TECHNICAL UNIVERSITY OF CRETE

ELECTRICAL AND COMPUTER ENGINEERING

MACHINE LEARNING TO DEVELOP A MODEL THAT WILL PREDICT EARLY IMPENDING SEPSIS IN NEUROSURGICAL PATIENTS

Diploma Thesis

Georgios Noikos

Supervisor: Michael Zervakis

Chania, October 2023

TECHNICAL UNIVERSITY OF CRETE

ELECTRICAL AND COMPUTER ENGINEERING

MACHINE LEARNING TO DEVELOP A MODEL THAT WILL PREDICT EARLY IMPENDING SEPSIS IN NEUROSURGICAL PATIENTS

Diploma Thesis

Georgios Noikos

Supervisor: Michael Zervakis

Chania, October 2023

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΓΙΑ ΚΑΤΑΣΚΕΥΗ ΜΟΝΤΕΛΟΥ ΠΟΥ ΘΑ ΠΡΟΒΛΕΠΕΙ ΠΡΩΙΜΑ ΕΠΕΡΧΟΜΕΝΗ ΣΗΨΗ ΣΕ ΝΕΥΡΟΧΕΙΡΟΥΡΓΙΚΟΥΣ ΑΣΘΕΝΕΙΣ

Διπλωματική Εργασία

Γεώργιος Νόικος

Επιβλέπων:
Μιχαήλ Ζερβάκης

Μέλη Επιτροπής:
Γεώργιος Σταυρακάκης
Γεώργιος Γιαννακόπουλος

Χανιά, Οκτώβριος 2023

Approved by the Examination Committee:

Supervisor	**Michael Zervakis** Professor, Department of Electrical and Computer Engineering, Technical University of Crete
Member	**Georgios Stavrakakis** Professor, Department of Electrical and Computer Engineering, Technical University of Crete
Member	**Georgios Giannakopoulos** Grade B / Researcher (Specialized Functional Scientist), NCSR “Demokritos”

Acknowledgements

I would like to give my warmest thanks to Mr. Zervakis and the entirety of the academic staff of the Technical University of Crete, for the knowledge they have provided me with throughout my student years. In addition, I would like to express my sincere gratitude to Mr. Giannakopoulos, Mr. Kosmopoulos and NCSR DEMOKRITOS in general, without whom and their unlimited help, the completion of this work may not have been possible. I would be remiss not to mention Mr. Tsitsipanis and Mr Ntotsikas, doctors of the University General Hospital of Heraklion, for gathering, providing and explaining the data. Finally, I would like to thank my family and friends for their understanding and continuous support all these years, along with the sacrifices they made to make this possible.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Diploma Thesis

MACHINE LEARNING TO DEVELOP A MODEL THAT WILL PREDICT EARLY IMPENDING SEPSIS IN NEUROSURGICAL PATIENTS

Georgios Noikos

Abstract

As sepsis, we currently define a “life-threatening organ dysfunction caused by a dysregulated host response to infection”. Prevention of sepsis, demands its early prediction, a task that has been quite a challenge for the scientific community. With our study, we attempt contributing to this effort, by taking processed, anonymised data, which will be used to build a machine learning predictive model that would predict an upcoming infection, potentially leading to sepsis. Although this model originally takes into consideration, among others, medical measurements of 5 consecutive days, at the end of our study we examine the model’s predictive capacity with a more limited span of days. We even end up predicting based on a single day’s medical measurements, four days prior to infection, obtaining satisfactory results. This goal’s significance is high, since achieving it, would provide the doctors and the nursing staff with some valuable time, constructing an efficient plan to deal with the infection before it causes sepsis. This interval of time could be proven to be decisive about the life of the patient, since sepsis is one of the most frequent reasons for an Intensive Care Unit (ICU) admission and the primary reason for death in the ICU. Data cleaning and pre-processing helped us to feed the best possible dataset to our model, maximizing its predictive capacity for this binary classification problem. Moreover, via the important features of our model, conclusions could potentially be drawn concerning the relation between some clinical input features and the occurrence of sepsis, leading to an enhanced, data-driven understanding of this heterogeneous dysfunction. Early findings indicate efficient classification performance resulting in promising forecasting ability, using various machine learning models, while leaving considerable scope for extending the time between the prediction of the infection and its occurrence .

Keywords:

Prediction, Infection, Sepsis, Machine Learning, Classification

Διπλωματική Εργασία

ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΓΙΑ ΚΑΤΑΣΚΕΥΗ ΜΟΝΤΕΛΟΥ ΠΟΥ ΘΑ ΠΡΟΒΛΕΠΕΙ ΠΡΩΙΜΑ ΕΠΕΡΧΜΟΜΕΝΗ ΣΗΨΗ ΣΕ ΝΕΥΡΟΧΕΙΡΟΥΡΓΙΚΟΥΣ ΑΣΘΕΝΕΙΣ

Γεώργιος Νόικος

Περίληψη

Ως σήψη ορίζεται μια ``απειλητική για τη ζωή δυσλειτουργία οργάνου που προκαλείται από μια απορρυθμισμένη ανταπόκριση του ξενιστή στη λοίμωξη". Η πρόληψη της σήψης απαιτεί την έγκαιρη ανίχνευσή της, κάτι που αποτελεί μεγάλη πρόκληση για ολόκληρη την επιστημονική κοινότητα. Με τη μελέτη μας, επιχειρούμε να συμβάλουμε σε αυτή την προσπάθεια, λαμβάνοντας επεξεργασμένα, ανώνυμα δεδομένα, τα οποία θα χρησιμοποιηθούν για τη δημιουργία ενός μοντέλου πρόβλεψης με τη χρήση μηχανικής μάθησης, που θα προβλέπει μια επερχόμενη λοίμωξη πριν αυτή οδηγήσει σε σήψη. Αν και το μοντέλο αυτό αρχικά λαμβάνει υπόψιν, μεταξύ άλλων, ιατρικές μετρήσεις 5 διαδοχικών ημερών, στο τέλος της μελέτης μας εξετάζουμε την προβλεπτική ικανότητα του μοντέλου, με ένα πιο περιορισμένο εύρος ημερών. Καταλήγουμε μάλιστα να προβλέπουμε με βάση τις ιατρικές μετρήσεις μιας μόνο ημέρας, τέσσερις ημέρες πριν από τη μόλυνση, λαμβάνοντας ικανοποιητικά αποτελέσματα. Η σημασία αυτού του εγχειρήματος είναι μεγάλη, καθώς η επίτευξή του, θα δώσει πολύτιμο χρόνο στους γιατρούς και το νοσηλευτικό προσωπικό να κατασευάσουν ένα αποτελεσματικό σχέδιο αντιμετώπισης της λοίμωξης, πριν αυτή προκαλέσει σήψη. Αυτό το χρονικό διάστημα θα μπορούσε να αποδειχθεί καθοριστικό για τη ζωή του ασθενούς, δεδομένου ότι η σήψη είναι ένας από τους συχνότερους λόγους εισαγωγής σε Μονάδα Εντατικής Θεραπείας (ΜΕΘ) και η πρωταρχική αιτία θανάτου μέσα στη ΜΕΘ. Η εφαρμογή data cleaning και feature selection μας βοήθησαν να τροφοδοτήσουμε το καλύτερο δυνατό σύνολο δεδομένων στο μοντέλο μας, μεγιστοποιώντας την ικανότητα πρόβλεψής του για αυτό το πρόβλημα δυαδικής ταξινόμησης. Επιπλέον, βάσει των χαρακτηριστικών που φάνηκαν να έχουν τη μεγαλύτερη επιρροή στην εκάστοτε πρόβλεψη του μοντέλου μας, θα μπορούσαν ενδεχομένως να εξαχθούν συμπεράσματα σχετικά με τη σχέση μεταξύ ορισμένων κλινικών χαρακτηριστικών ή μετρήσεων και της εμφάνισης σήψης, οδηγώντας σε μια βελτιωμένη κατανόηση αυτής της ετερογενούς δυσλειτουργίας. Τα πρώτα ευρήματα δείχνουν αποτελεσματική απόδοση της ταξινόμησης, γεγονός που υποδεικνύει πολλά υποσχόμενη ικανότητα πρόβλεψης, με τη χρήση διαφόρων μοντέλων μηχανικής μάθησης.

Λέξεις Κλειδιά:

Πρόβλεψη, Λοίμωξη, Σήψη, Μηχανική Μάθηση, Ταξινόμηση

Table of Contents

List of Figures

List of Tables

Abbreviations

WBC	White Blood Cell
NEU	Neutrophils
CRP	C-Reactive Protein
HCT	Hematocrit
PLT	Platelets
GLU	Glucose
UR	Urine
CR	Creatinine
SBP	Systolic Blood Pressure
DBP	Diastolic Blood Pressure
PUL	Pulses
FEV	Fever
SGOT	Serum Glutamic Oxaloacetic Transaminase
SGPT	Serum Glutamic Pyruvic Transaminase
gGT	gamma-Glutamyl transferase
INR	International Normalised Ratio (Prothrombin)
APTT	Activated Partial Thromboplastin Clotting Time
K	Potassium
NA	Sodium
LR	Logistic Regression
NB	Naive Bayes

Chapter 1

1 Introduction

Sepsis, is defined as "life threatening organ dysfunction caused by a dysregulated host response to infection" [1]. It is the most frequent reason for admission to an intensive care unit (ICU) and the primary reason for ICU deaths [2]. However, it is frequently not detected in time. [3] From 2009 to 2019, according to 170 studies concerning the mortality rate of a septic shock (either 30-day or 90-day), they came up with the following results: The average septic shock mortality was about 33.5%. There was a statistically significant decrease of 30-day septic shock mortality rate between 2009 and 2011, which unfortunately, did not continue decreasing afterwards [4]. This, indicates a remaining unmet need for improving sepsis management.

Early diagnosis of sepsis can result to earlier initiations of the appropriate treatment protocols, leading to a significant decrease of sepsis-related mortality, considering that it increases by approximately 8% for each hour treatment is delayed [5].

In the United States, it is considered to be a major public health concern, since the cost of its treatment rises up to $20.3 billion, consisting a massive 5.2% of total US hospital costs in 2011, being the most expensive condition treated amongst the expenses for all hospitalizations [6]. It is easy to see how an early detection of sepsis will help reduce the above expendutes, even though it is one of the most difficult to solve problems in modern medicine. However, in this study, we attempt to do it using Machine Learning.

1.1 Motivation

Several studies have been conducted, aiming at the prediction of downstream events related to sepsis, like an organ malfunction or any kind of a severe septic shock. Nevertheless, the most important aspect of the sepsis prediction, which is to identify upcoming infections before the domain experts discover it, remains unsolved [5].

Many researchers have periodically attempted to provide valid solutions to this problem, conducting researches towards this direction using a variety of machine learning techniques [7], constructing their own sepsis early warning algorithms, like InSight [8], artificial neural networks [9] etc. However, none of these studies has ever been actually applied, because of sepsis' heterogeneous nature, due to which each patient presents different symptoms [5].

The most common issue observed in the existing surveys has to do with the features used. Many researchers are limited to the use of risk scores [10], but these mainly give information about the severity of the condition and the probability of mortality, without including the diversity of sepsis [11].

1.2 Contribution

This study's main objective is to identify an objective early enough for a septic shock to be prevented, so that hospitalized patients will have a better chance of survival. The main idea is to try approaching this as a binary classification problem, where we are going to use 5 consecutive days' data of hospitalized patients in order to predict the appearance of an infection, so that impending sepsis could be prevented. These data, should be innovative compared to similar attempts, highly associated with infections according to domain experts. In light of the foregoing, our work's contributions will be:

- Our first concern was collecting a very comprehensive set of clinical data to form our dataset, describing each patient in detail, while of course meeting all ethical requirements as defined by the law. General University Hospital of Heraklion provided us with this dataset.
- After reading many relevant papers and trying different machine learning algorithms, we ended up examining 3 different classifiers, evaluating their performance in predicting upcoming infection in the given dataset.
- Additional experimentation on various preprocessing techniques concerning filling of missing values, normalization etc.
- Testing a time-related representation of some features appropriate for this analysis in our dataset.
- Identifying the medical measurements which influenced our model's predictions the most.

- Attempting to extend the time interval between the prediction and the occurrence of an infection. Managed to efficiently predict an infection 4 days prior to its appearance.

Summarizing, our survey focuses on both finding the best possible predictive machine learning model for our problem and creating the most innovative and inclusive dataset.

1.3 Structure of Paper

Here we are going to present some important parts of our thesis. To begin with, In Chapter 2 we explain some basic terms used in our study and we proceed with presenting some similar studies stating some noteworthy points about them that could help understanding where our study differs. The entirety of Chapter 3 is dedicated to our dataset, thoroughly analysing it, explaining how we gathered it and all the necessary changes made. In Chapter 4, we start by defining the problem we are trying to solve, then continue with analyzing our approach to solve it. Next, all the experiments made until we conclude to our model, alongside their results and discussion upon them are displayed in Chapter 5. Finally, in Chapter 6 we present the conclusions we made over this study and some suggestions for potential future work in this direction.

Chapter 2

2 Background and Related Work

In this section, we are going to discuss anything that has to be explained or simplified, that contributed to our work. We will try to do that in a way that every possible reader can get the idea of our analysis, even if he is not relevant to this research field. Our work can be seen as two parts of research, with the first one concerning necessary medical terms and the other one being about the selection of an appropriate methodology to train an efficient predictive machine learning model. Last but not least, in this chapter we are going to deal with some other relevant studies, the methods they used, the conclusions they drew and how our work compares to theirs.

2.1 Basic Terms and Concepts

We are now going to explain some necessary theoretical definitions concerning either medical or computer science terms. This knowledge is necessary for someone to understand what is going to be discussed further on.

2.1.1 Infection and Sepsis

Over the years, there have been many tries to define infection and sepsis. However, none of them managed to do so satisfactorily. Currently they are defined as shown below:

- **Infection:** "Infection is the term that defines the entrance and development of an infectious agent in a human or animal body, whether or not it develops into a disease." [12]
- **Sepsis:** As sepsis, we currently define a "life-threatening organ dysfunction caused by a dysregulated host response to infection". [1]
- **Organ dysfunction:** An organ dysfunction, that was mentioned above, can be identified as an acute change in total SOFA (Sequential [Sepsis-related] Organ

Failure Assessment) score ≥ 2 points consequent to the infection. It is worth noting that the baseline SOFA score for someone that did not have an organ dysfunction in the past, is equal to zero and SOFA score ≥ 2 represents a mortality risk around 10%. [1]

2.1.2 Machine Learning

In this section, we are going to explain some important terms in the field of Machine Learning models, that are going to be used later on our study.

- **Supervised Learning**: Training data consists of a collection of labeled examples, based on which the model is trained in order to make a prediction.
- **Unsupervised Learning**: Training data consists of unlabeled examples, which the learner uses to get trained and then attempts to provide a forecast.
- **Labels & Classes**: By the term label, we refer to a category which allows us to appropriately differentiate (label) our data. As class, we define the target field of our problem. In our case, we have a multi-label (e.g. demographic data, each medical measurement etc), multi-class (positive and negative) classification.
- **Classification**: Supervised machine learning method, where the model, after being appropriately trained, tries to predict the correct class for each example of the data. In the event of having just two classes to classify the data, as happens in our problem where we have to distinguish the data between infected and non-infected, this is called “binary classification”.
- **Clustering**: Unsupervised learning technique is used to divide data points based on their similarities or differences into different groups.
- **Instances**: An instance is an observation from our domain of study. In our case, as instances we have patients of the neurosurgical clinic of the University General Hospital of Heraklion.
- **Features**: As features of our dataset, we basically mean individual measurable piece of data, which act like the model’s input for the desired analysis to happen.

In our dataset, the features represent various demographic data (such as age, gender), 5 days of each medical measurement where every day constitutes a different feature etc.

- **Numerical Features**: Features that contain discrete values with no numerical relationship between the different categories, (such as Comordibities, Disease, ICD 10 etc).
- **Nominal Features**: Features that can only take real numeric values (all the medical measurements, age and days of hospitalization in ICU among others).
- **Feature Importances**: Scores, used to determine the relative importance of the features to the final prediction of a predictive ML model.
- **Training Set**: A split of the given dataset, used to train the model to make a prediction.
- **Test Set**: The rest of the dataset, that does not belong to the training set. An unseen, by the ML model, set of examples, used for prediction. A model is evaluated based on these predictions.
- **Cross Validation**: Method used to test and train ML models, based on train-test splits (training set and test set) of the given dataset, on different iterations.
- **Leave-One-Out Cross Validation (LOOCV)**: Exhaustive Cross Validation method where each instance is used as a test set exactly once. It is particularly useful for small datasets but can be computationally expensive for larger ones. In particular, this method, provides a robust estimate of a model's performance by training on all data except one instance (training set) and testing on that one instance (test set), repeating this process for all instances and then calculating the average performance metric.

2.1.3 Classifiers

In this study, after some research that is going to be analysed in Section 4.2.2, we ended up testing three different classifiers. These are presented and explained right below.

- **Logistic Regression**:

  Logistic Regression is a popular machine learning model used mainly for binary classification problems, where the goal is to predict one of two possible outcomes or classes. By default, the model predicts that an instance belongs to a certain class if the probability estimation (prediction confidence) is greater than 50%, otherwise this instance belongs to the other class. Some key points concerning Logistic Regression are:

  – Model Representation: Logistic Regression models the relationship between the input features (also known as independent variables or predictors) and the binary target variable (also known as the dependent variable or response) using a logistic function.

  – Logistic Function (Sigmoid Function): The logistic function, also called the sigmoid function, maps any real-valued number to the range [0, 1]. It is defined as follows:

  $$\boldsymbol{\sigma(z) = \frac{1}{1 + e^{(-z)}}}$$

  where:

  z : is the linear combination of the input features and their corresponding weights.

  – Hypothesis Function: The hypothesis function in logistic regression is the sigmoid function applied to the linear combination of input features and their respective weights:

  $$\boldsymbol{h_\theta(x) = \sigma(\theta^T \cdot x)}$$

where:

$h_\theta(x)$: predicted probability that the target variable is 1 given x.

θ : weights that need to be learned during training.

x : is the input feature vector.

- Loss Function: In order to learn the best set of parameters θ, logistic regression uses the maximum likelihood estimation (MLE) as the loss function. The most common loss function for logistic regression is the log-loss (also known as cross-entropy loss):

$$\boldsymbol{J(\theta) = -\frac{1}{m} \cdot \sum_{i=1}^{N}[y \cdot log(h \cdot \theta(x)) + (1-y) \cdot log(1 - h \cdot \theta(x))]}$$

where:

$h_\theta(x)$: is the predicted probability of the positive class.

m : is the number of training examples.

y : is the true target label (0 or 1).

- Training: The goal of training is to find the optimal values of the parameter vector θ that minimize the loss function. This is usually achieved using optimization algorithms like gradient descent or its variants.

- Decision Boundary: As stated before, assuming the two classes we have are the positive and the negative, the decision boundary is the boundary that separates these two classes. In logistic regression, the decision boundary is a linear function of the input features. When the sigmoid function's output is greater than or equal to 0.5, the predicted class is the positive class; otherwise, it is the negative class.

Finally, it is worth mentioning that even though logistic regression is originally designed for binary classification, it can be extended to handle multi-class problems using techniques like one-vs-all (OvA) or one-vs-one (OvO) strategies. However, this is not an issue in this research.

- **Gaussian Naive-Bayes**:

  Gaussian Naive Bayes is a supervised machine learning model used for classification tasks, particularly when dealing with continuous or real-valued features. It's a probabilistic algorithm that applies Bayes' theorem with the "naive" assumption of feature independence given the class. Despite its simplifying assumptions, Gaussian Naive Bayes can perform surprisingly well in various real-world scenarios. Right below, follows an overview of Gaussian Naive Bayes:

  – Bayes' Theorem: At its core, Gaussian Naive Bayes relies on Bayes' theorem, which is a fundamental concept in probability theory. It calculates the conditional probability of a class given the observed features:

  $$\boldsymbol{P\left(y \mid x_1, x_2, ..., x_n\right) = \frac{\left(\ P\left(x_1, x_2, ..., x_n \mid y\right) \cdot P\left(y\right)\ \right)}{P\left(x_1, x_1, ..., x_n\right)}}$$

  where:

$P(y \mid x_1, x_2, ..., x_n)$:	is the probability of class y given features x_1 through x_n.
$P(x_1, x_2, ..., x_n \mid y)$:	is the joint probability of observing features x_1 through x_n given class y.
$P(y)$:	is the prior probability of class y.
$P(x_1, x_2, ..., x_n)$:	is the evidence or marginal likelihood.

  – Naive Assumption: The "naive" assumption in Gaussian Naive Bayes is that the features are conditionally independent given the class. This means that the presence or value of one feature does not affect the presence or value of another feature, given the class.

  – Gaussian Distribution: Gaussian Naive Bayes assumes that the continuous features follow a Gaussian (normal) distribution within each class. This assumption simplifies the estimation of probabilities.

  – Training: During training, Gaussian Naive Bayes estimates the mean and variance of each feature for each class. These estimates are used to calculate the probability density function of each feature given a class.

- Prediction: To make predictions for a new instance, Gaussian Naive Bayes calculates the posterior probabilities for each class using Bayes' theorem and chooses the class with the highest probability.
- Scalability and Interpretability: Gaussian Naive Bayes is computationally efficient and scales well to high-dimensional data. It's particularly useful when the number of features is large compared to the number of instances. Additionally, the model provides interpretability, allowing you to understand the importance of each feature for classification.
- Strengths and Limitations: Gaussian Naive Bayes is simple, easy to implement, and efficient when the naive assumption holds reasonably well. However, the assumption of feature independence might not always be met in real-world scenarios, which can lead to suboptimal performance.

Last bun not least, as also mentioned in logistic regression, although the ”naive” assumption is primarily designed for binary classification, Gaussian Naive Bayes can be extended to handle multi-class classification using techniques like one-vs-all (Ova).

- **XGBoost Classifier**: XGBoost (Extreme Gradient Boosting) is a powerful and widely used machine learning algorithm for both classification and regression tasks. It is an ensemble learning method that builds a strong predictive model by combining the predictions of multiple weak models, typically decision trees, in an additive manner. XGBoost is known for its high performance, scalability, and ability to handle complex relationships in data. Here are the key features and concepts of the XGBoost Classifier:
  - Gradient Boosting: XGBoost is based on the gradient boosting framework, which involves sequentially adding new models to correct the errors made by the previous ones. It focuses on minimizing a loss function by optimizing the model's predictions.

- Decision Trees as Base Learners: XGBoost uses decision trees as its base learners. These trees are shallow (usually with a limited number of nodes) to prevent overfitting. Decision trees are added iteratively, and each new tree tries to minimize the errors of the previous ensemble.
- Regularization: XGBoost includes regularization techniques to prevent overfitting. It has parameters for controlling the complexity of individual trees, as well as the overall complexity of the ensemble.
- Gradient-Based Optimization: XGBoost employs gradient-based optimization techniques to efficiently minimize the loss function. It uses a combination of first-order gradient and second-order gradient information to make optimization more accurate.
- Handling Missing Values: XGBoost has built-in support for handling missing values in the dataset during training and prediction. This process is going to be analyzed in depth in Section 2.1.4
- Training: During training, the XGBoost Classifier goes through several iterative steps to build an ensemble of decision trees that collectively form a strong predictive model.
- Prediction: To predict a new instance, the trained XGBoost Classifier uses the ensemble of decision trees it has built to make predictions on new, unseen instances.

2.1.4 Filling of Missing Values

Machine Learning Algorithms' knowledge is based on the quality of the data induced. Therefore, since in the majority of clinical studies, missing data issue is very common, we must deal with it efficiently so that the quality of MLA's knowledge is as good as possible. Usually, the filling of the missing data is implemented in the preprocessing of the dataset. Nevertheless, this is not always the case. In our project, there were some missing values in the features that represent medical measurements in the 5-day interval. In order to deal with that, we tested a variety of different approaches on the 3 aforementioned classifiers. Now we are going to explain these methods of filling the missing data.

- **kNN Imputation:**
  k-Nearest Neighbour (kNN), is an imputation method widely used in Machine Learning, which contributes to handling missing values from a dataset. kNN has the ability to predict both discrete (most common value amongst the neighbours) and continuous (mean value amongst the neighbours) attributes. In the same time, it is not necessary to create a predictive model for each missing data, as it is in other methods. It is the best applicable method when there is zero prior knowledge regarding data distribution. kNN method can work with any kind of attributes as a class, just by modifying the attributes we want to be considered in the distance metric. For example, picking for a distance metric Euclidean or Minkowski distance, our algorithm searches for the k closest neighbours, finds them and replaces the missing value with either the mean or the mode of the neighbours. In general, kNN imputation method handles pretty well multiple missing values. Its main disadvantage is that in every loop that the kNN looks for similar instances, the algorithm searches through the entire dataset, so if the database is vast, it will inevitably be time consuming.

- **Column-Wise Mean:**
  This method cannot be furtherly explained. We just computed the mean value of each feature and filled all feature's missing data with this value.

- **XGBoost's built-in "sparsity-aware split finding algorithm":**
  First of all, XGBoost classifier is consisted of multiple decision trees. When deciding how to split a node during the tree construction, XGBoost considers missing values as a possible direction for the split. It compares the quality of the split with and without the missing values and chooses whichever yields the highest gain in the objective function. For example, in the event of a binary classification, the objective function could be a log-loss function. Afterwards, missing values from parent node are assigned to the respective child node, according to the learnt splitting rule, so that the missing value is handled properly as the tree grows. The final prediction is based on the learned rules and XGBoost determines the appropriate path. But how does this splitting rule works?

  The process of finding the optimal split for each feature, could be described in a few steps. First, it makes an initial prediction for all training instances, based on whom it constructs the first decision tree. Then, XGBoost computes the gradient of the log-loss function, which is basically the difference between the predicted probability and the target variable. XGBoost iteratively grows the tree by adding new nodes. It searches for the best split points for each feature evaluating different thresholds and decides the optimal ones based on the improvement of the objective function. Next, it applies regularisation and feature sub-sampling along with pruning to avoid over-fitting and to reduce the tree's complexity. Once again, the decision whether to prune a node or not is based on the improvement of the objective function. Finally, XGBoost repeats this process by calculating new gradients, growing additional trees, and optimizing the objective function until a predefined stopping criterion is met (e.g., a maximum number of iterations or reaching a specific level of performance). So, by optimizing the objective function through iterative tree construction, XGBoost learns the optimal split points that maximize the model's predictive performance. The learning process allows the algorithm to capture complex relationships and make accurate predictions.

In general, it is safe to say that XGBoost treats missing values as a separate category and incorporates them in the tree construction and prediction processes, so it can handle them effectively. [13]

2.1.5 Scaling

In machine learning, scaling refers to the process of transforming features or variables to a specific range or distribution. Scaling is an important preprocessing step as it can help improve the performance of various machine learning algorithms. The technique we decided to use in our project, is a commonly used scaling technique, called Min-Max scaling, also known as normalization.

Min-Max scaling involves transforming the original features of a dataset into a new range, typically between 0 and 1. This is achieved by using the following formula for each feature:

$$\boldsymbol{X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}}$$

where:

X_{scaled} :	is the scaled value of the feature X
X :	is the original value of the feature X
X_{min} :	is the minimum value of the feature X in the dataset.
X_{max} :	is the maximum value of the feature X in the dataset.

2.1.6 Evaluation Tools

- **Confusion Matrix**: A confusion matrix is a fundamental concept in machine learning that is used to evaluate the performance of a classification model. It provides a comprehensive way of understanding the effectiveness of a classification algorithm by breaking down the predictions made by the model into various categories. It is typically represented as a table with rows and columns, where each row represents the instances in an actual class, and each column represents the instances predicted to be in a specific class by the model.

  The four possible outcomes of a binary classification problem, such as ours, are as follows:

  1. True Positive (TP): Instances that are correctly predicted as belonging to the positive class.
  2. True Negative (TN): Instances that are correctly predicted as belonging to the negative class.
  3. False Positive (FP): Instances that are incorrectly predicted as belonging to the positive class.
  4. False Negative (FN): Instances that are incorrectly predicted as belonging to the negative class.

  Table 1 visualizes the complete form of a confusion matrix:

  Now, we are going to see some important evaluation metrics based on the the above values, that are going to help us evaluate our models and choose the most suitable for our case.

	Actually Positive	Actually Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

Table 1: Confusion Matrix

- **Precision for the positive class**: Precision, also known as positive predictive value, measures the accuracy of positive predictions made by the model. It is the ratio of true positive predictions to the total number of positive predictions (both true positives and false positives).

$$Precision_{pos} = \frac{TP}{TP + FP} \tag{1}$$

- **Precision for the negative class**, also known as negative predictive value, measures the accuracy of negative predictions made. It is the ratio of true negative predictions to the total number of negative predictions (both true negatives and false negatives).

$$Precision_{neg} = \frac{TN}{TN + FN} \tag{2}$$

- **Recall for the positive class**, also known as sensitivity or true positive rate, measures the model's ability to correctly identify all instances of a positive class. It is the ratio of true positive predictions to the total number of actual positive instances (both true positives and false negatives).

$$Recall_{pos} = \frac{TP}{TP + FN} \tag{3}$$

- **Recall for the negative class**, also known as Specificity, measures the model's ability to correctly identify all instances of the negative class. It is the ratio of true negative predictions to the total number of actual negative instances (both true negatives and false positives).

$$\boldsymbol{Recall_{neg} = \frac{TN}{TN + FP}} \tag{4}$$

- **F1 Score**, is the harmonic mean of precision and recall for each class. It provides a single value that balances both metrics. Now let's see the formula for the F1 Score of positive and negative class:

$$\boldsymbol{F1\ Score_{pos} = 2 \cdot \frac{Precision_{pos} \cdot Recall_{pos}}{Precision_{pos} + Recall_{pos}}} \tag{5}$$

- **Macro F1 Score**, is the harmonic mean value of the F1 Scores for each class. In the case of a binary classification, Macro F1 Score is the mean of the F1 Score for the positive class and the F1 Score for the negative class.

$$\boldsymbol{Macro\ F1\ Score = \frac{F1\ Score_{pos} + F1\ Score_{neg}}{2}} \tag{6}$$

2.1.7 Time Series Clustering

A time series is a sequence of data points or observations collected at specific time intervals. It is a fundamental concept in various fields, including machine learning engineering. Time series analysis involves studying the patterns, trends, and behavior of data over time to make predictions (time series forecasting), identify underlying patterns, and extract meaningful insights.

In our case, we applied time series analysis in the 5-day measurements we had in our dataset. Then, we appropriately clustered them, using Agglomerative Clustering and K-Means Clustering.

- **Agglomerative Clustering**: Agglomerative clustering is a hierarchical clustering technique to group similar data points together based on their distance or similarity. It is a bottom-up approach, where individual data points are successively merged into larger clusters until a stopping criterion is met. Here's how agglomerative clustering works in the context of time series data:

    1. Distance Calculation: The first step is to compute a distance metric between pairs of time series. This distance metric measures the similarity or dissimilarity between two time series. Various distance measures can be used, such as Euclidean distance, Dynamic Time Warping (DTW) distance, or Pearson correlation coefficient.

    2. Initialization: Each time series is initially treated as a separate cluster.

    3. Agglomeration: The algorithm proceeds by iteratively merging the closest pair of clusters based on the chosen distance metric. This merging process continues until all time series are grouped into a single cluster.

    4. Hierarchy Formation: As clusters are merged, a hierarchy or dendrogram is constructed. This dendrogram provides a visual representation of the clustering process, showing how clusters are merged step by step.

    5. Selecting a Threshold Level: To obtain a specific number of clusters, you can choose a threshold level on the dendrogram. This threshold determines at what height the dendrogram should be pruned, resulting in the formation of distinct clusters. Each branch or subtree below the chosen threshold represents a separate cluster of similar time series.

- **K-Means Clustering**: K-Means clustering is a popular unsupervised machine learning algorithm used to partition a dataset into a predetermined number of clusters. While originally designed for vector-based data points, K-Means can be adapted for time series data by transforming the time series into a suitable format. However, using K-Means for time series clustering presents some challenges due to the sequential and temporal nature of the data. Now, let's see how K-Means clustering can be applied to time series data:

1. Choosing K: First, we need to specify the number of clusters, denoted as K. This can be determined using domain knowledge, elbow method, silhouette score, or other techniques.

2. Data Transformation & Distance Selection: Adapting K-Means for time series begins by transforming each time series into a feature vector. This transformation involves selecting an appropriate distance metric to measure the similarity or dissimilarity between time series. Distances like Euclidean, Dynamic Time Warping (DTW), cosine similarity, Pearson correlation, and others can capture various aspects of similarity, based on the characteristics of the data.

3. Initialization: K initial cluster centroids are randomly chosen from the transformed data. This step can significantly impact the final clustering result, and different initialization methods can be used to improve convergence.

4. Assignment and Update: Each time series is assigned to the nearest centroid, and then the centroids are updated based on the assigned data points. This process iteratively continues until convergence, where data point assignments and centroid positions stabilize.

5. Interpreting Clusters: Once the algorithm converges, the time series clusters can be interpreted. However, interpreting clusters in the context of time series can be more challenging than in vector-based data, as the sequence and temporal information are often not explicitly preserved in the clustering result.

6. Post-Processing: Depending on the application, you might need to post-process the clusters, such as analyzing the cluster characteristics, visualizing the time series within each cluster, or using the clusters for downstream tasks like anomaly detection or, in our case, classification.

2.2 Related Work

First of all, it is necessary to separate our work from another attempt to solve the same problem, made in the past [14]. In their study, they had a smaller and much more imbalanced dataset, while in our attempt, the instances available for the control group, were significantly more. Also, not only did they take into account the total days the patient spent in the clinic of the hospital, which is a feature we would not be aware of yet in the time our model was going to be applied, but it was one of the features with the highest importance in their model. So, in our work, we removed it during data cleaning and tried to make predictions without it. Moreover, a major factor that encourage us to construct a model for the same problem once again, was to examine whether a time series representation of the medical measurements could help the model perform better. Last but not least, in the end of our work, we try to predict with less days of measurements available and see how our model would react. In that way, we will be able to see how promptly we could make a valid prediction. Now let's see how our study compares to other similar researches.

Over the last years, the prediction of sepsis has been of concern to several researchers, who are constantly trying to construct better predictive models for predicting an upcoming infection, using a variety of machine learning algorithms. In most of these studies, the results in theory were quite encouraging, yet they could not be applied in reality, since they could predict the occurrence of an infection just a few hours before it happened.

Indicatively, some have used XGBoost classifier [15], Logistic Regression [16], Naive Bayes [17] etc. In the majority of the studies, the most efficient ML classification algorithm seemed to be XGBoost, something also reassured from a study [18] comparing performance of different ML classification algorithms, including XGBoost, Logistic Regression and Naive-Bayes, aiming to early sepsis prediction. In this experiment, they applied these algorithms to 3 different datasets consisting of a total of 100000 patients, to get a more objective perspective, while the input features were some demographic data and many medical measurements. As already mentioned, they concluded that in their case, the most appropriate model for predicting sepsis in ICU patients, was the XGBoost Classifier.

In the aforesaid study, they evaluated the classifiers using accuracy, precision, recall and the area under the ROC Curve. Moreover, in the study mentioned first, where they tried to deal with the same problem as we do, they also used precision, recall and the Log

Loss Function to compare their models' performance. In addition, in a study where they analysed classification algorithms for predictive analysis on banking data, finding out that the most efficient classification algorithms for their problem were XGBoost and Logistic Regression (confirming once again the very good performance of XGBoost classifier for predictive models), they also evaluate the models based on their recall, precision, accuracy and F1 Score, [19]. In general, this way of evaluating models (i.e. using mainly recall, precision and F1 Score) that are trying to make predictions through classification, is a very popular way, that is met in a variety of the respective studies.

Something worth mentioning, is that most researchers appear to have used as features, risk scores (e.g. MEWS, SIRS, SOFA and others), that offer a little amount of useful information about a patient's clinical state or likelihood of developing sepsis. [11] However, they are dependent on certain measurements taken at predetermined intervals, so they are unable to capture the heterogeneity of sepsis and the unique symptoms of each patient. Additionally, it is observed that in the vast majority of the surveys that use medical measurements, they are limited to a single measurement per feature.

Regarding our initial thoughts about time series clustering of the five-day medical measurements, the most intuitive approach was to implement time series clustering. In a study about a novel short-term load forecasting using time series clustering and classification algorithms, they come to the conclusion that KMeans and kernel KMeans are the best performing clustering methods, providing an accuracy of 0.95 for the predictions. [20] Moreover, in a study where they compare various clustering methods for time series data, results indicate Agglomerative clustering as the best performing method, followed by KMeans clustering [21].

Taking into consideration the issues described above, in this study we attempt to predict an upcoming infection via binary classification, testing between the three classifiers analysed above: Logistic Regression, Gaussian Naive Bayes and XGBoost. Also, we examine whether a time series analysis of certain data could result in better predictions, by implementing KMeans and Agglomerative clustering, since they seem very promising clustering methods based on the papers we read. In terms of features, we attempt to increase the number of input features that are considered while defining the patient-instance compared to similar studies, in order to test if the added data may respectively increase the prediction power of the algorithm. Finally, we attempt to make predictions as early as possible, by removing data collected in certain days, close to the day of the infection.

Chapter 3

3 University General Hospital of Heraklion Dataset

In this chapter, we are going to delve into our dataset, which we were given from the University Hospital of Heraklion. Here, we are going to view our dataset feature per feature, explain how we pre-processed the data so that it will be much easier to handle them, analyze the data retrieval methodology from the domain experts and finally share the experience gained from the visit to the hospital.

3.1 Data Collection

After following the corresponding protocols and filling the necessary paperwork, we received an anonymised patient data collection from the University General Hospital of Heraklion for the needs of this survey.

The total number of patient records (instances) we had in this dataset can be separated into 2 groups. The first consists of the patients who had an infection, forming our sepsis dataset, while the other group is filled with patients that did not appear any signs of sepsis during their stay in the clinic, who form our control group. The existence of these 2 groups, is enough to make our dataset, one for a binary classification problem. At first, we had 41 instances, of which 33 positive (developed an infection while hospitalized) and 8 negative (control group). This was a pretty unbalanced dataset, so we asked from the domain experts if they could gather some more instances for our control group. After some months of data collection, we ended up having a total 58 instances, 33 positive and 25 negative.

Something also noteworthy is the actual ratio of the infected patients to the non-infected. We posed this question to the domain experts, who, after some research in their clinic's records, replied that in reality there is approximately a 15 non-infected to 1 infected patient ratio. However this was not applicable in our dataset, since at this moment we had just 25 instances in our control group, meaning that we should remove almost all of our infected patients except for two, to achieve this analogy, something that of course, could not provide reliable results. However, to this day, the doctors of the neurosurgical clinic of the University General Hospital of Heraklion keep searching for patient records, old or new, to add in the dataset for further research. So in the future, our model's adaptation in the real world could be possible.

As for the features of this dataset, they contain many details for the patients, such as demographic data, several medical measurements (e.g. Fever, White Blood Cells, Systolic and Diastolic Blood Pressure etc.) and many other details from the clinic archive, such as the condition the patient finished his hospitalization at (Mortality, Total Days of Hospitalization, ICU admission etc.) which we are going to analyze in the following section of the paper, Section 3.2. Our target variable, represents whether this patient had an infection during his hospitalization or not. Regarding the medical measurements, these are measurements from 5 consecutive days, with the fifth day being the day the patient had an infection. For the control group, the 5-day measurements are the measurements of 5 random consecutive days. We need to stress that each day represents a unique feature. Now we are going to analyze our dataset for this study.

3.2 Data Description

This dataset, consists of 58 patient-instances, along with 103 features, giving us a total of 5974 data to work with. However, the dataset is not complete, as it has 1066 missing values we are trying to fill using the corresponding methods. So, $\simeq 17.84\%$ of the dataset consists of missing values. In our dataset, there are 6 features with nominal data, making up a total of 348 data which is $\simeq 5.83\%$ of our total data, leaving the rest 5626 our of 5974 data ($\simeq 94.17\%$) to be numerical, generating a 16 numerical to 1 nominal ratio. Here, we should note that all the missing values exist among the numerical data, since all the nominal are complete. As it will be explained later, all the nominal data will be turn into numerical (boolean values) by the use of the appropriate algorithms. Everything that was mentioned above, are presented in the Table 2.

	Instances	Features	Total Data	Empty	Numerical	Nominal
Total Number	58	103	5974	1066	5626	348
Percentage Out of Total	100%	100%	100%	17.84%	94.17%	5.83%

Table 2: Dataset analysis

Now, we are about to explain every single feature existing in our dataset.

- **Infection**: Binary variable informing whether or not this patient was infected while being hospitalized. This is our target variable.
- **Gender**: Binary variable, indicating patient's gender.
- **Age**: Patient's age.
- **Total Days of Hospitalization**: Total number of days this patient spent in the clinic.
- **Hospitalization in ICU**: Binary variable informing whether or not this patient was admitted in the ICU.
- **Days of Hospitalization in ICU**: Total number of days this patient spent in the ICU prior to the infection.
- **ICD 10**: International Classification of Diseases - Tenth (10th) Revision, is an international clinical cataloging system for defining and reporting diseases.
- **Disease**: A harmful deviation from the normal functioning of the human body, not due to injury.
- **Surgery**: Surgeries performed to this patient.
- **Comordibities**: Patient's co-existing diseases or conditions alongside alongside the main disease.
- **Identified Microbe**: The microbe discovered in this patient.
- **Cultures to Identify Microbe**: Type of culture that identified the microbe in this patient.
- **Antibiotic Treatment**: The antibiotics given to this patient to fight the infection.

- **Duration of Antibiotic Treatment**: For how long had this patient been taking the antibiotic treatment.

- **Route of Administration of Antibiotic Treatment**: The method that was used by the nursing staff to administer the antibiotics to this patient.

- **WBC**: White Blood Cells (WBC) - A part of our immune system that protects our body from infection. White blood cells circulate through the bloodstream and tissues, attacking unknown organisms entering the human body. They are the response to any injury or illness. A normal WBC count is approximately from $4.5 \cdot 1000/mm^3$ to $10.5 \cdot 1000/mm^3$. Common symptoms of WBC disorder are, among others: fatigue, chronic infections, unexplained weight loss and generally a feeling of being unwell.

- **NEU**: Neutrophils (NEU) are a type of white blood cell (leukocytes) and in fact the most common one in the human body, acting as its first line of defense, by helping the immune system fight infections and heal injuries. Normal NEU range in the human blood is from $2.5 \cdot 1000/mm^3$ to $6.0 \cdot 1000/mm^3$.

- **CRP**: C-Reactive Protein (CRP) is a pentameric protein, synthesized by the liver. CRP's level rise when the human body suffers from any kind of inflammation. A normal CRP range for healthy adults is from 0.3 mg/dL to 1.0 mg/dL. Higher values of CRP could indicate possible inflammation.

- **HCT**: Hematocrit (HCT) is the percentage of red cells in the blood. Normal range for healthy adults may vary, but in general, they lay between 40.5% and 50.5% for males and from 36.0% to 44.5% for females. High percentage of HCT could result to dizziness and feeling light-headed when low levels of HCT could cause unexplained fatigue, pale complexion and a generalized weakness.

- **PLT**: Platelets (PLT), blood's smallest component, perform a clot by clustering together, in order to stop bleeding. They are basically something like a natural bandage. A normal platelets count range in the blood is between 150000 platelets/mL and 450000 platelets/mL of blood. Some symptoms of platelets disorders could be: cuts or sores that don't heal or are slow to heal, skin that bruises easily and unexplained nosebleeds or bleeding from the gums.

- **GLU**: Glucose (GLU), blood's main type of sugar, is the main source of energy for the body's cells. It comes either from digesting food, or it can be produced by the body from other substances. A normal GLU range for healthy adults is from 90 mg/dL to 130 mg/dL. Low levels of glucose (hypoglycemia) could lead to symptoms like fast heartbeat, sweating, anxiety, shaking, hunger, dizziness etc. On the other hand, high levels of glucose (hyperglycemia) could result to increased thirst and a dry mouth, needing to pee frequently, tiredness, weight loss, even blurred vision.

- **UR**: Urine (UR) helps to evaluate kidney functionality. Normal levels of urine vary between 10 mg/dL and 50 mg/dL. Urine level disorders, have symptoms like a strong urge to urinate that doesn't go away and a burning feeling when urinating. These often indicate liver diseases.

- **CR**: Creatinine (CR) is a waste product the kidneys remove from the blood, used to evaluate kidney functionality. Normally, the expected range for a healthy male is from 0.7 mg/dL to 1.3 mg/dL and for a healthy female, between 0.6 mg/dL and 1.1 mg/dL. Some symptoms of elevated creatinine levels are: Mental confusion, elevated blood pressure, weakness, chest pain and muscle cramps, dehydration and changes in the frequency of urination.

- **SBP**: Systolic Blood Pressure (SBP), is the pressure applied at the arteries' walls at the moment of a heartbeat. Normal SBP range for healthy adults is from 90 $mmHg$ to 120 $mmHg$. Symptoms of high or low systolic blood pressure (systolic hypertension) can be: nausea, headaches, shortness of breath, confusion and vision issues.

- **DBP**: Diastolic Blood Pressure (DBP), is the force in the arteries' walls when the heart rests between heartbeats. Normal DBP range for healthy adults varies between 60 $mmHg$ and 80 $mmHg$. Symptoms of paranormal diastolic blood pressure values could be chest pain, breathing difficulties, speech changes and loss of consciousness.

- **PUL**: Pulses (PUL) are a rhythmical throbbing of the arteries, as the blood circulates through them. The normal pulse range for healthy adults lays between 60 to 100 beats/minute. Symptoms of low pulses are: Chest pain, confusion, dizziness, getting tired easily while doing any kind of physical activity, even fainting or near-fainting experience. On the contrary, some symptoms of higher than normal pulses, could be a sensation of racing, flopping in the chest, chest pain, fainting, shortness of breath etc.

- **FEV**: Fever (FEV) is any rise of body temperature above the normal range, meaning any body temperature measured over $37.0\ ^oC$. High fever (over $38.0\ ^oC$) often indicates illness or infection.

- **SGOT**: Serum Glutamic Oxaloacetic Transaminase (SGOT), is an enzyme present in the liver and the heart cells. It is released on the blood when there is a malfunction or damage in the liver or the heart. The normal SGOT range is 5 to 40 units per liter of serum (liquid part of the blood). When SGOT levels rise, this indicates a liver or heart damage.

- **SGPT**: Just like SGOT, Serum Glutamic Pyruvic Transaminase (SGPT), is also an enzyme normally found plasma, many body tissues and mostly in the liver. When the heart or the liver is damaged, SGPT is released into the bloodstream. Normal SGPT values lay between between 7 and 56 units per liter of blood serum. SGPT disorder is probably caused by a liver or heart malfunction.

- **gGT**: gamma-Glutamyl transferase (gGT), is an enzyme found throughout the whole body, but mainly in the liver. When a liver damage occurs, gGT is leaked into the blood. The expected gGT values rest between 5 and 40 units per liter of serum. High levels of gGT may indicate liver disease or damage to the bile ducts (tubes that carry bile -fluid made by the liver, important for digestion- in and out of the liver).

- **INR**: International normalised ratio (INR) measures the time needed for the blood to clot, through the prothrombin time (PT) test. Normal INR range for a healthy adult is from 0.8 to 1.1.

- **APTT**: Activated Partial Thromboplastin clotting Time (APTT), is one of the most common blood coagulation tests, used to evaluate the work of clotting factor in the blood, by measuring the time the blood takes to clot. Normal APTT range is from 21 to 35 seconds.

- **K**: Potassium (K) is a type of electrolyte, necessary for the functionality of human body. Helps nerves and muscles work normally hence regularizing heartbeat. Moreover, it helps nutrients move into and waste move out of cells. Normal K values are $3.5\ mEq/L$ to $5.2\ mEq/L$ for adults. Low levels of potassium may result in weakness, muscle cramps, confusion, constipation, arrhythmia, or in some cases, increased urination. On the contrary, some symptoms of high potassium levels are heart palpitations, shortness of breath, chest pain, nausea, or vomiting

- **Na**: Sodium (Na) is the major positive ion (cation) in the fluid surrounding cells in the body. It contributes to controlling the amount of fluids and acids in the human body. The normal range for blood sodium levels is from $135\ mEq/L$ to $145\ mEq/L$. High levels of sodium in the blood (hypernatremia), can cause thirst, low urination, vomiting, diarrhea, confusion, seizures etc, while low sodium levels (hyponatremia) may lead to nausea, headaches, constant fatigue, low blood pressure, muscle weaknesses and so forth.

- **Mortality**: Boolean variable informing whether this patient exited the clinic dead or alive.

3.3 Manual Data Cleaning

Something important in terms of processing the dataset, is to ensure its uniformity. Since three separate domain experts collected the data, there were many little differences between the way the measurements were written, along with some details that would be pretty helpful if they were recorded differently. These are the issues that are about to be discussed in this section of the paper.

First of all, we had to **remove some features** that were not suitable for our survey, for multiple reasons.

In particular, there were some features related to post-infection details and procedures, such as:

- <u>Identified Microbe</u>

- <u>Cultures to Identify Microbe</u>

- Antibiotic Treatment
- Duration of Antibiotic Treatment
- Route of Administration of Antibiotic Treatment

Then, there were some features that we could not possibly know at the time our model is going to be used, like:

- Total Days of Hospitalization
- Mortality

Lastly, we could merge “Hospitalization in ICU” and “Days of Hospitalization in ICU” in a single feature named “Days of Hospitalization in ICU”. This feature’s value would be zero (0) if the patient was never admitted in the ICU, otherwise it would just have the number of the days he spent in the ICU.

$$\left.\begin{array}{l}\bullet\ \text{Hospitalization in ICU}\\ \bullet\ \text{Days of Hospitalization in ICU}\end{array}\right\}\ \text{Days of Hospitalization in ICU}$$

Next, we replaced every comma (‘,’) in any feature with **decimal numerals** with a decimal point (‘.’), so that our model does not confuse them as categorical values.

This brings us to another change, where in features containing multiple categorical values per instance, such as “Comordibities” , we separated them with comma (‘,’) instead of slash (‘/’) or anything like that, while having only the first letter of each value as a capital letter.

Moreover, every shell containing no value, had a dash (‘-’). We erased these dashes and left the cells empty, so that the model can read them immediately as NaN (Not-a-Number).

Last but not least, every feature that contained multiple measurements (e.g. SGOT/SGPT/gGT, SPB/DBP etc) was divided so that each feature would have a single measurement.

3.4 Visiting University General Hospital of Heraklion & Proposed Changes

While I was working on this research, a trip to Crete emerged, so I decided to take the most of it and pay a visit to University Hospital of Heraklion to meet the doctors that were providing me with data and discuss about the model and potential changes.

3.4.1 Meeting in the University General Hospital of Heraklion

To begin with, the main topic of this meeting were the general aspects of the model and the changes needed in comparison with the previous attempt for the same issue [14]. We concluded that something really useful, among others, would be to increase the number of patients in the dataset, and more specifically the control group, to get close to an one-to-one match, since until then the infected patients outnumbered the non-infected significantly.

Next, for the retrieval of new data, I decided to propose them to proceed based on the changes that were made in the excel sheet, as mentioned above in Section 3.3.

3.4.2 Proposed Data Collection Method

In a few words, what was told and agreed in this meeting regarding the recording of the data, was:

- In features containing decimal numerals, the decimal point should always be a ‘.’ and never a ‘,’.
- In features with multiple categorical values per instance, these should be separated with a comma ‘,’ and just the first letter of each value should be capitalized.
- Missing value cells should be left empty.
- Only one medical measurement per feature.

Moreover, it was agreed to contact each other once every two weeks to get informed about any new cases appropriate to be added in the dataset.

3.4.3 Addition of New Data

Finally, after some months of gathering instances, as the model was taking its final form, the domain experts managed to gather 17 new instances for the control group, reaching the 25 in total. Comparing them to the 33 infected patients, we can see that we are pretty close to the one-to-one match, as in the beginning the instances for the control group were just 8.

In terms of percentage, at first we had 75.8% of our dataset filled with infected patients and just 24.2% of our dataset making up the control group. However, at the point that data collection was finished, the infected patients constituted the 56.9% of the total patients, leaving the rest 43.1% of the dataset to the control group.

3.5 Data Retrieval General Methodology

Another aspect of my visit in the neurosurgical clinic of the hospital, was to see by myself the process that has to be followed for the dataset to be filled.

To begin with, let's see the most complex procedure of these two, this of recording the infected patients. What it needs to be done, is to search through the digital archives of the University General Hospital of Heraklion (Crete) for neurosurgical patients who arrived in the hospital with no infection, yet they presented one during their hospitalization. Once a suitable case is found, values for certain features are recorded in a span of 5 days, with the fifth day being the day the patient was sent for culture in order to confirm an infection. However, to get to the desired patient-instances, we just have to exclude patients whose profile does not match our case of interest. In order to do this, there is a sequence of actions that needs to be followed. The first thing to check in the archives, was the total days of hospitalization, which had to be at least 5. So every neurosurgical patient who was hospitalized for 4 days or less, was immediately ruled out. Then, the notes for the patient's admission to the clinic were reviewed. If they entered the hospital with an infection alongside their neurological issue, they were also ruled out. Finally, patients that did not present any form of infection while they were being hospitalized were excluded from our dataset. The patients remaining, are the main candidates for our dataset. We tried our best to find the optimal cases. As optimal case we define a patient who has as many measurements as possible for the features wanted in the 5-day period of time we need. In the event of a patient having multiple measurements per day for a certain feature (e.g. fever, blood pressure etc), we insert to our dataset the most abnormal one.

As for the collection of data for our control group, the process was way simpler, yet not so common. All the patients that did not present an infection during their hospitalization, were potential instances for our dataset. Even though this might seem easy to find, the reality was that the patients that did not appear signs of infection, were not frequently examined by the nursing staff. This, made our work difficult, since we had to find patients with as much values filled as possible, in order to avoid having too many missing values in the final dataset.

Chapter 4

4 Proposed Method

In this section, we will define the exact issue that our model is trying to solve, along with the methodology adopted to deal with it. This methodology, contained a series of things needed to be done, such as the handling of our dataset, feature engineering and modeling of the problem, selection and configuration of the appropriate classifier and method to fill the missing values and finally, evaluation of the models to select the optimal one.

. We will also try to explain in detail how our model was constructed to achieve its goal. We will expand in every step of this process, including among others: data pre-processing, model training, selected classifier, computing feature importances and performing exploratory data analysis.

4.1 Problem Definition

The main issue this research is focused on, is the early prediction of impending sepsis, by predicting an upcoming infection before the clinicians find it out. In this thesis, we attempt finding the optimal machine learning algorithms to train our model, in a dataset of neurosurgical patients and classify them according to whether or not they're going to get infected, so that they can be helped in time via the use of the proper antibiotics. This classification is essentially a prediction as to how likely this patient is to develop sepsis in the future and put his life at risk. The time margin this prediction will gain, should give the domain experts the opportunity to find an effective way of dealing with the infection before it turns into sepsis. Hereinafter, we will discuss the general points of our approach, before we analyze them in our experiments in Section 5.3.

4.2 Our approach

In this section, we are going to explain our approach in order to train an efficient model step-by-step. The problem we were given, which was defined previously, we decided to handle it using binary classification. Since every binary classification problem, should have a target variable, ours was a feature that indicates whether or not a patient had an infection while in the clinic. Since the number of available instances was very low we used a leave-one-out approach, where for each instance a separate model is trained using all the other instances as training data. As a final step left before we could implement binary classification, we had to choose an efficient, for our case, classifier. Although our optimal model considered each day of every single medical measurement as a separate feature, we also had to test if a time series analysis of them could provide better results. Then, we had to see which are the important features in our model, while also making sure that there was no feature that could influence by itself the outcome of the prediction to a greater extent than wanted. Finally, we iteratively removed some features representing specific, each time, days of medical measurements, to see how our model would react and whether or not could we make any conclusions based on that. Now let's get into details about everything we mentioned above.

4.2.1 Training Model

We trained our model using Leave-One-Out cross validation (LOOCV), as the optimal form of k-fold cross validation, since the available instances were too few. It is worth mentioning that as a way of fully understanding its function, we wrote our on function for the LOOCV without using the existing one provided by Python. In the general case of a k-fold cross validation, we split the dataset into 2 subsets. The one is used as a training set and the other one as a test set, which change randomly in every loop. Yet, in the specific case of a LOOCV, the test set consists of a single instance, different in every loop, completing as many loops as the number of the instances. So, at the end, the model has been trained with every single instance as a test set, meaning it has tried to make a prediction for every single instance, based each time on all the others (as a training set). Having constructed our own function for the implementation of the LOOCV as noted previously, it was easier to get information for each one of these predictions, such as their confidence as we are going to see in Section 5.3.6.

4.2.2 Classifier Selection

Regarding the classifiers, it is understood that in order to select the optimal classifier to build our model, we could not test every single one. With that being said, we tried to choose 3 of them, that seemed to be a good fit for our case, after studying about them. First, we could not let out one of the most used classifiers, Logistic Regression. Many studies have shown that it is one the easiest to implement while also being accurate in its predictions [22] [23]. Next, we had a Naive Bayes classifier and in particular Gaussian Naive Bayes, a classifier who has several advantages to display [24]. First of all, it employs an intuitive way of operation. Contrary to, e.g. Neural Network and similar to Logistic Regression, Gaussian NB does not require from the user to set many parameters, something that makes the model design process significantly simpler. In addition, Gaussian NB returns probability values for each prediction, something easier applicable in various tasks and simpler to handle, than it would be if an arbitrary scale was used instead. Lastly, it does not require large amounts of data and it can make very fast computation in the decision making process. Finally, we decided to put into the tests a state of the art classifier, such as XGBoost. This classifier has been proven to be one of the most efficient, either compared to other machine learning algorithms for prediction of diseases [25] or to deep learning algorithms like Neural Networks [26]. As we are going to see later on in our study, XGBoost was indeed the optimal classifier for our model too and not due to randomness, as a paired T-test will prove (Section 5.3.5).

4.2.3 Time Series Analysis

Our initial thought about our approach to solve this problem, contained a time series analysis of the features that represented consecutive days of the same medical measurement. Although as we are going to see in the next chapter, our model at this point was already working excellently well, we decided to follow our initial plan just to see the performance of a predictive model using time series forecasting. Thus, we had to consider an easy to implement, yet proven to be efficient methodology. So, we had to cluster the time series data before we could make predictions based on them. As clustering methods for time series, we leaned into 2 of the most used in the papers we read, k-means [27] and agglomerative [28]. This process returned some cluster assignments which we used to turn the clusters into features by one hot encoding them. We are going to look further into this approach at the "Experiments" section below (Section 5.3), but these models were not as efficient as the previous ones, so we did not continue so we did not continue our work in that direction. So, now, we are going to continue with some tests about our feature's objectivity and find the features that have the most influence in the final prediction.

4.2.4 Feature's Importance and Potential Bias

Having selected our most efficient model, the only thing left was to make sure that our model is not biased. In order to do this, we used exploratory data analysis, a method that reveals important statistics about the features. Following a process that will be explained in Section 5.3.8, we made sure that no feature in our dataset used to train our model, could betray by its value, the actual value of the target variable. Of course, if there was such a feature, this would immediately lower our model's objectivity. After making sure that this was not an issue, we decided to find the features that had a significant impact in our model, helping our model to make the predictions. To do that, we just had to compute the feature importances in our model. The results we got, indicated that among others, fever, hematocrit and platelets had the biggest importance. Another clue we could get from this computation, could answer the question of whether or not there are important days of the measurements. As we will see in the corresponding section below (Section 5.3.7), first days appeared to have much more influence in the final prediction. This, aroused our interest to further investigate it and see how our model would react in case we removed specific days of measurements.

4.2.5 Reducing the available features by day

Even though this was not included in our initial approach, we were curious about our model's reaction to removing certain features from our dataset. We would each time extract features who stood for specific days of medical measurements. Our constructed methodology was split in two main parts. First part designated a removal from the final towards the first day, while the second part specified the other way around, i.e. start with extracting the first day, and continue until we are left with just the day of the culture that identified an infection. By doing this, we wanted to test the significance of the measurement's days in our model and if our conclusion was in line with what we saw in Section 4.2.4 where we found out that almost all of the most important features were measurements from the first day. The results we got were pretty enlightening, as the model performed way better when we started removing days close to the occurrence of the infection. Also, the biggest difference noticed in the efficiency of our model was based on whether the first day of measurements was included in the dataset or not. This showed once again how valuable the measurements of the first day are to the model. Having now described our approach to handling the problem posed to us, in the upcoming chapter, it is time to see in detail all the experiments made, until we concluded to the optimal model.

Chapter 5

5 Experiments and Discussion

In the previous chapter, we explained our approach in depth, analyzing every aspect of its methodology. Here, we will talk about the process followed to end up with the aforementioned results, focusing on the experiments. In general, in the previous chapter (Chapter 4) we explained "what" we did, while in this chapter, we are going to analyze "how" we did it.

5.1 Questions we need to answer

In the beginning of our study, our primary concern was to construct a credible methodology to deal with the problem. We decided to go with the obvious choice of a binary classification algorithm, but this caused a lot of questions that needed to be answered. First of all, we had to take advantage of some already existing data from the neurosurgical clinic of the General University Hospital of Heraklion and appropriately implement data cleaning and data pre-processing so that we could train our model on them. Next, we needed to find the most suitable evaluation tools that would give answers to what we are looking for. Using them, we are going to search for the best possible classifier for our model, selecting among the three mentioned in Section 4.2.2. As we were training the model on the existing dataset, new data came up, so we wanted to see how the model would react and if the performance of the tested classifiers would change with the dataset. Even though we had a clear view of our model, we would like to apply time series clustering in certain features and check our model's response. Finally and after having concluded to our most efficient model, we had to explore some other details of this, such as its predictions' confidence, the most important features, if its better performance compared to the other models was random and if there were and features that were so inextricably linked to the occurrence of infection that could betray the outcome of the prediction. Lastly, there was something that came up during our research and was not included in our initial methodology. This was about the reaction our model would have if we were to remove some days of medical measurements from the features. The answers to the questions stated above, are going to be given in the following sections, along with the experiments made to get them. For now, let's see everything that was needed to be done before we proceed with these experiments.

5.2 Experimental Setup

To begin with, regarding the coding aspect of our work, we used Spyder IDE in the Anaconda environment, to code in python language. As for the evaluation of our models, we had to focus on precision and recall, so the F1 Score that combines them, seemed like the obvious choice. However, we should take into consideration both classes, so eventually we used Macro F1 Score, which is the mean of the F1 Score for each class, in our case positive and negative. As for the dataset, it is important to note that we during our study, we trained our model in two datasets, not totally different, but the one being a more complete version of the other. What follows, are some more details for both the dataset and the evaluation process applied.

5.2.1 Dataset

At first, as it has already been mentioned above, for our dataset we had 41 patient-instances, 33 of them being infected and 8 non-infected. This dataset already existed when we decided to deal with this problem, so we started our work based on that. We tried our best to train an efficient model over these data, but it was impossible, since the data were totally unbalanced and biased. So, after contacting the domain experts, we acquired 17 new non-infected patient-instances, reaching a total of 58 instances, 33 infected and 25 for our control group. Then, we started from scratch to find the best fit for our model once again, this time training it on the new dataset. More details about the composition of the dataset, are listed in Section 3.2.

5.2.2 Evaluation Process

Next, we need to explain how exactly we evaluated each attempt, in order to find the optimal fits for our model. A detailed description of the evaluation tools used, has been made in the corresponding section (Section 2.1.6). It is easy to figure out that the most important score of all was the one combining all the others, the Macro F1 Score. Although this score was enough to evaluate a model's performance, in the following tables that contain information about the efficiency of each model based on these evaluation tools, precision and recall will be included, as they contain important information about the models. In addition, the confusion matrix had a decisive role in our experimental evaluation, since its visualization in the environment of Spyder IDE would give us an immediate point of view concerning the performance of our tested model, as it makes it really easy to keep track of the false predictions and their kind (false positive or false negative). We have to note here what the domain experts consulted as: If it is to have a wrong prediction, it would be better to mistake someone who is not going to get infected, rather than falsely predicting for a patient that is going to get infected. That is because in the majority of the cases, taking a non-necessary antibiotic treatment for a short period of time (until they see that this patient is not going to get infected), is better than neglecting a patient's treatment until the clinician finds out the infection.

5.3 Experiments

In this part of the study, we shall analyze everything that needed to happen, for the purpose of building our final model. Of course, we will start by explaining how we handled the given data. Then, the biggest challenge of our work, was to find the best combination of classifier and a method to fill the missing values, which constituted our main experimental target. As for these experiments, it has already been mentioned that we had two versions of our dataset which we used to train the models, so, after talking about the pre-processing of our data, we will evaluate our models' performance on both of them. Then, to conclude to the final model, we are going to implement time series clustering in our data. At this time, we will have concluded to our optimal model for this problem, so we are going to investigate some details about this model, such as the confidence of its predictions, its feature importances, even its reaction to the extraction of some of its features from the dataset, while also making sure that everything is legitimate.

5.3.1 Data Pre-Processing

First thing that should be done by the time we acquired the data, was data-cleaning as described in Section 3.3. Next, we had to handle the categorical data, turning them to numerical. We did that via one-hot-encoding, according to which, every categorical variable gets converted into dummy/indicator variables (i.e. as many 0/1 variables as there are different values). Now, with all the data in numerical form, we had to apply normalization, setting the values of each feature within the range of 0 and 1 (1 being the max value of each feature) using min-max scaling, as noted in Section 2.1.5. A significant part of the pre-processing of the available data, concerns the filling of the missing values. However, as mentioned earlier, this was put to test alongside the classifiers, something we will be experimenting with, right after.

5.3.2 Testing Classifiers and Data Filling Methods on First Dataset

Regarding the filling of the missing values, we attempted two different ways. First was the filling of each missing value with the mean value of the feature it belongs to. The second one, was using kNN Imputation method. Then, concerning the classifiers put to test, for the reasons explained in Section 4.2.2, we decided to examine Gaussian Naive Bayes, Logistic Regression and XGBoost.

Below, in the figures 1, 3 and 5 we can see the confusion matrices for the 3 classifiers (Gaussian Naive-Bayes, LR and XGBoost respectively) using kNN Imputation to fill the missing values, while in the figures 2, 4 and 6 are shown the confusion matrices for the same three classifiers in the same order, but this time using the mean value of each feature to fill the missing values.

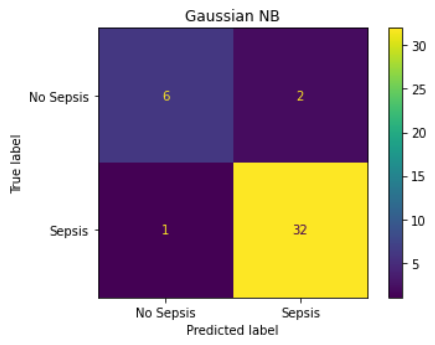

Figure 1: Initial Dataset - Gaussian NB using kNN imputation.

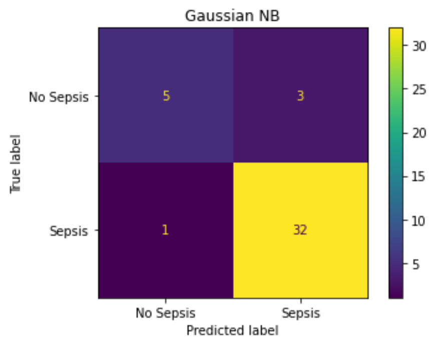

Figure 2: Initial Dataset - Gaussian NB using each Feature's Mean.

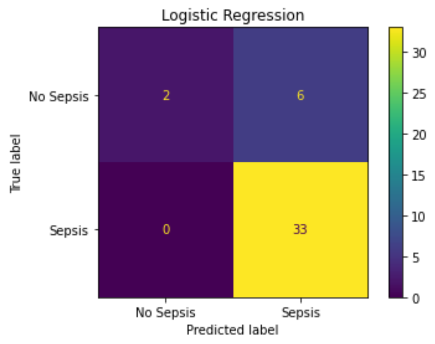

Figure 3: Initial Dataset - Logistic Regression using kNN imputation.

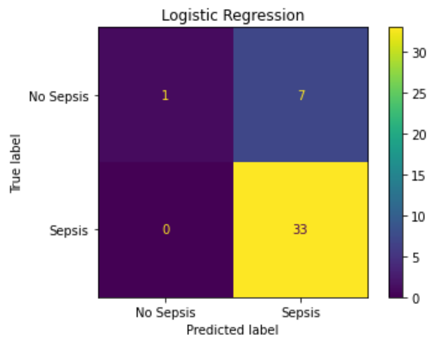

Figure 4: Initial Dataset - Logistic Regression using each Feature's Mean.

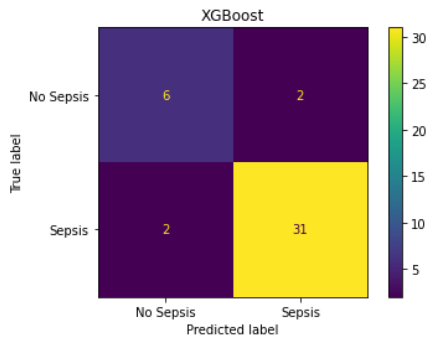

Figure 5: Initial Dataset - XGBoost using kNN Imputation.

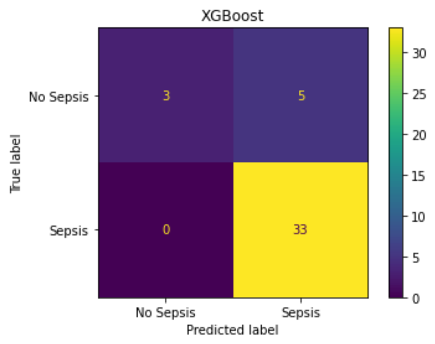

Figure 6: Initial Dataset - XGBoost using each Feature's Mean.

As we can see, regarding the filling of the missing values with kNN imputation, the Gaussian NB had the best performance out of them, by having just 3 false predictions, followed by XGBoost with 4 mistakes and last came Logistic Regression with 6 wrong predictions. As for the filling of the missing values using the mean of each feature, the Gaussian NB outperformed the other two classifiers once again, by making 4 mistakes against 5 and 7 of XGBoost and Logistic Regression respectively. We observe that the kNN imputation appears to be a better fit for this model at every classifier tested. Also, it is clear that in both cases, Logistic Regression classifier was very influenced by the unbalance of the data, since it predicted almost every non-infected patient, as infected.

Now, we are going to see some tables about the scores these models achieved in the evaluation metrics used (Precision, Recall and F1 Score). In Tables 3, 5 and 7, we see the scores performed by Gaussian Naive-Bayes, Logistic Regression and XGBoost Classifier respectively, while using kNN Imputation to fill the missing values. Next, the same scores are shown in Tables 4, 6 and 8, but this time the mean value of each feature was used as a filling method for the missing values.

Gaussian NB kNN	Positive Class	Negative Class
Precision	0.9412	0.8571
Recall	0.9697	0.7500
F1 Score	0.9552	0.8000

Table 3: Initial Dataset - Evaluation of Gaussian NB using kNN Imputation.

Gaussian NB Mean	Positive Class	Negative Class
Precision	0.9143	0.8333
Recall	0.9697	0.6250
F1 Score	0.9412	0.7143

Table 4: Initial Dataset - Evaluation of Gaussian NB using each Feature's Mean.

LR kNN	Positive Class	Negative Class
Precision	0.8462	1.0000
Recall	1.0000	0.2500
F1 Score	0.9167	0.4000

Table 5: Initial Dataset - Evaluation of Logistic Regression using kNN Imputation.

LR Mean	Positive Class	Negative Class
Precision	0.8250	1.0000
Recall	1.0000	0.1250
F1 Score	0.9041	0.2222

Table 6: Initial Dataset - Evaluation of Logistic Regression using each Feature's Mean.

XGBoost kNN	Positive Class	Negative Class
Precision	0.9394	0.7500
Recall	0.9394	0.7500
F1 Score	0.9394	0.7500

Table 7: Initial Dataset - Evaluation of XGBoost Classifier using kNN Imputation.

XGBoost Mean	Positive Class	Negative Class
Precision	0.8684	1.0000
Recall	1.0000	0.3750
F1 Score	0.9296	0.5455

Table 8: Initial Dataset - Evaluation of XGBoost Classifier using each Feature's Mean.

All the scores displayed in the tables above, can be merged in Table 9 that follows, which shows the Macro F1 Scores for the evaluation of these models in the initial form of our dataset. As we have already noted, this score by itself is enough to evaluate each model performance and find the best one.

Macro F1 Scores	Gaussian NB	Logistic Regression	XGBoost Classifier
Filling with kNN Imputation	**0.8776**	0.6583	0.8447
Filling with Feature's Mean	0.8277	0.5632	0.7375

Table 9: Initial Dataset - Macro F1 Score.

As we can see, everything we noticed by analysing the confusion matrices, is confirmed by the Macro F1 Scores. According to Table 9, the filling of the missing values using kNN Imputation, indeed performed significantly better in every possible model, while Gaussian NB seems to be the best performing classifier. So, our optimal model for this version of our dataset, is apparently Gaussian NB for our classifier while using kNN Imputation method to fill the missing values.

5.3.3 Testing Classifiers and Data Filling Methods on Final Dataset

However, we could not rely on the previous findings, since the dataset was not appropriate at all. So, following the procedure described previously, we now had a total of 58 instances in a much more balanced dataset, containing 33 infected and 25 non-infected patients. To train our model and find the best fit, we followed the exact same steps as before, in the first version of our dataset.

So now, we have to see again the confusion matrices and some tables containing the models' evaluation measures scores to compare the performance of the classifiers and the data filling methods for the final form of our dataset.

In Figures 7, 9 and 11 we can see the confusion matrices of Gaussian Naive-Bayes, Logistic Regression and XGBoost Classifier, while using kNN Imputation to fill the missing values. Respectively, in Figures 8, 10 and 12 are displayed the confusion matrices of the aforementioned classifiers in the same order, but this time using the mean of each feature to fill this feature's missing values.

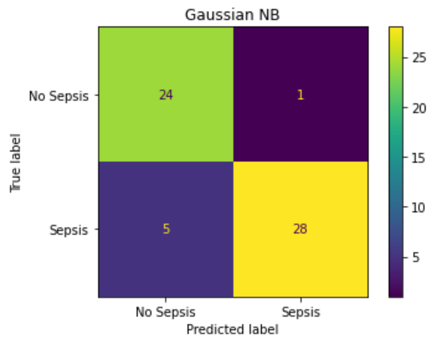

Figure 7: Final Dataset - Gaussian NB using kNN imputation.

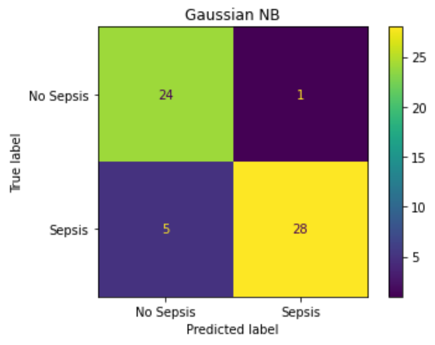

Figure 8: Final Dataset - Gaussian NB using each Feature's Mean.

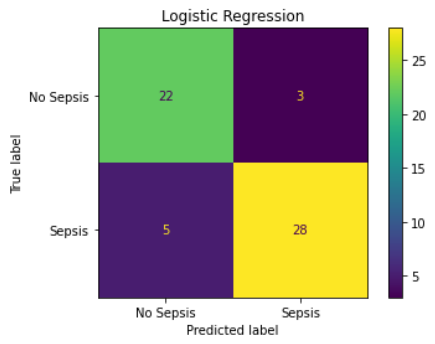

Figure 9: Final Dataset - Logistic Regression using kNN imputation.

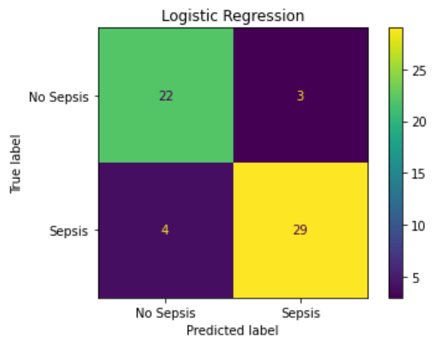

Figure 10: Final Dataset - Logistic Regression using each Feature's Mean.

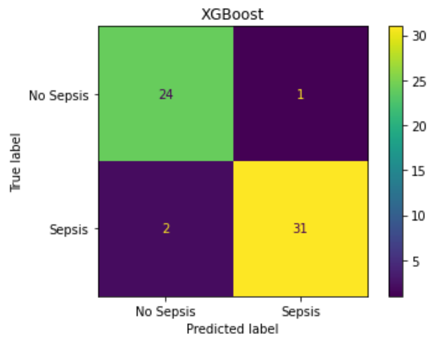

Figure 11: Final Dataset - XGBoost using kNN Imputation.

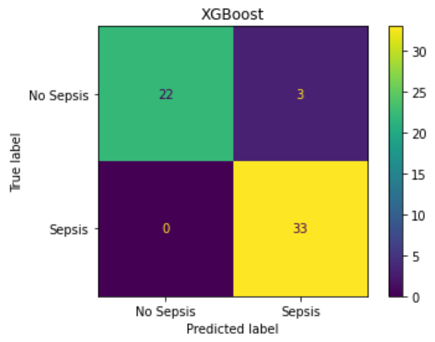

Figure 12: Final Dataset - XGBoost using each Feature's Mean.

As a first observation comparing these confusion matrices to the ones of the previous dataset, we can see that proportionally we have more false negatives, meaning our model predicted infected patients as non-infected. This make sense if we consider that the control group (i.e. patients that did not get infected during their hospitalization) was 3 times larger at this dataset, so the previously existing bias towards the positive class (infected) was now absent. As for the classifiers, we can see that in both data filling methods, Logistic Regression was the worst option with 7 and 8 errors while filling the missing values with kNN imputation or the feature's mean respectively. Gaussian NB was now the second best with 6 errors in each one of the data filling methods and the best performing classifier for this dataset was XGBoost, with 3 errors, 2 FNs (False Negatives) and 1 FP (False Positive) in the kNN imputation case and 3 FPs in the case where we take the mean of each feature to fill its missing values.

To proceed, we have some tables containing the evaluation metrics that will help us validate our prior observations based on the confusion matrices. Following the same format as the previous section, in tables 10, 12 and 14 we see the precision, recall and F1 Score for each class (positive and negative) of every classifier, while using kNN Imputation to fill the missing values. Respectively, in tables 11, 13 and 15 each classifier's precision, recall and F1 Score for both the positive and negative class is shown, but this time the mean value of each feature is used to fill this feature's missing values.

Gaussian NB kNN	Positive Class	Negative Class
Precision	0.9655	0.8276
Recall	0.8485	0.9600
F1 Score	0.9032	0.8889

Table 10: Final Dataset - Evaluation of Gaussian NB using kNN Imputation.

Gaussian NB Mean	Positive Class	Negative Class
Precision	0.9655	0.8276
Recall	0.8485	0.9600
F1 Score	0.9032	0.8889

Table 11: Final Dataset - Evaluation of Gaussian NB using each Feature's Mean.

LR kNN	Positive Class	Negative Class
Precision	0.9032	0.8148
Recall	0.8485	0.8800
F1 Score	0.8750	0.8462

Table 12: Final Dataset - Evaluation of Logistic Regression using kNN Imputation.

LR Mean	Positive Class	Negative Class
Precision	0.9062	0.8462
Recall	0.8788	0.8800
F1 Score	0.8923	0.8627

Table 13: Final Dataset - Evaluation of Logistic Regression using each Feature's Mean.

XGBoost kNN	Positive Class	Negative Class
Precision	0.9688	0.9231
Recall	0.9394	0.9600
F1 Score	0.9538	0.9412

Table 14: Final Dataset - Evaluation of XGBoost Classifier using kNN Imputation.

XGBoost Mean	Positive Class	Negative Class
Precision	0.9167	1.0000
Recall	1.0000	0.8800
F1 Score	0.9565	0.9362

Table 15: Final Dataset - Evaluation of XGBoost Classifier using each Feature's Mean.

Finally, in order to find the best model out of these 6, we need to compare their Macro F1 Scores, which incorporates all the previous measures. We can see the Macro F1 Scores of each model in Table 16 below.

Macro F1 Scores	Gaussian NB	Logistic Regression	XGBoost Classifier
Filling with kNN Imputation	0.8961	0.8606	**0.9475**
Filling with Feature's Mean	0.8961	0.8775	0.9463

Table 16: Final Dataset - Macro F1 Score.

Comparing this table (Table 16) to the Table 9, which contains the Macro F1 Scores of our models for the initial version of our dataset, first thing we notice is a generalized better performance of every model with the current dataset. More specifically, the biggest improvement is recorded by the Logistic Regression, even though it is once again the worst performing out of the 3. In general, every model with any classifier and any filling method in the final version of our dataset, significantly outperforms the corresponding model trained on the initial dataset, without a single exception. Furthermore, anyone can see that there is no essential difference between the 2 ways of missing value filling. Finally, the best fit for our model is indeed XGBoost as we had already figured out from the confusion matrices, with a very high Macro F1 Score.

At that time, taking into account the domain experts who advised us to prefer FPs over FNs, we considered that the best possible model was the one with the XGBoost classifier and filling the missing values with the mean of each feature as this model, presented 3 FPs and 0 FNs.

At this point, while searching through a variety of other studies, we studied about a the built-in algorithm in the XGBoost classifier [13], that allowed the classifier to fill by itself the missing values of a dataset, as we described in Section 2.1.4. So, we decided to test its efficiency in our case and put it into test. Basically, what we did was removing the filling of the missing values from the pre-processing of the data and just "feed" them, along with their missing values, to the classifier. As presented below using a confusion matrix in Figure 13 and the scores achieved on the evaluation measures used in Table 17, the performance of this model was outstanding.

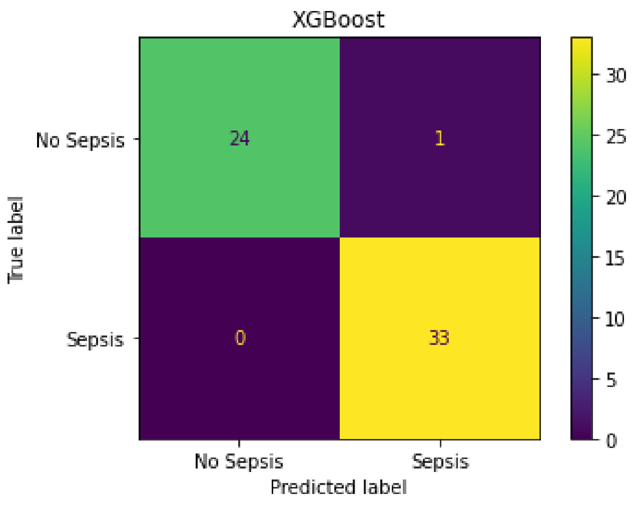

Figure 13: XGBoost without pre-processing filling of missing values

XGBoost self-fillng	Positive Class	Negative Class
Precision	0.9706	1.0000
Recall	1.0000	0.9600
F1 Score	0.9851	0.9796
Macro F1 Score	**0.9823**	

Table 17: Scores achieved from XGBoost, filled missing values using built-in algorithm

Not only does this model make just one false prediction, but it is in fact a False

Positive, which is the “preferable” kind of mistake, as stated before. So, even though comparing this model to the rest of the models analysed previously, this one seems to be the most efficient of all, there are still some more left to examine with the additional experiments that follow.

5.3.4 Time Series Clustering

Despite the already highly satisfactory results, we wanted to test one last change in our model. We were intrigued to see how our model would react if we were to take into account the time variable. On that basis, we decided to represent each one of the the 5-day medical measurements as a time series.

Visualizing it, we now had 19 different representations (since we have 19 different medical measurements) of 58 time series (for the 58 patient-instances), which we wanted to cluster. For the time series clustering, after some research we ended up trying Agglomerative Clustering and KMeans Clustering (for more details, see Section 2.1.7) and create 5 different clusters for each medical measurement. The clusters obtained, should be turned into 5 features to replace the initial 5 features which represented the 5 days of medical measurements. In order to do this, we decided to use One Hot Encoding. Thus, we ended up replacing every original 5 features with 5 one-hot encoded cluster assignments, always for the same kind of medical measurement. In Figure 14 and 15, we can see the confusion matrices for the models using Agglomerative and KMeans clustering respectively.

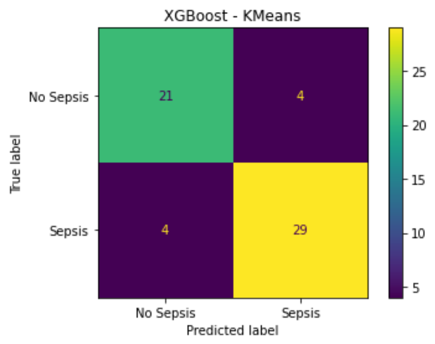

Figure 14: KMeans Clustering.

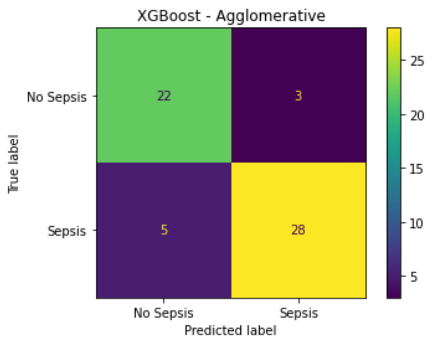

Figure 15: Agglomerative Clustering.

Observing these confusion matrices, we can see that compared to the previous models, the number of false predictions rises significantly. In fact, each one of the methods, makes a total of 8 mistakes. Although these matrices may seem pretty informative about the performance of the models with time series clustering, we also have to take a look at their scores. In table 18 we see the score for the performance of the model using KMeans clustering for a time series forecasting and in Table 19 we see how the one using Agglomerative clustering performed.

KMeans Clustering	Positive	Negative
Precision	0.8788	0.8400
Recall	0.8788	0.8400
F1 Score	0.8788	0.8400
Macro F1 Score	**0.8594**	

Table 18: Scores for model with KMeans Time Series Clustering.

Agglomerative Clustering	Positive	Negative
Precision	0.9032	0.8148
Recall	0.8485	0.8800
F1 Score	0.8750	0.8462
Macro F1 Score	**0.8606**	

Table 19: Scores for model with Agglomerative Time Series Clustering.

Indeed, these scores for both of the time series models, indicate a much weaker predictive capacity than most of the previous models. Something worth noticing is that the least efficient model that took the 5-days measurements as 5 separate features (Logistic Regression using kNN Imputation) had the same performance with the best performing model that clustered these features (Agglomerative Clustering) with a Macro F1 Score of 0.8606. From this, we could probably make the assumption that a time series analysis of these data, may sounded good at first, but as we can see, it gets significantly outperformed by other models and it does not seem to be an appropriate approach for this problem. However, we can not state that as a valid conclusion, since the relatively small size of the dataset, does not allow it.

5.3.5 Is Model's better performance Random?

There will always be the issue of the small dataset that is holding us back. So, we decided to implement a statistical test, to be precise a paired t-Test, which will help us see if the outcome of our experiments is accidental/random based on the dataset we have, or it will be performing better in almost every dataset. In general, "The Paired Samples t-Test is a non-parametric technique of statistics used to compare average values of two groups" [29]. With this test, we are going to compare our best performing model (XGBoost with the built-in algorithm for filling of missing values) with the two classifiers having the next best performance. These, based on the Macro F1 Scores, are the Gaussian NB and the Logistic Regression, both using the mean of each feature as a way of filling the missing values. In order to draw our results, we needed to examine two variables: T-statistic and p-value. The first one shows how much the models differ from each other. The greater the value is by absolute price, the more these models differ. Regarding the p-value, it represents the randomness of their difference. The lower the absolute value, the less the randomness in the results. Usually, to prove that the difference in efficiency between the models is not random, p-value has to be less or equal to 0.05.

Now we are going to explain the way we implemented this test. First of all, regarding the target variable, there is its actual value and the predicted value of each model. In our case, this of a binary classification, the actual value of the target variable is 0 for a patient who was not infected while being hospitalized, or 1 if the patient got infected. Respectively, if the model predicted that the patient would get infected, the predicted value would be 1, otherwise it would be equal to 0. Now, for each model, we created a list filled with 0 and 1, as many as the instances. For each patient-instance, we filled the list with 1 if the predicted value of the target variable was equal to the actual one, or else we filled it with 0. Now that we have a list for every one of the 3 models mentioned before, we compare our best one with the other two in pairs, using an existing python function for the paired T-test. The results we got are the following:

- Comparing XGBoost with Gaussian NB:

$$T - statistic \ = \ 3.7961 \quad , \quad p - value = 0.0004$$

- Comparing XGBoost with Logistic Regression:

$$T - statistic \ = \ 3.5916 \quad , \quad p - value = 0.0007$$

Based on the condition defining that p-value $< \ 0.05$ corresponds to a result that is not random, we can see that both of these comparisons between our tests, have a p-value much lower than the threshold of 0.05. Hence, it is clear that our best performing model is very possible to be the most efficient out of these 3, in almost every other similar dataset.

5.3.6 Confidence of Predictions

By the term “confidence”, we basically mean the probability our model gives for each instance’s target variable to be either 0 or 1. The default threshold in our case is 0.5, meaning that if an instance’s probability is larger than 0.5, the target variable’s value will be 1, otherwise it will be 0. Nevertheless, this threshold can change in certain cases. So, we computed each prediction’s confidence, to see if a threshold change could provide us with even better results.

After looking in detail every classes’ probabilities, we noticed that for an instance which belongs to the positive class (infected patient / probability > 0.5), the lowest probability given by the model is 0.5341. On the contrary, the highest probability for an instance of the negative class is 0.4302. We observe that both of these values are pretty close to the 0.5 threshold. At the same time, the false prediction is made with a probability of 0.9180 for an instance of the negative class. This is a prediction with a very big confidence, so there is no threshold change that could make our model perform better. It is worth noting that after seeing the probability our model gave for this prediction, we even contacted the domain experts to make sure the listing of this patient was correct. After some research, they reassured us it was indeed a patient who was not infected, even though our model suggested otherwise.

5.3.7 Feature Importance

Having concluded at our model with the XGBoost classifier using its built-in algorithm to fill missing values, something worth examining is the feature importance, to find out the features that had the biggest influence for our model to make the final prediction. These are presented in descending order in the Figure 16.

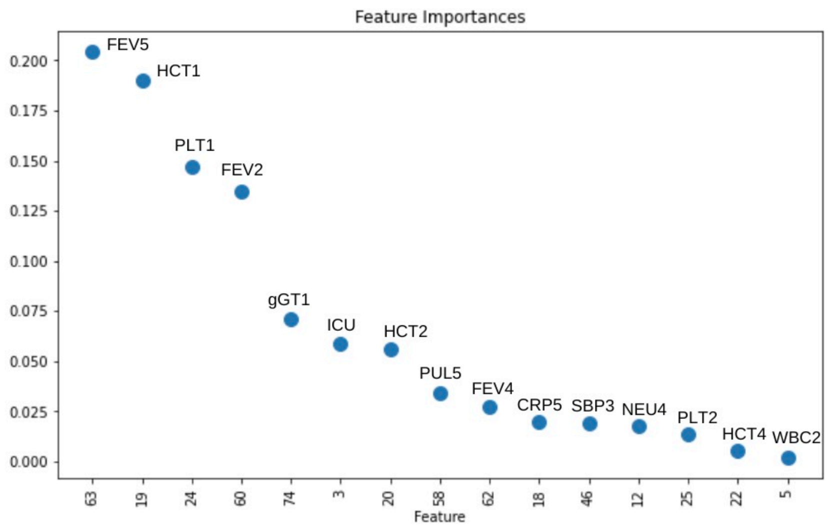

Figure 16: Feature Importance

First of all, looking at the plot, we can see that the nominal data had no influence on the decision of the model, something that of course may be due to the small amount of data. As for the features, it seems that the most important ones for the model were the fever (FEV), the hematocrit (HCT) and the Platelets (PLT). Regarding the days of measurements, even though the most important feature was a fifth day measurement, in general the first days and in particular the first day, appear to have greater importance. The most important feature of all, is the fever measurement on the day of the infection's occurrence. This was something that we expected, since it is known that fever is one of the most common symptoms of an infectious disease [30].

5.3.8 Features that could Give Away the Result?

At this point, we were concerned by the fact that every model we tested so far, appeared to be able to make a valid number of predictions correct. So now, we wanted to make sure that there is no feature in our dataset predetermining the outcome of the prediction. Our first move in this direction, was to consult the doctors of the General University Hospital of Heraklion, as domain experts. In the same time, we applied an exploratory data analysis (using an existing python function) on our dataset, that generated an HTML report which could be opened in the browser. In this report, called DataPrep Report, we could experiment with our data and find different statistics about them. We just had to look at the correlations between every single feature and the target variable. What we were looking for, was to make sure that there is not an obvious threshold in any of our features, where the instances to the one side of it, would all belong to one class, while the ones on the opposite side, to the other class. Both the reply of the doctors concerning this subject and the results of the exploratory data analysis, confirmed that there was nothing biased about our features, meaning that our model was totally legitimate and objective. Hence, we can continue our experiments for the analysis of our selected model.

5.3.9 Prediction with less days' measurements

At this time, having already concluded to arguably the best possible predictive model for our quest based on our data, we decided to take it one step further and try to see how our model would react if we extracted some days of medical measurements and by extension, if we could make a prediction with even less data. Thus, we tried to remove some features from our dataset, which represented certain days of medical measurements. For example, we wanted to see what would happen if we were to remove every measurement from the day of the culture that gave away the infection (day 5), or the first day of measurements etc. By doing this, we could see not only how well our model would perform with less measurements available, but primarily how many days before the occurrence of an infection was our model able to predict it. Also, it could provide us with an answer about the days' measurements that matter the most for the prediction: The ones close to the infection or the ones much prior?

- **Removing days from Day 5 to Day 1**

  To give an answer to this question, we started removing features from our dataset, beginning from the ones representing the day 5, which is the day of the culture that confirmed the infection. We continued removing features close to the day of the infection from our dataset until we were left only with the first day of medical measurements. In Figure 17 we see the predictions using the measurements from the days 1 to 4, in Figure 18 using days 1-3, in Figure 19 the confusion matrix for predicting with the first 2 days and in Figure 20 using just the first day. Finally, in Table 20, we see the scores of the aforesaid models.

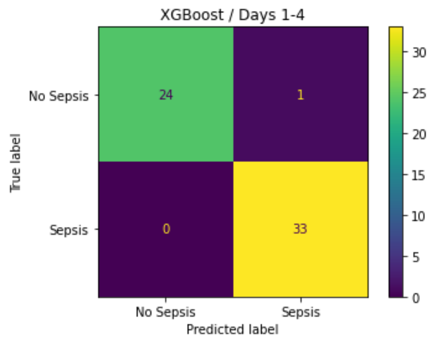

Figure 17: Predicting without measurements of day 5.

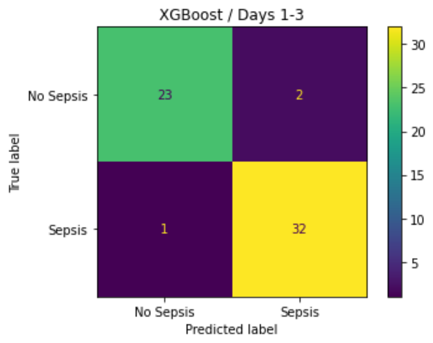

Figure 18: Predicting without measurements of days 4,5.

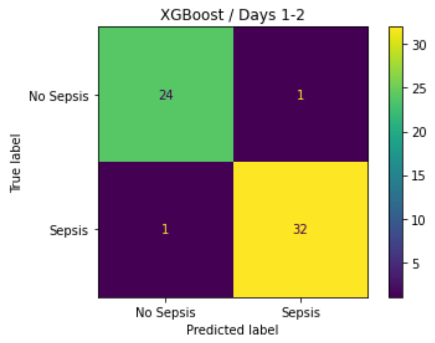

Figure 19: Predicting without measurements of days 3,4,5.

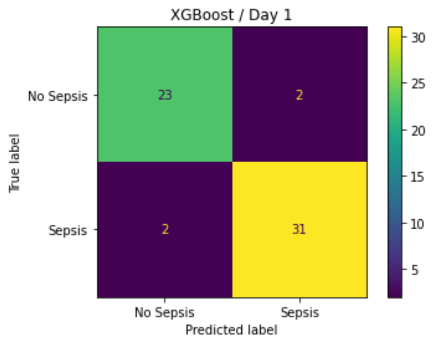

Figure 20: Predicting without measurements of days 2,3,4,5.

Days Considered	1-4		1-3		1-2		1	
Class	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
Precision	0.9706	1.0000	0.9412	0.9583	0.9697	0.9600	0.9394	0.9200
Recall	1.0000	0.9600	0.9697	0.9200	0.9697	0.9600	0.9394	0.9200
F1 Score	0.9851	0.9796	0.9552	0.9388	0.9697	0.9600	0.9394	0.9200
Macro F1 Score	**0.9823**		**0.9470**		**0.9648**		**0.9297**	

Table 20: Scores of our model while removing days backwards.

From the results, we uncover some important findings:

1. After removing the measurements from the day of the infection (day 5), the predictive ability of our model remains the same. This means that we can predict an upcoming infection without measurements from the day of its occurrence. So, we can efficiently make a prediction even a whole day prior to the infection.

2. Even though we took away the feature with the highest importance, the fever measurement of the fifth day, as mentioned in Section 5.3.7, the model was not affected.

3. When predicting with the measurements of 3 days (3,4,5) we get worse results than the prediction using the measurements for 2 days (4,5). This is possibly random and could be due to a variety of cases, such as the dataset, the confidence threshold we explained at Section 5.3.6 etc.

4. It is remarkable that even just with the first day of measurements, the model performs way better than expected, especially if we consider that it tries to predict an infection 4 whole days before it happens, with a Macro F1 Score almost equal to 0.93.

- **Removing days from Day 1 to Day 5**

  Here, we are going to do the opposite to what we did before. We are going to remove feature-days onwards, meaning we are going to remove features from day 1 to day 4 of measurements, until we are left just with medical measurements from day 5, the day of the infection. Just as before, there are 4 confusion matrices in Figures 21, 22, 23 and 24 that show the predictions of the models using just days 2-5, 3-5, 4-5 and 5 respectively. Then, there is Table 21 which contains all the evaluation measures scores related to these models.

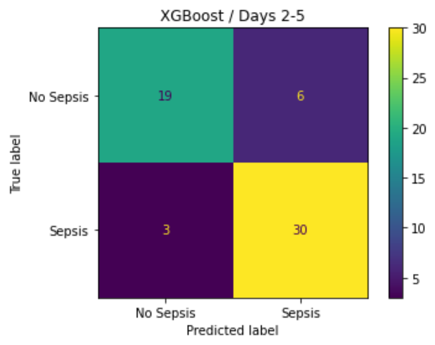

Figure 21: Predicting without measurements of day 1.

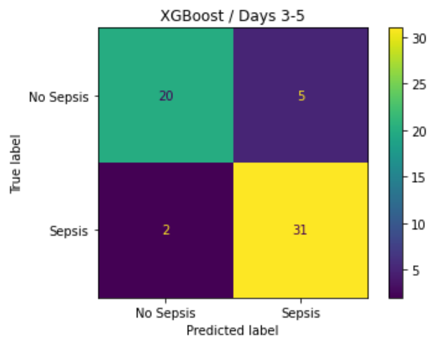

Figure 22: Predicting without measurements of days 1,2.

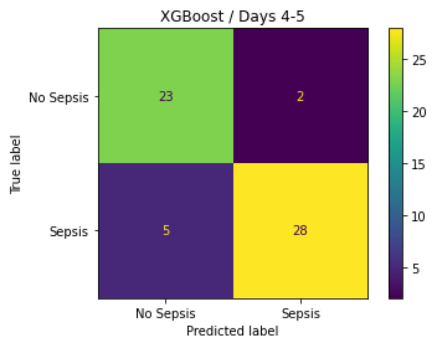

Figure 23: Predicting without measurements of days 1,2,3.

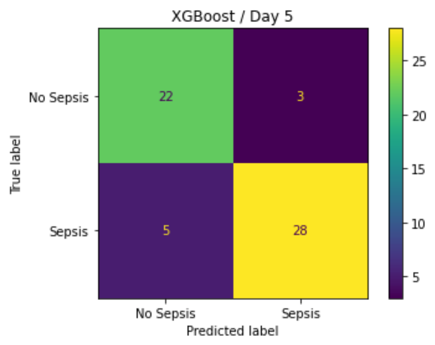

Figure 24: Predicting without measurements of days 1,2,3,4.

Days Considered	2-5		3-5		4-5		5	
Class	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
Precision	0.8333	0.8636	0.8611	0.9091	0.9333	0.8214	0.9032	0.8148
Recall	0.9091	0.7600	0.9394	0.8000	0.8485	0.9200	0.8485	0.8800
F1 Score	0.8696	0.8085	0.8986	0.8511	0.8889	0.8679	0.8750	0.8462
Macro F1 Score	**0.8390**		**0.8748**		**0.8784**		**0.8606**	

Table 21: Macro F1 Scores while extracting days onwards.

It is clear that the predictive capacity of this procedure is not even relatively close to the previous one, where we removed days backwards. These results highlight once more the significance of the first day of the measurements for the model.

To conclude, we can safely assume based on the above that it is easier to detect an upcoming infection if you consider the situation of the patient some days prior to the day of the infection, than the days close to it. Nevertheless, it is noteworthy that even in the worst case scenario, this of the prediction using the measurements of the days 2,3,4 and 5, our model had a Macro F1 Score of almost 0.84, making 9 false predictions out of 58. So, in the worst possible case, our model has a 84.5% chance of making the right prediction, which is relatively not so bad.

5.4 Discussion

In this section, we thoroughly described every single experiment performed. It is safe to say that the turning point in our work, were the additional data added in our dataset during our research, something that is obvious comparing the results between Sections 5.3.2 and 5.3.3. The vast majority of the combinations between the classifiers tested and the ways of filling missing data, produced models that performed surprisingly well, given the size of the available dataset. However, XGBoost seemed to be not only our selected classifier, but also having on average the best performance on the majority of the models trained. Even though the initial goal of this study was to have a time series analysis for a part of the data, this appeared to be the least efficient of all the methods attempted. Regarding the features, during our tests, although every model had of course different important features, there were many similarities. For example, fever, hematocrit, platelets and gGT appeared to have a high importance in most of our models. On the contrary CR, INR and APTT were not a factor of influence in any of our models. Concerning INR and APTT, this is maybe due to the big amount of missing data in their measurements, since these were the least completed among all the medical measurements. As for the creatinine (CR), it probably does not seem to be affected by the occurrence of infection. Also, none of the features containing nominal values were ever important for any model. Nonetheless, the small size of our dataset, reduces significantly our capability of generalizing these findings.

Something we consider to be a valuable result from our work, is the Section 5.3.9, where we tried to predict an upcoming infection with less days of measurements available in our dataset. This, revealed some really interesting findings that are challenging enough to be investigated even further. Based on these, this model can not only predict an upcoming infection at least a day before it happens, but it can predict an infection 4 whole days prior to its occurrence with a success rate of more than 93%, since it made only 4 wrong predictions our of 58. Of course, to achieve a satisfactory predictive capability with fewer days of data, is really important since having medical measurements from 5 consecutive days is really challenging. At the same time, predicting an upcoming infection as early as possible, could provide the domain experts with some important time to handle the patient's situation being better prepared. During these experiments, the important features were almost the same as in the previous ones. FEV, HCT, PLT and gGT of the first day were very important in every model tested in Section 5.3.9, while in the Section 5.3.9 where there was no data from the first day, the important features did not seem to follow some pattern and they were scattered among the dataset. However, even in these models, not even once CR, INR, or APTT had any influence in the outcome of a prediction. Finally, our final model, not only made just one false prediction on the available dataset, but it was the "preferable" kind of mistake based on the advice taken from the doctors of the University General Hospital of Heraklion.

In conclusion, our experiments can be described as successful since they all produced notable models and almost every change attempted, seemed to improve the model's performance. However, even though the relatively small size of our dataset will constantly be a barrier regarding the confidence we can show about our findings, we can consider that the problem posed to us has been successfully dealt with.

Chapter 6

6 Conclusion

In this final part of our study, we will make a summary of our work, highlighting the important findings, while also suggesting some potential changes that could improve our research in the future.

6.1 Summary of Work

In this study, we attempted to construct a machine learning model, able to predict impending sepsis. However, predicting an infection is a fundamental prerequisite for the early identification of sepsis, so our work was focused towards the timely detection of a possibly upcoming infection. In order to do this, we needed a dataset of patients, either having got infected during their hospitalization, or not. The neurosurgical clinic of the General University Hospital of Hearklion, responded to our request and provided us with a comprehensive dataset, containing a variety of features for each patient. We proceeded with data cleaning and data pre-processing over this dataset. The data pre-processing was a simple yet effective process of handling the nominal data existing in the dataset and normalizing every value within the range of 0 and 1. Then, the selection of the appropriate method for filling the missing values and the best possible classifier, emerged after several tests and model evaluations. This model was consisted of the XGBoost classifier using its built-in algorithm to fill the missing values, while all the features were handled as independent ones, without any specific treatment of the time series data. Next, the features that presented the highest importance in our work, were the fever, the hematocrit and the platelets, with the fever of the fifth day of measurements, having the biggest influence of all. As for the days, the first day of measurements appeared to be significantly more important, something that is also strongly confirmed from our attempt to remove medical measurements from certain days.

In general, it is important to note that our model can be applied every day to keep the medical team informed of each patient's health status. The medical personnel will be prepared in due time to treat the patient properly in the event of an infection, by administering the proper antibiotic treatment and having the patient concerned, undergo extra testing. In conclusion, our model can predict efficiently an upcoming infection days before it happens, in order to avoid impending sepsis and potentially save the life of a patient in the future.

6.2 Future Work

As already mentioned before, this paper's goal was to predict an impending infection, in order to avoid upcoming sepsis. To achieve this, we gathered data from the neurosurgical clinic of the General University Hospital of Heraklion. However, the dataset was nowhere near being large enough for our result to be as objective as possible. So, a dataset with many more patient-instances would surely give even higher value to our findings, especially if the real life analogy of 15 non-infected to 1 infected patient is achieved. Nevertheless, the dataset could not only be larger concerning the number of the instances. Since the days that are far from the infection seem to be more important, as we saw in Section 5.3.9, it would be interesting to see what happens if we get measurements from further back. So, we can talk about not only a growth of the dataset because of the addition of new patients, but also an enlargement due to records of more days of medical measurements for the existing patient-instances. Finally, it would be useful to test this model in other similar datasets to broaden the generalizability of our work, even compare it to others.

References

[1] M. Singer, C. S. Deutschman, C. W. Seymour, *et al.*, "The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)," *JAMA*, vol. 315, no. 8, pp. 801–810, Feb. 2016, ISSN: 0098-7484.

[2] A. Perner, A. C. Gordon, D. De Backer, *et al.*, "Sepsis: Frontiers in diagnosis, resuscitation and antibiotic therapy," *Intensive Care Med.*, vol. 42, no. 12, pp. 1958–1969, Dec. 2016.

[3] U. M. Wallgren, J. Sjölin, H. Järnbert-Pettersson, and L. Kurland, "The predictive value of variables measurable in the ambulance and the development of the predict sepsis screening tools: A prospective cohort study," *Scand. J. Trauma Resusc. Emerg. Med.*, vol. 28, no. 1, p. 59, Jun. 2020.

[4] M. Bauer, H. Gerlach, T. Vogelmann, F. Preissing, J. Stiefel, and D. Adam, "Mortality in sepsis and septic shock in Europe, North America and Australia between 2009 and 2019— results from a systematic review and meta-analysis," *Critical Care*, vol. 24, no. 1, p. 239, May 2020, ISSN: 1364-8535.

[5] S. Shashikumar, M. Stanley, I. Sadiq, *et al.*, "Early sepsis detection in critical care patients using multiscale blood pressure and heart rate dynamics," *Journal of Electrocardiology*, vol. 50, Aug. 2017.

[6] C. M. Torio and B. J. Moore, *National Inpatient Hospital Costs: The Most Expensive Conditions by Payer, 2013*. Agency for Healthcare Research and Quality (US), Rockville (MD), 2006.

[7] R. M. Sheykhmousa and M. Mahdianpari, "Support vector machine vs. random forest for remote sensing image classification: A meta-analysis and systematic review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, Oct. 2020.

[8] J. S. Calvert, D. A. Price, U. K. Chettipally, *et al.*, "A computational approach to early sepsis detection," *Comput. Biol. Med.*, vol. 74, pp. 69–73, Jul. 2016.

[9] Y.-Y. Kuo, S.-T. Huang, and H.-W. Chiu, "Applying artificial neural network for early detection of sepsis with intentionally preserved highly missing real-world data for simulating clinical situation," *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, p. 290, Oct. 2021, ISSN: 1472-6947.

[10] D. W. Shimabukuro, C. W. Barton, M. D. Feldman, S. J. Mataraso, and R. Das, “Effect of a machine learning-based severe sepsis prediction algorithm on patient survival and hospital length of stay: A randomised clinical trial,” *BMJ Open Respir. Res.*, vol. 4, no. 1, Nov. 2017.

[11] N. McLymont and G. W. Glover, “Scoring systems for the characterization of sepsis and associated outcomes,” *Ann. Transl. Med.*, vol. 4, no. 24, p. 527, Dec. 2016.

[12] M. L. Barreto, M. G. Teixeira, and E. H. Carmo, “Infectious diseases epidemiology,” *J. Epidemiol. Community Health*, vol. 60, no. 3, pp. 192–195, Mar. 2006.

[13] T. Chen and C. Guestrin, “Xgboost,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Aug. 2016.

[14] E. Vlachos, A. Salapatas Gkinis, V. Papastergiou, C. Tsitsipanis, and G. Giannakopoulos, “Machine learning to develop a model that predicts early impending sepsis in neurosurgical patients,” 2022, ISBN: 9781450395977.

[15] K. M and V. Akshaya, “A xgboost based algorithm for early prediction of human sepsis,” in *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2022, pp. 1643–1647.

[16] V. J. Ribas, A. Vellido, J. C. Ruiz-Rodríguez, and J. Rello, “Severe sepsis mortality prediction with logistic regression over latent factors,” *Expert Systems with Applications*, vol. 39, no. 2, pp. 1937–1943, 2012, ISSN: 0957-4174.

[17] E. Gultepe, H. Nguyen, T. Albertson, and I. Tagkopoulos, “A bayesian network for early diagnosis of sepsis patients: A basis for a clinical decision support system,” in *2012 IEEE 2nd International Conference on Computational Advances in Bio and medical Sciences (ICCABS)*, 2012, pp. 1–5.

[18] A. Shenoy and B. R. K, “Performance analysis of class imbalance handling techniques for early sepsis prediction using machine learning algorithms,” in *2022 8th International Conference on Smart Structures and Systems (ICSSS)*, 2022, pp. 1–6.

[19] K. K. Pandey, A. Giri, S. Sharma, and A. Singh, “Predictive analysis of classification algorithms on banking data,” in *2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON)*, 2021, pp. 1–5.

[20] Z. Chen, Y. Chen, T. Xiao, H. Wang, and P. Hou, “A novel short-term load forecasting framework based on time-series clustering and early classification algorithm,” *Energy and Buildings*, vol. 251, p. 111 375, 2021, ISSN: 0378-7788.

[21] V. Hautamaki, P. Nykanen, and P. Franti, “Time-series clustering by approximate prototypes,” in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1–4.

[22] K. Kirasich, T. Smith, and B. Sadler, “Random forest vs logistic regression: Binary classification for heterogeneous datasets,” 3, vol. 1, 2018.

[23] D. Khanna, R. Sahu, V. Baths, and B. Deshpande, “Comparative study of classification techniques (svm , logistic regression and neural networks) to predict the prevalence of heart disease,” 5, vol. 5, Oct. 2015.

[24] M. Stern, J. Beck, and B. Woolf, “Naive bayes classifiers for user modeling,” May 1999.

[25] N. James and J. Kaushik, “Comparison of machine learning algorithms for predicting chronic kidney disease,” in *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2022, pp. 1134–1139.

[26] R. Gusain, S. Sonker, S. K. Rai, A. Arora, and S. Nagarajan, “Comparison of neural networks and xgboost algorithm for music genre classification,” in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, 2022, pp. 1–6.

[27] R. K. Esfahani, F. Shahbazi, and M. Akbarzadeh, “Three-phase classification of an uninterrupted traffic flow: A k-means clustering study,” *Transportmetrica B: Transport Dynamics*, vol. 7, no. 1, pp. 546–558, 2019.

[28] S. Pasupathi, V. Shanmuganathan, K. Madasamy, H. R. Yesudhas, and M. Kim, “Trend analysis using agglomerative hierarchical clustering approach for time series big data,” *The Journal of Supercomputing*, vol. 77, no. 7, pp. 6505–6524, Jul. 2021.

[29] T. Riaji, S. E. Hassani, and F. E. M. Alaoui, “Application of paired samples t-test in engineering service-learning project,” in *2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC)*, 2022, pp. 1–4.

[30] J. J. González Plaza, N. Hulak, Z. Zhumadilov, and A. Akilzhanova, “Fever as an important resource for infectious diseases research,” *Intractable Rare Dis. Res.*, vol. 5, no. 2, pp. 97–102, May 2016.