Predicting the outcome of a PCR test for COVID-19 using a routine blood exam: A replication

An example of how AI can help identify possible infections in a primary care/triage system

By

Lukas Schießer

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Cognitive Science
to the
Institute of Cognitive Science at the Osnabrück University
January 16, 2021

Thesis Supervisor:
Johannes Schrumpf M. Sc., Institute of Cognitive Science, Osnabrück University
Thesis Supervisor:
Dr. Tobias Thelen, Institute of Cognitive Science, Osnabrück University

Abstract

Pandemics are stressful situation in which... Artificial Intelligence can help manage pandemic situations by managing ressource or identifying patients carrying the disease. This thesis reimplements several classifiers to predict a patient's PCR test result based on a routine blood test.

Acknowledgements

I want to thank...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Introduction for this thesis by explaining motivation for this topic and giving a short overview.
Motivation: overwhelmed health care systems, shortages of testing supplies, AI as supplementary technique to help.

Chapter 2

Materials and Methods

2.1 Data

The data used to train the classifiers was provided by Brinati et al. [5]. It was collected between the end of February 2020 and mid of March 2020 from patients admitted to the *IRCSS Ospedale San Raffaele* and consists of 279 individuals who were selected randomly. For each individual, the data set provides their age, gender, results of a routine blood screening, and a PCR test for Sars-CoV-2. A complete overview of the recorded variables is provided in A.1. The target variable *Swab* is binary and indicates the result of a PCR-test for Sars-CoV-2 taken by nasopharyngeal swab. A 0 indicates a negative test, while a 1 indicates a positive test. The data set is slightly imbalanced towards positive cases with 102 (37%) negative cases and 177 (63%) positive cases.

Since the variable *Gender* was provided as a string, it was transformed into two binary numerical variables called *female* and *male* by one-hot encoding. Further, two values of the variable *Age* were removed, specifically the values 0 and 1. This was sensible, seeing that there was no other data recorded from minors under the age of 18. Thus these two values can be presumed to be input errors during the collection process. Further, the variable *Lymphocytes* had to be recast using `pandas.to_numeric` because there was an input error for one of its values rendering the column's contents a string instead of a numerical value. This process created one missing value in place of the erroneous value.

Table A.4 provides standard statistics for the numerical features of the data set. All features are skewed. Blood values are skewed positively, exhibiting a left-leaning distribution because, for most blood values, a lower or centered value is significant for better health. Age is skewed negatively, exhibiting a right-leaning distribution

since COVID-19 affected older individuals more severely than younger individuals, especially in Italy.
As shown in the kernel density plots provided in Figure B.1, most of the data is non-normally distributed. In order to confirm this observation, a Shapiro-Wilk test for normality was performed. The test tests the null-hypothesis that the samples of a variable come from a normally distributed population. If the test's p-value is smaller than 0.05, a commonly chosen alpha level, the null-hypothesis is rejected, meaning the data is not normally distributed. The test results are provided in Table A.5 and show that all of the variables are non-normally distributed.
Table A.3 shows that most features have missing values. 196 samples have at least one feature missing, which amounts to 70 % of the data. Due to the small data set size, it is not feasible to exclude these individuals from the analysis process. It is rather more constructive to use an imputation method that models the missing values based on the data set's observed values. Therefore, Brinati et al. chose to use *Multivariate Imputation by Chained Equations*.

2.2 Multivariate Imputation by Chained Equations

Multivariate Imputation by Chained Equations or *MICE* for short is an imputation method proposed by Buuren and Groothuis-Oudshoorn [7]; it is also known as fully conditional specification (FCS). MICE is a method that imputes missing data by estimating a set of possible values from distributions of observed data. Each variable with missing data x_n is regressed on all other variables $x_1, ..., x_k$ which are restricted to the occurrences with observed data in x_n.
The imputation process is based on the following four main steps [14, 3]: Firstly, all missing values are imputed using a simple imputation method (e.g., mean imputation). These imputations can be thought of as "place holders" used during the first modeling phase. During step 2, the "place holder" imputations for one variable x are set back to missing. In step 3, all observed values from variable x in step 2 are regressed on the imputations model's other variables. Since this is the model building phase, this step only uses samples where x has observed values. Therefore, x is the dependent variable, and all other variables are independent variables used in the regression model. In step 4, the missing values in x are replaced with imputations (predictions) from the regression model built in step 3. All values of x, the observed and the imputed values, are then used in subsequent regression models of other variables.

Steps 2-4 are repeated for every variable with missing data. After the algorithm is done cycling through all variables, one iteration or "cycle" is completed. Steps 2-4 are repeated for a user-specified number of cycles. Generally, ten to twenty cycles should suffice to stabilize the imputation results, indicating the parameters controlling the imputations should have converged by then. This imputation process is usually repeated m times, creating m slightly differently imputed data sets, which are then used in the subsequent analysis. According to [6, 3, 14], already a small number of imputed data sets, usually three to ten, is sufficient to provide sensible results during analysis. It can, of course, be advantageous to use a higher number of imputed data sets to get a broader range of estimates. However, setting m higher requires more computations and storage and may not be worth preoccupying these extra resources [6].
MICE assumes that the data are missing at random (MAR). That is the probability of missing data does not depend on the unobserved data but is only dependent (conditional) on the observed data. Due to its individualistic approach, MICE can handle variables of different types using different modeling choices for different variable types.

2.3 Classifiers

The original paper [5] implements 7 classifiers, namely Random Forest, Logistic Regression, Decision Tree, k-nearest neighbors, naive Bayes, support vector machines, extremely randomized tree, and three-way Random Forest. Here, I will reproduce Random Forest, Logistic Regression, and Decision Tree. The following section will introduce them.

2.3.1 Decision Tree

To motivate the use of Random Forests and since we will use a Decision Tree during the validation to gain some insights into the workings of the Random Forest classifier, we will take a closer look at Decision Trees since they are the basic building block of Random Forests. A Decision Tree has a flowchart-like structure where each internal or decision node tests an attribute. Each branch corresponds to one attribute value, and each leaf node represents a classification. It is built up using the ID3 algorithm. ID3 tries to determine the best attribute of a given data set by the distribution of its values. The best attribute is then used as a root of the Decision Tree, and a branch

is created for every value this attribute can take, which also creates a subset of the data set that only has the attribute value of the branch. This process is repeated for every branch with the remaining subset of the training set until a leaf node is reached.

ID3 can use many different measures to decide which is the best attribute. Two of the most popular ones are information gain or the Gini Impurity.

Information gain uses the entropy measure to compute the impurity in an attribute. Entropy originates from information theory and describes the average information content of an attribute's possible outcomes, it is calculated, as shown in Equation 2.1. A high entropy represents a high average information content in an attribute. Information Gain measures the expected reduction in entropy of a set S caused by learning the state of a random variable A. It is calculated, as shown in Equation 2.2, and can be described as the difference between the entropy of a set S and the weighted average of the child entropies. This is computed for all remaining attributes, the attribute that maximizes the differences is then selected as a new node.

$$E(S) = \sum_{i=1}^{C} -p(i)\ log_2(p(i)) \tag{2.1}$$

$$Gain(S, A) = E(S) - \sum_{i}^{C} \frac{|S_i|}{S} E(S_i) \tag{2.2}$$

Gini Impurity or Gini index measures the impurity in a set S that is the probability of incorrect classification of a random sample. It is calculated as shown in Equation 2.3. The measure by which the algorithm bases its decision on which attribute to select as a node is called the Gini gain. It is calculated as shown in Equation 2.4 and can be described as the average Gini impurity. The attribute with the lowest value is selected for the Gini gain since we want to minimize the attributes' incorrect classification.

$$G = 1 - \sum_{i=1}^{C} p(i)^2 \tag{2.3}$$

$$Gini(S, A) = \sum_{i=1}^{C} \frac{|S_i|}{|S|} G(S_i) \tag{2.4}$$

ID3 is not able to process numerical attributes and missing values. It must therefore be extended to be able to deal with real-world data by the C4.5 algorithm.

2.3.2 Random Forest

The Random Forest Classifier is an ensemble classifier that uses multiple Decision Tree instances to classify the given data. It uses two methods to diversify the different classifications of the Decision Tree instances called tree bagging and feature bagging.[4]
During tree bagging, the Decision Tree is not trained on the whole training set but a subset of the training set. The subsets are generated by sampling the original data set with replacement. A certain percentage of samples is selected from the original data set and the remaining percentage are duplicates of the already selected samples. This process generates new bagged training sets S_i with the same size as the original training set S. If the size of the bagged set is smaller than the size of the original set, this process is called sub-bagging.[11]
Feature bagging or feature subset selection limits the number of features the individual Decision Trees can use at each new split in consideration. For each decision node consideration, several features are randomly selected and presented to the Decision Tree algorithm. A hyperparameter controls the number of features available to the algorithm. For classification problems, this is usually set to $\sqrt{p}$ where p is the number of features present in the complete data set.
Each Decision Tree in the forest generates a classification. A majority vote over all Decision Trees determines the final classification fo the Random Forest. Using tree and feature bagging, Random Forests average over all individual Decision Tree models, reducing the variance of classification and avoiding overfitting.[2]

2.3.3 Logistic regression

Logistic regression, especially binary Logistic Regression, is a modification of linear regression that models the probability $p(X)$ that a sample X has the label 1. To do so, it uses a special form of the sigmoid function called the logistic function:

$$p(\mathbf{X}) = \phi_{sig}(\mathbf{X}) = \frac{e^{\mathbf{X}\vec{\beta}}}{1 + e^{\mathbf{X}\vec{\beta}}} = \frac{1}{1 + e^{-\mathbf{X}\vec{\beta}}} \tag{2.5}$$

After some manipulation, we can see

$$\frac{p(\mathbf{X})}{1 - p(\mathbf{X})} = e^{\mathbf{X}\vec{\beta}} \tag{2.6}$$

The left-hand side of Equation 2.6 is called odds. It can take a value between 0 and ∞ indicating very low or very high probabilities of having the label 1. After

applying the logarithm to both sides of Equation 2.6, the log-odds or logit is obtained (Equation 2.7), which is linear in X.

$$ln\Big(\frac{p(\mathbf{X})}{1-p(\mathbf{X})}\Big) = \mathbf{X}\vec{\beta} \tag{2.7}$$

Since β_0 and all β_i of the vector $\vec{\beta}$ are unknown, they must be estimated on the available training data. The method that is used to achieve this is called maximum likelihood estimation (MLE). MLE tries to find estimates for the coefficients β such that the estimated probability $\hat{p}(x_i)$ of having label 1 for each individual is as close as possible to the individual's observed label. In other words, MLE tries to find estimates for the coefficients β of the model $P(\mathbf{X})$ described in Equation 2.5 such that the resulting probability is close to one if the individual's observed label is 1 and close to zero if the individual's label is 0. This can be formalized as a likelihood function as shown in Equation 2.8.

$$L(\vec{\beta}|\vec{y}, \mathbf{X}) = \prod_i p(x_i)^{y_i}(1-p(x_i))^{(1-y_i)} \tag{2.8}$$

The estimates for β are then chosen to maximize the likelihood function.[2]

2.4 Model training

The two models are trained using 5-fold nested cross validation. In each outer fold of the nested cross validation, the data is imputed using MICE and then split into training and test data. Since MICE usually generates several imputed data sets to account for the uncertainty in the imputation process, there are 5 different models for each classifier per fold. The cross validation's inner fold is used to determine the best hyperparameter combination for the classifiers using Grid Search, also called hyperparameter tuning. The parameters for Random Forest are as follows:

- Number of trees in the forest: [10, 100, 500]
- Number of features to consider looking for best split: [4, 8, 12, 16]
- Quality of split measure: Gini impurity or Information gain

For Logistic Regression, Grid Search searches for an optimum among the following hyperparameters:

- Inverse of regularization strength C: [0.001, 0.01, 0.1, 1, 10, 100, 1000],
- Penalty: L1 or L2 regularization
- Algorithm to use during optimization problem: liblinear or saga
- Maximal iteration: [200, 300, 400, 500]

The remaining hyperparameters use the default values specified in the documentation of sklearn for Random Forests and Logistic Regression. Grid Search selects the best hyperparameters according to the highest accuracy value.
After the training, the models are evaluated using accuracy, balanced accuracy, sensitivity, specificity, positive predictive value, and area under the ROC curve. The hyperparameters of the best performing models for Random Forest and Logistic Regression are then used to obtain the final classifiers. For the validation, the whole data set is split into training and validation data according to an 80:20 split. The data set is imputed before the split due to the working of the `mice()` package in R. The classifiers are trained on the training data and then validated using the validation data. In addition to the evaluation metrics above, the final classifiers were also evaluated using the ROC curve and the precision-recall curve. Additionally, the precision-recall curve is used to assess the classifiers' performance regarding the slight imbalance in the data set. To further gain some insights into the Random Forest classification reasoning a Decision Tree is trained using the standard hyperparameters according to the sklearn implementation.
To implement this procedure, the Python libraries scikit-learn, pandas, and numpy are used. To implement the imputation method, the R library `mice()` is used and connected to the Python implementation using a bridge package called rpy2. The R implementation of the imputation method is used instead of a Python implementation since it provides more functions to inspect the MICE procedure and since it is able to handle non-normally distributed data. To visualize the results, the libraries seaborn and matplotlib are used.

2.5 Evaluation Metrics

The following terms will be used in the equations to describe the evaluation metrics: True Positive, False Positive, False Negative, and True Negative. They are further characterized in Table 2.1.
Accuracy describes the closeness of an individual's predicted label to its true label. It is calculated as shown in Equation 2.9. Accuracy alone is not a sensible measure

		True condition	
		Condition positive	Condition negative
Predicted condition	Condition predicted positive	True Positive (TP)	False Positive (FP)
	Condition predicted negative	False Negative (FN)	True Negative (TN)

Table 2.1: Confusion matrix for binary classification

to determine the goodness of fit for a model since the model can be affected by imbalanced data or poor parameter initialization. Accordingly, other measures should be considered during model selection.[10]

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.9}$$

Balanced accuracy is used in addition to the average accuracy due to the slight imbalance in the data set. If the balanced accuracy differs significantly from the average accuracy, the imbalance affects the classifier regarding class prevalence. The balanced accuracy is defined as the average of sensitivity and specificity. [10, 5]

$$balanced\ accuracy = \frac{\frac{TP}{TP+FN}\frac{TN}{TN+FP}}{2} \tag{2.10}$$

Sensitivity is interpreted as the proportion of people infected with a disease who will test positive for this disease. It is calculated as shown in Equation 2.11. Since it is only calculated using the part of the population with the disease, sensitivity can only give evidence about the true positive rate and not the test's false positive rate.[1]

$$sensitivity = \frac{TP}{TP + FN} \tag{2.11}$$

Specificity is defined as the proportion of people without the disease that are identified correctly, i.e., who tested negative for the disease. It is calculated as shown in Equation 2.12. Specificity only gives information about the proportion of people without the disease and a negative test and cannot disclose anything about the false negative rate.[1]

$$specificity = \frac{TN}{TN + FP} \tag{2.12}$$

The positive predictive value (PPV) measures the post-test probability of the disease given a positive test. It, therefore, refers to the probability that a subject with a

positive test has the disease indeed. It is calculated as shown in Equation 2.13.[1]

$$PPV = \frac{TP}{TP + FP} \tag{2.13}$$

The receiver operating characteristic curve (ROC curve) is a commonly used metric to determine a model's performance describing the trade-off between specificity and sensitivity. It is calculated using the prediction scores of a model that are either discriminant values or posterior probabilities.[13] Most models do not produce a classification label as their output but rather a prediction score that is then thresholded to correspond to one of the target variable levels. The metrics mentioned above are all so called single-thresholded metrics since they are defined only for the classifier's thresholded output.[13, 10] Since ROC uses the prediction scores instead of the thresholded output of a model, it can compute the sensitivity and specificity for every sensible threshold. The resulting values are then plotted with sensitivity or true positive rate as the vertical axis and the false positive rate or 1 - specificity as the horizontal axis. The points are interpolated linearly to create the ROC curve. [13] Usually, a reference line is added to the plot illustrating a model's performance that only makes random predictions. The curve for the trained model is expected to be above the reference line at all times. A model with perfect performance will appear in the top left-hand corner of the ROC space where the sensitivity is one, and the false positive rate is zero.[10, 9]
The area under the ROC curve can also be measured and is called either AUC for area under the ROC curve or ROC index. It can give a numeric summary of the ROC curve. It is calculated using the ROC curve's integral which can be easily done using the trapezoidal method since the ROC curve is discrete and stepped. A large AUC value indicates a better model performance. As a rule of thumb, an AUC over 0.7 indicates a strong predictive model, and an AUC below 0.6 indicates a weak model. AUC and ROC both are quite robust against imbalanced data, but other non-single-threshold metrics can be used in this case as well.[10, 9]

The Precision-Recall curve (PRC) is another non-single-threshold metric that is especially useful when dealing with an imbalanced data set. It calculates the precision and recall values for each feasible threshold. The resulting values are then plotted with precision, also called positive predictive value, as the vertical axis and recall, also called sensitivity, as the horizontal axis. The points are interpolated non-linearly to create a curve. In contrast to the ROC curve, which uses a fixed reference line, the PRC's baseline is determined by the ratio of positive (P) and negative (N) samples in the data set and represented as $y = \frac{P}{P+N}$. Each point in the PRC plot corresponds to exactly one point in the ROC plot since one of the

measures in the plot is the same. The area under the curve can also be calculated for the precision-recall curve similar to the AUC of the ROC curve and is then denoted by AUC (PRC) or average precision. Similarly, a large value for the AUC (PRC) indicates a strong model, and a lower value indicates a weak model.[8, 13]

Chapter 3

Results

3.1 Results of own implementation

Table A.6 and A.7 show the mean and standard deviation for the accuracy and balanced accuracy of the models during nested cross-validation. The authors of the original paper provided a 95% confidence interval for these two measures. Since there is no information on how they computed this interval, it is assumed that they used a two-sided Student's t-distribution which is used to estimate the mean of normally distributed populations where sample size is small. It is calculated using the following formula $\bar{x} \pm \frac{t_{\alpha,v}s}{\sqrt{n}}$ where $t_{\alpha,v}$ is the t-value which can be taken from a table with α as the degrees of freedom $n-1$ and v as the confidence level, s is the sample variance, $\bar{x}$ is the sample mean, and n is the number of samples. For the Random Forest, the 95% confidence interval of the average accuracy is [0.76, 0.80] and of the average balanced accuracy is [0.73, 0.77]. For Logistic Regression, the confidence interval of the average accuracy is [0.72, 0.77] and of the average balanced accuracy is [0.68, 0,72]. Since the confidence intervals of the accuracy and balanced accuracy for both classifiers overlap or are near each other, it can be assumed that the slight imbalance of the data set had no significant effect on the classifier and that the classifier therefore is not biased against the outcomes of the target variable. All evaluation metrics for each fold and imputed data set are provided in Table A.9.

The best models where selected by searching for the model with the most numbers of highest evaluation measures. The best Random Forest Classifier used maximally 4 features, 500 estimators and Gini impurity, it achieved an accuracy of 87.5% and a sensitivity of 90%. The best Logistic Regression Classifier used a C of 10, a maximal iteration of 200, the 'l2' penalty and the 'liblinear' solver, it achieved an accuracy

of 83.9% and a sensitivity of 85%. These settings were then used to retrain the classifiers and compute the validation metrics.
The Random Forest Classifier is also the best performing classifier after retraining it with the best hyperparameter with an accuracy of 76.8% and a balanced accuracy of 73.7%. The Logistic Regression achieves an accuracy of 73.2% and a balanced accuracy of 69.4%. Accordingly, there is also no significant difference between the accuracy and balanced accuracy for all three classifiers, meaning no classifier is biased. The aforementioned metrics are lower than the evaluation metrics since the data set used to train during the validation phase is different from the one used during the evaluation phase due to the way MICE imputes missing data. All single-threshold metrics are provided in Table A.8. The two multi-threshold metrics, ROC and PRC curve, are shown in Figure B.3. Both plots show that the classifiers perform better than the reference line. The values for the area under the ROC and under the PRC curve suggest that Logistic Regression (AUC: 80.8%, AUC (PRC): 85.5%) performs better than the Random Forest (AUC: 80.5%, AUC (PRC): 82.9%) although all other metrics indicate a better performance for the latter. It should be noted that the difference between the AUC scores for both curves are minimal and are not meaningful enough to declare the Logistic Regression as the better classifier. The more apparent differences in the AUC (PRC) for both classifiers on the other hand could indicate that the Logistic Regression is less affected by the imbalance in the data set than the Random Forest is.
The Decision Tree Classifier exhibits a lower discriminative performance especially when looking at the AUC and AUC (PRC) with 68.5% and 69.8% indicating a rather weak classifier. Additionally, it only achieves an accuracy of 71.4% and a balanced accuracy of 68.5%. Despite these low metrics, the Decision Tree can still be used as an aid to help clinicians make quick decisions and to interpret the result of the Random Forest.

Figure B.4 shows all feature importance plots for the three classifiers. The feature importance for Decision Tree and Random Forest were obtained using the sklearn implementation. The implementation calculates the feature importance by quantifying the mean decrease in impurity. It should be noted that impurity-based feature importance can suffer from favoring features with a high number of unique values as well as making predictions based on statistics derived from the training set. This method is therefore not necessarily informative about whether a feature makes for a good prediction or not.[12] For Logistic Regression, the feature importance is inferred by looking at the coefficients of each feature in the data set. A positive coefficient can be interpreted as contributing to a positive (1) result and a negative coefficient can be interpreted as contributing to a negative (0) result. It is difficult

to compare non tree-based feature importance with tree-based importance because they have different scales and means of calculation.
Both tree-based methods show a similar feature importance plot with AST being the most important feature. The remaining features are differently distributed, but mostly exhibit similar values. Consequently, the Decision Tree and its visualization (Figure B.5) in combination with the feature importance plots for both tree-based classifiers can be used to give some rough insights into the classification procedure of the Random Forest.

3.2 Comparison with original paper

The 95% confidence intervals for the accuracy and balanced accuracy of this replication overlap with the ones from the original paper. The average accuracy for Random Forest in the original is [0.74, 0.80] (replication: [0.76, 0.80]) and [0.70, 0.81] (replication: [0.72, 0.77]) for Logistic Regression. The average balanced accuracy in the original is [0.70, 0.82] (replication: [0.73, 0.77]) for Random Forest and [0.65, 0.74] (replication: [0.68, 0.72]) for Logistic Regression. Since there is no information about individual values during the nested cross validation and it is also not reported how the authors computed the 95% confidence intervals, it can be difficult to compare them. Nevertheless, the overlap of the intervals suggests that the classifiers can be compared to a certain degree. Unfortunately, there is no more information about further evaluation metrics.
The validation metrics of the retrained classifiers show a lower performance than the classifiers in the original paper. The differences between the Random Forest and Logistic Regression are however similar to the ones in the original paper. In this replication, the single-threshold metrics are higher for the Random Forest and the multi-threshold metrics are higher for the Logistic Regression reflecting a behavior similar to that in the original paper. The ROC curve for both classifiers exhibits a similar course as in the original except that the curves in the original rise higher in the beginning due to their higher discriminative power. Seeing that the precision-recall curve only plots the performance of the Random Forest, we can only compare these results. The replication and the original differ greatly in this plot, but still exhibiting a similar course at some points, e.g., the sharp drop between sensitivity = [0.0, 0.2].

Chapter 4

Discussion

4.1 Discussion of Results

Plausibility of models in terms of recorded metrics, i.e., sensitivity, specificity, ... and what that means for their predictive "statements"
Accordance of feature importance and scientific research (do most important features in model coincide with blood values identified as being evidence for a COVID-19 illness)

4.2 Discussion of Methods

Shortly mention that R implementation is not able to apply MICE model to other data only to data it is "trained"/ "fitted" on (or maybe that's for the 4 Discussion section)
discuss imputation process further and its connection to cross-validation, keyword: data leakage
discuss the original paper in light of replication and open science, e.g., lack of information about parameters/settings, missing information regarding results, ...

4.3 Conclusion

Declaration

I declare that..

Appendix A

Supplementary Tables

Feature	Data Type
Gender	Categorical
Age	Numerical (discrete)
WBC (White blood cell count)	Numerical (continuous)
Platelets	Numerical (continuous)
Neutrophils	Numerical (continuous)
Lymphocytes	Numerical (continuous)
Monocytes	Numerical (continuous)
Eosinophils	Numerical (continuous)
Basophils	Numerical (continuous)
CRP (C-reactive protein)	Numerical (continuous)
AST (aspartate aminotransferase)	Numerical (continuous)
ALT (alanine aminotransferase)	Numerical (continuous)
ALP (alkaline phosphatase)	Numerical (continuous)
GGT (gamma glutamyl transferase)	Numerical (continuous)
LDH (lactate dehydrogenase)	Numerical (continuous)
SWAB	Categorical

Table A.1: Overview over all features of the data set

Feature	Unit	Mean	Std	Median
Age	Years	61.33	18.05	64
White Blood Cell Count (WBC)	10^9/L	8.49	4.89	7.10
Platelets	10^9/L	224.91	102.61	204.00
Neutrophils	10^9/L	4.64	4.50	3.90
Lymphocytes	10^9/L	0.88	0.87	0.80
Monocytes	10^9/L	0.45	0.44	0.40
Eosinophils	10^9/L	0.04	0.12	0.00
Basophils	10^9/L	0.01	0.03	0.00
C-reactive protein (CRP)	mg/L	88.93	94.32	53.10
Aspartate Aminotransferase (AST)	U/L	53.81	57.59	36.00
Alanine Aminotransferase (ALT)	U/L	42.82	45.43	30.00
Alkaline Phosphatase (ALP)	U/L	42.21	75.71	68.00
Gamma Glutamyl Transferase (GGT)	U/L	40.20	101.29	0.00
Lactate dehydrogenase (LDH)	U/L	264.54	238.53	254.00

Table A.2: Descriptive statistics for numerical features in data set (including missing values as in [5])

Feature	Number of NaN (in %)
Gender	0 (0 %)
Age	2 (0.72 %)
WBC (White blood cell count)	2 (0.72 %)
Platelets	2 (0.72 %)
Neutrophils	70 (25.09 %)
Lymphocytes	71 (25.45 %)
Monocytes	70 (25.09 %)
Eosinophils	70 (25.09 %)
Basophils	71 (25.45 %)
CRP (C-reactive protein)	6 (2.15 %)
AST (aspartate aminotransferase)	2 (0.72 %)
ALT (alanine aminotransferase)	13 (4.66 %)
ALP (alkaline phosphatase)	148 (53.05 %)
GGT (gamma glutamyl transferase)	143 (51.25 %)
LDH (lactate dehydrogenase)	85 (30.47 %)
SWAB	0 (0 %)

Table A.3: Number of missing values and their proportion the total number of data points

Feature	Unit	Mean	Std	Median
Age	Years	61.78	17.81	64
White Blood Cell Count (WBC)	10^9/L	8.55	4.86	7.10
Platelets	10^9/L	226.5	101.2	205.00
Neutrophils	10^9/L	6.20	4.17	5.10
Lymphocytes	10^9/L	1.19	0.80	1.00
Monocytes	10^9/L	0.61	0.41	0.50
Eosinophils	10^9/L	0.06	0.13	0.00
Basophils	10^9/L	0.01	0.04	0.00
C-reactive protein (CRP)	mg/L	90.89	94.42	54.20
Aspartate Aminotransferase (AST)	U/L	54.20	57.61	36.00
Alanine Aminotransferase (ALT)	U/L	44.92	45.50	31.00
Alkaline Phosphatase (ALP)	U/L	89.89	89.09	71.00
Gamma Glutamyl Transferase (GGT)	U/L	82.48	132.70	41.00
Lactate dehydrogenase (LDH)	U/L	380.45	193.98	328.00

Table A.4: Descriptive statistics for numerical features in data set (excluding missing values)

Feature	Test statistic	p-value
Age	0.976	0.000
WBC (White blood cell count)	0.873	0.000
Platelets	0.930	0.000
Neutrophils	0.838	0.000
Lymphocytes	0.785	0.000
Monocytes	0.811	0.000
Eosinophils	0.457	0.000
Basophils	0.395	0.000
CRP (C-reactive protein)	0.836	0.000
AST (aspartate aminotransferase)	0.556	0.000
ALT (alanine aminotransferase)	0.629	0.000
ALP (alkaline phosphatase)	0.420	0.000
GGT (gamma glutamyl transferase)	0.500	0.000
LDH (lactate dehydrogenase)	0.877	0.000

Table A.5: Results of Shapiro-Wilk test for normality for numerical features of the data set

	Fold1		Fold2		Fold3		Fold4		Fold5		Combined	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
RF	0.72	0.01	0.84	0.03	0.81	0.03	0.75	0.03	0.77	0.03	0.78	0.05
LR	0.71	0.03	0.81	0.02	0.74	0.03	0.70	0.06	0.73	0.04	0.74	0.05

Table A.6: Mean training accuracies and their standard deviations (std) for each training fold and all training folds combined. Random Forest = RF, Logisitic Regression = LR

	Fold1		Fold2		Fold3		Fold4		Fold5		Combined	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
RF	0.69	0.01	0.81	0.04	0.79	0.03	0.73	0.03	0.71	0.04	0.75	0.05
LR	0.67	0.02	0.78	0.03	0.72	0.03	0.68	0.06	0.67	0.04	0.70	0.06

Table A.7: Mean training balanced accuracies and their standard deviations (std) for each training fold and all training folds combined. Random Forest = RF, Logisitic Regression = LR

Model	acc	b_acc	sensitivity	specificity	ppv	auc	a_precision
RF	0.768	0.737	0.909	0.565	0.750	0.802	0.829
LR	0.732	0.694	0.909	0.478	0.714	0.808	0.855
DT	0.714	0.685	0.848	0.522	0.718	0.685	0.698

Table A.8: Validation metrics for all classifiers; RF = Random Forest, DT = Decision Tree, LR = Logisitic Regression

Fold	Imputed Set	Model	Model Count	acc	b_acc	sensitivity	specificity	ppv	auc
0	1	lr	0	0.750	0.709	0.939	0.478	0.721	0.816
0	2	lr	1	0.714	0.679	0.879	0.478	0.707	0.837
0	3	lr	2	0.696	0.663	0.848	0.478	0.700	0.806
0	4	lr	3	0.696	0.663	0.848	0.478	0.700	0.797
0	5	lr	4	0.679	0.642	0.848	0.435	0.683	0.826
1	1	lr	5	0.804	0.787	0.825	0.750	0.892	0.845
1	2	lr	6	0.839	0.831	0.850	0.812	0.919	0.872
1	3	lr	7	0.804	0.787	0.825	0.750	0.892	0.856
1	4	lr	8	0.786	0.738	0.850	0.625	0.850	0.822
1	5	lr	9	0.804	0.769	0.850	0.688	0.872	0.833
2	1	lr	10	0.732	0.713	0.818	0.609	0.750	0.742
2	2	lr	11	0.768	0.750	0.848	0.652	0.778	0.764
2	3	lr	12	0.768	0.757	0.818	0.696	0.794	0.823
2	4	lr	13	0.732	0.720	0.788	0.652	0.765	0.773
2	5	lr	14	0.696	0.683	0.758	0.609	0.735	0.758
3	1	lr	15	0.714	0.695	0.771	0.619	0.771	0.774
3	2	lr	16	0.750	0.733	0.800	0.667	0.800	0.788
3	3	lr	17	0.643	0.619	0.714	0.524	0.714	0.686
3	4	lr	18	0.750	0.733	0.800	0.667	0.800	0.777
3	5	lr	19	0.625	0.605	0.686	0.524	0.706	0.680
4	1	lr	20	0.800	0.723	0.972	0.474	0.778	0.801
4	2	lr	21	0.727	0.680	0.833	0.526	0.769	0.781
4	3	lr	22	0.709	0.629	0.889	0.368	0.727	0.773
4	4	lr	23	0.691	0.627	0.833	0.421	0.732	0.754
4	5	lr	24	0.727	0.680	0.833	0.526	0.769	0.785
0	1	rf	0	0.732	0.694	0.909	0.478	0.714	0.792
0	2	rf	1	0.714	0.685	0.848	0.522	0.718	0.801
0	3	rf	2	0.732	0.694	0.909	0.478	0.714	0.782
0	4	rf	3	0.732	0.700	0.879	0.522	0.725	0.739
0	5	rf	4	0.714	0.692	0.818	0.565	0.730	0.760
1	1	rf	5	0.875	0.856	0.900	0.812	0.923	0.848
1	2	rf	6	0.821	0.800	0.850	0.750	0.895	0.842
1	3	rf	7	0.839	0.812	0.875	0.750	0.897	0.850
1	4	rf	8	0.839	0.812	0.875	0.750	0.897	0.809
1	5	rf	9	0.804	0.750	0.875	0.625	0.854	0.806
2	1	rf	10	0.804	0.787	0.879	0.696	0.806	0.825
2	2	rf	11	0.857	0.839	0.939	0.739	0.838	0.839
2	3	rf	12	0.786	0.772	0.848	0.696	0.800	0.816
2	4	rf	13	0.804	0.781	0.909	0.652	0.789	0.847
2	5	rf	14	0.804	0.787	0.879	0.696	0.806	0.831
3	1	rf	15	0.750	0.714	0.857	0.571	0.769	0.826
3	2	rf	16	0.696	0.700	0.686	0.714	0.800	0.783
3	3	rf	17	0.768	0.738	0.857	0.619	0.789	0.848
3	4	rf	18	0.786	0.762	0.857	0.667	0.811	0.812
3	5	rf	19	0.768	0.729	0.886	0.571	0.775	0.793
4	1	rf	20	0.782	0.734	0.889	0.579	0.800	0.836
4	2	rf	21	0.764	0.695	0.917	0.474	0.767	0.798
4	3	rf	22	0.727	0.655	0.889	0.421	0.744	0.848
4	4	rf	23	0.764	0.708	0.889	0.526	0.780	0.798
4	5	rf	24	0.800	0.760	0.889	0.632	0.821	0.777

Table A.9: Evaluation metrics for Random Forest (rf) and Logisitc Regression (lr) during the nested cross validation

	AGE		WBC		Platelets		Neutrophils		Lymphocytes		Monocytes		Eosinophils		Basophils		CRP		AST		ALT		ALP		GGT		LDH	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
Dataset																												
original	61.776	17.816	8.553	4.855	226.532	101.174	6.200	4.173	1.187	0.806	0.606	0.410	0.055	0.132	0.014	0.039	90.889	94.421	54.202	57.613	44.917	45.503	89.893	89.090	82.478	132.703	380.448	193.984
0.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
1.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
2.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
3.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
4.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
5.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
6.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
7.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
8.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
9.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
10.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
11.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
12.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
13.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
14.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
15.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
16.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
17.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
18.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
19.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
20.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
21.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
22.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
23.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
24.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
25.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
26.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
27.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
28.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
29.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
30.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
31.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
32.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
33.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
34.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
35.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
36.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
37.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
38.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
39.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
40.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
41.0	61.839	17.773	8.574	4.863	226.654	100.820	6.597	4.482	1.260	0.899	0.649	0.455	0.066	0.147	0.017	0.041	92.581	96.875	54.208	57.458	47.459	49.868	82.362	66.091	79.681	119.072	379.516	199.693
42.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
43.0	61.832	17.768	8.557	4.840	225.951	101.067	6.597	4.301	1.236	0.807	0.655	0.458	0.052	0.140	0.015	0.039	91.247	94.973	54.269	57.422	47.405	50.648	81.803	65.194	73.470	103.199	365.706	178.115
44.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
45.0	61.871	17.792	8.600	4.870	226.288	100.896	6.650	4.439	1.220	0.801	0.659	0.473	0.054	0.123	0.014	0.038	91.278	94.059	54.161	57.417	46.774	48.087	83.484	67.061	75.283	106.761	367.760	191.073
46.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
47.0	61.728	17.768	8.557	4.839	226.571	100.911	6.625	4.353	1.229	0.807	0.643	0.429	0.054	0.124	0.016	0.043	91.447	94.246	54.029	57.452	46.864	48.285	83.065	66.928	75.384	113.305	374.771	195.309
48.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479
49.0	61.832	17.806	8.534	4.843	226.446	101.323	6.592	4.412	1.209	0.799	0.642	0.446	0.055	0.126	0.015	0.039	91.255	94.812	54.004	57.453	45.943	45.804	83.964	67.706	76.444	117.722	374.405	196.479

Table A.10: Numerical summaries for all variables of the original not-imputed data set (denoted by original) and the imputed data sets (denoted by a number)

Appendix B

Supplementary Figures

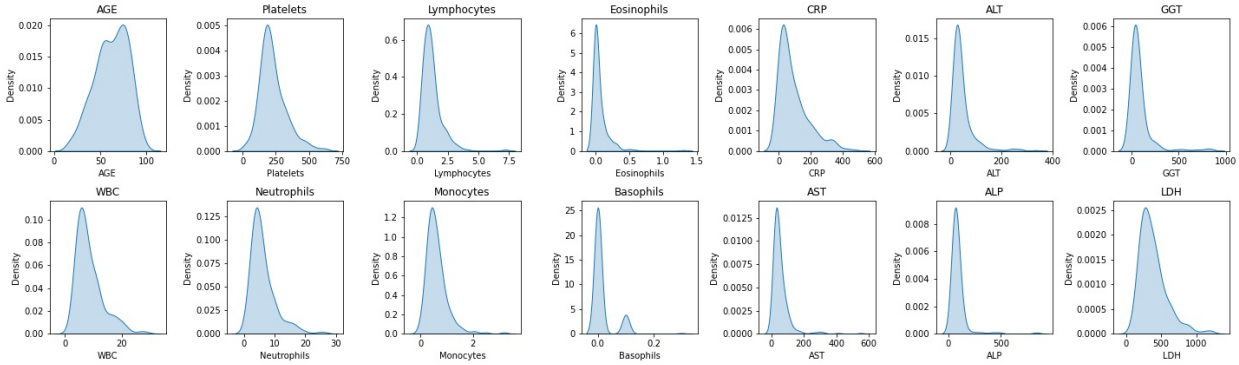
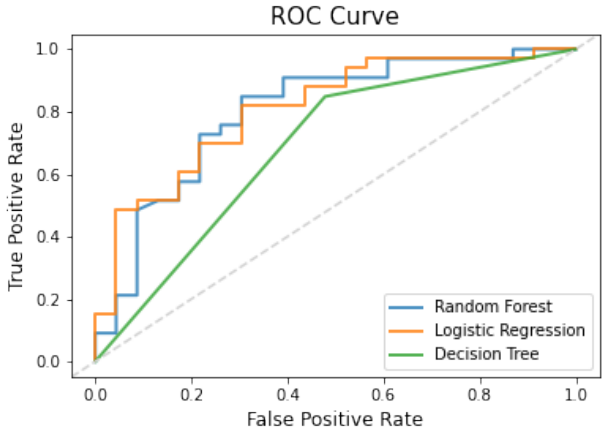


Figure B.1: Kernel density plots of the numerical features of the data set

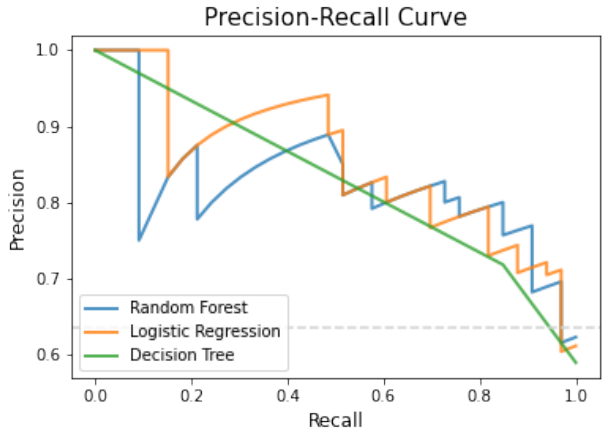
Nested k-fold Cross-Validation

1. Define set of hyper-parameter combinations, ***C***, for current model. If model has no hyper-parameters, ***C*** is the empty set.
2. Divide data into *K* folds with approximately equal distribution of cases and controls
3. (**outer loop**) For fold k_i in the *K* folds:
    1. Set fold k_i as the test set
    2. Perform automated feature selection on the remaining *K-1* folds
    3. For parameter combination c in ***C***:
        1. (**inner loop**) For fold k_j in the remaining *K-1* folds:
            1. Set fold k_j as the validation set
            2. Train model on remaining *K-2* folds
            3. Evaluate model performance on fold k_j
        2. Calculate average performance over *K-2* folds for parameter combination *c*
    4. Train model on *K-1* folds using hyper-parameter combination that yielded best average performance over all steps of the **inner loop**
    5. Evaluate model performance on fold k_i.
4. Calculate average performance over *K* folds

Figure B.2: K-fold nested cross validation

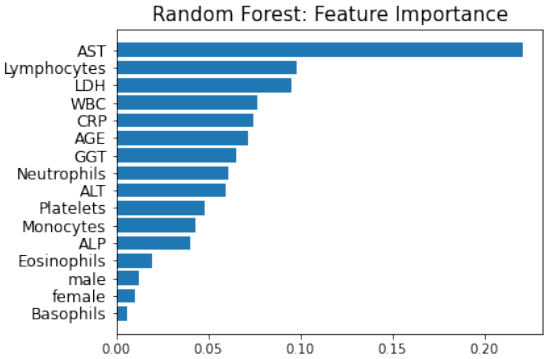


(a) Receiver Operating Characterisitcs (ROC) curve

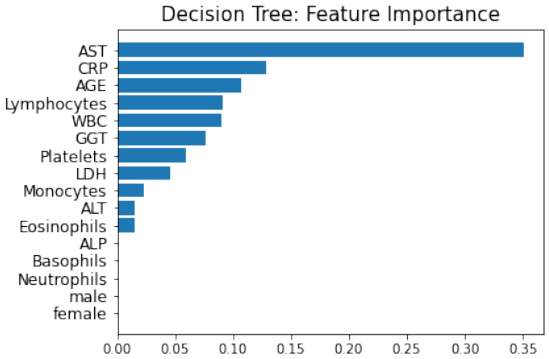


(b) Precision-Recall Curve

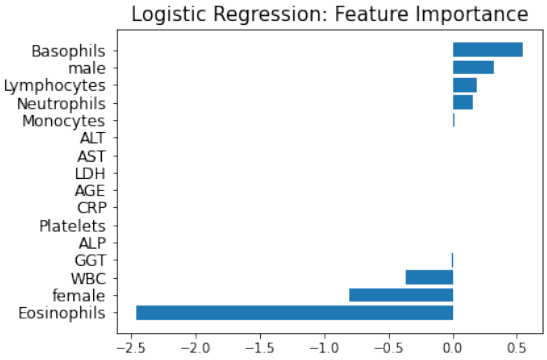
Figure B.3: ROC and Precision-Recall curve



(a) Random Forest



(b) Decision Tree



(c) Logistic Regression

Figure B.4: Feature importance plots

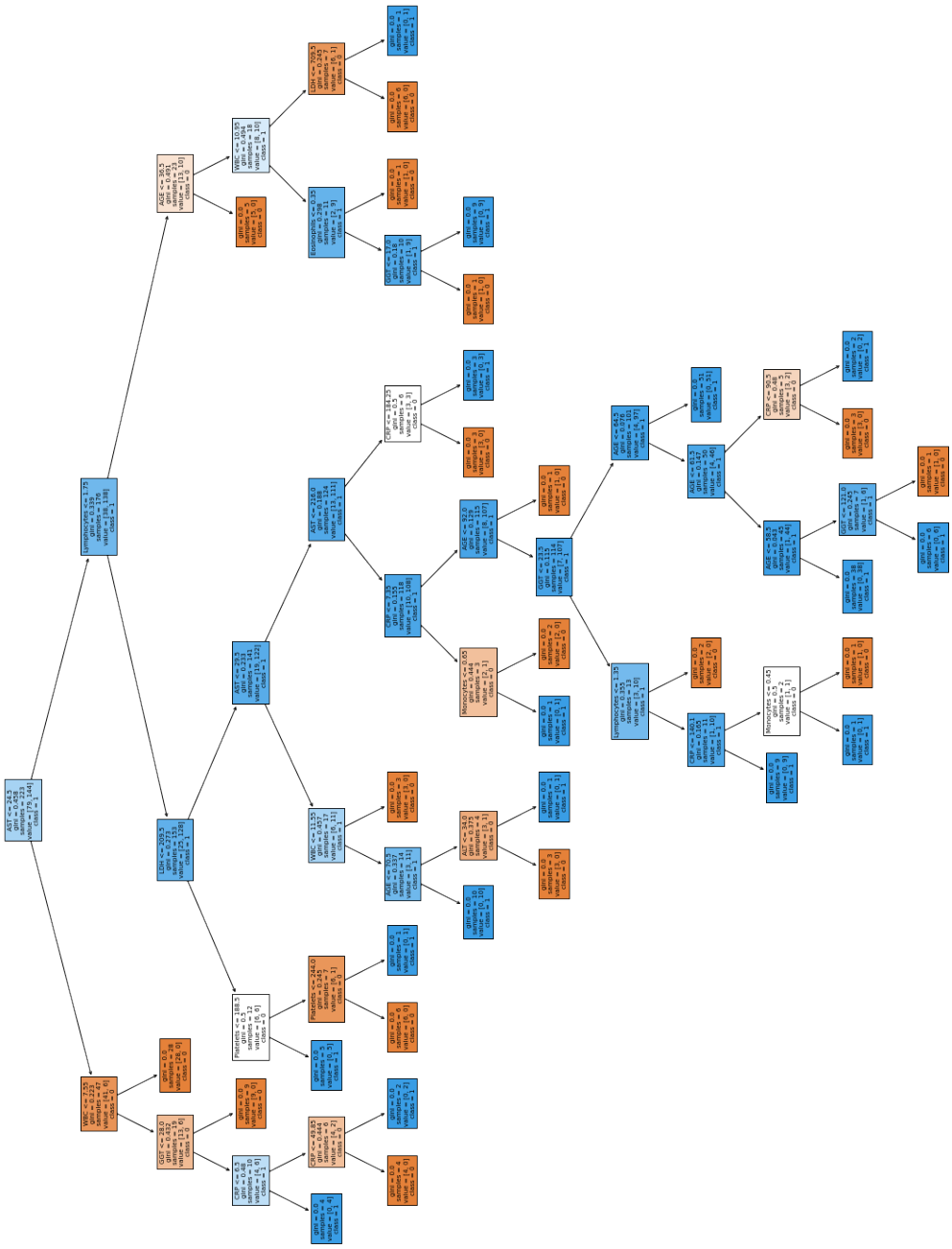


Figure B.5: Visualization of the Decision Tree produced during the validation phase

Appendix C

Supplementary Formulas

Bibliography

[1] A. K. Akobeng. "Understanding diagnostic tests 1: sensitivity, specificity and predictive values". In: *Acta Paediatr* 96.3 (2007), pp. 338–41. ISSN: 0803-5253 (Print) 0803-5253 (Linking). DOI: 10.1111/j.1651-2227.2006.00180.x. URL: https://www.ncbi.nlm.nih.gov/pubmed/17407452.

[2] *An Introduction to Statistical Learning*. Springer Texts in Statistics. 2013. ISBN: 978-1-4614-7137-0 978-1-4614-7138-7. DOI: 10.1007/978-1-4614-7138-7.

[3] M. J. Azur et al. "Multiple imputation by chained equations: what is it and how does it work?" In: *Int J Methods Psychiatr Res* 20.1 (2011), pp. 40–9. ISSN: 1557-0657 (Electronic) 1049-8931 (Linking). DOI: 10.1002/mpr.329. URL: https://www.ncbi.nlm.nih.gov/pubmed/21499542,%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/pdf/MPR-20-40.pdf.

[4] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 08856125. DOI: 10.1023/a:1010933404324. URL: https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf.

[5] D. Brinati et al. "Detection of COVID-19 Infection from Routine Blood Exams with Machine Learning: A Feasibility Study". In: *J Med Syst* 44.8 (2020), p. 135. ISSN: 1573-689X (Electronic) 0148-5598 (Linking). DOI: 10.1007/s10916-020-01597-4. URL: https://www.ncbi.nlm.nih.gov/pubmed/32607737,%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7326624/pdf/10916_2020_Article_1597%20.pdf,https://zenodo.org/record/3886927#.X3xhPO1CRPY.

[6] Stef van Buuren. *Flexible Imputation of Missing Data, Second Edition*. 2nd ed. Boca Raton, Fla: CRC Press, 2018. ISBN: 978-0-429-96034-5. URL: https://stefvanbuuren.name/fimd/.

[7] Stef van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate Imputation by Chained Equations inR". In: *Journal of Statistical Software* 45.3 (2011). ISSN: 1548-7660. DOI: 10.18637/jss.v045.i03.

[8] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Journal* (2006), pp. 233–240. ISSN: ISSN. DOI: 10.1145/1143844.1143874. URL: https://dl.acm.org/doi/pdf/10.1145/1143844.1143874.

[9] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874. ISSN: 01678655. DOI: 10.1016/j.patrec.2005.10.010. URL: https://www.sciencedirect.com/science/article/abs/pii/S016786550500303X?via%3Dih%20ub.

[10] John D. Kelleher, Aoife DArcy, and Brian Mac Namee. *Fundamentals of Machine Learning for Predictive Data Analytics: algorithms, worked examples, and case studies*. 1st ed. Camebridge, MA: The MIT Press, 2015. URL: https://mitpress.mit.edu/books/fundamentals-machine-learning-predictive-data-ana%20lytics.

[11] K. Kühnberger and N. Potyka. *Methods of Artifical Intelligence*. Lecture. 2018.

[12] Gilles Louppe. "Understanding Random Forests: From Theory to Practice". Thesis. 2014. DOI: http://dx.doi.org/10.13140/2.1.1570.5928.

[13] T. Saito and M. Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PLoS One* 10.3 (2015), e0118432. ISSN: 1932-6203 (Electronic) 1932-6203 (Linking). DOI: 10.1371/journal.pone.0118432. URL: https://www.ncbi.nlm.nih.gov/pubmed/25738806,%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4349800/pdf/pone.0118432.pdf.

[14] I. R. White, P. Royston, and A. M. Wood. "Multiple imputation using chained equations: Issues and guidance for practice". In: *Stat Med* 30.4 (2011), pp. 377–99. ISSN: 1097-0258 (Electronic) 0277-6715 (Linking). DOI: 10.1002/sim.4067. URL: https://www.ncbi.nlm.nih.gov/pubmed/21225900,%20https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4067.