Diagnosis of Bloodstream Infections Using Machine Learning

Hector Jakobsson, Erik Rydengård

Master's thesis
2024:E45

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Computer Vision and Machine Learning

Diagnosis of Bloodstream Infections Using Machine Learning

Hector Jakobsson & Erik Rydengård

Centre for Mathematical Sciences
Lund University

Supervisors:
Ida Arvidsson
Johanna Engman
Gustav Torisson
Oskar Ljungquist

Examiner:
Alexandros Sopasakis

June 13, 2024

Abstract

Bloodstream infections (BSIs) are among the top causes of death in Europe and as such, a serious health concern. Blood cultures are the most common method to diagnose this condition, bringing certain disadvantages. Mainly, it is time-consuming as it can take several days to get the test results back. Furthermore, the blood cultures carry a high risk of contamination. For this reason, a successful deployment of machine learning in the field could reduce arbitrary antibiotic usage and expedite correct treatment. This was the motivation for our thesis, where XGBoost, TabNet and a Multilayer Perceptron are used to predict blood culture outcomes.

The main question is which model performs the best on the provided dataset? This dataset contains vital measurements and laboratory results in tabular format. Furthermore, due to the required preprocessing of the raw data and handling of its missing values, which imputation method is most suitable? To answer these questions, we conduct a study where the models and multiple imputation methods are evaluated and compared. We find that XGBoost is the superior model, while imputing with median values and including missing indicators obtains the best results. This combination of methods obtained an Area under Receiver operating characteristic of 0.763 and an Area under Precision-Recall curve of 0.361. With this model and a threshold of 5%, the amount of blood cultures could be reduced by 29%, with the drawback that 1% of true positives are missed.

Acknowledgements

We would like to thank our supervisors Ida Arvidsson and Johanna Engman for their guidance and feedback regarding the report and project. Additionally, we would like to thank Gustav Torisson for interesting discussions at Skåne University Hospital in Malmö and for preparing the inital data, and Oskar Ljungquist for providing medical expertise and background. Lastly, we would like to thank Alexandros Sopasakis for taking on the role as examiner.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This report describes the exploration of diagnosing bloodstream infections using machine learning. In this first chapter, we will introduce the project and the aim of the thesis. In the following chapters, essential theory is presented, where the machine learning models and evaluation metrics are described as well as the preprocessing and its content. In the next section, the method is presented, with the results and the discussion of the findings following. Lastly, the final chapter states our conclusion.

1.1 Background

A bloodstream infection (BSI) is a serious health concern that is associated with a high mortality rate. The yearly cases of BSIs in Europe are estimated to be more than 1,200,000, resulting in deaths of approximately 157,000 patients (1).

A BSI is defined by a relevant pathogen in a patient's bloodstream causing disease (2). BSIs are commonly diagnosed by blood culture growth, which indicates a positive outcome. Due to the lack of other viable methods, blood cultures are seen as the best method for diagnosing BSIs (3).

The use of blood cultures is associated with two main disadvantages. Firstly, taking blood cultures is time-consuming and a positive outcome can take 24-48 hours to determine. Additionally, it can take 5-7 days to rule out a positive outcome, due to potential delay in bacterial growth. Secondly, blood cultures are likely to be contaminated, leading to false positive results. Research has indicated that 40-55% of all blood cultures with growths may be due to contamination. Due to these disadvantages, it has been reported that only 0.18-2.8% of decisions related to treatment are influenced by the outcome of blood cultures (4).

The absence of a viable diagnostic method for early detection of BSI impedes the possibility of optimal treatment. Finding a better diagnostic method could result in lower antibiotic usage and shorter hospital stays, as the right treatment can be used earlier.

1.2 Previous work

In the early stages of this thesis, a literature study revealed that no previous research has been conducted on predicting BSI outcomes from tabular data using artificial neural networks (ANNs). However, there is research that uses machine learning methods such as logistic regression and tree-based models like XGBoost.

In previous research, a comparison between an XGBoost and a logistic regression model was made (4). In that study, multiple datasets were provided by medical institutions from the Netherlands and the United States. The authors concluded that it was possible to predict BSI outcomes using clinical information such as age and sex together with vital and laboratory data. The study came to the conclusion that XGBoost was the superior model compared to logistic regression, achieving an Area Under the Receiver Operating Characteristics curve (AUROC) of 0.81 in a test set and up to AUROC of 0.8 in external validation. It was also concluded that if a threshold of 5% is set, then 30% of blood culture tests could be avoided, but in turn, 1% of true-positive outcomes were missed.

The lack of studies investigating ANNs, particularly those incorporating large datasets and including patients of all ages, has inspired the ideas behind this thesis.

1.3 Aim

The aim of this thesis is to evaluate if deep learning can be used to reliably diagnose bloodstream infections as an alternative to XGBoost. Furthermore, it will cover the process needed to go from raw medical data to a format suitable for machine learning models and analyze the impact of this preprocessing on the results.

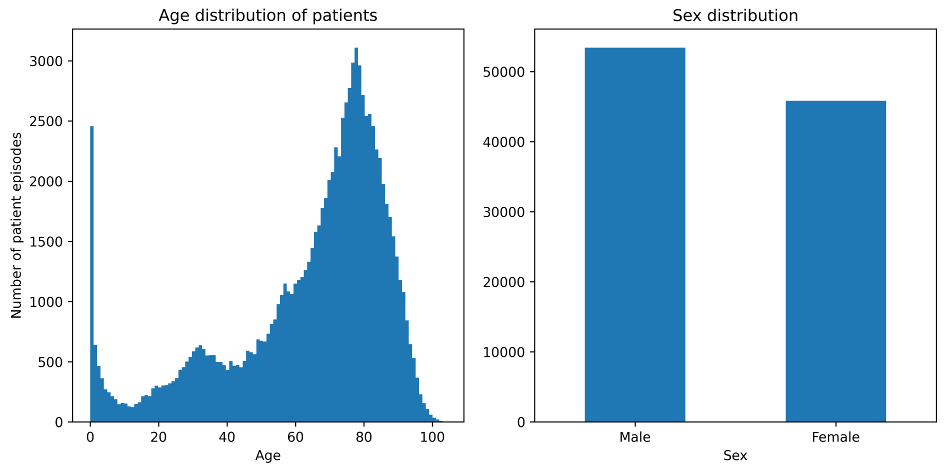
This gives rise to the following questions, which will be addressed in the thesis:

- Which imputation method performs the best?
- How does deep learning perform compared to XGBoost?
- Which features have a high impact on predictions?

Chapter 2

Data

Region Skåne provided the data used in this thesis, containing information about patients who underwent blood culture testing between 2021 and 2023. The data had been pseudo-anonymized, meaning personal information such as social security numbers had been removed and replaced with a patient ID to protect patient privacy. However, the data is not fully anonymized, meaning that it needs to be handled with care, as it is possible to figure out a patient's identity given the data.



(a) Figure showing the age distribution for patient episodes. Notably, it can be seen that the majority of the patients are above 50 years old, with a mean age of 64 years.

(b) The distribution of sexes for patient episodes. Males are slightly over-represented, accounting for 53 % of the episodes.

Figure 2.1: Distribution of age and sex in the data.

The data came divided into three separate datasets, containing vital measurements, laboratory results and outcomes respectively. The datasets, described in further detail below, consist of results for approximately 73,800 unique patients.

During 2021-2023, these patients corresponded to about 100,000 patient episodes. An episode is defined as a period of hospitalization. During each patient's stay, about 900 different tests can be performed; however, in most cases, a majority of the tests are not done.

The study population is broad regarding age, which can be seen in Figure 2.1a, with patients ranging from newborns to 105-year-olds. However, it is worth noting that 75% of the patients are older than 53 years old, with the mean age being 64 years old.

As seen in Figure 2.1b, the probability of a patient being male is slightly higher than that of a female, with males accounting for 53% of the episodes in the data.

Below follows a more detailed explanation of the contents of the datasets used in this thesis.

2.1 Vital measurements dataset

One dataset contains vital measurements like respiratory rate and pulse rate. These measurements are typically taken many times during a patient's stay.

2.2 Laboratory result dataset

An additional dataset contains various laboratory results, such as C-reactive protein and P-sodium. Typically, multiple tests are taken to account for contaminants or faulty tests, leading to multiple entries representing the same result. Before this thesis, an initial preprocessing of this data had been done, where these duplicates were removed.

2.3 Outcome dataset

The outcome dataset contains the outcomes from blood culture testing. It also contains general information about each episode, such as the patient's sex and age. Additionally, it includes times of importance, such as the start and end of hospitalization and the time for when a patient was subject to a blood culture, referred to as the baseline.

In some cases, the baseline is not recorded. Figure 2.2 shows the different ways the baseline has been set. In most cases, the exact culture time is provided, but in some cases, the time is gathered from other sources. Below follows an explanation of the sources which the baseline comes from:

Culture time provided: The exact baseline is available.

From laboratory value: It is unlikely that a blood culture test was not taken when other blood-related laboratory tests were taken. The baseline is set to the same time as when other tests were done.

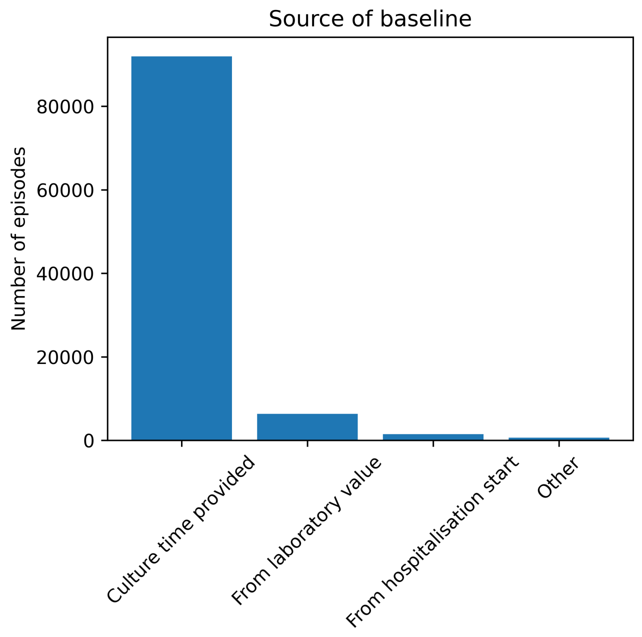


Figure 2.2: Source of baseline in data. The columns describe the source from where the baseline of a patient episode was taken. It can be seen that the majority of episodes already had supplementary baselines, while the second most frequent origin of baseline was extracted from when other blood-related laboratory tests were taken.

From hospitalization start: If there are no laboratory results, the blood culture test would likely have been taken at the start of hospitalization.

Other: In this case, there is often just a date associated with the episode. The baseline is set as the available date with time 00:00:00.

As mentioned in the introduction chapter, a blood culture test can give a false positive result if the blood culture is contaminated. The process of determining the outcome of a blood culture is quite advanced, meaning that individual decisions are taken based on which bacteria is found and in how many blood culture sets, consisting of two bottles, the bacteria is found. To account for this, the dataset contains two different outcomes:

- **Outcome**: The finding of a relevant pathogen in at least one blood culture set, or a potential contaminant in more than one blood culture set, leads to a positive outcome.
- **Outcome without contaminants**: Contrary to the outcome above, the finding of a potential contaminant in a blood culture set leads to a negative

outcome.

The motivation behind constructing an outcome without contaminants is that a blood culture can be contaminated despite a potential contaminant not being present in more than one blood culture set, potentially leading to an overestimation of the number of relevant positive outcomes.

Chapter 3

Models and evaluation metrics

There are multiple types of machine learning models, based on different concepts and architectures. In this chapter, we present the relevant theory to cover the models used in this thesis.

3.1 Extreme Gradient Boosting

XGBoost (5), short for Extreme Gradient Boosting, is a gradient boosting system based on decision trees. It has been deployed in a range of machine learning challenges, achieving state of the art results.

With the XGBoost algorithm, new trees are constructed and added iteratively based on earlier trees, where the training of the model consists of minimizing the following objective function for all N data points and K decision trees:

$$Obj = \sum_{n=1}^{N} L(\hat{y}_n, y_n) + \sum_{k=1}^{K} \Omega(f_k), \tag{3.1}$$

where $L(\hat{y}_n, y_n)$ is the loss function with true labels y_n and predicted labels $\hat{y}_n$. The loss function that will be used for the XGBoost model in this thesis is Binary Cross Entropy, explained in Section 3.2.2 further down. The second term in the equation is the regularization term, with $\Omega(f_k)$ representing the complexity of a tree f_k. The motivation for the inclusion of this term is to prevent over-fitting by penalizing the complexity of the model. The function for a single tree, f_k, includes its structure and leaf values. The set of functions f_k are the parameters that are learned during the training. Output predictions $\hat{y}$ from the model are calculated as the sum of the predictions from all trees K, according to

$$\hat{y} = \sum_{k=1}^{K} f_k(\mathbf{x}). \tag{3.2}$$

3.2 Artificial Neural Networks

There are many types of Artificial Neural Networks (ANNs), and which one is suitable for use depends on the data. The preprocessed data used in this thesis is non-sequential. For this reason, a simple Multilayer Perceptron (MLP), or feed-forward neural network as it may also be called, is deemed feasible for the work at hand.

Additionally, a second more complex type of ANN, called TabNet, will be used. This model will be described further down.

3.2.1 Activation functions

Activation functions are used in ANN to introduce non-linearity; without it, there would be no point in having multiple layers, as the output would be a linear combination of the weights and input, which can be represented using just one layer.

In this thesis, three different activation functions with certain properties will be used. Below are descriptions of each of them.

Rectified Linear Unit

Rectified linear unit or ReLU is a widely used activation function that can be applied to all layers but the input and output layers (6). ReLU is presented as

$$ReLU(x) = \max(0, x). \tag{3.3}$$

From the formula, it can be seen that the ReLU function preserves all positive input values while negative values are set to zero.

Sigmoid

The sigmoid function converts the output to a value between zero and one, which can be interpreted as a probability. This property of the function makes it suitable for use as the activation function in binary classification tasks. Therefore, it will be the activation function of choice in the output layer since the output will consist of a single value representing the probability of a positive outcome for BSI. The formula for the sigmoid function is defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

for input x.

Gated Linear Unit

The Gated Linear Unit, or GLU, is an activation function which uses a gating mechanism for controlling how much of the input should be let through. The GLU function is defined as

$$GLU(\mathbf{x}) = (\mathbf{x} * \mathbf{w} + b) \otimes \sigma(\mathbf{x} * \mathbf{v} + c), \tag{3.5}$$

where $\mathbf{x}$ is the input and $\mathbf{w}$, $\mathbf{v}$, b and c contains trainable parameters which are adjusted during training.

Sparsemax

The sparsemax activation function is closely related to the softmax activation function, which is commonly used in the output layer of models for multi-class classification (7). Both sparsemax and softmax have the property that the output is a probability distribution, meaning the output is both positive and sums to one. The difference between them is that the sparsemax output can be sparse, which is crucial when creating a mask that should filter features. The formula can be seen below:

$$\text{sparsemax}(\mathbf{x}) = \underset{p \in \Delta^{K-1}}{\text{argmin}} \|p - \mathbf{x}\|^2, \tag{3.6}$$

where K is the number of features in $\mathbf{x}$ and Δ^{K-1} refers to the $(K-1)$ dimensional probability simplex. The output of the sparsemax function is the point on the probability simplex closest to the input by euclidean distance. It is often the case that the closest point on the probability simplex will be at the boundary of the simplex, resulting in the output being sparse.

3.2.2 Loss function

Loss functions measure the difference between the output value and target value. While there are many different loss functions, Binary Cross Entropy (BCE) (8) will be used in this thesis. In this case, binary implies that the target value should be binary, meaning either zero or one. The function is defined as

$$L = -\frac{1}{N} \sum_{n=1}^{N} y_n \cdot \log(\hat{y}_n) + (1 - y_n) \cdot \log(1 - \hat{y}_n), \tag{3.7}$$

where y_n is the labels found in the outcome and $\hat{y}_n$ is the model output values, for a batch n and number of batches N.

3.2.3 Learning phase

The learning phase in machine learning refers to the phase where the weights of the model are updated to minimize the loss function, carried out by optimization. The basis for many optimization algorithms, and probably the most commonly used, is Stochastic Gradient Descent (SGD).

With this algorithm, the weights are updated for subsets of the dataset, called minibatches. The average gradient is calculated for a randomly selected minibatch. When all the batches have been processed, one epoch has been completed. This method of more frequently updating the weights, compared to updating for the whole dataset, speeds up the process and reduces computational time.

The gradient for a minibatch of size N is calculated as

$$\hat{g} = \frac{1}{N} \nabla \sum_{n=1}^{N} L(\hat{y}, y), \tag{3.8}$$

for a loss function L, predicted labels $\hat{y}$ and true labels y. Accordingly, the weights are updated as

$$w = w - \epsilon \hat{g}, \tag{3.9}$$

for a learning rate ϵ, which determines the step size and affects how fast the function converges.

The algorithm that will be used in this thesis will be Adam (6) (deriving from adaptive moments), a further development of SGD. Contrary to SGD, this algorithm introduces an adaptive learning rate, meaning that the learning rate is divided into individual rates for different weights which are adjusted while training. Furthermore, Adam incorporates the use of momentum. In short, this means that the weights are not only updated based on the previous gradient, but a sequence of previous gradients. This is carried out through the use of a velocity variable that increases when successive gradients have similar values, thus giving the learning algorithm "momentum".

3.2.4 Multilayer Perceptron architecture

The overall architecture in the MLP is resembling that of a human brain. It consists of multiple layers of neurons, or nodes, that are connected by weights. The first layer is called the input layer and the last layer is called the output layer, while the layers in between are called hidden layers. There is always at least one hidden layer in an MLP.

The output y from each node in a layer is the sum of the weighted input $\mathbf{x} = (x_1, x_2, ..., x_n)$ with weights $\mathbf{w} = (w_1, w_2, ..., w_n)$ and bias b. Finally, this sum is put through an activation function f. This gives the following mathematical formula

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right). \tag{3.10}$$

In Figure 3.1, a visualisation of an MLP with two hidden layers is shown. Here, h_m are the calculated node values that are used as input for the next layer.

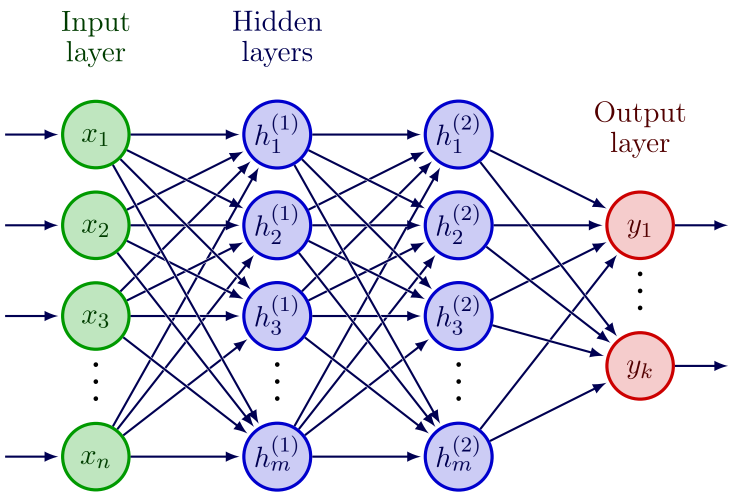


Figure 3.1: Visualisation of a multilayer perceptron

3.2.5 TabNet

TabNet (9) is a deep learning architecture developed by Google which is specifically designed to work with tabular data. The architecture uses sequential attention which makes it possible to adjust the features used at any given step based on the input features. This means that there is no need to do feature selection, that is reducing the number of features by selecting the ones deemed most impactful, before training the model as the model can learn which features to use.

One of the main ideas with TabNet is to introduce interpretability for tabular deep learning models. Interpretability is important, especially when the model should be used in healthcare. A physician would like to know which features contributed to a specific prediction. TabNet provides two types of interpretability:

Local interpretability: Provides insights into which features contributed most to a specific prediction.

Global interpretability: Provides insights into which features are most important to the model during training.

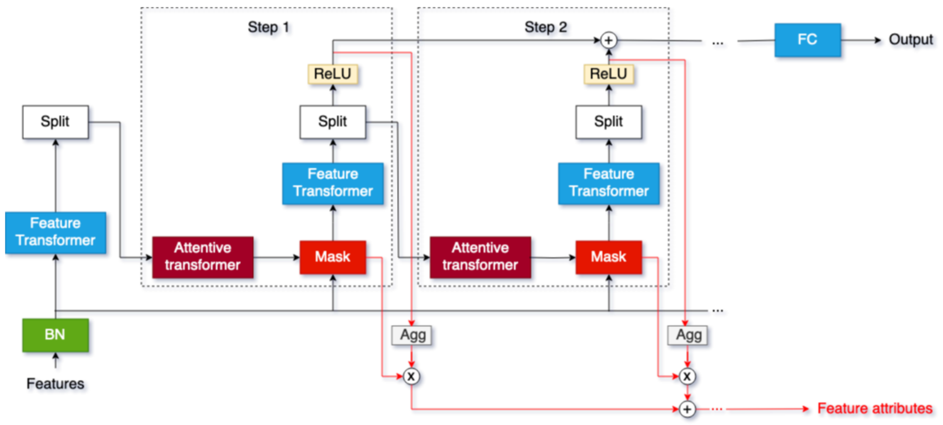


Figure 3.2: Figure showing the architecture of TabNet.

In Figure 3.2, it can be seen that the TabNet architecture contains multiple repeating steps. Understanding one step means that it is possible to understand the global architecture. It is essential to understand two components; the attentive transformer and the feature transformer which will be described in detail below.

Feature Transformer

The feature transformer has two responsibilities:

- Create an output of the dimension n_d used for decision step.
- Create an output of dimension n_a which is used as input to the next attentive transformer, described below.

The feature transformer consists of multiple GLU blocks as can be seen in Figure 3.3. Each block includes a fully connected (FC) layer with batch normalization (BN) and a GLU activation function. A specific type of BN called Ghost Batch Normalization (GBN) is used. Unlike regular BN, where weights are updated based on entire batches, GBN updates parameters using smaller parts of each batch, known as virtual batches.

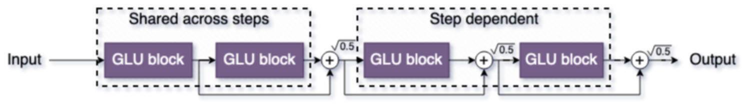


Figure 3.3: Figure showing the architecture of the feature transformer as described in the TabNet research paper. The output of each addition is normalized by a factor $\sqrt{0.5}$ to stabilize the learning.

The weights in the FC layers in these GLU blocks can be shared between feature transformers. In the original TabNet paper, each feature transformer contains four GLU blocks, two of which share weights between steps.

Attentive Transformer

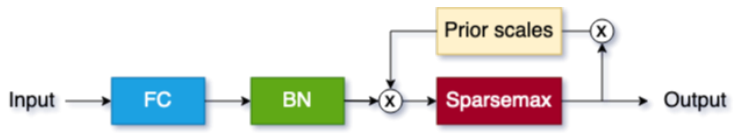


Figure 3.4: Figure showing the architecture of the attentive transformer.

The purpose of the attentive transformer is to generate a mask that filters features for the next step in the model. The architecture of the attentive transformer is shown in Figure 3.4. Notably, each attentive transformer contains a block with an FC layer and a BN layer, using GBN. The output of the BN layer is piecewise multiplied by a prior scale P and transformed into a mask, using the sparsemax function.

The prior scale P is calculated as follows:

$$P_0 = 1,$$
$$P_i = \prod_{j=1}^{i} (\gamma - M_j),$$

where P_0 represents how much each feature has been used at step 0, and P_i is the prior scale from the previous steps. The hyperparameter γ is a relaxation parameter that controls how many times a feature can be utilized. If $\gamma = 1$, then a feature can only be used once. Setting γ to a higher value allows features to be used across multiple steps. The variable M_j corresponds to the mask used at a previous step. For example M_2 corresponds to the mask generated by the attentive transformer at step 1 in the architecture.

3.3 Evaluation metrics

Evaluation metrics will be used to evaluate and compare the performances of XGBoost, MLP and TabNet. There is a multitude of different metrics and since they each provide specific information, a relevant selection will be used in this project.

3.3.1 Precision and recall

A binary dataset can be divided into positives (P) and negatives (N) (10). When a binary classification model has been used to make predictions on this dataset, the outcome labels can then be compared to the dataset and its actual values. This comparison results in four different categories:

- True positives (TP) are labels that are predicted as positive and the real values are also positive.
- False positives (FP) are labels that are predicted as positive but the real values are negative.
- True negatives (TN) are labels that are predicted as negative and the real values are also negative.
- False negatives (FN) are labels that are predicted as negative but the real values are positive.

Precision and recall are then defined as:

$$Precision = \frac{TP}{TP + FP} \tag{3.11}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.12}$$

In other words, precision can be seen as the fraction of positive predictions that were correctly predicted, while recall can be seen as the amount of positives that were covered or included in the prediction.

From a medical standpoint, both high precision and high recall is sought-after. High precision means that patients are not incorrectly predicted to have a BSI, which results in fewer unnecessary tests. High recall means that patients that do have a BSI are not incorrectly predicted to not have one. Having a low recall could be fatal since it can result in patients going undiagnosed.

3.3.2 AUROC

The Area Under the Receiver Operating Characteristics (AUROC) (11), or simply the area under the ROC curve, which in turn will be explained below, is an evaluation metric that can be interpreted as the ability to distinguish between positive and negative classes.

The ROC curve is constructed with the False Positive Rate (FPR) on the x-axis and True Positive Rate (TPR) on the y-axis, and is based on the continues output for all possible threshold values. The threshold value determines when a sample is classified as positive or negative based on the output of the model.

TPR is equal to recall, which was defined in equation 3.12 above. The FPR is defined as

$$FPR = \frac{FP}{TN + FP} \tag{3.13}$$

An ideal model, being able to perfectly classify samples, would have a TPR of one and a FPR of zero for every data point, leading to the curve in blue colour seen in Figure 3.5. Consequently, for a model that has no classification capacity whatsoever, the TPR and FPR would be equal for every data point. This scenario is described by the red-coloured curve in Fig. 3.5.

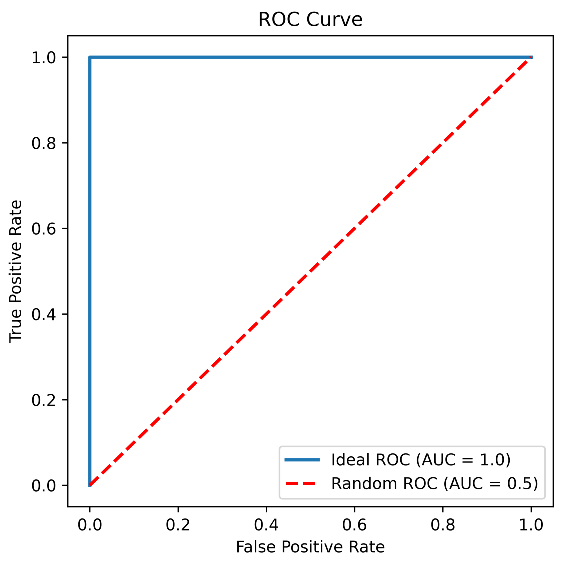


Figure 3.5: The ROC curves for an ideal model, in blue colour, and a model which randomly classifies samples, in red colour, and their respective AUROC values.

As mentioned earlier, AUROC is the area under the ROC curve, ranging from zero to one. As illustrated in Figure 3.5, an ideal model would have an AUROC value of 1.0 while a random model would have a value of 0.5. On the other hand, a model which misclassifies all samples would have an AUROC value of 0.0.

3.3.3 AUPRC

Similar to AUROC, in that they both represent areas under curves plotted across all possible threshold values, the Area Under the Precision-Recall Curve (AUPRC) (10) is an evaluation metric that represents the relationship between precision and recall, which were defined in Equation 3.11 and Equation 3.12 respectively. Contrary to the ROC curve, the Precision-Recall Curve (PRC) is plotted with recall on the x-axis and precision on the y-axis.

It is of interest to observe both precision and recall, and the trade-off between them, since if isolated, they can potentially provide uninformative results. For instance, a model classifying every sample as positive would give a perfect recall of 1.0 while the precision would be low.

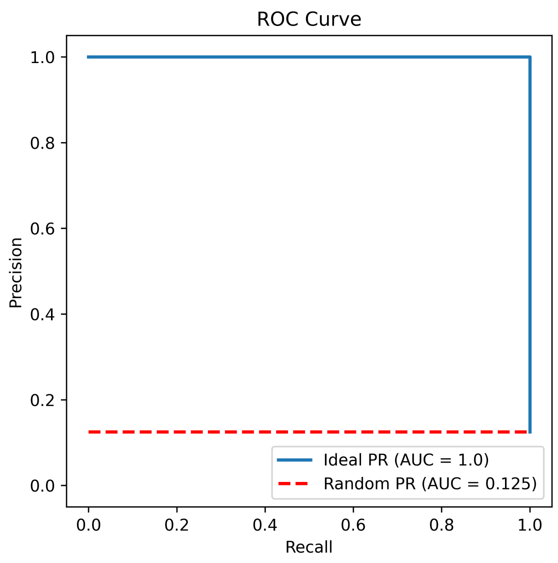


Figure 3.6: The PR curves for an ideal model, in blue colour, and a model which randomly classifies samples, in red colour, and their respective AUPRC values.

The AUPRC is the area under the PRC. While the baseline value for a random model is 0.5 for AUROC, the baseline for AUPRC is equal to the proportion of positive samples in the dataset. As an example, for a dataset consisting of 12.5 % positive samples, the AUPRC for a random model would be 0.125. This scenario is exemplified in Figure 3.6 above, where the curves of an ideal and a random model are drawn.

3.3.4 SHAP

The SHapley Additive exPlanations (SHAP) (12) is a way of calculating the contributions of specific features to predictions. In this thesis, SHAP values will be used to display feature importance for XGBoost.

Chapter 4

Preprocessing

In this chapter we will provide the requisites for understanding the preprocessing used in this thesis, namely imputation and standardization.

4.1 Imputation

For models such as TabNet and MLP, missing values need to be replaced with an estimate, which is done through imputation. Out of the models used in this thesis, XGBoost is the only model that supports handling of missing values. In short, this is carried out through trial and error, where the model learns the optimal decision to make when a missing value is encountered (5).

When deciding which imputation method to use, it is important to understand why values are missing. Missing values are usually divided into three cases (13). A description of each case can be seen below.

- **Missing completely at random**: Missing values can not be explained by any features, unobserved or observed. This would, for example, be the case if a test result is missing due to some kind of malfunction with the test equipment.
- **Missing at random**: Missing values are explained by observed features. An example of this would be if other tests indicate that the patient does not have BSI so additional testing is not necessary.
- **Missing not at random**: Missing values are explained by unobserved features. This would be the case if a physician chooses not to test based on unobserved features such as intuition or experience.

It is unclear if the missing values in the datasets can be classified as **missing at random** or **missing not at random** as it is unknown to what degree a physician bases decisions on, whether it is information present in the data or not.

Previous research has shown that no single imputation method works for all types of data (13). Many of the imputation methods assume that the data belongs to one of the cases above. It has been shown that missing indicators can be used to improve performance in some cases. Below are explanations of imputation methods used in this thesis.

4.1.1 Mean and most frequent imputation

One of the most straightforward methods for handling missing data is to impute with the mean or most frequent value, depending on the feature type. For continuous values, missing values can be replaced with the average of the non-missing values, while for categorical values, empty values can be replaced with the most frequent value.

4.1.2 Median and most frequent imputation

Missing values can be imputed with median instead of mean values. This approach can be suitable when the distribution of the values is skewed, meaning that the distribution is not symmetric (14).

4.1.3 Multivariate Imputation by Chained Equations

Multivariate Imputation by Chained Equations (MICE) is an advanced imputation method that utilizes machine learning algorithms to impute missing values using the values that are not missing in a dataset. MICE assumes that the missing data is **missing at random** (15).

The following steps are done in MICE (16):

- **Step 1:** All missing values are temporarily replaced with a standard value such as mean.
- **Step 2:** A feature is chosen to be imputed so all missing values for that feature is set back to missing.
- **Step 3:** A predictive model is trained on other features to be able to predict the missing values in the chosen feature.
- **Step 4:** The predictions of the trained model is used to replace the missing values in the chosen feature
- **Step 5:** Steps 2-4 are then repeated until there are no missing values left for any features.

The steps in Mice can be repeated multiple times to potentially produce better imputations.

4.2 Missing indicator

Each imputation method can be extended to include indicator variables indicating if a value was missing before the data was imputed. This means, that for every feature in the dataset, a new feature is added, containing zeros (the value in the corresponding feature column **was not** missing before imputation) and ones (the value in the corresponding feature column **was** missing before imputation).

4.3 Standardization

Standardization was applied to the dataset to transform each feature to have a mean of zero and a standard deviation of one. The formula for this is

$$y(x) = \frac{x - \mu}{\sigma}, \tag{4.1}$$

where μ is the mean and σ is the standard deviation of a feature x. This function is applied on every value for each feature in the dataset.

The motivation for standardizing the dataset is to unify the values between features, since they are measured in different scales and units. In turn, this reduces the risk of giving unintended weight to features based on their initial distribution.

Chapter 5

Method

The methodology in this thesis consisted of two major parts. The first step, described in Section 5.1, was to preprocess the dataset to make it suitable for use with the models. The second step was to construct the different models, presented in Section 5.2. The Pandas library was used to handle the large datasets, while the XGBoost and PyTorch libraries were used to implement XGBoost and the MLP. For TabNet, an open source solution implemented in PyTorch, called pytorch-tabnet (17), was used, chosen for its simplicity and PyTorch compatibility.

5.1 Preprocessing

The data used in this thesis was given in raw form. As a consequence, the data needs to be processed before it can be used. This chapter will explain the steps taken to transform the data into a representation that can be used with machine learning methods.

5.1.1 Combining datasets

This section explains how the multiple datasets used in this thesis were combined. The raw data for vital signs and laboratory results are sequential, with each row associated with the time the result was taken. The models used in this thesis can not make predictions on sequential data. To make the data compatible with the models, each row should be associated with a patient episode, containing all relevant results taken during that episode. The following steps were taken to remove irrelevant data and change the data format.

1. **Splitting the data by patients:** The data was divided into parts, each containing data for a specific patient.
2. **Splitting patients data by episode:** Each patient's data was further divided into parts containing data for a specific episode.
3. **Removing irrelevant data:** Only the data collected during the time frame when a patient typically waits for a blood culture test is considered useful.

This time frame was determined to start six hours before baseline and last until 48 hours after.

After following the steps described above, the data could be combined based on patient episodes. Additional information, such as sex, age, and the blood culture outcome corresponding to the episode, was also added.

The next step is to structure the data so that each row corresponds to a specific episode, and the columns represent the features. The data for each episode is transformed into a feature vector defined as follows:

$$\mathbf{x} = \begin{bmatrix} f_1 & \cdots & f_n \end{bmatrix}, \tag{5.1}$$

where n is the total number of possible features that a patient can have. The feature value f is either the value of an available result or missing.

The feature vectors are then combined into a feature matrix $\mathbf{X}$, defined as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_m} \end{bmatrix}, \tag{5.2}$$

,

where m is the total number of patient episodes.

In the same way, the ground truth vector $\mathbf{y}$ was created where the value in each row corresponds to the outcome of an episode:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{5.3}$$

,

This step concludes that the matrix $\mathbf{X}$ and vector $\mathbf{y}$ have been created and are in a format suitable for machine learning models. The values are still in raw form, meaning that values are the same as those in the raw data.

5.1.2 Parsing

Apart from the datasets being combined, additional steps must be taken to transform $\mathbf{X}$ and $\mathbf{y}$ to only include numerical or missing values. Furthermore, the values in the dataset have entry errors, such as misspellings or inconsistent use of decimal separators. As the dataset is quite large, a systematic approach is required to handle values in the data.

After analyzing the data, a few cases were decided on that must be dealt with. The cases can be summarized as follows:

- Handle cases where commas are used instead of dots.

- Values contain not only numerical values but also extra text such as units of measurement.
- Value is misspelled.
- Value contains logical operator.

If cases cannot be handled, the value should be replaced with a missing value.

This step concludes that the matrix $\mathbf{X}$ and vector $\mathbf{y}$ contain values that are either numerical or missing.

5.1.3 Outlier removal

The parsing of values may result in unreasonable values that are not valid. For instance, there may be patients with temperatures exceeding 1000 °C after parsing, mainly due to faulty input or typographical errors.

An expert was consulted to review the values of the features in the dataset, as most of them require medical expertise to be interpreted. After examination, the expert determined that most features contain reasonable values. However, certain features contain outliers. The expert then provided a range of valid values for each feature that includes outliers. If a value falls outside of the range, then it is replaced with a missing value. The provided ranges can be seen in Table 5.1.

Table 5.1: Ranges for valid values in selected features.

Feature	Min	Max
Temperature	30	42
Oxygen saturation	40	100
Total points NEWS	0	25
Systolic blood pressure	30	300
Diastolic blood pressure	20	200
Pulse	10	300
Respiratory rate	5	80
P-sodium	90	–

5.1.4 Data reduction

The models should be trained on features that are present in many patient episodes. Figure 5.1 shows a histogram of the ratio of missing values in patient episodes. It can be noted that most of the episodes contain few non-missing values.

After consulting with an expert, it was decided to remove features not present in at least 20% of the patient episodes. The percentage is set quite low to keep features that are rarely used but might be important for prediction. After filtering the features based on this constraint, only 47 of approximately 900 features were kept.

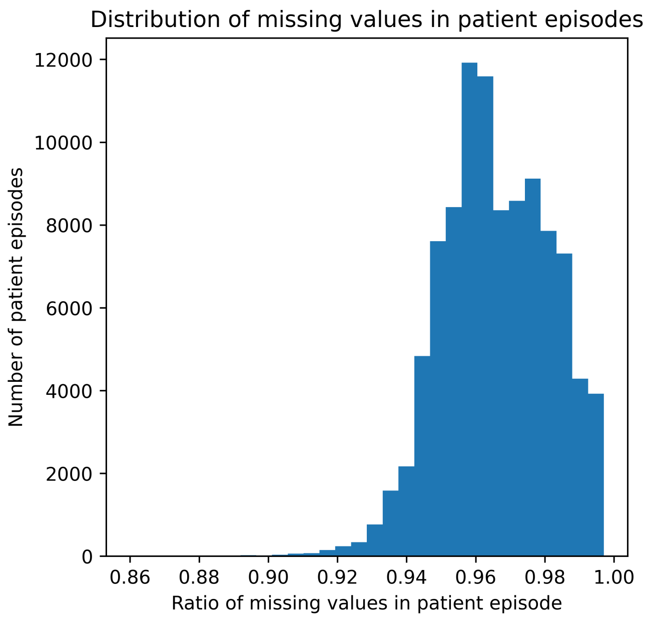


Figure 5.1: Distribution of the ratio of missing values in patient episodes

5.1.5 Data split

Finally, the dataset was divided into a training and a test set, used for final validation and evaluation. The training set consisted of data from the years 2021 and 2022 while data from 2023 was selected as test set. The former set contained 65,210 patient episodes compared to 34,070 episodes in the latter, with an almost identical proportion of positive outcomes.

5.2 Models

Three different machine learning models were implemented and evaluated. These were XGBoost and two different ANNs, MLP and TabNet, all three described in Sections 3.1, 3.2.4 and 3.2.5 respectively.

The XGBoost model was optimized and validated for three different imputation methods, in addition to using no imputation. In turn, each method was tested both with and without a missing indicator. The MLP and TabNet models were solely tested with median imputation and with missing indicator. The motivation for these decisions were that since XGBoost was the fastest and most efficient model to train by far, it was used to evaluate and compare the impact of the different imputation methods. Similarly, the same imputation method was used for both the MLP and TabNet, so that they could be compared with XGBoost and each other.

The first section below describes the setup stage, while the second one gives more detail to the evaluation.

5.2.1 Setup

Cross validation was used in the training phase for each model. The data from 2021 and 2022 was split into training and validation data for five folds. Additionally, the cross validation was implemented to generate stratified folds, meaning that the ratio between positive and negative samples were preserved.

The Optuna framework (18) was utilised for all models to find the optimal hyperparameters. In short, an upper and a lower bound is provided and then a so called study is created which iteratively finds the optimal hyperparameter values between the boundaries.

Binary cross entropy was the loss function of choice for the models. This function, described in Section 3.2.2, was chosen because of its suitability for binary classifiers.

5.2.2 Evaluation

After the models had been implemented, optimized and trained on all data from 2021 and 2022, they were evaluated on unseen data from 2023, and finally evaluated.

To compare the three models and the imputation methods, AUROC and AUPRC were used as metrics, defined in Sections 3.3.2 and 3.3.3. Additionally, several plots were generated to provide visualisation of the respective performances.

Chapter 6

Results

In this chapter, our results will be presented in subsections for each model. Each subsection is then divided into two parts, results related to model performance and results related to interpretability.

6.1 XGBoost

This section will present the results from the evaluation and the figure used for interpretability, for XGBoost. The figures included are for the imputation method that provided the best results on the test data.

6.1.1 Performance

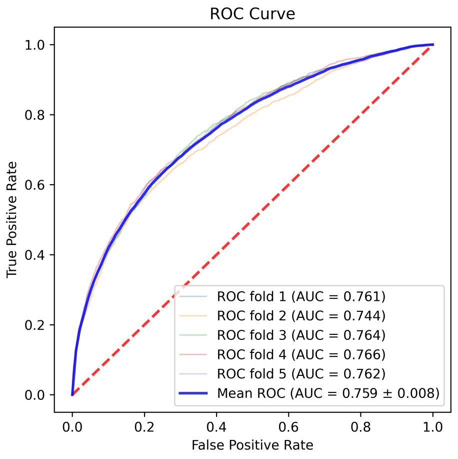
Here, the AUROC and AUPRC are presented in both tables and figures, for both the training and test sets. In Table 6.1 it can be seen that the results are similar for all imputations methods, with overlapping standard deviations.

Table 6.1: Cross-validation results for XGBoost on the training data. The table shows the AUROC and AUPRC values for varying imputation methods, including the standard deviation for the folds.

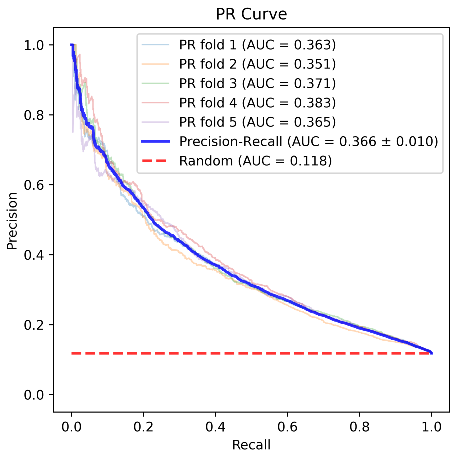
Imputation method	Missing indicator	AUROC	AUPRC
Mean	No	0.756 ± 0.008	0.360 ± 0.012
Mean	Yes	0.759 ± 0.007	0.365 ± 0.010
Median	No	0.759 ± 0.007	0.363 ± 0.011
Median	Yes	0.759 ± 0.008	0.366 ± 0.010
MICE with Bayesian Ridge	No	0.754 ± 0.008	0.354 ± 0.012
MICE with Bayesian Ridge	Yes	0.758 ± 0.007	0.361 ± 0.009
No imputation	Not applicable	0.758 ± 0.007	0.366 ± 0.011

Figure 6.1a and Figure 6.1b show that XGBoost perform considerably better than

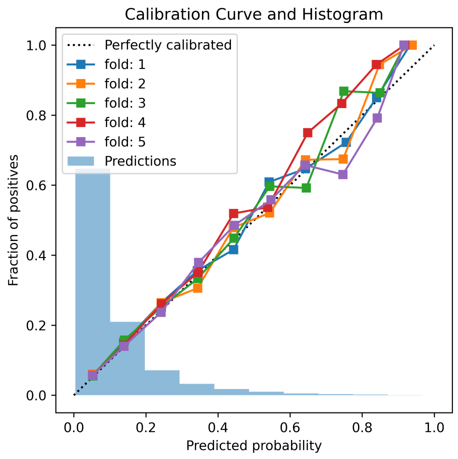
a random model. Similarly, the calibration plot in Figure 6.1c points to a well calibrated model.



(a) ROC curve. The red curve represents the performance of a model which randomly classifies samples.



(b) PR curve. The red curve represents the performance of a random model.



(c) Calibration and histogram plot. This plot shows how well-calibrated a model is. The histogram divides the predictions into bins, where the height of a bin represents the number of predictions for a certain probability. The axes represents the predicted and true probability of each bin.

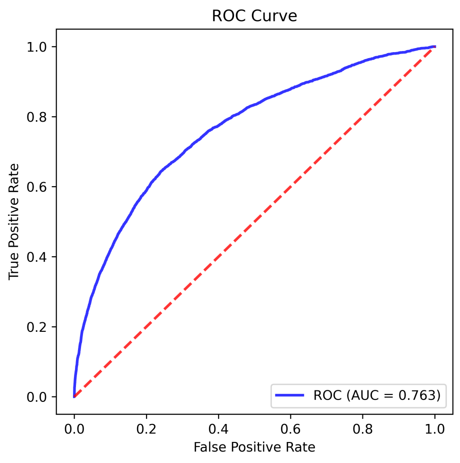
Figure 6.1: ROC, PR, and calibration plots for XGBoost with median imputation and missing indicator. The plots are for the training data.

The results in Table 6.2 are fairly similar for the different imputation methods. Although, imputing with median and utilising missing indicators obtain marginally higher scores, while MICE got slightly worse results.

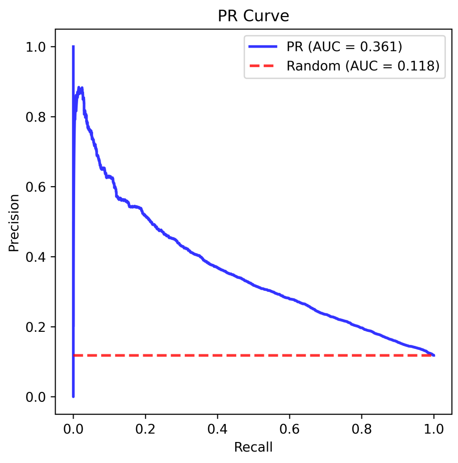
Table 6.2: Results for the XGBoost model on the test data. The table shows the AUROC and AUPRC values for varying imputation methods.

Imputation method	Missing indicator	AUROC	AUPRC
Mean	No	0.762	0.359
Mean	Yes	0.762	0.359
Median	No	0.762	0.360
Median	Yes	**0.763**	**0.361**
MICE with Bayesian Ridge	No	0.761	0.360
MICE with Bayesian Ridge	Yes	0.762	0.358
No imputation	Not applicable	0.762	0.360

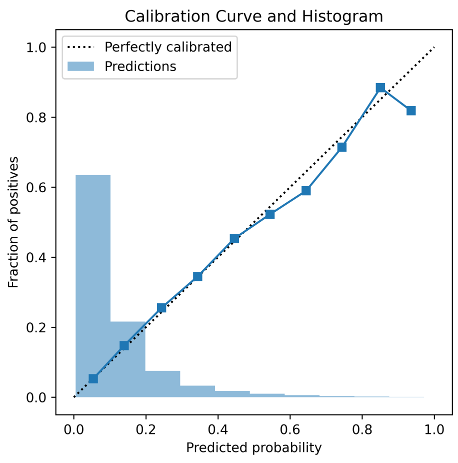
Similarly to the plots for the training data, the XGBoost model performed comparably well on the test set, as seen in Figure 6.2a and Figure 6.2b. Once again, the calibration plot in Figure 6.2c shows a well calibrated model, especially for predictions with lower probabilities.



(a) ROC curve. The red curve represents the performance of a model which randomly classifies samples.



(b) PR curve. The red curve represents the performance of a random model.



(c) Calibration and histogram plot. This plot shows how well-calibrated a model is. The histogram divides the predictions into bins, where the height of a bin represents the number of predictions for a certain probability. The axes represents the predicted and true probability of each bin.

Figure 6.2: ROC, PR, and calibration plots for XGBoost with median imputation and missing indicator. The plots are for the test data.

The confusion matrix in Figure 6.3 shows that while many labels are incorrectly classified as positive, the amount of true negatives are relatively high while the percentage of false negatives are low. A threshold of 5% was chosen in accordance to previous work (4). The significance of this value is further discussed in Section 7.

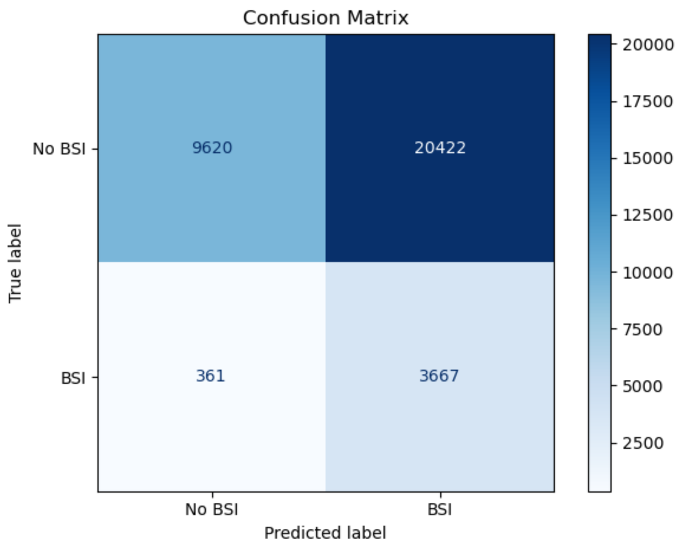


Figure 6.3: Confusion matrix corresponding to a threshold of 5%. This plot shows the predicted and true labels for all data points in the test set.

6.1.2 Interpretability

The motivation for this section is to provide interpretability. This will be carried out through SHAP values, showing the importance and influence of the features, presented in Figure 6.4 below. Notably, P-CRP, P-Kreatinin and age ranks among the top three features with the highest influence over the predicted values, where high values for these features are shown to contribute to positive outcomes, with exception of a few cases of low age.

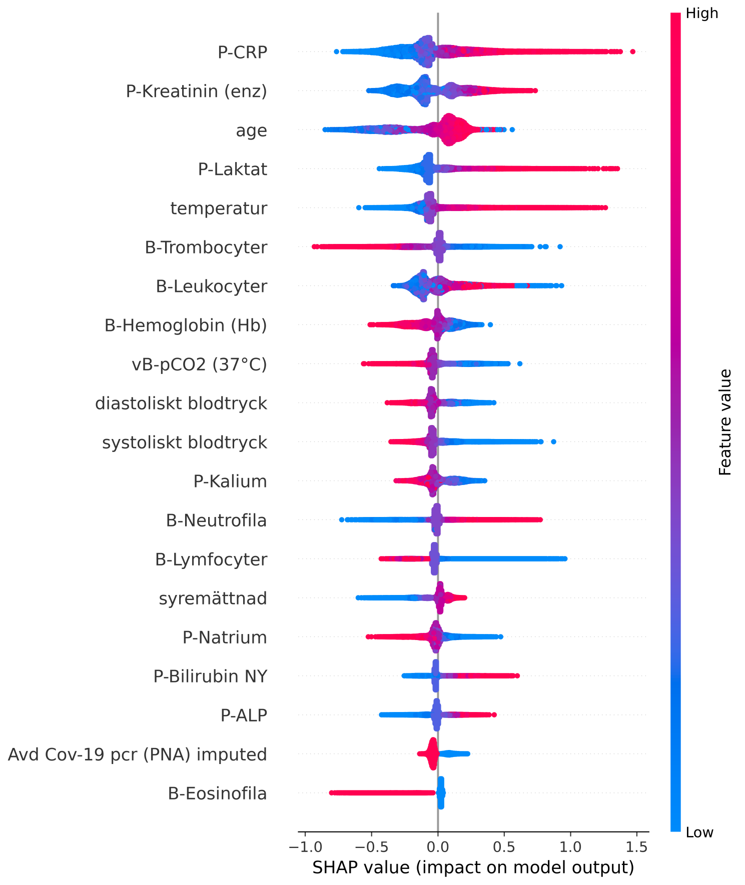


Figure 6.4: The figure shows the features corresponding the the 20 largest SHAP values. It can be seen that the three most important features for prediction on the test set is P-CRP ,P-Kreatinin and age.

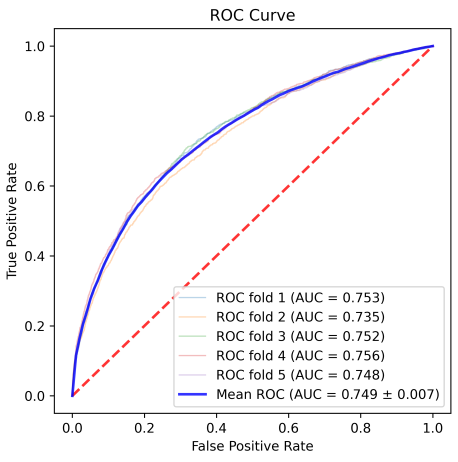
6.2 Multilayer Perceptron

The results from the evaluation of the MLP will be presented in this section. Due to the inherent lack of interpretability of a MLP, there will be no such section for this model, contrary to XGBoost and TabNet.

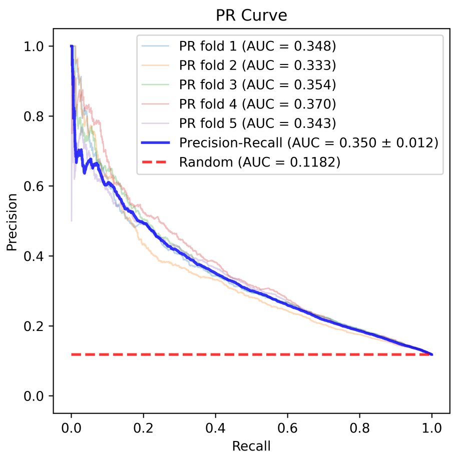
6.2.1 Performance

This section will present the AUROC and AUPRC scores in plots, for the training and test datasets, with the results for the test set summarized in a table in Section 6.4 further down. The imputation method being used will be median and missing indicator.

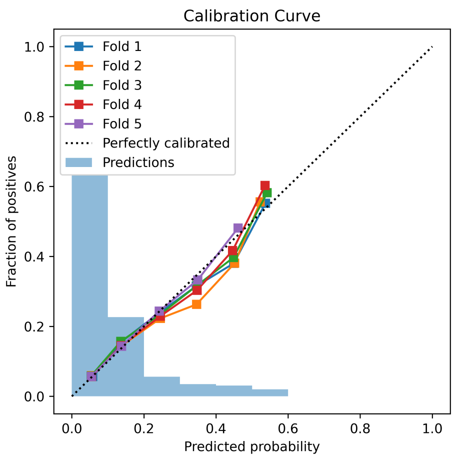
MLP obtained mean AUROC and AUPRC scores of 0.749 and 0.350 on the training data, as seen in Figure 6.5a and Figure 6.5b. Regarding the calibration curve and histogram in Figure 6.5c, MLP had no predicted labels with probabilities above approximately 0.6, which is why the curve stops at this value. Nonetheless, the model proved itself slightly worse calibrated than XGBoost.



(a) ROC curve. The red curve represents the performance of a model which randomly classifies samples.



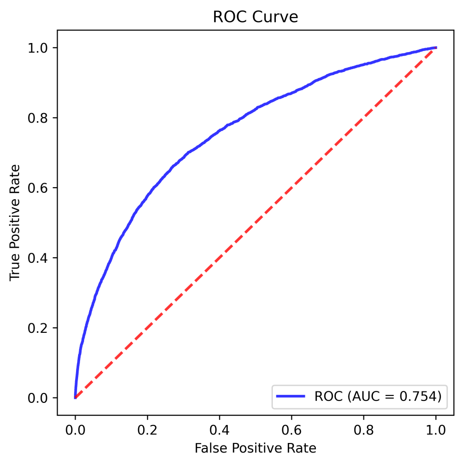
(b) PR curve. The red curve represents the performance of a random model.



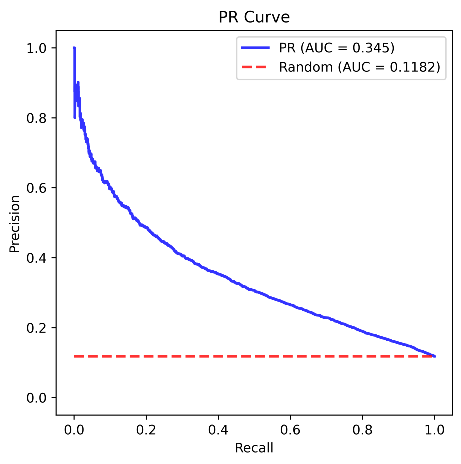
(c) Calibration and histogram plot. This plot shows how well-calibrated a model is. The histogram divides the predictions into bins, where the height of a bin represents the number of predictions for a certain probability. The axes represents the predicted and true probability of each bin.

Figure 6.5: ROC, PR, and calibration plots for MLP with median imputation and missing indicator. The plots are for the training data.

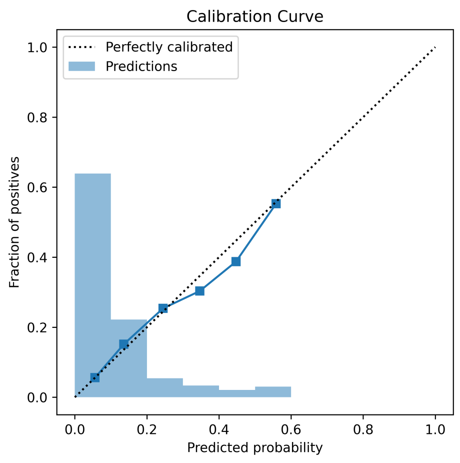
Evaluated on the test set, MLP achieved an AUROC of 0.754 and an AUPRC of 0.345, as displayed in Figure 6.6a and Figure 6.6b respectively. Once again, as presented in Figure 6.6c, the calibration plot has no values for probabilities higher than 0.6. Below this value, the model is decently calibrated.



(a) ROC curve. The red curve represents the performance of a model which randomly classifies samples.



(b) PR curve. The red curve represents the performance of a random model.



(c) Calibration and histogram plot. This plot shows how well-calibrated a model is. The histogram divides the predictions into bins, where the height of a bin represents the number of predictions for a certain probability. The axes represents the predicted and true probability of each bin.

Figure 6.6: ROC, PR, and calibration plots for MLP with median imputation and missing indicator. The plots are for the test data.

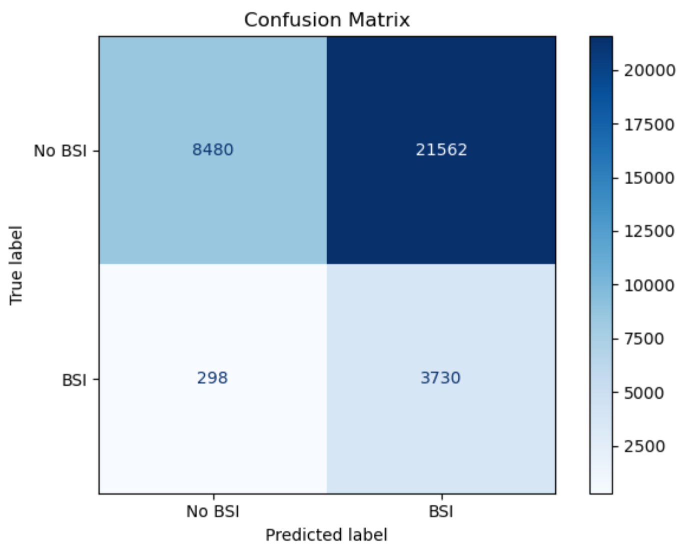


Figure 6.7: Confusion matrix for MLP with a threshold of 5 %. This plot shows the predicted and true labels for all data points in the test set.

As a consequence to performing similarly to XGBoost, the confusion matrix in Figure 6.7 looks similar to the one from XGBoost, with a low amount of false negatives but many false positives.

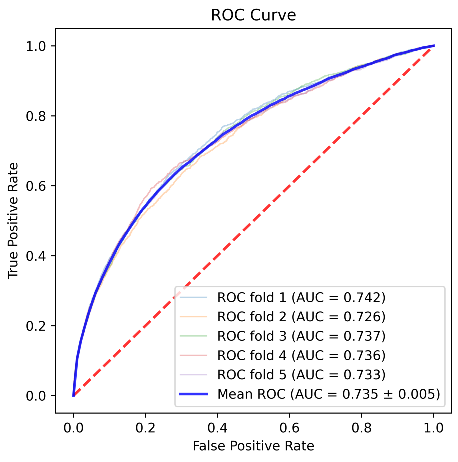
6.3 TabNet

In this section, the evaluation metrics for TabNet will be presented in plots. As with MLP, the scores will be summarized in a table in Section 6.4. Additionally, a figure showing feature importances will be included in the section about interpretability.

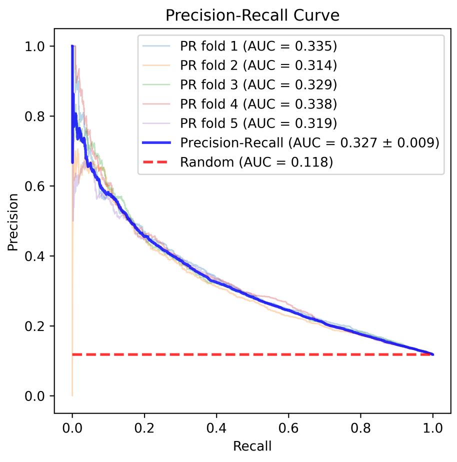
6.3.1 Performance

Here, the AUROC and AUPRC scores will be given for the training and test datasets. Median imputation and missing indicator will be utilised.

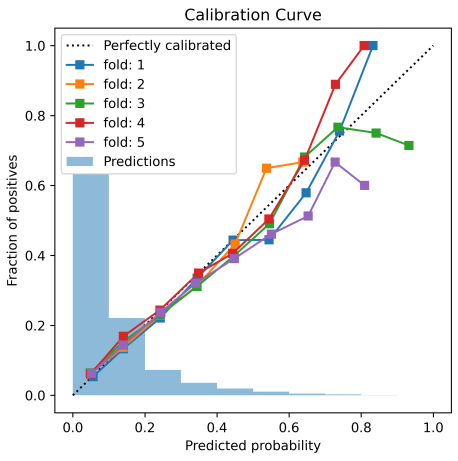
The cross-validation results for TabNet on the training data can be seen in Figure 6.8a and Figure 6.8b. Notably, TabNet performed considerably better than a random model, obtaining an AUROC of 0.735 and an AUPRC of 0.327. Figure 6.8c shows that the model is well calibrated for lower probability values but less so for higher values.



(a) ROC curve. The red curve represents the performance of a model which randomly classifies samples.



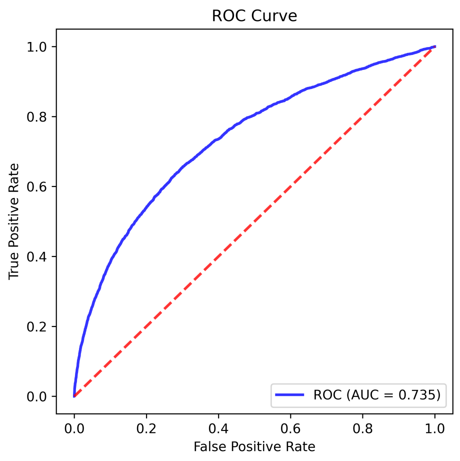
(b) PR curve. The red curve represents the performance of a random model.



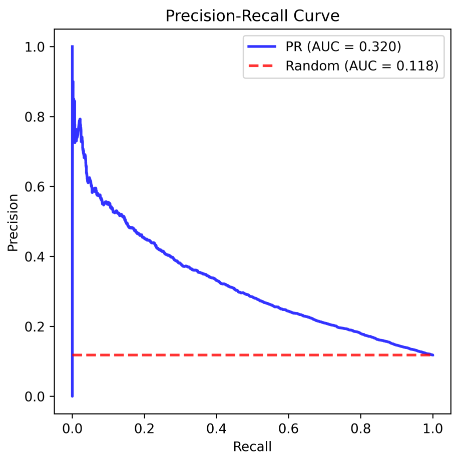
(c) Calibration and histogram plot. This plot shows how well-calibrated a model is. The histogram divides the predictions into bins, where the height of a bin represents the number of predictions for a certain probability. The axes represents the predicted and true probability of each bin.

Figure 6.8: ROC, PR, and calibration plots for TabNet with median imputation and missing indicator. The plots are for the training data.

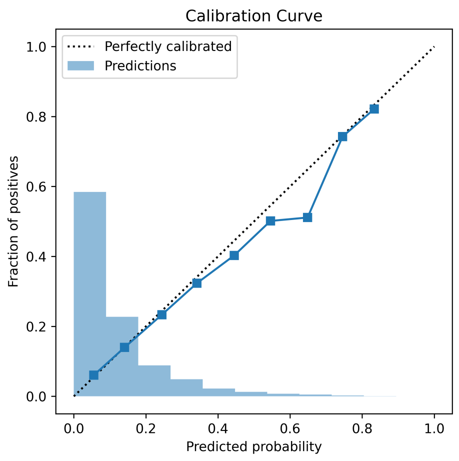
TabNet achieved AUROC and AUPRC values of 0.735 and 0.320, given by the Figure 6.9a and Figure 6.9b. Once again, the model gets worse calibrated for higher predicted probabilities, according to the curve in Figure 6.9c.



(a) ROC curve. The red curve represents the performance of a model which randomly classifies samples.



(b) PR curve. The red curve represents the performance of a random model.



(c) Calibration and histogram plot. This plot shows how well-calibrated a model is. The histogram divides the predictions into bins, where the height of a bin represents the number of predictions for a certain probability. The axes represents the predicted and true probability of each bin.

Figure 6.9: ROC, PR, and Calibration plots for TabNet with median imputation and missing indicator. The plots are for the test data.

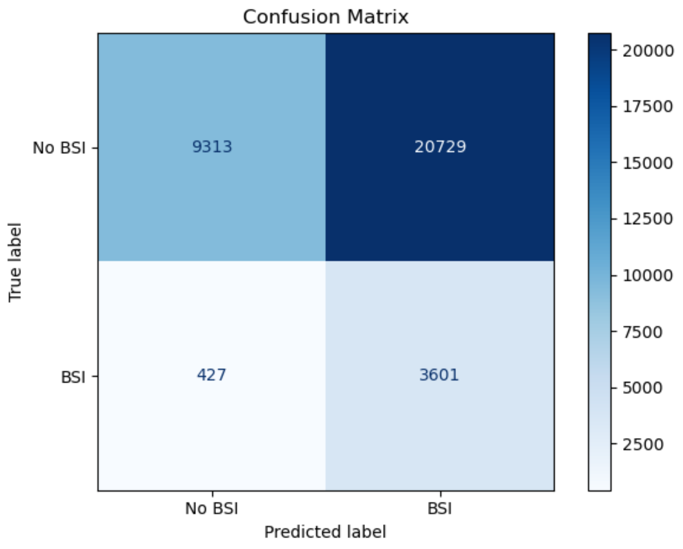


Figure 6.10: Confusion matrix for TabNet with a threshold of 5%. This plot shows the predicted and true labels for all data points in the test set.

With a threshold of 5%, the confusion matrix in Figure 6.10 is comparable to those for XGBoost and MLP. Notably, 9313 labels are correctly classified as negative outcomes, while 427 are incorrectly classified as the same.

6.3.2 Interpretability

Figure 6.11 shows the global feature importances based on predictions on the test set, which was used for comparability with XGBoost and its SHAP values. According to the figure, age and pregnancy are the two most important features.

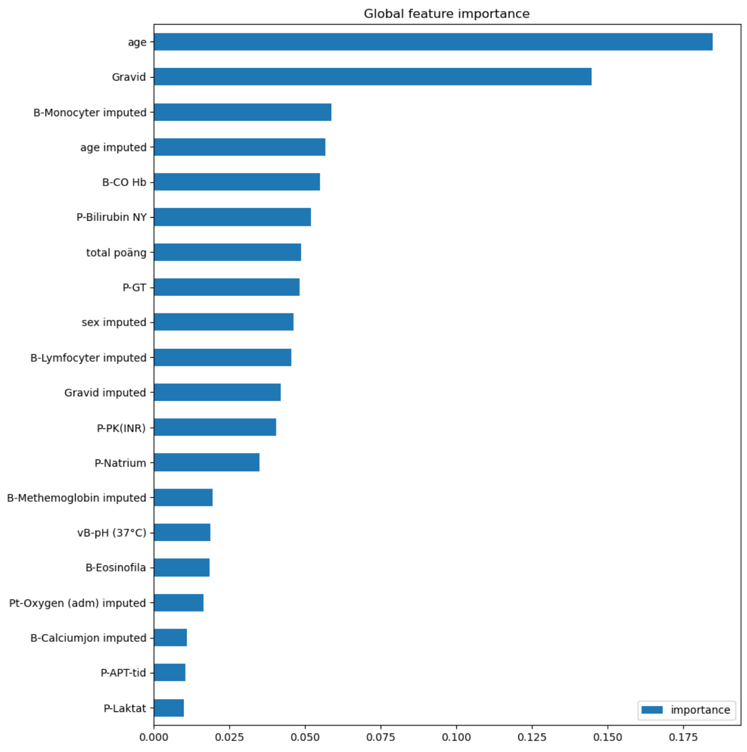


Figure 6.11: The figure shows the global feature importances for predictions on the test set.

6.4 Summary

In this section the evaluation of the three different models are summarized in a single table, with results of median imputation and missing indicators on the test data. The table, seen in Table 6.3, shows that XGBoost performed best on the test set with an AUROC of 0.763 and AUPRC of 0.361. MLP performed slightly worse, while TabNet achieved the lowest scores of the three.

Table 6.3: Summary of results for XGBoost, MLP and TabNet, using median imputation and missing indicator, with the highest scores in bold text.

Model	AUROC	AUPRC
XGBoost	**0.763**	**0.361**
MLP	0.754	0.345
TabNet	0.735	0.320

Chapter 7

Discussion

In this chapter, we will discuss the results for the different models and imputation methods. It will be divided into two sections. In the first one, the results of the different imputation methods will be discussed, while the three different models (XGBoost, MLP and TabNet), described in Section 3, will be discussed in the proceeding section.

7.1 Imputation methods

As seen in Table 6.1, the AUROC and AUPRC scores are similar for all imputation methods, especially when taking standard deviations into account. This indicates that there are no significant differences between the results obtained using different imputation methods. Due to this, a superior imputation method can not be determined.

However, in Table 6.2, it can be noted that there is a slight drop, or at least no improvement, in performance when imputing using MICE. Since MICE assumes data is missing at random, and tries to impute based on data not missing, a poorer performance in comparison to the other imputation methods may indicate that the data in the dataset is not missing at random, explained in Section 4.1.

Additionally, as seen in Table 6.1 and Table 6.2, missing indicators seem to marginally improve the results during validation, but have no significant impact on the test set.

7.2 Models

This section will compare the results of the XGBoost, MLP, and TabNet models based on the AUROC and AUPRC scores. For XGBoost and TabNet, which have increased support for interpretability, the feature importances will be discussed. Furthermore, since XGBoost obtained the best results, it will be used as the point of reference when discussing applicability in practice.

When training the models, we observed that XGBoost was noticeably faster to train. While it took around 15 seconds to train it on all five cross-validation folds, it took MLP and TabNet approximately 45 and 140 minutes respectively. We believe the difference in efficiency is due to the complexity of the models and their various training algorithms. The MLP used in this thesis consisted of 118,401 trainable parameters while TabNet had 509,732 trainable parameters. XGBoost and its structure of tree ensembles has no direct counterpart to this measurement. Nonetheless, its complexity is dependant on the number of trees constructed during training and their maximum depths and number of leaves.

7.2.1 AUROC and AUPRC

As mentioned in the summary of the results in Section 6.4 and as shown in the corresponding Table 6.3, XGBoost obtained the best results. This outcome was to be expected, due to earlier proven results of XGBoost in varying applications, and its suitability for tabular data (19)(20).

An additional contributing factor to the superiority of XGBoost could be that its computational efficiency meant that many more iterations of hyperparameter optimization could be run, potentially leading to a more well calibrated model, as noted in the calibration plots in Figure 6.2c, 6.6c and 6.9c. On another note, the calibration curve and histogram for MLP showed that there were no predictions with values above approximately 0.6, while the same was true for TabNet for a respective value of about 0.8.

Comparing MLP and TabNet, there is a trade-off between performance and interpretability. While MLP obtained superior scores, TabNet provides the knowledge of why a certain prediction was made.

7.2.2 Applicability in practice

As mentioned earlier in this thesis, the main motivation for using a machine learning model to predict BSI outcomes is to decrease the amount of redundant blood cultures. In the confusion matrix for XGBoost in Figure 6.3, we want to increase the amount of episodes in the upper left section, that is patients where the outcome is correctly predicted as negative, while minimizing the values in the lower left section, consisting of patients incorrectly classified as having no BSI.

The aforementioned confusion matrix was constructed with a threshold value of 5%. Here, 9620 patient episodes were correctly predicted to have no BSI, resulting in a 29% decrease in blood culture testing. Meanwhile, with this threshold 361 episodes, equivalent to 1%, would incorrectly be classified to have no BSI, potentially having fatal consequences for the affected patients. Therefore, the threshold value of choice would mainly be up to the physicians, where an increased threshold, up to a certain limit, would lead to a more predicatively correct model at the expense of increased false negatives.

7.2.3 Interpretability

The SHAP values in Figure 6.4 presents the features with the highest influences on the output from XGBoost. According to the plot, P-CRP was established to have the highest impact on the output, where a low content of the protein was associated with a lower risk of having a BSI, while a high content increased the risk. Other high influence indicators of being diagnosed with a BSI was high age and high body temperature, while low contents of B-Trom and hemoglobin were indicative of BSIs.

Figure 6.11 shows the global feature importances for predictions using the TabNet model. It can be seen that XGBoost and TabNet seem to value features differently, with the exception of age, which both deemed important for predicting BSI outcomes. One notable difference between XGBoost and TabNet is that TabNet seems to value missing indicators more than XGBoost. Some of the feature importances can be questioned, such as the missing indicators for sex and age. These variables are always zero as the sex and age of a patient are always known in our dataset. A more well calibrated model might have weighted the feature importances differently, and perhaps more logically.

Chapter 8

Conclusion

In this thesis, a comparison between different imputation methods and between XGBoost, MLP and TabNet has been carried out. The imputation comparisons were conducted on the XGBoost, since it was the fastest model to train by far, and since it appeared to perform the best early in the process. The models were mainly evaluated through the evaluations metrics AUROC and AUPRC.

A considerable part of the work consisted of preprocessing the data, where missing data was imputed with multiple methods. In conclusion, the results indicated that the choice of method had no significant impact on the outcome. Notably, MICE was the only method that showed a somewhat differential, slightly worse, result. This has shown that the choice of imputation method is not arbitrary. The most advanced method is not the solution for every application.

It was found that XGBoost was the superior model when it comes to performance, scoring an AUROC of 0.763 and an AUPRC of 0.361 on the test set. This is no surprise as gradient boosting trees have been the gold standard for tabular data for a long time, proving to give good results on a variety of datasets. It was found that using XGBoost to predict BSI outcomes could reduce the number of blood culture tests by approximately 29%, with the drawback of predicting a false negative in 1% of the cases.

Application of this model in practice would have to be treated with care due to the potential fatal consequences of incorrect predictions, where the aforementioned percentage of false negatives might be regarded as unacceptable.

Using the interpretability properties of XGBoost, it was found that three most important laboratory results for predicting was P-CRP,P-Kreatinin and P-Laktat. The most important vital measurements were temperature, systolic and diastolic blood pressure. In addition, the age of a patient were among the most impactful features. The reason the model considered age an important feature may be related to the distribution of the variable, where some age groups were more represented than others, namely infants and elderly.

Further investigation into the preprocessing of the data could be of interest to potentially obtain improved results. One idea could be to further constrain which features and patient episodes that are included in the study. Finally, it could be interesting to keep the sequential property of the raw data with the intent of broadening the selection of applicable models to explore.

References

[1] T. Xu, S. Wu, J. Li, L. Wang, and H. Huang, "Development of a risk prediction model for bloodstream infection in patients with fever of unknown origin," *Journal of Translational Medicine*, vol. 20, no. 1, p. 575, 2022.

[2] L. E. Huerta and T. W. Rice, "Pathologic difference between sepsis and bloodstream infections," *The Journal of Applied Laboratory Medicine*, vol. 3, no. 4, pp. 654–663, 2019.

[3] F. B. Hertz, M. G. Ahlström, M. H. Bestle, L. Hein, T. Mohr, J. D. Lundgren, T. Galle, M. H. Andersen, D. Murray, A. Lindhardt, T. S. Itenov, and J.-U. S. Jensen, "Early biomarker-guided prediction of bloodstream infection in critically ill patients: C-reactive protein, procalcitonin, and leukocytes," *Open Forum Infectious Diseases*, vol. 9, no. 10, p. ofac467, 2022.

[4] M. Schinkel, A. W. Boerman, F. C. Bennis, T. C. Minderhoud, M. Lie, H. Peters-Sengers, F. Holleman, R. P. Schade, R. de Jonge, W. J. Wiersinga, and P. W. B. Nanayakkara, "Diagnostic stewardship for blood cultures in the emergency department: A multicenter validation and prospective evaluation of a machine learning prediction tool," *EBioMedicine*, vol. 82, p. 104176, 2022.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, ACM, Aug. 2016.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[7] A. F. T. Martins and R. F. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," 2016.

[8] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, IEEE, Oct. 2020.

[9] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," 2020.

[10] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets,” *PloS one*, vol. 10, p. e0118432, Mar 2015.

[11] P. Adeodato and S. Melo, “A geometric proof of the equivalence between auc_roc and gini index area metrics for binary classifier performance assessment.,” *2022 International Joint Conference on Neural Networks (IJCNN), Neural Networks (IJCNN), 2022 International Joint Conference on*, pp. 1 – 6, 2022.

[12] S. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” 2017.

[13] J. Choi, O. M. Dekkers, and S. le Cessie, “A comparison of different methods to handle missing data in the context of propensity score analysis,” *European Journal of Epidemiology*, vol. 34, no. 1, pp. 23–36, 2019.

[14] H. Wafaa Mustafa and A. Nzar A., “Missing value imputation techniques: A survey.,” *UHD Journal of Science and Technology*, vol. 7, no. 1, pp. 72 – 81, 2023.

[15] C. Mack, Z. Su, and D. Westreich, “Managing missing patient data in patient registries,” White Paper, addendum to Registries for Evaluating Patient Outcomes: A User’s Guide, Third Edition AHRQ Publication No. 17(18)-EHC015-EF, L&M Policy Research, LLC, Rockville, MD, February 2018.

[16] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, “Multiple imputation by chained equations: what is it and how does it work?,” *International Journal of Methods in Psychiatric Research*, vol. 20, no. 1, pp. 40–49, 2011.

[17] “Dreamquark tabnet.” https://github.com/dreamquark-ai/tabnet. Accessed: 2024-05-07.

[18] Optuna, “Optuna: A hyperparameter optimization framework.” https://optuna.org, 2024. Accessed: 2024-05-31.

[19] R. Shwartz-Ziv and A. Armon, “Tabular data: Deep learning is not all you need,” 2021.

[20] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on tabular data?,” 2022.

Master’s Theses in Mathematical Sciences 2024:E45
ISSN 1404-6342

LUTFMA-3545-2024

Computer Vision and Machine Learning
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/