University ”Politehnica” of Bucharest
Faculty of Electronics, Telecommunications and Information Technology

Estimation of patient survival chances based on the medical data acquired in the first 24 hours in the Intensive Care Unit

Diploma Thesis

submitted in partial fulfillment of the requirements for the
Degree of *Engineer* in the domain *Electronics and Telecommunications*
study program *Applied Electronics (ETC-ELAeng)*

Thesis Advisor
Conf.dr.ing. Dragoș Daniel ȚARĂLUNGĂ

Student
Elena-Briana Boeru

July 2023

University "Politehnica" of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Study program **ELAen**

Anexa 1

DIPLOMA THESIS

of student **BOERU M. Elena-Briana, 441F**

1. Thesis title: Estimarea șanselor de supraviețuire pe baza datelor medicale înregistrate în primele 24 h de la intrarea în terapie intensivă

2. The student's original contribution will consist of (not including the documentation part) and design specifications:
Proiectul are ca obiectiv implementarea unui algoritm de prelucrare digitala a semnalelor biomedicale pentru a estima sansele de supravieutire pentru pacientii admisi la sectia de terapie intensiva. Trebuie analizate o varietate mare de tipuri de semnale si parametri biomedicali, proveniți de la diversi senzori. Se va implementa un model de calsificare bazat pe inteligența artificială pentru estimarea sanselor de supravietuire.

3. Academic courses the thesis is based on:
Electronică și Informatică Medicală, Prelucrarea digitală a semnalelor, Structuri de date și algoritmi

4. Thesis registration date: 2022-12-22 14:06:37

Thesis advisor(s),
Conf.dr.ing. Dragoș Daniel ȚARĂLUNGĂ

Student,
BOERU M. Elena-Briana

Departament director,
Ș.L. dr. ing Bogdan FLOREA

Dean,
Prof. dr. ing. Mihnea UDREA

Validation code: **5c766482c0**

University “Politehnica” of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Study program **ELAen**

Anexa 1 (translated)

DIPLOMA THESIS

of student **BOERU M. Elena-Briana, 441F**

1. Thesis title: Estimation of patient survival chances based on the medical data acquired in the first 24 hours in the Intensive Care Unit

2. The student’s original contribution will consist of (not including the documentation part) and design specifications:
The project aims to implement a digital processing algorithm of biomedical signals to estimate the chances of survival for patients admitted to the intensive care unit. A large variety of types of signals and biomedical parameters, coming from various sensors, must be analyzed. A calcification model based on artificial intelligence will be implemented to estimate the chances of survival.

3. Academic courses the thesis is based on:
Medical Electronics and Informatics, Digital Signal Processing, Data Structures and Algorithms

4. Thesis registration date: 2022-12-22 14:06:37

Thesis advisor(s),
Conf.dr.ing. Dragoș Daniel ȚARĂLUNGĂ

Student,
BOERU M. Elena-Briana

Departament director,
Ș.L. dr. ing Bogdan FLOREA

Dean,
Prof. dr. ing. Mihnea UDREA

Validation code: **5c766482c0**

Annex 5

Statement of Academic Honesty

I hereby declare that the thesis ***Estimation of patient survival chances based on the medical data acquired in the first 24 hours in the Intensive Care Unit***, submitted to the **Faculty of Electronics, Telecommunications and Information Technology** in partial fulfillment of the requirements for the degree of Engineer in the domain **Electronic Engineering and Telecommunications**, study program **Applied Electronics (ETC-ELAeng)** is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited ”as is” or translated from other languages are written between quotes and are referenced to the sources. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations or measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations or measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, July 2023

Graduate Elena-Briana BOERU

……

Contents

List of figures

List of tables

List of Abbreviations

COVID = Coronavirus Disease
ICU = intensive care unit
AI = artificial intelligence
APACHE = Acute Physiology and Chronic Health Evaluation
SAPS = Simplified Acute Physiology Score
SOFA = Sequential Organ Failure Assessment
US = United States
ROC = receiver operating characteristic
AIDS = acquired immunodeficiency syndrome
ANZROD = Australian and New Zealand Risk of Death
ANZICS = Australian and New Zealand Intensive Care Society
CORE = Centre for Outcome and Resource Evaluation
APD = Adult Patient Database
ANZ = Australia and New Zealand
APD = Adult Patient Database
APS = acute physiology score
AUROC = area under the receiver operating characteristic curve
SMR = standardized mortality ration
ROD = risk of death
ALT = alanine transaminase
ALB = albumin
AST = aspartate transaminase
BIL = bilirubin
CA = calcium
CL = chloride
CREA = creatinine
CRP = C-reactive protein
DD = D-dimer
FERRI = ferritin
GLU = glucose
IL6 = interleukin 6
LDH = lactate dehydrogenase
K = potassium ion
PROCAL = procalcitonin
PT = prothrombin time
NA = sodium ion
TROP-T = troponin T
CKD-EPI = Chronic Kidney Disease Epidemiology Collaboration glomerular filtration rate
paCO2 = arterial blood partial pressure of carbon dioxide
paO2 = arterial blood partial pressure of oxygen
apH = pH in arterial blood
aSatO2 = arterial blood oxygen saturation
paO2 = partial pressure of oxygen in arterial blood
FiO2 = fraction of inspired oxygen quotient value
#BAS = blood count of basophils
#EOS = eosinophils
#LEU = leucocytes
#LYM = lymphocytes
#MON = monocytes
#NEU = neutrophils
PLT = platelets
ERY = blood count of erythrocytes

HGB = blood concentration of hemoglobin
HCT = hematocrit
MCV = mean cell volume
MCH = mean cell hemoglobin
MCHC = mean cell hemoglobin concentration
MPV = mean platelet volume
RDW-CV = red blood cells distribution width based on the coefficient of variation
BLR = binary logistic regression
ANN = artificial neural networks
MLP = multilayer perceptron architecture
IMV = invasive mechanical ventilation
NIMV = non-invasive mechanical ventilation
IFCC = International Federation of Clinical Chemistry
PPV = positive predictive value
NPV = negative predictive value
WiDS = Women in Data Science
MIT = Massachusetts Institute of Technology
GOSSIS = Global Open-Source Severity of Illness Score
BUN = blood urea nitrogen
ATP = adenosine triphosphate
INR = international normalized ratio
VKA = vitamin K antagonists
PT = prothrombin time
WHO = World Health Organization
TC = thrombosis centers
POC = point-of-care
EMG = electromyography
PPG = photopletysmogram
GFR = glomerular filtration rate
S-A = sino-atrial
A-V = atrio-ventricular
LCD = liquid-crystal display
MBP = mean blood pressure
DBP = diastolic blood pressure
SBP = systolic blood pressure
ECG = electrocardiogram
BMI = body mass index
GCS = Glasgow Coma Scale
HIV = human Immunodeficiency Virus
GI = gastrointestinal
ALF = acute liver failure
NHL = Non-Hodgkin lymphoma
IQR = interquartile ranges
KDE = kernel density estimate
SMOTE = Synthetic Minority Over-sampling Technique
SMOTEEN = SMOTE + Edited Nearest Neighbours
NN = neural network
ReLU = Rectified Linear Unit
ELU = Exponential Linear Unit
TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

IDE = integrated development environment
IoT = internet of things
EEG = electroencephalogram
GUI = graphical user interface

Introduction

The race for the development of good estimators of patient survival chances is a novel subject in the engineering community, and has proven itself to be one of the most important prospects for the advancement of biomedical technology. Many attempts have been made over the last few years to make a good prediction algorithm that will help the medical community; however, lack of varied patient data has made this endeavor quite difficult for various groups of developers.

The COVID-19 pandemic has made it obvious that in situations of crisis, when the Intensive Care Unit (to be known further on as "ICU") wards are overfilled and many lives are dependent on good quality medical care, the staff can get overwhelmed and some preventable negative outcomes can be ensured. This is the point in which technology can ease the burden of such situations, just as it has done in various other domains. With a good prediction system, the medical staff can know in advance how much care a patient will require and therefore make the triage stage shorter and keep the ICU wards less crowded. This can ensure that each patient will receive the care that they require without overworking the doctors and nurses, therefore saving many more lives and keeping the general wellbeing of all the parties involved (the medical staff, the patients and their families) at a higher level.

Advancements in technology have made it clear that Artificial Intelligence (further referred to as "AI") [1]–[3]will play a more significant role in every part of our lives, so much so that it will become indispensable in the very near future in every domain of our lives. The medical domain makes no exception and since every ground breaking advancement comes with its own new set of issues to be solved, the introduction of AI in such an intimate area of our lives may prove challenging in adjacent domains outside of the technological experts' control. As the founder of the World Economic Forum, Klaus Schwab, said: *"We must address, individually and collectively, moral and ethical issues raised by cutting-edge research in artificial intelligence and biotechnology, which will enable significant life extension, designer babies, and memory extraction."*.

Our approach of this subject is founded by taking into consideration many statistics[1]–[3] made during the COVID-19 era, which presented us with an unprecedented prospect of how our collective efforts can be used to ensure that a massive number of members of our communities survive, that state that during the three years of pandemic we had 3.3 million infection cases out of which 67 thousand died and many more are now facing lasting health complications which diminish their quality of life. Taking into consideration other ICU statistics from before and after the pandemic we can say with certainty that solutions offered by the engineering community can do a great deal of good to both the patients and the medical staff.

Currently, it has been proven that the introduction of a scoring system based on the medical data acquired in the first 24 hours spent in the ICU ward can be detrimental for a favorable outcome in regards to the patient's survival chances. Scoring systems such as APACHE, SAPS or SOFA that use this data already exist and are used in common medical practice. Recent studies show, however, that there is a growing need for new scoring and prediction systems, that make better use of technological advancements by introducing AI components in their architecture. These scoring systems have been found to be prone to biasing when used in geographical regions different than the ones in which they have been developed, and even if their results can be used as benchmarks for contemporary research, there is no doubt that further development of novel algorithms is necessary.

The purpose of this project is to develop an estimative prediction algorithm for patient survival chances using the data acquired in the first 24 hours spent in the ICU ward, using modern methods that include, but are not restricted to, machine learning and statistical analysis.

1. State of the art

1.1. General presentation of the issue

In our current times, there have been a lot of studies made over the issue of patient diagnosis prediction. Due to the novelty of this domain, there are still a lot of advancements that can be made, especially considering that with modern legislation changes engineers can now have access to much more data than before. Currently, we do have a few prediction models that have proved their usefulness in most cases, the most well known and most used of them being the APACHE III prognostic system, which is used in the first 24 hours of ICU admission. Released in 1991, the APACHE III prognostic system attributes a score between 0 and 299 points to a patient based on a few carefully selected attributes such as: age, sex, reason for admission to ICU, comorbidities and other influencing factors registered during ICU admission. A normal APACHE III score should be between 0 and 71 points, but it is quite common to have less than 55 points as a patient. However, it has been proven in the recent period that such prediction systems are yielding a good prediction estimate over a longer period of time only in the regions where they have been developed. As such, over a long period of time the APACHE III score keeps working well in the North American continent, but in places such as Australia and New Zeeland it has proven to be less efficient.

There are two ways in which such an issue can be solved. Firstly, we could try to develop a world-wide system that takes into consideration patient data from hospitals around the world. It is true that this approach would get rid of the bias we have identified in most prediction models, but this would mean that each region would need to adapt its current methodology of data gathering to a worldwide accepted one, which could mean that great investments will have to be made locally. Not only that, but we also have to analyze if the regional bias is such a bad thing in itself, since we know that cultural, economic, and climatical changes are shaping each population from region to region giving it specificity. The second method, which is also the one that seems to be the most approached currently, consists in developing local prediction models, tailored for the regional population. This is either done by building new models from scratch, adapting world acclaimed prediction models or retraining them with data from the specific region in which we want to use them.

1.2. Detailed presentation of the methods

The APACHE III prognostic system. Risk prediction of hospital mortality for critically ill hospitalized adults

According to *The APACHE III prognostic system. Risk prediction of hospital mortality for critically ill hospitalized adults*[4], this model was developed taking into consideration the data registered from 17.440 unselected adult medical/ surgical ICU admissions at 40 US hospitals. The analysis consisted in studying the likelihood of a variable in being a determining death factor for a patient. The predicting variables were: major medical and surgical disease categories, acute physiologic abnormalities, age, preexisting functional limitations, major comorbidities, and treatment location immediately prior to ICU admission. The prognostic system is divided into two main branches: the APACHE III score, which can be used as a provider for initial risk stratification for severely ill ICU patients, and the APACHE III predictive equation, which uses the APACHE III score and other data such as major disease categories and treatment location immediately prior to ICU admission to provide risk estimates for patient mortality. [4]

Since APACHE III is based on its predecessor, APACHE II, after having the complete data collection gathered from the 40 partnered hospitals the APACHE II equation was first used for validating the death risk between 14 volunteer hospitals and 26 randomly selected hospitals (the 40 hospitals previously discussed) and the yielded probability was the same for both groups ($p=0.20$). Therefore, the APACHE II parameters served as control markers in the development of the APACHE III system. Taking into consideration physiological variables a weighting for each component was deemed necessary. Therefore, in order to compute the weights of each variable, a multivariable logistic regression analysis was made, in order to determine the relationship between death rate and each of the 20 candidate physiologic variables, controlling for 19 other physiologic

variables, age, chronic health conditions, operative status, and major disease categories by using both categoric and continuous weighing approaches. Both individual and combined variable weighting were taken into consideration. Furthermore, the data gathered from each of the hospitals was split roughly in half, 50% going to estimation and the other 50% going to validation. In order to develop the weights for each component, only the estimation part of the data was used. The collected data included both the initial admission parameters (admission value), collected in the first hour of admission or, if reported missing, the values recorded one hour prior to admission, and the worst physiological value over 24 hours and over 23 hours. There were multiple aspects to take into consideration when considering weights. The physiological variables were firstly divided in ranges, then incorporated in the analysis as a series of separate predictor variables for each range, based on cell size and clinical judgement. In the case of discrepancies, the ranges were empirically changed, in order to obtain better results. Then, a smoothing technique was applied (cubic splines analysis) in order to obtain a continuous varying weight for each physiologic variable. Disease specific weighting of physiologic variables was tried, but it provided no improvement over the explanatory power, but its usefulness was not discarded for some cases. For missing variables, a weight equal to zero was assigned after examination of the cases with dummy variables. In regards to weighting the comorbid conditions, the 34 candidate variables were analyzed with a regression analysis, and it was determined that each comorbidity had a variable influence over the dichotomous result (dead or alive). In this case, potential interactions between chronic conditions were explored. In order to produce the final APACHE III equation for mortality prediction the disease and patient location coefficients were combined with the relative weights of the first day values of APACHE III's score, physiology, age and chronic health. Since previous evidence suggested that treatment location immediately prior to ICU admission (emergency room, floor, other ICU, other hospital) held prognostic significance, the explanatory power of a variable describing selection for intensive care was assessed. The time spent in the ER and the fact whether emergency surgery was performed were also investigated factors on total explanatory power. [4]

The obtained results are extensive. As previously stated, the majority (89 percent) of the 40 hospitals studied were nonprofit, and 54 percent were affiliated with a medical school. Of the 42 ICUs studied, 71 percent were mixed medical-surgical, 16 percent were surgical, 10 percent were medical. A total of 9,195 patients had positive answers to chronic health and functional status questions. Physiologic variables used in APACHE III were weighted differently from those used in the previous version of the study (APACHE II). For some variables, such as blood pressure, the risk associated with extremely high recordings is different from that associated with equally extreme but low recordings. Overall explanatory power for patient outcome increased when they accounted for the following interactions: serum pH with PCO2, creatinine with urine output, and respiratory rate with ventilator use. Serum potassium and serum bicarbonate did not meet the minimal statistical inclusion criteria to be included in the analysis of human death rates, and the relationship between death rate and each of the 20 candidate physiologic variables was inconclusive. The Glasgow Coma factors have been reorganized to get rid of scores that were identical for various clinical presentations and to make it easier to assess eye opening. By reducing the criterion for eye opening, they did away with distinctions between unclear words and improper noises, flexion withdrawal and decorticate stiffness, decerebrate rigidity and no reactions, and more. As the most suitable scoring method for the physiologic component of APACHE III, they kept the worst value for the first 24 hours of ICU treatment. Total physiologic weight has a range of 0 to 252; individual variables have a range of 0 to 48. A missing physiologic value receives a weight of zero. The validation sample was accurately predicted by the APACHE III physiologic weights (receiver operating characteristic [ROC] = 0.88 estimate; 0.87 validation). [4]

In the figure 1.1, they compare the final weights for these 17 physiologic variables in the estimate and validation portions of the data set in terms of their explanatory power and discrimination. Seven variables (acquired immunodeficiency syndrome [AIDS], hepatic failure, lymphoma, solid tumor with metastases, leukemia/multiple myeloma, immunocompromise, and cirrhosis) were estimated from the 34 candidate chronic health items and satisfied the predetermined statistical requirements for inclusion. These factors were not necessary for this study's inclusion since elective postoperative ICU hospitalizations did not typically involve these variables. Using 78 diagnostic categories modified from the original 212 based on frequency and death rate, they re-estimated the equation on the whole data source. They also computed the final weights for the age (range, 0 to 24) and chronic health assessment (range, 0 to 23) variables using this equation. [4]

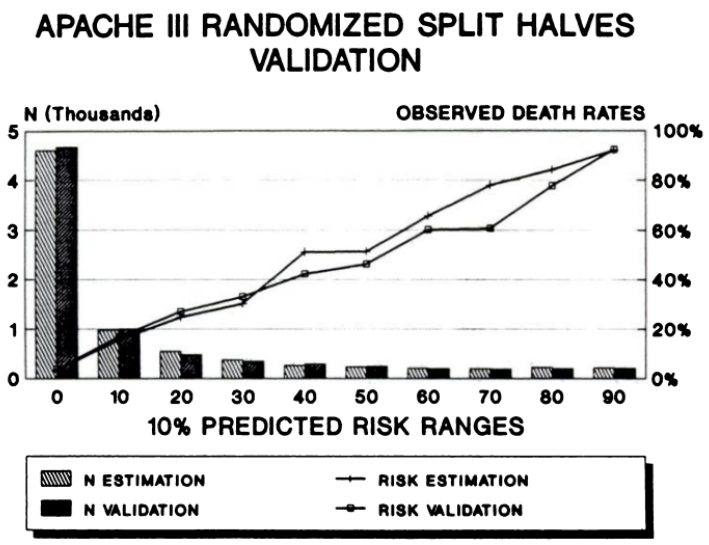


Figure 1.1. APACHE III physiology randomized split halves validation [4]

The total of the three categories of factors (physiology, age, and chronic health) yields a score that ranges from 0 to 299 in cardinal numbers. The 78 major illness categories have individual logistic regression analyses for each of them. The equations are not reliant on other variables, such as population size or mortality rates, and are statistically independent of one another. Figure 1.2, which plots anticipated risk categories for four main diseases versus APACHE III score, further exemplifies the connection between illness categorization and risk prediction. [4]

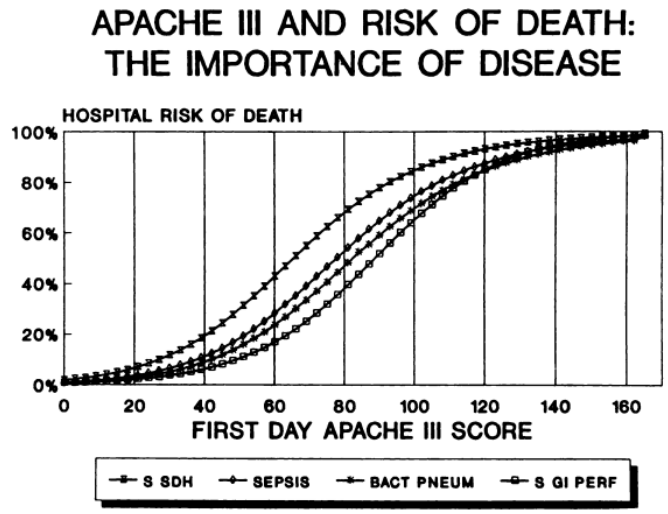


Figure 1.2. Relationship between APACHE III score and predicted risk of death for patients with different afflictions [4]

The relative significance of the illness is negligible whether the APAC HE III score is either low (<=20) or high (>=140). However, changes in illness classifications are significantly correlated with variations in risk projections within the middle range of scores. At a 0.50 expected risk, the overall right categorization on day one was 88.2%. The majority of the significant difference in the observed death rates from these units is explained by the initial or first-day APACHE III equation when applied across the various ICUs (r squared = 0.90 p 0.0001). For two septic shock patients selected from one of the 40 hospitals, the daily probability of hospital death increased with time. The most recent and first day's APACHE III scores, coupled with the main illness category and patient location factors, are used to calculate each daily risk estimate. The researchers discovered that using the initial and most recent ratings had the most explanatory power. [4]

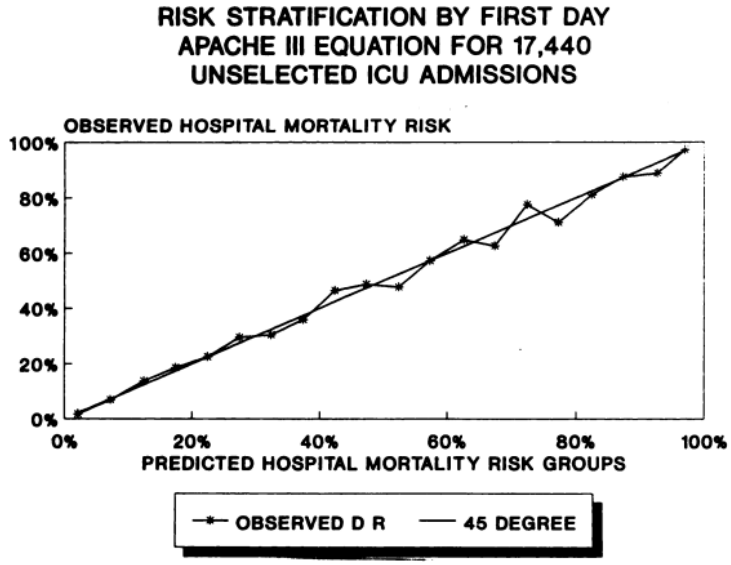


Figure 1.3. Risk stratification by first-day APACHE III equation; Assessment of predictive accuracy [4]

This prediction algorithm is still used today, albeit with a few improvements, and its predictive accuracy can be observed in figure 1.3. Its estimative power was a novelty at the time of its apparition on the market, given that in the first 24 hours 95 percent of ICU admissions could be given a risk estimate within 3 percent of actual observed statistics. The fact that this risk estimate could also be updated each day spent in the ICU made it so that the first day APACHE III equation was proved to account the vast majority of possible changes in patient status. Many other estimative procedures were based on the work made in the APACHE project, but the advancement of the general technology made it so that better algorithms could be developed by specialist with the help of AI, a fact that will be discussed further on in this work. [4]

Risk prediction of hospital mortality for adult patients admitted to Australian and New Zealand intensive care units: Development and validation of the Australian and New Zealand Risk of Death model

This study[5], [6] detailly presents the steps taken for the development and validation of the ANZROD prediction model for death estimates of patients admitted to the Australian and New Zealand ICUs. The issue which determined the need for replacing the previously used APACHE III model in favor of building a new algorithm were the findings of the researchers from the Australian and New Zealand Intensive Care Society (ANZICS) Centre for Outcome and Resource Evaluation (CORE) who adopted the APACHE methodology for benchmarking outcomes in local ICUs, specifically, the APACHE III-j (tenth iteration) model. It seems that even though benchmarks are being provided, the effectiveness of the APACHE model is in continuous deterioration, overpredicting mortality in Australasia, and thus is no longer providing a reflection of the real risk of death for the Australian ICU population. Considering these factors, work was begun to develop the *Australian and New Zealand Risk of Death* [ANZROD] model, whose performance was compared to the previously used *Acute Physiology and Chronic Health Evaluation [APACHE] III-j*. [5], [6]

The ANZICS Adult Patient Database (APD), one of the world's biggest ICU data sets with more than 1.4 million ICU admissions, served as the source for the study's data. This top-notch bilingual database keeps track of demographic, illness-severity, and treatment-related information from adult ICU admissions starting in 1990. Individual ICUs in all ANZ jurisdictions gather data, which is then forwarded to ANZICS CORE for processing and benchmarking. This research comprised all adult admissions reported to the ANZICS APD from January 1, 2004, and December 31, 2009. Patients under the age of 16, those in for palliative care, those without an acute physiology score on the first day in the intensive care unit, and those without missing hospital outcomes were all disqualified. The study was carried out with the agreement of the Monash University Human Research Ethics Committee, and all data were deidentified. Age, ongoing medical problems, and physiological information were all needed to determine the acute physiology score (APS) for APACHE III. The APS is based on the 16 physiologic components' worst measurement on ICU Day 1. Additional characteristics that they

retrieved were lead time, elective admission, hospital source of admission, treatment restriction, and others. The ANZICS APD's hospital categorization was also considered. Figure 1.4 illustrates the data selection process. [5], [6]

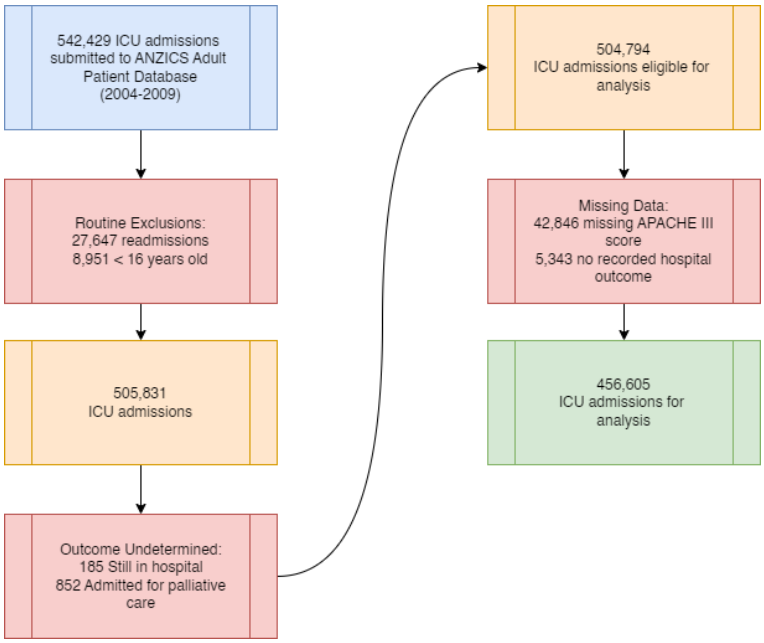


Figure 1.4. Construction of ANZROD database

The selected data was randomly split into two mutually excluding categories: derivation data (67%), which was used for the development of the prediction model, and the validation data (33%) which was used for testing the model. To ascertain the correlation between hospital mortality and each of the predictor factors specified in the APACHE III model, multivariate logistic regression analysis was performed. By introducing or removing particular variables, a number of models were built, and their performances were compared. Through a review of the variable selection and weighting, the aim was to enhance the risk prediction produced by the globally accepted APCHE III model. The final model was selected as the one with the least difference between observed and predicted outcomes (using Hosmer-Lemeshow C and H statistics) and that showed the best discrimination due to area under the receiver operating characteristic curve [AUROC] measurement. The development of the predictive model can be described in four steps. [5], [6]

The first step consisted in the analysis of the existing of the APACHE III methodology and the reweighting of the ANZ diagnostic categories. The model was created using the APACHE III approach that is now in use. Using the weightings suggested by Knaus et al., the age score, chronic health status score, and APS score were objectively determined. They reweighted all 124 diagnoses (94 original APACHE II diagnoses and 30 new locally generated diagnoses) using logistic regression analysis, and they computed model coefficients. The effectiveness of the model was evaluated after reweighting 124 diagnostic categories. The second step approached the reweighting of the components of APACHE III score). Age, chronic health condition, and APS are all weighted into the APACHE III score. Each of these 16 APS components, as well as the age score and chronic health score, were reweighted. These elements were combined via logistic regression, and model coefficients were generated for each predictor. At stage II of model development, the total score for each component was incorporated into the model. After reweighting this score's constituent parts, model performance was evaluated. Step three is characterized by the addition of new variables to the model. The chronic health score was de-bundled at stage III and each of the 7 chronic diseases were reweighted. In the logistic regression model, fresh factors were investigated such as treatment restriction, hospital admission source, lead time, APACHE II chronic diseases, gender, smoking status, and ventilation. Only variables with a $P < .001$ were included in the final model due to the size of the data collection. With the inclusion of additional predictor variables, the calibration and discrimination of models were evaluated. Lastly, step four tackled individual models for each major diagnostic group. To enhance overall mortality prediction, several logistic

regression models were fitted for each main diagnostic category. Major diagnostic categories incorporated operational and nonoperative diagnoses in each system, with the exception of cardiovascular, and were drawn from the major structural categories used in the APACHE III score system (cardiovascular, respiratory, etc.). Each of the following models included all relevant variables discovered at stage III. Both a combined model and each separate model's calibration and discrimination were evaluated. The final predictor variables were chosen as the following: APACHE III score components, source of admission, lead time, elective surgery, treatment limitations, ventilation status and APACHE III diagnostics. For the assessment of the validity of the predictive model, a number of methods were used: the standard mortality ratio, Hosmer-Lemeshow C and H statistics, Brier score, cox calibration regression, AUROC, and calibration curves. [5], [6]

The results in regards to the performance of the models are most satisfactory. At the bottom end of the calibration curve, where the majority of cases are located, the calibration is almost perfectly calibrated. Figure 1.5 showcases the comparison of the ANZROD versus the APACHE III-j calibration curves. The model (ANZROD) with the lowest Hosmer-Lemeshow C and H statistics and the highest AUROC is built on independent equations for the major diagnostic groupings. The aggregate predicted hospital mortality for the validation data set is 11.3%, which is quite similar to the actual mortality (11.4%) and the ANZROD model shows excellent discrimination (AUROC, 0.902). Except for the risk deciles of 70% to 80% (1.1%), 80% to 90% (2.2%), and 90% to 100% (1.6%), the gap between observed and mean projected hospital mortality varies from 0.02% to 0.5% across all risk deciles. The Brier score is modest, the Cox calibration regression results are negligible, and the SMR is 1.6%. [5], [6]

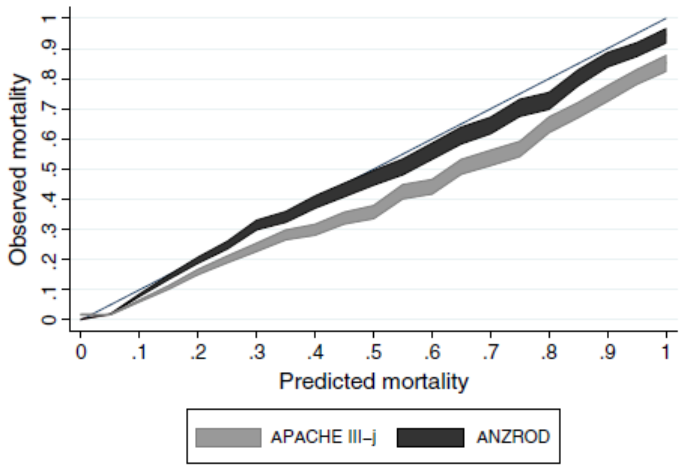


Figure 1.5. Calibration curve of ANZROD vs. APACHE III-j [5], [6]

In tables 1.1 and 1.2 we can view the assessment results for both the derivation and the validation data sets. The models have used the following predictor variables [5], [6]:

- APACHE III-j: APS + age score + chronic health score + ICU admission source + elective surgery + 94 diagnoses.
- Model 1: APACHE III score + ICU admission source + elective surgery + 124 diagnoses.
- Model 2: Components of APACHE III APS + APACHE III age score + APACHE III chronic health score + ICU admission source + elective surgery + 124 diagnoses.
- Model 3: Components of APACHE III APS + APACHE III age score + APACHE III chronic health variables + ICU admission source + elective surgery + 124 diagnoses + treatment limitation + lead time + hospital admission source + APACHE II chronic respiratory disease + ventilation.
- ANZROD: Same as model 3 but a separate model for each major disease group

Model performance measure	Ideal value	APACHE III-j	Model 1	Model 2	Model 3	ANZROD
Homer-Lemeshow						
C statistic	0	3404.2	497.3	367.6	399.0	189.5
H statistic	0	4359.4	534.6	374.8	405.9	174.1
Brier score	0	0.074	0.069	0.068	0.066	0.065
SMR	1	0.83	1	1	1	1
Cox calibration regression						
Intercept	0	-0.49	0	0	0	0
Slope	1	0.88	1	1	1	1
AUROC	1	0.887	0.892	0.896	0.902	0.905

Table 1.1. Derivation data set analysis [5], [6]

Model performance measure	Ideal value	APACHE III-j	Model 1	Model 2	Model 3	ANZROD
Homer-Lemeshow						
C statistic	0	1596.6	259.7	196.7	228.3	104.9
H statistic	0	2087.3	255.9	198.5	208.9	111.4
Brier score	0	0.075	0.07	0.069	0.067	0.066
SMR	1	0.84	1.01	1.02	1.01	1.01
Cox calibration regression						
Intercept	0	-0.48	0.02	0.02	1.01	-0.01
Slope	1	0.88	1	1	0.99	0.98
AUROC	1	0.885	0.891	0.895	0.90	0.902

Table 1.2. Validation data set [5], [6]

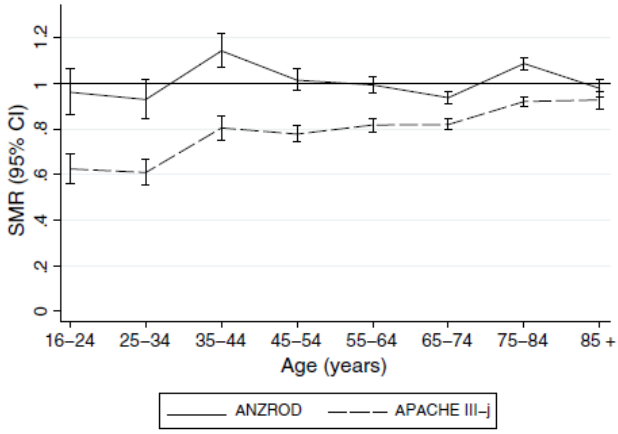


Figure 1.6. Performance of ANZROD vs APACHE III-j [5], [6]

Ultimately, the ANZROD model brings a sense of novelty in the domain of prediction tools, proving that for the New Zealand and Australian population this proposed model can have batter performances in regards to death prediction, as seen in figure 1.6. The authors of the analysed work do concede that further work will be required to improve this model and that the current development they have achieved will serve as a stepping stone for further research. [5], [6]

Assessing contemporary intensive care unit outcome: development and validation of the Australian and New Zealand Risk of Death admission model

This work[7] shows an in-depth analysis of the $ANZROD_0$ model, a new and improved variant of the previously discussed ANZROD prediction algorithm. The approach in regards to the development of this method is quite similar to the one used in the creation of the ANZROD model as a replacement for its previously used predecessor, the APACHE III-j model. The main difference between the ANZROD and $ANZROD_0$ models is the fact that the ANZROD model utilizes the data from the first 24 hours since the admission to the ICU ward, while the $ANZROD_0$ model also takes into consideration the data presented at the moment of the admission ICU ward, hence the full title for the algorithm approached in this work: the Australian and New Zealand Risk of Death admission model. [7]

The ANZICS Adult Patient Database (APD), one of the world's biggest collections of ICU patient data, was used to extract the information. The APD has information on more than 1.6 million distinct ICU episodes of treatment. Currently, more than 80% of all adult ICU admissions in ANZ24, or around 130,000 ICU admissions per year, provide data to the APD. Each ICU director submits data, and each hospital agrees to authorize its usage as necessary in accordance with established protocols. The study was carried out in compliance with the established procedures of the ANZICS CORE Management Committee (ANZICS). Patients hospitalized for palliative care, organ transplants, and those with unreported hospital outcomes were all eliminated, as were readmissions to an ICU during the same hospital stay. In accordance with the protocol, the ANZROD model also disqualified patients who had a blank Acute Physiology score on ICU Day 1. The study received approval from the Monash University Human Research Ethics Committee, and all retrieved data were de-identified. Each patient's age, location prior to ICU admission, surgical status, admission diagnosis, presence of chronic diseases, and existence of treatment restrictions were all retrieved. Patients who were sent home, moved to a hospital for chronic care or rehabilitation, or transferred from one hospital to another were not included in the analysis since the outcome under consideration was "in hospital mortality". Age, location prior to ICU admission, surgical status, admission diagnosis, presence of chronic diseases, and presence of treatment restrictions were among the data collected for each patient. Patients who received a home release, were sent to a hospital for chronic care or rehabilitation, or were moved from one hospital to another were not included. [7]

The actual development of the $ANZROD_0$ model followed a very similar path to the one of the ANZROD model, even going as far as using it as a basis. Thus, the first step involved identifying all the candidate variables necessary for an accurate prediction. The final set included all the previously discussed ANZROD

variables and adding to them the Acute Physiology score and the ventilation status. The second step dives straight into the development of the model. Firstly, with the acquired data an estimate for the in-hospital death is made using a logistical regression model. A chi-square model was then applied for each variable in order to find out its relative importance for the in-hospital death estimate. Some a priori criteria for the model performance were: AUROC (Area Under the Receiver Operating Characteristic) curve >= 0.85, acceptable calibration consisting in a Brier score < 0.1 and SMR (standardized mortality ratio) > 1.0. Then, multiple models were developed with either added or missing variables in order to comparatively asses their performance, and the model with the most accurate prediction was chosen. The validation of the model was done by assessing ANZROD and $ANZROD_0$ performances side by side, using AUROC, Brier score and SMR. Last but not least, the translational equation was developed. In order for this to happen, the distributions of the risk of death prediction were evaluated for both ANZROD and $ANZROD_0$. Then, a logarithmic transformation is applied to the distributions and in order to obtain linear relations between the RODs (predicted risk of death) linear regressions were used. Finally, estimations of the intercept and slope parameters and the coefficient of determination (R^2) were reported. [7]

The results were a bit surprising. In the validation sample, the $ANZROD_0$ model had an adjusted Brier score of 0.19 and an AUROC of 0.851 (95% confidence interval, 0.849 to 0.853). SMRs for the validation set across the main diagnostic categories varied from 0.97 to 1.06. Compared to the ANZROD model, the model had lower AUROC and Brier score values, but the SMR was about the same for both. Calibration curves also showcase better performances for the ANZROD model, as depicted in figure 1.7. [7]

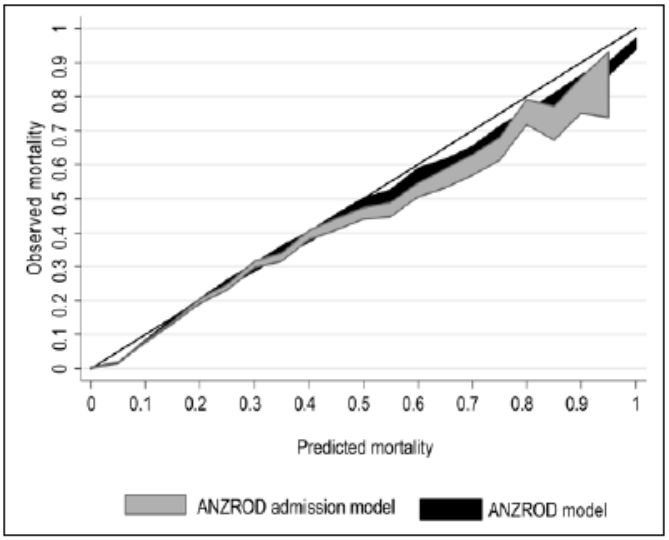


Figure 1.7. Calibration curves of ANZROD and $ANZROD_0$ [7]

With correlations within key diagnostic categories ranging from 0.33 to 0.8, the two models' estimated death probability had a correlation of 0.73. In five of the eight primary diagnostic groupings, the ANZROD projected risk of mortality generated using ANZROD0 had an AUROC of 0.8 or above. In the end, the model was modified to exclude one of the APACHE II chronic health variables (chronic cardiovascular disease), one of the seven APACHE III chronic health variables (acquired immune deficiency syndrome), the source of hospital admission, and four of the seven APACHE III chronic health variables (hepatic failure, lymphoma, immunosuppression, and lymphoma). The performance comparisons between ANZROD versus $ANZROD_0$ can be assessed with figure 1.8 and table 1.3. [7]

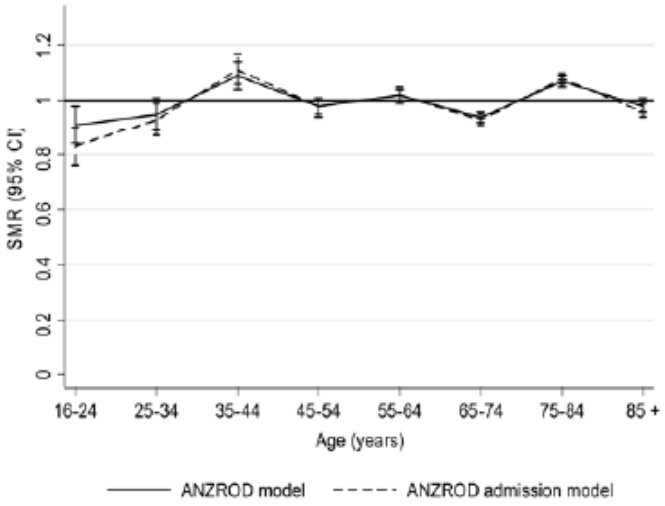


Figure 1.8. Performance of ANZROD versus ANZROD$_0$ models [7]

	ANZROD	**ANZROD$_0$**
Hosmer-Lemeshow		
C statistic	252.9	221.4
Brier Score	0.057	0.069
Adjusted Brier Score	0.333	0.194
AUROC, 95% CI	0.909 (0.907 - 0.910)	0.853 (0.851 - 0.854)
SMR	1.00 (0.99 - 1.01)	1.00 (0.99 - 1.01)

Table 1.3. Model performance with the validation sample)

In the end, ANZROD$_0$ thanks to in-hospital mortality risk may be calculated independently of the treatment received in the ICU thanks to its forecasts of hospital mortality, which have adequate calibration and discrimination and are complimentary to the ANZROD model. With the possible exception of heart surgery, trauma, and sepsis, all main diagnostic groupings had high correlations between estimated mortality probabilities from ANZROD0 and ANZROD. Although ANZROD$_0$ could be useful in ICUs with limited resources, more research is necessary before suggesting its use in everyday practice. [7]

Individual outcome prediction models for patients with COVID-19 based on their first day of admission to the intensive care unit

As stated in the beginning of this work, the COVID era made it abundantly clear that the introduction of AI into the development of prediction models is not only inevitable, but also completely reasonable taking into consideration the advancements made by the latest technology. This study[1] proves that in the context of a global crisis, where many ICU wards are overwhelmed, a death prediction model that uses binary logistic regression (BLR) and artificial neural networks (ANN) analysis of the data collected in the first 24 hours spent in the intensive care units is crucial for ensuring the highest possible quality care for patients at a smaller human cost paid by the medical staff. [1]

The proposed solution consists in an individual outcome prediction model for COVID-19 patients based on BLR assessed using Wald Forward Stepwise method and an ANN model constructed with the help of a multilayer perceptron architecture (MLP). The data was acquired in Barcelona, Spain, in a 750-bed tertiary

care public hospital for adults. The study was granted ethical committee permission (approval code PR168/20) and, due to the observational nature of the study, the mandated isolation precautions used during in-hospital care, and the fact that the data were anonymized for analysis, informed permission was not required. In this study, which covered the period from March 15, 2020, to March 15, 2021, during the first year of the pandemic, 326 patients (age 18 or older) who were admitted to the ICU were included. ICU admission requirements for patients included at least one of the following: patients who needed invasive mechanical ventilation (IMV) due to respiratory failure, patients with non-invasive mechanical ventilation (NIMV) for deteriorating respiratory problems, and patients with other organ failure who needed ICU monitoring. Results included patients admitted to the ICU who lived and were later released from the hospital as well as non-survivor patients who passed away while they were there. [1]

In the acquired data information on demographics, clinical history (comorbidities and pharmaceutical therapies), symptoms onset, hospitalization, and standard laboratory variables were all included. Patient outcomes were compared using this data. Age and gender were included in the demographics. Comorbidities were categorized as dichotomous categorical variables and included cancer, heart disease, chronic renal disease, chronic liver disease, chronic obstructive pulmonary disease, diabetes, dyslipidemia, hypertension, obesity, and smoking (presence or absence). At the time of hospital admission, antihypertensives, anticoagulants, anti-aggregants, and immunosuppressive/corticosteroid medications were all taken into consideration. Information on the interval between the beginning of clinical symptoms and admission to the hospital, the ICU, and the interval between hospital admission and ICU admission is referred to as the "symptoms onset and admission information". According to the IFCC's guidelines, the biological quantities employed for COVID-19 patients' monitoring were the focus of the laboratory routine variables: plasma concentrations of alanine transaminase (ALT), albumin (ALB), aspartate transaminase (AST), bilirubin (BIL), calcium (CA), chloride (CL), creatinine (CREA), C-reactive protein (CRP), D-dimer (DD), ferritin (FERRI), glucose (GLU), interleukin 6 (IL6), lactate dehydrogenase (LDH), potassium ion (K), procalcitonin (PROCAL), prothrombin time (PT), sodium ion (NA), troponin T (TROP-T), and urea (UREA); Chronic Kidney Disease Epidemiology Collaboration glomerular filtration rate (CKD-EPI); arterial blood partial pressure of carbon dioxide (paCO2), arterial blood partial pressure of oxygen (paO2); pH in arterial blood (apH), arterial blood oxygen saturation (aSatO2), partial pressure of oxygen in arterial blood/fraction of inspired oxygen quotient value (paO2/FiO2); blood count of basophils (#BAS), eosinophils (#EOS), leucocytes (#LEU), lymphocytes (#LYM), monocytes (#MON), neutrophils (#NEU), and platelets (PLT); leucocytes relative differential blood count (%BAS, %EOS, %LYM, %MON, and %NEU); blood count of erythrocytes (ERY), blood concentration of hemoglobin (HGB), hematocrit (HCT), mean cell volume (MCV), mean cell hemoglobin (MCH), mean cell hemoglobin concentration (MCHC), mean platelet volume (MPV), and red blood cells distribution width based on the coefficient of variation (RDW-CV). [1]

The BLR analysis was performed on all of the previously discussed variables in order to identify independent predictors that can be used for the development of a prediction model. To prevent multicollinearity, the highly correlated variables (such as CKD-EPI, which includes CREA, age, and gender) were eliminated before analysis. Accordingly, factors including CKD-EPI, #BAS,%EOS, #LYM,%MON, #NEU, MPV, HCT, HGB, MCV, and MCH were removed based on the univariate analysis's p-value results. The predictive variables were chosen using the Wald Forward Stepwise technique, with a cut-off of 0.5, a maximum of 20 iterations, and stepwise probabilities of p 0.05 (for entry) and p 0.10 (for outliers) (for removals). To build the BLR model, patient data were initially randomly split into a training dataset with 70% of the cases (n = 234) and a test dataset with 30% of the cases (n = 92). This distribution of the data was carried out to resemble the ranges that the ANN model developed. [1]

The ANN analysis used a MLP architecture. The network had an input layer with 75 neuron units based on standardized variables (15 dichotomic factors and 47 covariates), a hidden layer with six neuron units, and an output layer with two neuron units (the outcomes). Gender and clinical history data were dichotomous factors, whereas laboratory variables, symptoms, and admission variables were covariates. For the hidden layer, they utilized the hyperbolic tangent activation function, and for the output layer, they used the Softmax activation with the Cross-entropy error functions. The ANN loss function was optimized using the gradient descent-based approach. Using a tenfold cross-validation procedure with a 7:3 training/test sample ratio (234:92 patient data), the ANN model was verified. [1]

The results of the two models were quite remarkable. The univariate analysis revealed that older age, cancer, chronic renal disease, immunosuppressive/corticosteroid medication, and a longer period of time between the

onset of clinical symptoms and admission to the ICU were all linked with an increased risk of mortality in ICU patients ($p < 0.01$). Higher values of CREA, DD, K, TROP-T, UREA, paCO2, and RDW-CV as well as lower values of CKD-EPI, paO2, apH, aSatO2, paO2/FiO2,%BAS,%LYM, and MCHC in relation to laboratory variables were also linked to a higher risk of death ($p < 0.05$). Age, hypertension, chronic renal disease, K, CL, apO2, and apH were independent predictors of death in ICU patients with COVID-19 according to multivariate analysis for the BLR model utilizing stepwise variable selection based on the Wald Forward technique ($p < 0.001$). Based on the patients' initial day of ICU admission, the multivariate logistic regression equation used to calculate the percentage probabilities of death ($P_{death}(\%)$) and survival ($P_{survival}(\%)$) for COVID-19 patients was as seen in equations 2.1 and 2.2 [1]:

$$P_{death}(\%) = \frac{1}{1 + e^{(-76.81 - 0.112 * x1 + 1.050 * x2 - 2.537 * x3 - 0.945 * x4 + 0.116 * x5 + 0.020 * x6 + 9.986 * x7)}} * 100 \quad 1.1$$

$$P_{survival}(\%) = 100 - P_{death}(\%) \quad 1.2$$

where x1 is the age (in years); x2, the dichotomic hypertension value (0 if the patients do not present hypertension, and 1 if they do); x3, the dichotomic chronic kidney disease value (0 if the patients do not have chronic kidney disease, and 1 if they have it); and x4, x5, x6, x7, the K, CL, paO2, and pH values obtained from the patients during the first day of their ICU admission. The AUROC, sensitivity, specificity, PPV, and NPV statistics were 0.810 (95% CI, 0.762-0.858), 78.9%, 74.3%, 37.2%, and 94.8%, respectively, for the training data group. These percentages were 0.789 (95% confidence interval [CI], 0.739-0.839), 77.4%, 75.7%, 38.1%, and 94.5% for the testing data group. No statistically significant differences between the two groups could be seen when the AUROCs were compared ($p = 0.542$). For the ANN model, the Hosmer and Lemeshow C goodness-of-fit value was non-significant (= 0.05; $p = 0.843$). The AUROC, sensitivity, specificity, PPV, and NPV for the training data group were 0.917 (95% CI, 0.873-0.961), 80.4%, 89.8%, 60.4%, and 95.9%, respectively. The testing data group's sensitivity, specificity, PPV, and NPV values were 84.2%, 87.6%, 56.8%, and 96.6%, respectively. The factors chronic pulmonary illness, gender, and the "use of antihypertensive and antiaggregant medications at hospital admission" were placed last, whereas the variables "number of days between manifestation of clinical symptoms and admission to the hospital", DD, aSatO2, and pH were ranked first. When utilizing data gathered on the first day of ICU admission to predict the individual outcomes of COVID-19 patients, the AUROC of the ANN model was considerably higher than the BLR model (0.917 vs 0.810; p 0.001), as seen in figure 1.9. Additionally, the post-test probability demonstrated that the ANN model's PPV was greater than the BLR model's (60.4% vs. 37.2%), but that the NPV was comparable (95.9% vs. 94.8%). [1]

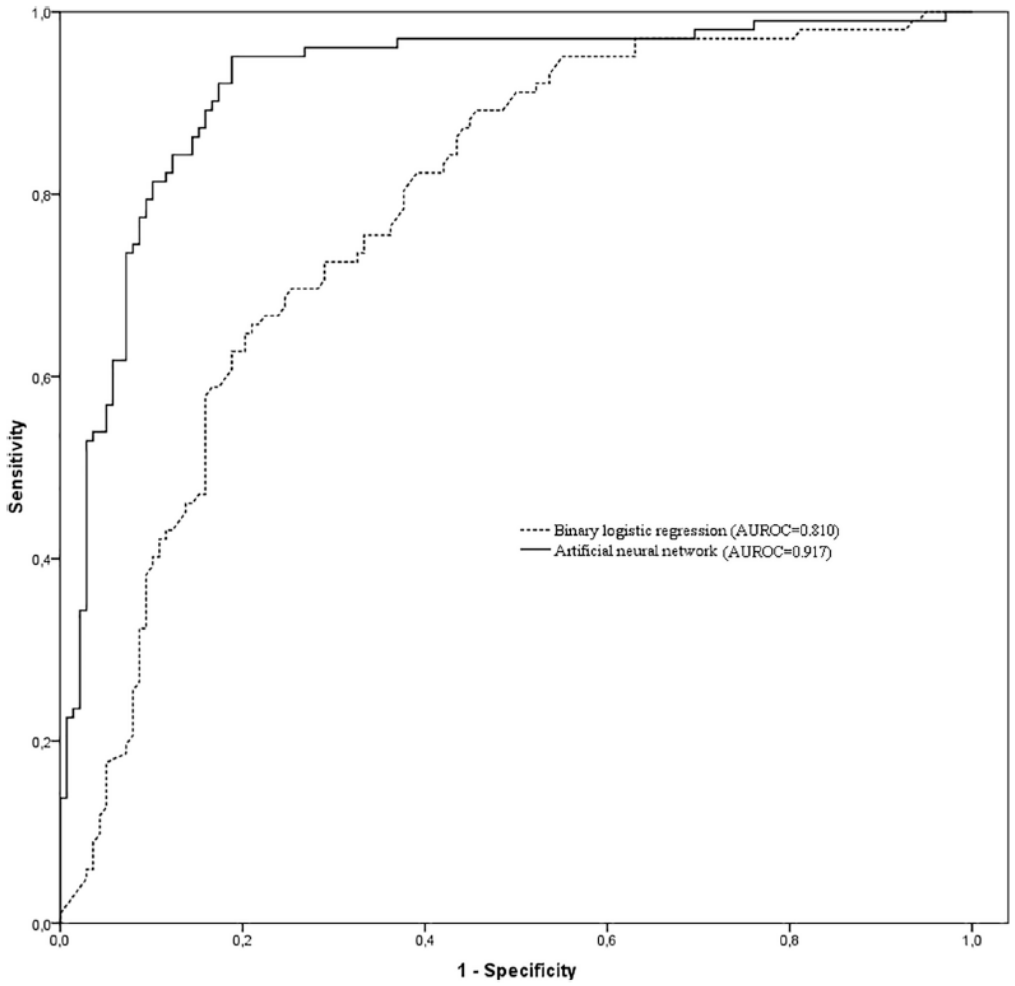


Figure 1.9. Comparison of AUROC for BLR and ANN [1]

This study provides an amazing start for the development of multiple prediction algorithms that make use of BLR and ANN architectures. Even if the ANN algorithm seems to perform better, due to the fact that the used dataset can be considered quite small (only 326 cases), its prediction power is cut short, being prone to bias and vulnerable to cases that are external to the dataset. Future prospects should take into consideration the usage of a much larger dataset in order to offer conclusive results for this algorithm, not to mention that the data needs to be collected on a larger area and period of time, in order to have more diversity and to ensure that this solution doesn't overperform in a specific area (namely, where the study where data was gathered was conducted) and underperforms in all other regions. [1]

1.3.Chosen method for the hereby project

Considering the studies showcased previously and the initial hypothesis, the method which will be implemented in this hereby project will be based on the *Individual outcome prediction models for patients with COVID-19 based on their first day of admission to the intensive care unit* study [1] which had a very interesting and modern approach of the issue of development of a prediction algorithm for death prediction using the data gathered in the first 24 hours of ICU care. The usage of ANNs is proven to be very effective, however the employed data base needs to be larger in order to achieve conclusive results. The advancements made in the development of $ANZROD_0$[7] aren't something to be discarded, and the presented procedures will also be taken into consideration in the development of the current proposed approach.

2. Data Description

The data set we have chosen for the hereby project belongs to the 2020 Global Women in Data Science (WiDS) at Stanford Datathon, that focused on patient health through data from MIT's GOSSIS (Global Open Source Severity of Illness Score) initiative. The challenge of this Datathon was to create a model that uses data from the first 24 hours of intensive care to predict patient survival. This precondition already supports our choice of this dataset for this thesis, since it was built specifically with the same requirement in mind. [8]

MIT's GOSSIS community initiative, with privacy certification from the Harvard Privacy Lab has provided a dataset of more than 130 000 hospital Intensive Care Unit (ICU) visits from patients, spanning a one-year timeframe. This data is part of a growing global effort and consortium spanning Argentina, Australia, New Zealand, Sri Lanka, Brazil, and more than 200 hospitals in the United States. Therefore, not only is the dataset far more comprehensive size-wise than the ones employed by the works referenced before, but it also has a far larger diversity of the studied ethnic groups due to the large geographical areas in which the study was conducted. This diversity will undoubtedly reduce the impact of the problem stated in the ANZROD study, in which it was stated that survival prediction algorithms are bound to have better results in the areas where they were developed and poorer results in other geographical areas.

Labelled training data are provided for model development as well as a dictionary for the features and an unlabelled test set on which the models are supposed to be tested. The dictionary has a total of 6 columns and 187 entries, which means that we have a total number of 187 described features. Further on, in this section we will describe each category of features and their meaning.

There are 10 categories of features inside the dictionary (as seen in figure 2.1):

- Labs (60 entries)
- Vitals (52 entries)
- Apache covariate (28 entries)
- Demographic (16 entries)
- Labs blood gas (16 entries)
- Apache comorbidity (8 entries)
- Identifier (3 entries)
- Apache prediction (2 entries)
- Apache grouping (2 entries)
- GOSSIS example prediction (1 entry)

```
In [6]: data_dictionary["Category"].value_counts()

Out[6]: labs                        60
        vitals                      52
        APACHE covariate            28
        demographic                 16
        labs blood gas              16
        APACHE comorbidity           8
        identifier                   3
        APACHE prediction            2
        APACHE grouping              2
        GOSSIS example prediction    1
        Name: Category, dtype: int64
```

Figure 2.1. Data dictionary categories

Labs category

The Labs category contains all the laboratory works features collected both in the first hour of their admission to the ICU ward and the ones collected during the first 24 hours since the admission. In order to differentiate between these features, we have the label "h1_" for the hourly rates and "d1_" for the daily rates. Furthermore, we have the minimum and maximum values for each of these features, labelled "_min" and "_max". The laboratory works results can give us vast information on whether there is something wrong with the essential bodily functions of the patient. Since most of these values cannot be acquired electronically through sensors in real time, we need to wait a considerable amount of time to receive them from the laboratory analysis, which can greatly influence the response time of the doctors in case of a sudden worsening of the patients' state. Therefore, in this work we will analyse one model that only takes into consideration the values that we can obtain immediately through the medical records of the patient or through electronic means.

"d1_albumin_max"," d1_albumin_min"," h1_albumin_max"," h1_albumin_min" are the features containing the highest and lowest albumin concentration of the patient in their serum during both the first hour and during the first 24 hours of their unit stay. Serum albumin can also be referred to as blood albumin and is a type of protein that can be found in vertebrate blood, such as humans, cattle and more. This protein is one of the most common types of complex molecules in the human body and even though it is produced by the liver, it is usually found in the blood plasma under a dissolved format. This substance is essential for the normal distribution of liquids inside the body. For example, an abnormal serum albumin level can lead to a disturbance in the colloid osmotic pressure that can cause the body to increase or decrease the level of fluids in the capillaries. Albumin also acts like a carrier of quite a few types of steroids, hormones and fatty acids and like a binder for calcium in blood serum. The normal range for albumin levels should be between 34 and 54 grams per litter. Abnormal levels of albumin in the plasma can be divided into two affections: hypoalbuminemia (when the albumin is at a lower level than the normal range) and hyper albuminemia (when the albumin level is at a higher level than the normal range). Hypoalbuminemia can be caused by several factors such as: renal loss, gut loss, intravascular volume expansion, low albumin production, sepsis and critical illness, heart failure, burns and nutritional deficiency. On the other hand, hyperalbuminemia can be caused by dehydration, metabolic syndrome or insulin resistance. [9]–[14]Our dataset tells us that the albumin levels have been registered as numerical values with the unit of measure "g/L", therefore the normal range in which we should have expected values is the one that we discussed previously.

"d1_bilirubin_max", "d1_bilirubin_min", "h1_bilirubin_max", "h1_bilirubin_min" are the features containing the highest and lowest bilirubin concentration of the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay. Bilirubin is a compound of red-orange colour that appears as an effect of the breaking down process of old red blood cells inside the bile. The bile is secretion made by the liver and stored in the gallbladder, whose main role is to help with digestion, more specifically, to break down fats into fatty acids compounds that can be further metabolised by the body through the rest of the digestive system. [15]–[19] A serum bilirubin test is done in order to assess the health of the patients' liver. If the liver is healthy, then we should have very low levels of bilirubin in the blood, the reason being that the liver should be able to filter most of it. If bilirubin is found in urine tests, then that can be a good indicator of liver damage. Jaundice is an affliction that is characterised by high levels of bilirubin inside the body. People suffering from jaundice have a certain aspect, characterised by a yellow-ish colour of the skin and the whites of the eye. Jaundice is very common among new-borns and often not harmful, since they can be born in excellent health but without fully developed livers. This condition usually clears up after a certain period of time, mostly spanning about a few weeks. The normal range for the total serum bilirubin level is 1.71 to 20.5 μmol/L. Low bilirubin levels shouldn't be any cause for concern, and patients usually don't even notice this fact until they've had a routine bloodwork test done. On the other hand, high bilirubin levels can be quite concerning. High bilirubin usually manifests itself through visual symptoms such as jaundice, dark urine or stomach pain, which could indicate various health complication such as: blockages in the bile ducts, liver diseases such as cirrhosis or hepatitis of a general problem with the breaking down process of breaking down red blood cells called hemolytic anemia. Abnormally high levels of bilirubin can often lead to brain damage, especially in infants, so this test is often done as a precaution. In our data set, the bilirubin entries are of numeric type and the unit of measure is specified as micromol/L, therefore, we can expect numbers in the previously described range.[20]–[22]

“d1_bun_max”, “d1_bun_min”, “h1_bun_max”, “h1_bun_min” are the labels referring to the features containing the highest and lowest blood urea nitrogen (BUN) concentration of the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay. [23], [24] BUN was one of the first widely used endogeneous filtration marker for evaluating the health of the kidneys of a patient. Blood urea nitrogen is a serum byproduct of the protein metabolization process. Doubling the assement marker for kidney health, BUN is also considered to be one of the oldest early prediction markers for heart failure. Urea is formed by the liver and is carried by the blood to the kidneys for excretion. If the liver is affected by a disease or otherwise has some sort of damage then the BUN levels spike in the blood and the GFR (glomerular filtration rate) values drop. High BUN levels can also be indicative of a high protein diet, shock, heart failure or any kind of bleeding into the gastrointestinal tract. Low levels of BUN are indicative of a low protein diet, malnutrition or severe liver damage. The normal range of BUN levels should be between 2.1 and 8.5 mmol/L. In our data set, the BUN entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range.

“d1_calcium_max”, “d1_calcium_min”, “h1_calcium_max”, “h1_calcium_min” are the features containing the highest and lowest calcium concentration of the patient in their serum during both the first hour and during the first 24 hours of their unit stay. Calcium is one of the most abundant substances inside the human body. This mineral is the main building material of any kind of bony tissue, such as teeth or bones. Good calcium absorption results in the maintenance of the normal range of motion and gives the bones their unique properties of rigidity, strength and, paradoxically, flexibility. In serum form, calcium found in the circulatory system is a good indicator of whether the individual has any form of blood clotting or if the muscular, nerve transmission and hormonal secretion functions are working as expected. The absorption of calcium is done through the digestive system, through the mucous membrane lining the digestive tract. The mineral is synthesised from various food sources, such as dairy, fish and certain vegetables, such as kale or broccoli. Other ways through which calcium cand be ingested is through the form of supplements, such as pills or effervescent tablets. In order for the calcium to be efficiently absorbed through the digestive lining and transported through the blood, the body needs a certain amount of Vitamin D, which can be acquired through dietary and non-dietary supplements, or, in a simpler manner, by making sure that the patient is getting an optimal amount of sunlight. Normal calcium levels should be between 2.2 and 2,7 mmol/L, but certain laboratories can allow a slight variation. Lower levels of calcium in the body, further referred to as calcium deficiency, can be a warning signal for a great number of diseases. Calcium deficiency is directly linked to osteoporosis, an affliction characterised by weak and fragile bones and a tendency of falling. Although this disease is common in patients with a higher age, children can suffer from it too. A more common disease in children that stems from the same cause, is rickets, a disease that consists in the abnormal mineralization of the growth cartilage. Hypocalcemia can usually be traced back to a vitamin D or magnesium deficiency, hypoparathyroidism, critical illnesses and usage of certain medicine. On the other hand, higher levels of calcium, to be further referred to as hypercalcemia or hypercalciuria (for calcium found in urine), can come with its own plethora of associated risks. For example, abnormally high levels can indicate the presence of cancer, primary hyperparathyroidism, renal insufficiency, heart arrhythmias and many other ailments. In our data set, the calcium entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range. [25]

“d1_creatinine_max”, “d1_ creatinine _min”, “h1_ creatinine _max”, “h1_ creatinine _min” are the labels referring to the features containing the highest and lowest creatinine concentration of the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay. The breakdown of creatine phosphate from the metabolism of proteins and muscles results in creatinine. It is continuously secreted by the body (depending on muscle mass). Since it is an easily measurable result of muscle metabolism that is eliminated unaltered by the kidneys, serum creatinine is an essential biomarker of kidney health. Creatinine is synthesized by a biological mechanism that includes creatine, phosphocreatine (commonly known as creatine phosphate), and adenosine triphosphate (ATP, the body's immediate energy supply). It is subsequently delivered by blood to various organs, muscle, and the brain, where it is phosphorylated to form the high-energy molecule phosphocreatine. Creatine kinase catalyzes the conversion of creatine to phosphocreatine; spontaneous creatinine production occurs throughout the process. The normal range for creatinine is different for men and women, therefore we have 61.9 to 114.9 µmol/L for men and 53 to 97.2 µmol/L for women. [26] A higher level of creatinine than normal can be an indicator of kidney damage or failure, blocked urinary tracts or muscle problems, while lower values are usually linked to malnutrition and various muscular and neurological disorders that involve the decrease of muscle mass. In our data set, the creatinine entries are of

numeric type and the unit of measure is specified as micromol/L, therefore, we can expect numbers in the previously described range. [27]–[29]

“d1_glucose_max”, “d1_ glucose _min”, “h1_ glucose _max”, “h1_ glucose _min” are the features containing the highest and lowest glucose concentration of the patient in their serum during both the first hour and during the first 24 hours of their unit stay. The chemical formula for glucose is $C_6H_{12}O_6$. Glucose is the most prevalent monosaccharide, a kind of carbohydrate. Glucose is primarily produced by plants and most algae during photosynthesis from water and carbon dioxide with the help of sunshine, where it is utilized to produce cellulose in cell walls, the world's most abundant carbohydrate. The World Health Organization lists glucose as an essential medicine in the form of an intravenous sugar solution. [6] It is sometimes included in conjunction with sodium chloride. [6],[30], [31] In humans, glucose is one of the most important sources of energy for almost every cell, but most importantly, for the brain. This type of carbohydrate can be naturally found in various foods, such as fruit, bread and a multitude of grains, such as wheat, oats, rice and many more. The World Health Organisation stated that the expected range of values for fasting blood glucose should be between 3.9 mmol/L and 5.6 mmol/L.[32] Usually, blood glucose tests are performed on patients who are undergoing investigation or treatment for diabetes, but there are other diseases that can cause a spike of the glucose concentration in a patients serum, such as: pancreatic cancer, pancreatitis, glaucoma and other rare tumors, overactive thyroid gland, stroke, heart attack, and many others. Hypoglycemia can indicate pituitary gland disorders, underactive thyroid gland, malnutrition, too much insulin, liver or kidney disease, and so on. In our data set, the glucose entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range.[33]

“d1_hco3 _max”, “d1_ hco3 _min”, “h1_ hco3 _max”, “h1_hco3 _min” are the labels referring to the features containing the highest and lowest bicarbonate concentration of the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay. [34] Bicarbonate is a form of CO_2 (carbon dioxide), a compound left behind after the burning (i.e. consumption) of food for energy inside the body. A bicarbonate test is usually done as part of a larger electrolyte test that evaluate whether the individuals’ body can keep itself hydrated and that the blood ph level is in the right range. Normal ranges of bicarbonate concentration vary between 23 and 29 mmol/L. High levels of bicarbonate can be good indicators of lung diseases, anorexia or adrenal gland issues, while low values can be important pointers to kidney disease, aspirin overdose, diabetic ketoacidosis (very low levels of insulin) or metabolic acidosis (the body makes too much acid). In our data set, the bicarbonate entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range.

“d1_hemaglobin_max”, “d1_hemaglobin _min”, “h1_hemaglobin _max”, “h1_hemaglobin _min” are the features containing the highest and lowest hemoglobin concentration of the patient in their serum during both the first hour and during the first 24 hours of their unit stay. Hemoglobin is the basic protein of red blood cells. Its main function is to ensure that the blood has the ability to carry and appropriate amount of oxygen to all the cells in the body. This test measures the amount of hemoglobin in the body and it’s one of the most common bloodwork tests that are done. Normal value ranges differ for men and women; therefore, we have 13.8 to 17.2 grams per decilitre for men, and 12.1 to 15.1 g/dL for women. Lower values than these ranges may suggest afflictions such as anemia, bleeding, low red blood cell count, leukimia, malnutrition, kidney disease or hemolysis, while higher values than normal can most of the time be traced back to hypoxia (low oxygen level in the blood), failure of the right side of the heart, scarring or thickening of the lungs or dehydration. In our data set, the hemoglobin entries are of numeric type and the unit of measure is specified as g/dL, therefore, we can expect numbers in the previously described range.[35]

“d1_hematocrit _max”, “d1_hematocrit_min”, “h1_hematocrit_max”, “h1_hematocrit_min” are the labels referring to the features containing the highest and lowest volume proportion of red blood cells in a patient's blood during both the first hour and during the first 24 hours of their unit stay. The hematocrit compares the amount of red blood cells to the total volume of blood (red blood cells and plasma). Men have a normal hematocrit of 40 to 54%, while women have a hematocrit of 36 to 48%. This value can be estimated directly or indirectly using microhematocrit centrifugation. The hematocrit is calculated by multiplying the red cell number (in millions/mm3) by the mean cell volume (MCV, in femtoliters). When tested in this manner, it is susceptible to the uncertainties inherent in achieving an exact measurement of the MCV. Low levels of hematocrit can indicate anemia, polycythemia or erythrocytosis. In our data set, the hemoglobin entries are of numeric type and the unit of measure is specified as fraction, therefore, we can expect numbers in the previously described range. [36], [37]

“d1_inr_max”, “d1_inr_min”, “h1_inr _max”, “h1_inr _min” are the features containing the highest and lowest international normalized ratio for the patient during both the first hour and during the first 24 hours of their unit stay.[38] For individuals on vitamin K antagonists, the recommended test is the international normalized ratio (INR) (VKA). It can also be used to measure a patient's risk of bleeding or coagulation status. Since INR varies between patients, patients using oral anticoagulants must monitor their INR to modify their VKA dosage. The INR is generated from prothrombin time (PT), which is determined as a ratio of the patient's PT to a control PT normalized for the potency of the World Health Organization's (WHO) thromboplastin reagent using the following formula: INR = Patient PT ÷ Control PT. The time required in plasma to form a clot in the presence of appropriate calcium and tissue thromboplastin concentrations by initiating coagulation via the extrinsic route is measured in seconds. The reference values for INR take into consideration device-related changes, reagent type, and sensitivity differences in the TF activator when measuring PT. The INR value is dimensionless and varies from 2.0 to 3.0. Tuning the patient's INR therapeutic range can be difficult because to the limited therapeutic range observed in VKAs, which can be influenced by patient features, co-morbid conditions, food, and other medication interactions. Patients are checked at thrombosis centers (TC), point-of-care (POC) clinics, or at home every 3-4 weeks or fewer. In our data set, the INR entries are of numeric type and the unit of measure is specified as micromol/L, therefore, we can expect that the numbers may differ from the previously described range.

“d1_lactate _max”, “d1_lactate _min”, “h1_lactate _max”, “h1_lactate _min” are the labels referring to the features containing the highest and lowest lactate concentration for the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay.[39], [40] Lactic acid is created in physiologically normal processes and is frequently seen in pathological conditions. When increased production is combined with poor clearance, the clinical course becomes more severe. Furthermore, excessively high lactic acid levels can have serious hemodynamic repercussions and could result in mortality. Serum lactate levels can serve as both a danger indicator and a treatment target. The higher the level of increased serum lactate and the longer it takes to normalize, the greater the risk of mortality. This exercise examines the genesis, presentation, diagnosis, and management of lactic acidosis, as well as the interprofessional team's involvement in assessing, diagnosing, and treating the disease. Lactate concentrations within the normal range (quartile 2, lactate 1.4Y2.3 mmol/L) are as relevant as higher lactate concentrations above the normal range (quartile 3, lactate 2.3Y4.4 mmol/L) as a predictive indication of unfavourable outcomes in septic shock. Additionally, low lactate concentrations may indicate a favourable response to vasopressin infusion vs noradrenaline infusion. Our findings imply that the threshold value for blood lactate-guiding treatment in septic shock patients should be revised [41] . In our data set, the INR entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range.

“d1_platelets_max”, “d1_platelets_min”, “h1_platelets_max”, “h1_platelets_min” are the features containing the highest and lowest international normalized ratio for the patient during both the first hour and during the first 24 hours of their unit stay. [42] Platelets, also called thrombocytes, are parts of the blood that help form a blood clot in the case of injury. These cells are smaller than red or white blood cells and their number can be a good indicator of various diseases. The normal range for a platelets count is 150 to 400 X 10^9/L. A low platelet count can be a good indicator of autoimmune disorders, various drugs and medications or traces of cancer treatments present in the body. A high platelet count, also referred to as thrombocytosis, can be caused by iron deficiency, major surgery or trauma, cancer, bone marrow disease or spleen removal. A high risk associated with thrombocytosis is the formation of blood clots in the blood stream, which can lead to pulmonary embolism, strokes or heart attacks. In our data set, the platelets entries are of numeric type and the unit of measure is specified as 10^9/L, therefore, we can expect numbers in the previously described range.

“d1_potassium_max”, “d1_potassium_min”, “h1_potassium _max”, “h1_potassium _min” are the labels referring to the features containing the highest and lowest potassium concentration for the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay. [43] Potassium is an essential mineral that is needed in various cells and processes inside the body. When referred to in a context of electrical cellular activity, this mineral can be considered and electrolyte, and is commonly seen working together with sodium to maintain the normal levels of fluids inside and outside our cells.[44]–[48] Potassium levels are controlled by a hormone called aldosterone and its presence is a great aid in the communication between nervous and muscular tissues. This blood test is usually done in order to diagnose or monitor afflictions of the kidney. A normal reading of potassium should be between 3.70 to 5.20 millimoles per litre. Higher levels of potassium can be a good indicator tissue injury, hyperkalemic paralysis, kidney failure, metabolic or respiratory acidosis, red blood cell destruction or just too much potassium in the diet. Low levels of potassium

can be traced back to acute or chronic diarrhea, diuretics, renal artery stenosis, vomiting, and many others. In our data set, the potassium entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range.

"d1_sodium_max", "d1_sodium_min", "h1_sodium_max", "h1_sodium_min" are the features containing the highest and lowest sodium concentration for the patient in their serum or plasma during both the first hour and during the first 24 hours of their unit stay.[49] Sodium is an alkali metal found in various compounds in nature. Inside the human body, sodium is a very important mineral that helps the good development of inter- and intracellular processes, by making sure alongside potassium that the level of fluids inside and outside the cells is within normal ranges. Sodium is a basic electrolyte that the body needs in order to function correctly, and the most common source of sodium is table salt. Normal sodium reading should be between 136 and 145 mmol/L.[50] Higher sodium levels can indicate problems with the adrenal glands, major fluid loss, excessive salt intake in the diet or various types of diabetes. Lower sodium levels can be linked to Addison disease, hyperglycemia, heart failure, kidney diseases, cirrhosis, increased fluid loss, hypothyroidism and many others. In our data set, the sodium entries are of numeric type and the unit of measure is specified as mmol/L, therefore, we can expect numbers in the previously described range.

"d1_wbc_max", "d1_wbc_min", "h1_wbc_max", "h1_wbc_min" are the labels referring to the features containing the highest and lowest white blood cell count for the patient during both the first hour and during the first 24 hours of their unit stay.[51] White blood cells, also known as leukocytes, are the parts of the blood that help fight infections. There are five categories of white blood cells: basophils, eosinophils, lymphocytes, monocytes and neutrophils. Normal ranges for a wbc reading should be between 4.5 X $10^{9/L}$ and 11 X 10^{9}/L. A low white blood cell count can be a good indicator of bone marrow failure, cancer treating drugs, lupus, spleen or liver disease, radiation treatment, viral illnesses, severe bacterial infections or severe emotional or physical stress. A high wbc count can be traced back to smoking, spleen removal surgery, infections, inflammatory diseases, leukimia or Hodgkin disease, tissue damage or pregnancy. In our data set, the wbc entries are of numeric type and the unit of measure is specified as 10^9/L, therefore, we can expect numbers in the previously described range.

Vitals category

This data category contains all the information we have about the vital signs of the patient, acquired both during the first hour and the first 24 hours of their ICU stay. This time, we have two additional notations to some of the labels, unlike for the "labs" category where we had only the labels "h1_" for the hourly rates, "d1_" for the daily rates, "_min" for the minimum values and "_max" for the maximum values of each of these features. The notations are "_invasive_" for the invasively measured values and "_noninvasive_" for the non-invasively measured parameters. The features not specifying if the reading was done invasively or non-invasively, refer to the best obtained value for the feature obtained from either reading depending on the case, or to a value that was obtained as an additional feature from other readings through some specific conventions or algorithms. The invasive and non-invasive categories are references to the types of sensors used in data acquisition. [105] There are multiple ways to classify the sensors used in biomedical applications, but the main two are:

- By patient interaction (as depicted in figure 2.2):
    - Without contact = sensors that don't need a direct contact with the patients' body in order to get a reading, for example the non-contact thermometers that were uses during the pandemic
    - With contact but non-invasive = sensors that touch the patient but don't penetrate the skin barrier, for example the electrode
    - With contact but invasive = sensors that are inserted inside of the patient's body tissue, for example needle electrodes used in electromyography (EMG)

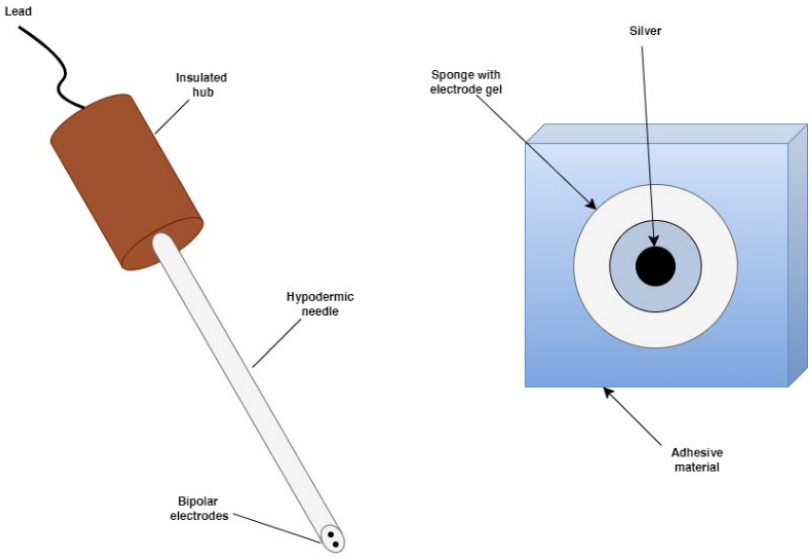


Figure 2.2. Invasive needle electrode vs. non-invasive surface electrode composition

- By sensitive part (as depicted in figure 2.3):
  - Resistive = the sensor acquires the data by detecting changes on their resistance
  - Capacitive = the sensor acquires the data by detecting changes on their capacitive element, found for example in a pressure sensor
  - Optical = the sensors acquire data by sending a beam of light towards the tissue and reading the reflected light on the receiver. This type of sensor is usually used in a photopletysmogram (PPG) reading
  - Inductive = the sensor acquires the data by detecting changes on their inductive element, found for example in blood pressure sensors[52]

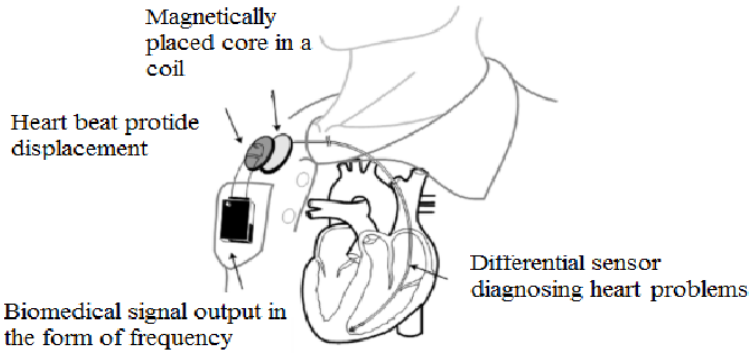


Figure 2.3. Inductive transducer pressure sensor on the respiratory system

The electrode gel is the substance used alongside with electrodes in order facilitate the contact between the Ag/AgCl sensors and the skin of the patient. Gel electrodes allow for very high signal quality and can offer stable recording for long time monitoring, but the skin must be prepared before the recording by cleaning the skin with a lightly abrasive material to make sure that there is no dead skin or dirt on the surface. [105] The equivalent electrical circuit schema for the gel electrode-skin interface can be seen in figure 2.4:

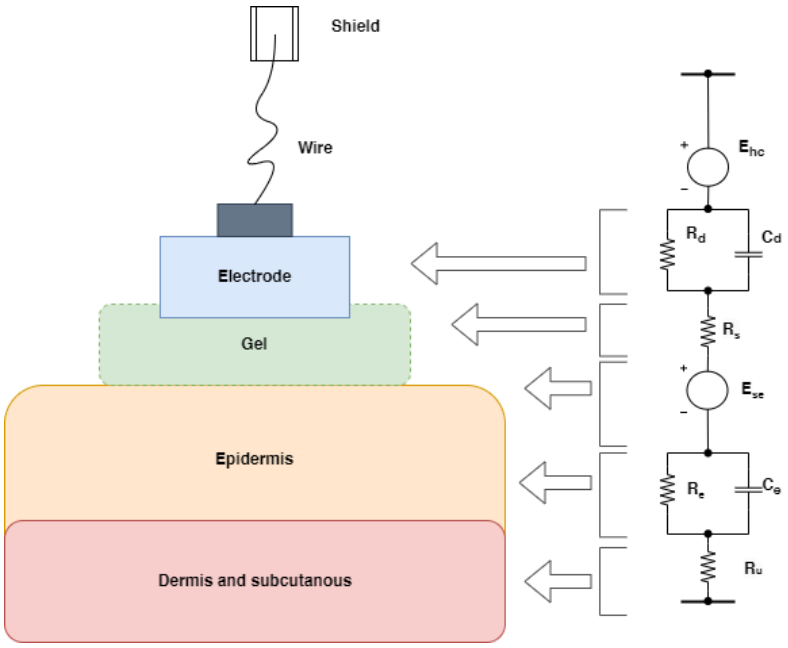


Figure 2.4. Equivalent electrical circuit of the electrode - skin interaction

In our case, the most relevant aspect that can put the "vitals" category in front of the "labs" category is the fact that the electronic sensors can provide the medical staff with instantaneous readings, which can then be uploaded in any database or program through a IoT architecture, while the laboratory works need a lot more time for the entire procedure: drawing blood from the patient, taking the samples to the laboratory, performing the analysis and the uploading the results in a database. Another advantage of the "vitals" category is the fact that in the case of the non-invasive acquisition the patient is not under any additional stress that may result in a slight decline of their well-being, due to physical or psiho-emotional factors (for example, a patient can suffer a panic attack or faint due to seeing blood being drawn or because of needle phobia). The only disadvantage of the surface electrodes is the possibility of the patient developing a mild allergic reaction to the silver compounds of the sensor, which can be manifested through swelling and itchiness. The large number of advantages are some of the reasons why, in this work, we will try to build an additional prediction model only with this category and with any other factors that should be known before the admittance to the ICU ward from the patients' medical records.

The labels "d1_diasbp_invasive_max", "d1_diasbp_invasive_min", "d1_diasbp_max", "d1_diasbp_min", "d1_diasbp_noninvasive_max", "d1_diasbp_noninvasive_min", "h1_diasbp_invasive_max", "h1_diasbp_invasive_min", "h1_diasbp_max", "h1_diasbp_min", "h1_diasbp_noninvasive_max", "h1_diasbp_noninvasive_min" are referring to the features containing the highest and lowest reading for the patients' diastolic blood pressure, measured invasively and/or non-invasively, during both the first hour and during the first 24 hours of their unit stay. On the other hand, the labels "d1_sysbp_invasive_max", "d1_sysbp_invasive_min", "d1_sysbp_max", "d1_sysbp_min", "d1_sysbp_noninvasive_max", "d1_sysbp_noninvasive_min", "h1_sysbp_invasive_max", "h1_sysbp_invasive_min", "h1_sysbp_max", "h1_sysbp_min", "h1_sysbp_noninvasive_max", "h1_sysbp_noninvasive_min" are referring to the features containing the highest and lowest reading for the patients' systolic blood pressure, measured invasively and/or non-invasively, during both the first hour and during the first 24 hours of their unit stay. The systole and diastole are the two main stages of the cardiac cycle and can be seen in figure 2.5:

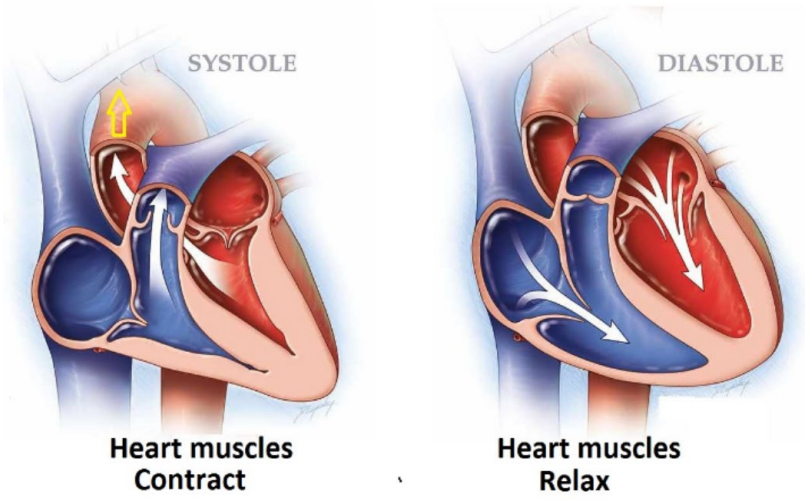


Figure 2.5. Visual anatomy representation of the systole and diastole [53]

The systole is the process characterised by the ventricular contraction of the heart, followed by the pumping of the oxygenated blood in the vascular system, through which it is carried to the organs and tissue of the individual. This stage is started by the electrical activity of the pacemaker cells inside the heart, also known as Purkinje cells. Figure 2.6 showcases the anatomic schema of the cells distribution. These cells are spread out in a tendril-like pattern inside the heart, with a node in the sino-atrial (S-A) part of the heart. From here, the initial electrical impulse travels to the atrio-ventricular (A-V) node, determining the ventricle to contract and push the accumulated blood into the vascular system. During the diastole, the heart muscle relaxes and allow for the next intake of blood inside the heart. First, the closure of the valves allows the atriums to fill up with blood. Then, because the ventricles have been empty, a pressure difference is created during the filling of the atriums, causing the valves between the chambers of the heart to open and allow the blood to fill up the ventricles. Then, because of the newly forms pressure difference, the valves between the heart chambers close and the electrical impulse is propagated through the pacemaker cells, making the ventricles contract and push the blood through the aortic valve. At this point, the systole can be considered done and the aortic valves close in order to stop the pumped blood from returning into the ventricles while the heart relaxes and the diastole takes place. [105]

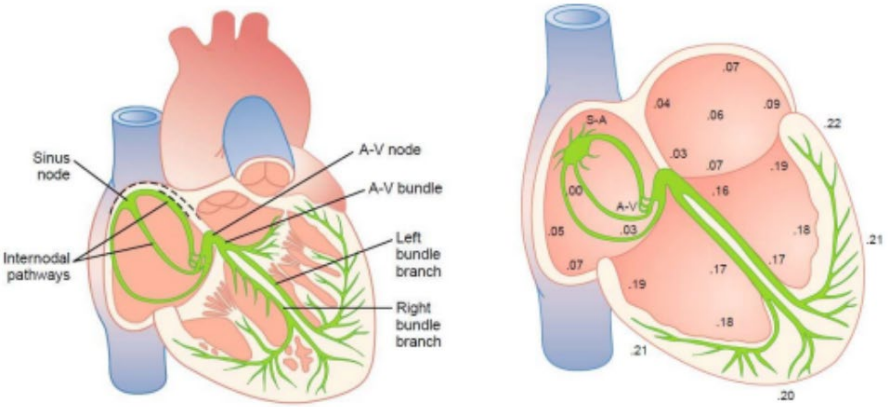


Figure 2.6. The electrical conduction system of the heart - with green the Purkinje cells

Systole means to contract, while diastole means to relax, therefore we can say with certainty that we have different systole and diastole cycles for both the atria and ventricles, as shown in Wigger's diagram for atrial systole in figure 2.7:

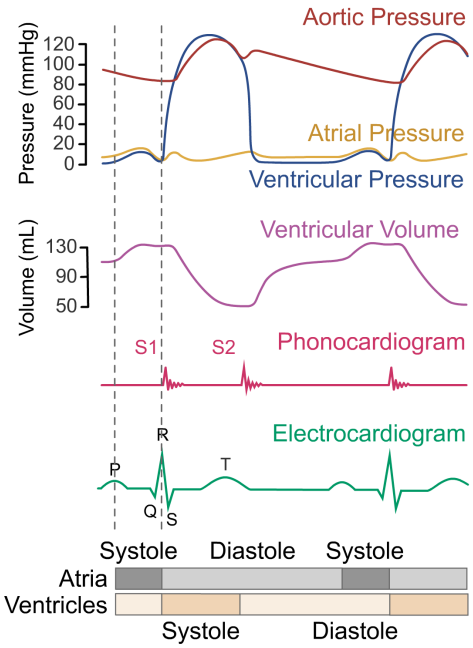


Figure 2.7. Systole and Diastole intervals and their corresponding waveforms [54]

The unit of measure for blood pressure is mm Hg, or millimetres of mercury, because mercury was the material used in the first sphygmomanometers to measure the arterial blood pressure alongside a stethoscope. These devices are still used today in some cabinets, although the more modern versions are preferred, for example, for the manual devices the mercury manometer is usually replaced by an aneroid one while some other practitioners prefer the digital devices. Each variant is depicted in figure 2.9, and has its own advantages and disadvantages but the main difference is the fact that while manual devices are more accurate when used by experts, the digital ones can be more convenient and easier to use even by someone without a medical background. The digital devices rely on oscillometric measurements and computation of values with different algorithms, while abandoning the auditory measurements altogether. The reading is done in the following manner: the cuff of the digital device is placed either around the upper arm or the wrist (it depends on the type of device) and then the start button is pressed. The cuff then inflates and the vibrations of the arterial wall are registered through the sensor, which can be either a piezoresistance, differential capacitor or a deformable membrane. At this point, systolic pressure and pulse rate are registered, and while the cuff deflates and the blood flow is no longer restricted the diastolic pressure is computed by the device. [105] The whole measurement process is controlled by a micro controller as seen in figure 2.8:

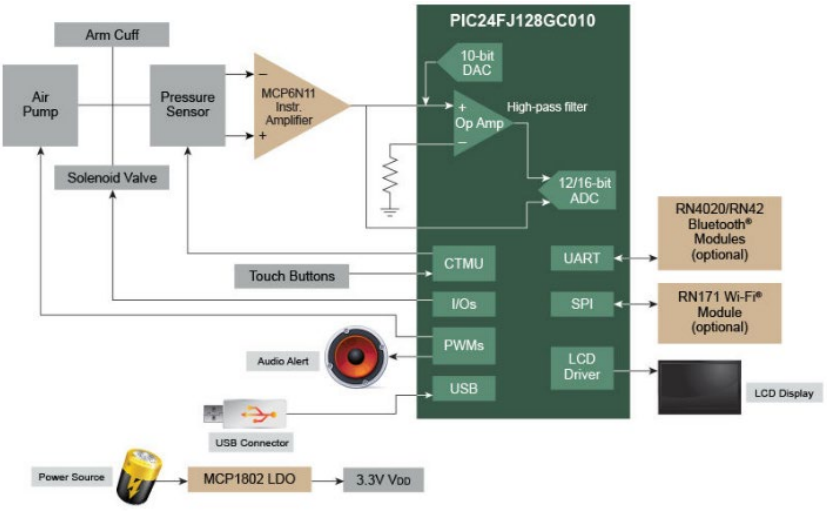


Figure 2.8. Functional diagram of a typical blood pressure monitor [55]

Figure 2.9. Various types of sphygmomanometers (Left - manual mercury gauge, Middle -manual aneroid gauge, Right - digital)

Normal ranges for bp readings, as recommended by the American Heart Association are less than 120 mm Hg for systolic pressure (the upper number on the LCD screen) and less than 80 mm Hg for diastolic pressure (the lower number on the LCD screen). The blood pressure can be a good indicator of various afflictions of the patient. For example, a lower-than-normal blood pressure is called hypotension and it can be traced back to: blood loss, light physical exertion for long periods of time and others. Higher than normal readings can indicate various stages of hypertension (which can be a big contributor to increased cardiovascular disease risk), a hypertensive crisis, a high risk of ischemic heart disease or stroke. In our data set, the bp entries are of numeric type and the unit of measure is specified as millimetres of mercury, therefore, we can expect numbers in the previously described range.

The labels "d1_mbp_invasive_max", "d1_mbp_invasive_min", "d1_mbp_max", "d1_mbp_min", "d1_mbp_noninvasive_max", "d1_mbp_noninvasive_min", "h1_mbp_invasive_max", "h1_mbp_invasive_min", "h1_mbp_max", "h1_mbp_min", "h1_mbp_noninvasive_max", "h1_mbp_noninvasive_min" are referring to the features containing the highest and lowest reading for the patients' mean blood pressure, measured invasively and/or non-invasively, during both the first hour and during the first 24 hours of their unit stay. The mean blood pressure is usually calculated with the standard formula:

$$MBP = DBP + \frac{1}{3}(SBP - DBP) or\ DBP + \frac{1}{3}PP \tag{2.1}$$

Where MBP is the mean blood pressure, DBP is the diastolic blood pressure, SBP is the systolic blood pressure value and PP is the pulse pressure. These values are chosen in such a way because the entire time of the cardiac cycle is proportionally divided between the systolic and diastolic periods for about 1/3 and 2/3 respectively. [56], [57] The associated risks, methods of measurement and unit of measure are the same as for the systolic and diastolic blood pressure. The normal range of values for the mbp should be between 70- and 100-mm Hg. In our data set, the mbp entries are of numeric type and the unit of measure is specified as millimetres of mercury, therefore, we can expect numbers in the previously described range.

The labels "d1_heartrate _max", "d1_heartrate_min", "h1_heartrate_max", "h1_heartrate _min", are referring to the features containing the highest and lowest readings for the patients' heart rate measured during both the first hour and during the first 24 hours of their unit stay. The heart rate is measured in beats per minute and it represents the number of times the patients' heart beat in one full minute. There are multiple ways to measure the heart beats, but one of the easiest is with an ECG measurement. The 6 seconds method is a good algorithm that works well for both regular or irregular heart rated. For a digital measurement, we simply identify a full QRS complex, then select a time frame of 6 seconds and count the number of QRS complexes we have in that time frame. Then we take that value and multiply it by 10 and we have a good estimation of the heart rate, as seen in figure 2.10:

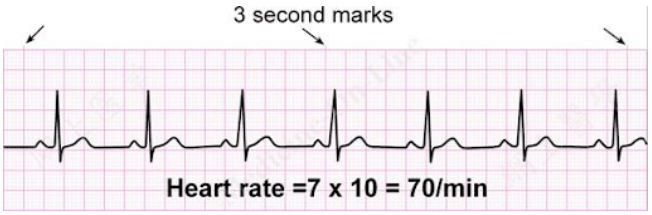


Figure 2.10. Visual representation of the way we can compute the HR [58]

Normal values of the heart rate should be between 60 to 100 beats per minute. A higher value than normal for the heart rate can indicate the presence of tachycardia, anxiety or stress, caffeine consumption, electrolyte imbalance, fever, smoking, usage of drugs (like cocaine), etc. A slower than normal heartrate can be traced back to side effects of medication, electrolyte imbalance, old age or problems with the conduction system of the heart. [58] In our data set, the heart rate entries are of numeric type and the unit of measure is specified as beats per minute, therefore, we can expect numbers in the previously described range.

The labels “d1_resprate_max”, “d1_resperate_min”, “h1_resperate_max”, “h1_resperate_min”, are referring to the features containing the highest and lowest readings for the patients’ respiratory rate measured during both the first hour and during the first 24 hours of their unit stay. This measurement usually does not involve any kind of equipment. The measurement is done simply by counting the number of breaths the patient takes in one minute. Normally, a person at rest should take between 12 and 16 breaths in one minute without any difficulty. During this assessment, the medical personnel is also listening for abnormal breathing patterns and sounds, such as wheezing. The patients’ respiratory rate can increase in conditions of stress (both mental and physical), fever and others. In our data set, the respiratory rate entries are of numeric type and the unit of measure is specified as breaths per minute, therefore, we can expect numbers in the previously described range.

The labels “d1_spo2_max”, “d1_spo2_min”, “h1_spo2_max”, “h1_spo2_min”, are referring to the features containing the highest and lowest readings for the patients’ peripheral oxygen saturation measured during both the first hour and during the first 24 hours of their unit stay. This measurement can be done continuously and non-invasively with a finger pulse oximeter. [59]This device is very small, and usually clips on the index finger of the patient, which can be observed in figure 2.11. The transmitter of the device is sending a beam of infra-red light through the tissue and the receiver is captures the reflected light and sends the information to the microcontroller which calculates the rate of oxygen saturation. Hemoglobin is the compound that absorbs light inside the finger, and oxyhemoglobin is especially absorbent of infra-red light, therefore, the more oxyhemoglobin in the finger, the less reflected light will be captured by the receiver.

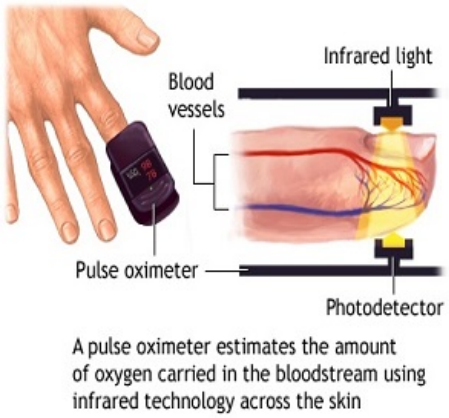


Figure 2.11. Functional schema of a pulse oximeter [59]

Normal levels of oxygen saturation should be above 95%, if measured with a medical grade pulse oximeter. Lower values can indicate various respiratory tract problems, such as respiratory diseases, COVID, smoking, etc. In our data set, the peripheral oxygen saturation entries are of numeric type and the unit of measure is specified as percentage, therefore, we can expect numbers in the previously described range.

The labels “d1_temp_max”, “d1_temp_min”, “h1_temp_max”, “h1_temp_min”, are referring to the features containing the highest and lowest readings for the patients’ temperature measured during both the first hour

and during the first 24 hours of their unit stay. The temperature can be measured through a multitude of ways and devices: digital/stick thermometers (most used for adults), tympanic thermometers (usually used for infants), infrared thermometers (used without direct contact with the skin), mercury thermometers (not used in medical practice anymore). The stick thermometers are the most widely used in medical practice today. They are very accurate and can be used to measure the temperature through various procedures: orally, axillary or rectally. The normal temperature range for an adult can be between 36.1 and 37.2 Degrees Celsius. A higher-than-normal value is regarded as a fever and can be due to multiple reasons: infections in the body, vaccination, malaria, overheating of the body, etc. Lower than normal values are called hypothermia and they are caused by prolonged exposure to factors that are causing the body to lose more heat than it can produce, such as very cold temperatures and harsh environmental factors. In our data set, the temperature entries are of numeric type and the unit of measure is specified as Degrees Celsius, therefore, we can expect numbers in the previously described range.

Demographic category

The Demographic category contains data linked to the grouping of the patients and their particular characteristics. One of the most important labels for our project, "hospital_death" is a binary attribute that indicates whether the patient died during this hospitalization. If the patient died then the attribute will take the value 1 and if they didn't it will be 0. This is the attribute that we will ultimately have to predict based on all the other attributes present in this dataset.

The label "age", is a numeric attribute which represents the age of the patient on unit admission. The measurement unit is of course years and the value range we can expect is between 0 and 100 years.

The label "bmi" contains the values associated to the body mass index of each patient. BMI is the metric currently used to gauge the individuals' level of "fatness" and categorise them into different groups. This measurement is very important because each out of bounds category comes with its own associated risks that the medical staff should be aware of in deciding the best course of action in giving treatment. Combined with age, the BMI can play a very important role in computing the associated mortality risk. BMI values are computed for each individual as the total weight divided by the total height of the patient, and while this type of computation offers us a good enough measurement, it does not tell us any information about the location of fatty tissue in the individual's body. The classification of patients based on BMI is shown in table 2.1 [60]:

Categories of BMI	
Underweight	15-19.9
Normal weight	20-24.9
Overweight	25-29.9
Preobesity	
Class I obesity	30-34.9
Class II obesity	35-39.9
Class III obesity	≥40

Table 2.1. BMI value map

The unit of measure for the body mass index is kg/m^2, and in our dataset the attribute type is given as string, therefore we can expect to be needed to change this in the preprocessing phase.

The label "elective_surgery" is a binary attribute that indicates whether the patient was admitted to the hospital for an elective surgical operation. If the patient had a surgery by personal choice, then the attribute will take the value 1 and if they didn't it will be 0.

The "ethnicity" label is a string attribute that indicates the common national or cultural tradition which the person belongs to. This is relevant because we have data taken from multiple areas of the world and, as stated in the introduction of this work, it has been proven that algorithms trained on data coming from different ethnic

groups have the best results when tested on individuals belonging to the same ethnic categories on which the algorithm was trained.

The label “gender” contains information about the genotypical sex of the patient in a string format. Therefore, we can expect the values for this attribute to be either F, for female, or M, for male.

The label “height” contains the values in centimetres describing the height of the person on unit admission. The label “weight” contains the values in kilograms describing the weight (body mass) of the person on unit admission. Both of these labels are of numeric type.

The label “hospital_admit_source” contains information about the location of the patient prior to being admitted to the hospital. This attribute of is of string type and a possible value can be for example “Home”.

The label “icu_admit_source” contains information about location of the patient prior to being admitted to the unit and although it could be confused with the previously described attribute, the values are different and could be considered, in some cases, even more important. This attribute of is of string type and a possible value can be for example “Operating room”.

The label “icu_id” contains information about the type of unit admission for the patient. This attribute of is of string type and a possible value can be for example “Cardiothoracic”.

The label “icu_stay_type” contains another kind of information about the type of unit admission for the patient. For example, this string type attribute can tell us if the patient in in the ICU ward through admittance, transfer or readmittance.

The label “icu_type” represents a classification which indicates the type of care the unit is capable of providing. This attribute of is of string type and a possible value can be for example “Neurological ICU”.

The label “pre_icu_los_days ” contains informationa about the length of stay of the patient between hospital admission and unit admission. This attribute of is of numeric type and the unit of measure is days.

The label “readmission_status” is a binary attribute that indicates whether the current unit stay is the second (or greater) stay at an ICU within the same hospitalization. If the patient has been readmitted to the ICU ward, then the attribute will take the value 1 and if they weren’t it will be 0.

APACHE covariate category

The APACHE covariate category contains the features that are taken into consideration when computing the APACHE III score. In this dataset, the values that were recorded were the ones that contributed to the highest APACHE III score for each patient. Many of the features can also be found in the “labs” and “vitals” categories, therefore we don’t need to detail them again.

The label “albumin_apache” registers the albumin concentration measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “g/L” and the values are of numeric type.

The label “bilirubin_apache” registers the bilirubin concentration measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “micromol/L” and the values are of numeric type.

The label “bun_apache” registers the blood urea nitrogen concentration measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “mmol/L” and the values are of numeric type.

The label “creatinine_apache” registers the creatinine concentration measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “micromol/L” and the values are of numeric type.

The label “glucose_apache” registers the glucose concentration measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “mmol/L” and the values are of numeric type.

The label “heart_rate_apache” registers the heart-rate measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “beats per minute” and the values are of numeric type.

The label “hematocrit_apache” registers the hematocrit measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “fraction” and the values are of numeric type.

The label “resprate_apache” registers the respiratory rate measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “breaths per minute” and the values are of numeric type.

The label “sodium_apache” registers the sodium concentration measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “mmol/L” and the values are of numeric type.

The label “temp_apache” registers the temperature measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “Degrees Celsius” and the values are of numeric type.

The label “wbc_apache” registers the white blood cell count measured during the first 24 hours which results in the highest APACHE III score. As in the previous treated case, the unit of measure is “10^9/L” and the values are of numeric type.

The label “apache_2_diagnosis” is referring to the APACHE II diagnosis for the ICU admission. APACHE II is the predecessor of APACHE III and is a scoring system that attributes points based on the initial values of 12 routine physiologic measurements, age and previous health status in order to provide a general measure of severity of disease. An increased score in the range 0 and 71 was associated to a higher risk of hospital death for many usual afflictions.[61] In our data set, the apache 2 diagnosis entries are of string type with no unit of measure, therefore, we can expect that there will be a need to convert this data into a numeric type or discard it altogether in the pre-processing phase of our project.

The label “apache_3j_diagnosis” is referring to the APACHE III sub-diagnosis code which best describes the reason for the ICU admission. APACHE III is a prognostic system which is used in the first 24 hours of ICU admission. Released in 1991, the APACHE III prognostic system attributes a score between 0 and 299 points to a patient based on a few carefully selected attributes such as: age, sex, reason for admission to ICU, comorbidities and other influencing factors registered during ICU admission. A normal APACHE III score should be between 0 and 71 points, but it is quite common to have less than 55 points as a patient. In our data set, the apache 3j diagnosis entries are of string type with no unit of measure, therefore, we can expect that there will be a need to convert this data into a numeric type or discard it altogether in the pre-processing phase of our project.

The label “apache_post_operative” is a binary attribute which is correlated with the APACHE operative status: 1 for post-operative and 0 for non-operative. This means that if a patient has the value 1 for this attribute, then the ICU admission was done after surgery was performed.

The label “arf_apache” is also binary attribute which indicates whether the patient had acute renal failure during the first 24 hours of their unit stay, defined as a 24-hour urine output. Also like in the previous case, if the value for this attribute is 1 then the patient had acute renal failure and if it is 0 then then they didn’t.

The label “fio2_apache” registers the fraction of inspired oxygen from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for oxygenation. This attribute is of numeric type and the unit of measure is stated as fraction, therefore we can expect to see a decimal value.

The labels “gcs_eyes_apache”,” gcs_motor_apache”,” gcs_unable_apache” and “gcs_verbal_apache” are indicators of various Glasgow Coma Scale components that were assessed during the first 24 hours of the patients ICU stay. The Glasgow Coma Scale is a vital component of numerous trauma scoring systems and it

can be defined as a neurological scale that assesses the conscious state of a person. The numerical range of this scoring system is from 3 to 15 and, generally, comas are classified as: severe, with GCS ≤8, moderate, GCS 9–12, and minor, GCS ≥13. The GSC is computed as the score associated for the eye response plus the score for the verbal response plus the score for the motor response. Table 2.2 illustrates the way the scores are attributed for each of the factors taken into consideration[62]–[67]:

Eye Response (E)	**Verbal Response (V)**	**Motor Response (M)**	**Duration of Amnesia (A)**	**SCORE**
No eye opening	No verbal response	No motor response	31–90 days	1
Eye opening in response to pain	Eyes opening spontaneously	Extension (decerebrate response) on painful stimuli	8–30 days	2
Eye opening to speech	Inappropriate words	Abnormal flexion (decorticate response) on painful stimuli	1–7 days	3
Eyes opening spontaneously	Confused	Withdrawal from pain	3–24 h	4
–	Oriented	Localizes to pain	30 min to 3 h	5
–	–	Obeys commands	30 min	6
–	–	–	None	7
Glasgow Coma Score (E+V+M)				Out of 15
Glasgow Coma-Extended score (E+V+M+A)				Out of 22

Table 2.2. GSC value map

Therefore, the meanings of the labels are as follows: “gcs_eyes_apache” - the eye opening component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score, ”gcs_motor_apache” - the motor component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score, “gcs_verbal_apache” - the verbal component of the Glasgow Coma Scale measured during the first 24 hours which results in the highest APACHE III score and ”gcs_unable_apache” - whether the Glasgow Coma Scale was unable to be assessed due to patient sedation. All labels besides” gcs_unable_apache” are of integer type, with no unit of measure and with values matching the ones stated in the table above.” gcs_unable_apache” is a binary attribute for which if the value is 1 then the Glasgow Coma Score couldn’t be assessed and if it is 0 then then it could.

The label “intubated_apache” is another binary attribute which indicates whether the patient was intubated at the time of the highest scoring arterial blood gas used in the oxygenation score. Also like in the previous cases, if the value for this attribute is 1 then the patient was intubated and if it is 0 then then they weren’t.

The label “map_apache” holds the value of the mean arterial pressure measured during the first 24 hours which results in the highest APACHE III score. This value is often calculated in the same way as mean blood pressure (mbp):

$$MBP = DBP + \frac{1}{3}(SBP - DBP) or\ DBP + \frac{1}{3}PP \quad 2.2$$

The data type of this attribute is numeric and the unit of measure is stated as millimetres of mercury. Given the above formula, we can expect the same value range as for mbp.

The labels “paco2_apache” and “paco2_for_ph_apache” refer to the partial pressure of carbon dioxide from the arterial blood gas taken during the first 24 hours of unit admission. “paco2_apache” holds the value that produces the highest APACHE III score for oxygenation and “paco2_for_ph_apache” holds the value which produces the highest APACHE III score for acid-base disturbance. PCO_2 is the measurement of the amount of

carbon dioxide found in arterial or venous blood. Its main purpose is to serve as an indicator of the goodness of the alveolar ventilation within the lungs. Normally, the values of PCO_2 range between 35 to 45 mmHg.[68] In our data set, the data type of this attribute is numeric and the unit of measure is stated as millimetres of mercury, therefore we can expect values inside the previously described range.

The label "pao2_apache" holds the value of the partial pressure of oxygen from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for oxygenation. [69] PAO_2 represents the value of the partial pressure of oxygen in the alveoli, which is a direct indicator of the goodness of the alveolar ventilation within the lungs. The normal range of this measurement is 75 to 100 millimetres of mercury, but the measured value can be affected by age or altitude [70]In our data set, the data type of this attribute is numeric and the unit of measure is stated as millimetres of mercury, therefore we can expect values inside the previously described range.

The label "ph_apache" holds the value the pH from the arterial blood gas taken during the first 24 hours of unit admission which produces the highest APACHE III score for acid-base disturbance.[70]The pH factor measures the hydrogen ion activity and is done as a usual part of any blood gas measurement. The normal value for this attribute ranges between 7.35 to 7.45. In our data set, the data type of this attribute is numeric and we have no unit of measure, so we should find values inside the previously described range.

The label "urineoutput_apache" is an attribute which the total urine output of the patient during the first 24 hours. A normal urine output should be anything higher than 500 ml/day. Smaller values can be a good indicator of acute kidney injury or oliguria.[71] In our data set, the data type of this attribute is numeric and the unit of measure is stated as Millilitres, therefore we should find values like the one described previously.

The label "ventilated_apache" is another binary whether the patient was invasively ventilated at the time of the highest scoring arterial blood gas using the oxygenation scoring algorithm, including any mode of positive pressure ventilation delivered through a circuit attached to an endo-tracheal tube or tracheostomy. Also like in the previous case, if the value for this attribute is 1 then the patient had was ventilated and if it is 0 then then they weren't.

Labs blood gas category

"d1_arterial_pco2_max", "d1_arterial_pco2_min", "h1_arterial_pco2_max", "h1_arterial_pco2_min" are the labels referring to the features containing the highest and lowest arterial partial pressure of carbon dioxide for the patient during both the first hour and during the first 24 hours of their unit stay. Arterial blood gases, or plainly blood gases, is a term used for three measurements (pH, PCO_2 and PAO_2) that are being done in order to evaluate if the acid-base status, ventilation and arterial oxygenation are within normal ranges.[70] As stated in the previous category, the normal values for PCO_2 range between 35 to 45 mmHg. In our data set, the data type of this attribute is numeric and the unit of measure is stated as millimetres of mercury, therefore we can expect values inside the previously described range.

"d1_arterial_ph_max", "d1_arterial_ph_min", "h1_arterial_ph_max", "h1_arterial_ph_min" are the labels referring to the features containing the highest and lowest arterial pH for the patient during both the first hour and during the first 24 hours of their unit stay. As stated previously, [70]the pH factor measures the hydrogen ion activity and is done as a usual part of any blood gas measurement. The normal value for this attribute ranges between 7.35 to 7.45. In our data set, the data type of this attribute is numeric and we have no unit of measure, so we should find values inside the previously described range.

"d1_arterial_po2_max", "d1_arterial_po2_min", "h1_arterial_po2_max", "h1_arterial_po2_min" are the labels referring to the features containing the highest and lowest arterial partial pressure of oxygen for the patient during both the first hour and during the first 24 hours of their unit stay. As stated previously, [70]the pH factor measures the hydrogen ion activity and is done as a usual part of any blood gas measurement. The normal value for this attribute ranges between 7.35 to 7.45. In our data set, the data type of this attribute is numeric and we have no unit of measure, so we should find values inside the previously described range.[19] PAO2 represents the value of the partial pressure of oxygen in the alveoli, which is a direct indicator of the goodness of the alveolar ventilation within the lungs. The normal range of this measurement is 75 to 100 millimetres of mercury, but the measured value can be affected by age or altitude [20]. In our data set, the data

type of this attribute is numeric and the unit of measure is stated as millimetres of mercury, therefore we can expect values inside the previously described range.

“d1_pao2fio2ratio_max”, “d1_pao2fio2ratio_min”, “h1_pao2fio2ratio_max”, “h1_pao2fio2ratio_max” are the labels referring to the features containing the highest and lowest fraction of inspired oxygen for the patient during both the first hour and during the first 24 hours of their unit stay. This attribute represents an estimation of the oxygen content that reaches the alveoli during one inhale of the subject. This estimation is detrimental in diagnosing hypoxemia, which has been shown to have a great contribution to the increase of all-cause mortality. The formula to calculate FiO_2 has two versions as follows: 0.21 + (1/(4 x minute ventilation)) x L/min of O2 when the inspiratory time to total inspiratory-expiratory time ratio is 0.33 and 0.21 + (1/(2.5 x minute ventilation)) x L/min of O2 when the inspiratory time to total inspiratory expiratory time ratio changes to 0.5. In our data set, the data type of this attribute is numeric and the unit of measure is a fraction, so the values we can expect should be decimal based.

APACHE comorbidity category

All the APACHE comorbidity labels are binary attributes representing the state of positivity of any one of the following afflictions that can result in a heightened risk of mortality. Therefore, if the value of this attribute is 1 then the patient has that comorbidity and if it is 0 then the patient doesn’t exhibit that disease.

The labels are “aids”,” cirrhosis”,” diabetes_mellitus”,” hepatic_failure”, “chemotherapy”,” leukimia”,” lyphoma”, and” solid_tumor_with_metastasis”. The label “aids” checks whether the patient has a definitive diagnosis of acquired immune deficiency syndrome (AIDS) (not HIV positive alone). The HIV-1 pandemic can be considered one of the most defining epidemics that took place in modern age. Since the dawn of this epidemic, the research has produced very good results, even if a vaccine or a cure is yet to be discovered. Antiretroviral treatments have turned AIDS from a fatal condition into a manageable chronic disease, even ensuring that with enough treatment, positive individuals would not spread the virus. The main way of transmission of HIV is unprotected sexual contact, but a multitude of cases also showed the possibility of transmission within the communities of injection drug users. The woman population suffers an even greater burden because of this virus compared to their male counterparts, mainly because of the fact that the virus is also transmissible through the mother-infant bond. The child receives nutrients through the umbilical cord from the mother, but the bond that helps the fetus to grow also becomes an inevitable path towards infection. HIV-1 attacks the patients’ immunity on all fronts, destroying certain components of the immune system and more specifically takes over CD4 cells (white blood cells) and make copies of itself. AIDS is the late stage of HIV which usually occurs after about a decade of untreated infection with HIV. At this point, the immune system of the body becomes so damaged that it cannot protect the individual even from very light illnesses, such as the common cold. It is very important that people with high risk of developing HIV test themselves regularly as to be able to detect the disease as early as possible and to start the administration of treatment immediately in order to minimise the risk of putting themselves or other in danger. [72]

The label “cirrhosis” checks whether the patient has a history of heavy alcohol use with portal hypertension and varices, other causes of cirrhosis with evidence of portal hypertension and varices, or biopsy proven cirrhosis. This comorbidity does not apply to patients with a functioning liver transplant. Cirrhosis can be defined as the histological development of regenerative nodules surrounded by fibrous bands, to be further known as fibrosis, resulted from chronic liver injury which can be extended to end stage liver disease. Currently, the only known method to cure this disease is liver transplantation, but therapy with pharmaceutical compounds can halt the progress of the decaying process or even halt it if it is begun in the incipient stages of development. This process compromises the exchange between the liver sinusoids and hepatocytes due to the distortion of hepatic vasculature. The main consequences of cirrhosis are characterised by impaired hepatic functions, development of hepatic cancer cells and increased hepatic resistance. The main causes for this disease are different when we analyse the cases demographically: alcoholic liver disease and hepatitis C are more common in the Western part of the world, while hepatitis B is more common in Asia and Africa. It is very important to know what the cause for cirrhosis because it can be used to predict possible complications or direct treatment plans. [73]

The label" diabetes_mellitus" checks whether the patient has been diagnosed with diabetes, either juvenile or adult onset, which requires medication. Diabetes mellitus is a general term used to encompass multiple types of diabetes: type 1, type 2, gestational diabetes. This category of diseases is characterised by hyperglycemia that results from defects in insulin secretion from the pancreas, insulin action, or both. The symptoms of diabetes are closely linked to the type of diabetes and the length of time for which the patient has been sick. Most often the patients with type 2 diabetes are asymptomatic at least for the first few years, while type 1 patients can show clear signs of hyperglycemia, polyuria, polyphagia, weight loss, and blurred vision. Untreated diabetes can lead to stupor, coma or even death. Type 1 and type 2 diabetes are the same disorder of insulin resistance that are correlated to different genetic backgrounds. Another classification of diabetes mellitus that can require medication is: autoimmune type 1 diabetes, idiopathic (unknown origin) type 1 diabetes, fulminant type 1 diabetes, type 2 diabetes mellitus, insulin resistance, monogenic diabetes. [74]

The label" hepatic_failure" checks whether the patient has cirrhosis and additional complications including jaundice and ascites, upper GI bleeding, hepatic encephalopathy, or coma. Acute liver failure is considered to be a complex syndrome characterized by coagulopathy, elevated liver biochemistry, and hepatic encephalopathy without underlying chronic liver disease. The origin of this affliction is most often times drug-induced liver injury or viral hepatitis. Symptoms can include: jaundice, pain in the upper right abdomen, swelling, nausea, vomiting and dehydration. Usually, patients succumb to this affliction without a transplant and prior to it admittance to the ICU ward is considered mandatory. This disease can be categorised into four distinct groups in regards to the time it takes for the ALF to take its toll on the patient: within 7 days – hyperacute liver failure, 1 to 4 weeks – acute, subacute 5 to 12 weeks – subacute. [75]

The label "chemotherapy", checks whether the patient has their immune system suppressed within six months prior to ICU admission for any of the following radiation therapy, chemotherapy, use of non-cytotoxic immunosuppressive drugs, high dose steroids (at least 0.3 mg/kg/day of methylprednisolone or equivalent for at least 6 months). Chemotherapy is a way of treatment that has been used to turn the terminal prospects of cancers into merely curable illnesses that can be handled with the right treatment schema. The main purpose of chemotherapy is to inhibit carcinoma cell production and production, thus halting the overall progression of the cancer. However, this type of treatment has an unwanted effect on the healthy cells as well, weakening them and taking a significant toll on the immune system. The treatment of cancer is traced back to the alkylating agents, the most well-known of them being mustard gas derivatives, which can be administered under the form of pills or injections. All cancer treatments have a main characteristic in common, being that they affect the rapidly replicating cells more than other types of cells in the human body. This means that cancer drugs affect and ultimately destroy cancer cells more than others, but it also means that it has the same effect on other types of cells inside the human body, such as bone marrow cells, hair follicle cells, and the cells that comprise the intestinal tract. Due to this, the most common side effects of chemotherapy are: hair loss, infertility, fatigue, nausea and vomiting. Most commonly, various drugs are prescribed in order to manage these side effects, and they can have different rates of success from one individual to another. Chemotherapy is a very important schema of treatment to be started as early as possible because it can kill remaining carcinoma cells after tumour extraction or just halt the development of new ones with high rates of success. If the treatment is begun in the final cancer stage (tumour with metastasis), then the success and subsequently survival rates are quite low. [76]

The label" leukimia", checks whether the patient has been diagnosed with acute or chronic myelogenous leukemia, acute or chronic lymphocytic leukemia, or multiple myeloma. Leukemia is a type of aggressive cancer that affects the development of leukocytes. It can be divided into either acute or chronic leukemia based on the rapidity of replication of blood cancer cells. The treatment for this type of cancer can vary, but usually most doctors resort to prescribing chemotherapy in order to fight this disease. Most times, as is the case for most kind of cancers, leukemia is a result of genetic factors that make the patient more susceptible to developing this disease during the course of their lifetime. Acute leukemia affects more than 20% of the immature bone marrow cells (known as blasts), while chronic leukemia affects less than 20% of blasts. Leukemia can be further divided into four subtypes: acute lymphoblastic leukimia, acute myelogeneous leukemia, chronic lymphocytic leukemia and chronic myelogeneous leukemia. Factors that can trigger leukemia are: exposure to radiation, benezene or chemotherapy, history of hematologic malignancy, viral infections or various genetic syndromes. Common symptoms associated with leukemia are low immune response, low oxygenation rate of cells and low coagulation rates. Leukemia chemotherapy can be very aggressive and thus can lead to the death of essential bone marrow cells, but this problem can be solved with bone marrow transplants from a suitable donor. [77]

The label" lyphoma", checks whether the patient has been diagnosed with non-Hodgkin lymphoma. Non-Hodgkin lymphomas are a set of malignant neoplasms derived from lymphoid tissues, mainly the lymph nodes. These types of tumours can appear due to infections, toxins, chronic inflammation and chromosomal translocations. Non-Hodgkin lymphomas are divided into multiple categories: Burkitt lymphoma, endemic (African) lymphoma, non-endemic (sporadic), mantle cell lymphoma, gastrointestinal lymphoma, and primary central nervous system lymphoma. Common symptoms are: lymphadenopathy, fever, weight loss, bone marrow or spinal cord issues. The stages of Non-Hodgkin lymphomas are detrimental for choosing the treatment plan. There are four stages of NHLs: stage 1 - single lymph node, stage 2 - two or more involved lymph node regions on the same side of the diaphragm, stage 3 - lymph node involvement on both sides of the diaphragm, and stage 4 - widespread involvement of one or more extra lymphatic organs with or without associated lymph node involvement. Treatment usually consists of chemotherapy or radiation therapy. [78]

The label" solid_tumor_with_metastasis" checks whether the patient has been diagnosed with any solid tumour carcinoma (including malignant melanoma) which has evidence of metastasis. Metastasis is the fourth and last stage of cancer development which is the lead cause of cancer death known today. This stage is characterised by the migration of cancer cells from the initial tumour through the body and their infection of multiple tissues in their path. This stage of cancer has very low survival rates, even with state-of-the-art treatments, as it can very easily spread through the body and affect various essential systems. [79]

Identifier category

The data in this category is of integer type as it represents a unique code for each entity involved into the data acquisition process. The three labels and their meanings are: "encounter_id" - unique identifier associated with a patient unit stay, "patient_id" - unique identifier associated with a hospital, and "hospital_id" - unique identifier associated with a patient.

APACHE prediction category

The labels "apache_4a_hospital_death_prob" and "apache_4a_icu_death" represent the APACHE IVa probabilistic predictions of patient mortality both in regards to the in-hospital death and ICU mortality. Both of these attributes use the APACHE III score and other covariates, including diagnosis. In our dataset, these attributes are of numeric type and in most cases they are shown in decimal format, correlating them to the death probability.

APACHE grouping category

The labels "apache_3j_bodysystem" and "apache_2_bodysystem represent the APACHE II and III admission diagnosis groups for each of the patients. These groups mainly represent the bodily systems that have been affected by some disease, for example "Respiratory" or "Cardiovascular". In our data set these attributes are of string type, therefore we can expect the need to make some conversions for these values in the pre-processing phase.

GOSSIS example prediction category

The label "GOSSIS example prediction" holds the value of an example mortality prediction, shared as a 'baseline' based on one of the GOSSIS (Global Open-Source Severity of Illness Score) algorithm development models. The GOSSIS model is a novel in-hospital mortality prediction algorithm initiated by MIT that has been developed with the same dataset that this project uses.[80], [81] In our data set this attribute is of numeric type, and the values are in a decimal format, correlating them to the death probability.

3. Implemented Method Description

3.1.Steps for method development

The hereby project has the purpose to develop an algorithm for that will successfully estimate whether a patient admitted to an intensive care unit (ICU) will be more likely to survive or decease based on the parameters registered in the first 24 hours from the initial moment of admittance. This issue is ultimately reduced to a binary classification that will output one out of two possible results: "0", for a patient that is more likely to survive, or "1", for a patient that is more likely to decease. Based on this result the medical staff will be able to rush the decision-making process and ensure that the patient that is at a higher risk of death receives the best treatment possible in order to better their odds of survival.

The implemented method is split into two main parts: the data engineering part and the model development part. The development process is illustrated in figure 3.1:

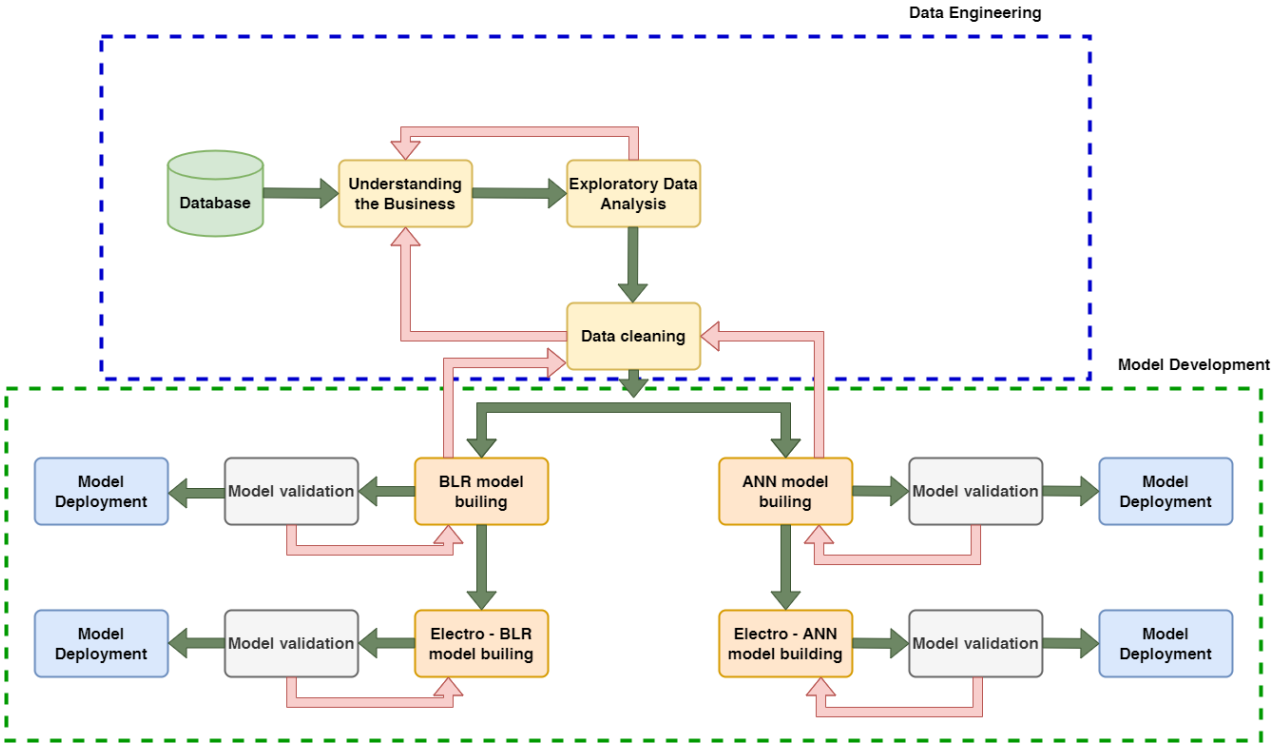


Figure 3.1. Development process

After acquiring the WIDS 2020 Datathon database that was deemed satisfactory for the development of the solution of our approached issue, both size and diversity-wise, we delved into the first step of our implementation, namely **Understanding the Business**. Since our background was very technical based, this step was detrimental in ensuring that our solution was fit for the environment in which we wished to deploy the final product, namely the medical field. This step consisted in extensive research of the State of the Art, the medical environment and terminology and interviews sustained with both medical students and doctors that were essential in gaining a true perspective of the hands-on needs that exist in the Romanian and international ICU environments.

The next step was the **Exploratory Data Analysis**, which consisted in the true understanding of the database that we had, more specifically the training dataset. This step was implemented by making a first pass through the massive dataset, building statistics and intuitive graphs for all the attributes. The statistics were built through processes such as counting of all the values by categories, counting all of the categorical and numerical values, finding unique values for each attribute, identifying the *"nan"* values (values that are undefined), finding the shape of the dataset (how many columns and how many lines), computing the mean, standard

deviation, minimum, maximum and the interquartile ranges (IQR) for each of the attributes. The IQR values are computed in the following manner: firstly, we split the total values in half. The first quartile (denoted as **25%**) is computed as the median of the first half, while the second quartile (denoted as **75%**) is the median of the second half of the values. The middle quartile (denoted as **50%**) is computed as the difference of the first and second quartile. This measurement is detrimental in gaining a first view of how the data varies in the dataset or if there are any outliers.

The graphical representation developed in this step served as a good measure at taking a look at the variability of the dataset, whether the values of the attributes had any outliers, the variance of the data and whether the dataset is balanced or unbalanced. The metrics used for this were: barplots, boxplots, kernel density estimate (kde) plots, scatter plots and heatmaps.

A heatmap is a color-encoded matrix that is plotted in order to view the correlation of two or more attributes. A high intensity color or a very warm color will usually be matched with a high correlation, while a low intensity color or a colder color will be specific for decorrelated values. A good example of a heatmap is illustrated in figure 3.2. [82]

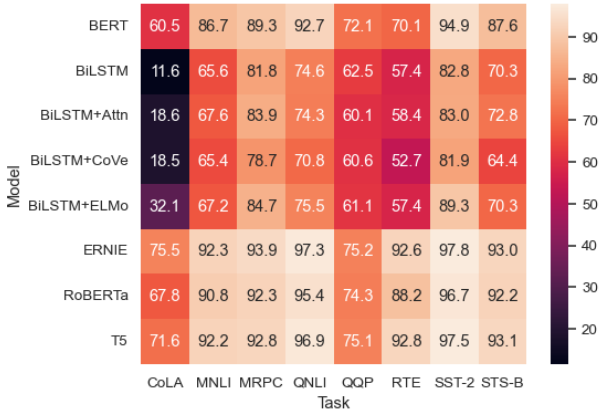


Figure 3.2. Example of a heatmap [86]

The next step was **Data cleaning**. This step involved multiple pre-processing operations that were chosen based on the results of the previous step. During this step we dropped the attributes that we considered of little relevance to the end result or that were included in other attributes (for example in the case of bmi, the values are computed using the values of the height and weight attributes which are also present in the dataset), we dealt with missing values by dropping the attributes (if they had many missing values), entries (if there were few entries with missing values) or by copying the values from another attribute (for example if we had the values for the invasive sensors, but not for the non-invasive). Here, we also transformed string features to categorical features using label encoder and replaced them in the original dataset.

The label encoding process is done by indexing all the unique string entries and attributing them a code starting from zero to n-1, n representing the total number of unique attributes. An example for this would be: presume that we have three types of roses that are being sold at a flower shop, blue, red and white. The encoding in a dataset that contains information about the flower transactions would be done by attributing the code 0 for blue, 1 for red and 2 for white, thus replacing in the set all of the string entries with the attributed codes.

This step had to be done with extreme care because in our dataset we deal with very sensitive data, thus we weren´t able to somehow replace most of the values by using statistically computed ones, which led to a large decrease of the training data while preserving the imbalance of the dead versus alive data. To deal with this problem we made a final step in the pre-processing part of the data cleaning, namely, balancing the data.

In order to do this but still maintain the validity of the data we used a combination of artificially over- and under-sampling the data.[87]–[89] The Synthetic Minority Over-sampling Technique (SMOTE) aims at creating synthetic entries of the minority class rather than performing over-sampling by replacement, which can create overfitting and bias due to the fact that we clone the same data. The creation of synthetic examples is done in feature-space rather than data-space by implementing an algorithm that introduces the synthetic entries along the line segments that join any/all of k minority class nearest neighbours. The under-sampling is

done by employing the Edited Nearest Neighbours method which cleans the data by removing any samples of the majority class that are too close to the decision boundary. The decision boundary is the line or region between two classes in which the space becomes ambiguous, meaning that samples found on that line may be of either class A or class B but we are not able to say which one with confidence. The boundary line is depicted in figure 3.3:

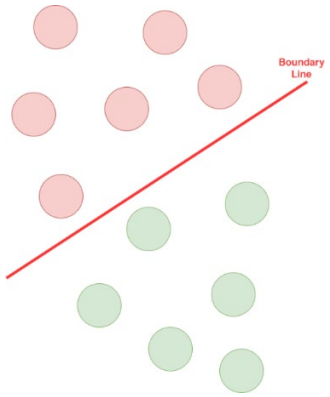


Figure 3.3. Example of a boundary line

The combination of these two techniques is called SMOTEEN and it is the strategy that we employed in order to balance the data that we had without corrupting the dataset and invalidating our results.

The model development process has been done based on the approaches studied in the State-of-the-Art chapter. As seen previously, some of the best models that can be implemented for a requirement that is reduced to a binary classification would be a Binary Logistic Regression machine learning model and Multilayer Perceptron deep learning model.

The Binary Logistic Regression (BLR) studies the association of a binary response (in our case either "0" for alive or "1" for dead) in regards to a set of input parameters (in our case, the attribute values gathered in the first 24 hours from the initial admittance to the ICU ward). It is one of the most commonly used models for biomedical applications since most problems can be reduced to a dichotomous Yes or No/ True or False result. For a binary logistic function, the outcome probability is modelled on various individual attributes. In order to ensure the boundaries of the model, the logarithm of the OR is applied to the ratio corresponding to the probability, as seen in equation 3.1 [90]:

$$logit(\pi) = \ln\left(\frac{\pi}{1-\pi}\right) \tag{3.1}$$

Π represents the probability of an event, which can also be described as the mean of the binary variable. The equation for the probability is as follows:

$$\pi = \Pr(Y=1|X=x) \tag{3.2}$$

where Y is the binary response where "1" represents the situation in which the event occurs and "0" is the situation in which the event didn't take place, and X is representative for the set $(X_1,X_2,\ldots,X_n)$ of predictors for the event that can either discrete of continuous. The BLR model can relate the logarithmic odds represented as $[ln(\pi/1- \pi)]$ in a linear way to the variation in the predictor variables, such that the boundaries can be expressed as:

$$0 \leq \pi \leq 1 \tag{3.3}$$

$$-\infty \leq \text{logit}(\pi) \leq +\infty \tag{3.4}$$

Therefore, for event i, we have the following expressions:

$$\pi_i = \Pr\left(Y_i = 1|X_i = x_i\right) = \frac{\exp(\beta_0+\beta_1 x_i)}{1+\exp(\beta_0+\beta_1 x_i)} \tag{3.5}$$

$$\text{logit}(\pi_i) = \ln(\pi_i/(1-\pi_i)) = \ln[\exp(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik})] \quad 3.6$$

And if the logit is defined as linear, we have:

$$\text{logit}(\pi_i) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik} \quad 3.7$$

However popular, the BLR model has been proven through previous studies to perform worse than a Deep Learning model as it faces serious limitations such as: limitations when working with features that have non-linear relationships with the outcome, limited complexity and bad performance on large datasets with complex relations.

In order to work around the obvious limitations of the BLR model we used regularization techniques in order to artificially simplify the complexity of the data and help the model avoid overfitting and perform better on new data. The chosen method was Elastic net, which is a combination between L1(Lasso) and L2 (Ridge) regularization. [91] Both of these methods add a penalty term to the loss function. L1 adds the sum of the absolute values of the weights of the parameters as a penalty term, thus encouraging some of the weights to be 0 which leads to less parameters to be taken into consideration in the final model. L2 on the other hand adds the sum of the squared values of the weights to the loss, thus making sure that the model will pay more attention to more important attributes.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \quad 3.8$$

Loss function with L1

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \quad 3.9$$

Loss function with L2

Machine learning models characterised by algorithms that learn a set of rules and then apply them on new cases. This greatly limits their performance as situations might change and in real-life scenarios things might be quite different than what the model learned. In order to overcome this limitation, researchers have developed Deep Learning models and algorithms that have gained tremendous popularity in the recent years due to their drastically improved performances and ease of implementation. Deep Learning models are characterised by their specific structure composed of elements called perceptrons or neurons, that are structured into layers and react much like human neurons, meaning that they create links between them and enable the algorithm to keep actively learning, surpassing the stage of simply memorising rules and applying them, and reaching a somewhat semblance of real intelligence by gaining the ability to adapt and make informed decisions in the case of new data.

In machine learning, the goal is to generate a classifier that takes the measurements of a pattern as input, and provides its correct classification as output. A popular type of classifier [92] is feed-forward NNs, which consists of an input layer of neurons, an arbitrary number of hidden layers, and an output layer. For the purpose of classifying patterns, feed-forward NNs use as many input neurons as there are measurements for each pattern in the data set, i.e., there is precisely one input neuron per measurement. As many neurons make up the output layer as there are classes in the data collection. Given the weights of all the connections between the neurons, one may categorize a pattern by feeding its measurements as input to the input neurons, propagating the output signals from layer to layer, and obtaining the output signals of the output neurons. Each output neuron is assigned to a certain class. A pattern is classified by the output neuron that generates the greatest output signal.

A multilayer Perceptron is a modification of Rosenblatt's original Perceptron model from 1950. The neurons are arranged in layers, connections are always made from lower levels to upper layers, and neurons within the same layer are not linked. It may also have one or more hidden layers between its input and output layers. Our

main goal is to optimize it for a suitable network with sufficient parameters and good generalization for classification or regression task. The number of neurons in the input layer corresponds to the number of measurements for the pattern problem, and the number of neurons in the output layer corresponds to the number of classes. This choice of layers number, number of neurons in each layer, and connections is known as the architecture problem. Such an architecture is depicted in figure 3.4.

In order to achieve a small variation between the network output and the desired output, learning for the MLP entails adjusting the connection weights. For this reason, various algorithms, including Ant colonies and the most popular Back-propagation algorithm, which is based on descent gradient techniques, are used in the literature.

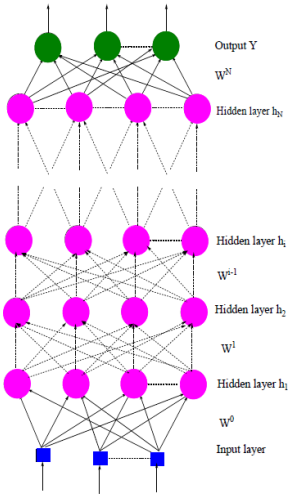


Figure 3.4. Feed forward NN structure [92]

Assuming that we used an input layer with n_0 neurons $X = (x_0, x_1, \dots, x_{n0})$ and a sigmoid activation function $f(x) = \frac{1}{1+e^{-x}}$. To obtain the network output we need to compute the output of each unit in each layer.

Consider a set of hidden layers (h_1, h_2,….., h_N). Assuming that n_i are the neurons number by each hidden layer h, then for the output of the first hidden layer we have:

$$h_j^i = f(\sum_{k=1}^{n_{i-1}} w_{k,j}^0 x_{k)}) \quad j = 1, \dots, n_i \tag{3.10}$$

The outputs h_i^j of neurons in the hidden layers are computing as flows:

$$h_j^i = f(\sum_{k=1}^{n_{i-1}} w_{k,j}^{i-1} h_{i-1}^k) \;\; i = 2, \dots, N \;\; and \; j = 1, \dots, n_i \tag{3.11}$$

Where $w_{k,j}^i$, is the weight between the neuron k in the hidden layer i and the neuron j in the hidden layer +1, n_i is the number of the neurons in the i-th hidden layer. The output of the i-th can be formulated as follows:

$$h_i = {}^t(h_i^1, h_i^2, \dots, h_i^{n_i}) \tag{3.12}$$

The the final result is computed by:

$$y_i = f(\sum_{k=1}^{n_N} w_{k,j}^N h_N^k) \;\; where \; Y = \left(y_1, \dots, y_j, \dots, y_{N+1}\right) = F(W, X) \tag{3.13}$$

Our model can be visualized through figure 3.5:

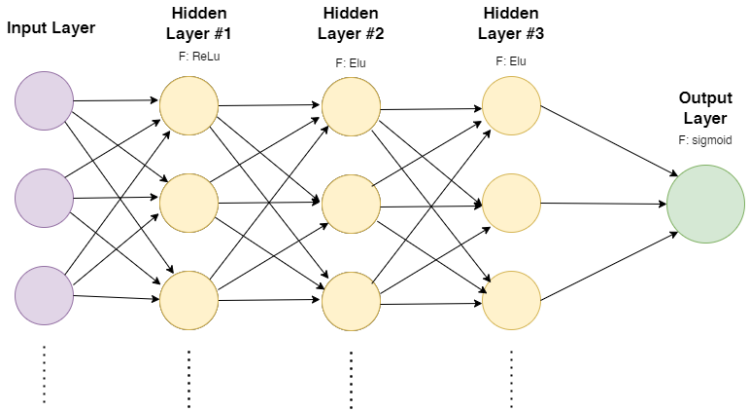
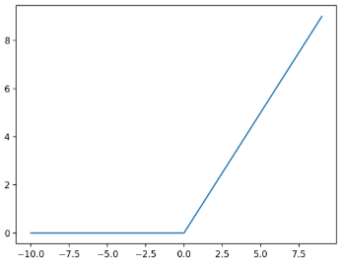


Figure 3.5. MLP model

The model consists of multiple dense (fully connected) layers arranged in a sequential manner. The input layer receives data from the cleaned and final training dataset and has the number of neurons equal to the number of attributes in the dataset. The first hidden layer has 1500 neurons which help capture the complex patterns of the input data and a Rectified Linear Unit activation function. This function reduces non-linearity and further identifies relationships between the initial attributes. [93] The ReLU function is characterised by the following relation, which can be illustrated through figure 3.7:

$$f(x) = \max(0, x) \qquad 4.14$$



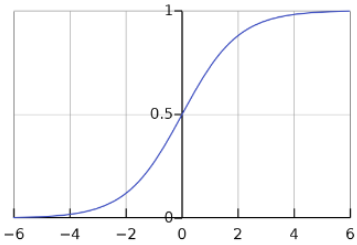


Figure 3.7. Graphical representation of the ReLU function [93] *Figure 3.6. Graphical representation of the sigmoid function [95]*

The second layer has 900 neurons and the activation function set as Exponential Linear Unit, which helps mitigate the vanishing gradient problem and provides better learning capabilities. The ELU function with $0 < \alpha$ is mathematically described as:

$$f(x) = x \text{ if } x > 0 \qquad 4.15$$
$$\alpha(\exp(x) - 1) \text{ if } x \leq 0$$

The third hidden layer has 850 neurons and also used the ELU activation function. [94] The output layer of the model has only one neuron in order to result in a single output, in our case either “0” or “1”. The activation function is sigmoid which, much like for the BLR model, squashes the output to a range between 0 and 1, representing the probability of the input belonging to a positive class, in our case the dead category[95]. Mathematically, the sigmoid function is graphically represented as seen in figure 3.6 and is defined as:

$$f(x) = \frac{1}{(1 + \exp(-x))} \qquad 4.16$$

In order to optimise the learning process of the model we added a regularization technique called Early Stopping in order to avoid overfitting and optimise the training time of the model. [96]–[98]This technique stops the training process of the model at the epoch in which the validation loss function starts to diverge from the training loss function, which symbolises the start of the overfitting process. This is illustrated in figure 3.8. A high number of epochs is necessary for better learning, but it can also lead to overfitting, which is what Early Stopping avoids. If the model overfits then it just learns the training data by heart and thus becomes unable to generalize for new data.

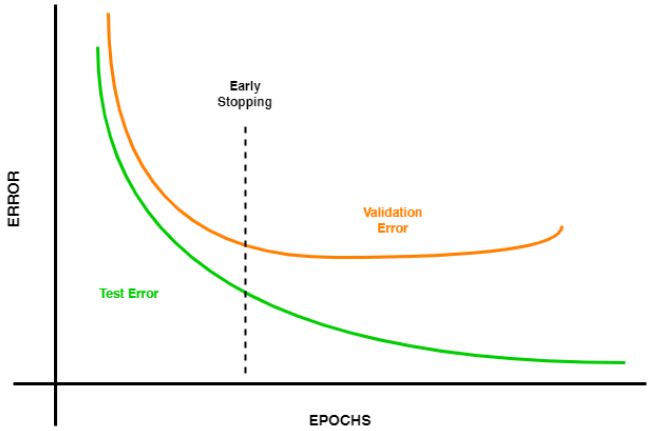


Figure 3.8. Early Stopping

Due to discussions with various members of the medical field we developed the following hypothesis that will be challenged in the stages that will be furthered explained:

These models take into consideration various laboratory works done by the ICU staff for the patient. For each hospital there is only one laboratory ward that handles all the laboratory analysis requests from all the wards of the hospital. This means that even if the ICU ward requests take priority, the results will still be received after a considerable amount of time. We propose the further development of two models, one of BLR type and one of the MLP type, that take into consideration only the attributes that can be received in real-time through the electronical devices and sensors that ensure the continuous monitorisation of the patients' state, respectively, the attributes that are already known at the moment of admission, such as comorbidities. If the models output a performance score close to the one of the models that also take into consideration the laboratory works, then it can be safely assumed that we don´t need 24 hours to make a good enough prediction of the patients' chances of survival.

The models that only take into consideration the strictly necessary attributes described previously will be named the same as the ones that take into consideration the full array of attributes but will contain the identificatory "Electro-" in the title for use of reference.

3.2.Performance metrics

The performance metrics for the hereby project have been chosen in conformity with the works studied in the State-of-the-Art section and with the needs that have been made obvious by the practical implementation of the project.

For each model, during the **Model Validation** step, multiple performance metrics have been assessed. These metrics consist of: test accuracy, variance, confusion matrices, Area Under a Receiver Operating Curve and Standardized Mortality Ratio. All of these validation metrics are used extensively in biomedical projects and represent a good overview of how well the developed models work.

The **confusion matrix** represents in our case a two-by-two matrix that assesses the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) values computed with the predicted and expected values of our models. [99], [100] The positives refer to the cases in which our event of interest has taken place, in our case this is represented by the event in which the patient has died in the ICU ward, while the negatives represent the cases in which the event didn't happen, in our case if the patient has lived. A good example of a confusion matrix is illustrated in figure 3.9.

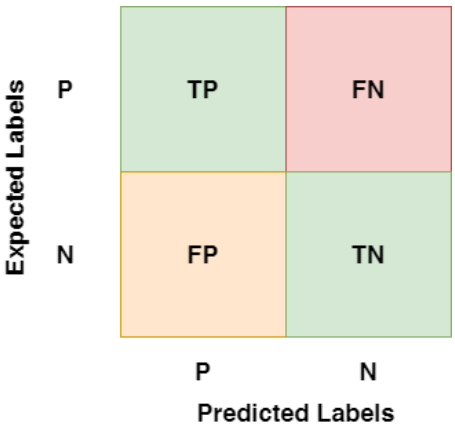


Figure 3.9. Confusion Matrix

The true positives value represents the number of entries that are positive, meaning that the event of interest took place, and have been predicted as such by our model. The true negative value represents the number of entries that are negative, meaning that the event of interest didn't take place, and have been predicted as such by our model. False Positives are the entries that are negative and have been predicted as positive, while False Negatives are entries that are positive but have been predicted as negative. In our case, the most concerning value from the confusion matrix is the False Negative, since this value shows us how many people the model predicted as survivors but have actually died.

By adding all the numbers in the confusion matrix, we will obtain the total number of entries used in the testing of our model. If we add the TP and FN values, we will obtain the total number of people that have died, while if we add the TN and FP values, we will obtain the total number of people that have lived.

The **test accuracy** is computed based on the values that we see in the confusion matrix. The formula for this metric is as follows:

$$\frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4.17}$$

meaning that the accuracy of our model is computed as the number of correct predictions divided by the total number of entries.

In the context of machine learning models, **variance** refers to the spread of the predicted probabilities across all instances of a dataset in regards to their mean value. It can be computed as the squared deviation from the mean of a random variable with the following formula: [101], [102]

$$Variance = \left(\frac{1}{n}\right) * \sum (y - \mu)^2 \tag{4.18}$$

Where n is the total number of predictions, y is an individual prediction and μ is the mean of all the predicted values.

The **Area Under the Receiver Operating Curve** is a very popular metric used to analyse the performance of binary classification models as it provides a result that assesses the capability of the model to distinguish between positive (dead) and negative (alive) instances. It employs the values of Sensitivity, or True Positive Rate, and Specificity, or True Negative Rate, that care computed with the values of the confusion matrix as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.19}$$

$$Specificity = \frac{TN}{TP + FN} \tag{4.20}$$

A value of the AUROC of 0.5 means that the model is simply randomly guessing the results, while an AUROC of 1.0 means that the model correctly predicts the result 100% of the time. The graphical representation of the

AUROC is done by plotting the Receiver Operating Characteristic (ROC) curve using the computed values for Sensitivity and Specificity at different thresholds. [103]

Another very important metric for the evaluation is the **Standardized Mortality Ratio** (SMR). This measure is commonly used in epidemiology and public health to compare the observed number of deaths in a specific population versus the expected number of deaths based on a standard population. This metric provides a good evaluation of our model because we can see through a percentage if the algorithm will prompt the user to ensure better care for more patients that are at risk. [104]

The formula for this metric is as follows:

$$SMR = \frac{Observed\ Deaths}{Expected\ Deaths} \tag{4.21}$$

The Observed Deaths represents the total number of deaths that have occurred in the ICU ward, in our case the value of True Positives plus the value of False Negatives, over the value of the True Positives plus the value of the False Positives. Therefore, the previous equation will take the form:

$$SMR = \frac{TP + FN}{TP + FP} \tag{4.22}$$

In order to interpret the SMR, we have the table 4.1:

Value of the SMR	**<100**	**=100**	**>100**
Interpretation	The observed number of deaths is smaller than expected; low mortality compared to reference	The observed number of deaths is equal to expected; standard mortality	The observed number of deaths is higher than expected; high mortality compared to reference

Table 3.1. SMR value map

4. Results

In this chapter we will analyse the results that each relevant step of our practical implementation provided and how they influenced the decision-making process as we proceeded towards the final form of our project. The entire code with its subsequent diagrams and outputs will be listed in Annex 1.

4.1. Model Implementation Results

The model development and implementation were done using Python as a programming language and the web computing platform Jupyter Notebook as the IDE. Libraries used include:

- **Numpy –** for numerical computations and efficient manipulation of the dataset
- **Pandas –** for the manipulation and analysis of the data
- **Seaborn –** for the exploratory analysis of the data and a better visual look of the graphs
- **Matplotlib –** for generating the graphs
- **Sklearn –** for machine learning algorithms
- **Tensorflow & keras –** for the development, training and deploying the machine learning and neural networks models
- **Imblearn –** for balancing out the dataset

In the first two steps of model implementation, namely, the **Exploratory Data Analysis** and the **Data Cleaning**, we performed various visualisation and statistically computation techniques that have helped us make informed decisions on what to do with the data. Our first task was to gain a general idea of the dataset: its shape, if it is balanced or not, if we have erroneous entries, the ranges of the parameters, and how many undefined values we had. This is process is depicted in figures 4.1 to 4.3 and table 4.1 and 4.2:

```
data_train.shape
```

```
(91713, 186)
```

Figure 4.1. Shape of dataset

Look for undefined:	
encounter_id	0
pre_icu_los_days	0
icu_type	0
icu_stay_type	0
icu_id	0
readmission_status	0
apache_post_operative	0
hospital_death	0
hospital_id	0
patient_id	0
elective_surgery	0
gender	25
icu_admit_source	112
d1_heartrate_min	145
d1_heartrate_max	145
…..	…..
h1_albumin_min	83824
h1_lactate_min	84369

h1_lactate_max	84369
h1_bilirubin_max	84619
h1_bilirubin_min	84619

Table 4.1 Number of undefined values

Number of unique genders	2
Number of unique ICU results	2
Number of unique categories post surgery	2

Table 4.2. Erroneous values

```
data_train.describe()
```

	encounter_id	patient_id	hospital_id	hospital_death	age	bmi	elective_surgery	height	icu_id	pre_icu_los_days	readmission_status
count	91713.000000	91713.000000	91713.000000	91713.000000	87485.000000	88284.000000	91713.000000	90379.000000	91713.000000	91713.000000	91713.0
mean	65606.079280	65537.131464	105.669262	0.086302	62.309516	29.185818	0.183736	169.641588	508.357692	0.835766	0.0
std	37795.088538	37811.252183	62.854406	0.280811	16.775119	8.275142	0.387271	10.795378	228.989661	2.487756	0.0
min	1.000000	1.000000	2.000000	0.000000	16.000000	14.844926	0.000000	137.200000	82.000000	-24.947222	0.0
25%	32852.000000	32830.000000	47.000000	0.000000	52.000000	23.641975	0.000000	162.500000	369.000000	0.035417	0.0
50%	65665.000000	65413.000000	109.000000	0.000000	65.000000	27.654655	0.000000	170.100000	504.000000	0.138889	0.0
75%	98342.000000	98298.000000	161.000000	0.000000	75.000000	32.930206	0.000000	177.800000	679.000000	0.409028	0.0
max	131051.000000	131051.000000	204.000000	1.000000	89.000000	67.814990	1.000000	195.590000	927.000000	159.090972	0.0

Figure 4.2. Dataset ranges

```
plt.figure()
data_train['hospital_death'].value_counts().plot(kind='bar')
plt.show()
```

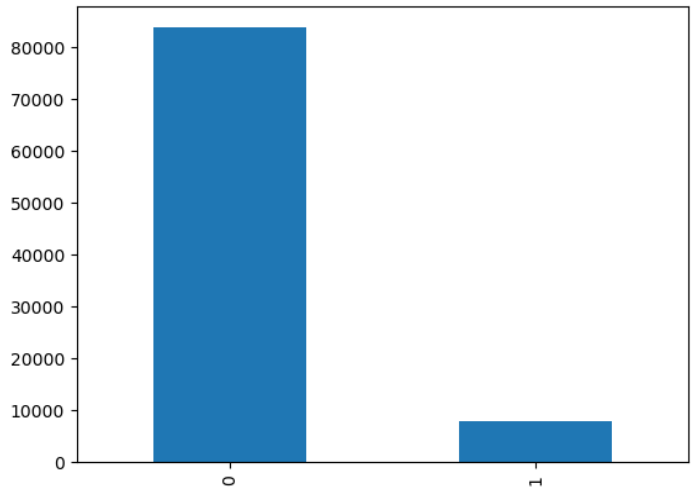


Figure 4.3. Dataset unbalance

During this process we found out that the dataset has 91713 entries, corresponding to the same number of patients, 186 attributes with various numbers of undefined values and highly different ranges, no erroneous entries. When we approached the issue of the balance of the dataset with respect to the end result, namely the 'hospital_death' attribute, we have seen that the dataset is highly imbalanced and that it will require a massive amount of pre-processing in order to be deemed fit for training the models. To do this, we agreed that we will need to drop the attributes with a lot of undefined values (within the order of tens of thousands) and for the attributes with only a few undefined values we would drop the subjects.

Since we had a lot of data attributes, the next task was to find out if some of them are highly correlated and if some of them could directly influence the mortality of the patient in any way. To do this, we employed the use of correlation matrices as seen in figure 4.4:

	encounter_id	patient_id	hospital_id	hospital_death	age	bmi	elective_surgery	height	icu_id	pre_icu_los_days	readmission_status	weight	albumin_apache	apache_2_diagnosis	apache_3j_diagnosis	apache_post_operative	arf_apache	bilirubin_apache	bun_apache	creatinine_apache	fio2_apache	gcs_eyes_apache	g
encounter_id	1.000000	-0.009575	-0.004532	-0.005243	-0.003990	-0.001068	0.002036	-0.005400	-0.000992	-0.000570	nan	-0.004376	0.006262	0.000113	-0.000651	0.001138	0.007920	-0.002198	0.002619	0.005435	-0.000824	0.003276	
patient_id	-0.009575	1.000000	-0.007075	0.004877	0.006343	-0.001380	0.001387	0.002902	-0.001770	-0.004412	nan	0.000275	-0.001534	-0.001539	0.004215	0.002260	0.003980	-0.004292	0.002860	0.004134	0.003811	0.001551	
hospital_id	-0.004532	-0.007075	1.000000	-0.001255	-0.008673	0.012874	0.052123	0.027895	0.004526	-0.001285	nan	0.026314	0.011685	0.006806	0.031896	0.053985	0.000844	0.017921	-0.014084	-0.001126	0.014941	-0.011727	
hospital_death	-0.005243	0.004877	-0.001255	1.000000	0.111017	-0.031247	-0.093574	-0.019526	0.000994	0.063316	nan	-0.038362	-0.193809	-0.089862	-0.090715	-0.083674	0.027309	0.137464	0.181435	0.114699	0.212249	-0.260373	
age	-0.003990	0.006343	-0.008673	0.111017	1.000000	-0.087077	0.067320	-0.109937	-0.024257	0.049872	nan	-0.127252	-0.116633	0.022914	-0.056060	0.059246	-0.001684	-0.044981	0.237187	0.056178	0.037935	0.026363	
bmi	-0.001068	-0.001380	0.012874	-0.031247	-0.087077	1.000000	0.015921	-0.056316	0.001403	-0.001531	nan	0.877339	0.052009	0.026047	-0.006514	0.015420	-0.005823	-0.001372	0.049392	0.069176	0.036961	0.012927	
elective_surgery	0.002036	0.001387	0.052123	-0.093574	0.067320	0.015921	1.000000	0.023620	-0.054997	0.133704	nan	0.026900	-0.024966	0.367040	0.775025	0.908247	-0.027357	-0.015510	-0.156585	-0.100575	-0.013290	0.009830	
height	-0.005400	0.002902	0.027895	-0.019526	-0.109937	-0.056316	0.023620	1.000000	0.018419	-0.008075	nan	0.391967	0.061671	0.001630	0.015268	0.025276	-0.010473	0.040743	0.019395	0.058072	0.047750	-0.008601	
icu_id	-0.000992	-0.001770	0.004526	0.000994	-0.024257	0.001403	-0.054997	0.018419	1.000000	-0.016783	nan	0.009510	0.134906	-0.029820	-0.034326	-0.052204	-0.008718	-0.030204	-0.017828	-0.020795	0.017081	-0.025385	
pre_icu_los_days	-0.000570	-0.004412	-0.001285	0.063316	0.049872	-0.001531	0.133704	-0.008075	-0.016783	1.000000	nan	-0.004423	-0.144441	0.086303	0.089141	0.127809	0.047991	0.071883	0.040184	0.025510	0.057288	-0.024837	
readmission_status	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	
weight	-0.004376	0.000275	0.026314	-0.038362	-0.127252	0.877339	0.026900	0.391967	0.009510	-0.004423	nan	1.000000	0.079116	0.026648	0.000560	0.026936	-0.010785	0.016438	0.052859	0.088643	0.057024	0.008699	
albumin_apache	0.006262	-0.001534	0.011685	-0.193809	-0.116633	0.052009	-0.024966	0.061671	0.134906	-0.144441	nan	0.079116	1.000000	0.039685	-0.085785	-0.055479	-0.030451	-0.139136	-0.215400	-0.117418	-0.110084	0.104544	
apache_2_diagnosis	0.000113	-0.001539	0.006806	-0.089862	0.022914	0.026047	0.367040	0.001630	-0.029820	0.086303	nan	0.026648	0.039685	1.000000	0.393110	0.394168	-0.003556	0.035519	-0.062268	0.004041	-0.074001	0.046596	
apache_3j_diagnosis	-0.000651	0.004215	0.031896	-0.090715	-0.056060	-0.006514	0.775025	0.015268	-0.034326	0.089141	nan	0.000560	-0.085785	0.393110	1.000000	0.875655	-0.026673	-0.002392	-0.107993	-0.049699	-0.068682	-0.030000	
apache_post_operative	0.001138	0.002260	0.053985	-0.083674	0.059246	0.015420	0.908247	0.025276	-0.052204	0.127809	nan	0.026936	-0.055479	0.394168	0.875655	1.000000	-0.028020	-0.012354	-0.158303	-0.100486	-0.009074	-0.010565	
arf_apache	0.007920	0.003980	0.000844	0.027309	-0.001684	-0.005823	-0.027357	-0.010473	-0.008718	0.047991	nan	-0.010785	-0.030451	-0.003556	-0.026673	-0.028020	1.000000	0.002400	0.181602	0.523163	0.001427	-0.004979	
bilirubin_apache	-0.002198	-0.004292	0.017921	0.137464	-0.044981	-0.001372	-0.015510	0.040743	-0.030204	0.071883	nan	0.016438	-0.139136	0.035519	-0.002392	-0.012354	0.002400	1.000000	0.105869	0.070756	0.047429	-0.033459	
bun_apache	0.002619	0.002860	-0.014084	0.181435	0.237187	0.049392	-0.156585	0.019395	-0.017828	0.040184	nan	0.052859	-0.215400	-0.062268	-0.107993	-0.158303	0.181602	0.105869	1.000000	0.685445	0.054668	-0.052208	
creatinine_apache	0.005435	0.004134	-0.001126	0.114699	0.056178	0.069176	-0.100575	0.058072	-0.020795	0.025510	nan	0.088643	-0.117418	0.004041	-0.049699	-0.100486	0.523163	0.070756	0.685445	1.000000	0.051726	-0.043944	
fio2_apache	-0.000824	0.003811	0.014941	0.212249	0.037935	0.036961	-0.013290	0.047750	0.017081	0.057288	nan	0.057024	-0.110084	-0.074001	-0.068682	-0.009074	0.001427	0.047429	0.054668	0.051726	1.000000	-0.171819	
gcs_eyes_apache	0.003276	0.001551	-0.011727	-0.260373	0.026363	0.012927	0.009830	-0.008601	-0.025385	-0.024837	nan	0.008699	0.104544	0.046596	-0.030000	-0.010565	-0.004979	-0.033459	-0.052208	-0.043944	-0.171819	1.000000	
gcs_motor_apache	0.007664	0.002216	-0.020512	-0.282449	0.025843	0.021091	0.015783	-0.013907	-0.023557	-0.014105	nan	0.013837	0.094410	0.057737	0.000214	-0.000022	-0.003129	-0.024412	-0.049718	-0.045939	-0.178063	0.800127	
gcs_unable_apache	0.002400	-0.006673	-0.017576	0.051774	-0.007754	0.006613	0.012191	-0.001541	-0.031625	0.008790	nan	0.004637	-0.028053	-0.005247	0.017742	0.026695	-0.005637	0.012761	0.002836	-0.002445	0.053490	nan	
gcs_verbal_apache	0.006516	0.000471	-0.002954	-0.241044	-0.017829	0.028515	-0.013365	0.009360	-0.033426	-0.045830	nan	0.030524	0.127784	0.020736	-0.052223	-0.037816	-0.003966	-0.029206	-0.070808	-0.037446	-0.185506	0.779128	
glucose_apache	0.002108	-0.000270	-0.000057	0.066430	0.007043	0.101440	-0.027103	-0.006517	0.019566	-0.012219	nan	0.093042	-0.032745	-0.054400	-0.001969	-0.026678	0.008209	-0.051921	0.146017	0.069675	0.058660	-0.054023	
heart_rate_apache	-0.001718	0.004735	-0.007649	0.107818	-0.149495	-0.021118	-0.069853	-0.024712	0.014562	0.050570	nan	-0.031767	-0.168265	-0.100654	0.003609	-0.055357	-0.012926	0.059229	0.019846	0.004863	0.081502	-0.093994	
hematocrit_apache	0.001566	-0.002944	0.002436	-0.062600	-0.118853	0.097660	-0.119344	0.127095	0.012340	-0.146998	nan	0.149615	0.417014	-0.049419	-0.150827	-0.133949	-0.104730	-0.129360	-0.227658	-0.178815	-0.019445	0.028605	
intubated_apache	-0.003172	0.001303	0.027670	0.173139	0.015396	0.037714	0.160831	0.019906	-0.075458	0.051842	nan	0.043666	-0.115567	0.021830	0.130453	0.185249	0.002043	0.022737	0.038005	0.031059	0.302014	-0.393832	
map_apache	0.000759	-0.002413	0.001011	-0.040526	-0.015870	0.055916	-0.003141	0.033535	-0.006784	-0.033403	nan	0.068163	0.153773	0.041950	-0.016333	-0.004632	0.003355	-0.055822	-0.075047	-0.004915	-0.011681	0.001585	
paco2_apache	0.015002	-0.006433	-0.020200	-0.035241	0.033000	0.180154	-0.042261	0.012735	0.000227	0.004541	nan	0.174372	0.103070	-0.052743	-0.116602	-0.055427	-0.016073	-0.117714	-0.032954	-0.078771	0.091900	0.051873	
paco2_for_ph_apache	0.015002	-0.006433	-0.020200	-0.035241	0.033000	0.180154	-0.042261	0.012735	0.000227	0.004541	nan	0.174372	0.103070	-0.052743	-0.116602	-0.055427	-0.016073	-0.117714	-0.032954	-0.078771	0.091900	0.051873	
pao2_apache	-0.001088	-0.001393	0.006726	0.012445	-0.020300	-0.105925	0.033604	-0.011149	0.008272	0.010903	nan	-0.103107	0.009397	0.016459	0.042297	0.046346	0.024911	0.001184	-0.034220	-0.012578	0.419487	-0.128401	
ph_apache	-0.005632	-0.006994	0.013026	-0.205075	0.046525	-0.051723	0.045551	-0.003760	0.001660	0.030282	nan	-0.051735	0.122269	0.072285	0.032313	0.040068	0.020384	-0.026770	-0.178154	-0.182575	-0.220608	0.079302	
resprate_apache	0.006498	0.000179	-0.023262	0.086666	0.037328	0.001725	-0.140857	-0.056965	-0.010378	0.017839	nan	-0.022152	-0.037571	-0.094633	-0.133946	-0.145773	0.007672	0.031126	0.071109	0.043130	0.062591	0.005172	
sodium_apache	-0.007004	-0.011433	0.001482	0.013216	0.037734	-0.020509	0.046775	-0.024410	-0.048874	-0.007318	nan	-0.031476	-0.003840	-0.027074	0.028527	0.053556	-0.058471	-0.104257	-0.005957	-0.072766	0.031149	-0.107701	
temp_apache	0.005162	0.002535	-0.033490	-0.158634	-0.082265	0.038597	-0.040545	0.014942	0.005871	0.008597	nan	0.043359	0.051831	0.008502	-0.001988	-0.047148	-0.020141	-0.029633	-0.094585	-0.065663	-0.066753	0.150900	
urineoutput_apache	-0.009225	-0.006058	-0.049851	-0.100210	-0.176577	0.054969	0.022915	0.080948	0.079749	-0.007246	nan	0.087051	0.133573	0.002892	0.035385	0.014287	-0.112303	-0.065980	-0.145557	-0.147847	-0.069711	0.057392	
ventilated_apache	-0.005775	-0.004433	0.030761	0.228661	0.041296	0.071014	0.146526	0.007192	0.003087	0.086566	nan	0.069173	-0.141802	-0.025350	0.110098	0.177874	0.002092	0.011228	0.063000	0.034206	0.280052	-0.488464	
wbc_apache	-0.009363	0.002185	0.006752	0.134265	-0.010786	0.030684	0.078414	-0.016120	0.015548	0.056522	nan	0.021845	-0.162105	-0.030257	0.094573	0.089643	-0.010124	0.051915	0.118825	0.060174	0.120054	-0.121295	
d1_diasbp_invasive_max	-0.013306	0.003669	0.007217	-0.034407	-0.190410	0.065902	-0.004486	0.019290	0.020016	-0.030097	nan	0.069473	0.061298	0.006866	0.021820	0.001640	-0.035550	-0.067603	-0.120029	-0.066219	-0.008265	-0.041827	
d1_diasbp_invasive_min	-0.004230	-0.001260	0.011703	-0.173632	-0.251508	0.051903	-0.012423	0.072085	0.013476	-0.054262	nan	0.080214	0.197068	0.028464	-0.025239	-0.028842	-0.067255	-0.080753	-0.197971	-0.148253	-0.086306	0.136932	

Figure 4.4. Correlation Matrix

These matrices were absolutely massive, having almost 40 000 values, so we needed a lot of time and attention in order to find the attributes that we needed to drop. In the case illustrated above, we dropped the attribute that had only "nan" values and we kept the others. In other cases, we dropped the features that had a high correlation between them, for example in the case of some daily and hourly attributes, we dropped the daily attributes because we were more interested in the hourly ones as they offered information of the patients' status faster.

The way in which we saw if one attribute influenced the mortality of the patient was to plot a barplot or a kde plot with the values of said attribute and the hospital death values. If the results were varied then we kept that attribute and if not, we dropped it. A depiction of such a process can be seen in figures 4.5 to 4.8:

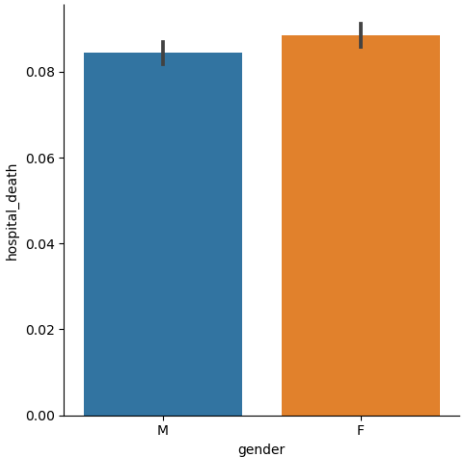


Figure 4.5. Barplot of dropped attribute

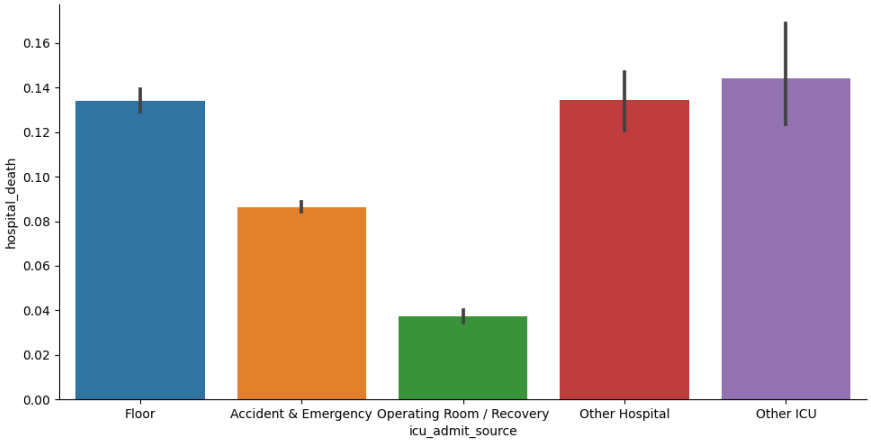


Figure 4.6. Barplot of kept attribute

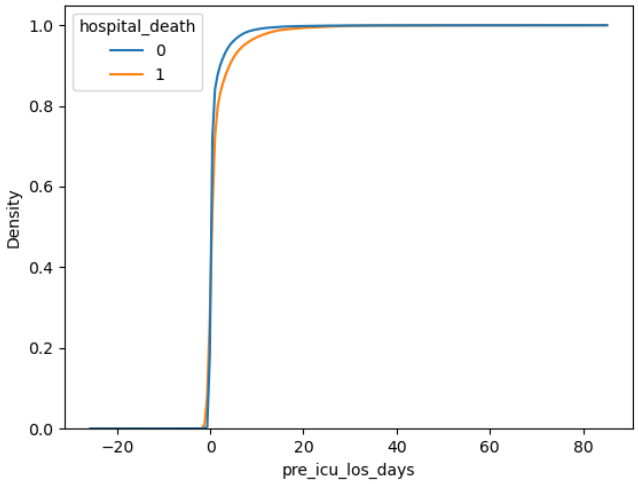


Figure 4.7. KDE plot of dropped attribute

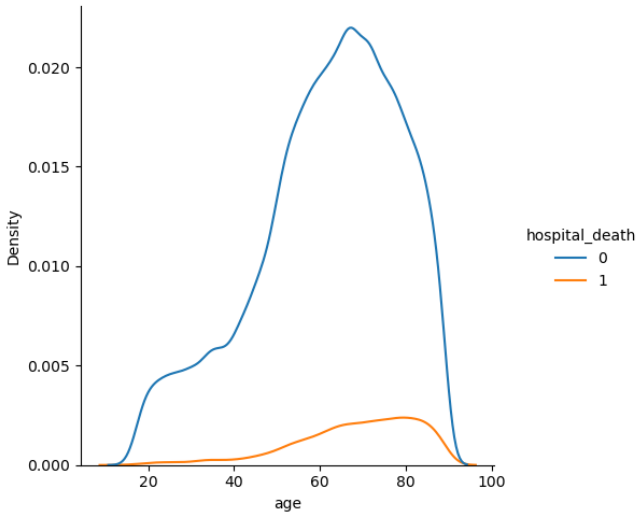


Figure 4.8. KDE plot of kept attribute

Sometimes, if we were unsure of what decision to make for a certain attribute, we had to employ the use of additional plots such as scatter plots or box plots in order to gain more information and find outliers that we had to drop, as depicted in figures 4.9 and 4.10:

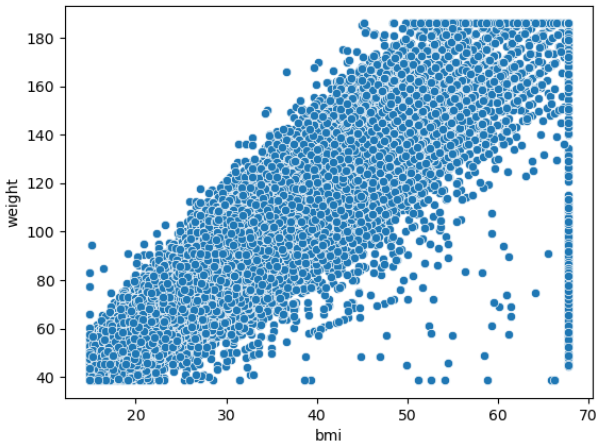


Figure 4.9. Scatter plot

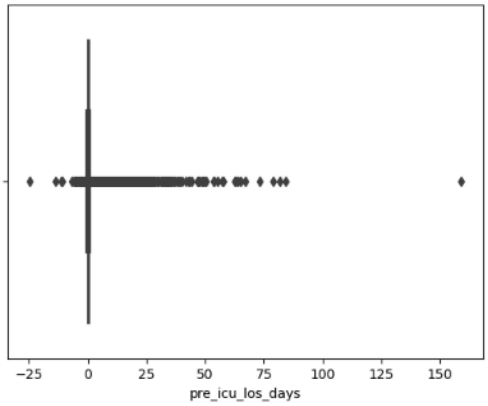


Figure 4.10. Box plot

In some cases, two attributes had almost the same meaning, values and influence over the final result, so we had to choose between them. Such a process can be observed in figure 4.11 and table 4.3:

Apache_3j_bodysystem	Apache_2_bodysystem
Neurological	Neurologic
Cardiovascular	Cardiovascular
Respiratory	Respiratory
Sepsis	
Trauma	Trauma
Metabolic	Metabolic
Gastrointestinal	Gastrointestinal
Genitourinary	Renal/ Genitourinary
Musculoskeletal/Skin	Undefined diagnoses
Hematological	Hematologic
Gynecological	Undefined diagnoses

Table 4.3. Decision process for APACHE body system

	apache_4a_hospital_death_prob	apache_4a_icu_death_prob
apache_4a_hospital_death_prob	1.000000	0.878959
apache_4a_icu_death_prob	0.878959	1.000000

Figure 4.11. Hospital vs. ICU death probability

In the case of values registered by electronic sensors we have concluded that the values obtained using invasive sensors are more relevant than the ones obtained with non-invasive devices, as they are more precise. However, the usage of invasive sensors is less usual than for non-invasive, therefore we had a lot of cases with values for the non-invasive attributes than for invasive. To deal with this issue, we identified the entries where we had values for non-invasive measurements but not for invasive and we copied those values in the invasive fields. Afterwards, we deleted the non-invasive attributes.

The next step implied identifying and transforming the String categories to categorical features so that they may be used in the training of the model. After dropping the features with multiple missing values and the entries from the attributes with fewer missing values, we were left with the clean dataset:

```
data_train_clean.shape
```

```
(48877, 72)
```

Figure 4.12. Clean dataset

For the “Electro_BLR” and “Electro_MLP” models we further modified the dataset by dropping the attributes that employed the results of various laboratory works. It is important to mention that the only difference between the “Electro_” and simple “BLR” and “MLP” models is the dataset that was used in training the models was different, the simple models using 71 attributes, while the “Electro_” models used 65 attributes.

Even if now we were left with about half of the entries that we originally had, the dataset was still very unbalanced, with a ratio of about 10:1 in favour of the survivors. After splitting the “hospital_death” attribute from the rest of the dataset in order to be used as the desired result in the development of our models, we had to apply the SMOTEEN technique in order to balance out the data.

Ultimately, we were left with a somewhat balanced dataset, in which we do indeed have more values for our event of interest. This will help us have better results when identifying the patients that have a high associated risk of death.

For all of the four models we split the data into training and testing sets, setting 30% of the data for the testing set and 70% of the data to the training set. Then we applied a standard scaling to all of the data in order to have the same ranges for all of the attributes. For the BLR models we next defined the model itself using the LogisticRegressionCV function from the sklearn.linear_model library and applying elastic net regularization. The ratio of L1 versus L2 regularization inside the elastic net was chosen through hyper parameter tuning.

For the MLP models we next defined the model itself. The model employs three hidden layers with a various number of neurons and different activation functions. The compilation of the model was done using the “adam” optimizer and “binary_crossentropy” for computing the loss. The “adam” optimizer is a popular optimization algorithm for neural networks due to its efficiency, adaptive learning rate and momentum features that contribute towards a robust performance and fast convergence. The “binary_crossentropy” is the loss function specifically designed for binary classification problems and it measures the dissimilarity between the predicted probability distribution and the true binary labels of the training data outputting a probability between 0 and 1. Next, we defined the early stopping function using the EarlyStopping function from the tensorflow.keras.callbacks library. The patience parameter specifies the number of epochs with no improvement in the monitored quantity after which training will be stopped, and in our case if the validation loss does not improve for 5 consecutive epochs, training will be stopped. The restore_best_weights parameter indicates whether to restore the weights of the model to the best observed state, thus, in our case, the weights of the model will be restored to the state where the validation loss was the lowest.

The training of the model has been done using 50 epochs, a batch size of 50, a validation split of 20% and the early stopping callback. The “epochs” parameter specifies the number of times the training data should be iterated during the training process, while the “batch_size” parameter determines the number of samples processed in each training iteration before updating the model's weights. The “validation_split” parameter determines the portion of the training data that will be used for validation during training.

4.2.Model Assessment Results

In this chapter we will discuss the results of the accuracy metrics used in validating our models and rank them accordingly.

4.2.1. BLR model

```
Accuracy: 0.8480259847202525
```

Figure 4.2.1. Accuracy of the BLR model

The accuracy of our BLR model was 84.8% (figure 4.2.1) and while this value is not entirely unliked, it is not very good either. The confusion matrix (figure 4.2.2) and the SMR value (figure 4.2.3) show that the prediction was done quite well, since we generally have less observed deaths than we expected. However, we have a pretty high number of patients that were predicted as alive but were actually dead, which, in a real-life scenario, means that those people would've received less appropriate care than they needed.

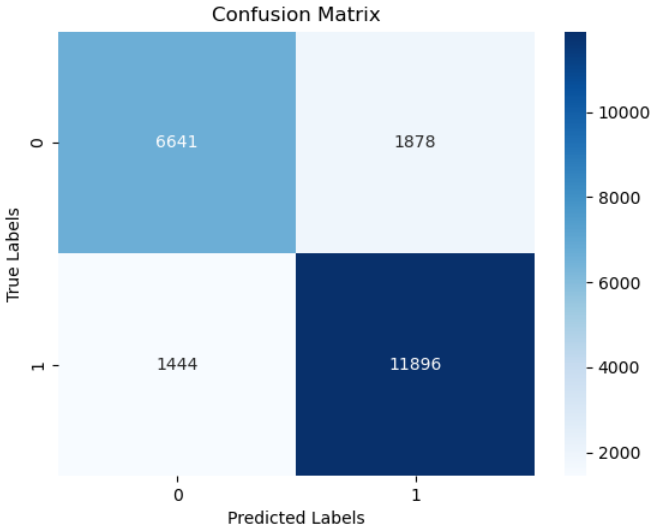


Figure 4.2.2. Confusion matrix of the BLR model

```
Standardized Mortality Ratio (SMR): 0.8917541229385307
```

Figure 4.2.3. SMR of BLR

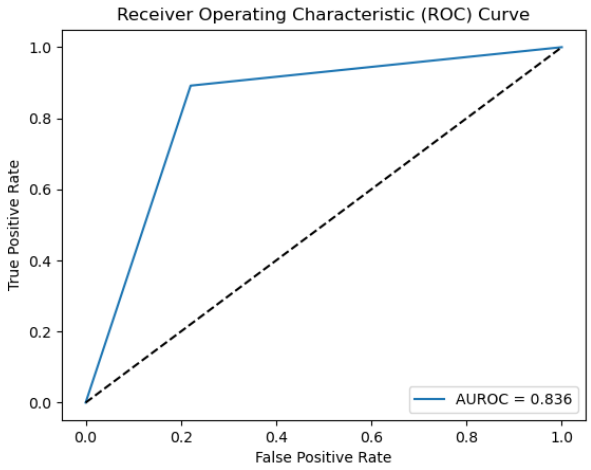


Figure 4.2.4. AUROC of the BLR model

The shape of the ROC curve (figure 4.2.4) is quite weird, but not unexpected, since the model predicts only results of either 0 or 1, thus not computing any additional threshold. The AUROC value is also too low for our liking, since in order to deem the model fit for clinical use has to be at the very least 0.85.

4.2.2. Electro_BLR model

The Electro_BLR model performs slightly worse than the simple BLR model. As we can see, we have an accuracy of 83.82%, and a SMR 89.11%. However, as we can see form the confusion matrix, we have again a quite high number of falsely predicted alive people. The AUROC is 0.823, which again, determines the fact that this model should not be clinically used. The shape of the ROC curve is again quite sharp, but this is normal as we can see from the probability distribution histogram of the BLR models. Even if the performances are not the best, the Electro_BLR model does not perform that much worse than the simple BLR model, especially in terms of the AUROC score, which is of most interest to us (the AUROC for the BLR model is 0.836 while for the Electo_BLR is 0.823). As mentioned before, since the result is either 0 or 1, the probability distribution will have a very clear-cut shape. All of the results can be visualised in figures 4.2.5 through 4.2.9

```
Accuracy: 0.8382231594388925
```

Figure 4.2.5. Accuracy of the Electro_BLR model

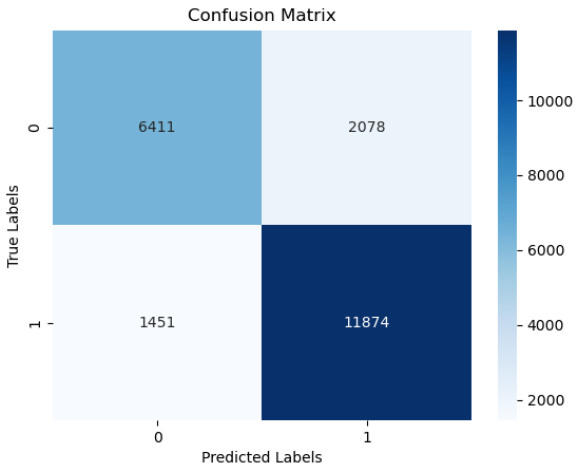


Figure 4.2.6. Confusion matrix of the Electro_BLR model

```
Standardized Mortality Ratio (SMR): 0.8911069418386491
```

Figure 4.2.7. SMR of Electro_BLR

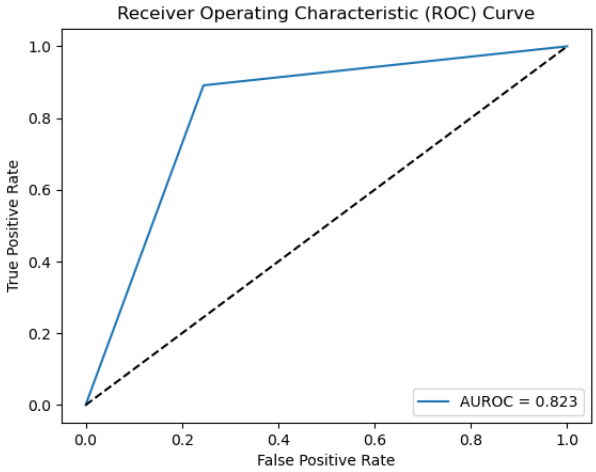


Figure 4.2.8. AUROC of the Electo_BLR model

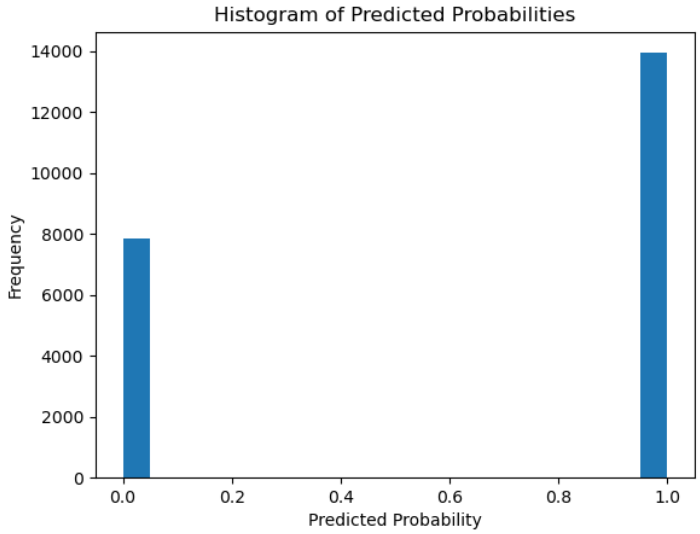


Figure 4.2.9. Probability distribution of the BLR models

4.2.3. MLP model

Form the start, we can see a big difference between the BLR models and the MLP model in terms of the accuracy scores. If the BLR models didn't even reach 90% accuracy, the MLP model has an accuracy of 95.22%. In the confusion matrix, if the erroneous predictions were among about 1500 and 2000 patients for the BLR model, the MLP model barely goes of 500 badly predicted cases. This means that among 21 842 cases only 541 patients were wrongly predicted as alive and subsequently died. The SMR is again under 1, thus we had less observed deaths than predicted, but the AUROC score is valued at 0.987, which means that our model makes a highly accurate prediction that can be successfully used in clinical tests. The ROC curve looks within parameters, which is due to the fact that in the case of the MLP model we were able to compute multiple thresholds, since the predictions were not done as simply 0 or 1, but due to the sigmoid function we were able to have multiple predictions within the interval [0,1], as can be seen from the probability distribution histogram of the MLP model. In order to have only predictions labelled as 0 or 1, we made sure to round any intermediary predictions as such: if the values were under 0.5 then the prediction was classified as 0, namely the patient will live, respectively, if they were higher than or equal to 0.5, then the prediction was classified as 1, thus the patient would be more likely to die. Since we have an interval of predictions rather than only two separate predictions, we chose an additional metric for how well the model performs, namely variance, which gave a score of 0.21, meaning that the outputs have a very low dispersion along the interval, which can also be seen from the histogram. All of the results can be visualised in figures 4.2.10 through 4.2.15.

```
Test accuracy: 0.9522022008895874
```

Figure 4.2.10. Accuracy of the MLP model

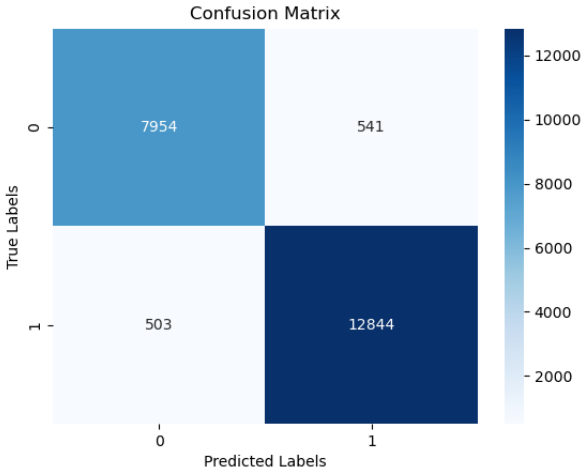


Figure 4.2.11. Confusion matrix of the MLP model

```
Standardized Mortality Ratio (SMR): 0.962313628530756
```

Figure 4.2.12. SMR of MLP

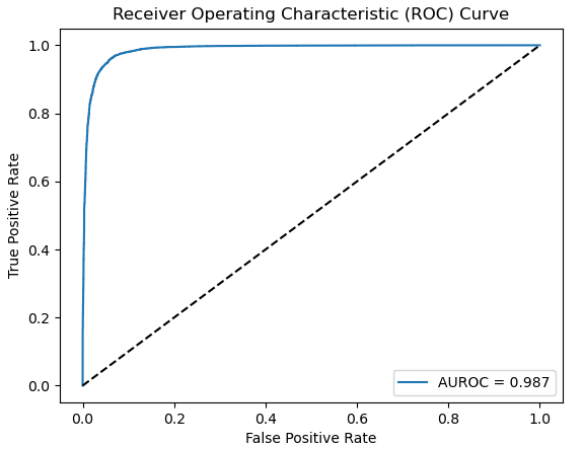


Figure 4.2.13. AUROC of the MLP model

```
Variance of predicted probabilities: 0.21887188
```

Figure 4.2.14. Variance of the MLP model

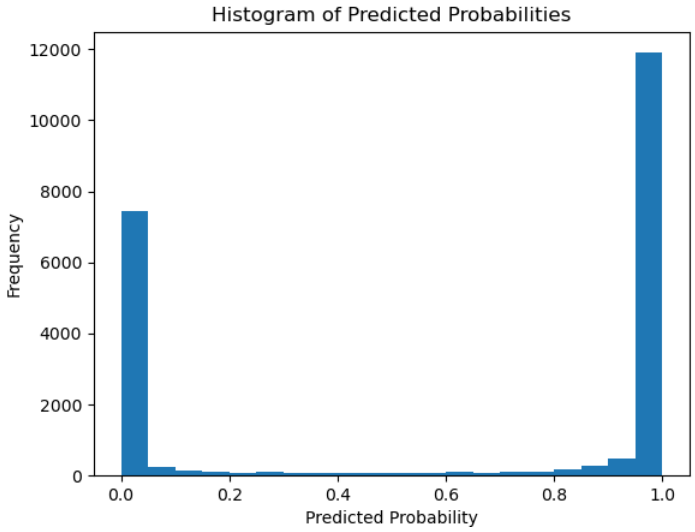


Figure 4.2.15. Probability distribution of the MLP model

4.2.4. Electro_MLP model

The Electro_MLP model also outputs a very high performance. The accuracy is indeed slightly lower than in the case of the MLP model at 92.73%, but the erroneous cases inside the confusion matrix are not that much higher than in the case of the MLP model (only 884 worst case scenario predictions out of 21 800 cases). The SMR in this case is also under 1 because we had less deaths than we expected, but the AUROC is 0.979, which is only 0.008 lower than in the MLP case, which is not a significant difference, thus meaning that the Electro_MLP model is also a very good candidate for making predictions in a clinical setting. The variance of the model is 0.2 which is slightly lower than in the MLP case, however, such a low difference can be deemed as insignificant. The probability distribution of the Electro_MLP model is similar to the MLP model. All of the results can be visualised in figures 4.2.16 through 4.2.21.

```
Test accuracy: 0.9273394346237183
```

Figure 4.2.16. Accuracy of the Electro_MLP model

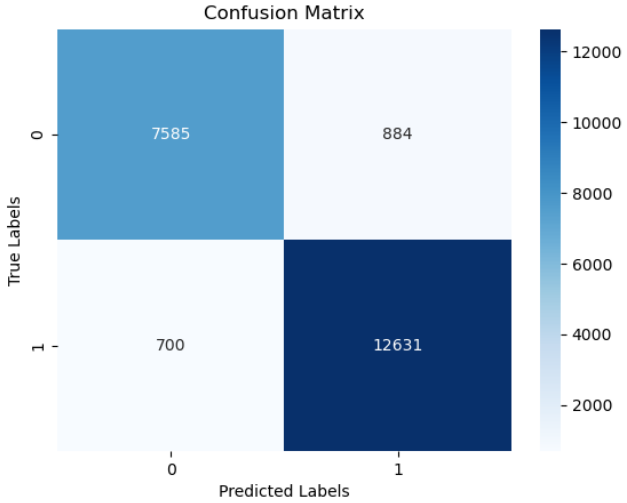


Figure 4.2.17. Confusion matrix of the Electro_MLP model

```
Standardized Mortality Ratio (SMR): 0.9474908108919061
```

Figure 4.2.18. SMR of Electro_MLP

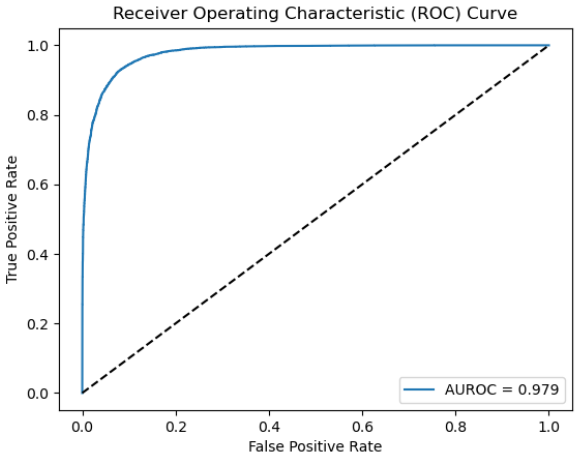


Figure 4.2.19. AUROC of the Electro_MLP model

```
Variance of predicted probabilities: 0.20082842
```

Figure 4.2.20. Variance of the Electro_MLP model

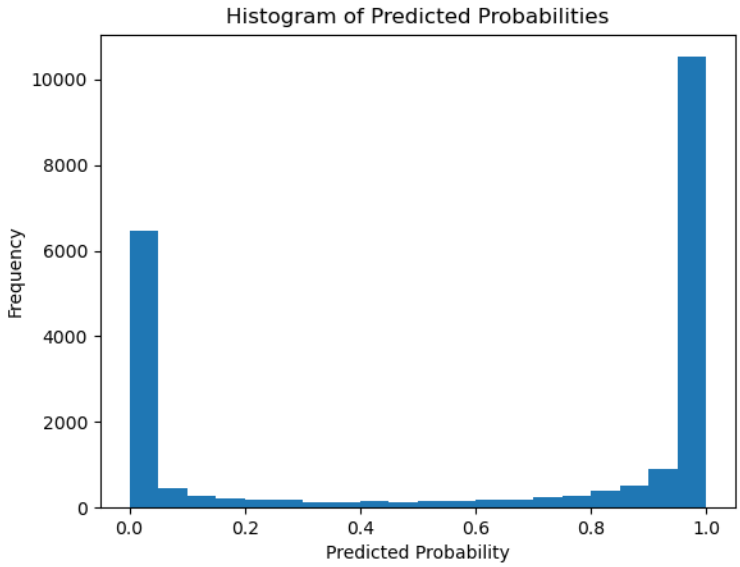


Figure 4.2.21. Probability distribution of the Electro_MLP model

4.3.Final Deployment and Discussion

The final deployment of the models was done on a new dataset provided by the WIDS 2020 Datathon. This dataset didn't contain any values for the "hospital_death" parameter, thus making it fit for a virtual simulation of how the models would act in a real-life clinical setting. This dataset had to be processed in the same way as the training dataset in order to make sure that we have valid inputs for our models to work with. The patient indexes were completely randomly chosen in order to have a fully objective view of their real-life performance. In order to additionally see how the models will perform, we also added a confidence rate, that tells us how sure is the model in regards to a prediction. A very low score means that the model is not sure at all of the prediction, while a higher score means that the model is fairly sure of the prediction it made. The confidence is computed as the absolute difference between the predicted probability and 0.5, subtracted from 1. This assumes that the prediction lies between 0 and 1. The closer the predicted probability is to 0.5, the lower the confidence, while the closer it is to 0 or 1, the higher the confidence. The computed confidence value ranges from 0 to 1, where 1 indicates the highest level of confidence in the prediction. This would occur when the predicted probability is either 0 or 1, meaning the prediction is very certain. The minimum possible value for the confidence is 0, indicating the lowest level of confidence in the prediction. This would occur when the predicted probability is exactly 0.5, which represents a situation where the model is uncertain about the prediction and considers it equally likely to be either class.

4.3.1. Deployment of the BLR model

For the BLR model we can see that the results are fairly inconsistent in terms of confidence. Even if the model makes a prediction, in some cases it is not really sure if the prediction it made is the correct one (for example in the case of patient 0, patient 150 and patient 500).

```
Patient Index: 0 - Prediction: 0 - Confidence: 0.07715721424609112
Patient Index: 10 - Prediction: 1 - Confidence: 0.49172573919110774
Patient Index: 150 - Prediction: 1 - Confidence: 0.11274535453616885
Patient Index: 230 - Prediction: 0 - Confidence: 0.35961991513785124
Patient Index: 500 - Prediction: 1 - Confidence: 0.1837963466754048
```

Figure 4.3.1. Deployment result of the BLR model

4.3.2. Deployment of the Electro_BLR model

In the case of the Electro_BLR model we can see that even if the prediction results are the same for each patient, the confidence scores are still very inconsistent and in some cases it performs either slightly better or worse than the BLR model.

```
Patient Index: 0 - Prediction: 0 - Confidence: 0.02354221325211059
Patient Index: 10 - Prediction: 1 - Confidence: 0.47561422630959005
Patient Index: 150 - Prediction: 1 - Confidence: 0.22601585148396475
Patient Index: 230 - Prediction: 0 - Confidence: 0.35709396164952534
Patient Index: 500 - Prediction: 1 - Confidence: 0.21539450207505062
```

Figure 4.3.2. Deployment result of the Electro_BLR model

4.3.3. Deployment of the MLP model

In the case of the MLP model we can see that the only predictions that remained the same as for the BLR models are the ones for patients 0, 10, and 230, however, the confidence scores are much higher which means that now the model is fairly sure of the prediction. The values of the confidence are also quite consistent, meaning that almost all the values are around 0.49 which is quite good.

```
617/617 [==============================] - 3s 5ms/step
Patient Index: 0 - Prediction: 0 - Confidence: 0.34842370450496674
Patient Index: 10 - Prediction: 1 - Confidence: 0.4993247389793396
Patient Index: 150 - Prediction: 0 - Confidence: 0.4991156570613384
Patient Index: 230 - Prediction: 0 - Confidence: 0.49999310845987566
Patient Index: 500 - Prediction: 0 - Confidence: 0.4933876181021333
```

Figure 4.3.3. Deployment result of the MLP model

4.3.4. Deployment of the Electro_MLP model

In the case of the Electro_MLP model we can see that the predictions are the same as the MLP model, however, some of the confidence scores have a much lower value, while others have stayed around the mark of 0.49. All in all, the Electro_MLP has a performance similar to the MLP model, only slightly worse, while still being better than the BLR models.

```
Patient Index: 0 - Prediction: 0 - Confidence: 0.10731622576713562
Patient Index: 10 - Prediction: 1 - Confidence: 0.4994514584541321
Patient Index: 150 - Prediction: 0 - Confidence: 0.19229960441589355
Patient Index: 230 - Prediction: 0 - Confidence: 0.4983331789262593
Patient Index: 500 - Prediction: 0 - Confidence: 0.4940086589194834
```

Figure 4.3.4. Deployment result of the Electro_MLP model

5. Conclusions

5.1. Main ideas

After validating the models and analysing how the algorithms perform both accuracy-wise and in a simulated real-life environment, we can confirm the fact that the simple machine learning binary logistic regression models perform worse than the deep learning multilayer perceptron models as was presented in the State of The Art, since their accuracy is overall smaller in all of the metrics used.

On another note, our proposed theory that we could be able to develop a good enough prediction model using just the immediate electronically acquired data and the information available at the moment of admission in the Intensive Care Unit has been confirmed. The "Electro_" models have performed only slightly worse than the models that also took in consideration the laboratory works, which is normal. Considering that normally the models with the laboratory works would require 24 hours to register the necessary data, the trade-off between the slightly worse "Electro_" models and the time required for them to make a good enough prediction is satisfactory, since we should be able to get a good result could be 12, 6 or maybe even 3 hours. This could enable the medical staff to make faster decisions concerning the transferring of the patient to another hospital, rushing the treatment schema or the amount of supervision the patient requires.

5.2. Limitations and Forward Approach

There are several limitations we should take into consideration when attempting a further approach: in order to actually implement an application that could be used by the medical staff, we should make sure that all the data acquired by the sensors is sent through an IoT system to our application, that should be able, for a given period of time, to compute the minimum and maximum of our attributes just as we have them in the training set. The application should also be able to receive manual input in order to give the medical staff the ability to insert the data we have at the admission of the patient in the ICU ward. Considering the discussions, we had with various doctors and medical students, the GUI of the application should be as simple to use as possible, in order to encourage the medical staff to implement the use of that application in their practice.

Another think we should take into consideration going forward would be to add in our database data about the EEG signal of the patient in order to include in our possible prediction cases the situations in which a patient is found clinically dead. Other approaches for future development of our application would be building a regional or European database in order to enable the application to be used successfully in our area, considering the fact that our currently used database doesn't have any data from Europe.

Future project ideas implemented using the hereby work as a basis should have the goal of implementing the infrastructure proposed before and running tests on the "Electro_" models in order to determine how fast can a reliable prediction concerning the survival chances of a patient be made. The proposed times are 3, 6 and 12 hours.

5.3. Personal Contributions

Personal contributions include, but are not limited to:

- Extensive research of the State of The Art;
- Using a highly diverse and comprehensive database and its successful handling, without corrupting or invasively modifying the data;
- The development of personal prediction models that use regularization techniques, more layers with a varied number of electrons and multiple activation techniques;
- Formulation and validation of a hypothesis regarding the necessary attributes required to obtain optimal performances, which enables a new precondition for future works developed on the same topic;

- Successfully presenting an intermediate stage of the project in front of an accredited evaluation commission, implementing the feedback given by this entity and obtaining the third place in the Student Scientific Communication Session (Annex 2);

6. Bibliography

[1] R. Rigo-Bonnin *et al.*, "Individual outcome prediction models for patients with COVID-19 based on their first day of admission to the intensive care unit," *Clin Biochem*, vol. 100, pp. 13–21, Feb. 2022, doi: 10.1016/j.clinbiochem.2021.11.001.

[2] L. Famiglini, A. Campagner, A. Carobene, and F. Cabitza, "A robust and parsimonious machine learning method to predict ICU admission of COVID-19 patients," *Medical and Biological Engineering and Computing*. Springer Science and Business Media Deutschland GmbH, 2022. doi: 10.1007/s11517-022-02543-x.

[3] C. Gholipour, F. Rahim, A. Fakhree, and B. Ziapour, "Using an artificial neural networks (ANNS) model for prediction of intensive care unit (ICU) outcome and length of stay at hospital in traumatic patients," *Journal of Clinical and Diagnostic Research*, vol. 9, no. 4, pp. 19–23, Apr. 2015, doi: 10.7860/JCDR/2015/9467.5828.

[4] W. A. Knaus *et al.*, "The APACHE III prognostic system: Risk prediction of hospital mortality for critically III hospitalized adults," *Chest*, vol. 100, no. 6, pp. 1619–1636, 1991, doi: 10.1378/chest.100.6.1619.

[5] E. Paul, M. Bailey, and D. Pilcher, "Risk prediction of hospital mortality for adult patients admitted to Australian and New Zealand intensive care units: Development and validation of the Australian and New Zealand Risk of Death model," *J Crit Care*, vol. 28, no. 6, pp. 935–941, Dec. 2013, doi: 10.1016/j.jcrc.2013.07.058.

[6] E. Paul, M. Bailey, and D. Pilcher, "Risk prediction of hospital mortality for adult patients admitted to Australian and New Zealand intensive care units: Development and validation of the Australian and New Zealand Risk of Death model," *J Crit Care*, vol. 28, no. 6, pp. 935–941, Dec. 2013, doi: 10.1016/j.jcrc.2013.07.058.

[7] E. Paul, M. Bailey, J. Kasza, and D. v Pilcher, "Assessing contemporary intensive care unit outcome: development and validation of the Australian and New Zealand Risk of Death admission model," 2017.

[8] "WiDS Datathon 2020 | Kaggle." https://www.kaggle.com/competitions/widsdatathon2020/overview (accessed Jun. 11, 2023).

[9] S. Kim and S. Kang, "Serum Albumin Levels: A Simple Answer to a Complex Problem? Are We on the Right Track of Assessing Metabolic Syndrome?," *Endocrinology and Metabolism*, vol. 28, no. 1, p. 17, 2013, doi: 10.3803/ENM.2013.28.1.17.

[10] P. B. Soeters, R. R. Wolfe, and A. Shenkin, "Hypoalbuminemia: Pathogenesis and Clinical Significance," *Journal of Parenteral and Enteral Nutrition*, vol. 43, no. 2, pp. 181–193, Feb. 2019, doi: 10.1002/JPEN.1451.

[11] S. Angielski and J. Rogulski, "Metabolic studies in experimental renal dysfunction resulting from maleate administration.," *Curr Probl Clin Biochem*, vol. 4, pp. 86–100, 1975, doi: 10.5061/DRYAD.P4S57.

[12] S. Angielski and J. Rogulski, "Metabolic studies in experimental renal dysfunction resulting from maleate administration.," *Curr Probl Clin Biochem*, vol. 4, pp. 86–100, 1975, doi: 10.5061/DRYAD.P4S57.

[13] M. E. Harper and A. Dugaiczyk, “Linkage of the evolutionarily-related serum albumin and alpha-fetoprotein genes within q11-22 of human chromosome 4.,” *Am J Hum Genet*, vol. 35, no. 4, p. 565, 1983, Accessed: Jun. 10, 2023. [Online]. Available: /pmc/articles/PMC1685723/?report=abstract

[14] J. W. Hawkins and A. Dugaiczyk, “The human serum albumin gene: structure of a unique locus,” *Gene*, vol. 19, no. 1, pp. 55–58, Jul. 1982, doi: 10.1016/0378-1119(82)90188-3.

[15] C. B. PUESTOW, “THE DISCHARGE OF BILE INTO THE DUODENUM,” *Archives of Surgery*, vol. 23, no. 6, p. 1013, Dec. 1931, doi: 10.1001/archsurg.1931.01160120127008.

[16] “Overview of Hemolytic Anemia - Hematology and Oncology - Merck Manuals Professional Edition.” https://www.merckmanuals.com/professional/hematology-and-oncology/anemias-caused-by-hemolysis/overview-of-hemolytic-anemia?query=Autoimmune%20Hemolytic%20Anemia (accessed Jun. 10, 2023).

[17] “Bilirubin blood test: MedlinePlus Medical Encyclopedia.” https://medlineplus.gov/ency/article/003479.htm (accessed Jun. 10, 2023).

[18] L. Mosqueda, K. Burnight, and S. Liao, “The Life Cycle of Bruises in Older Adults,” *J Am Geriatr Soc*, vol. 53, no. 8, pp. 1339–1343, Aug. 2005, doi: 10.1111/J.1532-5415.2005.53406.X.

[19] C. Pirone, J. Martin E Quirke, H. A. Priestap, and D. W. Lee, “The Animal Pigment Bilirubin Discovered in Plants,” *J Am Chem Soc*, vol. 131, no. 8, p. 2830, Mar. 2009, doi: 10.1021/JA809065G.

[20] “Bilirubin Blood Test: MedlinePlus Medical Test.” https://medlineplus.gov/lab-tests/bilirubin-blood-test/ (accessed Jun. 10, 2023).

[21] M. E. Smith and D. G. Morton, “LIVER AND BILIARY SYSTEM,” *The Digestive System*, pp. 85–105, Jan. 2010, doi: 10.1016/B978-0-7020-3367-4.00006-2.

[22] M. E. Smith and D. G. Morton, “LIVER AND BILIARY SYSTEM,” *The Digestive System*, pp. 85–105, Jan. 2010, doi: 10.1016/B978-0-7020-3367-4.00006-2.

[23] E. Macedo and R. L. Mehta, “Clinical approach to the diagnosis of acute kidney injury,” *National Kidney Foundation’s Primer on Kidney Diseases, Sixth Edition*, pp. 294–303, Jan. 2013, doi: 10.1016/B978-1-4557-4617-0.00033-9.

[24] Y. Xue, L. B. Daniels, A. S. Maisel, and N. Iqbal, “Cardiac Biomarkers,” *Reference Module in Biomedical Sciences*, 2014, doi: 10.1016/B978-0-12-801238-3.00022-2.

[25] “Calcium - Health Professional Fact Sheet.” https://ods.od.nih.gov/factsheets/Calcium-HealthProfessional/#en1 (accessed Jun. 10, 2023).

[26] “Creatinine blood test Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/creatinine-blood-test (accessed Jun. 10, 2023).

[27] “Creatinine Blood Test: Normal, Low, High Levels, Causes & Symptoms.” https://www.medicinenet.com/creatinine_blood_test/article.htm (accessed Jun. 10, 2023).

[28] “Creatinine tests - Mayo Clinic.” https://web.archive.org/web/20220605052251/https://www.mayoclinic.org/tests-procedures/creatinine-test/about/pac-20384646 (accessed Jun. 10, 2023).

[29] “Lewis’ Medical-Surgical Nursing Elsevier eBook on VitalSource, 11th Edition - 9780323595193.” https://evolve.elsevier.com/cs/product/9780323595193?role=student (accessed Jun. 10, 2023).

[30] “Cellulose and Cellulose Derivatives - Kenji Kamide - Google Books.” https://books.google.ro/books?id=28Vx9OkEtQcC&pg=PA1&redir_esc=y#v=onepage&q&f=false (accessed Jun. 10, 2023).

[31] “Handbook of Biodegradable Polymers - Google Books.” https://books.google.ro/books?id=iLjhl6AvfIsC&pg=PA275&redir_esc=y#v=onepage&q&f=false (accessed Jun. 10, 2023).

[32] “Mean fasting blood glucose.” https://www.who.int/data/gho/indicator-metadata-registry/imr-details/2380 (accessed Jun. 10, 2023).

[33] “Blood sugar test - blood Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/blood-sugar-test-blood (accessed Jun. 10, 2023).

[34] “Bicarbonate Blood Test & Carbon Dioxide (CO2) Levels in Blood.” https://www.webmd.com/a-to-z-guides/bicarbonate-blood-test-overview (accessed Jun. 10, 2023).

[35] “Hemoglobin Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/hemoglobin (accessed Jun. 10, 2023).

[36] H. H. Billett, “Hemoglobin and Hematocrit,” *Anesthesiology*, vol. 28, no. 4, pp. 763–763, Jul. 1990, doi: 10.1097/00000542-196707000-00028.

[37] “Hematocrit and Hemoglobin.” https://www.redcrossblood.org/donate-blood/dlp/hematocrit.html (accessed Jun. 10, 2023).

[38] S. Shikdar, R. Vashisht, and P. T. Bhattacharya, “International Normalized Ratio (INR),” *StatPearls*, May 2023, Accessed: Jun. 10, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK507707/

[39] C. D. Foucher and R. E. Tubben, “Lactic Acidosis,” *StatPearls*, Mar. 2023, Accessed: Jun. 10, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK470202/

[40] “Lactic Acidosis - StatPearls - NCBI Bookshelf.” https://www.ncbi.nlm.nih.gov/books/NBK470202/ (accessed Jun. 10, 2023).

[41] P. Wacharasint, T. A. Nakada, J. H. Boyd, J. A. Russell, and K. R. Walley, “Normal-range blood lactate concentration in septic shock is prognostic and predictive,” *Shock*, vol. 38, no. 1, pp. 4–10, Jul. 2012, doi: 10.1097/SHK.0B013E318254D41A.

[42] “Platelet count Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/platelet-count (accessed Jun. 10, 2023).

[43] “Potassium test Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/potassium-test (accessed Jun. 10, 2023).

[44] N. S. Gregory *et al.*, “Potassium citrate decreases bone resorption in postmenopausal women with osteopenia: A randomized, double-blind clinical trial,” *Endocrine Practice*, vol. 21, no. 12, pp. 1380–1386, Dec. 2015, doi: 10.4158/EP15738.OR.

[45] K. J. Aaron and P. W. Sanders, “Role of dietary salt and potassium intake in cardiovascular health and disease: A review of the evidence,” *Mayo Clin Proc*, vol. 88, no. 9, pp. 987–995, 2013, doi: 10.1016/j.mayocp.2013.06.005.

[46] “Potassium | The Nutrition Source | Harvard T.H. Chan School of Public Health.” https://www.hsph.harvard.edu/nutritionsource/potassium/ (accessed Jun. 10, 2023).

[47] M. Vinceti, T. Filippini, A. Crippa, A. de Sesmaisons, L. A. Wise, and N. Orsini, “Meta-analysis of potassium intake and the risk of stroke,” *J Am Heart Assoc*, vol. 5, no. 10, Oct. 2016, doi: 10.1161/JAHA.116.004210.

[48] Y. Ma *et al.*, “24-Hour Urinary Sodium and Potassium Excretion and Cardiovascular Risk,” *New England Journal of Medicine*, vol. 386, no. 3, pp. 252–263, Jan. 2022, doi: 10.1056/NEJMOA2109794.

[49] “Sodium blood test Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/sodium-blood-test (accessed Jun. 10, 2023).

[50] “Sodium (Blood) - Health Encyclopedia - University of Rochester Medical Center.” https://www.urmc.rochester.edu/encyclopedia/content.aspx?contenttypeid=167&contentid=sodium_blood (accessed Jun. 10, 2023).

[51] “WBC count Information | Mount Sinai - New York.” https://www.mountsinai.org/health-library/tests/wbc-count (accessed Jun. 10, 2023).

[52] A. Arshad and R. Tasnim, “An inductive transducer based pressure sensor for biomedical applications,” *2015 International Conference on Informatics, Electronics & Vision (ICIEV)*, Nov. 2015, doi: 10.1109/ICIEV.2015.7333987.

[53] “Difference Between Systolic And Diastolic Blood Pressures - Viva Differences.” https://vivadifferences.com/systolic-vs-diastolic-blood-pressures/ (accessed Jun. 10, 2023).

[54] “Cardiac Cycle (Lesson) – Human Bio Media.” https://www.humanbiomedia.org/cardiac-cycle-lesson/ (accessed Jun. 10, 2023).

[55] “How Do Blood Pressure Monitors Work? – Livongo Tech Blog.” https://techblog.livongo.com/how-do-blood-pressure-monitors-work/ (accessed Jun. 10, 2023).

[56] G. Sainas *et al.*, “Mean Blood Pressure Assessment during Post-Exercise: Result from Two Different Methods of Calculation,” *J Sports Sci Med*, vol. 15, no. 3, p. 424, Sep. 2016, Accessed: Mar. 29, 2023. [Online]. Available: /pmc/articles/PMC4974855/

[57] D. DeMers and D. Wachs, “Physiology, Mean Arterial Pressure,” *StatPearls*, Apr. 2022, Accessed: Mar. 29, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK538226/

[58] “Normal Heart Rate: Range, When It’s Dangerous, and More.” https://www.healthline.com/health/dangerous-heart-rate (accessed Jun. 10, 2023).

[59] “How pulse oximeters work explained simply.” https://www.howequipmentworks.com/pulse_oximeter/ (accessed Jun. 10, 2023).

[60] F. Q. Nuttall, “Body mass index: Obesity, BMI, and health: A critical review,” *Nutrition Today*, vol. 50, no. 3. Lippincott Williams and Wilkins, pp. 117–128, May 17, 2015. doi: 10.1097/NT.0000000000000092.

[61] J. Gil Cebrian, M. P. Bello Cámara, and R. Diaz-Alersi, “APACHE II.,” *Intensive Care Medicine*, vol. 13, no. 2. p. 143, 1987. doi: 10.1097/00003465-198603000-00013.

[62] E. T. Petridou and C. N. Antonopoulos, “Injury Epidemiology,” *International Encyclopedia of Public Health*, pp. 258–274, Oct. 2016, doi: 10.1016/B978-0-12-803678-5.00233-2.

[63] J. H. Park and J. W. Koo, “Neurosensory diagnostic techniques for mild traumatic brain injury,” *Neurosensory Disorders in Mild Traumatic Brain Injury*, pp. 279–302, Jan. 2019, doi: 10.1016/B978-0-12-812344-7.00017-0.

[64] S. Chakraborty, B. E. Skolnick, W. M. Alves, L. F. Marshall, and R. K. Narayan, “Traumatic Brain Injury,” *Handbook of Neuroemergency Clinical Trials*, pp. 85–109, Jan. 2017, doi: 10.1016/B978-0-12-804064-5.00005-9.

[65] R. Zangari, F. Biroli, and P. Gritti, “Predictors of outcome in moderate and severe traumatic brain injury,” *Diagnosis and Treatment of Traumatic Brain Injury: The Neuroscience of Traumatic Brain Injury*, pp. 15–26, Jan. 2022, doi: 10.1016/B978-0-12-823347-4.00001-4.

[66] J. L. McDougal, “Trauma Scoring Systems,” *The Parkland Trauma Handbook: Mobile Medicine Series*, pp. 18–23, Nov. 2008, doi: 10.1016/B978-0-323-05226-9.50011-6.

[67] N. M. VanDerHeyden and T. B. Cox, “Trauma Scoring,” *Current Therapy of Trauma and Surgical Critical Care*, pp. 26–32, 2008, doi: 10.1016/B978-0-323-04418-9.50010-2.

[68] D. DeMers and D. Wachs, “Physiology, Mean Arterial Pressure,” *StatPearls*, Apr. 2022, Accessed: Apr. 15, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK538226/

[69] S. Sharma and M. F. Hashmi, “Partial Pressure Of Oxygen,” *StatPearls*, Dec. 2022, Accessed: Apr. 16, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK493219/

[70] I. E. P. Trulock, “Arterial Blood Gases,” *Butterworths*, pp. 255–257, 1990, Accessed: Apr. 16, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK371/

[71] S. S. Prabhakar, “Nephrology,” *Medical Secrets*, pp. 190–219, Jan. 2011, doi: 10.1016/B978-0-323-06398-2.00009-6.

[72] V. Simon, D. D. Ho, and Q. Abdool Karim, “HIV/AIDS epidemiology, pathogenesis, prevention, and treatment,” *Lancet*, vol. 368, no. 9534, p. 489, Aug. 2006, doi: 10.1016/S0140-6736(06)69157-5.

[73] D. Schuppan and N. H. Afdhal, “Liver Cirrhosis,” *Lancet*, vol. 371, no. 9615, p. 838, Mar. 2008, doi: 10.1016/S0140-6736(08)60383-9.

[74] A. T. Kharroubi and H. M. Darwish, “Diabetes mellitus: The epidemic of the century,” *World J Diabetes*, vol. 6, no. 6, p. 850, Jun. 2015, doi: 10.4239/WJD.V6.I6.850.

[75] N. McAvoy, E. Thomson, and E. S. Wilson, “Hepatic Failure,” *Anaesthesia and Intensive Care Medicine*, vol. 13, no. 4, pp. 161–165, Feb. 2023, doi: 10.1016/j.mpaic.2012.01.013.

[76] H. Nakamura and H. Maeda, “Cancer Chemotherapy,” *Fundamentals of Pharmaceutical Nanoscience*, pp. 401–427, Feb. 2023, doi: 10.1007/978-1-4614-9164-4_15.

[77] T. Lightfoot, A. Smith, and E. Roman, “Leukemia,” *International Encyclopedia of Public Health*, pp. 410–418, Jan. 2023, doi: 10.1016/B978-0-12-803678-5.00253-8.

[78] P. Blombery, A. Bloor, and D. C. Linch, “Non-Hodgkin Lymphoma,” *Treatment of Cancer, Sixth Edition*, pp. 649–678, Feb. 2023, doi: 10.1201/b17751-30.

[79] E. Zacksenhaus and S. E. Egan, “Progression to Metastasis of Solid Cancer,” *Cancers (Basel)*, vol. 13, no. 4, pp. 1–8, Feb. 2021, doi: 10.3390/CANCERS13040717.

[80] “GOSSIS: Global Open Source Severity of Illness Score: International Benchmarking for Critical Care.” https://gossis.mit.edu/ (accessed Apr. 19, 2023).

[81] J. D. Raffa *et al.*, “The Global Open Source Severity of Illness Score (GOSSIS),” *Crit Care Med*, vol. 50, no. 7, pp. 1040–1050, Jul. 2022, doi: 10.1097/CCM.0000000000005518.

[82] “seaborn.heatmap — seaborn 0.12.2 documentation.” https://seaborn.pydata.org/generated/seaborn.heatmap.html (accessed Jun. 10, 2023).

[87] “SMOTEENN — Version 0.10.1.” https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTEENN.html#imblearn.combine.SMOTEENN (accessed Jun. 10, 2023).

[88] “SMOTE — Version 0.10.1.” https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html (accessed Jun. 10, 2023).

[89] “EditedNearestNeighbours — Version 0.10.1.” https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.EditedNearestNeighbours.html (accessed Jun. 10, 2023).

[90] “Application of Binary Logistic Regression Model to Cancer Patients: a case study of data from Erbil in Kurdistan region of Iraq.,” *Zanco J Pure Appl Sci*, vol. 33, no. 4, Aug. 2021, doi: 10.21271/zjpas.33.4.12.

[91] “L1 and L2 Regularization Methods. Machine Learning | by Anuja Nagpal | Towards Data Science.” https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c (accessed Jun. 10, 2023).

[92] H. Ramchoun, M. Amine, J. Idrissi, Y. Ghanou, and M. Ettaouil, “Multilayer Perceptron: Architecture Optimization and Training,” *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 1, p. 26, 2016, doi: 10.9781/ijimai.2016.415.

[93] “ReLU Explained | Papers With Code.” https://paperswithcode.com/method/relu (accessed Jun. 10, 2023).

[94] “ELU Explained | Papers With Code.” https://paperswithcode.com/method/elu (accessed Jun. 10, 2023).

[95] “Sigmoid Activation Explained | Papers With Code.” https://paperswithcode.com/method/sigmoid-activation (accessed Jun. 10, 2023).

[96] “A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks - MachineLearningMastery.com.” https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/ (accessed Jun. 10, 2023).

[97] L. Prechelt, “Early Stopping - But When?,” pp. 55–69, 1998, doi: 10.1007/3-540-49430-8_3.

[98] L. Prechelt, “Automatic early stopping using cross validation: Quantifying the criteria,” *Neural Networks*, vol. 11, no. 4, pp. 761–767, Jun. 1998, doi: 10.1016/S0893-6080(98)00010-0.

[99] A. Kulkarni, D. Chong, and F. A. Batarseh, “Foundations of data imbalance and solutions for a data democracy,” *Data Democracy: At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering*, pp. 83–106, Jan. 2020, doi: 10.1016/B978-0-12-818366-3.00005-8.

[100] D. K. Sharma, M. Chatterjee, G. Kaur, and S. Vavilala, “Deep learning applications for disease diagnosis,” *Deep Learning for Medical Applications with Unique Data*, pp. 31–51, Jan. 2022, doi: 10.1016/B978-0-12-824145-5.00005-8.

[101] L. Wasserman, “All of Statistics,” 2004, doi: 10.1007/978-0-387-21736-9.

[102] L. Wasserman, “All of statistics : a concise course in statistical inference,” 2004.

[103] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve,” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982, doi: 10.1148/radiology.143.1.7063747.

[104] J. L. Kelsey and E. B. Gold, “Observational Epidemiology,” *International Encyclopedia of Public Health*, pp. 295–307, Jan. 2017, doi: 10.1016/B978-0-12-803678-5.00310-6.

[105] Dragoș - Daniel Țarălungă, Instrumentație Biomedicală: Măsurarea și Analiza Biopotențialelor, Editura Matrix Rom, pp. 311, 2013, ISBN 978-973-755-945-6,

[106] Boeru Elena-Briana, „Estimation of patient survival chances based on the medical data acquired in the first 24 hours in the Intensive Care Unit”, Premiul III, Sesiunea de Comunicări Științifice Studențești, 5-6 mai 2023

7. Annex 1

7.1. Code snippets

7.1.1. CLEAN_COD_LICENTA.jpynb

```
#!/usr/bin/env python
# coding: utf-8

# # Cod Licență 24h survival prediction

# In[1]:


import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import math as math
import sklearn as sklearn
import random as random
from sklearn.preprocessing import
StandardScaler
from sklearn.preprocessing import
LabelEncoder
from sklearn.model_selection import
train_test_split
from sklearn.neighbors import
KNeighborsClassifier
from sklearn import metrics
from sklearn.metrics import
confusion_matrix
from sklearn.model_selection import
LeaveOneOut
get_ipython().run_line_magic('matplotlib'
, 'notebook')
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import
StandardScaler
from sklearn.metrics.cluster import
adjusted_rand_score
from sklearn.metrics.cluster import
rand_score
from IPython.display import clear_output
from sklearn.metrics import roc_curve,
roc_auc_score, auc
from sklearn.preprocessing import
LabelEncoder
from sklearn.model_selection import
StratifiedKFold
from tensorflow import feature_column


# In[2]:


import os
import tensorflow as tf

os.environ['CUDA_VISIBLE_DEVICES'] = '-1'

if tf.test.gpu_device_name():
    print('GPU found')
else:
    print("No GPU found")


# In[3]:


from tensorflow import keras
from tensorflow.keras.callbacks import
EarlyStopping


# In[4]:


pd.set_option("display.max_columns",
None)
pd.set_option("display.max_rows", None)


# # Data explore & pre-processing

# In[5]:


data_dictionary =
pd.read_csv('WiDS_Datathon_2020_Dictionar
y.csv') #data is presumed to be in the
same folder.
data_dictionary.info()
data_dictionary.head(188)


# In[6]:


data_train =
pd.read_csv('training_v2.csv')
data_train.head()


# In[7]:


data_dictionary["Category"].value_counts(
)


# In[8]:


columns_labs_new=data_dictionary[data_dic
tionary['Category']=="labs"]
print(columns_labs_new)


# In[9]:


data_test = pd.read_csv('unlabeled.csv')
data_test.head()


# In[10]:
```

```
data_train.info()


# In[11]:


data_test.info()


# In[12]:


data_train.shape


# In[13]:


data_test.shape


# In[14]:



undf=data_train.isnull().sum()
print("\n\n", '\033[1m'+ 'Look for
undefined:' +
'\033[0m',"\n",undf.sort_values())


# We have a lot of undefined values so we
need to do some pre-processing and see
which attributes are more detrimental for
the final result; Drop the ones with a
lot of undefined; For the ones with few
undefined, drop the subjects.

# In[15]:


print("Number of unique genders:
",data_train['gender'].nunique())


# In[16]:


print("Number of unique ICU results:
",data_train['hospital_death'].nunique())


# In[17]:


print("Number of unique categoies post
surgery:
",data_train['elective_surgery'].nunique(
))


# In[18]:


data_train.describe()


# In[19]:


data_train.corr().style.background_gradie
nt(cmap='coolwarm')


# In[20]:


data_train_clean=data_train.drop("readmis
sion_status", axis='columns') #all of
them are nan values so we drop


# In[21]:


plt.figure()
data_train['hospital_death'].value_counts
().plot(kind='bar')
plt.show()


# In[22]:


sb.catplot(x="gender",y="hospital_death",
kind="bar",data=data_train)
plt.show()
#I use seaborn catplot function because I
can make multiple types of plots with
only one function, by changing the "kind"
param


# In[23]:


data_train_clean=data_train_clean.drop("g
ender", axis='columns') #there is no real
difference between male and female
#death correlation so it doesn't
influence =» we drop


# In[24]:


data_train_clean.head() #check that it
worked


# In[25]:


sb.catplot(x="ethnicity",y="hospital_deat
h",kind="bar",data=data_train)
plt.show()


# In[26]:


data_train_clean=data_train_clean.drop("e
thnicity", axis='columns')#there is no
real difference between ethnicities
#death correlation so it doesn't
influence =» we drop
```

```
# In[27]:


data_train['hospital_id'].count()
print("Number of unique hospitals:
",data_train['hospital_id'].nunique())


# In[28]:



sb.catplot(x="hospital_id",y="hospital_de
ath",kind="bar",data=data_train)
plt.show() #a lot of difference =» we
keep
#also, all new checked values are the
same in clean and data_train, so we check
in data_train and modify in clean


# In[29]:


print("Number of unique patients:
",data_train['patient_id'].nunique())
#unique values, no info about group


# In[30]:


data_train_clean=data_train_clean.drop("p
atient_id", axis='columns')


# In[31]:


print("Number of unique encounters:
",data_train['encounter_id'].nunique())


# In[32]:


data_train_clean=data_train_clean.drop("e
ncounter_id", axis='columns')


# In[33]:



sb.catplot(x="age",y="hospital_death",kin
d="bar",data=data_train)
plt.show()#a lot of difference =» we
keep; older people = more death


# In[34]:


sb.displot(data=data_train, x="age",
hue="hospital_death", kind="kde")
plt.show()


# In[35]:


data_train["icu_id"].value_counts()


# In[36]:


sb.catplot(x="icu_id",y="hospital_death",
kind="bar",data=data_train)
plt.show()#a lot of difference =» we
keep;


# In[37]:


sb.displot(data=data_train, x="bmi",
hue="hospital_death", kind="kde")
plt.show() #we need a kernel density
estimate that plots dead vs. alive in a
normalized way


# In[38]:


plt.figure()
sb.kdeplot(
    data=data_train, x="bmi",
hue="hospital_death",
    cumulative=True, common_norm=False,
common_grid=True,
)
plt.show()


# In[39]:


data_train_clean=data_train_clean.drop("b
mi", axis='columns') #even if the first
kde seems to give us a lot of difference,
#in a normalised form we can see that no
matter the bmi, dead vs. alive is kind of
the same. =»


# In[40]:


plt.figure()
sb.scatterplot(data=data_train, x="bmi",
y="weight")
plt.show() #no real info, but it does
seem like the bmi vs. weight follows the
graph we expect from the equation:
bmi=weight/height


# In[41]:


sb.displot(data=data_train, x="weight",
hue="hospital_death", kind="kde")
plt.show() #same as for bmi


# In[42]:
```

```
data_train_clean=data_train_clean.drop("w
eight", axis='columns')


# In[43]:


sb.displot(data=data_train, x="height",
hue="hospital_death", kind="kde")
plt.show()


# In[44]:


plt.figure()
sb.kdeplot(
    data=data_train, x="height",
hue="hospital_death",
    cumulative=True, common_norm=False,
common_grid=True,
)
plt.show()


# In[45]:


data_train_clean=data_train_clean.drop("h
eight", axis='columns')


# In[46]:


sb.catplot(x="icu_stay_type",y="hospital_
death",kind="bar",data=data_train)
plt.show()


# In[47]:


sb.catplot(x="icu_type",y="hospital_death
",kind="bar",data=data_train)
plt.show()


# In[48]:


data_train_clean=data_train_clean.drop("i
cu_stay_type", axis='columns') #the ICU
type is more relevant than the stay type
bc it
#is more varied


# In[49]:


sb.catplot(x="hospital_admit_source",y="h
ospital_death",kind="bar",data=data_train
)
plt.show()


# In[50]:


sb.catplot(x="icu_admit_source",y="hospit
al_death",kind="bar",data=data_train)
plt.show()


# In[51]:


sb.displot(data=data_train,
x="pre_icu_los_days", kde=True)
plt.show()


# In[52]:


plt.figure()
sb.boxplot(data=data_train,
x="pre_icu_los_days")
plt.show()


# In[53]:


sb.displot(data=data_train,
x="pre_icu_los_days",
hue="hospital_death", kind="kde")
plt.show()


# In[54]:


df=data_train[['patient_id','pre_icu_los_
days']]


# In[55]:


df[df.pre_icu_los_days>120]


# In[56]:


data_train.drop(38854, inplace=True)


# In[57]:


plt.figure()
sb.boxplot(data=data_train,
x="pre_icu_los_days")
plt.show()


# In[58]:


sb.displot(data=data_train,
x="pre_icu_los_days",
hue="hospital_death", kind="kde")
plt.show()


# In[59]:
```

```
plt.figure()
sb.kdeplot(
    data=data_train,
x="pre_icu_los_days",
hue="hospital_death",
    cumulative=True, common_norm=False,
common_grid=True,
)
plt.show()


# In[60]:


data_train_clean=data_train_clean.drop("p
re_icu_los_days", axis='columns') #not
really enough difference


# In[61]:


plt.figure()
data_train['elective_surgery'].value_coun
ts().plot(kind='bar')
plt.show()


# In[62]:


plt.figure()
sb.countplot(data=data_train,
x="elective_surgery",
hue="hospital_death")
plt.show()


# In[63]:


sb.catplot(x="elective_surgery",y="hospit
al_death",kind="bar",data=data_train)
plt.show()


# In[64]:


print("Number of unique bodysystems for
apache 3j:
",data_train['apache_3j_bodysystem'].nuni
que())


# In[65]:


print("Number of unique bodysystems for
apache 2:
",data_train['apache_2_bodysystem'].nuniq
ue())


# In[66]:


print(data_test['apache_3j_bodysystem'].u
nique())


# In[67]:


print(data_test['apache_2_bodysystem'].un
ique())


# In[68]:


data_train_clean=data_train_clean.drop("a
pache_2_bodysystem", axis='columns')
#they are kind of the same as apache 3


# In[69]:


sb.catplot(x="apache_3j_bodysystem",y="ho
spital_death",kind="bar",data=data_train)
plt.show()


# In[70]:


print(data_dictionary[data_dictionary['Ca
tegory']=="APACHE prediction"])


# In[71]:


print(data_dictionary[data_dictionary['Ca
tegory']=="identifier"])


# In[72]:


print(data_dictionary[data_dictionary['Ca
tegory']=="APACHE grouping"])


# In[73]:


print(data_dictionary[data_dictionary['Ca
tegory']=="APACHE comorbidity"])


# In[74]:


print(data_dictionary[data_dictionary['Ca
tegory']=="demographic"])


# In[75]:


print(data_dictionary[data_dictionary['Ca
tegory']=="labs blood gas"])


# In[76]:
```

```
print(data_dictionary[data_dictionary['Ca
tegory']=="APACHE covariate"])


# In[77]:


print(data_dictionary[data_dictionary['Ca
tegory']=="labs"])


# In[78]:


print(data_dictionary[data_dictionary['Ca
tegory']=="vitals"])


# In[79]:


data_train[["apache_4a_hospital_death_pro
b","apache_4a_icu_death_prob"]].corr().st
yle.background_gradient(cmap='coolwarm')


# In[80]:


data_train_clean=data_train_clean.drop("a
pache_4a_hospital_death_prob",
axis='columns')


# In[81]:


mask = data_train_clean[['aids',
'leukemia',
'cirrhosis','diabetes_mellitus',

'hepatic_failure','immunosuppression','ly
mphoma',

'solid_tumor_with_metastasis']].notna().a
ll(axis=1)
data_train_clean = data_train_clean[mask]
#bc .dropna makes it so that I can't use
.head() or .shape


# In[82]:


undf_clean=data_train_clean.isnull().sum(
)
print("\n\n", '\033[1m'+ 'Look for
undefined:' +
'\033[0m',"\n",undf_clean.sort_values())


# In[83]:


data_train_clean.shape


# In[84]:


data_train[["d1_arterial_pco2_max","d1_ar
terial_pco2_min","d1_arterial_ph_max","d1
_arterial_ph_min",

"d1_arterial_po2_max","d1_arterial_po2_mi
n","d1_pao2fio2ratio_max","d1_pao2fio2rat
io_min",

"h1_arterial_pco2_max","h1_arterial_pco2_
min","h1_arterial_ph_max","h1_arterial_ph
_min",

"h1_arterial_po2_max","h1_arterial_po2_mi
n","h1_pao2fio2ratio_max","h1_pao2fio2rat
io_min",

]].corr().style.background_gradient(cmap=
'coolwarm')


# In[85]:


data_train_clean=data_train_clean.drop(["
d1_arterial_pco2_max","d1_arterial_pco2_m
in","d1_arterial_ph_max",

"d1_arterial_ph_min","d1_arterial_po2_max
","d1_arterial_po2_min",

"d1_pao2fio2ratio_max","d1_pao2fio2ratio_
min"],axis=1)


# In[86]:


columns_Apache_covar=data_dictionary[data
_dictionary['Category']=="APACHE
covariate"]
print(columns_Apache_covar)


# In[87]:


abs(data_train[columns_Apache_covar["Vari
able
Name"]].corr()).style.background_gradient
(cmap='coolwarm')


# In[88]:


data_train_clean=data_train_clean.drop(["
apache_3j_diagnosis","gcs_eyes_apache","g
cs_motor_apache",

"gcs_verbal_apache","paco2_apache"],axis=
1)


# In[89]:


columns_vitals=data_dictionary[data_dicti
onary['Category']=="vitals"]
print(columns_vitals)
```

```
# In[90]:


abs(data_train[columns_vitals["Variable
Name"]].corr()).style.background_gradient
(cmap='coolwarm')


# In[91]:


data_train_clean['d1_mbp_invasive_max']=d
ata_train_clean['d1_mbp_invasive_max'].fi
llna(data_train_clean['d1_mbp_noninvasive
_max'])


# In[92]:


data_train_clean=data_train_clean.drop("d
1_mbp_noninvasive_max", axis='columns')


# In[93]:


data_train_clean['d1_diasbp_invasive_max'
]=data_train_clean['d1_diasbp_invasive_ma
x'].fillna(data_train_clean['d1_diasbp_no
ninvasive_max'])


# In[94]:


data_train_clean=data_train_clean.drop("d
1_diasbp_noninvasive_max",
axis='columns')


# In[95]:


data_train_clean['d1_diasbp_invasive_min'
]=data_train_clean['d1_diasbp_invasive_mi
n'].fillna(data_train_clean['d1_diasbp_no
ninvasive_min'])


# In[96]:


data_train_clean=data_train_clean.drop("d
1_diasbp_noninvasive_min",
axis='columns')


# In[97]:


data_train_clean['d1_mbp_invasive_min']=d
ata_train_clean['d1_mbp_invasive_min'].fi
llna(data_train_clean['d1_mbp_noninvasive
_min'])


# In[98]:


data_train_clean=data_train_clean.drop("d
1_mbp_noninvasive_min", axis='columns')


# In[99]:


data_train_clean['d1_sysbp_invasive_max']
=data_train_clean['d1_sysbp_invasive_max'
].fillna(data_train_clean['d1_sysbp_nonin
vasive_max'])


# In[100]:


data_train_clean=data_train_clean.drop("d
1_sysbp_noninvasive_max", axis='columns')


# In[101]:


data_train_clean['d1_sysbp_invasive_min']
=data_train_clean['d1_sysbp_invasive_min'
].fillna(data_train_clean['d1_sysbp_nonin
vasive_min'])


# In[102]:


data_train_clean=data_train_clean.drop("d
1_sysbp_noninvasive_min", axis='columns')


# In[103]:


data_train_clean['h1_diasbp_invasive_max'
]=data_train_clean['h1_diasbp_invasive_ma
x'].fillna(data_train_clean['h1_diasbp_no
ninvasive_max'])


# In[104]:


data_train_clean=data_train_clean.drop("h
1_diasbp_noninvasive_max",
axis='columns')


# In[105]:


data_train_clean['h1_diasbp_invasive_min'
]=data_train_clean['h1_diasbp_invasive_mi
n'].fillna(data_train_clean['h1_diasbp_no
ninvasive_min'])


# In[106]:


data_train_clean=data_train_clean.drop("h
1_diasbp_noninvasive_min",
axis='columns')
```

```
# In[107]:


data_train_clean['h1_mbp_invasive_max']=d
ata_train_clean['h1_mbp_invasive_max'].fi
llna(data_train_clean['h1_mbp_noninvasive
_max'])


# In[108]:


data_train_clean=data_train_clean.drop("h
1_mbp_noninvasive_max", axis='columns')


# In[109]:


data_train_clean['h1_sysbp_invasive_max']
=data_train_clean['h1_sysbp_invasive_max'
].fillna(data_train_clean['h1_sysbp_nonin
vasive_max'])


# In[110]:


data_train_clean=data_train_clean.drop("h
1_sysbp_noninvasive_max", axis='columns')


# In[111]:


data_train_clean['h1_sysbp_invasive_min']
=data_train_clean['h1_sysbp_invasive_min'
].fillna(data_train_clean['h1_sysbp_nonin
vasive_min'])


# In[112]:


data_train_clean=data_train_clean.drop("h
1_sysbp_noninvasive_min", axis='columns')


# In[113]:


columns_labs=data_dictionary[data_diction
ary['Category']==""labs"]
print(columns_labs)


# In[114]:


abs(data_train[columns_labs["Variable
Name"]].corr()).style.background_gradient
(cmap='coolwarm')


# In[115]:


data_train_clean=data_train_clean.drop(['
d1_albumin_max','d1_albumin_min','d1_bili
rubin_max','d1_bilirubin_min',
'd1_bun_max','d1_bun_min','d1_calcium_max
','d1_calcium_min','d1_creatinine_max','d
1_creatinine_min',
'd1_glucose_max','d1_glucose_min','d1_hco
3_max','d1_hco3_min','d1_hemaglobin_max',
'd1_hemaglobin_min',
'd1_hematocrit_max','d1_hematocrit_min','
d1_inr_max','d1_inr_min','d1_lactate_max'
,'d1_lactate_min',
'd1_platelets_max','d1_platelets_min','d1
_potassium_max','d1_potassium_min','d1_so
dium_max',
'd1_sodium_min','d1_wbc_max','d1_wbc_min'
], axis=1)


# In[116]:


data_train_clean.head()


# In[117]:


data_train_clean.info()


# In[118]:


data_train_clean.shape


# In[119]:


for i in data_train_clean.columns:
    print(i)


# In[120]:


undf_clean=data_train_clean.isnull().sum(
)
print("\n\n", '\033[1m'+ 'Look for
undefined:' +
'\033[0m',"\n",undf_clean.sort_values())


# In[121]:


categorical_features = ["hospital_id",
"hospital_admit_source",
"icu_admit_source", "icu_id", "icu_type",

"apache_3j_bodysystem"]


# In[122]:


# Print the number of unique values for
each categorical feature
for feature in categorical_features:
```

```
    print(f"{feature}:
{data_train_clean[feature].nunique()}")


# In[123]:


print('Transform all String features to
category.\n')
   #Convert string features to category
data type and perform label encoding
for feature in categorical_features:
   # Convert the feature to string data
type
   data_train_clean[feature] =
data_train_clean[feature].astype('str')
   data_test[feature] =
data_test[feature].astype('str')
   # Fit a LabelEncoder on the unique
values of the feature from both the
training and test datasets
   le =
LabelEncoder().fit(np.unique(data_train_c
lean[feature].unique().tolist() +
data_test[feature].unique().tolist()))
   # Transform the feature values using
the fitted LabelEncoder
   data_train_clean[feature] =
le.transform(data_train_clean[feature]) +
1
   data_test[feature] =
le.transform(data_test[feature]) + 1
   # Replace NaN values with 0 and
convert the feature to integer and then
category data type
   data_train_clean[feature] =
data_train_clean[feature].fillna(0).astyp
e('int').astype('category')
   data_test[feature] =
data_test[feature].fillna(0).astype('int'
).astype('category')


# In[124]:


# Check for missing values in the
categorical features
missing_categorical =
data_train_clean[categorical_features].is
na().sum(axis=0)
print(missing_categorical)


# In[125]:


# Define lists of categorical and numeric
columns
categorical_columns = ["hospital_id",
"hospital_admit_source",
"icu_admit_source", "icu_id", "icu_type",
"apache_3j_bodysystem"]
numeric_columns =
['age','elective_surgery','albumin_apache
','apache_2_diagnosis','apache_post_opera
tive',
'arf_apache','bilirubin_apache','bun_apac
he','creatinine_apache','fio2_apache','gc
s_unable_apache','glucose_apache',
'heart_rate_apache','hematocrit_apache','
intubated_apache','map_apache','paco2_for
_ph_apache','pao2_apache','ph_apache',
'resprate_apache','sodium_apache','temp_a
pache','urineoutput_apache','ventilated_a
pache','wbc_apache','d1_diasbp_invasive_m
ax',
'd1_diasbp_invasive_min','d1_diasbp_max',
'd1_diasbp_min','d1_heartrate_max','d1_he
artrate_min','d1_mbp_invasive_max',
'd1_mbp_invasive_min','d1_mbp_max','d1_mb
p_min','d1_resprate_max','d1_resprate_min
','d1_spo2_max','d1_spo2_min',
'd1_sysbp_invasive_max','d1_sysbp_invasiv
e_min','d1_sysbp_max','d1_sysbp_min','d1_
temp_max','d1_temp_min',
'h1_diasbp_invasive_max','h1_diasbp_invas
ive_min','h1_diasbp_max','h1_diasbp_min',
'h1_heartrate_max','h1_heartrate_min',
'h1_mbp_invasive_max','h1_mbp_invasive_mi
n','h1_mbp_max','h1_mbp_min','h1_mbp_noni
nvasive_min','h1_resprate_max',
'h1_resprate_min','h1_spo2_max','h1_spo2_
min','h1_sysbp_invasive_max','h1_sysbp_in
vasive_min','h1_sysbp_max','h1_sysbp_min'
,
'h1_temp_max','h1_temp_min','h1_albumin_m
ax','h1_albumin_min','h1_bilirubin_max','
h1_bilirubin_min','h1_bun_max','h1_bun_mi
n',
'h1_calcium_max','h1_calcium_min','h1_cre
atinine_max','h1_creatinine_min','h1_gluc
ose_max','h1_glucose_min','h1_hco3_max',
'h1_hco3_min','h1_hemaglobin_max','h1_hem
aglobin_min','h1_hematocrit_max','h1_hema
tocrit_min','h1_inr_max','h1_inr_min',
'h1_lactate_max','h1_lactate_min','h1_pla
telets_max','h1_platelets_min','h1_potass
ium_max','h1_potassium_min','h1_sodium_ma
x',
'h1_sodium_min','h1_wbc_max','h1_wbc_min'
,'h1_arterial_pco2_max','h1_arterial_pco2
_min','h1_arterial_ph_max','h1_arterial_p
h_min',
'h1_arterial_po2_max','h1_arterial_po2_mi
n','h1_pao2fio2ratio_max','h1_pao2fio2rat
io_min','apache_4a_icu_death_prob','aids'
,
'cirrhosis','diabetes_mellitus','hepatic_
failure','immunosuppression','leukemia','
lymphoma','solid_tumor_with_metastasis']


# In[126]:


# Check for missing values in the numeric
columns
missing_numeric =
data_train_clean[numeric_columns].isna().
sum(axis=0)
print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())


# In[127]:
```

```
# Remove rows with all missing values
from the dataset
data_train_clean =
data_train_clean.dropna(how="all")


# In[128]:


# Get the labels from missing_numeric
that are less than 10000
labels_to_replace =
missing_numeric[missing_numeric <
10000].index.tolist()


# In[129]:


print(labels_to_replace)


# In[130]:


#Iterate over the labels to replace and
perform the replacement
for label in labels_to_replace:
    mask_new =
data_train_clean[[label]].notna().all(axi
s=1)
    data_train_clean =
data_train_clean[mask_new]


# In[131]:


# Check for missing values in the numeric
columns
missing_numeric =
data_train_clean[numeric_columns].isna().
sum(axis=0)
print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())


# In[132]:


data_train_clean.shape


# In[133]:


mask_new =
data_train_clean[['glucose_apache']].notn
a().all(axis=1)
data_train_clean =
data_train_clean[mask_new]


# In[134]:


# Check for missing values in the numeric
columns
missing_numeric =
data_train_clean[numeric_columns].isna().
sum(axis=0)
print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())


# In[135]:


data_train_clean.shape


# In[136]:


mask_new =
data_train_clean[['sodium_apache','creati
nine_apache','bun_apache']].notna().all(a
xis=1)
data_train_clean =
data_train_clean[mask_new]


# In[137]:


data_train_clean.shape


# In[138]:


# Check for missing values in the numeric
columns
missing_numeric =
data_train_clean[numeric_columns].isna().
sum(axis=0)
print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())


# In[139]:


mask_new =
data_train_clean[['hematocrit_apache','wb
c_apache']].notna().all(axis=1)
data_train_clean =
data_train_clean[mask_new]


# In[140]:


data_train_clean.shape


# In[141]:


# Check for missing values in the numeric
columns
missing_numeric =
data_train_clean[numeric_columns].isna().
sum(axis=0)
```

```
print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())


# In[142]:


labels_to_drop =
missing_numeric[missing_numeric >
9000].index.tolist()


# In[143]:


print(labels_to_drop)


# In[144]:


data_train_clean =
data_train_clean.drop(labels_to_drop,
axis=1)


# In[145]:


# Check for missing values in the numeric
columns
missing_all =
data_train_clean.isna().sum(axis=0)
print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_all.sort_values())


# In[146]:


data_train_clean.shape


# In[148]:


data_train_clean.head()


# In[147]:


plt.figure()
data_train_clean['hospital_death'].value_
counts().plot(kind='bar')
plt.show()


# In[149]:


column_names =
data_train_clean.columns.tolist()
print('Column Names in data_test_clean:')
for column_name in column_names:
    print(column_name)
```

```
# In[147]:


data_train_clean.to_csv(r'C:\Users\brian\
OneDrive\Desktop\Documente
licență\Documentație
scrisă\Date_clean.csv',index=False)
```

7.1.2. BLR model.jpybn

```
#!/usr/bin/env python

# coding: utf-8


# # BLR


# In[1]:



import numpy as np

import pandas as pd

import seaborn as sb

import matplotlib.pyplot as plt

import math as math

import sklearn as sklearn

import random as random

from sklearn.preprocessing import
StandardScaler

from sklearn.preprocessing import
LabelEncoder

from sklearn.model_selection import
train_test_split

from sklearn.neighbors import
KNeighborsClassifier

from sklearn import metrics

from sklearn.metrics import
confusion_matrix

from sklearn.model_selection import
LeaveOneOut

get_ipython().run_line_magic('matplotlib'
, 'notebook')

import seaborn as sns

from sklearn.decomposition import PCA

from sklearn.preprocessing import
StandardScaler

from sklearn.metrics.cluster import
adjusted_rand_score

from sklearn.metrics.cluster import
rand_score

from IPython.display import clear_output

from sklearn.metrics import roc_curve,
roc_auc_score, auc

from sklearn.preprocessing import
LabelEncoder

from sklearn.model_selection import
StratifiedKFold

from tensorflow import feature_column



# In[2]:



import os

import tensorflow as tf


os.environ['CUDA_VISIBLE_DEVICES'] = '-1'


if tf.test.gpu_device_name():

    print('GPU found')

else:

    print("No GPU found")



# In[3]:



from tensorflow import keras

from tensorflow.keras.callbacks import
EarlyStopping



# In[4]:



pd.set_option("display.max_columns",
None)

pd.set_option("display.max_rows", None)



# In[5]:



data_clean =
pd.read_csv('Date_clean.csv')
```

```
# In[6]:


data_clean.head()


# # BLR model


# In[7]:


X_BLR = data_clean.drop('hospital_death',
axis=1)

y_BLR = data_clean['hospital_death']


# In[8]:


import imblearn

from collections import Counter

from imblearn.combine import SMOTEENN

SMOTEENN = SMOTEENN()


print('Original dataset shape %s' %
Counter(y_BLR))


X_res, y_res =
SMOTEENN.fit_resample(X_BLR, y_BLR)


print('After undersample dataset shape
%s' % Counter(y_res))


# In[9]:


# Split the data into training and
testing sets

X_train_BLR, X_test_BLR, y_train_BLR,
y_test_BLR = train_test_split(X_res,
y_res, test_size=0.3, random_state=42)


# In[10]:


scaler = StandardScaler()

X_train_BLR =
scaler.fit_transform(X_train_BLR)

X_test_BLR = scaler.transform(X_test_BLR)


# In[11]:


#from sklearn.linear_model import
LogisticRegression

from sklearn.metrics import
accuracy_score, confusion_matrix


from sklearn.linear_model import
LogisticRegressionCV


# Create and train the logistic
regression model with elastic net
regularization

logreg =
LogisticRegressionCV(penalty='elasticnet'
, solver='saga', l1_ratios=[0.5])

#SAGA (Stochastic Average Gradient
Descent) extends Gradient Descent
algorithm for large-scale machine
learning problems.

logreg.fit(X_train_BLR, y_train_BLR)


# In[12]:


# Make predictions on the test set

y_pred_BLR = logreg.predict(X_test_BLR)


# In[13]:
```

```
# Calculate accuracy

accuracy = accuracy_score(y_test_BLR,
y_pred_BLR)

print('Accuracy:', accuracy)



# In[14]:



# Create a confusion matrix

confusion_mat =
confusion_matrix(y_test_BLR, y_pred_BLR)

# Plot the confusion matrix using seaborn

plt.figure()

sns.heatmap(confusion_mat, annot=True,
cmap='Blues', fmt='d')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.show()



# In[15]:



observed_deaths = confusion_mat[1, 1]  #
True positives

expected_deaths = y_test_BLR.sum()  # Sum
of true positives and false negatives

smr = observed_deaths / expected_deaths

print('Standardized Mortality Ratio
(SMR):', smr)



# In[16]:



# Predict probabilities for the test set

y_pred_prob_BLR =
logreg.predict(X_test_BLR)



# Compute the false positive rate, true
positive rate, and thresholds

fpr, tpr, thresholds =
roc_curve(y_test_BLR, y_pred_prob_BLR)


# Compute the AUROC score

roc_auc = roc_auc_score(y_test_BLR,
y_pred_prob_BLR)



# In[17]:



# Plot the ROC curve

plt.figure()

plt.plot(fpr, tpr, label='AUROC = %0.3f'
% roc_auc)

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating
Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()



# # DEPLOY


# In[56]:



data_test_clean =
pd.read_csv('Date_test_clean.csv')



# In[57]:



# Load the "data_test_clean" dataset

X_test_clean = data_test_clean  #
Assuming "data_test_clean" doesn't have
the 'hospital_death' column
```

```
# Standardize the preprocessed
"data_test_clean" dataset using the same
scaler used for training data

X_test_clean_scaled =
scaler.fit_transform(X_test_clean)
```

```
# In[58]:
```

```
# Make predictions on the standardized
"data_test_clean" dataset

y_pred_clean =
logreg.predict(X_test_clean_scaled)
```

```
# Select the indices of the patients you
want to include

selected_patient_indices = [0, 10, 150,
230, 500]  # Example indices
```

```
# Make predictions on the standardized
"data_test_clean" dataset

y_pred_prob_BLR =
logreg.predict_proba(X_test_clean_scaled)
[:, 1]
```

```
# Iterate over the selected patient
indices and print the individual
prediction and confidence for each
patient

for patient_index in
selected_patient_indices:

    individual_prediction =
int(round(y_pred_prob_BLR[patient_index])
)

    individual_confidence =
np.abs(y_pred_prob_BLR[patient_index] -
0.5)  # Assuming the prediction is
between 0 and 1


    print(f"Patient Index:
{patient_index} - Prediction:
{individual_prediction} - Confidence:
{individual_confidence}")
```

```
# # Electro BLR
```

```
# In[20]:
```

```
electro_data_clean =
pd.read_csv('electro_data_clean.csv')
```

```
# In[21]:
```

```
electro_X_BLR =
electro_data_clean.drop('hospital_death',
axis=1)

electro_y_BLR =
electro_data_clean['hospital_death']
```

```
# In[22]:
```

```
#SMOTEENN = SMOTEENN()


print('Original dataset shape %s' %
Counter(electro_y_BLR))


electro_X_res, electro_y_res =
SMOTEENN.fit_resample(electro_X_BLR,
electro_y_BLR)


print('After undersample dataset shape
%s' % Counter(electro_y_res))
```

```
# In[23]:
```

```
# Split the data into training and
testing sets

electro_X_train_BLR, electro_X_test_BLR,
electro_y_train_BLR, electro_y_test_BLR =
train_test_split(electro_X_res,
electro_y_res, test_size=0.3,
random_state=42)
```

```
# In[24]:



electro_X_train_BLR =
scaler.fit_transform(electro_X_train_BLR)

electro_X_test_BLR =
scaler.transform(electro_X_test_BLR)



# In[25]:



electro_logreg =
LogisticRegressionCV(penalty='elasticnet'
, solver='saga', l1_ratios=[0.3])

electro_logreg.fit(electro_X_train_BLR,
electro_y_train_BLR)



# In[26]:



# Make predictions on the test set

electro_y_pred_BLR =
electro_logreg.predict(electro_X_test_BLR
)



# In[27]:



# Calculate accuracy

electro_accuracy =
accuracy_score(electro_y_test_BLR,
electro_y_pred_BLR)

print('Accuracy:', electro_accuracy)



# In[28]:



# Create a confusion matrix

electro_confusion_mat =
confusion_matrix(electro_y_test_BLR,
electro_y_pred_BLR)

# Plot the confusion matrix using seaborn

plt.figure()

sns.heatmap(electro_confusion_mat,
annot=True, cmap='Blues', fmt='d')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.title('Confusion Matrix')

plt.show()



# In[29]:



electro_observed_deaths =
electro_confusion_mat[1, 1]  # True
positives

electro_expected_deaths =
electro_y_test_BLR.sum()  # Sum of true
positives and false negatives

electro_smr = electro_observed_deaths /
electro_expected_deaths

print('Standardized Mortality Ratio
(SMR):', electro_smr)



# In[30]:



# Predict probabilities for the test set

electro_y_pred_prob_BLR =
electro_logreg.predict(electro_X_test_BLR
)



# Compute the false positive rate, true
positive rate, and thresholds

electro_fpr, electro_tpr,
electro_thresholds =
roc_curve(electro_y_test_BLR,
electro_y_pred_prob_BLR)



# Compute the AUROC score

electro_roc_auc =
roc_auc_score(electro_y_test_BLR,
electro_y_pred_prob_BLR)
```

```
# In[31]:


# Plot the ROC curve
plt.figure()
plt.plot(electro_fpr, electro_tpr,
label='AUROC = %0.3f' % electro_roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating
Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()


# In[32]:


# Plot a histogram of the predicted
probabilities
plt.figure()
plt.hist(electro_y_pred_prob_BLR,
bins=20)
plt.xlabel('Predicted Probability')
plt.ylabel('Frequency')
plt.title('Histogram of Predicted
Probabilities')
plt.show()


# # DEPLOY


# In[59]:


electro_data_test_clean =
pd.read_csv('electro_Date_test_clean.csv'
)


# In[60]:


#electro_data_test_clean.shape


# In[61]:


# Load the "data_test_clean" dataset
electro_X_test_clean =
electro_data_test_clean  # Assuming
"data_test_clean" doesn't have the
'hospital_death' column

# Standardize the preprocessed
"data_test_clean" dataset using the same
scaler used for training data
electro_X_test_clean_scaled =
scaler.fit_transform(electro_X_test_clean
)


# In[62]:


# Make predictions on the standardized
"data_test_clean" dataset
electro_y_pred_clean =
electro_logreg.predict(electro_X_test_cle
an_scaled)

# Select the indices of the patients you
want to include
electro_selected_patient_indices = [0,
10, 150, 230, 500]  # Example indices

# Make predictions on the standardized
"data_test_clean" dataset
electro_y_pred_prob_BLR =
electro_logreg.predict_proba(electro_X_te
st_clean_scaled)[:, 1]

# Iterate over the selected patient
indices and print the individual
prediction and confidence for each
patient
```

```
for patient_index in
electro_selected_patient_indices:

    electro_individual_prediction =
int(round(electro_y_pred_prob_BLR[patient
_index]))

    electro_individual_confidence =
np.abs(electro_y_pred_prob_BLR[patient_in
dex] - 0.5)  # Assuming the prediction is
between 0 and 1


    print(f"Patient Index:
{patient_index} - Prediction:
{electro_individual_prediction} -
Confidence:
{electro_individual_confidence}")
```

7.1.3. MLP_model.jpynb

```
#!/usr/bin/env python

# coding: utf-8


# # MLP


# In[1]:



import numpy as np

import pandas as pd

import seaborn as sb

import matplotlib.pyplot as plt

import math as math

import sklearn as sklearn

import random as random

from sklearn.preprocessing import
StandardScaler

from sklearn.preprocessing import
LabelEncoder

from sklearn.model_selection import
train_test_split

from sklearn.neighbors import
KNeighborsClassifier

from sklearn import metrics

from sklearn.metrics import
confusion_matrix

from sklearn.model_selection import
LeaveOneOut

get_ipython().run_line_magic('matplotlib'
, 'notebook')

import seaborn as sns

from sklearn.decomposition import PCA

from sklearn.preprocessing import
StandardScaler

from sklearn.metrics.cluster import
adjusted_rand_score

from sklearn.metrics.cluster import
rand_score

from IPython.display import clear_output

from sklearn.metrics import roc_curve,
roc_auc_score, auc

from sklearn.preprocessing import
LabelEncoder

from sklearn.model_selection import
StratifiedKFold

from tensorflow import feature_column



# In[2]:



import os

import tensorflow as tf


os.environ['CUDA_VISIBLE_DEVICES'] = '-1'


if tf.test.gpu_device_name():

    print('GPU found')

else:

    print("No GPU found")



# In[3]:



from tensorflow import keras

from tensorflow.keras.callbacks import
EarlyStopping



# In[4]:



pd.set_option("display.max_columns",
None)

pd.set_option("display.max_rows", None)



# In[5]:



#!pip install imblearn
```

```
# In[6]:


import imblearn

from collections import Counter


# In[7]:


data_clean =
pd.read_csv('Date_clean.csv')


# In[8]:


data_clean.head()


# # MLP

# In[9]:


X_ANN = data_clean.drop('hospital_death',
axis=1)

y_ANN = data_clean['hospital_death']


# In[10]:


y_ANN.value_counts()


# In[11]:


# from imblearn.over_sampling import
RandomOverSampler

# ROS = RandomOverSampler()


# print('Original dataset shape %s' %
Counter(y_ANN))


# X_res, y_res = ROS.fit_resample(X_ANN,
y_ANN)


# print('After undersample dataset shape
%s' % Counter(y_res))


# In[12]:


from imblearn.combine import SMOTEENN

SMOTEENN = SMOTEENN()


print('Original dataset shape %s' %
Counter(y_ANN))


X_res, y_res =
SMOTEENN.fit_resample(X_ANN, y_ANN)


print('After undersample dataset shape
%s' % Counter(y_res))


# In[13]:


X_train_ANN, X_test_ANN, y_train_ANN,
y_test_ANN = train_test_split(X_res,
y_res, test_size=0.3, random_state=42)


# In[14]:


scaler = StandardScaler()

X_train_ANN =
scaler.fit_transform(X_train_ANN)

X_test_ANN = scaler.transform(X_test_ANN)
```

```
# In[15]:


model = keras.Sequential([
    keras.layers.Dense(1500,
activation='relu', input_shape=(71,)),
    keras.layers.Dense(900,
activation='elu'),
    keras.layers.Dense(850,
activation='elu'),
    keras.layers.Dense(1,
activation='sigmoid')
])


# In[16]:


model.compile(optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy'])


# In[17]:


early_stopping =
EarlyStopping(monitor='val_loss',
patience=5, restore_best_weights=True)


# In[18]:


history = model.fit(X_train_ANN,
y_train_ANN, epochs=50, batch_size=50,
validation_split=0.2,
callbacks=[early_stopping])


# In[19]:


test_loss, test_acc =
model.evaluate(X_test_ANN, y_test_ANN)
print('Test accuracy:', test_acc)


# In[20]:


# Predict probabilities for the test set
y_pred_prob_ANN =
model.predict(X_test_ANN)

# Compute the false positive rate, true
positive rate, and thresholds
fpr, tpr, thresholds =
roc_curve(y_test_ANN, y_pred_prob_ANN)

# Compute the AUROC score
roc_auc = roc_auc_score(y_test_ANN,
y_pred_prob_ANN)


# In[21]:


# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, label='AUROC = %0.3f'
% roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating
Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()


# In[22]:
```

```
from sklearn.metrics import
confusion_matrix

# Predict classes for the test set

y_pred_prob_ANN =
model.predict(X_test_ANN)



# Round the probabilities to obtain the
predicted class labels

y_pred_ANN =
y_pred_prob_ANN.round().astype(int)



# Create a confusion matrix

cm = confusion_matrix(y_test_ANN,
y_pred_ANN)



# Plot the confusion matrix using a
heatmap

plt.figure()

sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()



# In[23]:



observed_deaths = cm[1, 1]  # True
positives

expected_deaths = y_test_ANN.sum()  # Sum
of true positives and false negatives

smr = observed_deaths / expected_deaths

print('Standardized Mortality Ratio
(SMR):', smr)



# In[24]:



# # Convert y_test_ANN numpy array to a
pandas Series

# y_test_ANN_series =
pd.Series(y_test_ANN, name='True_Label')



# # Save the true labels to a file

# output_file = 'true_labels_output.csv'

# y_test_ANN_series.to_csv(output_file,
index=False)



# In[25]:



# data_y_pred =
pd.read_csv('true_labels_output.csv')



# In[26]:



# data_y_pred.value_counts()



# In[27]:



# data_y_pred.head()



# In[28]:



# Calculate the variance of predicted
probabilities

variance = np.var(y_pred_prob_ANN)



# Print the variance

print("Variance of predicted
probabilities:", variance)



# In[29]:
```

```
# Plot a histogram of the predicted
probabilities

plt.figure()

plt.hist(y_pred_prob_ANN, bins=20)

plt.xlabel('Predicted Probability')

plt.ylabel('Frequency')

plt.title('Histogram of Predicted
Probabilities')

plt.show()



# # DEPLOY


# In[69]:



data_test_clean =
pd.read_csv('Date_test_clean.csv')



# In[70]:




# Load the "data_test_clean" dataset

X_test_clean = data_test_clean  #
Assuming "data_test_clean" doesn't have
the 'hospital_death' column


# Standardize the preprocessed
"data_test_clean" dataset using the same
scaler used for training data

X_test_clean_scaled =
scaler.fit_transform(X_test_clean)



# In[72]:



# Make predictions on the standardized
"data_test_clean" dataset

y_pred_prob_ANN =
model.predict(X_test_clean_scaled)


# Retrieve the prediction and confidence
for one patient (e.g., patient at index
0)

# Select the indices of the patients you
want to include

selected_patient_indices = [0, 10, 150,
230, 500]  # Example indices


# Iterate over the selected patient
indices and print the individual
prediction and confidence for each
patient

for patient_index in
selected_patient_indices:

    individual_prediction =
int(round(y_pred_prob_ANN[patient_index][
0]))

    individual_confidence =
np.abs(y_pred_prob_ANN[patient_index][0]
- 0.5)  # Assuming the prediction is
between 0 and 1


    print(f"Patient Index:
{patient_index} - Prediction:
{individual_prediction} - Confidence:
{individual_confidence}")



# # Electro MLP


# In[30]:



electro_data_clean=data_clean.drop([

'bun_apache','creatinine_apache','hematoc
rit_apache','sodium_apache',

'wbc_apache',

'apache_4a_icu_death_prob'], axis=1)



# In[31]:



electro_data_clean.head()
```

```
# In[32]:


output_file = 'electro_data_clean.csv'

electro_data_clean.to_csv(output_file,
index=False)


# In[33]:


X_ANN_electro =
electro_data_clean.drop('hospital_death',
axis=1)

y_ANN_electro =
electro_data_clean['hospital_death']


# In[34]:


from imblearn.combine import SMOTEENN

SMOTEENN = SMOTEENN()


print('Original dataset shape %s' %
Counter(y_ANN_electro))


electro_X_res, electro_y_res =
SMOTEENN.fit_resample(X_ANN_electro,
y_ANN_electro)


print('After undersample dataset shape
%s' % Counter(electro_y_res))


# In[35]:


electro_X_train_ANN, electro_X_test_ANN,
electro_y_train_ANN, electro_y_test_ANN =
train_test_split(electro_X_res,
electro_y_res, test_size=0.3,
random_state=42)


# In[36]:


electro_X_train_ANN =
scaler.fit_transform(electro_X_train_ANN)

electro_X_test_ANN =
scaler.transform(electro_X_test_ANN)


# In[37]:


electro_model = keras.Sequential([

    keras.layers.Dense(1500,
activation='relu', input_shape=(65,)),

    keras.layers.Dense(900,
activation='elu'),

    keras.layers.Dense(850,
activation='elu'),

    keras.layers.Dense(1,
activation='sigmoid')

])


# In[38]:


electro_model.compile(optimizer='adam',
loss='binary_crossentropy',
metrics=['accuracy'])


# In[39]:


history =
electro_model.fit(electro_X_train_ANN,
electro_y_train_ANN, epochs=50,
batch_size=50, validation_split=0.2,
callbacks=[early_stopping])


# In[40]:
```

```
test_loss, test_acc =
electro_model.evaluate(electro_X_test_ANN
, electro_y_test_ANN)

print('Test accuracy:', test_acc)
```

```
# In[41]:
```

```
# Predict probabilities for the test set

electro_y_pred_prob_ANN =
electro_model.predict(electro_X_test_ANN)


# Compute the false positive rate, true
positive rate, and thresholds

electro_fpr, electro_tpr,
electro_thresholds =
roc_curve(electro_y_test_ANN,
electro_y_pred_prob_ANN)


# Compute the AUROC score

electro_roc_auc =
roc_auc_score(electro_y_test_ANN,
electro_y_pred_prob_ANN)
```

```
# In[47]:
```

```
# Plot the ROC curve

plt.figure()

plt.plot(electro_fpr, electro_tpr,
label='AUROC = %0.3f' % electro_roc_auc)

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating
Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()
```

```
# In[48]:
```

```
# Predict classes for the test set

electro_y_pred_prob_ANN =
electro_model.predict(electro_X_test_ANN)


# Round the probabilities to obtain the
predicted class labels

electro_y_pred_ANN =
electro_y_pred_prob_ANN.round().astype(in
t)


# Create a confusion matrix

electro_cm =
confusion_matrix(electro_y_test_ANN,
electro_y_pred_ANN)


# Plot the confusion matrix using a
heatmap

plt.figure()

sns.heatmap(electro_cm, annot=True,
fmt='d', cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.show()
```

```
# In[49]:
```

```
electro_observed_deaths = electro_cm[1,
1]  # True positives

electro_expected_deaths =
electro_y_test_ANN.sum()  # Sum of true
positives and false negatives

electro_smr = electro_observed_deaths /
electro_expected_deaths

print('Standardized Mortality Ratio
(SMR):', electro_smr)
```

```
# In[45]:
```

```
# Calculate the variance of predicted
probabilities

electro_variance =
np.var(electro_y_pred_prob_ANN)



# Print the variance

print("Variance of predicted
probabilities:", electro_variance)



# In[50]:



# Plot a histogram of the predicted
probabilities

plt.figure()

plt.hist(electro_y_pred_prob_ANN,
bins=20)

plt.xlabel('Predicted Probability')

plt.ylabel('Frequency')

plt.title('Histogram of Predicted
Probabilities')

plt.show()



# # DEPLOY


# In[73]:



electro_data_test_clean =
pd.read_csv('electro_Date_test_clean.csv'
)



# In[74]:



# Load the "data_test_clean" dataset

electro_X_test_clean =
electro_data_test_clean  # Assuming
"data_test_clean" doesn't have the
'hospital_death' column


# Standardize the preprocessed
"data_test_clean" dataset using the same
scaler used for training data

electro_X_test_clean_scaled =
scaler.fit_transform(electro_X_test_clean
)



# In[76]:



# Make predictions on the standardized
"data_test_clean" dataset

electro_y_pred_prob_ANN =
electro_model.predict(electro_X_test_clea
n_scaled)


electro_selected_patient_indices = [0,
10, 150, 230, 500]  # Example indices


# Iterate over the selected patient
indices and print the individual
prediction and confidence for each
patient

for patient_index in
electro_selected_patient_indices:

    electro_individual_prediction =
int(round(electro_y_pred_prob_ANN[patient
_index][0]))

    electro_individual_confidence =
np.abs(electro_y_pred_prob_ANN[patient_in
dex][0] - 0.5)  # Assuming the prediction
is between 0 and 1


    print(f"Patient Index:
{patient_index} - Prediction:
{electro_individual_prediction} -
Confidence:
{electro_individual_confidence}")
```

7.1.4. Test_data_cleanup.jpynb

```
#!/usr/bin/env python

# coding: utf-8


# # Test data cleanup


# In[1]:



import numpy as np

import pandas as pd

import seaborn as sb

import matplotlib.pyplot as plt

import math as math

import sklearn as sklearn

import random as random

from sklearn.preprocessing import
StandardScaler

from sklearn.preprocessing import
LabelEncoder

from sklearn.model_selection import
train_test_split

from sklearn.neighbors import
KNeighborsClassifier

from sklearn import metrics

from sklearn.metrics import
confusion_matrix

from sklearn.model_selection import
LeaveOneOut

get_ipython().run_line_magic('matplotlib'
, 'notebook')

import seaborn as sns

from sklearn.decomposition import PCA

from sklearn.preprocessing import
StandardScaler

from sklearn.metrics.cluster import
adjusted_rand_score

from sklearn.metrics.cluster import
rand_score

from IPython.display import clear_output

from sklearn.metrics import roc_curve,
roc_auc_score, auc

from sklearn.preprocessing import
LabelEncoder

from sklearn.model_selection import
StratifiedKFold

from tensorflow import feature_column



# In[2]:



import os

import tensorflow as tf


os.environ['CUDA_VISIBLE_DEVICES'] = '-1'


if tf.test.gpu_device_name():

    print('GPU found')

else:

    print("No GPU found")



# In[3]:



from tensorflow import keras

from tensorflow.keras.callbacks import
EarlyStopping



# In[4]:



pd.set_option("display.max_columns",
None)

pd.set_option("display.max_rows", None)



# In[277]:



data_test = pd.read_csv('unlabeled.csv')

data_test.head()
```

```
# In[278]:


data_test_clean=data_test.drop("readmissi
on_status", axis='columns') #all of them
are nan values so we drop


# In[279]:


data_test_clean=data_test_clean.drop("gen
der", axis='columns') #there is no real
difference between male and female

#death correlation so it doesn't
influence =» we drop


# In[280]:


data_test_clean=data_test_clean.drop("eth
nicity", axis='columns')#there is no real
difference between ethnicities

#death correlation so it doesn't
influence =» we drop


# In[281]:


data_test_clean=data_test_clean.drop("pat
ient_id", axis='columns')


# In[282]:


data_test_clean=data_test_clean.drop("enc
ounter_id", axis='columns')


# In[283]:


data_test_clean=data_test_clean.drop("bmi
", axis='columns') #even if the first kde
seems to give us a lot of difference,

#in a normalised form we can see that no
matter the bmi, dead vs. alive is kind of
the same. =»


# In[284]:


data_test_clean=data_test_clean.drop("wei
ght", axis='columns')


# In[285]:


data_test_clean=data_test_clean.drop("hei
ght", axis='columns')


# In[286]:


data_test_clean=data_test_clean.drop("icu
_stay_type", axis='columns') #the ICU
type is more relevant than the stay type
bc it

#is more varied


# In[287]:


data_test_clean=data_test_clean.drop("pre
_icu_los_days", axis='columns') #not
really enough difference


# In[288]:
```

```
data_test_clean=data_test_clean.drop("apa
che_2_bodysystem", axis='columns') #they
are kind of the same as apache 3
```

```
# In[289]:
```

```
data_test_clean=data_test_clean.drop("apa
che_4a_hospital_death_prob",
axis='columns')
```

```
# In[290]:
```

```
mask = data_test_clean[['aids',
'leukemia',
'cirrhosis','diabetes_mellitus',

'hepatic_failure','immunosuppression','ly
mphoma',

'solid_tumor_with_metastasis']].notna().a
ll(axis=1)

data_test_clean = data_test_clean[mask]

#bc .dropna makes it so that I can't use
.head() or .shape
```

```
# In[291]:
```

```
data_test_clean=data_test_clean.drop(["d1
_arterial_pco2_max","d1_arterial_pco2_min
","d1_arterial_ph_max",

"d1_arterial_ph_min","d1_arterial_po2_max
","d1_arterial_po2_min",

"d1_pao2fio2ratio_max","d1_pao2fio2ratio_
min"],axis=1)
```

```
# In[292]:
```

```
data_test_clean=data_test_clean.drop(["ap
ache_3j_diagnosis","gcs_eyes_apache","gcs
_motor_apache",

"gcs_verbal_apache","paco2_apache"],axis=
1)
```

```
# In[293]:
```

```
data_test_clean['d1_mbp_invasive_max']=da
ta_test_clean['d1_mbp_invasive_max'].fill
na(data_test_clean['d1_mbp_noninvasive_ma
x'])
```

```
# In[294]:
```

```
data_test_clean=data_test_clean.drop("d1_
mbp_noninvasive_max", axis='columns')
```

```
# In[295]:
```

```
data_test_clean['d1_diasbp_invasive_max']
=data_test_clean['d1_diasbp_invasive_max'
].fillna(data_test_clean['d1_diasbp_nonin
vasive_max'])
```

```
# In[296]:
```

```
data_test_clean=data_test_clean.drop("d1_
diasbp_noninvasive_max", axis='columns')
```

```
# In[297]:
```

```
data_test_clean['d1_diasbp_invasive_min']
=data_test_clean['d1_diasbp_invasive_min'
].fillna(data_test_clean['d1_diasbp_nonin
vasive_min'])
```

```
# In[298]:
```

```
data_test_clean=data_test_clean.drop("d1_
diasbp_noninvasive_min", axis='columns')
```

```
# In[299]:
```

```
data_test_clean['d1_mbp_invasive_min']=da
ta_test_clean['d1_mbp_invasive_min'].fill
na(data_test_clean['d1_mbp_noninvasive_mi
n'])
```

```
# In[300]:
```

```
data_test_clean=data_test_clean.drop("d1_
mbp_noninvasive_min", axis='columns')
```

```
# In[301]:
```

```
data_test_clean['d1_sysbp_invasive_max']=
data_test_clean['d1_sysbp_invasive_max'].
fillna(data_test_clean['d1_sysbp_noninvas
ive_max'])
```

```
# In[302]:
```

```
data_test_clean=data_test_clean.drop("d1_
sysbp_noninvasive_max", axis='columns')
```

```
# In[303]:
```

```
data_test_clean['d1_sysbp_invasive_min']=
data_test_clean['d1_sysbp_invasive_min'].
fillna(data_test_clean['d1_sysbp_noninvas
ive_min'])
```

```
# In[304]:
```

```
data_test_clean=data_test_clean.drop("d1_
sysbp_noninvasive_min", axis='columns')
```

```
# In[305]:
```

```
data_test_clean['h1_diasbp_invasive_max']
=data_test_clean['h1_diasbp_invasive_max'
].fillna(data_test_clean['h1_diasbp_nonin
vasive_max'])
```

```
# In[306]:
```

```
data_test_clean=data_test_clean.drop("h1_
diasbp_noninvasive_max", axis='columns')
```

```
# In[307]:
```

```
data_test_clean['h1_diasbp_invasive_min']
=data_test_clean['h1_diasbp_invasive_min'
].fillna(data_test_clean['h1_diasbp_nonin
vasive_min'])
```

```
# In[308]:
```

```
data_test_clean=data_test_clean.drop("h1_
diasbp_noninvasive_min", axis='columns')
```

```
# In[309]:
```

```
data_test_clean['h1_mbp_invasive_max']=da
ta_test_clean['h1_mbp_invasive_max'].fill
na(data_test_clean['h1_mbp_noninvasive_ma
x'])
```

```
# In[310]:
```

```
data_test_clean=data_test_clean.drop("h1_
mbp_noninvasive_max", axis='columns')
```

```
# In[311]:
```

```
data_test_clean['h1_sysbp_invasive_max']=
data_test_clean['h1_sysbp_invasive_max'].
fillna(data_test_clean['h1_sysbp_noninvas
ive_max'])
```

```
# In[312]:
```

```
data_test_clean=data_test_clean.drop("h1_
sysbp_noninvasive_max", axis='columns')
```

```
# In[313]:
```

```
data_test_clean['h1_sysbp_invasive_min']=
data_test_clean['h1_sysbp_invasive_min'].
fillna(data_test_clean['h1_sysbp_noninvas
ive_min'])
```

```
# In[314]:
```

```
data_test_clean=data_test_clean.drop("h1_
sysbp_noninvasive_min", axis='columns')
```

```
# In[315]:
```

```
data_test_clean=data_test_clean.drop(['d1
_albumin_max','d1_albumin_min','d1_biliru
bin_max','d1_bilirubin_min',
'd1_bun_max','d1_bun_min','d1_calcium_max
','d1_calcium_min','d1_creatinine_max','d
1_creatinine_min',
'd1_glucose_max','d1_glucose_min','d1_hco
3_max','d1_hco3_min','d1_hemaglobin_max',
'd1_hemaglobin_min',
'd1_hematocrit_max','d1_hematocrit_min','
d1_inr_max','d1_inr_min','d1_lactate_max'
,'d1_lactate_min',
'd1_platelets_max','d1_platelets_min','d1
_potassium_max','d1_potassium_min','d1_so
dium_max',
'd1_sodium_min','d1_wbc_max','d1_wbc_min'
], axis=1)
```

```
# In[316]:
```

```
data_test_clean.shape
```

```
# In[317]:
```

```
data_test_clean.head()
```

```
# In[318]:
```

```
categorical_features = ["hospital_id",
"hospital_admit_source",
"icu_admit_source", "icu_id", "icu_type",


"apache_3j_bodysystem"]



# In[319]:



print('Transform all String features to
category.\n')

   #Convert string features to category
data type and perform label encoding

for feature in categorical_features:

   # Convert the feature to string data
type

   data_test_clean[feature] =
data_test_clean[feature].astype('str')

   data_test[feature] =
data_test[feature].astype('str')

   # Fit a LabelEncoder on the unique
values of the feature from both the
training and test datasets

   le =
LabelEncoder().fit(np.unique(data_test_cl
ean[feature].unique().tolist() +
data_test[feature].unique().tolist()))

   # Transform the feature values using
the fitted LabelEncoder

   data_test_clean[feature] =
le.transform(data_test_clean[feature]) +
1

   data_test[feature] =
le.transform(data_test[feature]) + 1

   # Replace NaN values with 0 and
convert the feature to integer and then
category data type

   data_test_clean[feature] =
data_test_clean[feature].fillna(0).astype
('int').astype('category')

   data_test[feature] =
data_test[feature].fillna(0).astype('int'
).astype('category')



# In[320]:
```

```
# Define lists of categorical and numeric
columns

categorical_columns = ["hospital_id",
"hospital_admit_source",
"icu_admit_source", "icu_id", "icu_type",
"apache_3j_bodysystem"]

numeric_columns =
['age','elective_surgery','albumin_apache
','apache_2_diagnosis','apache_post_opera
tive',

'arf_apache','bilirubin_apache','bun_apac
he','creatinine_apache','fio2_apache','gc
s_unable_apache','glucose_apache',

'heart_rate_apache','hematocrit_apache','
intubated_apache','map_apache','paco2_for
_ph_apache','pao2_apache','ph_apache',

'resprate_apache','sodium_apache','temp_a
pache','urineoutput_apache','ventilated_a
pache','wbc_apache','d1_diasbp_invasive_m
ax',

'd1_diasbp_invasive_min','d1_diasbp_max',
'd1_diasbp_min','d1_heartrate_max','d1_he
artrate_min','d1_mbp_invasive_max',

'd1_mbp_invasive_min','d1_mbp_max','d1_mb
p_min','d1_resprate_max','d1_resprate_min
','d1_spo2_max','d1_spo2_min',

'd1_sysbp_invasive_max','d1_sysbp_invasiv
e_min','d1_sysbp_max','d1_sysbp_min','d1_
temp_max','d1_temp_min',

'h1_diasbp_invasive_max','h1_diasbp_invas
ive_min','h1_diasbp_max','h1_diasbp_min',
'h1_heartrate_max','h1_heartrate_min',

'h1_mbp_invasive_max','h1_mbp_invasive_mi
n','h1_mbp_max','h1_mbp_min','h1_mbp_noni
nvasive_min','h1_resprate_max',

'h1_resprate_min','h1_spo2_max','h1_spo2_
min','h1_sysbp_invasive_max','h1_sysbp_in
vasive_min','h1_sysbp_max','h1_sysbp_min'
,

'h1_temp_max','h1_temp_min','h1_albumin_m
ax','h1_albumin_min','h1_bilirubin_max','
h1_bilirubin_min','h1_bun_max','h1_bun_mi
n',

'h1_calcium_max','h1_calcium_min','h1_cre
atinine_max','h1_creatinine_min','h1_gluc
ose_max','h1_glucose_min','h1_hco3_max',

'h1_hco3_min','h1_hemaglobin_max','h1_hem
aglobin_min','h1_hematocrit_max','h1_hema
tocrit_min','h1_inr_max','h1_inr_min',

'h1_lactate_max','h1_lactate_min','h1_pla
telets_max','h1_platelets_min','h1_potass
ium_max','h1_potassium_min','h1_sodium_ma
x',

'h1_sodium_min','h1_wbc_max','h1_wbc_min'
,'h1_arterial_pco2_max','h1_arterial_pco2
_min','h1_arterial_ph_max','h1_arterial_p
h_min',
```

```
'h1_arterial_po2_max','h1_arterial_po2_mi
n','h1_pao2fio2ratio_max','h1_pao2fio2rat
io_min','apache_4a_icu_death_prob','aids'
,

'cirrhosis','diabetes_mellitus','hepatic_
failure','immunosuppression','leukemia','
lymphoma','solid_tumor_with_metastasis']



# In[321]:



# Remove rows with all missing values
from the dataset

data_test_clean =
data_test_clean.dropna(how="all")



# In[322]:



# Check for missing values in the numeric
columns

missing_numeric =
data_test_clean[numeric_columns].isna().s
um(axis=0)

print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())



# In[323]:



data_test_clean.head()



# In[324]:



labels_to_replace=['age',
'elective_surgery', 'apache_2_diagnosis',
'apache_post_operative',


'arf_apache', 'gcs_unable_apache',
'heart_rate_apache', 'intubated_apache',
'map_apache', 'resprate_apache',
'temp_apache', 'ventilated_apache',


'd1_diasbp_invasive_max',
'd1_diasbp_invasive_min',
'd1_diasbp_max', 'd1_diasbp_min',


'd1_heartrate_max', 'd1_heartrate_min',
'd1_mbp_invasive_max',
'd1_mbp_invasive_min',


'd1_mbp_max', 'd1_mbp_min',
'd1_resprate_max', 'd1_resprate_min',
'd1_spo2_max',


'd1_spo2_min', 'd1_sysbp_invasive_max',
'd1_sysbp_invasive_min', 'd1_sysbp_max',


'd1_sysbp_min', 'd1_temp_max',
'd1_temp_min', 'h1_diasbp_invasive_max',


'h1_diasbp_invasive_min',
'h1_diasbp_max', 'h1_diasbp_min',
'h1_heartrate_max',


'h1_heartrate_min',
'h1_mbp_invasive_max', 'h1_mbp_max',
'h1_mbp_min',


'h1_mbp_noninvasive_min',
'h1_resprate_max', 'h1_resprate_min',
'h1_spo2_max',


'h1_spo2_min', 'h1_sysbp_invasive_max',
'h1_sysbp_invasive_min', 'h1_sysbp_max',


'h1_sysbp_min',
'apache_4a_icu_death_prob', 'aids',
'cirrhosis', 'diabetes_mellitus',


'hepatic_failure', 'immunosuppression',
'leukemia', 'lymphoma',


'solid_tumor_with_metastasis']



# In[325]:



#Iterate over the labels to replace and
perform the replacement

for label in labels_to_replace:
```

```
    mask_new =
data_test_clean[[label]].notna().all(axis
=1)

    data_test_clean =
data_test_clean[mask_new]
```

```
# In[326]:
```

```
# Check for missing values in the numeric
columns

missing_numeric =
data_test_clean[numeric_columns].isna().s
um(axis=0)

print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())
```

```
# In[327]:
```

```
mask_new =
data_test_clean[['glucose_apache']].notna
().all(axis=1)

data_test_clean =
data_test_clean[mask_new]
```

```
# In[328]:
```

```
mask_new =
data_test_clean[['sodium_apache','creatin
ine_apache','bun_apache']].notna().all(ax
is=1)

data_test_clean =
data_test_clean[mask_new]
```

```
# In[329]:
```

```
mask_new =
data_test_clean[['hematocrit_apache','wbc
_apache']].notna().all(axis=1)

data_test_clean =
data_test_clean[mask_new]
```

```
# In[330]:
```

```
# Check for missing values in the numeric
columns

missing_numeric =
data_test_clean[numeric_columns].isna().s
um(axis=0)

print("\n\n", '\033[1m'+ 'Look for
missing:' +
'\033[0m',"\n",missing_numeric.sort_value
s())
```

```
# In[331]:
```

```
data_test_clean=data_test_clean.drop(['al
bumin_apache', 'bilirubin_apache',
'fio2_apache', 'paco2_for_ph_apache',

'pao2_apache', 'ph_apache',
'urineoutput_apache',
'h1_mbp_invasive_min',

'h1_temp_max', 'h1_temp_min',
'h1_albumin_max', 'h1_albumin_min',
'h1_bilirubin_max',

'h1_bilirubin_min', 'h1_bun_max',
'h1_bun_min', 'h1_calcium_max',
'h1_calcium_min',

'h1_creatinine_max', 'h1_creatinine_min',
'h1_glucose_max', 'h1_glucose_min',

'h1_hco3_max', 'h1_hco3_min',
'h1_hemaglobin_max', 'h1_hemaglobin_min',

'h1_hematocrit_max', 'h1_hematocrit_min',
'h1_inr_max', 'h1_inr_min',
'h1_lactate_max',

'h1_lactate_min', 'h1_platelets_max',
'h1_platelets_min', 'h1_potassium_max',

'h1_potassium_min', 'h1_sodium_max',
'h1_sodium_min', 'h1_wbc_max',
'h1_wbc_min',
```

```
'h1_arterial_pco2_max',
'h1_arterial_pco2_min',
'h1_arterial_ph_max',


'h1_arterial_ph_min',
'h1_arterial_po2_max',
'h1_arterial_po2_min',


'h1_pao2fio2ratio_max',
'h1_pao2fio2ratio_min'], axis=1)



# In[332]:



data_test_clean=data_test_clean.drop("hos
pital_death", axis='columns')



# In[333]:



data_test_clean.shape



# In[334]:



data_test_clean.head()



# In[335]:



# column_names =
data_test_clean.columns.tolist()

# print('Column Names in
data_test_clean:')

# for column_name in column_names:
#     print(column_name)



# In[336]:



data_test_clean.to_csv(r'C:\Users\brian\O
neDrive\Desktop\Documente
licență\Documentație
scrisă\Date_test_clean.csv',index=False)



# In[337]:



electro_data_test_clean=data_test_clean.d
rop([

'bun_apache','creatinine_apache','hematoc
rit_apache','sodium_apache',

'wbc_apache',

'apache_4a_icu_death_prob'], axis=1)



# In[338]:



electro_data_test_clean.to_csv(r'C:\Users
\brian\OneDrive\Desktop\Documente
licență\Documentație
scrisă\electro_Date_test_clean.csv',index
=False)



# In[339]:



undf_clean=data_test_clean.isnull().sum()

print("\n\n", '\033[1m'+ 'Look for
undefined:' +
'\033[0m',"\n",undf_clean.sort_values())
```

8. Annex 2

8.1. Sesiunea de Comunicări Științifice Studențești [106]


UNIVERSITATEA POLITEHNICA DIN BUCUREŞTI
Facultatea de Electronică, Telecomunicaţii
şi Tehnologia Informaţiei
ETTI
IEEE
Romania Section
DIPLOMĂ DE PARTICIPARE
la
Sesiunea de Comunicări Ştiinţifice Studenţeşti
se acordă lucrării
Estimation of patient survival chances based on the medical data acquired in the first 24 hours in the Intensive Care Unit
realizată de
Elena-Briana BOERU, anul IV, grupa 441F, Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
sub coordonarea ştiinţifică a
Conf. dr. ing. Dragoş-Daniel ŢARĂLUNGĂ, Departamentul de Electronică Aplicată şi Ingineria Informaţiei
DECAN
Prof. dr. ing. Radu Mihnea UDREA
2023
PREŞEDINTE COMISIE
Prof. dr. ing. Adriana FLORESCU

Ministerul Educaţiei

UNIVERSITATEA POLITEHNICA DIN BUCUREȘTI

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

DIPLOMĂ

la

Sesiunea de Comunicări Ştiinţifice Studenţeşti

PREMIUL III

Se acordă studentei **Elena-Briana BOERU**, anul IV,
de la Facultatea de Electronică, Telecomunicații și Tehnologia Informației.

DECAN,

Prof.dr.ing. Mihnea UDREA

5-6 mai 2023

PREȘEDINTE COMISIE,

Prof.dr.ing. Adriana FLORESCU