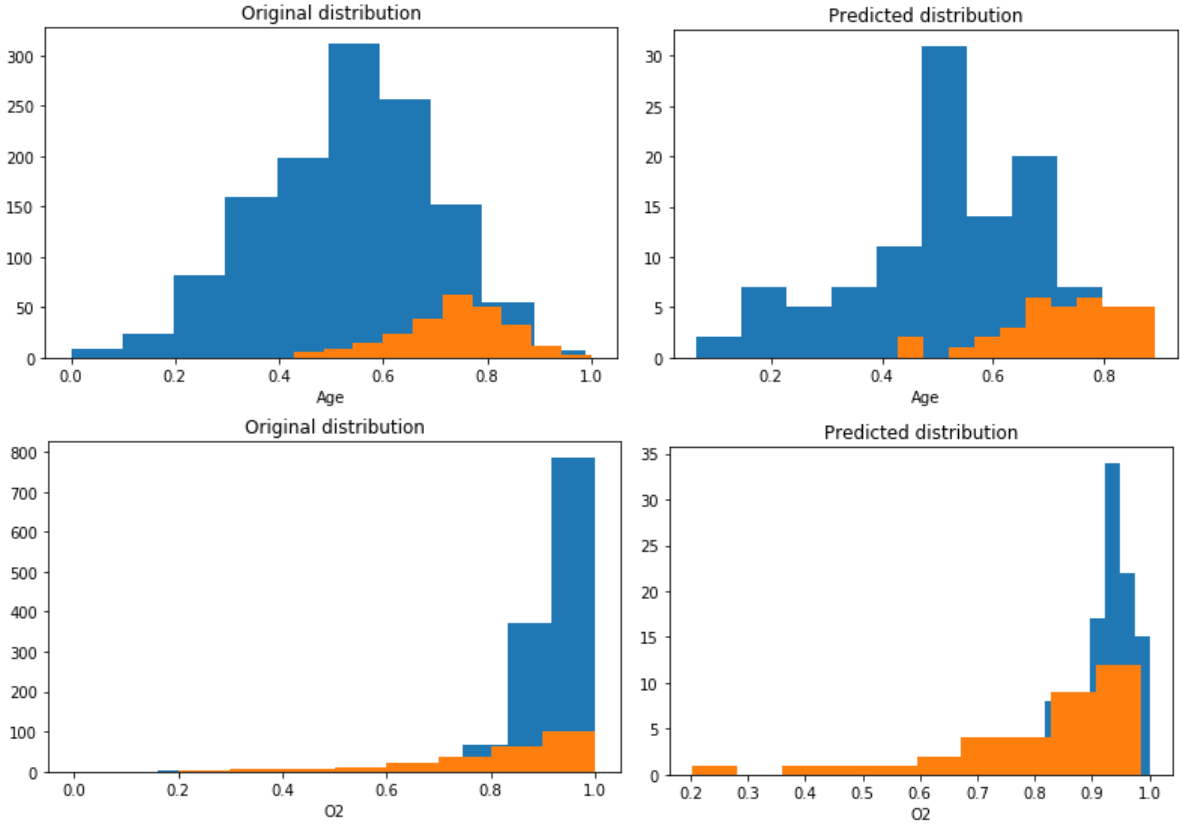

Figure 5.10: Age and SpO2 original (left) and prediction's (right) distributions

Comparing both results for data set 4 (see Fig. 5.2), the *SMOTE* approach has a $Recall$ distribution closer to 1, but more variability for the $Precision$ metric. This is observed on the outlier of the $MPCD$ value near to 0.95 which could suggest a possible improvement on the model's performance when introducing more training data. The same can be said about the final data sets, where the variability of the $MPCD$ score is bigger, but outliers with very high $MPCD$ values are also observed. Finally, the threshold value set by the *SMOTE* approach gets closer to 0.50 because the proportion between classes are closer to one another, as seen in section 4.3.

By comparing the results we can see that the *DL* model outperforms the *RF* model in practically every data set on both approaches. The *RF* model greatly benefits from the *SMOTE* approach, while the *DL* model appears to work better even when having unbalanced classes.

The *Recall* metric is further analyzed to quantify how good our predictions are. As we know from section 2.3.4, the *Recall* metric shows the proportion of the positive samples correctly classified. Fig. 5.11 shows the distribution of the *Recall* metric on every data set without any over-

sampling or imputation methods. For data set we have a mean *Recall* value of 0.92, which means that we have a 92% confidence of correctly classifying any positive prediction. Additionally, we can see folds where the *Recall* value reaches 1.00, indicating that no positive samples were miss-classified.

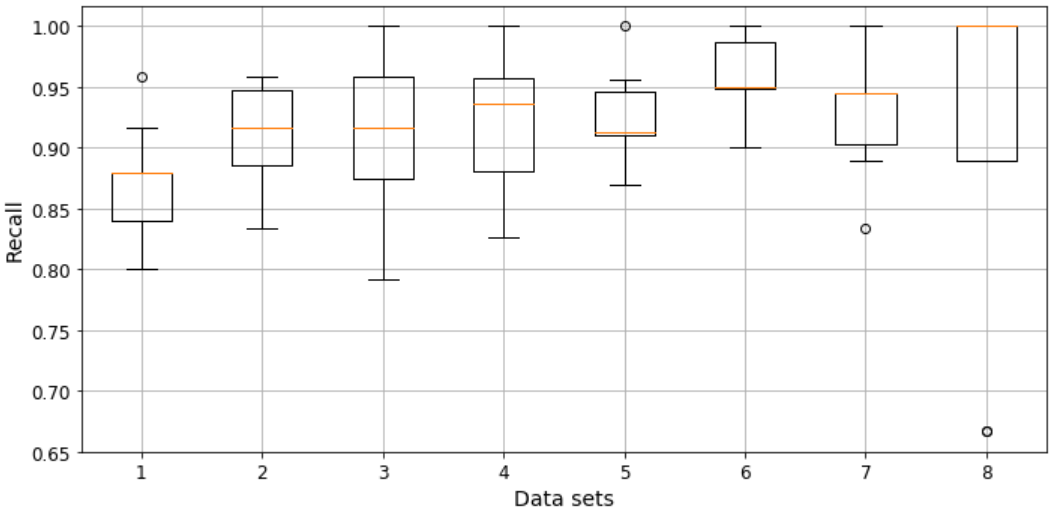

Figure 5.11: Recall score across every data set for the proposed *DL* model

Both imputation results suggest that model's performance can indeed benefit from the imputation of said biomarkers. The *Recall* metric got an overall mean value of around 0.90 which outperforms the 0.87 of the model without any imputations, while also reaching *Recall* values of about 0.95. Fig. 5.9 shows that the overall *RSE* when imputating the DD_m and DD_M features is smaller, with very low variance and just a couple of outliers. Imputating these features should yield a very similar result to the real feature value. Fig. 5.8 has more variance so the imputated value will have more error. Also, we can observe that most of the patients classified differently when using imputation are above the Q3 for the observed error distribution. This can suggest a lack of information in the current database to properly imputate these features.

Chapter 6

Conclusions

There is still much we do not understand about the *CoViD-19* disease. Its high reproduction rate demands hospitals to predict patient's evolution upon admission to efficiently manage hospitals resources. Therefore, a mortality risk calculator must not only accurately classify patients with a high mortality risk, but also do it using only the necessary features.

A mortality risk calculator for *CoViD-19* patients using a *DL* model is proposed. The trade-off between performance and amount of input features is analyzed. This can enable hospitals to make early predictions even when only basic features are available, while evaluating the benefits of later on obtaining more complex biomarker features. The proposed *DL* model using only most basic features had an average $MPCD$ score of 0.70, while the best $MPCD$ score was of 0.85 obtained using 24 input features, 10 basic and 7 biomarker data (both the maximum and minimum values). The proposed model outperformed a *RF* model when evaluating each of the proposed data sets. Both over-sampling and data imputation approaches were analyzed. A data imputation method using the *KNN* algorithm was proposed and evaluated to improve *MPCD*. The proposed imputation strategy improved the *MPCD* (0.75) and *Recall* (0.90) scores while only imputating 2 features.

The analysis presented in this research project can be applied to other research areas, e.g. Finance or manufacture. In the defect detection or prediction problem in the manufacturing area, where the positive (defect) to negative (non.defective) ratio is also very unbalanced, the prediction problem can be analyzed in a similar fashion.

6.1 Contributions

The main contribution of this research project is an analysis of the most important features to use in a *DL* model to predict mortality risk for *CoViD-19* patients and the trade-off between performance, and feature and sample space. The results are compared against a *RF* model to assess its benefits

when attempting to reduce the *FNR* value. Further, the benefits of using over-sampling and data imputation techniques is analyzed. An imputation method using the *KNN* algorithm is proposed and evaluated. While the *SMOTE* over-sampling technique appears to have no real benefit for the model, the proposed imputation method proves to improve the performance of the baseline model.

6.2 Publications

This research project supported the publication of a research paper that will appear in the International Journal on Interactive Design and Manufacturing (IJIDeM) 2020 as **"Data-Driven Risk Prediction Model to Help CoViD19 Patients: a Challenge-based Learning"**

6.3 Future work

- Test models using data from Mexican hospitals.
- Add other type of statistical representation for biomarkers time series data, by standardizing sampling frequency of both vital signs and lab test results.
- Test usage of a time series dedicated algorithm, i.e. Recurrent Neural Networks, ARMA models, etc. to predict patient's evolution trough time.
- Evaluate data imputation efficiency for every biomarker feature in a greedy way.

Bibliography

[Assaf *et al.*, 2020] Dan Assaf, Ya'ara Gutman, Yair Neuman, Gad Segal, Sharon Amit, Shiraz Gefen-Halevi, Noya Shilo, Avi Epstein, Ronit Mor-Cohen, Asaf Biber, et al. Utilization of Machine-Learning Models to Accurately Predict the Risk for Critical COVID-19. *Internal and emergency medicine*, pages 1–9, 2020.

[Bertsimas *et al.*, 2020] Dimitris Bertsimas, Leonard Boussioux, Ryan Cory Wright, Arthur Delarue, Vassilis Digalakis Jr, Alexandre Jacquillat, Driss Lahlou Kitane, Galit Lukin, Michael Lingzhi Li, Luca Mingardi, et al. From Predictions to Prescriptions: A Data-driven Response to COVID-19. *arXiv preprint arXiv:2006.16509*, 2020.

[Breiman, 2001] Leo Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.

[Burdick *et al.*, 2020] Hoyt Burdick, Carson Lam, Samson Mataraso, Anna Lynn-Palevsky, Gregory Braden, R Phillip Dellinger, Andrea McCoy, Jean-Louis Vincent, Abigail Green-Saxena, Gina Barnes, et al. Prediction of Respiratory Decompensation in Covid-19 Patients using Machine Learning: The READY Trial. *Computers in Biology and Medicine*, page 103949, 2020.

[Chawla *et al.*, 2002] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[Cohen *et al.*, 2020] Joseph Paul Cohen, Paul Morrison, and Lan Dao. Covid-19 image data collection. *arXiv preprint arXiv:2003.11597*, 2020.

[CONACYT and DataLab, 2020] GeoInt CONACYT, CentroGeo and DataLab. COVID-19 Tablero México, 2020.

[Cucinotta and Vanelli, 2020] Domenico Cucinotta and Maurizio Vanelli. WHO Declares COVID-19 a Pandemic. *Acta bio-medica: Atenei Parmensis*, 91(1):157–160, 2020.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conf on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Escobar and Morales-Menendez, 2017] Carlos A Escobar and Ruben Morales-Menendez. Machine Learning and Pattern Recognition Techniques for Information Extraction to Improve Production Control and Design Decisions. In *Industrial Conference on Data Mining*, pages 286–300. Springer, 2017.

[Escobar *et al.*, 2020] CA Escobar, D. Macias, and R. Morales-Menendez. Learning with Incomplete Information. *To appear in the Conf Proc of IEEE BigData*, (2020-01-1302), 2020.

[Ferdiansyah *et al.*, 2019] Ferdiansyah Ferdiansyah, Siti Hajar Othman, Raja Zahilah Raja Md Radzi, Deris Stiawan, Yoppy Sazaki, and Usman Ependi. A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market. In *2019 Int Conf on Electrical Eng and Computer Science*, pages 206–210. IEEE, 2019.

[Fernandes, 2020] Nuno Fernandes. Economic Effects of Coronavirus Outbreak (COVID-19) on the World Economy. *Available at SSRN 3557504*, 2020.

[Fraser *et al.*, 2009] Christophe Fraser, Christl A Donnelly, Simon Cauchemez, William P Hanage, Maria D Van Kerkhove, T Déirdre Hollingsworth, Jamie Griffin, Rebecca F Baggaley, Helen E Jenkins, Emily J Lyons, et al. Pandemic Potential of a Strain of Influenza A (H1N1): Early Findings. *science*, 324(5934):1557–1561, 2009.

[Goebel *et al.*, 2012] Kai Goebel, Abhinav Saxena, Matt Daigle, Jose Celaya, Indranil Roychoudhury, and Scott Clements. Introduction to Prognostics. 2012.

[Henning, 2013] Kagermann Henning. Recommendations for implementing the strategic initiative industrie 4.0. 2013.

[Hospitales, 2020] HM Hospitales. Covid Data Save Lives, Apr 2020.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*, 2015.

[Ji *et al.*, 2020] Dong Ji, Dawei Zhang, Jing Xu, Zhu Chen, Tieniu Yang, Peng Zhao, Guofeng Chen, Gregory Cheng, Yudong Wang, Jingfeng Bi, et al. Prediction for Progression Risk in Patients with COVID-19 Pneumonia: the CALL Score. *Clinical Infectious Diseases*, 2020.

[Jiang *et al.*, 2020] Xiangao Jiang, Megan Coffee, Anasse Bari, Junzhang Wang, Xinyue Jiang, Jianping Huang, Jichan Shi, Jianyi Dai, Jing Cai, Tianxiao Zhang, et al. Towards an Artificial Intelligence Framework for Data-Driven Prediction of Coronavirus Clinical Severity. *Computers, Materials & Continua*, 63:537–51, 2020.

[Kermany *et al.*, 2018] Daniel Kermany, Kang Zhang, and Michael Goldbaum. Labeled optical coherence tomography (oct) and chest x-ray images for classification. *Mendeley data*, 2, 2018.

[Khan *et al.*, 2020] YA Khan, SZ Abbas, and Buu-Chau Truong. Machine Learning-based Mortality Rate Prediction using Optimized Hyper-parameter. *Computer Methods and Programs in Biomedicine*, page 105704, 2020.

[Kim *et al.*, 2019] Young Joong Kim, Muhammad Saqlian, and Jong Yun Lee. Deep Learning-based Prediction Model of Occurrences of Major Adverse Cardiac Events during 1-year Follow-up after Hospital Discharge in Patients with AMI using Knowledge Mining. *Personal and Ubiquitous Computing*, pages 1–9, 2019.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Lemaître *et al.*, 2017] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[Li *et al.*, 2014] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient Mini-batch Training for Stochastic Optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014.

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[Ma and Lv, 2019] Xiaomeng Ma and Shuliang Lv. Financial Credit Risk Prediction in Internet Finance Driven by Machine Learning. *Neural Computing and Applications*, 31(12):8359–8367, 2019.

[Max Roser and Hasell, 2020] Esteban Ortiz-Ospina Max Roser, Hannah Ritchie and Joe Hasell. Coronavirus Pandemic (COVID-19). *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[Mohamad and Usman, 2013] Ismail Bin Mohamad and Dauda Usman. Standardization and its Effects on K-means Clustering Algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, 6(17):3299–3303, 2013.

[Nilsson, 2014] Nils J Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[Panwar *et al.*, 2020a] Harsh Panwar, PK Gupta, Mohammad Khubeb Siddiqui, Ruben Morales-Menendez, Prakhar Bhardwaj, and Vaishnavi Singh. A Deep Learning and Grad-CAM based Color Visualization Approach for Fast Detection of COVID-19 Cases using Chest X-ray and CT-Scan Images. *Chaos, Solitons & Fractals*, page 110190, 2020.

[Panwar *et al.*, 2020b] Harsh Panwar, PK Gupta, Mohammad Khubeb Siddiqui, Ruben Morales-Menendez, and Vaishnavi Singh. Application of Deep Learning for Fast Detection of COVID-19 in X-Rays using nCOVnet. *Chaos, Solitons & Fractals*, page 109944, 2020.

[Pattipati *et al.*, 2009] Sankavaram C.and B. Pattipati, A. Kodali, K. Pattipati, M. Azam, S. Kumar, and M. Pecht. Model-based and Data-driven Prognosis of Automotive and Electronic Systems. In *IEEE Int Conf on Auto Science and Eng*, pages 96–101, Aug 2009.

[Pourhomayoun and Shakibi, 2020] Mohammad Pourhomayoun and Mahdi Shakibi. Predicting Mortality Risk in Patients with COVID-19 using Artificial Intelligence to Help Medical Decision-making. *medRxiv*, 2020.

[Read, 2020] Mark Channels Read. EID: High Contagiousness and Rapid Spread of Severe Acute Respiratory Syndrome Coronavirus 2. *Emerg. Infect. Dis*, 26, 2020.

[S. *et al.*, 2020] Vashisth S., A. Linden, J. Hare, and P. Krensky. Hype Cycle for Data Science and Machine Learning, 2020. *Gartners Inc., Stanford, US*, 2020.

[Selvaraju *et al.*, 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-Cam: Visual Explanations from Deep Networks via Gradient-based Localization. In *Proc of the IEEE Int Conf on Computer Vision*, pages 618–626, 2017.

[Soares *et al.*, 2020] Eduardo Soares, Plamen Angelov, Sarah Biaso, Michele Higa Froes, and Daniel Kanda Abe. Sars-cov-2 ct-scan dataset: A large dataset of real patients ct scans for sars-cov-2 identification. *medRxiv*, 2020.

[The Johns Hopkins University, 2020] The Johns Hopkins University. The Johns Hopkins Coronavirus Resource Center. 2020.

[Vachtsevanos, 2006] George Z. Vachtsevanos. *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. John Wiley & Sons, 2006.

[World Health Organization, 2019] World Health Organization. International Statistical Classification of Diseases and Related Health Problems 10^{th} Revision, 2019.

[Wu *et al.*, 2020] Fan Wu, Su Zhao, Bin Yu, Yan-Mei Chen, Wen Wang, Zhi-Gang Song, Yi Hu, Zhao-Wu Tao, Jun-Hua Tian, Yuan-Yuan Pei, et al. A New Coronavirus Associated with Human Respiratory Disease in China. *Nature*, 579(7798):265–269, 2020.

[Yadaw *et al.*, 2020] Arjun S Yadaw, Yan-chak Li, Sonali Bose, Ravi Iyengar, Supinda Bunyavanich, and Gaurav Pandey. Clinical Features of COVID-19 Mortality: Development and Validation of a Clinical Prediction Model. *The Lancet Digital Health*, 2(10):e516–e525, 2020.

[Yan *et al.*, 2019] Ruqiang Yan, Xuefeng Chen, Peng Wang, and Darian M Onchis. Deep Learning for Fault Diagnosis and Prognosis in Manufacturing Systems, 2019.

[Zhao *et al.*, 2019] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. Deep Learning and its Applications to Machine Health Monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019.

[Zhao *et al.*, 2020] Zirun Zhao, Anne Chen, Wei Hou, James M Graham, Haifang Li, Paul S Richman, Henry C Thode, Adam J Singer, and Tim Q Duong. Prediction Model and Risk Scores of ICU Admission and Mortality in COVID-19. *PloS one*, 15(7):e0236618, 2020.

Appendix A

Acronyms Definitions

Table A.1: Acronyms definitions

Acronym	*Description*	*Acronym*	*Description*
Adam	Adaptive Momentum Estimation	FNR	False Negative Rate
AI	Artificial Intelligence	FP	False Positive
AMI	Acute Myocardial Infarction	FPR	False Positive Rate
ANN	Artificial Neural Network	GBM	Gradient Boosted Machine
ARDS	Acute Respiratory Distress Syndrome	GDP	Gross Domestic Product
AUC	Area Under the Curve	GLM	Generalized Linear Model
AWS	Amazon Web Services	GPR	Gaussian Process Regression
CBM	Condition Based Maintenance	GRACE	Global Registry of Acute Coronary Event
CM	Confusion Matrix	GRAD-CAM	Gradient Weighted Class Activation Mapping
CNN	Convolutional Neural Network	GRU	Gated Recurrent Unit
CoViD-19	Coronavirus Disease 2019	I4.0	Industry 4.0
CV	Cross-validation	ICU	Intensive Care Unit
DL	Deep Learning	KNN	K-Nearest Neighbor
DoE	Design of Experiments	KPI	Key Performance Indicator
FFNN	Feed-Forward Neural Network	LSTM	Long-Short Term Memory

Table A.1: Acronyms definitions (Continued)

Acronym	*Description*	*Acronym*	*Description*
FN	False Negative	RUL	Remaining Useful Life
MACE	Major Adverse Cardiac Events	SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
MEWS	Modified Early Warning Score	SGD	Stochastic Gradient Descent
ML	Machine Learning	SHAP	SHapley Additive exPlanations
MLP	Multi-Layer Perceptron	SM	Smart Manufacturing
MPCD	Maximum Probability of Correct Decision	SMOTE	Synthetic Minority Over-sampling TEchnique
NaN	Not a Number	SpO2	Saturation of Peripheral Oxygen
OCTM	Optimal Classification Threshold Method	SVM	Support Vector Machine
PCR	Polymerase Chain Reaction	TN	True Negative
RF	Random Forest	TP	True Positive
RMSE	Root Mean Squared Error	TPR	True Positive Rate
RNN	Recurrent Neural Network	XGBoost	Extreme Gradient Boosting
ROC	Receiver Operating Characteristics		

Appendix B

Variables Descriptions

Table B.1: Variables description

Variable	*Description*
sm	*SMOTE* over-sampled proportion
g_l	Activation function of l^{th} layer
y	True output
$\hat{y}$	Predicted output
$\mathbb{A}_l$	Activated outputs vector for the l^{th} layer
b_l	Bias vector for the l^{th} layer
FN	False Negatives
FP	False Positives
$L(\hat{y}, y)$	Loss function
TN	True Negatives
TP	True Positives
$S_{d\theta}$	Moving average for the second moment gradient
$S_{d\theta}^{corrected}$	Bias corrected moving average for the second moment gradient
$V_{d\theta}$	Moving average for the first moment gradient
$V_{d\theta}^{corrected}$	Bias corrected moving average for the first moment gradient
$\mathbb{W}_l$	Weights matrix for the l^{th} layer
$d\theta$	First moment gradient
$d\theta^2$	Second moment gradient
θ_i	i^{th} parameters
α	Learning rate
β	Level of importance of recall over precision

Table B.1: Variables description (Continued)

Variable	*Description*
β_1	Exponential decay rate for the first moment estimates
β_2	Exponential decay rate for the second moment estimates
ϵ	Small number to prevent any division by zero

Appendix C

Additional features graphs

The following figures show the distribution of data for every biomarker feature as shown in Table 3.3. The x-axis shows the biomarkers feature values grouped in 20 bins, while the y-axis shows the number of samples (patients) that are within the range values defined in each bin.

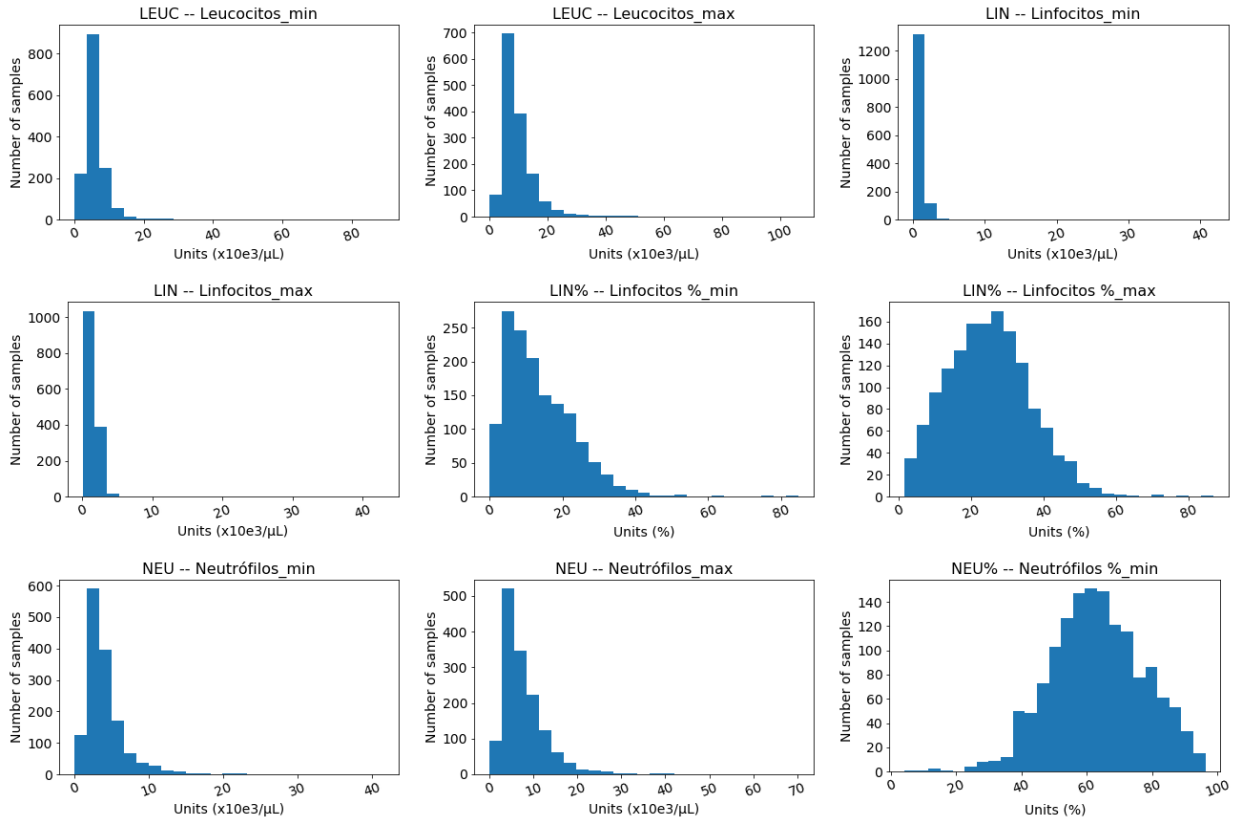

Figure C.1: Data distribution for biomarkers features (part 1)

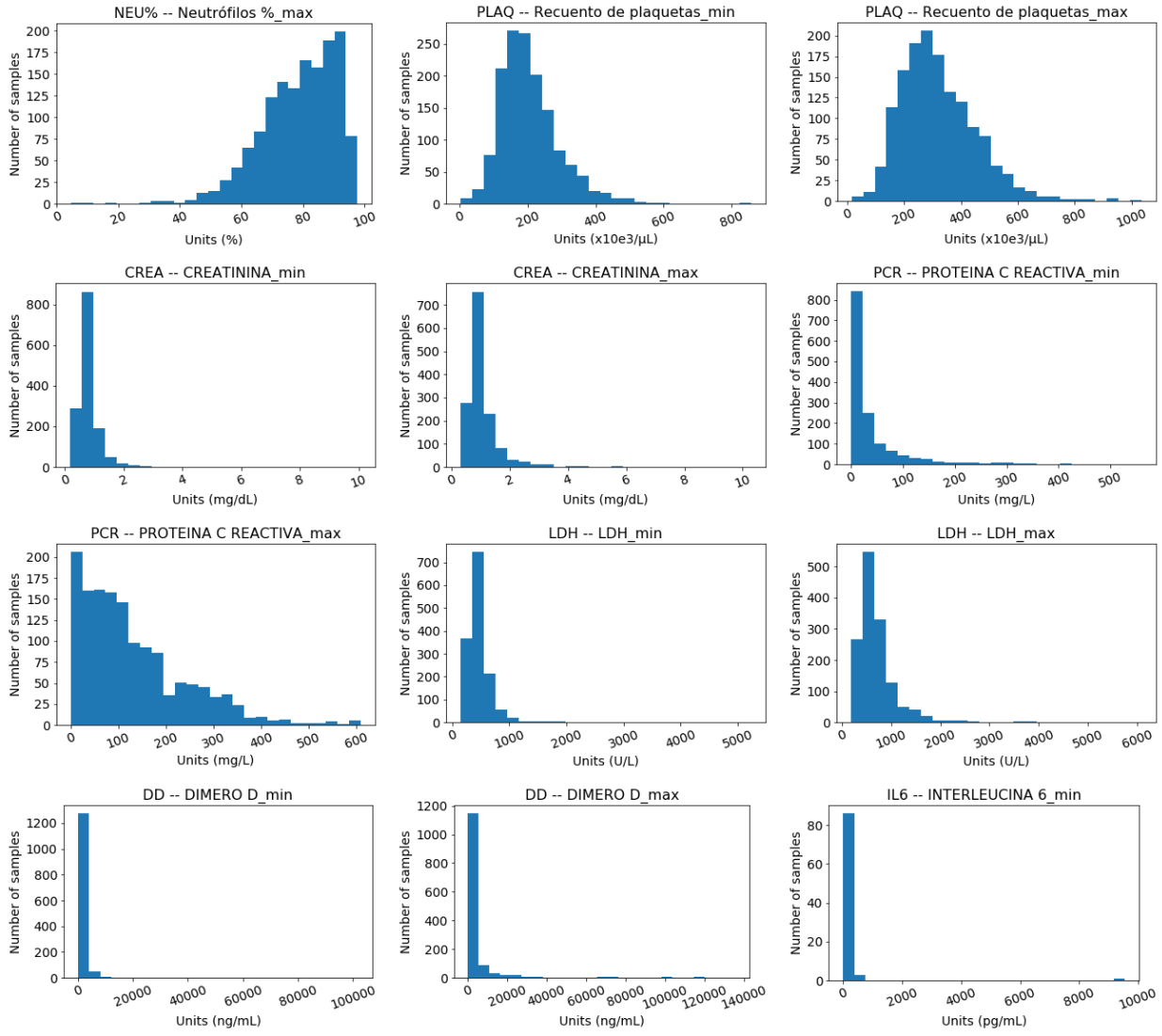

Figure C.2: Data distribution for biomarkers features (part 2)

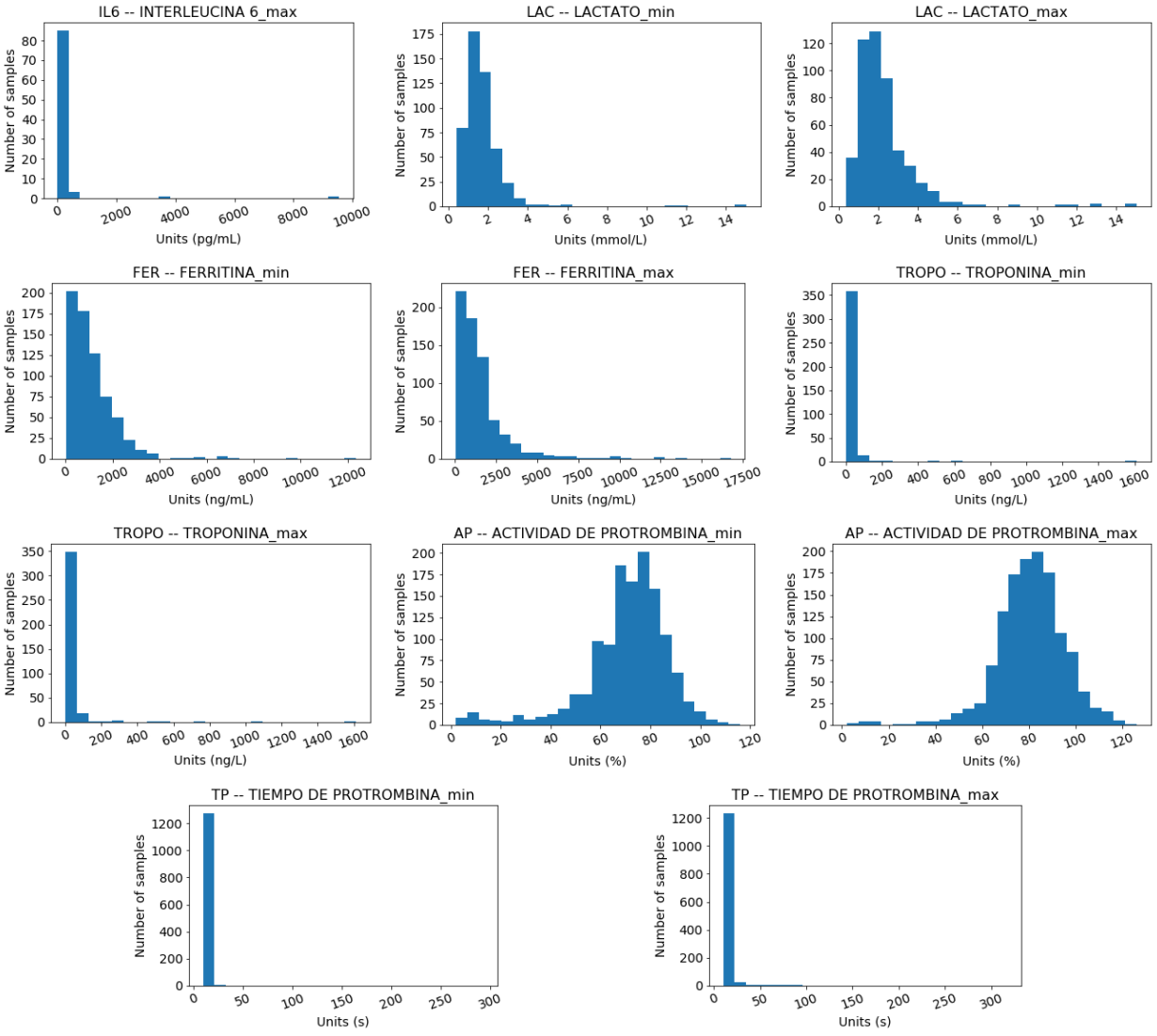

Figure C.3: Data distribution for biomarkers features (part 3)

Appendix D

Sub-data sets results

Section D.1 contains the tables and boxplot figures evaluating the performance of each one of the eight sub data sets on every fold for the *DL* model. Section D.2 shows the same tables and figures for the *RF* model. Results for both the normal data sets and the oversampling implementations are shown.

D.1 *DL* model

D.1.1 Normal dataset

Sub-data set 1

Table D.1: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.73**	**0.68**	**0.73**	**0.61**	**0.71**	**0.68**	**0.73**	**0.68**	**0.74**	**0.72**
Recall	0.84	0.84	0.88	0.92	0.96	0.80	0.84	0.88	0.88	0.88
F2 score	0.76	0.72	0.77	0.69	0.76	0.72	0.76	0.73	0.77	0.76
Precision	0.55	0.47	0.51	0.34	0.41	0.53	0.55	0.43	0.52	0.49
AUC	0.85	0.82	0.86	0.79	0.85	0.83	0.85	0.82	0.86	0.85
Accuracy	0.86	0.81	0.84	0.71	0.77	0.85	0.86	0.79	0.85	0.83
Threshold	0.17	0.19	0.13	0.08	0.16	0.16	0.26	0.14	0.17	0.15

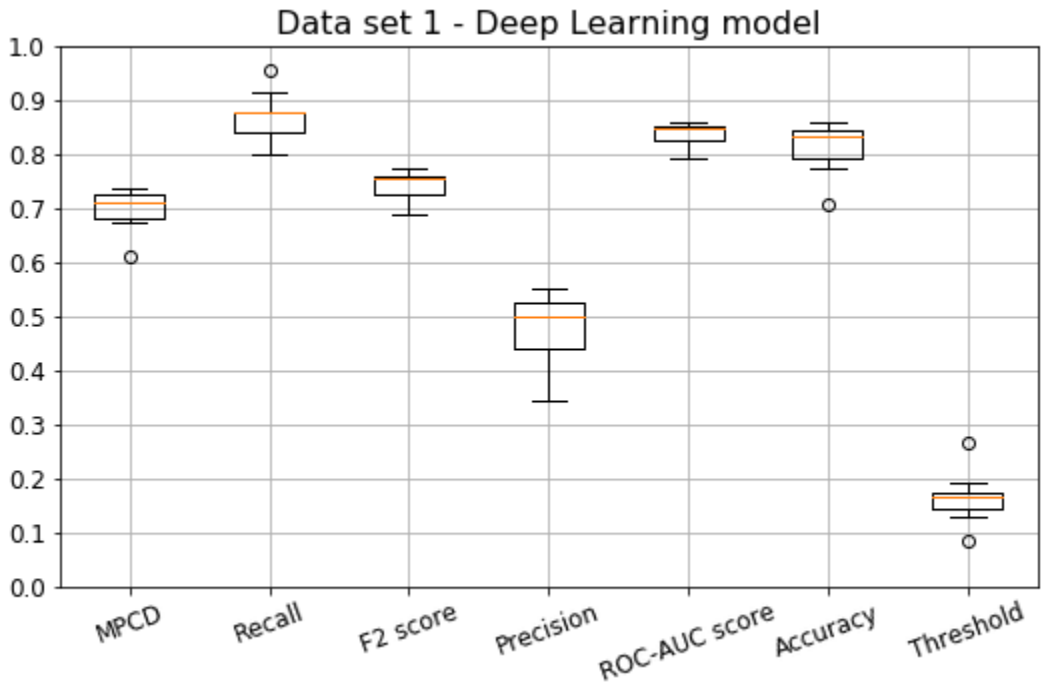

Figure D.1: Boxplot for 10-fold *CV* on data set 1 for the *DL* model

Sub-data set 2

Table D.2: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.81**	**0.80**	**0.78**	**0.88**	**0.76**	**0.75**	**0.77**	**0.89**	**0.78**	**0.78**
Recall	0.92	0.96	0.88	0.96	0.92	0.83	0.92	0.96	0.84	0.92
F2 score	0.83	0.83	0.81	0.89	0.79	0.78	0.80	0.91	0.81	0.81
Precision	0.61	0.53	0.62	0.70	0.51	0.62	0.54	0.74	0.70	0.55
AUC	0.90	0.90	0.88	0.94	0.87	0.87	0.88	0.95	0.88	0.88
Accuracy	0.89	0.86	0.89	0.92	0.84	0.89	0.86	0.94	0.91	0.86
Threshold	0.20	0.06	0.24	0.13	0.08	0.18	0.15	0.11	0.45	0.14

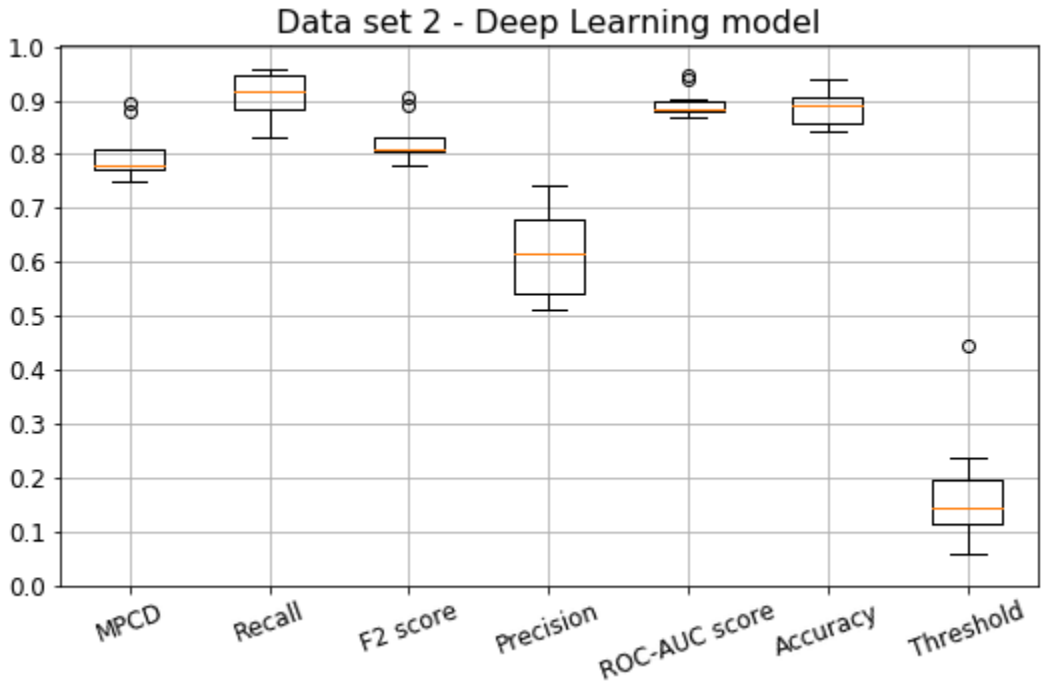

Figure D.2: Boxplot for 10-fold *CV* on data set 2 for the *DL* model

Sub-data set 3

Table D.3: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.91**	**0.81**	**0.82**	**0.73**	**0.69**	**0.81**	**0.83**	**0.77**	**0.86**	**0.86**
Recall	1.00	0.88	0.96	0.92	0.83	0.96	0.88	0.79	0.92	1.00
F2 score	0.92	0.83	0.85	0.77	0.73	0.83	0.85	0.80	0.88	0.88
Precision	0.69	0.70	0.57	0.47	0.49	0.55	0.78	0.83	0.76	0.59
AUC	0.95	0.90	0.91	0.85	0.83	0.90	0.91	0.88	0.93	0.93
Accuracy	0.92	0.92	0.88	0.81	0.83	0.86	0.94	0.94	0.94	0.88
Threshold	0.23	0.30	0.10	0.07	0.09	0.05	0.41	0.40	0.42	0.11

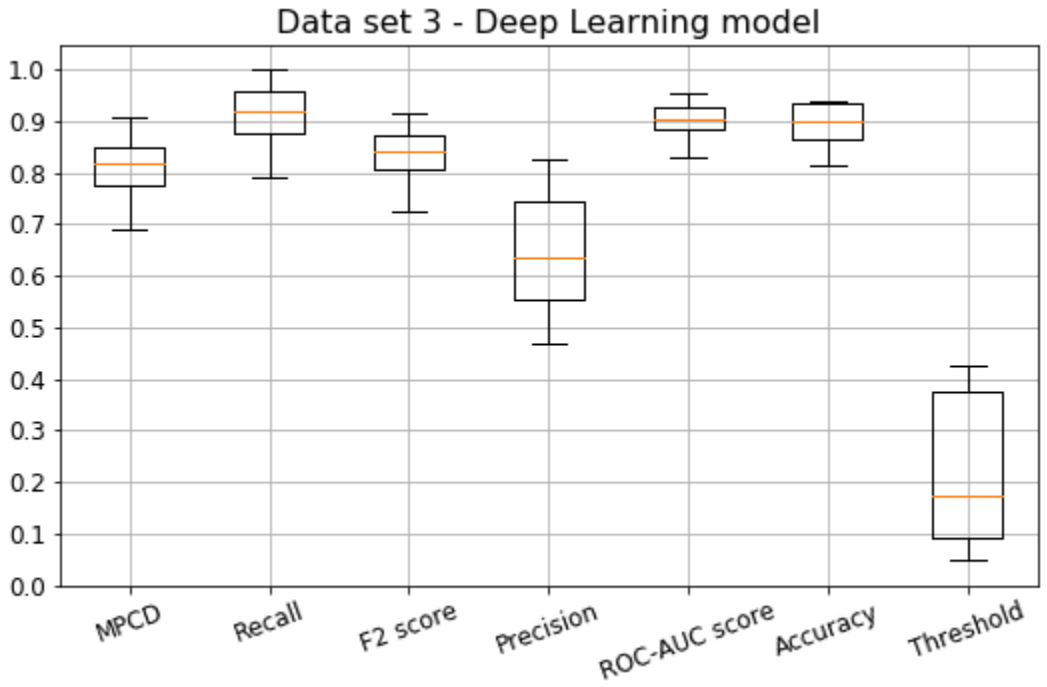

Figure D.3: Boxplot for 10-fold *CV* on data set 3 for the *DL* model

Sub-data set 4

Table D.4: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.85**	**0.84**	**0.91**	**0.90**	**0.87**	**0.82**	**0.77**	**0.80**	**0.75**	**0.90**
Recall	0.96	0.92	0.96	0.96	1.00	0.91	0.87	0.87	0.83	0.96
F2 score	0.87	0.86	0.92	0.91	0.88	0.84	0.80	0.83	0.78	0.91
Precision	0.64	0.69	0.79	0.76	0.61	0.64	0.61	0.69	0.63	0.76
AUC	0.92	0.92	0.95	0.95	0.94	0.91	0.88	0.90	0.87	0.95
Accuracy	0.90	0.92	0.95	0.94	0.89	0.90	0.89	0.92	0.89	0.94
Threshold	0.14	0.27	0.55	0.11	0.06	0.16	0.10	0.17	0.20	0.41

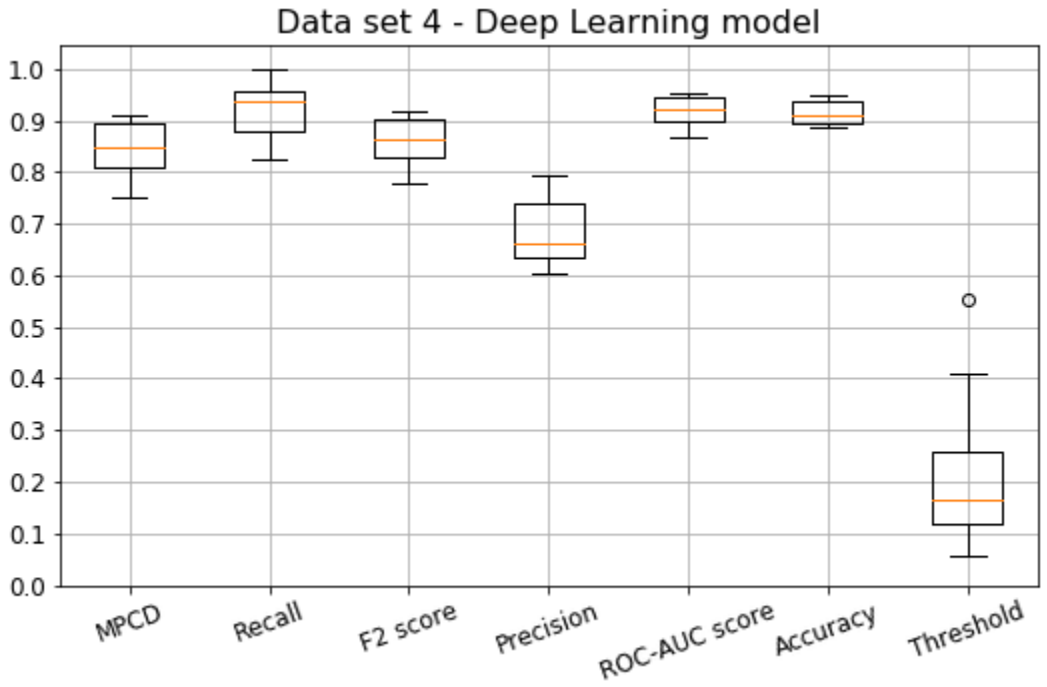

Figure D.4: Boxplot for 10-fold *CV* on data set 4 for the *DL* model

Sub-data set 5

Table D.5: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.80**	**0.77**	**0.83**	**0.85**	**0.83**	**0.89**	**0.84**	**0.92**	**0.86**	**0.87**
Recall	0.96	0.87	0.91	0.91	0.87	0.91	0.91	0.96	0.91	1.00
F2 score	0.83	0.79	0.85	0.87	0.85	0.91	0.85	0.93	0.87	0.88
Precision	0.54	0.59	0.66	0.72	0.77	0.88	0.68	0.85	0.74	0.61
AUC	0.90	0.88	0.91	0.92	0.91	0.94	0.91	0.96	0.92	0.94
Accuracy	0.86	0.88	0.91	0.93	0.94	0.96	0.91	0.96	0.94	0.89
Threshold	0.04	0.12	0.14	0.19	0.45	0.30	0.18	0.69	0.15	0.06

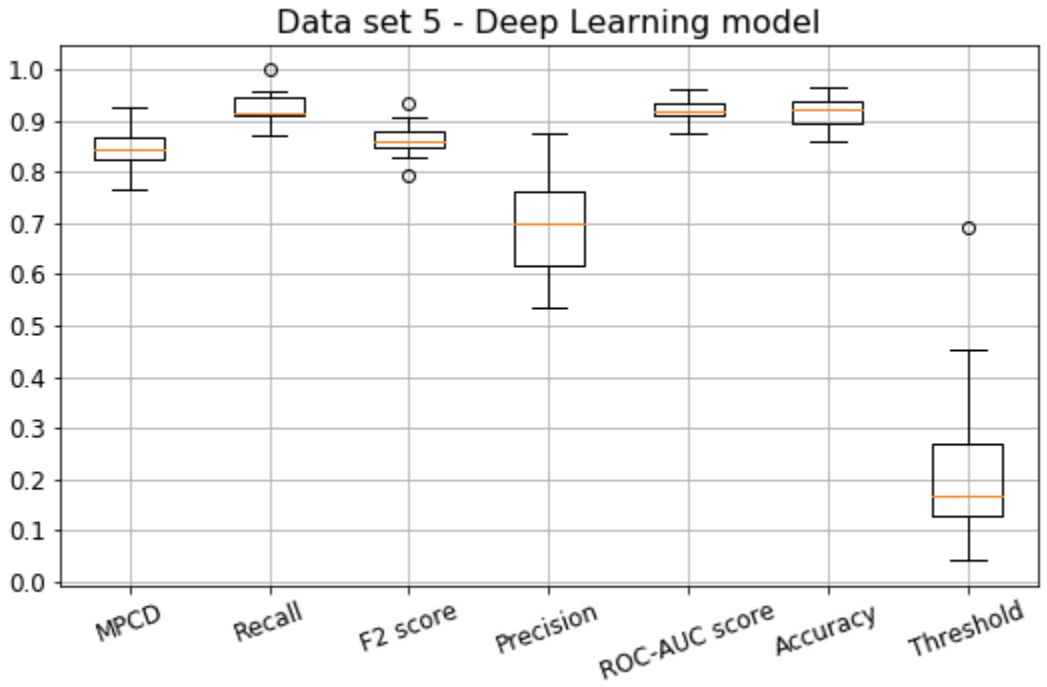

Figure D.5: Boxplot for 10-fold *CV* on data set 5 for the *DL* model

Sub-data set 6

Table D.6: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.87**	**0.93**	**0.88**	**0.93**	**0.87**	**0.79**	**0.88**	**0.83**	**0.81**	**0.79**
Recall	0.95	1.00	1.00	1.00	0.90	0.95	0.95	0.95	0.95	0.90
F2 score	0.87	0.93	0.88	0.93	0.88	0.81	0.89	0.83	0.83	0.80
Precision	0.66	0.71	0.59	0.71	0.82	0.50	0.70	0.56	0.54	0.56
AUC	0.93	0.96	0.94	0.96	0.93	0.89	0.94	0.91	0.90	0.89
Accuracy	0.92	0.94	0.89	0.94	0.95	0.85	0.93	0.89	0.87	0.88
Threshold	0.09	0.14	0.04	0.17	0.38	0.03	0.20	0.05	0.06	0.13

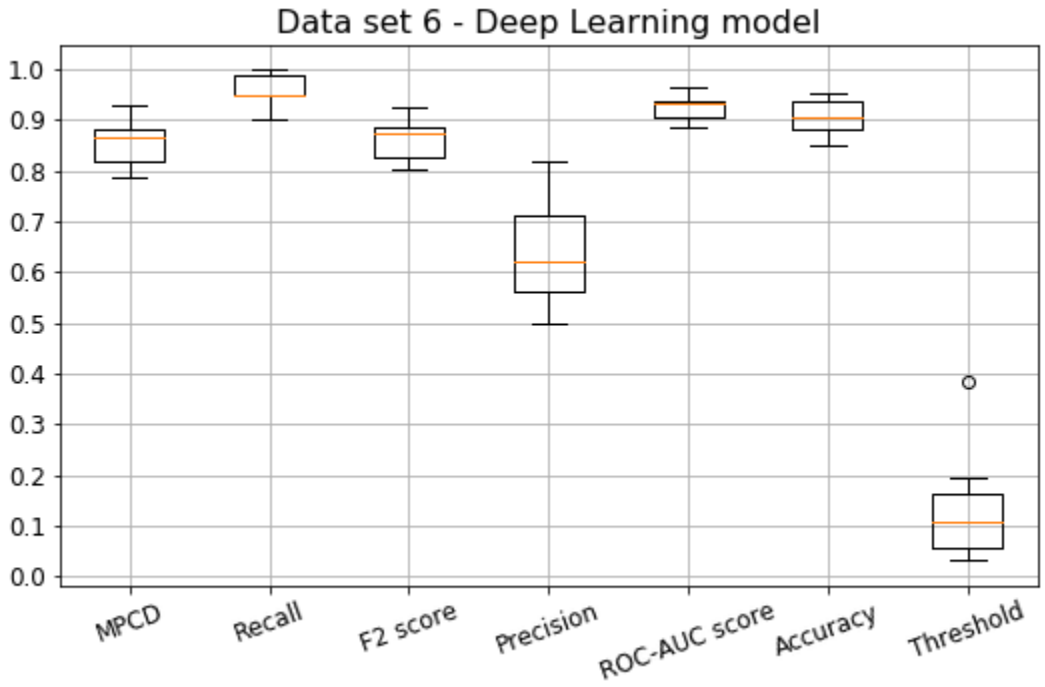

Figure D.6: Boxplot for 10-fold *CV* on data set 6 for the *DL* model

Sub-data set 7

Table D.7: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.85**	**0.82**	**0.88**	**0.88**	**0.70**	**0.86**	**0.90**	**0.91**	**0.87**	**0.89**
Recall	0.89	0.89	1.00	0.94	0.83	0.94	0.94	0.94	0.94	1.00
F2 score	0.86	0.83	0.88	0.89	0.72	0.87	0.90	0.91	0.88	0.89
Precision	0.76	0.67	0.60	0.71	0.47	0.65	0.77	0.81	0.68	0.62
AUC	0.92	0.91	0.94	0.94	0.83	0.93	0.95	0.95	0.93	0.95
Accuracy	0.94	0.92	0.90	0.93	0.83	0.92	0.95	0.96	0.92	0.91
Threshold	0.58	0.26	0.12	0.30	0.01	0.05	0.58	0.25	0.05	0.11

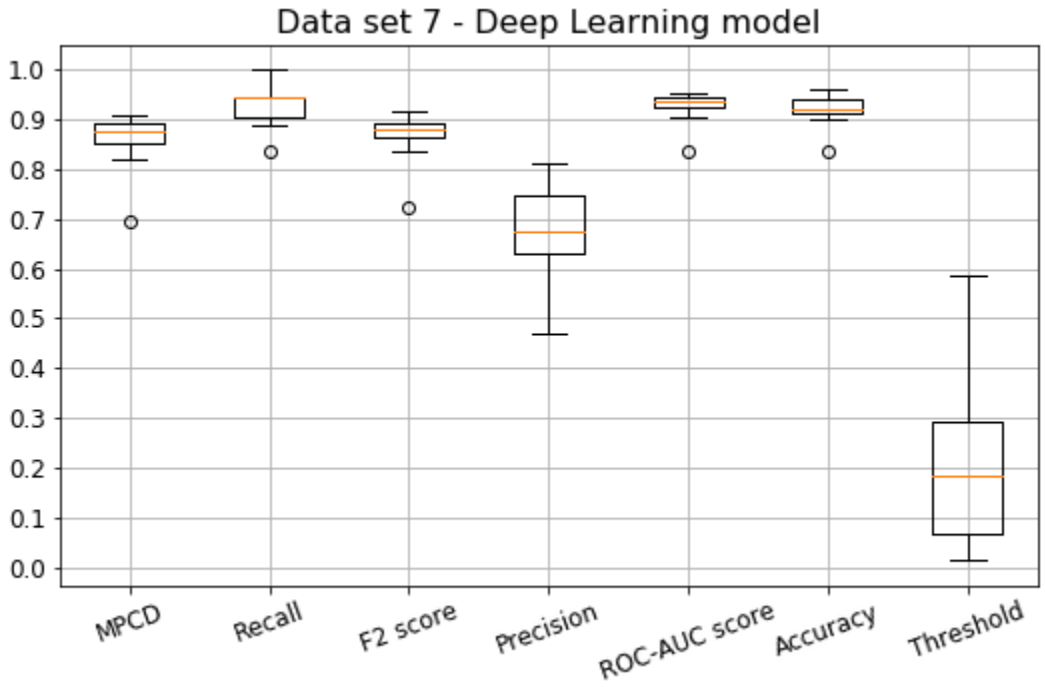

Figure D.7: Boxplot for 10-fold *CV* on data set 7 for the *DL* model

Sub-data set 8

Table D.8: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.79**	**0.91**	**0.60**	**0.83**	**0.79**	**0.96**	**0.87**	**0.88**	**0.60**	**0.96**
Recall	0.89	1.00	0.67	1.00	0.89	1.00	1.00	1.00	0.67	1.00
F2 score	0.80	0.90	0.64	0.83	0.80	0.96	0.87	0.88	0.64	0.96
Precision	0.57	0.64	0.55	0.50	0.57	0.83	0.56	0.60	0.55	0.82
AUC	0.89	0.95	0.79	0.92	0.89	0.98	0.93	0.94	0.79	0.98
Accuracy	0.89	0.92	0.87	0.85	0.89	0.97	0.89	0.90	0.87	0.97
Threshold	0.04	0.05	0.38	0.04	0.25	0.23	0.09	0.10	0.20	0.17

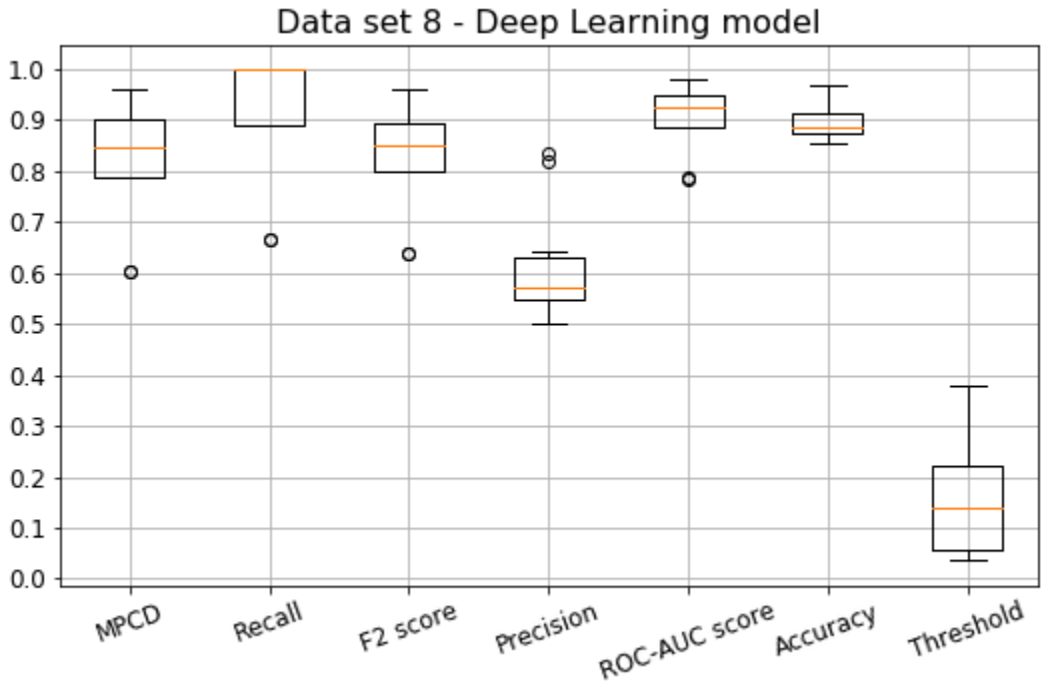

Figure D.8: Boxplot for 10-fold *CV* on data set 8 for the *DL* model

D.1.2 SMOTE dataset

Sub-data set 1

Table D.9: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.69**	**0.76**	**0.80**	**0.63**	**0.67**	**0.62**	**0.72**	**0.68**	**0.70**	**0.75**
Recall	0.76	0.92	0.92	0.92	0.83	0.80	0.88	0.80	0.80	0.88
F2 score	0.73	0.79	0.82	0.70	0.71	0.68	0.76	0.72	0.74	0.79
Precision	0.61	0.51	0.57	0.35	0.45	0.42	0.49	0.53	0.57	0.55
AUC	0.83	0.87	0.89	0.80	0.82	0.79	0.85	0.83	0.84	0.87
Accuracy	0.88	0.84	0.87	0.72	0.81	0.78	0.83	0.85	0.87	0.86
Threshold	0.71	0.57	0.60	0.17	0.42	0.34	0.63	0.51	0.60	0.63

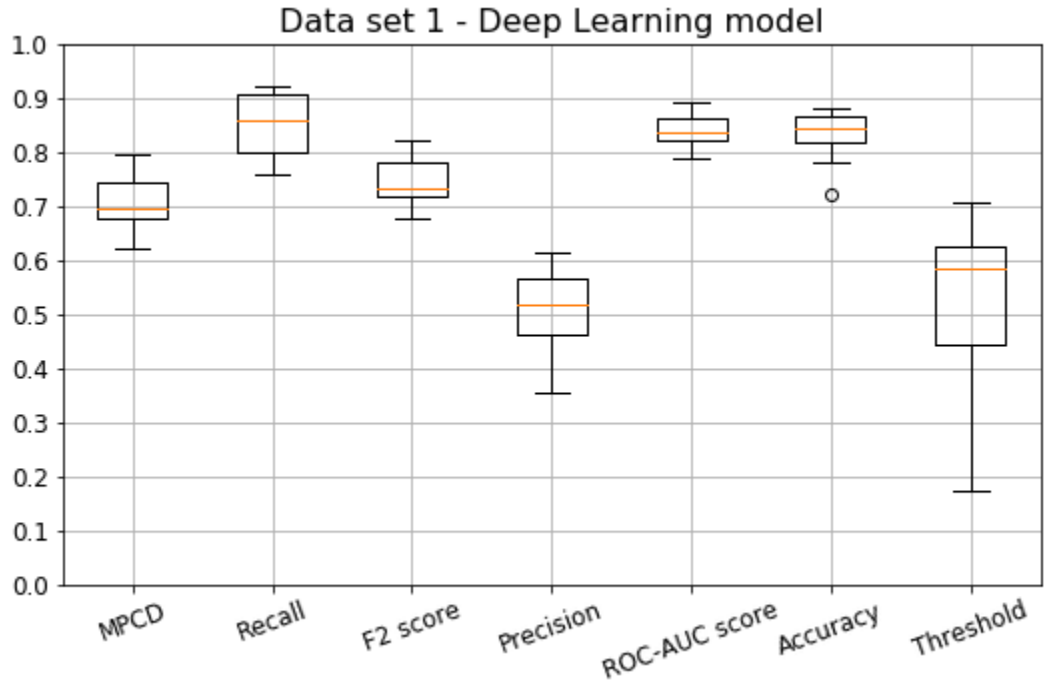

Figure D.9: Boxplot for 10-fold *CV* on data set 1 for the *DL* model

Sub-data set 2

Table D.10: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.76**	**0.85**	**0.80**	**0.84**	**0.83**	**0.79**	**0.81**	**0.81**	**0.85**	**0.73**
Recall	0.88	0.96	0.88	0.92	0.96	0.96	0.96	0.88	1.00	0.79
F2 score	0.79	0.86	0.82	0.86	0.85	0.82	0.83	0.83	0.87	0.76
Precision	0.57	0.62	0.66	0.69	0.59	0.52	0.55	0.70	0.58	0.66
AUC	0.87	0.92	0.89	0.92	0.91	0.89	0.90	0.90	0.92	0.85
Accuracy	0.87	0.90	0.90	0.92	0.88	0.85	0.86	0.92	0.88	0.90
Threshold	0.18	0.57	0.55	0.35	0.46	0.22	0.25	0.67	0.34	0.60

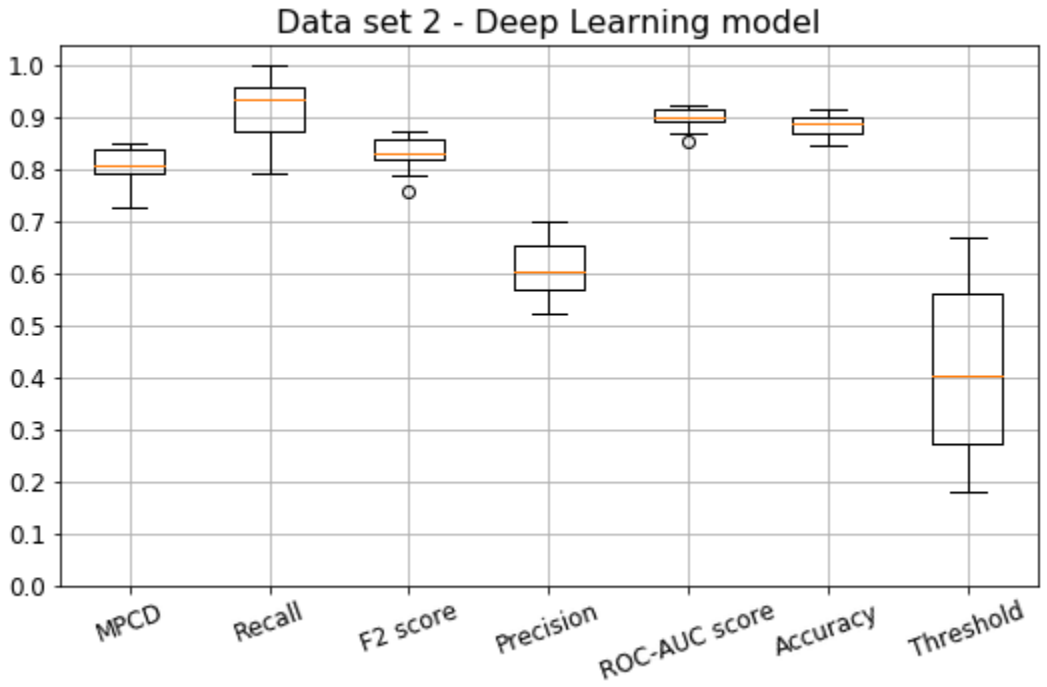

Figure D.10: Boxplot for 10-fold *CV* on data set 2 for the *DL* model

Sub-data set 3

Table D.11: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.83**	**0.75**	**0.85**	**0.86**	**0.79**	**0.76**	**0.79**	**0.80**	**0.87**	**0.76**
Recall	0.96	0.96	0.92	1.00	0.87	0.88	0.88	0.88	1.00	0.88
F2 score	0.85	0.79	0.87	0.88	0.81	0.79	0.81	0.83	0.89	0.79
Precision	0.59	0.47	0.71	0.59	0.65	0.57	0.64	0.68	0.62	0.57
AUC	0.91	0.87	0.92	0.93	0.89	0.87	0.89	0.90	0.94	0.87
Accuracy	0.88	0.81	0.92	0.88	0.90	0.87	0.90	0.91	0.90	0.87
Threshold	0.37	0.41	0.80	0.24	0.43	0.32	0.53	0.41	0.20	0.25

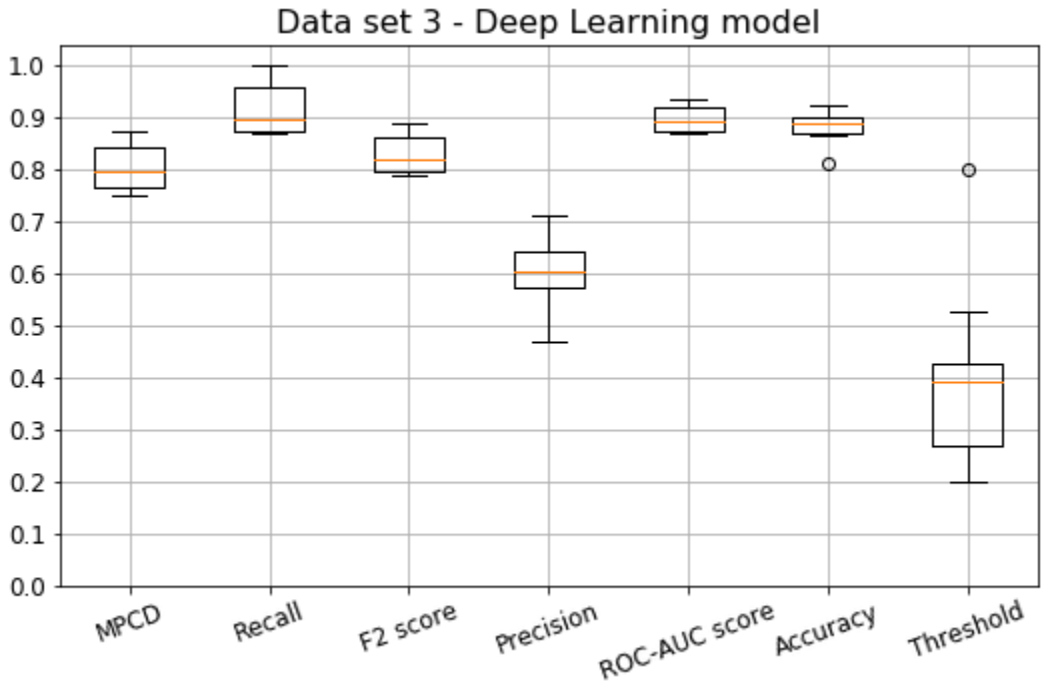

Figure D.11: Boxplot for 10-fold *CV* on data set 3 for the *DL* model

Sub-data set 4

Table D.12: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.79**	**0.87**	**0.86**	**0.75**	**0.81**	**0.81**	**0.83**	**0.92**	**0.85**	**0.97**
Recall	0.83	0.96	0.92	0.91	1.00	0.96	0.96	1.00	0.91	1.00
F2 score	0.82	0.88	0.87	0.78	0.83	0.83	0.85	0.93	0.87	0.97
Precision	0.77	0.68	0.73	0.50	0.50	0.55	0.58	0.72	0.72	0.85
AUC	0.89	0.93	0.92	0.87	0.90	0.90	0.91	0.96	0.92	0.98
Accuracy	0.93	0.92	0.93	0.84	0.84	0.87	0.88	0.94	0.93	0.97
Threshold	0.83	0.20	0.58	0.09	0.21	0.10	0.20	0.47	0.55	0.73

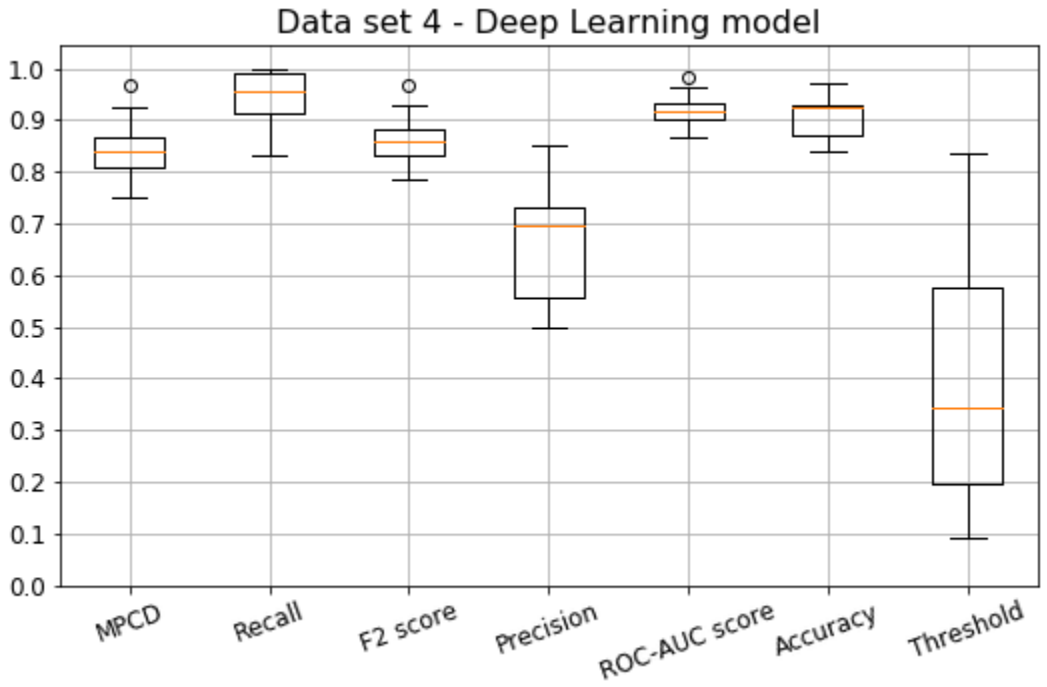

Figure D.12: Boxplot for 10-fold *CV* on data set 4 for the *DL* model

Sub-data set 5

Table D.13: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.80**	**0.89**	**0.78**	**0.90**	**0.77**	**0.88**	**0.89**	**0.85**	**0.85**	**0.88**
Recall	0.83	0.91	0.91	1.00	0.96	0.96	0.96	0.96	0.91	1.00
F2 score	0.83	0.91	0.81	0.91	0.80	0.89	0.90	0.87	0.86	0.89
Precision	0.83	0.88	0.55	0.66	0.49	0.71	0.73	0.63	0.71	0.62
AUC	0.90	0.94	0.88	0.95	0.88	0.94	0.94	0.92	0.92	0.94
Accuracy	0.94	0.96	0.87	0.91	0.83	0.93	0.94	0.90	0.93	0.90
Threshold	0.66	0.66	0.29	0.28	0.12	0.52	0.41	0.11	0.48	0.25

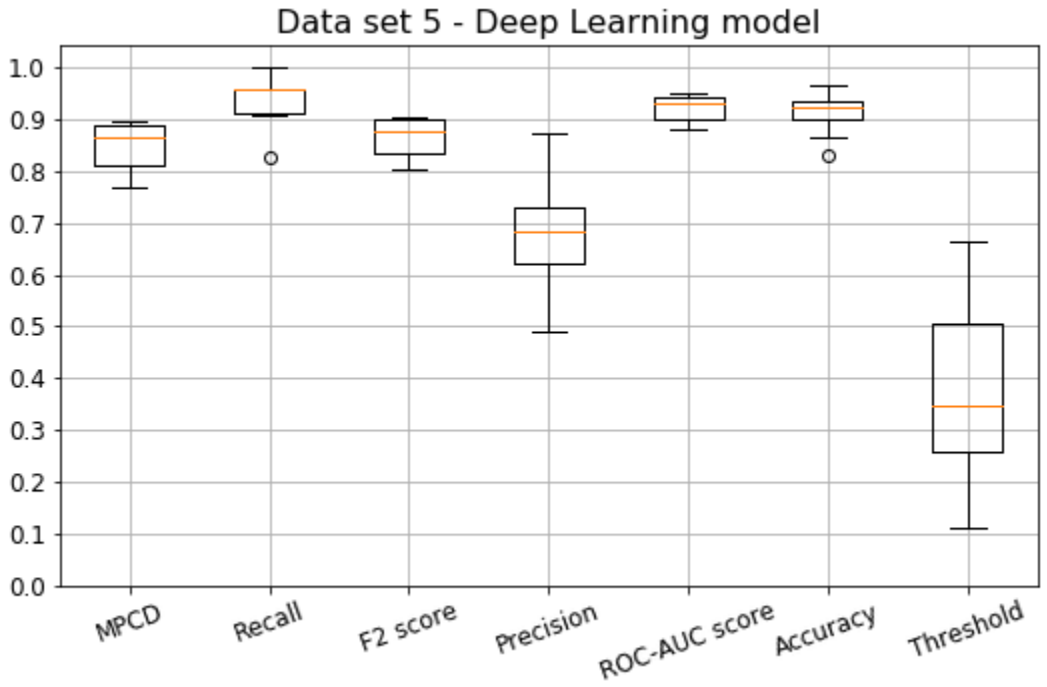

Figure D.13: Boxplot for 10-fold *CV* on data set 5 for the *DL* model

Sub-data set 6

Table D.14: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.87**	**0.84**	**0.83**	**0.80**	**0.87**	**0.92**	**0.89**	**0.80**	**0.82**	**0.91**
Recall	0.90	0.90	0.90	0.90	0.95	0.95	0.95	0.89	0.85	1.00
F2 score	0.88	0.85	0.84	0.82	0.88	0.92	0.90	0.81	0.84	0.91
Precision	0.82	0.69	0.67	0.60	0.68	0.83	0.73	0.59	0.81	0.67
AUC	0.93	0.91	0.91	0.90	0.93	0.96	0.94	0.89	0.91	0.95
Accuracy	0.95	0.92	0.92	0.89	0.92	0.96	0.94	0.89	0.95	0.92
Threshold	0.81	0.42	0.22	0.30	0.23	0.80	0.63	0.31	0.62	0.40

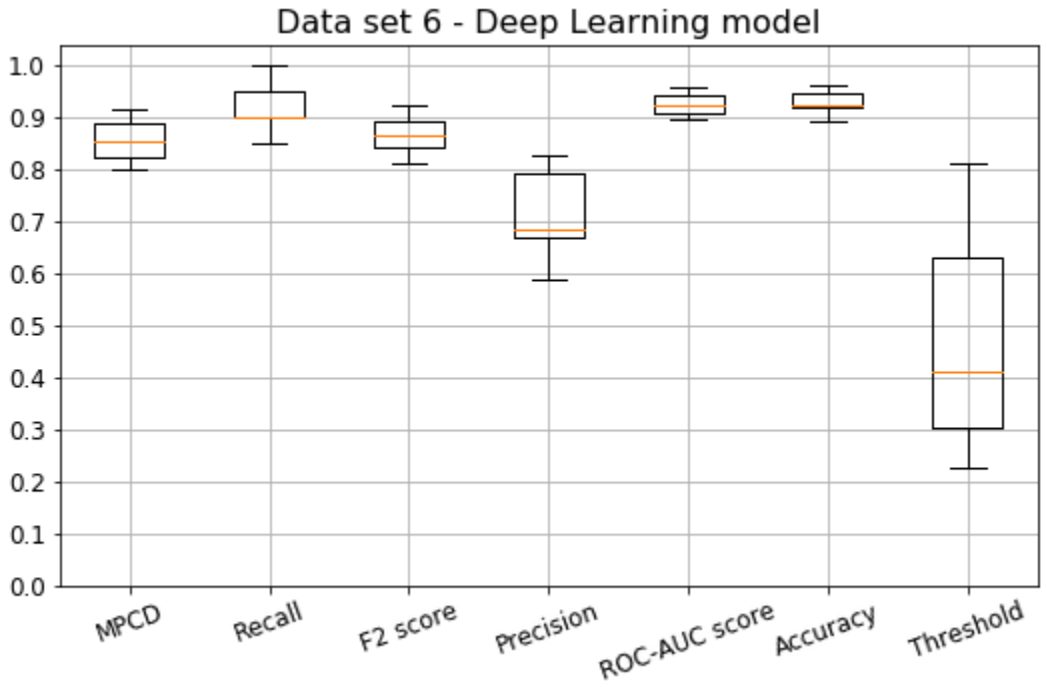

Figure D.14: Boxplot for 10-fold *CV* on data set 6 for the *DL* model

Sub-data set 7

Table D.15: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.81**	**0.90**	**0.89**	**0.96**	**0.88**	**0.84**	**0.82**	**0.73**	**0.88**	**0.80**
Recall	0.89	1.00	1.00	1.00	0.89	1.00	0.94	0.78	0.94	0.89
F2 score	0.82	0.90	0.89	0.96	0.90	0.85	0.83	0.76	0.89	0.82
Precision	0.64	0.64	0.62	0.82	0.94	0.53	0.55	0.70	0.71	0.62
AUC	0.90	0.95	0.95	0.98	0.94	0.92	0.90	0.86	0.94	0.90
Accuracy	0.91	0.92	0.91	0.97	0.98	0.87	0.88	0.92	0.93	0.90
Threshold	0.23	0.22	0.57	0.51	0.88	0.33	0.06	0.84	0.60	0.47

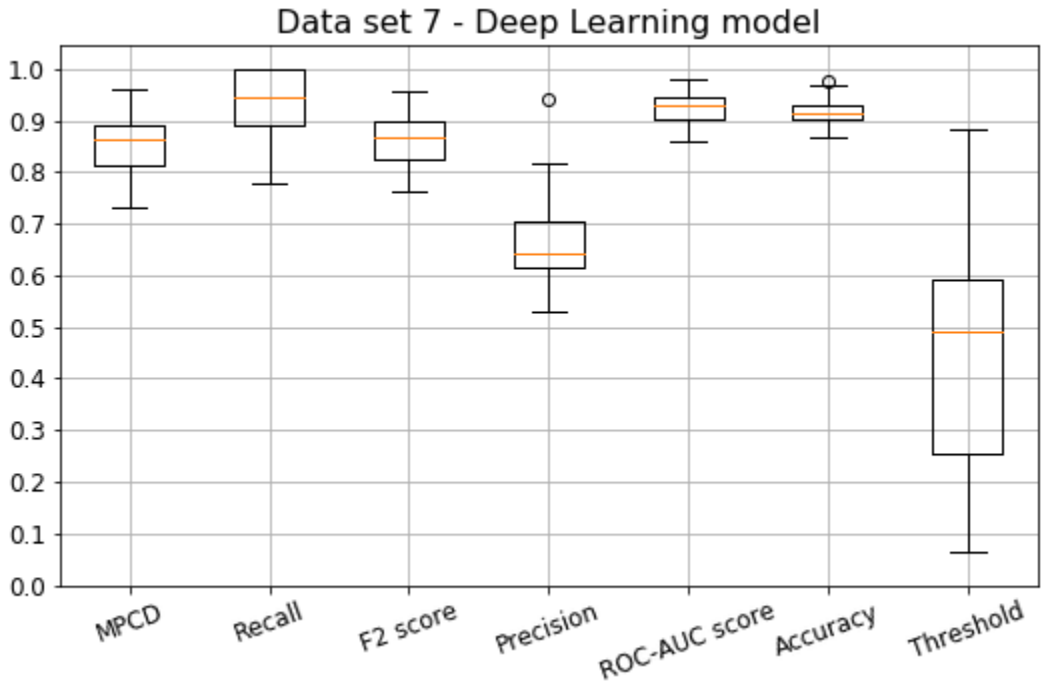

Figure D.15: Boxplot for 10-fold *CV* on data set 7 for the *DL* model

Sub-data set 8

Table D.16: Results for the *DL* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.62**	**0.87**	**0.86**	**0.84**	**0.75**	**0.87**	**0.84**	**0.82**	**0.65**	**0.94**
Recall	0.67	1.00	0.89	0.89	0.78	1.00	0.89	0.89	0.67	1.00
F2 score	0.65	0.87	0.87	0.85	0.78	0.88	0.85	0.83	0.70	0.94
Precision	0.60	0.56	0.80	0.73	0.78	0.59	0.73	0.67	0.86	0.75
AUC	0.80	0.93	0.93	0.92	0.87	0.93	0.92	0.91	0.82	0.97
Accuracy	0.89	0.89	0.95	0.94	0.94	0.89	0.93	0.92	0.93	0.95
Threshold	0.06	0.12	0.53	0.73	0.83	0.73	0.84	0.25	0.85	0.89

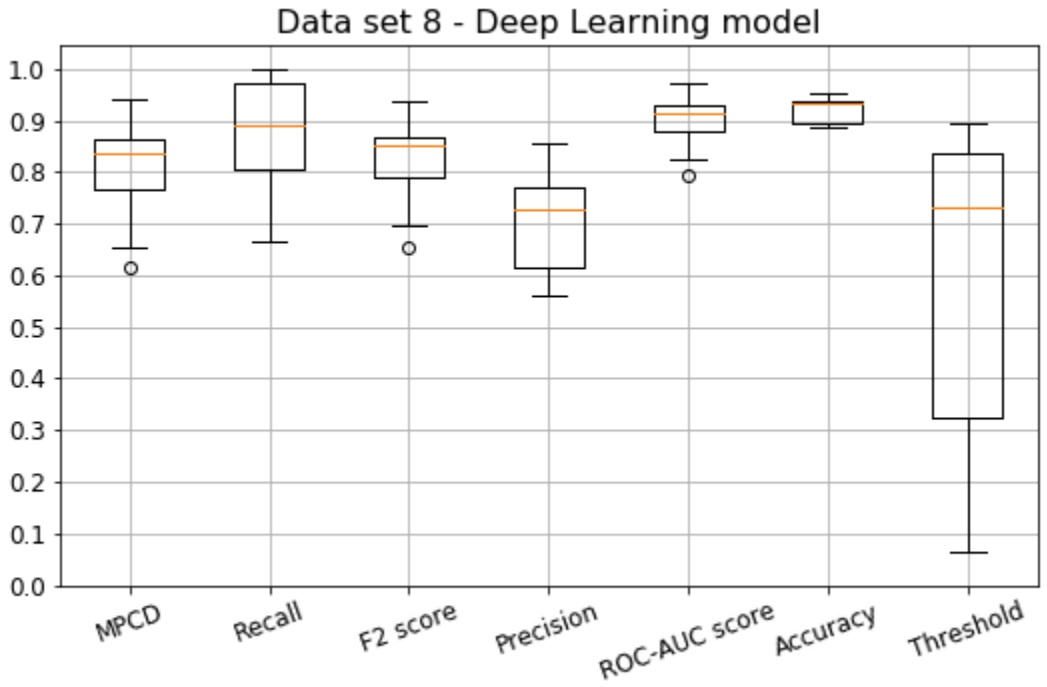

Figure D.16: Boxplot for 10-fold *CV* on data set 8 for the *DL* model

D.2 *RF* model

D.2.1 Normal dataset

Sub-data set 1

Table D.17: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.70**	**0.68**	**0.72**	**0.61**	**0.63**	**0.70**	**0.67**	**0.66**	**0.70**	**0.69**
Recall	0.80	0.88	0.76	0.71	0.88	0.92	0.76	0.72	0.88	0.84
F2 score	0.74	0.73	0.76	0.65	0.69	0.75	0.71	0.70	0.74	0.73
Precision	0.57	0.43	0.76	0.50	0.38	0.43	0.56	0.62	0.46	0.49
AUC	0.84	0.82	0.86	0.79	0.80	0.84	0.82	0.82	0.84	0.83
Accuracy	0.87	0.79	0.92	0.84	0.75	0.79	0.86	0.88	0.81	0.83
Threshold	0.25	0.19	0.31	0.31	0.18	0.13	0.31	0.34	0.13	0.20

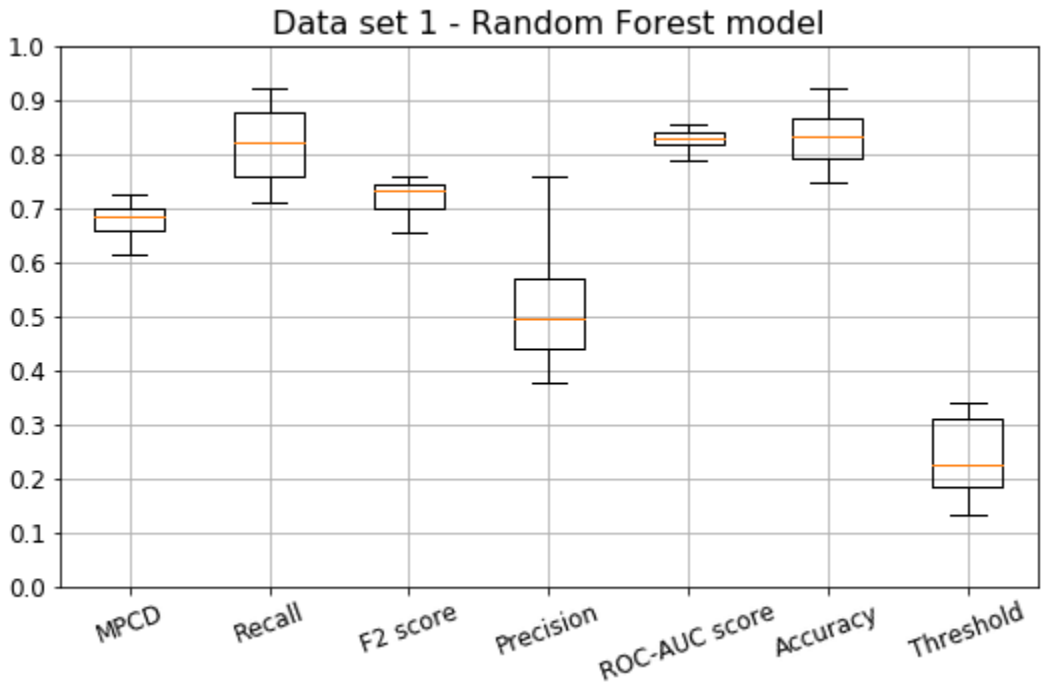

Figure D.17: Boxplot for 10-fold *CV* on data set 1 for the *RF* model

Sub-data set 2

Table D.18: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.74**	**0.82**	**0.72**	**0.77**	**0.73**	**0.66**	**0.62**	**0.82**	**0.81**	**0.76**
Recall	0.96	0.96	0.83	0.88	0.88	0.83	0.71	0.96	1.00	0.83
F2 score	0.78	0.85	0.75	0.80	0.77	0.71	0.66	0.84	0.84	0.79
Precision	0.45	0.57	0.54	0.60	0.51	0.44	0.53	0.56	0.52	0.67
AUC	0.86	0.91	0.85	0.88	0.85	0.81	0.79	0.90	0.90	0.88
Accuracy	0.80	0.88	0.86	0.88	0.84	0.80	0.85	0.87	0.84	0.90
Threshold	0.11	0.14	0.16	0.17	0.15	0.12	0.22	0.15	0.12	0.19

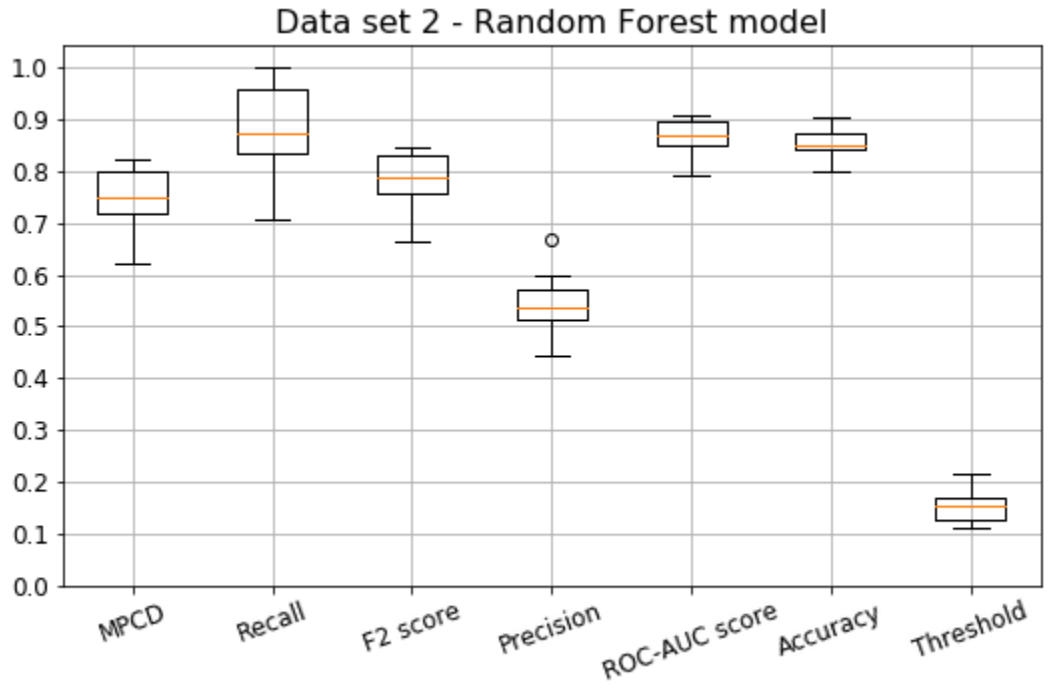

Figure D.18: Boxplot for 10-fold *CV* on data set 2 for the *RF* model

Sub-data set 3

Table D.19: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.79**	**0.74**	**0.76**	**0.77**	**0.66**	**0.76**	**0.72**	**0.75**	**0.76**	**0.72**
Recall	0.96	0.88	0.92	0.96	0.78	0.92	0.92	1.00	0.83	0.83
F2 score	0.82	0.78	0.80	0.80	0.70	0.80	0.77	0.80	0.79	0.76
Precision	0.52	0.54	0.52	0.49	0.49	0.52	0.47	0.44	0.67	0.56
AUC	0.89	0.86	0.87	0.88	0.81	0.87	0.85	0.87	0.87	0.85
Accuracy	0.85	0.85	0.85	0.83	0.83	0.85	0.81	0.79	0.90	0.86
Threshold	0.12	0.16	0.15	0.14	0.15	0.12	0.13	0.13	0.20	0.18

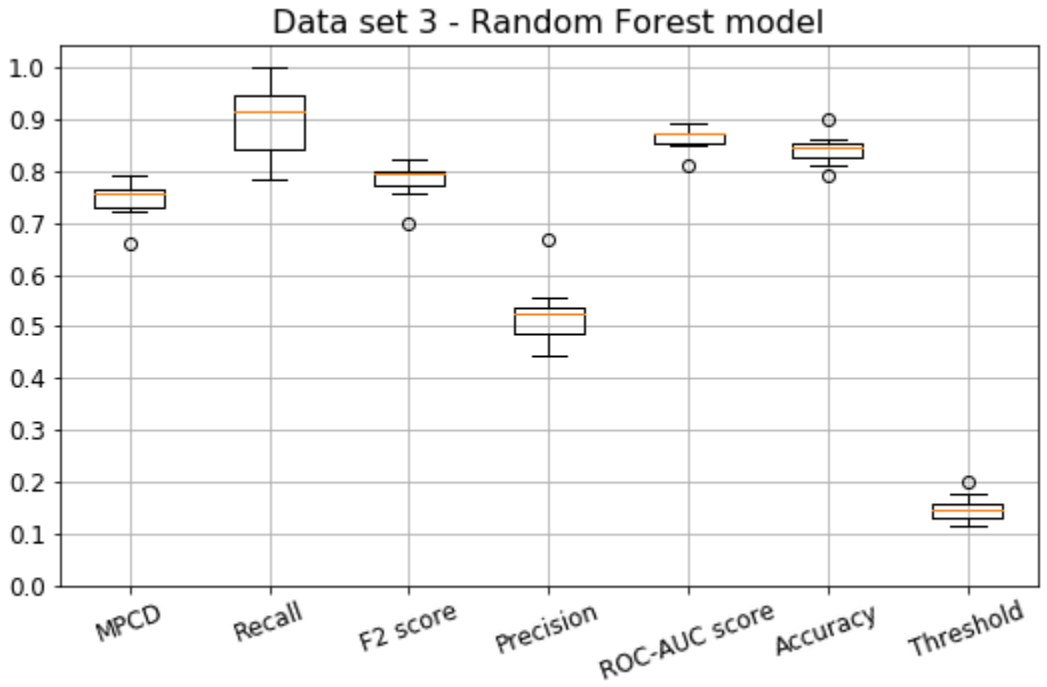

Figure D.19: Boxplot for 10-fold *CV* on data set 3 for the *RF* model

Sub-data set 4

Table D.20: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.81**	**0.82**	**0.73**	**0.74**	**0.81**	**0.75**	**0.70**	**0.70**	**0.64**	**0.84**
Recall	0.92	0.88	0.96	0.91	1.00	1.00	0.96	0.78	0.74	0.91
F2 score	0.83	0.85	0.78	0.77	0.83	0.79	0.75	0.73	0.68	0.85
Precision	0.61	0.75	0.45	0.48	0.50	0.43	0.41	0.58	0.52	0.68
AUC	0.90	0.91	0.86	0.86	0.90	0.87	0.84	0.84	0.80	0.91
Accuracy	0.89	0.93	0.80	0.82	0.84	0.79	0.77	0.87	0.85	0.92
Threshold	0.18	0.19	0.15	0.12	0.14	0.12	0.09	0.21	0.19	0.19

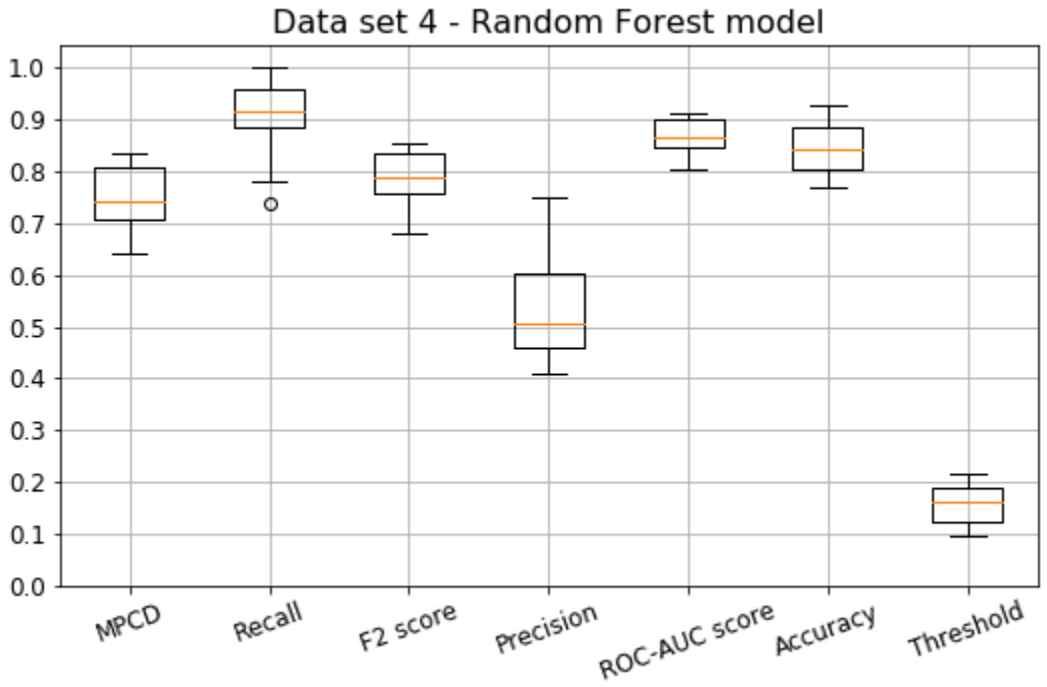

Figure D.20: Boxplot for 10-fold *CV* on data set 4 for the *RF* model

Sub-data set 5

Table D.21: Results for the *RF* model

KPI	*5-fold*	*2-fold*	*3-fold*	*5-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*50-fold*
MPCD	**0.66**	**0.68**	**0.75**	**0.78**	**0.80**	**0.85**	**0.80**	**0.84**	**0.84**	**0.66**
Recall	0.74	0.87	0.91	0.83	0.87	0.91	0.87	0.91	0.95	0.78
F2 score	0.70	0.72	0.78	0.81	0.83	0.87	0.82	0.85	0.85	0.70
Precision	0.57	0.43	0.50	0.76	0.69	0.72	0.67	0.68	0.60	0.49
AUC	0.81	0.82	0.87	0.89	0.90	0.92	0.89	0.91	0.92	0.81
Accuracy	0.87	0.79	0.84	0.93	0.91	0.93	0.91	0.91	0.89	0.83
Threshold	0.20	0.12	0.15	0.24	0.20	0.17	0.19	0.23	0.18	0.20

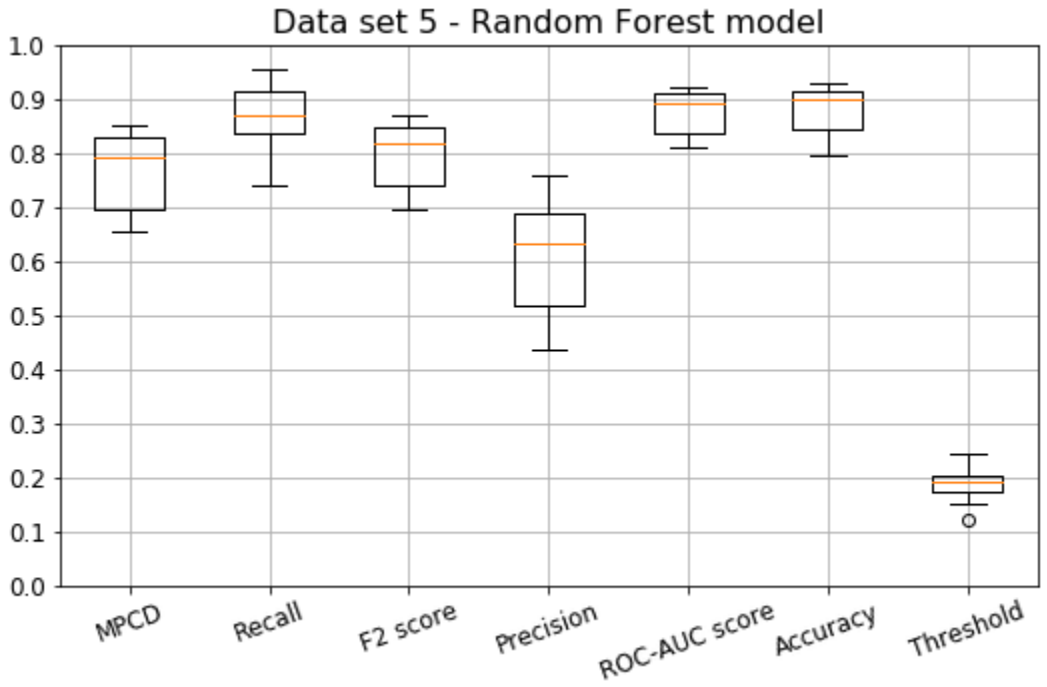

Figure D.21: Boxplot for 10-fold *CV* on data set 5 for the *RF* model

Sub-data set 6

Table D.22: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.66**	**0.68**	**0.75**	**0.78**	**0.80**	**0.85**	**0.80**	**0.84**	**0.84**	**0.66**
Recall	0.74	0.87	0.91	0.83	0.87	0.91	0.87	0.91	0.95	0.78
F2 score	0.70	0.72	0.78	0.81	0.83	0.87	0.82	0.85	0.85	0.70
Precision	0.57	0.43	0.50	0.76	0.69	0.72	0.67	0.68	0.60	0.49
AUC	0.81	0.82	0.87	0.89	0.90	0.92	0.89	0.91	0.92	0.81
Accuracy	0.87	0.79	0.84	0.93	0.91	0.93	0.91	0.91	0.89	0.83
Threshold	0.20	0.12	0.15	0.24	0.20	0.17	0.19	0.23	0.18	0.20

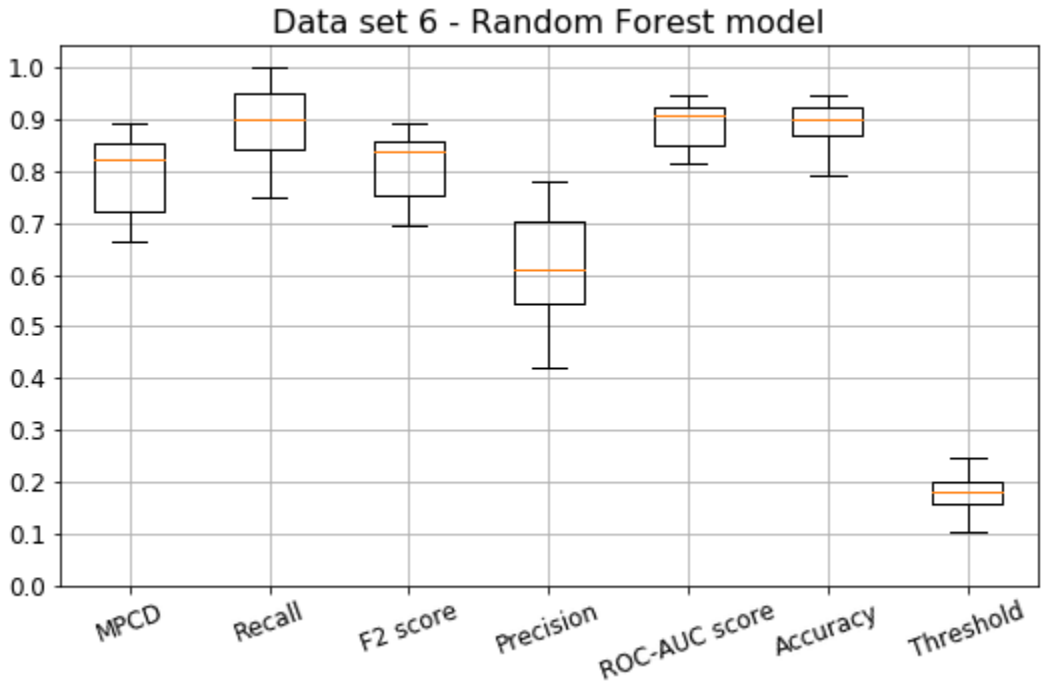

Figure D.22: Boxplot for 10-fold *CV* on data set 6 for the *RF* model

Sub-data set 7

Table D.23: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.84**	**0.74**	**0.74**	**0.82**	**0.70**	**0.65**	**0.72**	**0.83**	**0.87**	**0.82**
Recall	0.89	0.83	0.83	0.89	0.89	0.78	0.94	0.89	0.94	0.89
F2 score	0.85	0.77	0.77	0.83	0.73	0.68	0.75	0.84	0.88	0.83
Precision	0.73	0.58	0.58	0.67	0.42	0.45	0.41	0.70	0.68	0.67
AUC	0.92	0.86	0.86	0.91	0.84	0.81	0.86	0.91	0.93	0.91
Accuracy	0.93	0.88	0.88	0.92	0.80	0.83	0.79	0.92	0.92	0.92
Threshold	0.21	0.17	0.16	0.19	0.12	0.14	0.14	0.20	0.19	0.19

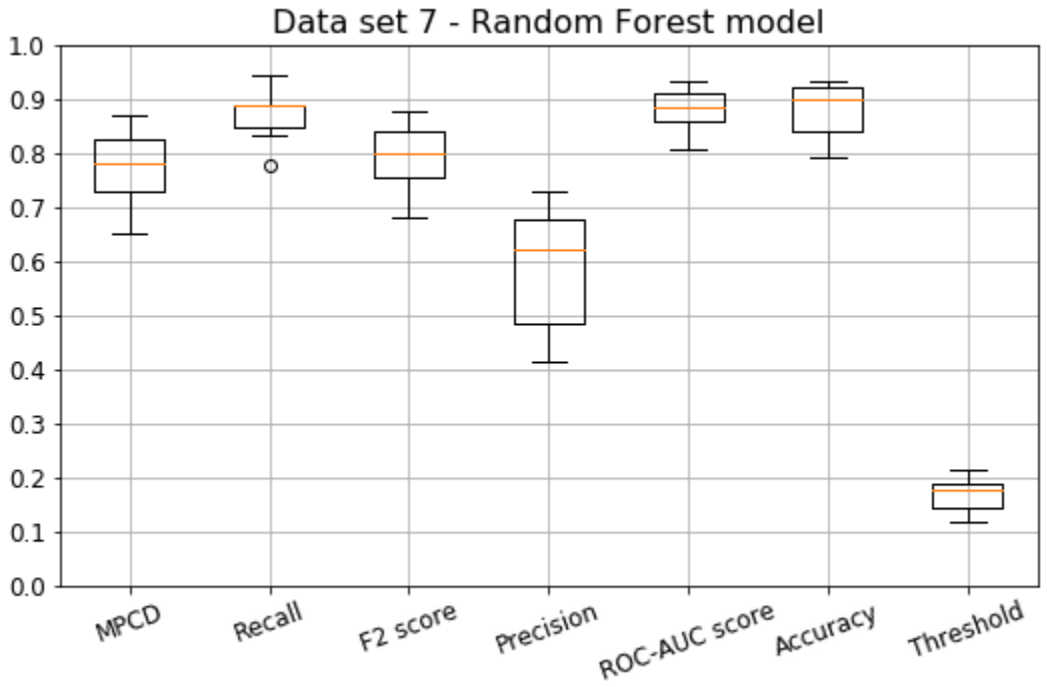

Figure D.23: Boxplot for 10-fold *CV* on data set 7 for the *RF* model

Sub-data set 8

Table D.24: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.84**	**0.98**	**0.62**	**0.77**	**0.81**	**0.98**	**0.75**	**0.92**	**0.79**	**0.85**
Recall	0.89	1.00	1.00	1.00	0.89	1.00	0.78	1.00	0.89	1.00
F2 score	0.85	0.98	0.69	0.79	0.82	0.98	0.78	0.92	0.80	0.85
Precision	0.73	0.90	0.31	0.43	0.62	0.91	0.78	0.69	0.57	0.53
AUC	0.92	0.99	0.81	0.89	0.90	0.99	0.87	0.96	0.89	0.92
Accuracy	0.94	0.98	0.68	0.81	0.90	0.98	0.93	0.93	0.89	0.87
Threshold	0.21	0.26	0.09	0.12	0.24	0.27	0.30	0.16	0.15	0.12

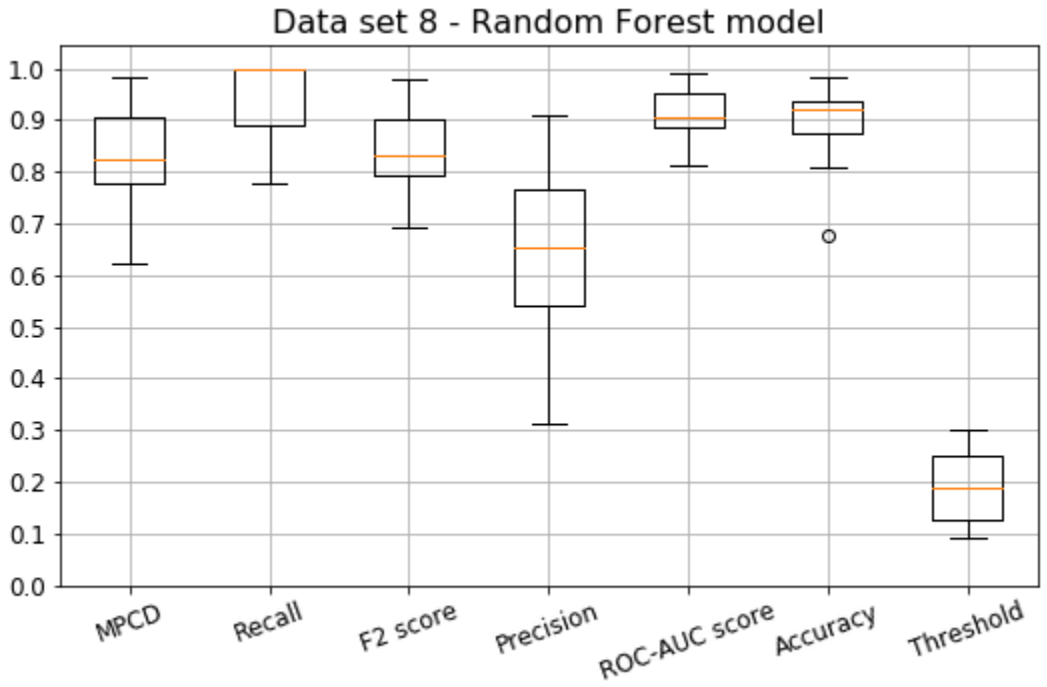

Figure D.24: Boxplot for 10-fold *CV* on data set 8 for the *RF* model

D.2.2 SMOTE dataset

Sub-data set 1

Table D.25: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.69**	**0.73**	**0.76**	**0.60**	**0.63**	**0.58**	**0.70**	**0.62**	**0.62**	**0.67**
Recall	0.80	1.00	0.92	0.79	0.83	0.96	0.84	0.92	0.72	0.76
F2 score	0.73	0.79	0.79	0.65	0.68	0.69	0.74	0.70	0.66	0.71
Precision	0.54	0.42	0.51	0.38	0.40	0.32	0.50	0.36	0.50	0.56
AUC	0.83	0.87	0.87	0.77	0.80	0.78	0.84	0.80	0.79	0.82
Accuracy	0.85	0.77	0.84	0.76	0.77	0.66	0.83	0.71	0.83	0.86
Threshold	0.57	0.39	0.58	0.41	0.52	0.18	0.59	0.23	0.58	0.60

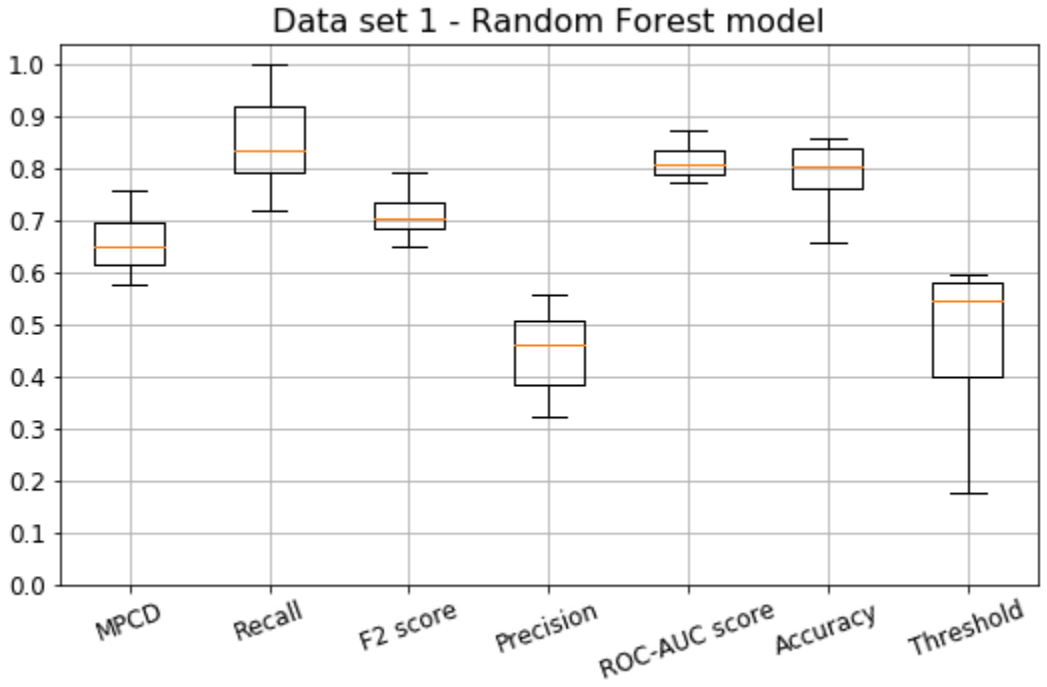

Figure D.25: Boxplot for 10-fold *CV* on data set 1 for the *RF* model

Sub-data set 2

Table D.26: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.66**	**0.69**	**0.67**	**0.67**	**0.78**	**0.74**	**0.70**	**0.69**	**0.80**	**0.75**
Recall	0.71	0.92	0.79	0.83	1.00	0.83	0.88	0.88	0.92	0.92
F2 score	0.70	0.74	0.71	0.71	0.82	0.77	0.74	0.74	0.83	0.79
Precision	0.68	0.42	0.50	0.45	0.47	0.59	0.47	0.46	0.59	0.50
AUC	0.82	0.83	0.82	0.82	0.89	0.86	0.84	0.83	0.89	0.87
Accuracy	0.90	0.78	0.83	0.81	0.81	0.88	0.81	0.81	0.88	0.83
Threshold	0.50	0.35	0.37	0.34	0.33	0.53	0.39	0.37	0.44	0.34

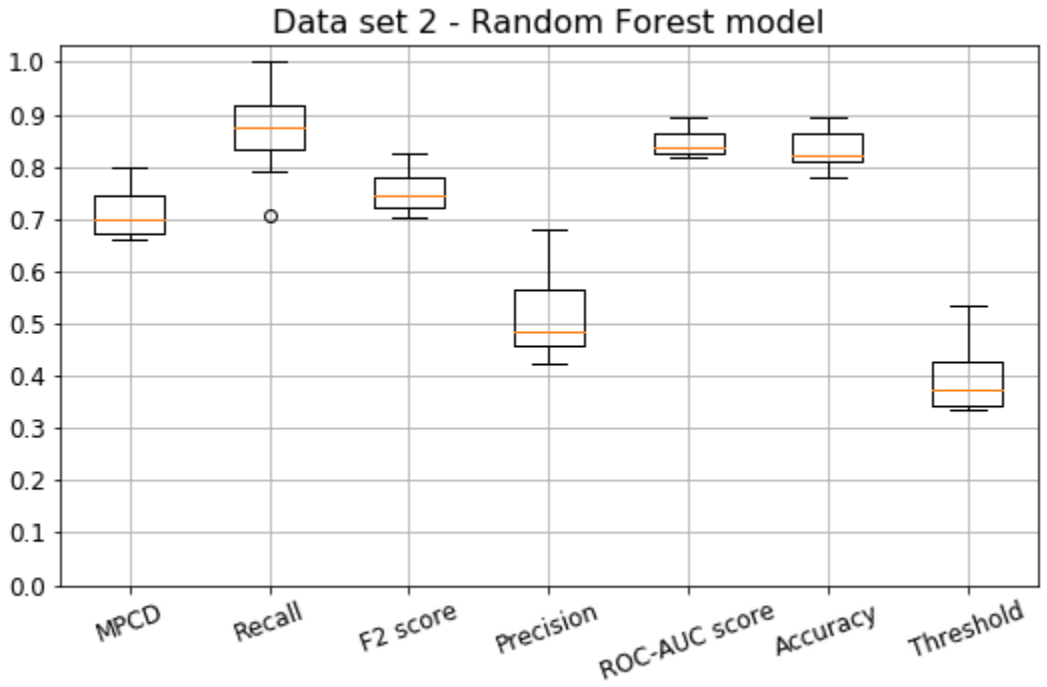

Figure D.26: Boxplot for 10-fold *CV* on data set 2 for the *RF* model

Sub-data set 3

Table D.27: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.77**	**0.78**	**0.72**	**0.85**	**0.75**	**0.69**	**0.72**	**0.66**	**0.78**	**0.69**
Recall	0.96	0.88	0.83	0.88	0.87	0.83	0.96	0.96	1.00	0.83
F2 score	0.81	0.81	0.75	0.88	0.78	0.73	0.77	0.74	0.82	0.74
Precision	0.50	0.62	0.54	0.88	0.56	0.49	0.43	0.38	0.48	0.50
AUC	0.88	0.88	0.85	0.92	0.87	0.83	0.85	0.82	0.89	0.83
Accuracy	0.83	0.89	0.85	0.96	0.87	0.83	0.78	0.73	0.82	0.83
Threshold	0.34	0.47	0.46	0.62	0.44	0.35	0.32	0.28	0.31	0.41

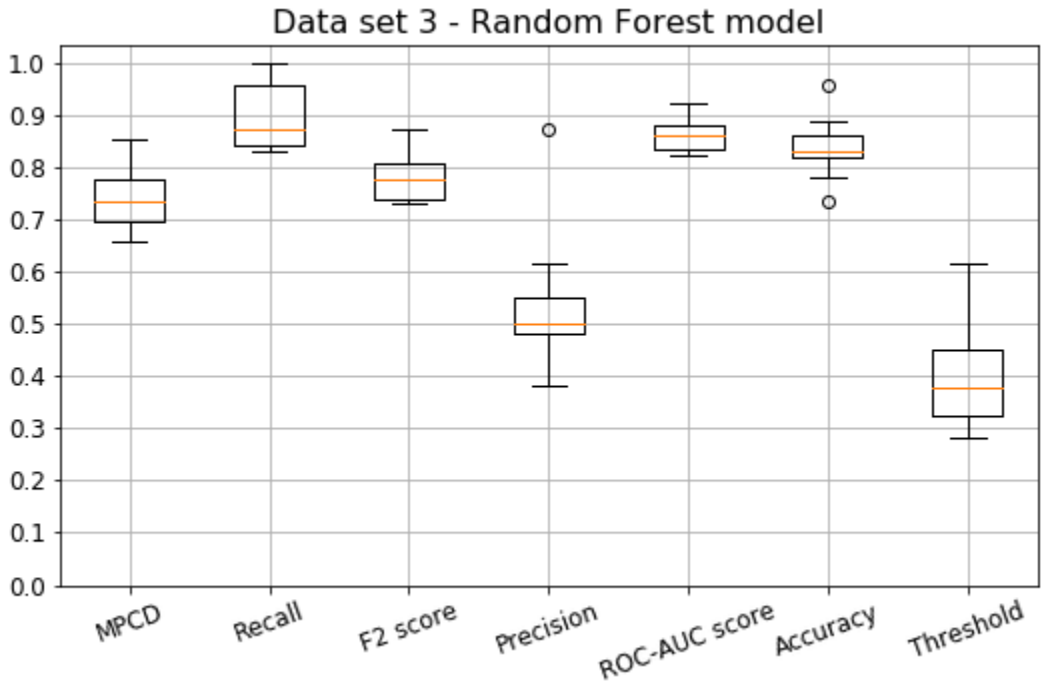

Figure D.27: Boxplot for 10-fold *CV* on data set 3 for the *RF* model

Sub-data set 4

Table D.28: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.73**	**0.75**	**0.82**	**0.84**	**0.72**	**0.72**	**0.68**	**0.77**	**0.83**	**0.89**
Recall	0.79	0.79	1.00	0.96	0.96	0.96	0.78	0.87	1.00	1.00
F2 score	0.77	0.78	0.85	0.86	0.76	0.77	0.71	0.80	0.85	0.90
Precision	0.68	0.73	0.53	0.61	0.42	0.43	0.53	0.61	0.53	0.64
AUC	0.86	0.87	0.91	0.92	0.85	0.86	0.82	0.88	0.92	0.95
Accuracy	0.90	0.92	0.85	0.89	0.78	0.79	0.85	0.89	0.86	0.91
Threshold	0.47	0.54	0.40	0.43	0.34	0.34	0.40	0.49	0.36	0.44

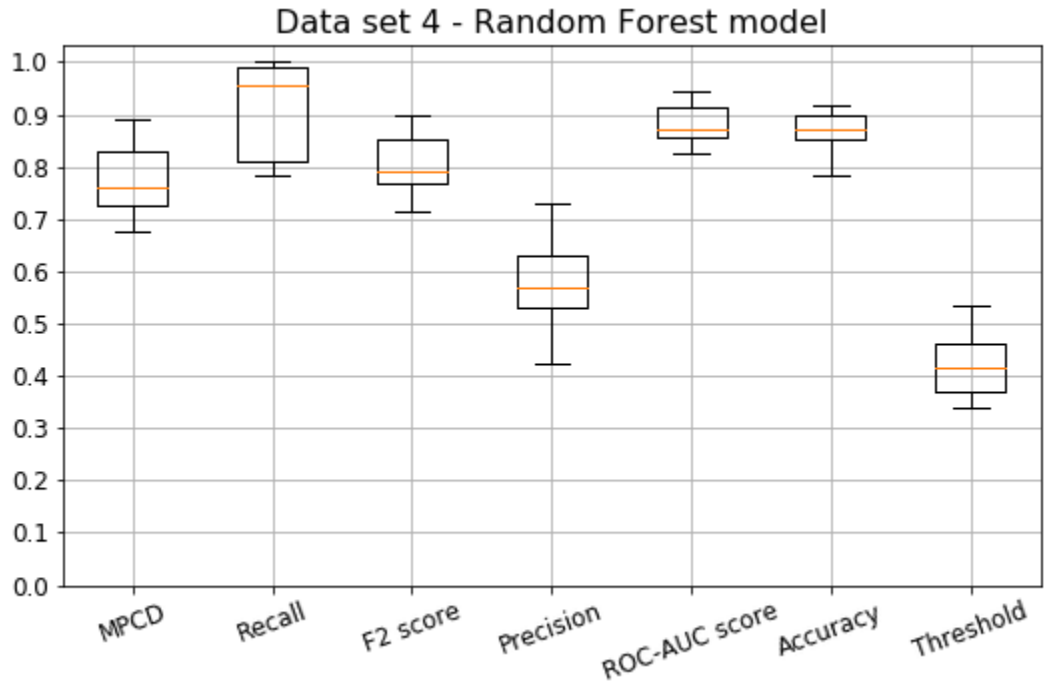

Figure D.28: Boxplot for 10-fold *CV* on data set 4 for the *RF* model

Sub-data set 5

Table D.29: Results for the *RF* model

KPI	*5-fold*	*2-fold*	*3-fold*	*5-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*50-fold*
MPCD	**0.79**	**0.76**	**0.71**	**0.83**	**0.88**	**0.80**	**0.82**	**0.78**	**0.76**	**0.65**
Recall	0.87	0.78	0.83	1.00	0.96	0.91	0.96	1.00	0.86	0.96
F2 score	0.81	0.79	0.75	0.85	0.89	0.83	0.84	0.82	0.79	0.73
Precision	0.65	0.82	0.54	0.53	0.69	0.60	0.56	0.47	0.58	0.37
AUC	0.89	0.87	0.85	0.92	0.94	0.90	0.91	0.89	0.87	0.82
Accuracy	0.90	0.94	0.86	0.86	0.92	0.89	0.87	0.82	0.88	0.73
Threshold	0.48	0.51	0.37	0.40	0.55	0.42	0.41	0.33	0.46	0.28

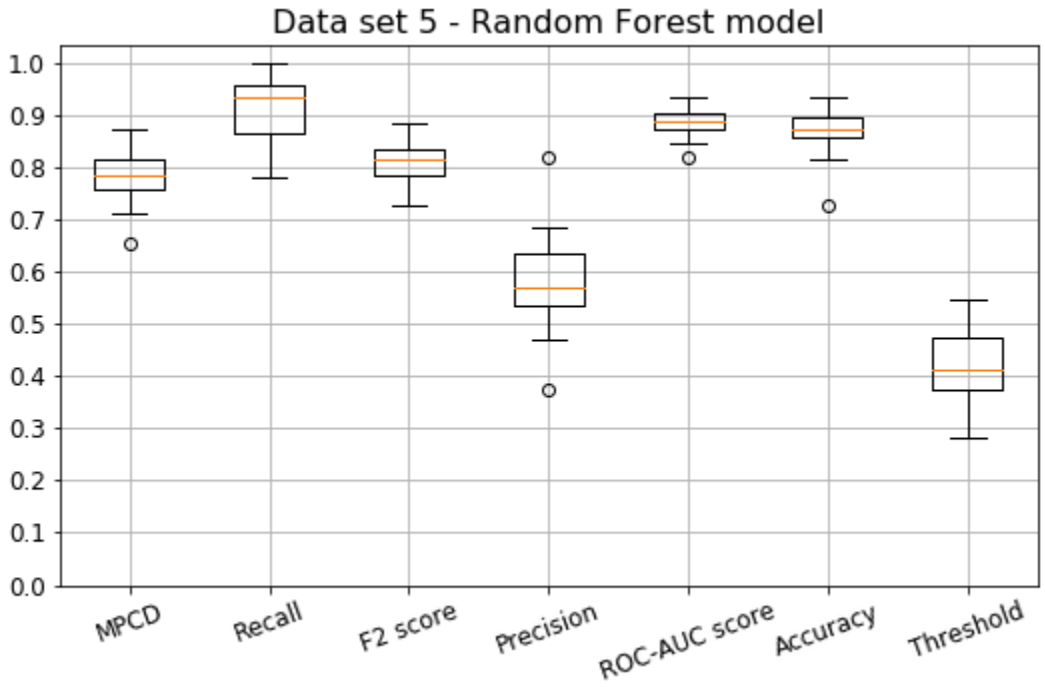

Figure D.29: Boxplot for 10-fold *CV* on data set 5 for the *RF* model

Sub-data set 6

Table D.30: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.88**	**0.72**	**0.82**	**0.81**	**0.76**	**0.84**	**0.84**	**0.71**	**0.79**	**0.80**
Recall	0.90	0.90	0.95	0.90	0.90	0.95	1.00	0.84	0.95	1.00
F2 score	0.89	0.75	0.83	0.83	0.78	0.85	0.85	0.73	0.81	0.82
Precision	0.86	0.45	0.56	0.62	0.51	0.59	0.53	0.48	0.50	0.48
AUC	0.94	0.85	0.91	0.90	0.87	0.92	0.92	0.85	0.89	0.90
Accuracy	0.96	0.82	0.88	0.90	0.86	0.89	0.86	0.85	0.85	0.83
Threshold	0.59	0.33	0.37	0.42	0.45	0.41	0.39	0.40	0.39	0.35

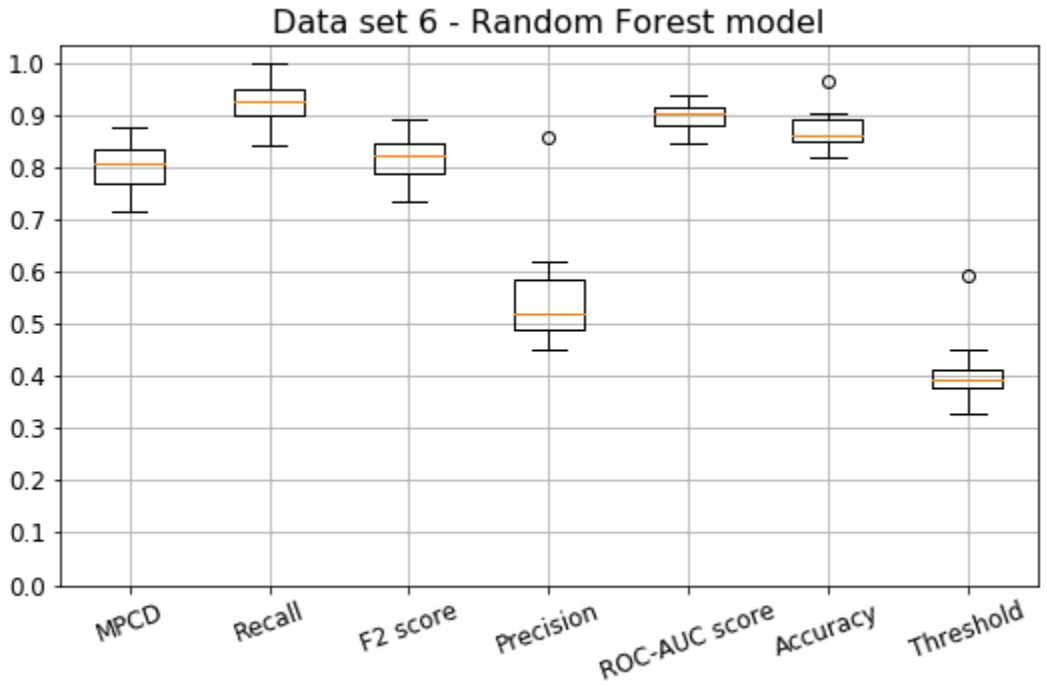

Figure D.30: Boxplot for 10-fold *CV* on data set 6 for the *RF* model

Sub-data set 7

Table D.31: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.78**	**0.86**	**0.86**	**0.87**	**0.78**	**0.84**	**0.72**	**0.69**	**0.78**	**0.77**
Recall	0.83	0.89	1.00	1.00	0.83	0.94	0.94	0.83	0.89	0.94
F2 score	0.81	0.88	0.87	0.87	0.80	0.85	0.75	0.71	0.79	0.79
Precision	0.71	0.84	0.56	0.58	0.68	0.61	0.41	0.45	0.55	0.47
AUC	0.89	0.93	0.93	0.94	0.88	0.92	0.86	0.83	0.88	0.88
Accuracy	0.93	0.96	0.88	0.89	0.92	0.90	0.79	0.82	0.88	0.83
Threshold	0.46	0.53	0.41	0.36	0.41	0.45	0.36	0.41	0.44	0.41

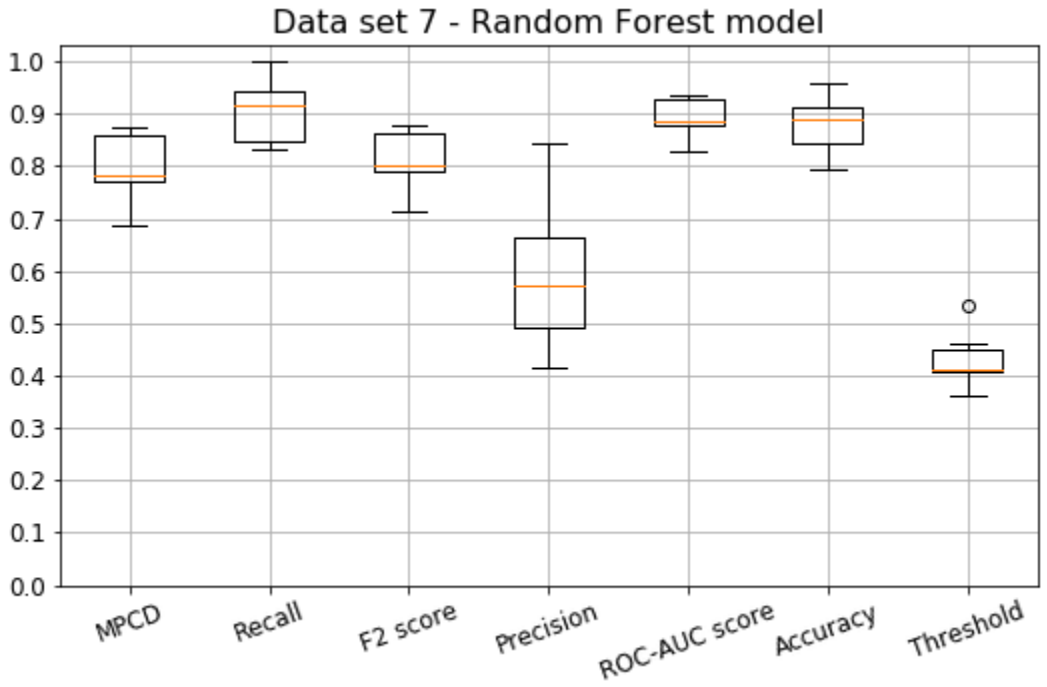

Figure D.31: Boxplot for 10-fold *CV* on data set 7 for the *RF* model

Sub-data set 8

Table D.32: Results for the *RF* model

KPI	*1-fold*	*2-fold*	*3-fold*	*4-fold*	*5-fold*	*6-fold*	*7-fold*	*8-fold*	*9-fold*	*10-fold*
MPCD	**0.65**	**0.91**	**0.86**	**0.65**	**0.81**	**0.83**	**0.87**	**0.83**	**0.67**	**0.85**
Recall	0.67	1.00	0.89	0.89	0.89	0.90	1.00	1.00	100	0.89
F2 score	0.70	0.90	0.87	0.69	0.82	0.85	0.87	0.83	0.73	0.87
Precision	0.86	0.64	0.80	0.36	0.62	0.69	0.56	0.50	0.35	0.80
AUC	0.82	0.95	0.93	0.81	0.90	0.91	0.93	0.91	0.84	0.93
Accuracy	0.94	0.92	0.95	0.76	0.90	0.92	0.89	0.85	0.72	0.95
Threshold	0.63	0.45	0.57	0.30	0.42	0.56	0.42	0.27	0.27	0.56

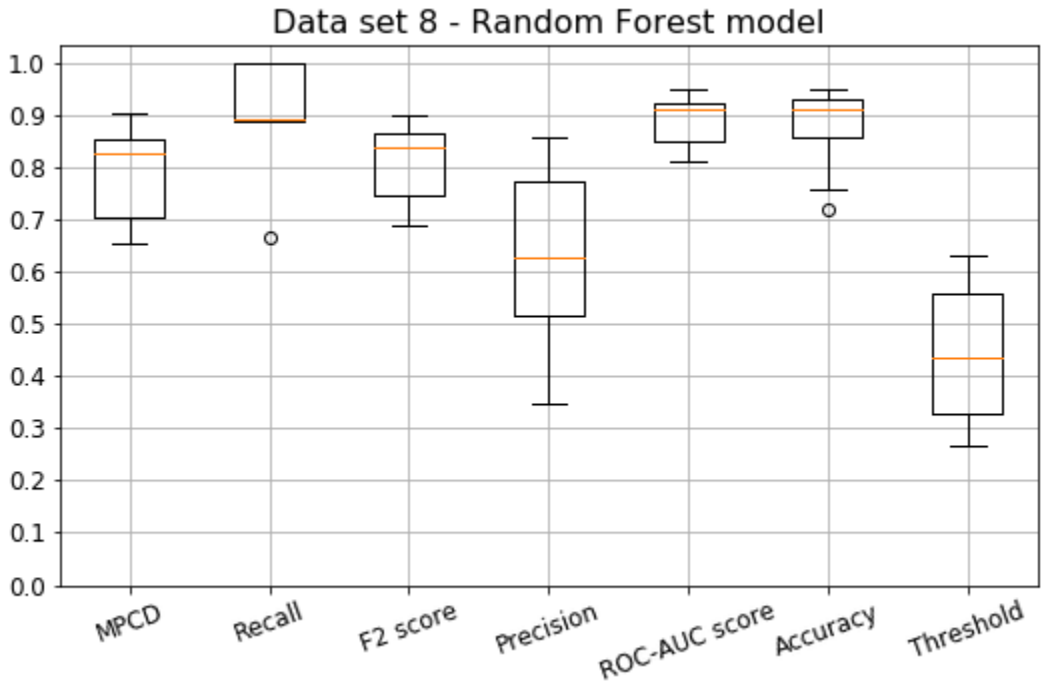

Figure D.32: Boxplot for 10-fold *CV* on data set 8 for the *RF* model

Appendix E

Python code

E.1 Create filtered data base

```
#Imports
import pandas as pd
import numpy as np

def create_database():
    #Raw data base loading
    df_1 = pd.read_csv("01.csv", sep=";", encoding="ANSI")
    df_2 = pd.read_csv("02.csv", sep=";", encoding="ANSI")
    df_3 = pd.read_csv("03.csv", sep=";", encoding="ANSI",
                       parse_dates=[[1,2]])
    df_4 = pd.read_csv("04.csv", sep=";", encoding="ANSI",
                       error_bad_lines=False, warn_bad_lines=False,
                       parse_dates=[[2,3]])
    df_5 = pd.read_csv("05.csv", sep=";", encoding="ANSI")
    df_6 = pd.read_csv("06.csv", sep=";", encoding="ANSI")

    #Data filtering
    df_filt = df_1.copy()
    df_filt = df_filt[df_filt["DIAG ING/INPAT"]=="COVID19 - POSITIVO"]
    df_filt = df_filt[(df_filt["MOTIVO_ALTA/DESTINY_DISCHARGE_ING"]=="
   Domicilio") | (df_filt["MOTIVO_ALTA/DESTINY_DISCHARGE_ING"]=="
   Fallecimiento")]
    df_filt = df_filt[df_filt["EDAD/AGE"]!=0]
    df_filt = df_filt[(df_filt["SAT_02_PRIMERA/FIRST_URG/EMERG"]!=0) | (
   df_filt["SAT_02_ULTIMA/LAST_URG/EMERG"]!=0)]

    df = df_filt.copy()
```

```
    df = df[["PATIENT ID", "EDAD/AGE", "SEXO/SEX", "MOTIVO_ALTA/DESTINY_DISCHARGE_ING"]]
    df["O2"] = df_filt[["SAT_02_PRIMERA/FIRST_URG/EMERG", "SAT_02_ULTIMA/LAST_URG/EMERG"]].max(axis=1)
    df.columns = ["ID", "Age", "Gender", "Deceased", "O2"]

    #Add comorbidities
    comorbilities = {}
    for pat in df_1["PATIENT ID"].sort_values():
        comorbilities[pat] = []

    for index, row in df_5.iterrows():
        codes = row[[col for col in df_5.columns if "DIA" in col]].tolist()
        comorbilities[row["PATIENT ID"]] = [c.split(".")[0] for c in codes if isinstance(c, str)]

    for index, row in df_6.iterrows():
        newcodes = row[[col for col in df_6.columns if "DIA" in col]].tolist()
        pat_id = row["PATIENT ID"]
        codes = comorbilities[pat_id]
        to_add = [code.split(".")[0] for code in newcodes if code not in comorbilities[pat_id] and isinstance(code, str)]
        comorbilities[pat_id] += to_add

    codes = ["N17", "I10", "E11", "I25", "J80"]

    for code in codes:
        df[code] = [1 if code in comorbilities[pat] else 0 for pat in df.ID]

    #Add biomarkers
    biomarkers = ['AP -- ACTIVIDAD DE PROTROMBINA',
     'CREA -- CREATININA',
     'DD -- DIMERO D',
     'FER -- FERRITINA',
     'IGG -- IgG  (INMUNOGLOBULINA G)',
     'IGM -- IgM  (INMUNOGLOBULINA M)',
     'IL6 -- INTERLEUCINA 6',
     'LAC -- LACTATO',
     'LDH -- LDH',
     'LEUC -- Leucocitos',
     'LEULCR -- Recuento Leucocitos',
     'LIN -- Linfocitos',
     'LIN% -- Linfocitos %',
     'NEU -- Neutrófilos',
     'NEU% -- Neutrófilos %',
```

```
    'PCR -- PROTEINA C REACTIVA',
    'PLAQ -- Recuento de plaquetas',
    'TP -- TIEMPO DE PROTROMBINA',
    'TROPO -- TROPONINA']

df = df.reset_index(drop=True)

for bio in biomarkers:
    df[bio + "_max"] = np.nan
    df[bio + "_min"] = np.nan

for pat in df.ID:
    idx = df[df.ID==pat].index
    temp = df_4[df_4["PATIENT ID"]==pat]
    for col in biomarkers:
            if temp[temp["DETERMINACION/ITEM_LAB"]==col].empty: continue
            vals = []
            try:
                for i in temp[temp["DETERMINACION/ITEM_LAB"]==col].iloc
[:,4]:
                    if i=="Sin resultado" or i=="----" or i=="No se
observan":
                        continue
                    i2 = i
                    if "<" in i or ">" in i:
                        i2 = i[2:]
                    vals.append(i2)
                if not vals:
                    print(pat, col)
                    vals = np.nan
            except:
                print(pat, col)
                print(temp[temp["DETERMINACION/ITEM_LAB"]==col].iloc[:,4])
                vals = np.nan
            finally:
                ar = np.array(vals, dtype="float")
                df.loc[idx, col + "_max"] = ar.max()
                df.loc[idx, col + "_min"] = ar.min()

#One-hot encoding for categorical features
df.Gender = (df.Gender=="MALE").astype(int)
df.Deceased = (df.Deceased=="Fallecimiento").astype(int)

#Save final csv data base
df.to_csv("Final_CSV.csv")
```

```
109
110     return df
```

E.2 Normalize data base

```
#Imports
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

def normalizer(df):
    cols_to_scale = ["Age", "O2"] + list(df.columns[10:])
    scaler = MinMaxScaler()
    df.loc[:, cols_to_scale] = scaler.fit_transform(df.loc[:,cols_to_scale])

    return df
```

E.3 Create data sets

```
#Imports
import pandas as pd
import numpy as np

def Create_DataSets(df):
    sorted_cols = (len(df) - df.isna().sum(axis=0)).sort_values(ascending=
    False)

    cuts = {}
    for n, i in enumerate(np.unique(sorted_cols.values)[::-1]):
        cuts[n+1] = len(sorted_cols[sorted_cols>=i])

    data_sets = {}
    for n, cut in enumerate(cuts.values()):
        if n==8: break
        data_sets[n+1] = df.loc[:, sorted_cols[:cut].index]

    return data_sets
```

E.4 OCTM

```
def OCTM_Thresholds(outputs):
    sorted_outs = sorted(outputs, reverse=True)

    return [(sorted_outs[i]+sorted_outs[i+1])/2 for i in range(len(sorted_outs
    )-1)]
```

E.5 Evaluation

```
#Imports
import pandas as pd
import numpy as np

from sklearn.metrics import recall_score, precision_score, accuracy_score,
    fbeta_score, roc_auc_score,plot_confusion_matrix, confusion_matrix,
    precision_recall_curve
from sklearn.ensemble import RandomForestRegressor

import tensorflow as tf
from tensorflow.keras.models import Model, Sequential, load_model
from tensorflow.keras.layers import Input, Dense, Dropout

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, KFold, StratifiedKFold


def evaluation(df, OCTM=True, smote=False):
    #Drop NaN values from data set
    df = df.dropna().reset_index(drop=True)

    #Create K-Folds while maintaining the class label preportion
    kf = StratifiedKFold(n_splits=10, shuffle=True)
    sm = SMOTE(sampling_strategy=0.80)

    cv_DL = []
    cv_RF = []
    tf.keras.backend.clear_session()

    #Initial Thresholds definition
    Thresholds = np.arange(0, 1, 0.01)

    for train_idx, test_idx in kf.split(df.drop(["Deceased", "ID"], axis=1),
    df.Deceased):
        df_train = df.loc[train_idx]
        df_test = df.loc[test_idx]

        X_train, y_train = df_train.drop(["Deceased", "ID"], axis=1).values,
    df_train.Deceased.values
        X_test, y_test = df_test.drop(["Deceased", "ID"], axis=1).values,
    df_test.Deceased.values

        #SMOTE
```

```
        if smote: X_train, y_train = sm.fit_sample(X_train, y_train)

        #Deep Learning model
        model_DL = Sequential([
            Input(X_train.shape[1:]),
            Dense(17, activation="sigmoid"),
            Dense(10, activation="sigmoid"),
            Dense(5, activation="sigmoid"),
            Dense(1, activation="sigmoid")])

        model_DL.compile(loss="binary_crossentropy",
                         optimizer="adam")

        history = model_DL.fit(X_train, y_train,
                               epochs=200,
                               verbose=0)

        best_mpcd = 0
        outputs = model_DL.predict(X_test).squeeze()

        #If OCTM is to be used
        if OCTM: Thresholds = OCTM_Thresholds(outputs)

        for t in Thresholds:
            y_pred = (outputs>=t).astype(int)
            cm_array = confusion_matrix(y_test, y_pred)
            fp = cm_array[0,1]/cm_array[0].sum()
            fn = cm_array[1,0]/cm_array[1].sum()
            mpcd = (1-fp)*(1-fn)

            if mpcd > best_mpcd:
                best_mpcd = mpcd
                best_t = t

        y_pred = (outputs>=best_t).astype(int)

        cm_array = confusion_matrix(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        f2 = fbeta_score(y_test, y_pred, beta=2)
        precision = precision_score(y_test, y_pred)
        auc = roc_auc_score(y_test, y_pred)
        acc = accuracy_score(y_test, y_pred)

        fp = cm_array[0,1]/cm_array[0].sum()
        fn = cm_array[1,0]/cm_array[1].sum()
```

```
84          mpcd = (1-fp)*(1-fn)
85
86          cv_DL.append([mpcd, recall, f2, precision, auc, acc, best_t])
87          print("Fold {} for DL model done".format(len(cv_DL)))
88
89
90          #Random Forest model
91          model_RF = RandomForestRegressor(n_estimators=500, max_depth=2,
    max_features=5)
92          model_RF.fit(X_train, y_train)
93
94          best_mpcd = 0
95          outputs = model_RF.predict(X_test)
96
97          #If OCTM is to be used
98          if OCTM: Thresholds = OCTM_Thresholds(outputs)
99
100         for t in Thresholds:
101             y_pred = (outputs>=t).astype(int)
102             cm_array = confusion_matrix(y_test, y_pred)
103             fp = cm_array[0,1]/cm_array[0].sum()
104             fn = cm_array[1,0]/cm_array[1].sum()
105             mpcd = (1-fp)*(1-fn)
106
107             if mpcd > best_mpcd:
108                 best_mpcd = mpcd
109                 best_t = t
110
111         y_pred = (outputs>=best_t).astype(int)
112
113         cm_array = confusion_matrix(y_test, y_pred)
114         recall = recall_score(y_test, y_pred)
115         f2 = fbeta_score(y_test, y_pred, beta=2)
116         precision = precision_score(y_test, y_pred)
117         auc = roc_auc_score(y_test, y_pred)
118         acc = accuracy_score(y_test, y_pred)
119
120         fp = cm_array[0,1]/cm_array[0].sum()
121         fn = cm_array[1,0]/cm_array[1].sum()
122         mpcd = (1-fp)*(1-fn)
123
124         cv_RF.append([mpcd, recall, f2, precision, auc, acc, best_t])
125         print("Fold {} for RF model done".format(len(cv_RF)))
126
127     results_DL = pd.DataFrame(cv_DL, columns=["MPCD", "Recall", "F2 score", "
```

```
    Precision",
128                                             "ROC-AUC score", "Accuracy", "
    Threshold"])
129     results_RF = pd.DataFrame(cv_RF, columns=["MPCD", "Recall", "F2 score", "
    Precision",
130                                             "ROC-AUC score", "Accuracy", "
    Threshold"])
131
132     return results_DL, results_RF
```

E.6 *KNN* Imputation

```
#Imports
import pandas as pd
import numy as np

#Define columns to impute from data set
cols_2_impute = ["DD -- DIMERO D_min", "DD -- DIMERO D_max"]

k = int(len(df_train)*0.10)

for i, row in df_test.drop(cols_2_impute, axis=1).iterrows():
    distances = np.sqrt(((df_train.drop(["ID", "Deceased"], axis=1)-row.drop(["ID", "Deceased"]))**2).sum(axis=1))
    k_indeces = distances.sort_values()[:k].index
    df_test.loc[i, cols_2_impute] = df_train.loc[k_indeces, cols_2_impute].mean()
```

Curriculum Vitae

José Luis Guadiana Álvarez was born in Monterrey, Nuevo León, Mexico on May 12^{th}, 1994. He received the Bachelor degree of Mechatronics Engineer from the Universidad Autónoma de Nuevo León, Mexico in December 2016. He worked in *Metalsa* as a Maintenance Specialist from 2016 to 2019. Afterwards, he joined the *Automotive Consortium* research group in January 2019 while studying the *Master of Science in Manufacturing Systems* program by the *Tecnológico de Monterrey, Escuela de Ingeniería y Ciencias, Campus Monterrey*.

This document was typed in using LaTeX by José Luis Guadiana Álvarez