Neonatal Sepsis Detection Using Decision Tree Ensemble Methods: Random Forest and XGBoost

Marwan Al-Bardaji and Nahir Danho

***Abstract*—Neonatal sepsis is a potentially fatal medical condition due to an infection and is attributed to about 200 000 annual deaths globally. With healthcare systems that are facing constant challenges, there exists a potential for introducing machine learning models as a diagnostic tool that can be automatized within existing workflows and would not entail more work for healthcare personnel. The Herlenius Research Team at Karolinska Institutet has collected neonatal sepsis data that has been used for the development of many machine learning models across several papers. However, none have tried to study decision tree ensemble methods. In this paper, random forest and XGBoost models will be developed and evaluated in order to assess their feasibility for clinical practice. The data contained 24 features of vital parameters that are easily collected through a patient monitoring system. The validation and evaluation procedure needed special consideration due to the data being grouped based on patients and being imbalanced. The proposed method has the potential to be generalized to other similar applications. Finally, using the measure receiver-operating-characteristic area-under-curve (ROC AUC), both models achieved around ROC AUC$= 0.84$. Such results suggest that the random forest and XGBoost models are potentially feasible for clinical practice. Another gained insight was that both models seemed to perform better with simpler models, suggesting that future work could create a more explainable model.**

***Sammanfattning*—Neonatal sepsis är ett potentiellt dödligt medicinskt tillstånd till följd av en infektion och uppges globalt orsaka 200 000 dödsfall årligen. Med sjukvårdssystem som konstant utsätts för utmaningar existerar det en potential för maskininlärningsmodeller som diagnostiska verktyg automatiserade inom existerande arbetsflöden utan att innebära mer arbete för sjukvårdsanställda. Herelenius forskarteam på Karolinska Institet har samlat ihop neonatal sepsis data som har använts för att utveckla många maskininlärningsmodeller över flera studier. Emellertid har ingen prövat att undersöka beslutsträds ensemble metoder. Syftet med denna studie är att utveckla och utvärdera random forest och XGBoost modeller för att bedöma deras möjligheter i klinisk praxis. Datan innehör 24 attribut av vitalparameterar som enkelt samlas in genom patientövervakningssystem. Förfarandet för validering och utvärdering krävde särskild hänsyn med tanke på att datan var grupperad på patientnivå och var obalanserad. Den föreslagna metoden har potential att generaliseras till andra liknande tillämpningar. Slutligen, genom att använda receiver-operating-characteristic area-under-curve (ROC AUC) måttet kunde vi uppvisa att båda modellerna presterade med ett resultat på ROC AUC$= 0.84$. Sådana resultat föreslår att både random forest och XGBoost modellerna kan potentiellt användas i klinisk praxis. En annan insikt var att båda modellerna verkade prestera bättre med enklare modeller vilket föreslår att framtida arbete skulle kunna vara att skapa en mer förklarlig maskininlärningsmodell.**

***Index Terms*—Machine Learning, Sepsis, Neonatal Sepsis, Random Forest, XGBoost, Imbalanced Data, Binary Classification, Cross-Validation, Hyperparameter Tuning.**

Supervisors: Antoine Honoré

TRITA number: TRITA-EECS-EX-2022:175

I. Introduction

The goal of the healthcare system is the maintenance and improvement of the health of a population. An essential step in that work is the diagnosis and detection of diseases, where physicians typically work using a combination of their own experience and medical guidelines using years of research [1]. This project focuses on neonatal sepsis, a possibly fatal medical condition due to an infection [2]. Globally sepsis is attributed to about 200 000 annual deaths [3]. There exist several scoring systems for the detection and prognosis estimation of sepsis built on international consensus; however, none is perfect [2], [4]. Meanwhile, healthcare systems are constantly facing issues such as rising costs [5], staffing shortages [6], and aging populations with ever-increasing healthcare needs. This could lead to situations where guidelines cannot be followed perfectly, thus endangering patient safety [1]. Any method that could increase diagnostic performance without inferring increased effort by healthcare personnel would be desirable.
Machine learning models have been on the rise since the 20th century, and adoption is continuously increasing with more powerful computers and the improved ability to collect large datasets. There exist many machine learning algorithms whose goal is to classify different data and could therefore have the potential as a diagnostic tool in healthcare that can be automatized within existing workflows [7].

A. Problem Formulation

This project will use neonatal sepsis data provided by the Herlenius Research team at Karolinska Institutet. Earlier work for the detection of sepsis using machine learning models has been conducted using the same data. Examples of previous models that have been used are Markov models, logistic regression, naïve Bayes, multi-later perceptrons, Gaussian mixture models, and normalizing flows [8], [9]. Many of the models achieved promising results. Nevertheless, no work has studied the performance of models that use decision tree ensemble methods.

B. Project Goal and Scope

The goal of the project is to study the feasibility of using decision tree ensemble methods to detect neonatal sepsis. Specifically, the performance of random forest and XGBoost

models will be evaluated and compared. Moreover, insightful learnings from the development of the models will be collected and discussed. The project is limited to a proof of concept and does not create a model ready for real-life implementation.

II. Background

A. Machine Learning

Machine Learning (ML) is a subset of artificial intelligence that deals with learning in the sense that the algorithm's performance on future tasks can be improved by making observations of the world [7]. Machine learning algorithms are generally divided into three categories (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning. Supervised learning is predictive, where the algorithm maps an input to an output. Hidden training data is provided for which the mapping is known beforehand, also known as labeled data. A trained model can later be used to predict the output of future inputs with unknown outputs. Unsupervised learning analyzes unstructured data and tries to find a pattern on its own; here, there exists only input and no output. On the other hand, reinforcement learning is an algorithm that reacts to the environment rewarding desired behaviors and punishing undesired ones [7]. There exists a myriad of algorithms under each category; moreover, there also exist algorithms that do not fall under one category, some of which are semi- supervised learning. This project will deal with a classification problem within supervised learning.

B. Medical Background

Sepsis is a possibly fatal medical condition which, in layman's terms, often somewhat incorrectly is called blood poisoning, thus not revealing the entire truth. According to international consensus [**?**], sepsis should be defined as "life-threatening organ dysfunction caused by a dysregulated host response to infection. Left untreated, sepsis can turn into sepsis shock. sepsis shock is defined as [**?**] "a subset of sepsis in which underlying circulatory and cellular metabolism abnormalities are profound enough to increase mortality substantially." Many survivors develop permanent neurologic impairment [10]. Sepsis does not have obvious symptoms, especially in the early course of the condition. To identify sepsis in clinical practice, international consensus recommends using the Sequential Organ Failure Assessment (SOFA) score or the quick SOFA (qSOFA) score [4], [11], [12]. These scoring systems use a combination of vital parameters, such as partial pressure of oxygen in the blood and blood pressure; blood tests, such as platelet count and bilirubin; and neurological status [11]. Moreover, after a sepsis suspicion has arisen, there exist international guidelines created by the "Surviving Sepsis Campaign" for the most appropriate tests and treatments to continue with [12]. Suspected sepsis can be confirmed through microbiologic blood cultures; the culture needs to be obtained before any antibiotic treatments. In principle, there exist two types of treatment (i) antimicrobial treatment, which usually starts with empiric broad-spectrum antibiotics, and (ii) organ-supportive such as fluid therapy and vasoactive medication [12]. An inherent limitation of using the SOFA scoring system is that doctors and nurses conduct additional tasks such as taking blood tests that do not necessarily need to happen for all patients [11].

This project will study neonatal sepsis, meaning sepsis in the first weeks of an infant's life. In neonates, sepsis is difficult to diagnose clinically since they may be asymptomatic until organ dysfunction is prominent [13]. There exists an adapted version of the SOFA score called the nSOFA score; however, this score still requires invasive testing such as blood tests which both are costly and take time compared to collecting vital parameters. Moreover, the nSOFA scoring is not widely adopted [14]. A study by Fairchild suggests that there exists potential with using heart rate variability, heart rate characteristics, and other vital signs in the detection of Neonatal Sepsis. These measurements are non-invasive and can easily be collected through a monitoring machine [10]. Current challenges in the treatment of neonatal sepsis include late detection, overuse of antibiotics, and difficulties with invasive testing [10].

C. Classification Problems

A classification problem within supervised machine learning refers to a predictive modeling problem where the aim is to predict an output given an input. The following definition of a classification problem will assume a one-dimensional output, also called the label. Each data input will is provided as a feature vector $\vec{x}_i$ with its corresponding output y_i. Since the input is in vector form, it may include multiple data points corresponding to several features.

To train the machine learning model, a dataset $\mathcal{D}$ with n examples and m features is provided.

$$\mathcal{D} = \{(\vec{x}_i, y_i)\} \quad (|\mathcal{D}| = n, \vec{x}_i \in \mathbb{R}, y_i \in \mathbb{R}) \tag{1}$$

(Binary classification is the case when $y_i \in \mathbb{B} = \{0, 1\}$).

A prediction function ϕ gives the prediction.

$$\hat{y}_i = \phi(\vec{x}_i) \tag{2}$$

The behavior of ϕ depends on internal model parameters that are unique for each machine learning implementation. The procedure of finding the internal parameters is what training a model means [7]. The internal parameters of a model are fit by minimizing the objective function $\mathcal{L}$ which typically is of the following form.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \Omega(\theta) \tag{3}$$

Here l is a differentiable convex loss function that measures the difference between y_i and $\hat{y}_i$. The second term Ω is a function of the internal parameters which penalize complex models. There exist several loss and penalty functions depending on the implementation that shares the concept of distance and model complexity, respectively. [7]

If an unseen data point $\vec{x}^{\text{unseen}}$ is provided without a label then there exists no way to know whether the prediction $\hat{y}^{\text{unseen}} = \phi(\vec{x}^{\text{unseen}})$ is correct. However, if a theoretical label y^{real} is allowed to exist then the goal of any classification algorithm is that $\hat{y} = y^{\text{real}}$. To imitate this procedure, the

original dataset $\mathcal{D}$ is commonly divided into a test set and a training set.
The training set is used for fitting the internal parameters of the selected model, while the test set is used to evaluate the performance of the model. Sometimes the original dataset is divided into yet another set called the validation set to optimize external parameters, also called hyperparameters, on unseen data before evaluating the results on the test set. External parameters affect a model's behavior and do not change depending on the training data [7].

D. Decision Trees

A decision tree is a representation of a function that maps a feature vector to a single output value which is called the decision [7]. In essence, it is similar to a flowchart of questions that are commonly used within the healthcare system [15]. The decision tree starts with a root node. Each node may, in turn, split into several other nodes. A node that does not split into other nodes is called a leaf and contains the final decision. At each node, a question regarding the question is asked. In implementations using numerical data the question are comparisons of the types $<$, $<=$, $=$, $>=$, $>$.
The prediction functions ϕ works by using an algorithm that finds the feature and the question to ask that provide the highest "importance." Importance is measured using information gain, which is defined in terms of entropy. These quantities are fundamental in information theory. To train the decision tree, all the feature vectors in the training data set to go through the entire decision tree, and the number of samples and their class are calculated at the leaf nodes [7], which means that each node sorts the incoming data into smaller sets.
Entropy is a measure of uncertainty of a random variable; the more information, the less entropy. In general, the entropy of a random variable V with values v_k having the probability $P(v_k)$ is defined as [7].

$$\text{Entropy} = H(V) = -\sum_k P(v_k)\log_2 P(v_k) \tag{4}$$

Entropy is measured in bits and corresponds to the expected number of $50/50$ guesses it would require to narrow down to a specific value v_k. For example, a fair coin flip has an entropy of $H(\text{Fair coinflip}) = -2 \cdot (0.5 \log_2 0.5) = 1$ and a fair six sided die has an entropy of $H(\text{Fair six-sided die} = -6 \cdot (1/6 \log 1/6) \approx 2.6$ [7].
The information gain at a node is calculated as the expected reduction in entropy by sorting the incoming data. The information gain on attribute A and data S is defined as [7].

$$\text{Information gain} = H(S) - \sum_{\text{iin } v} \frac{|S_i|}{|S|} H(S_v) \tag{5}$$

where S is the incoming data to the node, v is a set of mutually exclusive questions, and S_i is the sorted version of S after asking a question i. S is considered to be a random variable with values and probabilities according to the distribution in the data [7].
The maximization of the information is thus the objective function of the decision tree. However, in order to limit the complexity of the model and reduce bias, it is possible to penalize complex models with a regularization parameter Ω [7].

E. Random Forests

Random forests are an ensemble of decision trees that uses bootstrap aggregating to reduce variance in a noisy dataset by training multiple different trees. A majority vote of all trees then decides the output [16].
A random forest classifier R is defined as

$$R = \text{Majority vote of } \{h(\vec{x}, \Theta_k), k = 1, ...\} \tag{6}$$

Where h is a decision tree, and Θ_k is an independent random vector that the decision tree k uses in the construction of the tree [16].
The primary source of randomness in a random forest is feature subsampling, where a random number of features are selected for each tree. This reduces the bias by increasing the probability that the trees are uncorrelated. The objective function and penalization of complex trees are analogous to the decision tree [16].

F. XGBoost

XGBoost, an implementation of gradient boosted decision trees, has shown state-of-the-art results in many machine learning challenges. XGBoost is an ensemble method of regression trees that uses boosting, which aims to improve performance by creating a strong classifier from many weak classifiers. Regression trees use the same concepts as a decision tree but have a continuous target variable. Moreover, XGBoost is designed with system performance in mind and is easily scalable [17].

1. Objective Function
   For a given dataset with n examples and m features

   $$\mathcal{D} = \{(\vec{x}_i, y_i)\} \quad (|\mathcal{D}| = n, \vec{x}_i \in \mathbb{R}, y_i \in \mathbb{R}) \tag{7}$$

   A tree ensemble method uses K additive functions to predict the output according to

   $$\hat{y} = \phi(\vec{x}_i) = \sum_{k=1}^{K} f_k(\vec{x}_i), \quad f_k \in \mathcal{F} \tag{8}$$

   where $\mathcal{F} = \{f(\vec{x}) = w_{q(\vec{x})}\} \quad (q : \mathbb{R}^m \to T.w \in \mathbb{R}^T)$ is the space of regression trees. T is the number of leaves in each tree and q represents the structure of each tree that maps an input to the leaves. Each tree structure q and leaf weights w correspond to a specific f_k.
   To learn the weights w used in the model, the following regularized objective is minimized [17].

   $$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{9}$$

   $$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{10}$$

   Here l is a differentiable convex loss function, $\hat{y}_i$ is the prediction and y_i is the target. The second term penalizes

a complex model regarding the size of the weights and the number of leaves [17].

2. Gradient Tree Boosting
Since the equation

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{11}$$

contains functions as parameters it cannot be optimized using traditional optimization methods in Euclidean space and has to train in an additive manner. The prediction of instance i at iteration t is defined as $\hat{y}_i^{(t)}$. Then $\hat{y}_i^{(t)}$ is calculated by adding $f_t(\vec{x}_i)$ to $\hat{y}_i^{(t-1)}$ [17]. Therefore the objective to minimize turns into

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\vec{x}_i)) + \Omega(f_t) \tag{12}$$

The greedy approach is to add the f_t that improves the model the most. By using a second-order approximation of the objective, it turns into

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\vec{x}_i) + \frac{1}{2} h_1 f_t(\vec{x}_i)] + \Omega(f_t) \tag{13}$$

$$\text{where } g_1 = \frac{\partial l(\hat{y}_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \text{ and } h_i = \frac{\partial^2 l(\hat{y}_i, \hat{y}_i^{(t-1)})}{\partial \left(\hat{y}_i^{(t-1)}\right)^2} \tag{14}$$

A simplified objective function $\tilde{\mathcal{L}}^{(t)}$ at step t is obtained by removing the constant terms

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\vec{x}_i) + \frac{1}{2} h_1 f_t(\vec{x}_i)] + \Omega(f_t) \tag{15}$$

Define $I_j = \{i | q(\vec{x}_i) = j\}$ as the instance set of leaf j meaning all the instances that correspond to the leaf. The equation can be rewritten by expanding Ω

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n} [g_i f_t(\vec{x}_i) + \frac{1}{2} h_1 f_t(\vec{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2] + \gamma T \end{aligned} \tag{16}$$

For a fixed structure $q(\vec{x})$ the optimal weight w_j^* of leaf j is calculated by

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{17}$$

The corresponding optimal value is given by

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{18}$$

Equation 15 can be used as a scoring function to measure the quality of the tree structure q. Usually, it is impossible to enumerate all the possible tree structures q. A greedy algorithm is developed by starting from a single leaf and iteratively adding. Assume that I_R and I_L are instance sets of the right and left nodes after a split. Letting $I = I_R \cup I_L$ then the loss reduction after the split is given by [17]

$$\mathcal{L}_{\text{split}} = \frac{1}{2} \left[\frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} + \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} - \gamma \right] \tag{19}$$

Equation 16 is the formula that is used in practice in XGBoost [17].

3. Split Finding Algorithms Using equations 14 and 15, two algorithms for split finding can be written according to the following pseudocode [17].

Algorithm 1 Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d feature dimension
$gain \leftarrow 0$
G $\leftarrow \sum_{i \in I} g_i$, $H \leftarrow sum_{i \in I} h_i$
for k=1 **to** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j om sorted(I, *by* $\vec{x}_{jk}$) **do**
    $G_L \leftarrow G_L + g_j$ $H_L \leftarrow H_L + h_J$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
  end for
end for
Output: Split with maximum score

Algorithm 2 Approximate Algorithm for Split Finding

for k=1 **to** m **do**
  Propose $S_k = \{s_{k1}, s_{k2}, s_{k3}, ..., s_{kl}\}$ by percentilies on feature k
  Proposal can be done per tree (global), or per split(local).
end for
for k=1 **to** m **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \vec{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \vec{x}_{jk} > s_{k,v-1}\}} h_j$
end for
Follow same step as in previous section to find maximum score only among proposed splits.

III. Methods

A. Selection of Machine Learning Models

A binary classifier was to be created that could distinguish sepsis using patient data. The random forest and XGBoost models were selected to study their potential for this endeavor using the implementations of the sci-kit learn and XGBoost libraries in python, respectively [18], [19].

B. Study population

Data was acquired from the Herlenius Research team at Karolinska Institutet that are currently conducting research regarding neonatal healthcare. The population consisted of

very low birth weight infants (< 1500 g) hospitalized in the Neonatal Intensive Care Unit (NICU) at Karolinska University Hospital, Stockholm, Sweden [9].

C. Data Description

Time-series data were collected from all patients was collected from a high-frequency Phillips IntelliVue MX800 Patient Monitor [9]. The data initially contained monitor data sampled at 1 Hz. The times-data was split into windows of 70 minutes where 19 different features were extracted. If a patient received a sepsis diagnosis at a specific time instance, all the windows 24 hours before the time of the diagnosis was categorized as sepsis-like. Sepsis-like windows were set to the value 1, and non-sepsis-like time windows were set to the value 0, and these values corresponded to the target y_i in our binary classification problem. Moreover, five more features were also provided that included parameters regularly updated in the medical history. In total, the data consisted of 24 features. Table I provides a list of the target label and all the features that were provided in the data.

Data is provided in a tabular manner where each data row corresponds to a specific time window. The data initially contained 134668 data rows from 118 different patients. A total of 10 patients experienced sepsis during their hospitalization together, totaling 556 rows included sepsis-like characteristics, thus having a target_y $= 1$.
It can be noticed that many features seem to contain large negative values down to -99999; also there appears to exist missing data for the feats_cirk_vikt feature. A more detailed description of the data, including the median and percentiles, can be found in the appendix of the project.

D. Preprocessing

The data in its unmodified state is not universally usable due to several reasons. The following issues have been identified and need to be dealt with. When discussing the following naming convention is used in this project: the feature vector corresponds to a row, and one specific value in the feature vector is called a data point.

- Erroneous data points - Some data points have been identified where the value of some features are exactly -99999 or -9999 and are far outliers relative to the rest of the data. The reason for this data is related to errors during the patient monitoring data collection.
- Categorical data - The unmodified data includes the feats_group_uid feature, which is categorical. Many machine learning algorithms do not work with categorical data [7].
- Large variations in the data - There exist large variations in the data which may affect the weights of the internal parameters in distance-based models [7].
- Missing data - One of the features has missing data that needs to be dealt with. The unmodified data contains 6595 missing data points in only the feats_cirk_vikt feature. If all the rows containing a missing data point were removed, it would result in an additional 158280 data points lost. This is equal to almost 5% of all rows.

With a background in solving the aforementioned issues, the following data preprocessing steps are taken. These steps aim to prepare the data for the random forest and XGBoost algorithms but also prepare for other eventual algorithms that would want to be tested in the future, including distance-based machine learning algorithms.

1. Label Encoding - The 'group_uid' for the entire dataset is encoded using a Label Encoder, creating a 1-to-1 mapping from the String domain to the integer domain. This is to facilitate future slicing and grouping of the dataset.
2. Removal of Erroneous data points - Some of the data points from features that initially were collected from the patient monitor was exactly equal to -99999 and -9999. Since the probability of achieving an exact integer in any continuous distribution is zero, there exists no risk of real data being removed.
3. Scaling - This part is not necessary for random forests and XGBoost. However, regarding distance-based algorithms, the internal parameters are affected by the values of the datapoint. Having a large variance in the sizes of the data points makes it more difficult to fit appropriate internal parameters [7]. All the data except 'target_y' and 'group_uid' is scaled using a standard scaling that removes each data point's mean within a feature and scales to unit variance.
4. Missing Data - The remaining missing data points are assigned values based on the neighboring data points. More specifically, a KNN imputer is used, which stands for K nearest neighbor imputer. Initial testing resulted in the selection of $K = 5$ due to not significantly changing the mean and the standard deviation of the feats_cirk_vikt feature. In other words, the algorithm looks at the five nearest neighbors. The specifics of how the algorithm works are explored in further detail in the appendix to the project.

Table I contains a description of the data after removing erroneous data points. A more detailed description describes the data after each step also, including the median and percentiles, can be found in the appendix to the project.

E. Cross-Validation for Highly Imbalanced Grouped Time Series Data

Data is needed both to train the internal parameters and to test the performance of the model data. Learning the internal parameters of a prediction function and testing on the same data results in a methodological mistake. The model could simply repeat the labels that it has seen without learning any patterns [7]. A trivial method to carry out such a split would be to randomly select a specific percentage in the training set and the rest to be in the testing set. Typical values are to select about 70% to 80% in the training set [20]. However, when trying different hyperparameters of the models, there may exist situations where part model finds patterns in the training set that does not exist in the testing set. This situation is called overfitting, where the patterns found do not display the entire truth of the data. Moreover, indirect knowledge about

TABLE I
FEATURES THAT WERE PROVIDED AND A DESCRIPTION OF THE DATA AFTER ERRONEOUS DATA POINTS HAVE BEEN REMOVED

Feature Name	Description	Count	Mean	Std	Min	Max
feats_btb_mean	The mean value of the beat to beat interval over a time window.	95849	0.000203	0.00104	-0.0284	0.0239
feats_rf_mean	The mean value of the respiratory frequency over a time window.	95849	50.2	11.5	8.01	97.0
feats_spo2_mean	The mean value of the oxygen saturation over a time window.	95849	93.5	3.47	35.6	100.0
feats_btb_std	The standard deviation value of the beat to beat interval over a time window.	95849	0.0201	0.0144	0.000556	0.241
feats_rf_std	The mean value of the respiratory frequency over a time window.	95849	13.5	4.30	0.0	38.1
feats_spo2_std	The mean value of the oxygen saturation over a time window.	95849	4.31	2.73	0.0	27.8
feats_btb_maximum	The maximum value of the beat to beat interval over a time window.	95849	0.146	0.149	0.0016	1.64
feats_rf_maximum	The maximum value of the respiratory frequency over a time window.	95849	89.7	16.52	15.3	163
feats_spo2_maximum	The maximum value of the oxygen saturation over a time window.	95849	99.7	0.911	79.9	100.0
feats_btb_minimum	The minimum value of the beat to beat interval over a time window.	95849	-0.0430	0.0265	-0.808	-0.00140
feats_rf_minimum	The minimum value of the respiratory frequency over a time window.	95849	20.1	6.85	0.394	55.3
feats_spo2_minimum	The minimum value of the oxygen saturation over a time window.	95849	74.5	14.7	0.0333	100.0
feats_btb_skew	The sample skewness of the beat to beat interval over a time window.	95849	2.18	2.93	-5.24	14.8
feats_rf_skew	The sample skewness of the respiratory frequency over a time window.	95849	0.151	0.626	-17.8	12.7
feats_spo2_skew	The sample skewness of the oxygen saturation over a time window.	95849	-1.34	1.19	-23.2	2.51
feats_btb_kurtosis	The kurtosis of the beat to beat interval over a time window. A measure of the ”tailedness” of the sampling points.	95849	20.4	31.8	-1.33	240
feats_rf_kurtosis	The kurtosis of the respiratory frequency over a time window. A measure of the ”tailedness” of the sampling points.	95849	0.0564	4.33	-3.0	341
feats_spo2_kurtosis	The kurtosis of the oxygen saturation over a time window. A measure of the ”tailedness” of the sampling points.	95849	4.01	10.3	-3.0	592
feats_btb_sampAs	The sample asymmetry of the beat to beat interval over a time window. Assessing the asymmetry of the sampling points [10].	95849	3.64	5.72	0.0652	276
feats_btb_sampEn	The sample entropy of the beat to beat interval over a time window. Assessing the complexity and thus possible deterioration [10].	95849	0.410	0.150	0.0124	1.01
feats_cirk_vikt	The weight of the patient during the time window. (Not sampled through the monitor)	91550	1.28	0.470	0.497	3.72
feats_bw	The birth weight of the patient. (Not sampled through the monitor)	95849	839	258	400	150
feats_sex	Categorical data with values of biological sex at birth. Either 1 or 2 for males and females. (Not sampled through the monitor)	95849	1.56	0.496	1.0	2.0
feats_pnage_days	Patient age in days since birth. Negative days are possible due to inconstencies with registration of exact birth time. (Not sampled through the monitor)	95849	31.3	22.2	0.0250	13
feats_group_uid	An anonymized personalized string for each patient. (Not sampled through the monitor)4					
target_y	1 for sepsis-like behavior over a time window, 0 for non-sepsis like.					

the testing set through iterative evaluation of the model may "leak" into the choice of hyperparameters, and in such an instance, the testing set does not show an unbiased picture of the model's performance. A possible option to overcome this problem is to partition the original dataset into three parts where a validation set is added. This data is not touched until the final valuation of the model after the hyperparameters have been found. Nevertheless, this partitioning drastically reduced the number of samples available for training the model [21]. Since the available data was heavily imbalanced, with a small proportion of the data having positive labels, a partition of the dataset into three parts would result in partitions with only about 100 to 200 sepsis-like rows.

A solution to reduce overfitting without using a validation set is to use cross-validation. The principle idea is that the model is trained several times for a specific hyperparameter choice with different choices of test sets each time. Every instance of model training is called a split. If the model would overfit one of the splits, then an eventual loss of generalization due to overfitting would affect the performance of the other splits. There exist several types of methods of cross-validation, and the choice of the cross validator depends on the data. The data used in this report had the following characteristics that need to be taken into account. (i) The data is heavily imbalanced, with less than one percent of the data points having a positive label. If a split is selected entirely at random, there exists a probability that no positive labels are included in either the training or the testing set of that split. (ii) The data included time-series data grouped on a patient basis. It is salient for any cross-validation to take into not having data rows from the same patient in both the training set and test set of the split. This would result in a methodological error since the model possibly would train on past data and test on future data from the same patient finding patient-specific patterns instead of general patterns.

The following cross-validation iterators were considered from which one was selected to be used.

- k-fold - This cross-validation iterator randomly divides the data into k groups of samples where all the samples. In total there are k splits where one of the groups is selected as the testing set for each split. In total the entire dataset will have been used in testing across all the splits. However, this cross-validation iterator does not solve either issue (i) or (ii) [21].

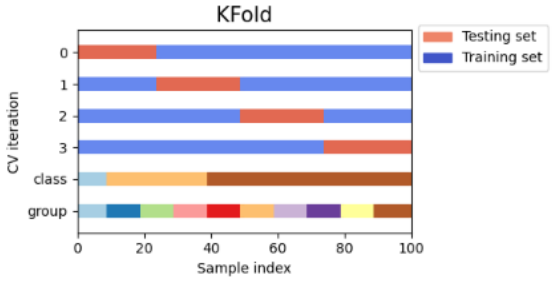

Fig. 1. How the data can be split in a GroupKFold. Source: [21]

- Group k-fold - This cross-validation iterator works in the same way as k-fold; however it ensures that that the same group is not represented in both the training and testing set. The division of groups is made possible by providing an id with each data row. In the data used in this report, the id corresponds therefore to the feats_group_uid, meaning each patient. Therefore this cross-validation iterator takes into account issue (ii) but not (i). Due to an imbalance of the data the sizes of the training and testing sets in each split are not necessarily the same [21].

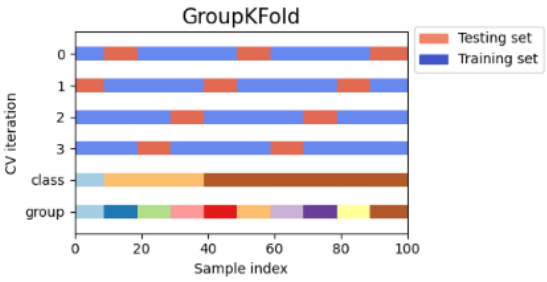

Fig. 2. How the data can be split in a GroupKFold. Source: [21]

- Stratified k-fold - This cross-validation iterator is a variation of the k-fold where each set in the splits contains approximately the same percentage of samples of each target class as the complete set. This means that the iterator tries to preserve the ratios of the classes in both the training and the test set. This cross-validation iterator solves the issue (i) but not (ii). Due to imbalances of the data, the ratios are not necessarily the same but given that it is possible to divide the data a split will not result in a set missing any positive labels [21].

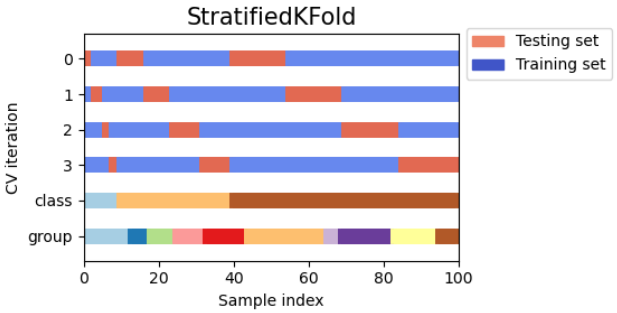

Fig. 3. How the data can be split in a StratifiedKFold. Source: [21]

- Stratified group k-fold - This cross-validation iterator combines both the group k-fold and stratified k-fold cross-validation iterators. Thus, this iterator preserves both the ratios of classes in each split and keeps each group within either the test or training split within a single split. This cross-validation iterator solves both issues (i) and (ii).

Finally, the best cross-validation iterator that caters to the data that does not provide methodological errors is the Stratified

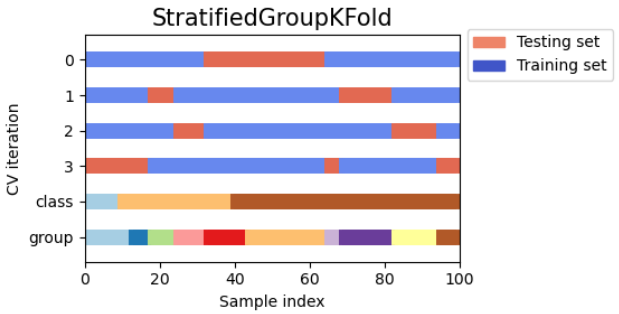

Fig. 4. How the data can be split in a StratifiedGroupKFold. Source: [21]

Group k-fold cross-validation iterator. Henceforth, when referring to a cross-validation iterator it is assumed that stratified group k-fold is used.

F. Model Evaluation for Highly Imbalanced Data

The four possible outcomes when the model makes a prediction are the following:

1) True Positive (TP): True positives occur when the model correctly predicts that a patient has sepsis.
2) True Negative (TN): True negatives occur when the model correctly predicts that a patient does not have sepsis.
3) False Positives (FP): False positives occur when the model incorrectly predicts that a patient has sepsis.
4) False Negatives (FN): False Negatives occur when the model incorrectly predicts that a patient does not have sepsis. sepsis

With these four outcomes, they can be used to determine the overall quality of the model. Typically, the following metrics are used [22].

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{20}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{21}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{22}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{23}$$

$$\text{True Positive Rate} = \text{TPR} = \frac{\text{TP}}{\text{P}} \tag{24}$$

$$\text{False Positive Rate} = \text{FPR} = \frac{\text{FP}}{\text{P}} \tag{25}$$

However, due to the data being highly imbalanced, the accuracy would be high just by creating a model that predicts all inputs as non-sepsis-like. Given the distribution of classes in the provided data, such a model would have an accuracy of about 99.5% even if $\text{TP} = 0$. Moreover, these metrics are threshold-dependent, which is not of importance in the process of discriminating between sepsis-like and non-sepsis-like time windows. A better approach is thus to look at the receiver operating characteristic, which is received by plotting the true positive rate against the false positive rate. In other words, the ROC curve shows the true positive rate for the classifier given an accepted value of the false positive rate. A random classifier follows a linear slope in which the true positive rate and the true negative rate are always equal. The area under the ROC, also called the ROC AUC, can be calculated to be used as a scoring metric. A ROC AUC of 0.5 indicates that there is no discrimination between classes, and 1 indicates perfect discrimination [22]. Henceforth, the ROC AUC will be used as the scoring metric in all machine learning models.

G. Hyperparameter Tuning

1) Workflow: An important part of training a machine learning model is to find the parameters that generate the best performance according to an evaluation method, which in this instance is the best ROC AUC. The workflow of finding the best hyperparameters consists of first defining a hyperparameter space meaning all potential values of hyperparameters that aims to be tested. Combinations from the hyperparameter space are tested together with a cross-validation iterator. A brute force solution would consist of testing all the possible combinations of parameters from the parameters space to select the best performing combination. After having found the retrained model, final cross-validation over the entire dataset is tested to find a final evaluation of the performance of the model.
The specific hyperparameters that are available to work within the parameter space are the machine learning model and the specific implementation in the library.

2) Random Forest Hyperparamters: The following hyperparameters for the random forest model are tuned according to their implementation in the sci-kit-learn library [18].

- n `estimators` - The number of trees in the forest. The default value is 100. Typical values to consider are integers ranging from 10 to 300 [18].
- `maximum depth` - The maximum depth of the tree. The default value is unlimited. Typical values to consider are integers ranging from 1 to 30 [18].
- `maximum features` - The number of features to consider when looking for the best. The default value is the same as the number of features which correspond to 24 for the provided dataset. Typical values to consider are integers ranging from 1 to the number of features $= 24$ [18].
- `minimum samples split` - The minimum number of samples required to split an internal node. The default value is 2. Typical values to consider are integers ranging from 2 to 30 [18].
- `minimum samples leaf` - The minimum number of samples required to be at a leaf node. The default value is 1. Typical values to consider are integers ranging from 1 to 30 [18].

3) XGBoost Hyperparameters: The following hyperparameters for the random forest model are tuned according to their implementation in the XGBoost library [19].

- n `estimators` - The number of trees [19]. Increasing this will make the model more complex and more likely to overfit. The default value is 100. Typical values to

consider in the hyperparameter space are integers ranging from 10 to 500 [19].
- `maximum depth` - Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. The default value is 6. Typical values to consider in the hyperparameter space are integers ranging from 1 to 10 [19].
- `learning rate` - Step size shrinkage used in the update to prevent overfitting. After each boosting step, weights of new features can be obtained directly, and the `learning rate` shrinks the feature weights to make the boosting process more conservative. The default value is 0.3. Typical values to consider in the hyperparameter space are values ranging from 0.01 to 1 in 0.01 step [19].
- `minimum child weight` - Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than `minimum child weight`, then the building process will give up further partitioning. The default value is 1. Typical values to consider in the hyperparameter space are values ranging from 0.5 to 1.5 in 0.01 steps [19].
- `gamma` - Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger `gamma` is, the more conservative the algorithm is. The default value is 0. Typical values to consider in the hyperparameter space are values ranging from 0 to 0.5 in 0.01 steps [19].
- `column sample by tree` - The subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. The default value is 1. Typical values to consider in the hyperparameter space are values ranging from 0.5 to 1 in 0.01 steps [19].
- `subsample` - `subsample` ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data before growing trees and this will prevent overfitting. Subsampling will occur once in every boosting iteration. The default value is 1. Typical values to consider in the hyperparameter space are values ranging from 0.5 to 1 in 0.01 steps [19].
- `lambda` - L2 regularization term on weights. Increasing this value will make the model more conservative, punishing complex models. The default value is 1. Typical values to consider in the hyperparameter space are values ranging from 0.5 to 1.5 in 0.01 steps [19].
- `alpha` - L1 regularization term on weights. Increasing this value will make the model more conservative, punishing complex models. The default value is 0. Typical values to consider in the hyperparameter space are values ranging from 0 to 0.5 in 0.01 steps [19].
- `scale positive weight` - Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative instances) / sum(positive instances) [19].

4) Hyperparameter spaces notation: The following notation is used to define the entire hyperparameter space H of q different parameters:

$$H = P_1 \times \cdots \times P_q = \bigotimes_i P_i \tag{26}$$

Each hyperparameter space P_i corresponds to hyperparameter i and consists of the set of possible hyperparameter values that can be chosen. The following notation for the sets is used:

$$[a, b, s] = \text{All real values from a to b (including b) with step sizes of } s \text{ starting from } a \tag{27}$$

The size or cardinality of the hyperparameter space is given by $|H|$.

5) The cardinality of initial hyperparameter spaces: Using the notation in equations (26) and (27) the cardinality of the hyperparameter space that uses the entire range of typical values would be: $|H_{\text{Typical values Random Forest}}| \sim 10^8$ and $|H_{\text{Typical values XGBoost}}| \sim 10^{14}$. Assuming a generous computation time of 0.01 seconds for each cross-validation, an entire exhaustive search of the entire hyperparameter space would take about a month and 20 thousand years, respectively. An exhaustive search is therefore not an option.

6) Hyperparameter Search: To reduce the number of parameter combinations that are searched for from the initial total hyperparameter spaces $H_{\text{Typical values Random Forest}}$ and $H_{\text{Typical values XGBoost}}$ a combination of the following methods is used.

- Cross validated grid search - This method exhaustively goes through all the hyperparameter combinations in H and runs an iteration of cross-validation to calculate an average scoring value using an evaluator function which is ROC AUC [23].
- Cross validated random halving grid search - This method starts with selecting one ”resource” parameter that has the property that it correlates with the increased resource usage of the model as it increases. For both the random forest and XGBoost a suitable hyperparameter is the n `estimators`. The minimum value and maximum values of the n estimators follow the limits of the set $P_{n\text{ estimators}}$. From $H \setminus P_{n\text{ estimators}}$ random values are sampled. During each iteration of the search, the cross-validated score of each chosen combination is calculated and the best third combinations are chosen to continue to the next iteration. This continues until there only exists one combination left. The number of candidates in the initial iteration is chosen so that the last iteration uses as many resources as possible within the limits of the maximum value for the ”resource” parameter [24].

The chosen approach to finding the hyperparameter spaces in this project is a combination of, (1) a cross-validated random halving search to find a starting position using $H_{\text{Typical values Random Forest}}$ and $H_{\text{Typical values XGBoost}}$, (2) one or more narrow cross-validated grid searches with Hs that use the previous best parameters $\pm \approx 10\%$ of range in $H_{\text{typical values}}$ for each P_i. The hyperparameter space is extended if the best parameter found is on the boundary of the H. In this step, only two to four hyperparameters are searched for in each iteration of the grid search using the previous best value when continuing with the next search.

This limitation is needed to reduce the cardinality of the Hs. When finding the hyperparameters for the random forest all the grid searches are divided into two groups in the following order [n `estimators`, `maximum depth`, `maximum features`],[`minimum samples split`, `minimum samples leaf`]. The reasoning is that the first group mainly deals with the complexity of the model and the second group deals with node splits [18].
When finding the hyperparameters for the XGBoost model the grid searches are divided into three groups in the following order [n `estimators`, maximum depth, `learning rate`, `minimum child weight`], [`gamma`,`column sample by tree`, `subsample`], [`lambda`, `alpha`]. The reasoning is that the first group mainly deals with general tree parameters, the second group deals with node construction, and the third group deals with general regularization parameters that penalize complex models. The regularization parameters are also left out during the initial cross-validated random halving search.

IV. Results

A. Random Forest

1) Before hyperparameter Tuning: Using the default parameters of the random forest model before any tuning resulted in a mean ROC AUC of 0.67 with a standard error of 0.04 in the final cross-validation iteration. The ROC curve is presented in Figure5.

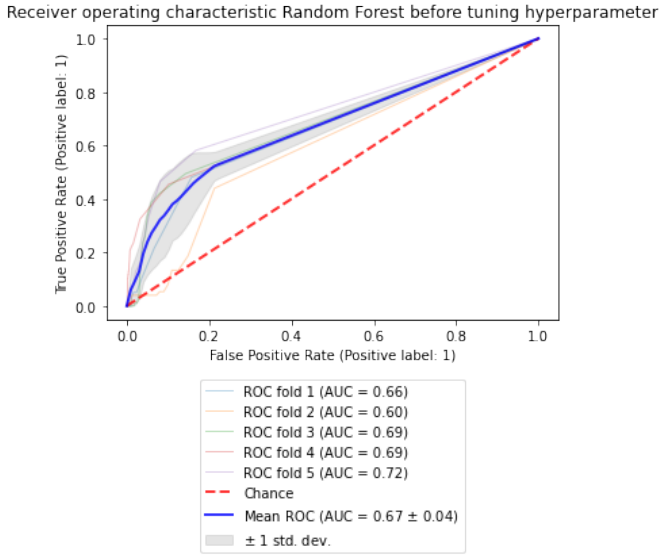

Fig. 5. Results of the Random Forests model before tuning for hyperparameters

2) Final Hyperparameters: The following hyperparameters were found as the best hyperparameters after following the tuning procedure described in the methods section. See the appendix for a detailed description of all steps.

$$\begin{cases} n \text{ estimators} = 190 \\ \text{maximum depth} = 1 \\ \text{maximum features} = 2 \\ \text{minimum samples split} = 3 \\ \text{minimum samples leaf} = 23 \end{cases} \tag{28}$$

The final performance of the model received a mean ROC AUC of 0.842 with a standard error of 0.10 in the cross-validation iteration. The ROC curve is presented in Figure6.

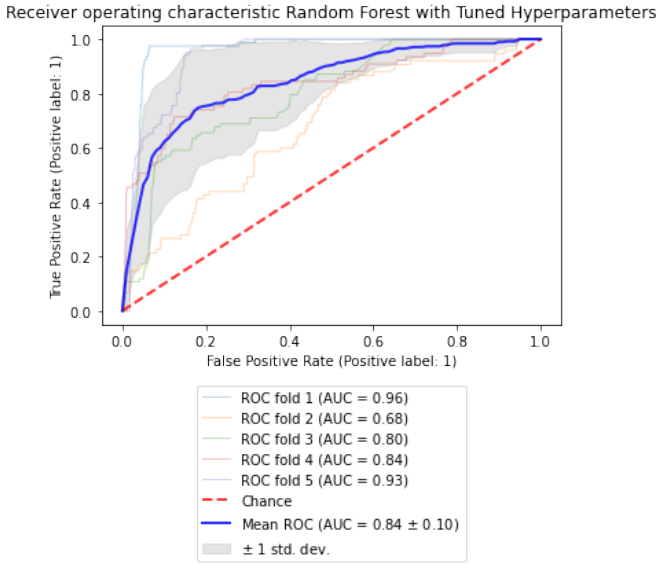

Fig. 6. Results of the Random Forests model after tuning for hyperparameters

B. Results XGBoost

1) Before hyperparameter Tuning: Using the default parameters of the XGBoost model before any tuning, except for the `scale positive weight` hyperparameter that, according to the typical convention, was set to sum(negative instances) / sum(positive instances), which for the dataset was equal to 232.78, resulted in a mean ROC AUC of 0.75 with a standard error of 0.14 in the final cross-validation iteration. The ROC curve is presented in Figure5.

2) Final hyperparameters: The following hyperparameters were found as the best hyperparameters after following the tuning procedure described in the methods section. See the

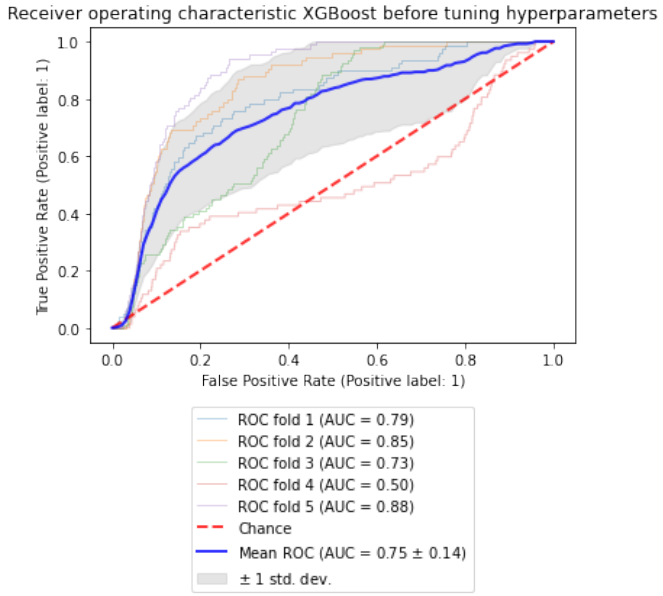

Fig. 7. Results of the XGBoost model before tuning for hyperparameters

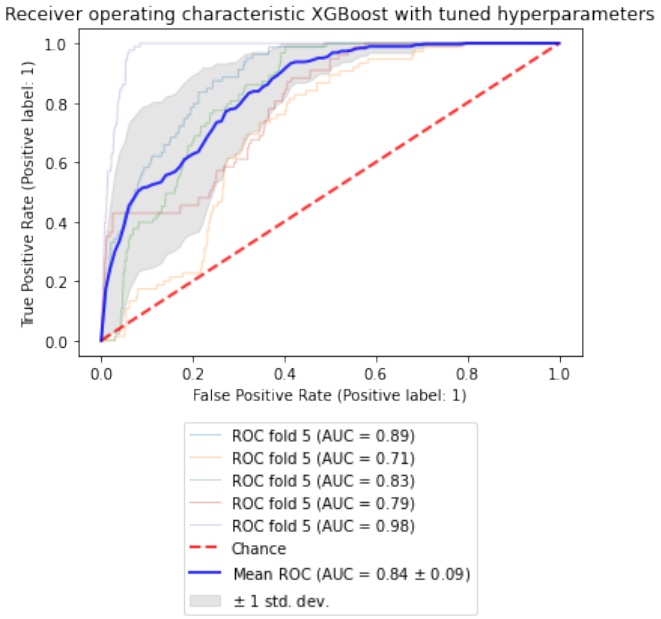

Fig. 8. Results of the XGBoost model after tuning for hyperparameters

appendix for a detailed description of all steps.

$$\begin{cases} n\ \texttt{estimators} = 140 \\ \texttt{maximum depth} = 1 \\ \texttt{learning rate} = 0.34 \\ \texttt{minimum child weight} = 0 \\ \texttt{gamma} = 0 \\ \texttt{column sample by tree} = 0.92 \\ \texttt{subsample} = 0.6 \\ \texttt{lambda} = 0.86 \\ \texttt{alpha} = 0.17 \\ \texttt{scale positive weight} = 232.78 \end{cases} \tag{29}$$

The final performance of the model received a mean ROC AUC of 0.840 with a standard error of 0.10 in the cross-validation iteration. The ROC curve is presented in Figure8.

V. Discussion

A. Model Performance

The final random forest model demonstrated a mean ROC AUC $\overline{\text{AUC}} = 0.84$ with a standard deviation $s = 0.1$ and the final XGBoost model demonstrated a mean ROC AUC $\overline{\text{AUC}} = 0.84$ with a standard deviation $s = 0.09$. Any value for the ROC AUC that is greater than 0.5 indicates that the model has the ability to distinguish between sepsis-like and non-sepsis-like time windows [22]. It is natural to perform a hypothesis test to evaluate whether $\overline{\text{AUC}}$ differs significantly from 0.5. The test statistic given by $(\overline{\text{AUC}} - 0.5)/d$ is approximately normally distributed [25]. The standard error $d = s/\sqrt{n}$ where $n = 5$ is equal to the number of folds in the cross-validation iterator. With the null and alternative hypotheses defined as $H0 : \overline{\text{AUC}} = 0.5$ versus $H1 : \overline{\text{AUC}} \neq 0.5$ yields a test statistics with p-values of $7 \cdot 10^{-12}$ and $9 \cdot 10^{-14}$ for the random forest model and XGBoost model respectively. Thus, significantly rejecting the null hypothesis, showing that both models can distinguish sepsis-like windows. The grade of a

TABLE II
GRADES OF DIAGNOSTIC TESTS ACCORDING TO THE ROC AUC [25]

ROC AUC	Grade
0.9 - 1	Outstanding
0.8 - 0.9	Excellent
0.7 - 0.8	Acceptable
0.6 - 0.7	Poor
≤ 0.6	Unacceptable

diagnostic test predictor in medicine can be defined according to the categories in table II [25]. Testing whether a model at least has a specific grade g can be done by formulating the null hypotheses $H0 : \overline{\text{AUC}} < l_g$ against the alternative hypothesis $H1 : \overline{\text{AUC}} \geq l_g$ where l_g is the lower limit int the ROC AUC for the grade g. The p values for models that at least achieve different grades are given in Table III It is shown

TABLE III
P VALUES FOR THE MODELS TO AT LEAST HAVE A SPECIFIC GRADE

At least ...	XGBoost p-value	Random forest p-value
Outstanding	0.91	0.88
Excellent	0.19	0.20
Acceptable	$1.1 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$
Poor	$7.1 \cdot 10^{-8}$	$6.0 \cdot 10^{-7}$

that both the random forest and the XGBoost models with significance have at least an acceptable grade as a diagnostic test. Although measures have been taken to reduce bias and overfitting, it is still not guaranteed that the model performs at the calculated level. Only patients from Sweden have been used in the data set, which could have introduced selection bias. Introducing a validation set would have provided more information on overfitting. A common denominator regarding

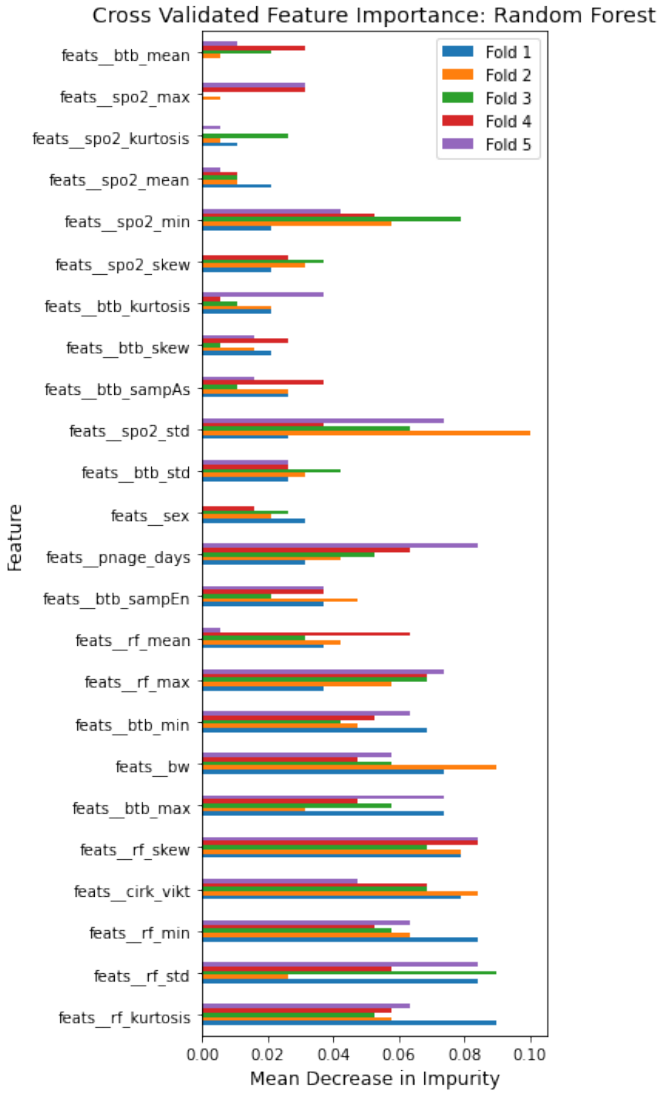

Fig. 9. Feature importance of the Random Forest model

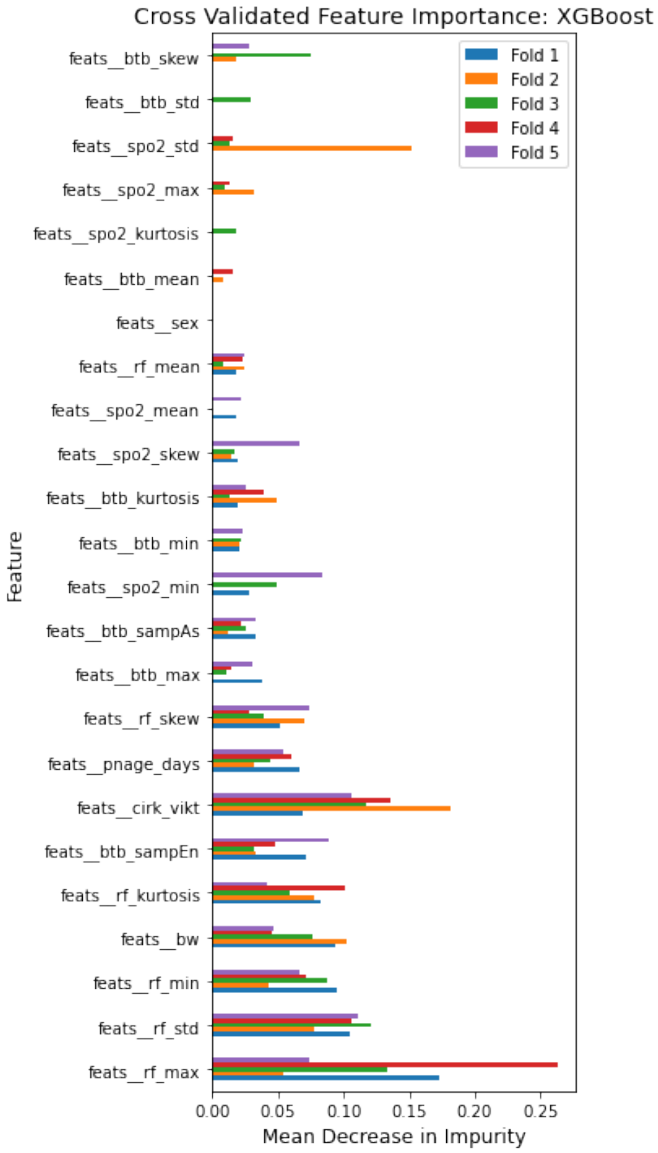

Fig. 10. Feature importance of the XGBoost model

the pitfalls of the results is that they could be partially eluded if the data set had been larger.

1) Comparison with the SOFA score: The SOFA score has a highly distinctive ability to predict sepsis with a ROC AUC of 0.89 [26]. However, machine learning models that are close to that level would still be of interest, especially if they do not require invasive methods and increased tasks for healthcare personnel, as in the case of the SOFA score [4].

B. Feature Importance

Feature importances for the final random forest and XGBoost models were calculated by finding the mean decrease in impurity when a feature is removed. The (Gini) impurity is a metric similar to entropy from information theory. A large decrease when a feature is removed signifies greater importance. Calculations were performed on a cross-validation iteration to get a sense of whether the importances were random or not. The importance of each parameter is found in Figures 9 and 10.

1) Insights: For both models, the feature importances for the different folds fluctuated, indicating a component of randomness within the models. However, it is possible to find some general patterns that apply to both models.

- Features of high importance - All features regarding the respiratory frequency and body weight.
- Features of low importance - All features regarding oxygen saturation and sex.
- Features with varying importance - The beat-to-beat interval features achieved varying importance with the maximum and sample entropy features scoring on the higher side within the group.

The results go somewhat against the SOFA score that takes into account oxygen saturation, which was shown to be less important, but none of the respiratory frequencies, bodyweight measurements, or beat-to-beat intervals were shown to be important [11].

C. Model Complexity

Both the random forest and XGBoost models ended up with hyperparameters toward the less complex side, the most significant hyperparameter being a maximum depth of only 1. This signifies that only one comparison of one feature is carried out in every tree. It is up for discussion whether a single node even should be called a tree. An analysis of the model performance dependence on hyperparameter values was performed using data from the hyperparameter tuning procedure. Some of the discovered relationships are as follows.

- Keeping all other hyperparameters the same, the performance of the XGBoost model was greatly reduced with depths of 5 having ROC AUCs of 0.65 to 0.7, while depths of 1 and 2 performed between 0.8 and 0.85.
- Keeping all other hyperparameters the same, the performance of the random forest model was greatly reduced for `maximum features` greater than 20 with a ROC AUC of 0.65 to 0.7, while depths of `maximum features` less than 5 performed closer to 0.8.
- For both models, performance plateaued for n `estimators` larger than 150.

See the appendix for more relationships and graphs of the dependence of the ROC AUC score on the hyperparameters for both the training and the testing sets.

1) Implications for explainability: Since both models tended towards the less complex side, a relevant question is whether the models chosen are too complex. On the one hand, it can be argued that there exists room for adding more features and finding more complex patterns. On the other hand, one could see the possible implications for explainability. It is highly debated whether artificial intelligence in healthcare should be explainable, which in essence means that a human should understand why the model makes its decisions [27]. In the current state of the world, it is easier to overcome legal and ethical hurdles with explainable models [27]. Together with the fact that simpler models tend to be more explainable, this implies that having a simple model that is the best performing model brings hope for easy adoption [28]. Since both the random forest and XGBoost are built on the concept of a decision tree that asks simple questions at each node, it is theoretically possible to reverse engineer the decision tree and come up with an easier set of questions that could replace the SOFA scoring system.

VI. Conclusion

In conclusion, the use of random forests and XGBoost for the detection of neonatal sepsis is feasible, offering performance within the range of the SOFA scoring system. At the same time, requiring less invasive methods and being automated in the sense that healthcare personnel is not burdened with more tasks. However, both models tended to perform better with reduced complexity. This suggests that it could be possible to gain benefit from adding more features, thus discovering more complex patterns in the data. Or else generate even simple models that focus on the explainability of the data to ease the adoption of the algorithm. One possible method would be to reverse engineer the questions asked at specific nodes in the decision tree ensemble models.
The project has also found appropriate methods for validation and evaluation that could be generalized to all types of highly unbalanced grouped tabular data that can generalize to other similar applications.

A. Future Work

A possible extension of this project is to calibrate the final models. Currently, due to the usage of non-threshold-dependent scoring systems, the model does not find the exact probability of input being sepsis-like. Generating the probabilities from the model predictions would require calibration with prior probabilities [7]. Another extension is to reverse engineer the models to find simple questions that enhance explainability. Other future work includes acquiring a larger dataset with more sample points and features and testing other diseases or conditions.

Ethics Statement

All patient data were anonymized by removing personal identification numbers and not providing the exact date and time of the collected data. Therefore, precluding any attempt to identify specific patients. Accessing the data required multi-factor authentication and the generation of one-time passwords for each login session.

Acknowledgment

The authors would like to express gratitude to Antoine Honoré for his support and assistance throughout the course of the project. Furthermore, expressing a special appreciation for the Herlenius Research Team for their work in collecting, working with, and providing the dataset for this project.

Source Code

The source code for the project and all plots can be found in the following GitHub repository: https://gits-15.sys.kth.se/mmmab/Neonatal-Sepsis-Detection-Using-Decision-Tree-Ensemble-Methods-Random-Forest-and-XGBoost.

References

[1] A.-M. Audet, S. Greenfield, and M. Field, "Medical practice guidelines: current activities and future directions," *Annals of Internal Medicine*, vol. 113, no. 9, pp. 709–714, 1990.

[2] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche, C. M. Coopersmith, R. S. Hotchkiss, M. M. Levy, J. C. Marshall, G. S. Martin, S. M. Opal, G. D. Rubenfeld, T. van der Poll, J.-L. Vincent, and D. C. Angus, "The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)," *JAMA*, vol. 315, no. 8, pp. 801–810, 02 2016. [Online]. Available: https://doi.org/10.1001/jama.2016.0287

[3] C. Fleischmann, F. Reichert, A. Cassini, R. Horner, T. Harder, R. Markwart, M. Tröndle, Y. Savova, N. Kissoon, P. Schlattmann *et al.*, "Global incidence and mortality of neonatal sepsis: a systematic review and meta-analysis," *Archives of Disease in Childhood*, vol. 106, no. 8, pp. 745–752, 2021.

[4] E. P. Raith, A. A. Udy, M. Bailey, S. McGloughlin, C. MacIsaac, R. Bellomo, D. V. Pilcher, for the Australian, N. Z. I. C. S. A. C. for Outcomes, and R. E. (CORE), "Prognostic Accuracy of the SOFA Score, SIRS Criteria, and qSOFA Score for In-Hospital Mortality Among Adults With Suspected Infection Admitted to the Intensive Care Unit," *JAMA*, vol. 317, no. 3, pp. 290–300, 01 2017. [Online]. Available: https://doi.org/10.1001/jama.2016.20328

[5] B. Cummings, "Rising healthcare costs are a rising concern," *Journal of Financial Planning*, vol. 35, no. 2, pp. 19–19, 2022.

[6] M. Marć, A. Bartosiewicz, J. Burzyńska, Z. Chmiel, and P. Januszewicz, "A nursing shortage–a prospect of global and local policies," *International nursing review*, vol. 66, no. 1, pp. 9–16, 2019.

[7] R. N. Stuart Peter, *Artificial Intelligence A Modern Approach*, 3rd ed. Harlow, England: Pearson Education Limited, 2016.

[8] A. Honore, D. Liu, D. Forsberg, K. Coste, E. Herlenius, S. Chatterjee, and M. Skoglund, "Hidden markov models for sepsis detection in preterm infants," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020 :*, ser. International Conference on Acoustics Speech and Signal Processing ICASSP. Institute of Electrical and Electronics Engineers (IEEE), 2020, pp. 1130–1134, qC 20210324.

[9] A. Honoré, D. Forsberg, K. Jost, K. Adolphson, A. Stålhammar, E. Herlenius, and S. Chatterjee, "Classification and feature extraction for neonatal sepsis detection," 2022.

[10] K. D. Fairchild and T. M. O'Shea, "Heart rate characteristics: physiomarkers for detection of late-onset neonatal sepsis," *Clinics in perinatology*, vol. 37, no. 3, pp. 581–598, 2010.

[11] A. Oscarson, C. Bjurman, J. E. Wallér, and M. Werner, "Sepsis hos vuxna – tidig upptäckt och initial behandling," *Läkartidningen*, Mar 2017.

[12] A. Rhodes, L. E. Evans, W. Alhazzani, M. M. Levy, M. Antonelli, R. Ferrer, A. Kumar, J. E. Sevransky, C. L. Sprung, M. E. Nunnally *et al.*, "Surviving sepsis campaign: international guidelines for management of sepsis and septic shock: 2016," *Intensive care medicine*, vol. 43, no. 3, pp. 304–377, 2017.

[13] P.-Y. Iroh Tam and C. M. Bendel, "Diagnostics for neonatal sepsis: current approaches and future directions," *Pediatric research*, vol. 82, no. 4, pp. 574–583, 2017.

[14] J. L. Wynn and R. A. Polin, "A neonatal sequential organ failure assessment score predicts mortality to late-onset sepsis in preterm very low birth weight infants," *Pediatric research*, vol. 88, no. 1, pp. 85–90, 2020.

[15] M. J. Aspinall, "Use of a decision tree to improve accuracy of diagnosis." *Nursing Research*, vol. 28, no. 3, pp. 182–185, 1979.

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[18] SKlearn, "Sklearn.ensemble.randomforestclassifier," 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[19] XGBoost, "Xgboost parameters," 2021. [Online]. Available: https://xgboost.readthedocs.io/en/stable/parameter.html

[20] J. Brownlee, "Train-test split for evaluating machine learning algorithms," Aug 2020. [Online]. Available: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

[21] Scikit, "3.1. cross-validation: Evaluating estimator performance," 2022. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[22] X.-H. Zhou, D. K. McClish, and N. A. Obuchowski, *Statistical methods in diagnostic medicine*. John Wiley & Sons, 2009.

[23] SKlearn, "Sklearn.model_selection.gridsearchcv," 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[24] Scikit, "Sklearn.model_selection.halvingrandomsearchcv," 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingRandomSearchCV.html

[25] J. N. Mandrekar, "Receiver operating characteristic curve in diagnostic test assessment," *Journal of Thoracic Oncology*, vol. 5, no. 9, pp. 1315–1316, 2010.

[26] A. K. Toker, S. Kose, and M. Turken, "Comparison of sofa score, sirs, qsofa, and qsofa+ l criteria in the diagnosis and prognosis of sepsis," *The Eurasian Journal of Medicine*, vol. 53, no. 1, p. 40, 2021.

[27] J. Amann, A. Blasimme, E. Vayena, D. Frey, and V. I. Madai, "Explainability for artificial intelligence in healthcare: a multidisciplinary perspective," *BMC Medical Informatics and Decision Making*, vol. 20, no. 1, pp. 1–9, 2020.

[28] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2020.

Appendix to Neonatal Sepsis Detection Using Decision Tree Ensemble Methods: Random Forest and XGBoost

Marwan Al-Bardaji and Nahir Danho

APPENDIX A - UNMODIFIED DATA

TABLE I
CAPTION

	Count	Mean	Std	Min	25%	50%	75%	Max
feats btb mean	134668	-28439.0	45112.0	-99999.0	-99999.0	-0.00010418	0.00037728	0.023877
feats rf mean	134668	-28407.0	45133.0	-99999.0	-99999.0	44.812	54.54	97.018
feats spo2 mean	134668	-28373.0	45154.0	-99999.0	-99999.0	91.844	94.871	100.0
feats btb std	134668	-28439.0	45112.0	-99999.0	-99999.0	0.012186	0.02	0.24101
feats rf std	134668	-28433.0	45116.0	-99999.0	-99999.0	11.001	14.686	38.089
feats spo2 std	134668	-28437.0	45114.0	-99999.0	-99999.0	2.739	4.5539	27.82
feats btb max	134668	-28439.0	45112.0	-99999.0	-99999.0	0.053329	0.13255	1.6416
feats rf max	134668	-28379.0	45150.0	-99999.0	-99999.0	81.389	96.0	163.22
feats spo2 max	134668	-28368.0	45157.0	-99999.0	-99999.0	100.0	100.0	100.0
feats btb min	134668	-28439.0	45112.0	-99999.0	-99999.0	-0.047756	-0.031711	-0.0013959
feats rf min	134668	-28429.0	45119.0	-99999.0	-99999.0	18.111	22.778	55.333
feats spo2 min	134668	-28386.0	45146.0	-99999.0	-99999.0	70.722	81.878	100.0
feats btb skew	134668	-28438.0	45113.0	-99999.0	-99999.0	0.13312	2.4165	14.834
feats rf skew	134668	-28443.0	45110.0	-99999.0	-99999.0	-0.092083	0.33747	12.711
feats spo2 skew	134668	-28441.0	45112.0	-99999.0	-99999.0	-1.608	-0.81382	2.5072
feats btb kurtosis	134668	-28425.0	45122.0	-99999.0	-99999.0	1.0974	14.668	239.58
feats rf kurtosis	134668	-28443.0	45110.0	-99999.0	-99999.0	-0.64149	-0.084224	341.28
feats spo2 kurtosis	134668	-28437.0	45114.0	-99999.0	-99999.0	0.52569	2.8307	591.91
feats btb sampAs	134668	-28437.0	45114.0	-99999.0	-99999.0	0.95311	2.5452	275.82
feats btb sampEn	134668	-28439.0	45113.0	-99999.0	-99999.0	0.34335	0.48042	1.0138
feats cirk vikt	128073	1.2967	0.49423	0.4958	0.92919	1.2139	1.5765	5.253
feats bw	134668	844.64	257.06	400.0	636.0	802.0	1013.0	1498.0
feats sex	134668	1.5655	0.49569	1.0	1.0	2.0	2.0	2.0
feats pnage days	134668	31.543	23.368	-0.85139	12.372	26.915	47.071	251.16

APPENDIX B - PREPROCESSING

A. Data description after removal of erroneous data points

TABLE II
CAPTION

	Count	Mean	Std	Min	25%	50%	75%	Max
feats btb mean	95849	0.00020301	0.0010439	-0.028372	-0.00018921	0.0001638	0.00056228	0.023877
feats rf mean	95849	50.194	11.525	8.0081	43.223	50.542	57.607	97.018
feats spo2 mean	95849	93.491	3.4743	35.618	91.363	93.521	95.971	100.0
feats btb std	95849	0.020122	0.014444	0.00055649	0.011286	0.016114	0.023798	0.24101
feats rf std	95849	13.479	4.3061	0.0	10.486	13.064	16.066	38.089
feats spo2 std	95849	4.3072	2.7283	0.0	2.5023	3.679	5.358	27.82
feats btb max	95849	0.14566	0.14853	0.001649	0.047296	0.085492	0.18946	1.6416
feats rf max	95849	89.722	16.519	15.25	79.111	89.889	100.56	163.22
feats spo2 max	95849	99.681	0.91062	79.078	99.944	100.0	100.0	100.0
feats btb min	95849	-0.042934	0.026538	-0.80757	-0.051221	-0.037783	-0.027746	-0.0013959
feats rf min	95849	20.076	6.851	0.39394	17.111	21.0	24.139	55.333
feats spo2 min	95849	74.461	14.721	0.033333	68.122	77.822	84.444	100.0
feats btb skew	95849	2.1813	2.9255	-5.2424	-0.0014337	0.92294	3.8908	14.834
feats rf skew	95849	0.15149	0.62641	-17.822	-0.16126	0.15826	0.47455	12.711
feats spo2 skew	95849	-1.3403	1.1861	-23.228	-1.7823	-1.1137	-0.61499	2.5072
feats btb kurtosis	95849	20.382	31.826	-1.3333	0.74752	4.5817	27.817	239.58
feats rf kurtosis	95849	0.056369	4.3335	-3.0	-0.70815	-0.34926	0.16646	341.28
feats spo2 kurtosis	95849	4.0127	10.32	-3.0	0.31891	1.5552	4.227	591.91
feats btb sampAs	95849	3.6363	5.7169	0.065211	0.84322	1.5616	3.8936	275.82
feats btb sampEn	95849	0.41042	0.15026	0.012408	0.3141	0.42854	0.5171	1.0138
feats cirk vikt	91550	1.2757	0.47036	0.49672	0.92301	1.194	1.549	3.724
feats bw	95849	839.31	258.12	400.0	636.0	789.0	1011.0	1498.0
feats sex	95849	1.5601	0.49637	1.0	1.0	2.0	2.0	2.0
feats pnage days	95849	31.256	22.156	0.025035	12.83	26.993	46.612	134.26

B. Data after standard scaling

TABLE III
CAPTION

	Count	Mean	Std	Min	25%	50%	75%	Max
feats btb mean	95849	6.428e-17	1.0	-27.373	-0.37572	-0.037554	0.34416	22.678
feats rf mean	95849	-7.008e-16	1.0	-3.6604	-0.60485	0.030175	0.64319	4.0628
feats spo2 mean	95849	-4.6673e-16	1.0	-16.658	-0.61254	0.0086296	0.71387	1.8734
feats btb std	95849	-1.5383e-16	1.0	-1.3546	-0.61175	-0.27749	0.25455	15.293
feats rf std	95849	-1.7911e-17	1.0	-3.1303	-0.69519	-0.096462	0.60055	5.7151
feats spo2 std	95849	3.4707e-16	1.0	-1.5787	-0.66155	-0.23023	0.38516	8.6184
feats btb max	95849	1.2715e-16	1.0	-0.96959	-0.66226	-0.4051	0.29487	10.072
feats rf max	95849	1.027e-17	1.0	-4.5083	-0.64232	0.010129	0.65585	4.4495
feats spo2 max	95849	4.7662e-15	1.0	-22.625	0.2895	0.35051	0.35051	0.35051
feats btb min	95849	3.9339e-17	1.0	-28.813	-0.31225	0.19411	0.57233	1.5652
feats rf min	95849	-6.2789e-17	1.0	-2.8729	-0.43279	0.13485	0.59302	5.1463
feats spo2 min	95849	8.8555e-16	1.0	-5.0559	-0.4306	0.22832	0.67817	1.7349
feats btb skew	95849	1.1817e-17	1.0	-2.5376	-0.7461	-0.43013	0.58436	4.3249
feats rf skew	95849	4.0558e-17	1.0	-28.693	-0.49929	0.010799	0.51573	20.05
feats spo2 skew	95849	-1.1326e-16	1.0	-18.454	-0.3727	0.19105	0.61148	3.2439
feats btb kurtosis	95849	-7.9114e-17	1.0	-0.68232	-0.61694	-0.49647	0.23361	6.8872
feats rf kurtosis	95849	1.3601e-16	1.0	-0.7053	-0.17642	-0.093604	0.025405	78.743
feats spo2 kurtosis	95849	5.8921e-17	1.0	-0.67956	-0.35794	-0.23814	0.020765	56.969
feats btb sampAs	95849	-9.2738e-17	1.0	-0.62466	-0.48857	-0.3629	0.045008	47.61
feats btb sampEn	95849	-6.2922e-16	1.0	-2.6488	-0.64106	0.12056	0.70993	4.0156
feats cirk vikt	91550	-4.6828e-16	1.0	-1.6562	-0.74984	-0.1737	0.58105	5.2052
feats bw	95849	2.4497e-15	1.0	-1.702	-0.78764	-0.19489	0.66518	2.5519
feats sex	95849	-3.5855e-14	1.0	-1.1285	-1.1285	0.88617	0.88617	0.88617
feats pnage days	95849	3.841e-15	1.0	-1.4096	-0.83166	-0.1924	0.69308	4.6488

C. Data after filling missing data points

This corresponds to the final data used in the machine learning models.

TABLE IV
CAPTION

	Count	Mean	Std	Min	25%	50%	75%	Max
feats btb mean	95849	6.428e-17	1.0	-27.373	-0.37572	-0.037554	0.34416	22.678
feats rf mean	95849	-7.008e-16	1.0	-3.6604	-0.60485	0.030175	0.64319	4.0628
feats spo2 mean	95849	-4.6673e-16	1.0	-16.658	-0.61254	0.0086296	0.71387	1.8734
feats btb std	95849	-1.5383e-16	1.0	-1.3546	-0.61175	-0.27749	0.25455	15.293
feats rf std	95849	-1.7911e-17	1.0	-3.1303	-0.69519	-0.096462	0.60055	5.7151
feats spo2 std	95849	3.4707e-16	1.0	-1.5787	-0.66155	-0.23023	0.38516	8.6184
feats btb max	95849	1.2715e-16	1.0	-0.96959	-0.66226	-0.4051	0.29487	10.072
feats rf max	95849	1.027e-17	1.0	-4.5083	-0.64232	0.010129	0.65585	4.4495
feats spo2 max	95849	4.7662e-15	1.0	-22.625	0.2895	0.35051	0.35051	0.35051
feats btb min	95849	3.9339e-17	1.0	-28.813	-0.31225	0.19411	0.57233	1.5652
feats rf min	95849	-6.2789e-17	1.0	-2.8729	-0.43279	0.13485	0.59302	5.1463
feats spo2 min	95849	8.8555e-16	1.0	-5.0559	-0.4306	0.22832	0.67817	1.7349
feats btb skew	95849	1.1817e-17	1.0	-2.5376	-0.7461	-0.43013	0.58436	4.3249
feats rf skew	95849	4.0558e-17	1.0	-28.693	-0.49929	0.010799	0.51573	20.05
feats spo2 skew	95849	-1.1326e-16	1.0	-18.454	-0.3727	0.19105	0.61148	3.2439
feats btb kurtosis	95849	-7.9114e-17	1.0	-0.68232	-0.61694	-0.49647	0.23361	6.8872
feats rf kurtosis	95849	1.3601e-16	1.0	-0.7053	-0.17642	-0.093604	0.025405	78.743
feats spo2 kurtosis	95849	5.8921e-17	1.0	-0.67956	-0.35794	-0.23814	0.020765	56.969
feats btb sampAs	95849	-9.2738e-17	1.0	-0.62466	-0.48857	-0.3629	0.045008	47.61
feats btb sampEn	95849	-6.2922e-16	1.0	-2.6488	-0.64106	0.12056	0.70993	4.0156
feats cirk vikt	95849	0.0031612	1.0	-1.6562	-0.74468	-0.17128	0.57615	5.2052
feats bw	95849	2.4497e-15	1.0	-1.702	-0.78764	-0.19489	0.66518	2.5519
feats sex	95849	-3.5855e-14	1.0	-1.1285	-1.1285	0.88617	0.88617	0.88617
feats pnage days	95849	3.841e-15	1.0	-1.4096	-0.83166	-0.1924	0.69308	4.6488

APPENDIX C - KNN IMPUTER

KNN Imputing is an algorithm used to fill in missing values. The missing value will be predicted in reference to the mean of the neighbors.

APPENDIX D - HYPERPARAMTER TUNING PROCEUDURE FOR RANDOM FOREST

1) Hyperparameter spaces notation: The following notation is used to define the entire hyperparameter space H of q different parameters:

$$H = P_1 \times \cdots \times P_q = \bigotimes_i P_i \tag{1}$$

Each hyperparameter space P_i corresponds to hyperparameter i and consists of the set of possible hyperparameter values that can be chosen. The following notation for the sets is used:

$$\begin{gathered}[a, b, s] = \text{All real values from a to b (including b)} \\ \text{with step sizes of } s \text{ starting from } a\end{gathered} \tag{2}$$

The size or cardinality of the hyperparameter space is given by $|H|$.

2) Hyperparameter tuning procedure: Random Forest: The hyperparameter tuning procedure started with the following hyperparameter space using the notation in equations (1) and (2)

- The hyperparameter search procedure started with a cross-validated random halving search

$$\begin{aligned} H_{\text{initial}} &= \bigotimes_{i \in \text{Random forest hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} &= [10, 1000, 10], \\ P_{\text{maximum depth}} = [1, 30, 1], &P_{\text{maximum features}} = [1, 30, 1], \\ P_{\text{minimum samples split}} = [1, 30, 1], &P_{\text{minimum samples leaf}} = [1, 30, 1] \end{aligned} \tag{3}$$

  After the cross-validated random halving search, the following hyperparameters achieved the best performance.

$$\begin{gathered} n \text{ estimators} = 160, \text{maximum depth} = 4, \text{maximum features} = 2, \\ \text{minimum samples split} = 3, \text{minimum samples leaf} = 29 \end{gathered} \tag{4}$$

  After the cross-validated random halving search, the model performed with a mean ROC AUC of 0.84 with a standard error of 0.11

- First cross-validated grid search iteration - In the next cross-validated grid search iteration the model was tuned for the n estimator, maximum features, and maximum depth hyperparameters. The hyperparameter space consisted of:

$$\begin{aligned} H_{\text{first grid search}} &= \bigotimes_{i \in \text{Random forest hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} &= [140, 200, 10], \\ P_{\text{maximum depth}} = [1, 8, 1], &P_{\text{maximum features}} = [1, 5, 1], \\ P_{\text{minimum samples split}} = \{3\}, &P_{\text{minimum samples leaf}} = \{29\} \end{aligned} \tag{5}$$

  After the first iteration of the cross-validated grid search, the following hyperparameters achieved the best performance.

$$\begin{gathered} n \text{ estimators} = 150, \text{maximum depth} = 1, \text{maximum features} = 2, \\ \text{minimum samples split} = 3, \text{minimum samples leaf} = 29 \end{gathered} \tag{6}$$

  After the first iteration of the cross-validated grid search, the model performed with a mean ROC AUC of 0.82 with a standard error of 0.12

- Second iteration of cross-validated grid search - In the next cross-validated grid search iteration, the model was tuned for the minimum samples split and the minimum samples leaf. The hyperparameter space consisted of:

$$\begin{aligned} H_{\text{second grid search}} &= \bigotimes_{i \in \text{Random forest hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} &= \{150\}, \\ P_{\text{maximum depth}} = \{1\}, &P_{\text{maximum features}} = \{2\}, \\ P_{\text{minimum samples split}} = [2, 5, 1], &P_{\text{minimum samples leaf}} = [20, 30, 1] \end{aligned} \tag{7}$$

  After the second iteration of the cross-validated grid search, the following hyperparameters achieved the best performance.

$$\begin{gathered} n \text{ estimators} = 160, \text{maximum depth} = 4, \text{maximum features} = 2, \\ \text{minimum samples split} = 3, \text{minimum samples leaf} = 23 \end{gathered} \tag{8}$$

  After the second iteration of the cross-validated grid search, the model performed with a mean ROC AUC of 0.84 with a standard error of 0.11.

3) Hyperparameter Tuning Procedure for XGBoost: The hyperparameter tuning procedure started with the following hyperparameter space using the notation in equations (1) and (2).

- The hyperparameter search procedure started with a cross-validated random halving search.

$$\begin{aligned} H_{\text{initial}} &= \bigotimes_{i \in \text{XGBoost hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} &= [10, 500, 10], \\ P_{\text{maximum depth}} = [1, 10, 1], P_{\text{learning rate}} &= [0.01, 1, 0.01], \\ P_{\text{minimum child weight}} = [0.5, 1.5, 0.01], P_{\text{gamma}} &= [0, 0.5, 0.01], \\ P_{\text{column sample by tree}} = [0.5, 1, 0.01], P_{\text{subsample}} &= [0.5, 1, 0.01], \\ P_{\text{lambda}} = \{1\}, P_{\text{alpha}} = \{1\}, P_{\text{scale positive weight}} &= \{232.78\} \end{aligned} \tag{9}$$

After the cross-validated random halving search, the following hyperparameters achieved the best performance:

$$\begin{aligned} n \text{ estimators} = 100, \text{maximum depth} = 1, \\ \text{learning rate} = 0.34, \text{minimum child weight} = 0.86, \\ \text{gamma} = 0.36, \text{column sample by tree} = 0.93, \\ \text{subsample} = 0.6, \text{lambda} = 1, \\ \text{alpha} = 1, \text{scale positive weight} = 232.78 \end{aligned} \tag{10}$$

After the cross-validated random halving search, the model performed with a ROC AUC of 0.84 with a standard error of 0.11

- First cross iteration validated grid search - maximum

$$\begin{aligned} H_{\text{first grid search}} &= \bigotimes_{i \in \text{XGBoost hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} &= [50, 150, 10], \\ P_{\text{maximum depth}} = [1, 3, 1], P_{\text{learning rate}} &= [0.24, 0.34, 0.44], \\ P_{\text{minimum child weight}} = [0, 0.96, 0.01], P_{\text{gamma}} &= \{0.36\}, \\ P_{\text{column sample by tree}} = \{0.93\}, P_{\text{subsample}} &= \{0.6\}, \\ P_{\text{lambda}} = \{1\}, P_{\text{alpha}} = \{0\}, P_{\text{scale positive weight}} &= \{232.78\} \end{aligned} \tag{11}$$

After the first iteration of the cross-validated grid search, the following hyperparameters achieved the best performance.

$$\begin{aligned} n \text{ estimators} = 140, \text{maximum depth} = 1, \\ \text{learning rate} = 0.34, \text{minimum child weight} = 0, \\ \text{gamma} = 0.36, \text{column sample by tree} = 0.93, \\ \text{subsample} = 0.6, \text{lambda} = 1, \\ \text{alpha} = 0, \text{scale positive weight} = 232.78 \end{aligned} \tag{12}$$

After the first iteration of the cross-validated grid search, the model performed with a ROC AUC of 0.84 with a standard error of 0.09

- Second iteration of cross-validated grid search - In the next cross-validated grid search iteration, the model was tuned for the minimum samples split, and minimum samples leaf. The hyperparameter space consisted of

$$\begin{aligned} H_{\text{second grid search}} &= \bigotimes_{i \in \text{XGBoost hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} &= \{140\}, \\ P_{\text{maximum depth}} = \{1\}, P_{\text{learning rate}} &= \{0.34\}, \\ P_{\text{minimum child weight}} = \{0\}, P_{\text{gamma}} &= [0, 0.41, 0.01], \\ P_{\text{column sample by tree}} = [0.87, 0.98, 0.01], P_{\text{subsample}} &= [0.55, 0.65, 0.01], \\ P_{\text{lambda}} = \{1\}, P_{\text{alpha}} = \{0\}, P_{\text{scale positive weight}} &= \{232.78\} \end{aligned} \tag{13}$$

After the second iteration of the cross-validated grid search, the following hyperparameters achieved the best performance.

$$\begin{aligned} n \text{ estimators} = 140, &\text{maximum depth} = 1, \\ \text{learning rate} = 0.34, &\text{minimum child weight} = 0, \\ \text{gamma} = 0, &\text{column sample by tree} = 0.92, \\ \text{subsample} = 0.6, &\text{lambda} = 1, \\ \text{alpha} = 0, &\text{scale positive weight} = 232.78 \end{aligned} \tag{14}$$

After the second iteration of the cross-validated grid search, the model performed with a ROC AUC of 0.84 with a standard error of 0.09.

- Third iteration of cross-validated grid search - In the third cross-validated grid search iteration, the model was tuned for the alpha and lambda hyperparameters. The hyperparameter space consisted of

$$\begin{aligned} H_{\text{third grid search}} = &\bigotimes_{i \in \text{XGBoost hyperparameters}} P_i \\ \text{where} \quad P_{n \text{ estimators}} = \{140\}&, \\ P_{\text{maximum depth}} = \{1\}, P_{\text{learning rate}} = \{0.34\}&, \\ P_{\text{minimum child weight}} = \{0\}, P_{\text{gamma}} = \{0\}&, \\ P_{\text{column sample by tree}} = \{0.92\}, P_{\text{subsample}} = \{0.6\}&, \\ P_{\text{lambda}} = [0.8, 1.2, 0.01], P_{\text{alpha}} = [0, 0.2, 0.01]&, \\ P_{\text{scale positive weight}} = \{232.78\}& \end{aligned} \tag{15}$$

After the third iteration of the cross-validated grid search, the following hyperparameters achieved the best performance.

$$\begin{aligned} n \text{ estimators} = 140, &\text{maximum depth} = 1, \\ \text{learning rate} = 0.34, &\text{minimum child weight} = 0, \\ \text{gamma} = 0, &\text{column sample by tree} = 0.92, \\ \text{subsample} = 0.6, &\text{lambda} = 0.86, \\ \text{alpha} = 0.17, &\text{scale positive weight} = 232.78 \end{aligned} \tag{16}$$

After the third iteration of the cross-validated grid search, the model performed with a ROC AUC of 0.84 with a standard error of 0.09.

APPENDIX E - MODEL PERFORMANCE BEFORE AND AFTER HYPERPARAMETER TUNING

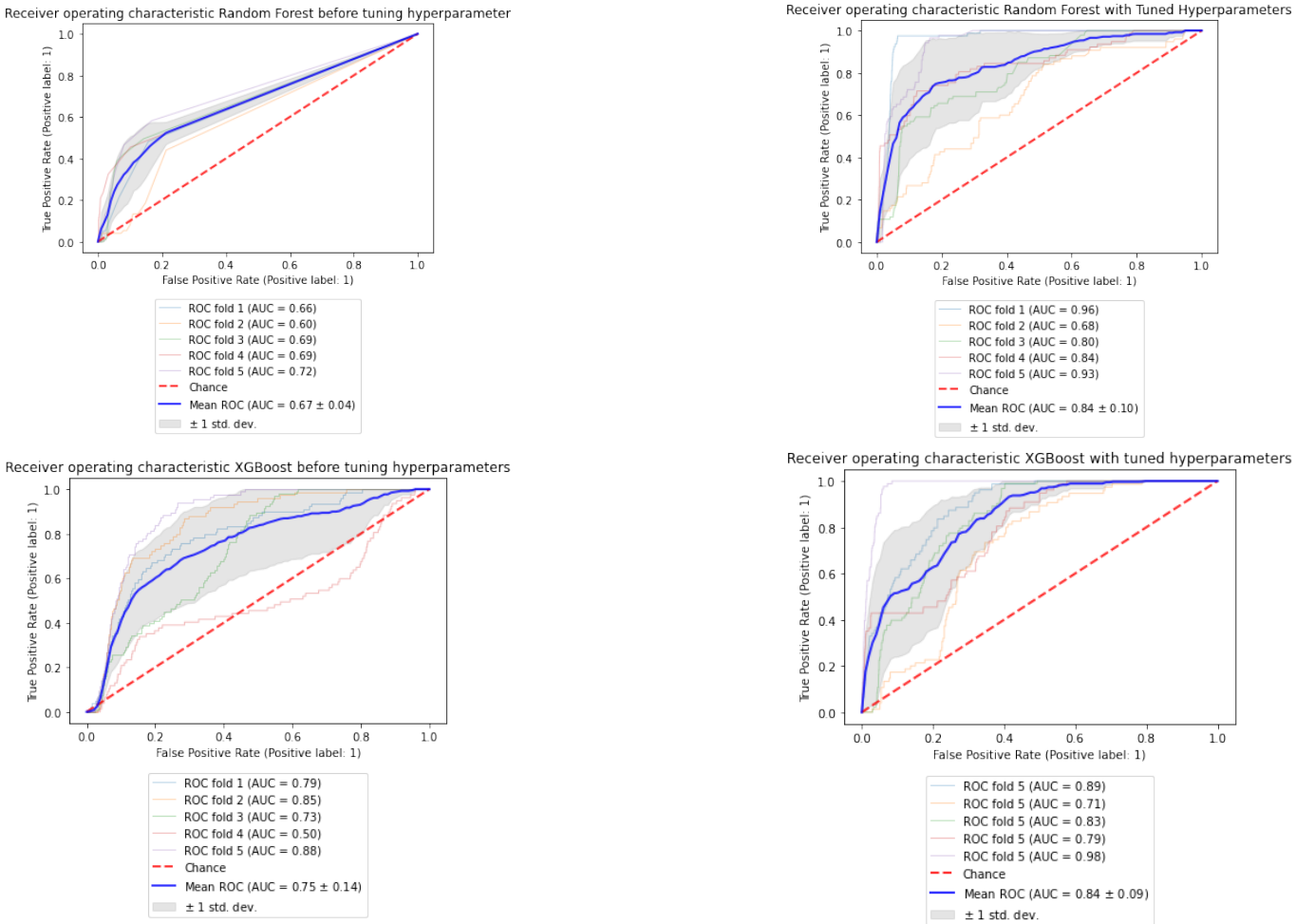

Fig. 1. The results of the AUC ROC for both Random Forests and XGBoost

APPENDIX F - FEATURE IMPORTANCE

.

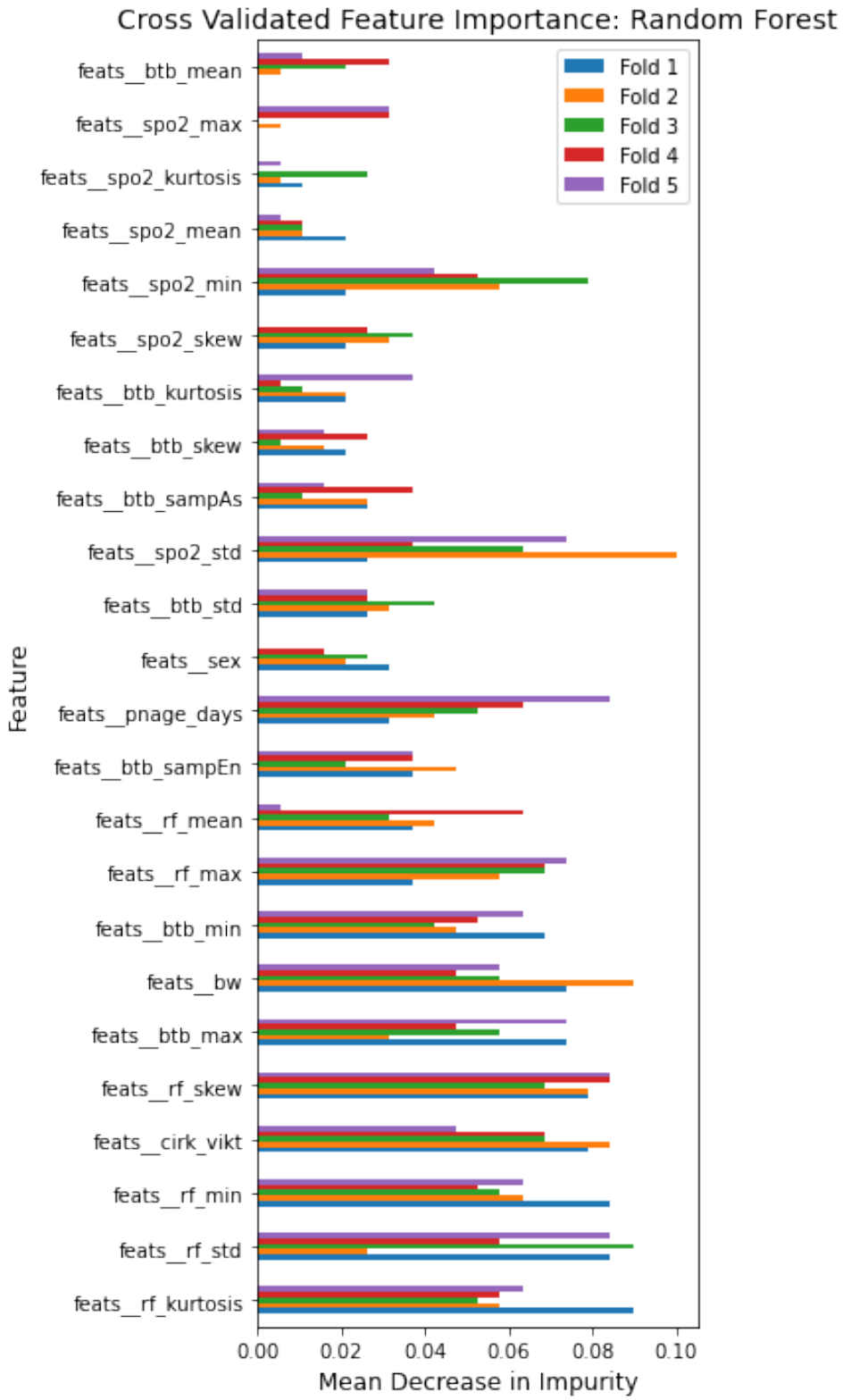

Fig. 2. Feature importance of Random Forest

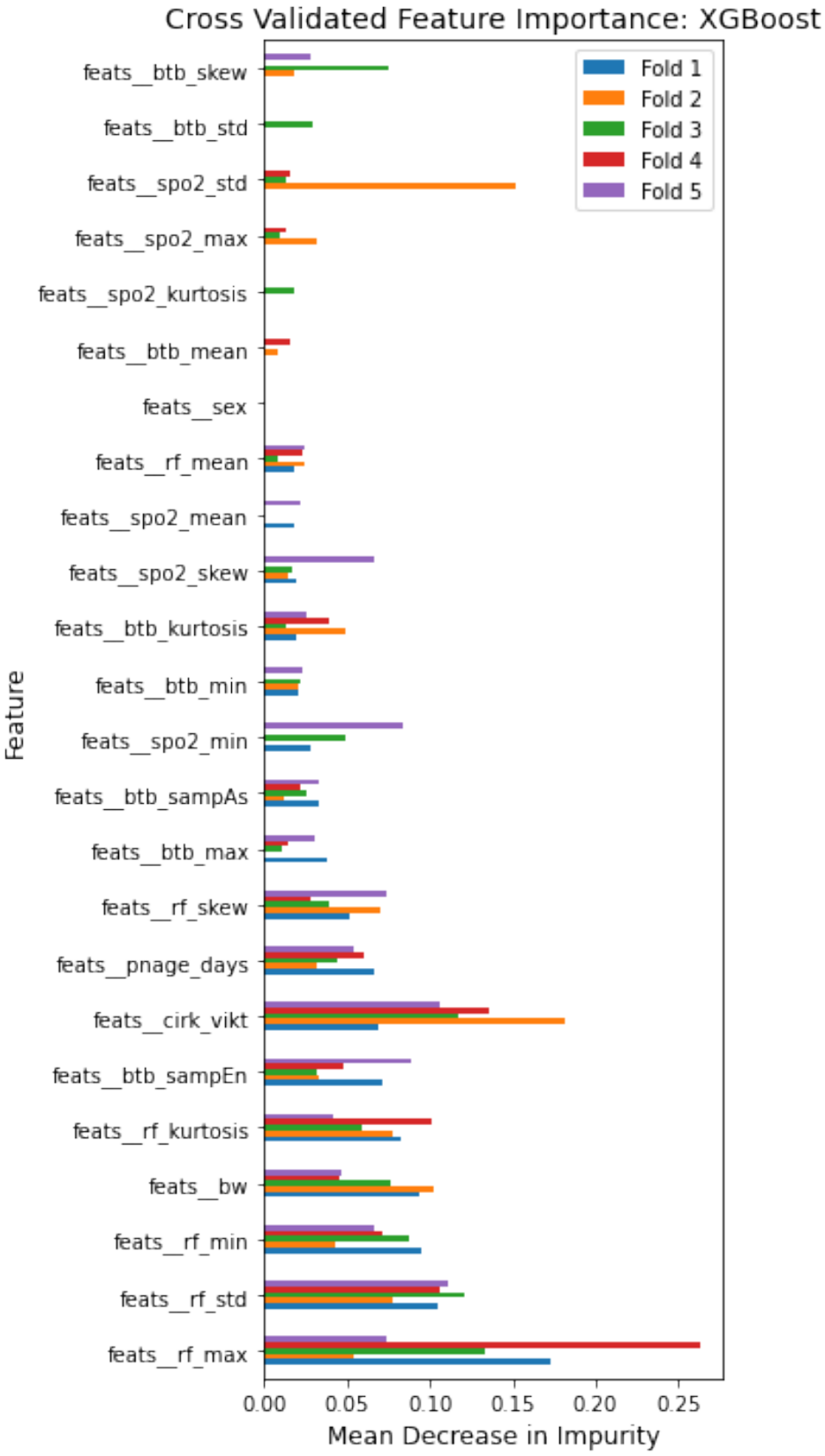

Fig. 3. Feature importance of XGBoost

APPENDIX G - ROC AUC SCORE DEPENDENCE ON HYPERPARAMTERS

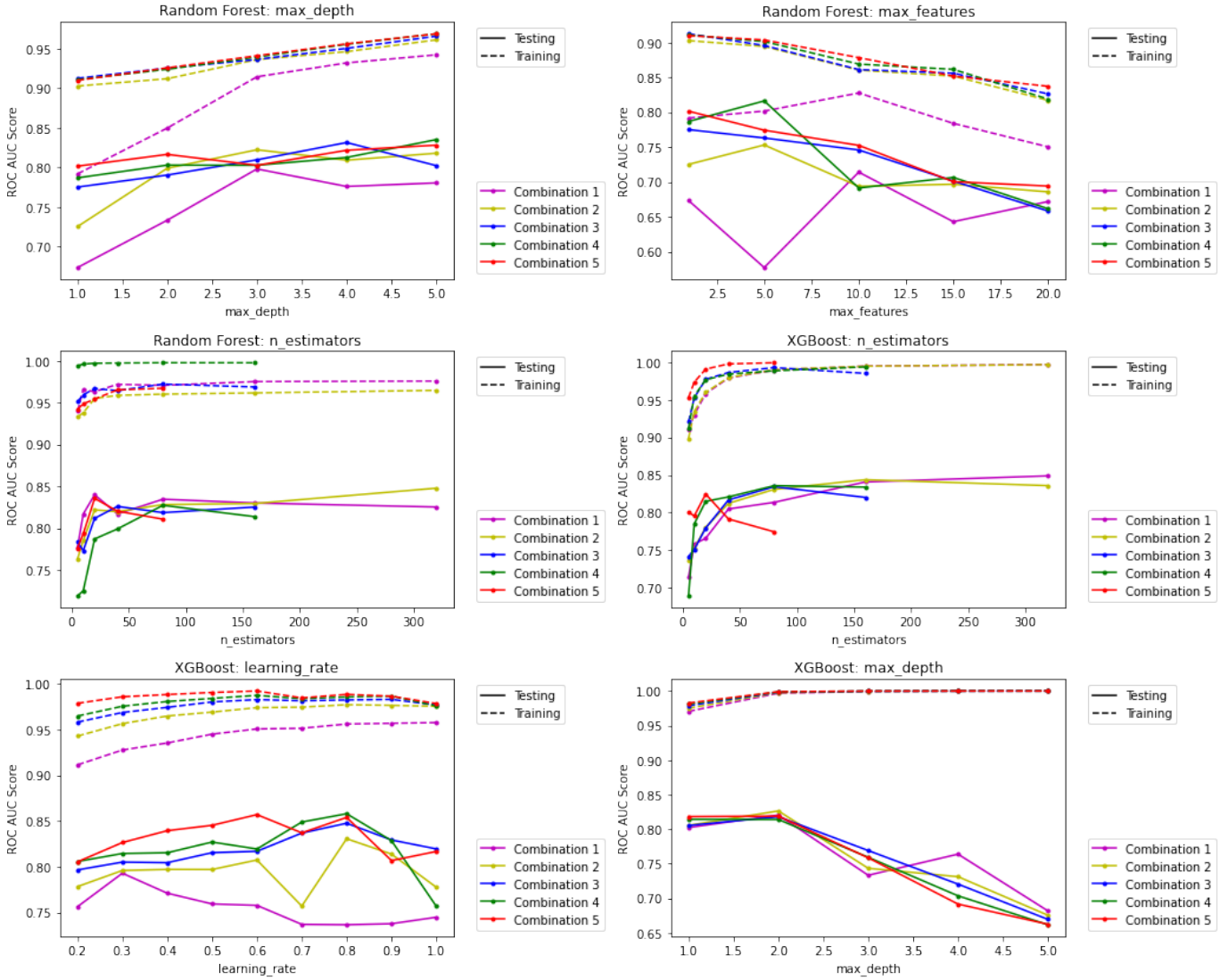

Fig. 4. Parameters impact the ROC AUC score for both Random Forest and XGBoost