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
December 14, 2020

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

Abstract

Acknowledgements

I want to thank...

Contents

1 Introduction **6**

2 Materials and Methods **7**

2.1 Data 7

2.2 Multivariate Imputation by Chained Equations 8

2.3 Classifiers 9

2.3.1 Decision Tree 9

2.3.2 Random Forest 10

2.3.3 Logistic regression 11

2.4 Model training 11

3 Results **13**

3.1 Results of own implementation 13

3.2 Comparison with original paper 13

4 Discussion **14**

4.1 Discussion of Results 14

4.2 Discussion of Methods 14

4.3 Scenarios for real-world validation 14

A Supplementary Tables **16**

B Supplementary Figures **20**

C Supplementary Formulas **21**

List of Figures

A.1 Kernel density plots of the numerical features of the data set 16

List of Tables

A.1 Overview over all features of the data set 17
A.2 Descriptive statistics for numerical features in data set (including missing values as in [3]) 17
A.3 Number of missing values and their proportion the total number of data points 18
A.4 Descriptive statistics for numerical features in data set (excluding missing values) 18
A.5 Results of Shapiro-Wilk test for normality for numerical features of the data set 19

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. [3]. 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 the result of a PCR test for Sars-CoV-2. A complete overview over the recorded features 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 naso-pharyngeal 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 and 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 contents of the column a string instead of a numerical value. This process created one missing values in place of the erroneous value.
Table A.4 provides common 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 a 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 you can see in Figure A.1, most of the data is non-normally

distributed. By inspecting the kernel density plots provided in Figure A.1, it is apparent that most of the features are non-normally distributed. Additionally, a Shapiro-Wilk test for normality was performed to confirm this observation, its results are provided in table A.5. 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 size of data set 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 observed values in the data set. 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 [5], 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 [6, 2]: 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 other variables in the imputation model. 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 results of the imputation that is 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 [4, 2, 6], 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, but setting m higher requires more computations and storage and may not be worth preoccupying these extra resources [4].
MICE assumes that the data is missing at random (MAR) that is the probability of data being missing 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.

Describe MICE algorithm (keine Bewertung, findet in Discussion statt)
Short description of implementation in Python using rpy2
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)
Maybe include PMM?

2.3 Classifiers

The original paper [3] implements 7 classifiers.

2.3.1 Decision Tree

In order to explain Random Forests, we will take a closer look at Decision Treessince they are the basic building block 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. 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 a 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 subset 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 an incorrect classfication of random sample. It is calculated as shown in Equation 2.3. The measure 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. For the Gini gain the attribute with lowest value is selected since we want to minimize the incorrect classification of attributes.

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

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

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 classification of the DT instances called tree bagging and feature subset selection.

2.3.3 Logistic regression

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 inner fold of the cross validation is used to determine the best hyperparameter combination for the classifiers using Grid Search which is 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.
The models are then evaluated using accuracy, balanced accuracy, sensitivity, specificity, positive predictive value, and area under the ROC curve. 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 has an effect on the classifier regarding class prevalence. Assuming the standard layout of a confusion matrix in a binary case, the balanced accuracy is defined as the average of sensitivity and specificity

$$balanced\ accuracy = \frac{\frac{TP}{TP+FN}\frac{TN}{TN+FP}}{2} \tag{2.5}$$

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 2.6. 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 false positive rate of a test. [1]

$$sensitivity = \frac{TP}{TP + FN} \tag{2.6}$$

Specificity is defined as the proportion of people without the disease and are identified correctly, i.e., who tested negative for a disease. It is calculated as show in Equation 2.7. Specificity only gives information about the proportion of people without the disease and a negative test and therefore cannot disclose anything about the false negative rate. [1]

$$specificity = \frac{TN}{TN + FP} \tag{2.7}$$

The positive predictive value (PPV) measures the post-test probability of the disease given a positive test. It is calculated as shown in Equation 2.8. [1]

$$PPV = \frac{TP}{TP + FP} \tag{2.8}$$

The hyperparameters of the best performing models for Random Forest and Logistic Regression respectively are then used to obtain the final classifiers. For the validation, the whole data set is split into training and validation data according to a 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 data set and then validated using the validation data. Additionally 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 performance of the classifiers regarding the slight imbalance in the data set.

Describe here k-fold nested cross validation, evaluation metrics used
Here maybe also implementation of MICE in Python using rpy2

Chapter 3

Results

3.1 Results of own implementation

3.2 Comparison with original paper

Chapter 4

Discussion

4.1 Discussion of Results

4.2 Discussion of Methods

4.3 Scenarios for real-world validation

Declaration

I declare that..

Appendix A

Supplementary Tables

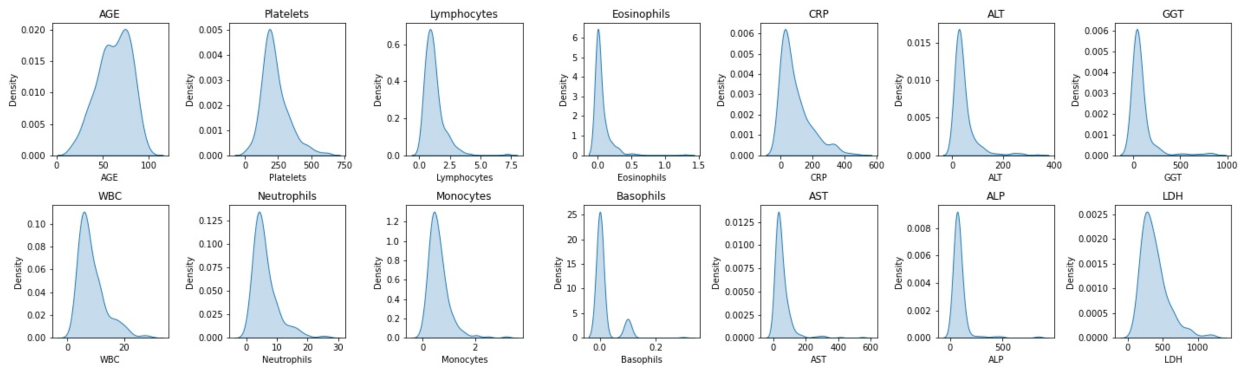

Figure A.1: Kernel density plots of the numerical features of the data set

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 [3])

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

Appendix B

Supplementary Figures

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] 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.

[3] 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.

[4] 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/.

[5] 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.

[6] 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.