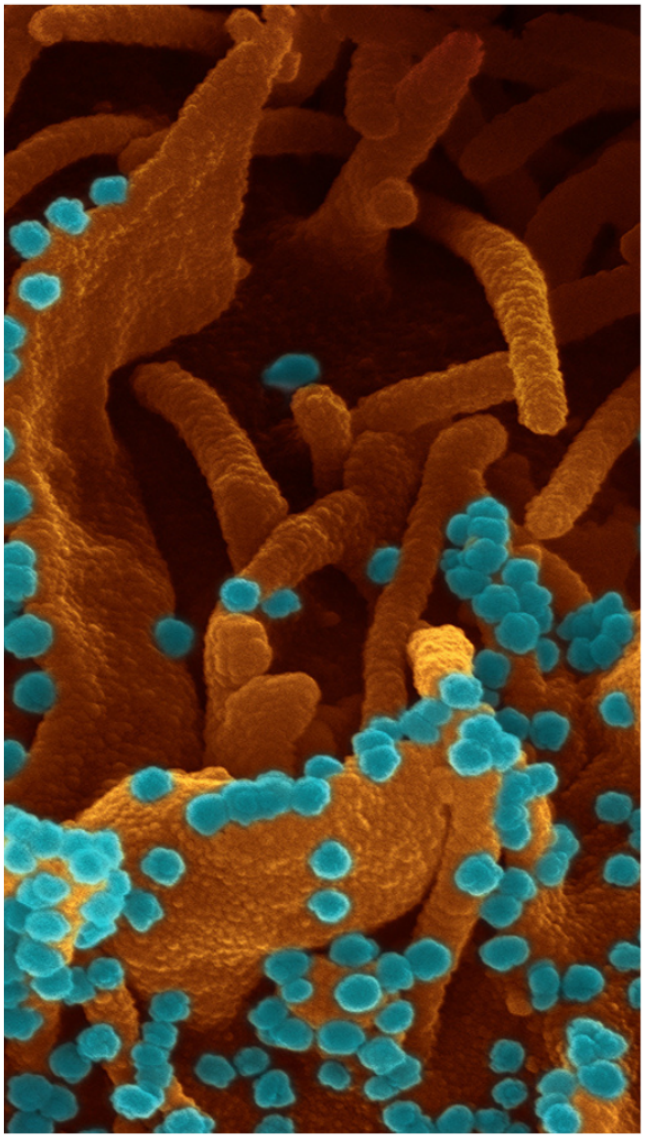
NOVA IMS Information Management School

DECEMBER 2022

THE SMITH PARASITE

AN UNKNOWN PARASITIC DISEASE



Machine Learning Unit

Professors:
Roberto Henriques
Carina Albuquerque
Ricardo Santos

Group 20:
André Cunha 20191224
Catarina Duarte 20191211
Fernando Cruz 20220646
Inês Magessi 20220590
Pedro Fernandes 20220592

Index

1. Introduction

The Cambridge Dictionary defines the word *disease* as "*(an) illness of people, animals, plants, etc., caused by infection*" and that is precisely what Dr. Smith discovered recently in England. The newly designated "Smith Parasite" has contaminated more than 5000 people, and there is still a lot to uncover about its characteristics and conditions of transmission.

Our data science team's purpose is to assess, in the most accurate way, if a person is likely to be a bearer of the parasite. As tools we were given small data sets regarding sociodemographic, health, and behavioral information of 800 people, some previously sick and others healthy.

2. Exploration

We started by exploring the 3 datasets (*Health*, *Habits* and *Demo*) separately to check if there were some problems or incoherences to be fixed, as well as to try to spot some relevant correlations and better understand which variables might have the biggest discriminatory power. Some general overviews can be found in Tables 1 to 3. During this step, we have also concluded that the data is well balanced in terms of the target variable, having a similar number of infected and healthy people.

2.1 Health

In this first dataset there were no inconsistencies, no missing values (either in the form of NaN, null, or any other unusual form) and no duplicates. However, there were some features with odd values that might be potential outliers. We registered these cases and later dealt with them. With the help of a pair grid, we checked the discriminatory power of pairs of variables [Fig 1] and found that there was no pair of variables considerably discriminatory. However, with both *Physical_Health* and *Mental_Health* we were able to see tendencies/patterns together with all other variables, meaning, both these features would probably be considered important when performing feature selection.

2.2 Habits

This data set was mainly composed of categorical variables, and again, we verified that the data had no relevant inconsistencies. There were also no unusual or missing values and no duplicates. After a general overview, we performed some visualizations using barplots to get to know the distributions of each feature. We also compared, across the discrete values of each variable, the number of people that had and did not have the disease, to see which variables seemed more relevant [Fig 2]. We reached the conclusion that the variables *Exercise*, *Fruit_Habit* and *Drinking_Habit* showed a bigger discriminatory power than the rest.

2.3 Demo

Finally, the *Demo* dataset was the only one with significant problems and inconsistencies to be treated. Firstly, we noticed that the variable *Education* had 13 missing values and the variable *Region* had two "London" values written with different cases, which we joined to make it only one. After taking care of these inconsistencies, we performed some visualizations [Fig 3] and, as in the *Habits* data set, we explored the discriminatory power of features. During this phase, we noticed that there were some outliers with respect to the *Birth_Year* feature that we later decided how to deal with.

3. Preprocessing

Throughout the exploration step, we found that there were some problems to be treated, so we proceeded to data preprocessing. This is an important step when building machine learning models because it helps improve the quality and reliability of the data, which can later be reflected in better models' performances. The preprocessing steps are described below.

3.1. Outliers

Outliers are extreme values that are clearly different from the other observations, and they should never be removed without proper investigation because they may contain a lot of revealing information if they are not due to error. We identified the potential outliers using the IQR method for the metric features *Height, Weight, Hight_Cholesterol, Blood_Pressure, Mental_Health, Physical_Health and Birth_Year* by plotting boxplots [Figs 4 and 5].

For the health characteristics, there were some unusual values for *High_Cholesterol* and *Blood_Pressure*, but after further research [1] , we concluded that these values, although unusual, are possible in people who are not so healthy and young. For this reason, we decided not to consider them outliers. There were also some observations with a Physical_Health value above 17, which means that these individuals had difficulty walking during most of the last 30 days. We checked to see if they were infected with the disease, but only one was. This means that they are either outliers due to misinterpretation of the question or reliable responses caused by another factor, such as other diseases. Therefore, we decided to evaluate a series of models with and without these observations using K-fold CV, which will be explained in detail later [4.1.], and we concluded that excluding them led to slightly worse results and also, given the small size of the dataset, removing those 25 observations could potentially cause further issues. Therefore, we decided not to remove them.

Finally, during exploration, we found that there were some observations with a *Birth_Year* below 1900, which we found curious. We also tested a series of models using CV with and without these observations to see to what extent excluding would affect the results. Excluding the observations improved the overall results, so we decided to remove them, ending up with minus 12 observations, 1.5% of the original dataset.

3.2. Types conversion

There were some data type conversions to be made since Python does not recognize the meaning of variables written as strings. While doing data exploration, two variables caught our attention: *Smoking_Habit* and *Exercise,* as those were boolean variables registered using a 'Yes' or 'No' response. Having that in mind we transformed the first type of responses to 1 and the latter to 0. This way, we would avoid any conflict when using models that only consider numerical entities.

3.3. Encoding of ordinal features

Besides transforming the boolean variables, we also decided to transform other ordinal features *Education, Drinking_Habit, Fruit_Habit, Water_Habit*, given as string responses, into numerical ones. As a result, their values become interpretable to models that only take numerical values.

Starting with *Education*, we created a new variable called *Years_Studied* to replace the old one. This way, we would have the amount of years corresponding to the level of education of the individual. A similar reasoning was done for *Fruit_Habit* and *Water_Habit*, in which we replaced people's responses to the number of fruit pieces and liters of water that people consumed on average, respectively. Finally, the *Drinking_Habit* possible responses were '*I usually consume alcohol every day*', '*I consider myself a social drinker*', '*I do not consume any type of alcohol*'. This feature's answers are not as easy to quantify as the features mentioned above. Because of that, we decided to convert them to simple 2, 1, 0 values, respectively. All the conversions are detailed in Table 4.

3.4. Feature Engineering

To obtain the maximum information about the patients, we created the democratic binary variable *female* that was obtained through the existing contractions of Mr or Mrs title at the beginning of the variable *Name,* and dropped this variable. When analyzing this new feature, it was seen that women seemed to be more prone to catching the disease, which meant that it had discriminating power and would likely be kept during feature selection [Table 5].

3.5. Scaling

Scaling is one of the most important steps of preprocessing and can make a difference between a weak and a good model. There are some models that are not sensitive to feature scaling, such as Decision Trees, but for the majority of them it is a crucial factor.

For these reasons we chose to use the *MinMaxScaler* to map the features values to the range $[-1, 1]$ maintaining the shape of the data, thus not distorting the original shape. This scaler does not make any assumptions about the distribution of data, contrary to the *StandardScaler,* for example, and it was also the scaler that showed the best results in the majority of the models. Although it is not robust to the presence of outliers, given that we have already excluded them that should not be a problem.

3.6. Missing values

Missing data can cause biased estimates, leading to invalid conclusions. If data has missing values, it is likely that it is not representative of the population and the models will generalize poorly.

During exploration, checking for missing values was undoubtedly in our checklist and we got lucky given that there was only one feature with missing values, *Years_Studied,* counting 12 missing entries. We decided to fill this missing information using the *KNNImputer* and using only the features *Birth_Year* and *Female* to calculate the distances between neighbors. This method was chosen as the best alternative for our problem because we made the assumption that people with those similar characteristics would also have a high probability of having spent a similar amount of years in school.

3.7. Feature Selection

Feature selection involves narrowing down the features to help improve the model's performance and reduce its complexity. It is also important in order to reduce redundancy and avoid

passing irrelevant features to the model. To perform feature selection we resorted to several methods below described.

3.7.1. Filter methods

Filter methods are used to select relevant features based on certain statistical methods and their correlation with the target variable. They evaluate each feature independently from the others.

First, we checked if any of the numerical variables was univariate (variance is equal to 0). For categorical variables, we used the Chi-Squared test to check the relevance of the variables in the target. This is the only method used to select categorical variables as there are very few methods that can deal with categorical data. For the numerical variables we used the following criteria: *Mutual Information*, which measures linear and non-linear dependence between two variables, to select the 8 most important features; *Spearman Correlation* between all variables and the target, to keep only the features with correlation value greater than 0.75, as we consider this a reasonable threshold [Fig 6].

3.7.2. Wrapper methods

Wrapper methods are used to select features based on the performance of a model using a subset of features. These methods are more computationally expensive than filter methods, but give more accurate results because they capture all features working together instead of independently.

The approach employed was RFE (*Recursive Feature Elimination)*, which recursively fits a model and removes the least important feature until a desired number of features is reached. The importance of the features is defined by the model itself and, given that the model used is a Decision Tree, the features importance is given by their information gain. The number of features to keep is also dynamically chosen: we perform the RFE with a number of features to select ranging from 1 to the maximum possible and finally keep the amount that results in the best performance of the model.

3.7.3. Embedded methods - Lasso regularization

Embedded methods are a type of feature selection method that combines the selection of features with the model training process. Lasso regularization is a type of embedded method that uses the L1 norm of the coefficients as a penalty term in the optimization objective. This penalty term shrinks the coefficients of less important features towards zero, effectively removing them from the model. In our case we decided to remove any feature with a coefficient smaller than 0.1 using a SVM with Lasso penalty, since they are also considered not important by the model.

3.8. Transforming Categorical Variables

Due to machine learning models requiring numerical input, our categorical variables, *Region*, *Check-up and Diabetes*, had to be transformed into the mandatory format. In order to give some kind of meaning to the transformation and not have only dummies for the values, each category within the variable was replaced by the percentage of infected people with that characteristic.

4. Model and Feature Selection

4.1. Model Selection with Stratified K-Fold Cross Validation

To compare and select the models with the greatest potential, we averaged each model's performance using Stratified K-Fold Cross Validation. This data resampling technique is able to restrain overfitting values by focusing on evaluating the generalization power of the built models, seeing that it gives the model the opportunity to train on multiple train-test splits. The number of folds, *k*, was set to 5 seeing that, although 10 is a commonly agreed high performance value, due to the small amount of data, each fold would be trained with a recognizable small amount of observations.

In each fold, the generated sample passes through some of the preprocessing stages described above. The removal of outliers, types conversion, encoding of ordinal features and feature engineering were done *à priori* on the whole dataset. However, there are some preprocessing steps that need to be done using only train data in order to avoid inducing bias on the validation set, which is used to predict how the model is expected to perform on unseen data. For this reason, the *scaling*, *missing values treatment*, *feature selection* (including all mentioned methods) and *transformation of categorical variables* were performed for each fold. Next, a series of models were trained, the results were accessed and the ones that had the best average F1 scores, both in train and validation. were the ones we decided to proceed with, later going through fine tuning. The models assessed and respective results of this phase can be seen in Table 6.

4.2. Final Feature Selection

To make the final feature selection and choose the features we will be using in the remainder of our analysis, we used the results of the feature selection performed on the previously described cross validation. For numerical variables, given that there are 4 feature selection methods but we noticed that the *Spearman Correlation* would almost always not select any feature, we were left with 3 reliable methods. Consequently, if the feature was chosen by less than 2 of these remaining methods in the majority of the folds, meaning, none or only one method chooses the feature in at least 3 folds, it is discarded. For categorical variables, since we are using only one selection method, we decided that if it was not chosen in the majority of the folds, it is also discarded. Finally, we dropped all the variables that were decided to be discarded on the training, validation and test sets. These variables were *Height, Weight, Smoking_Habit, Drinking_Habit, Water_Habit, Years_Studied and Region.*

5. Best Models Fine Tuning

As mentioned above, we chose some of the models with the best cross validation average F1 score to proceed with. The *RandomForestClassifier*, *GradientBoostingClassifier*, *MLPClassifier, DecisionTree*, *BaggingClassifier* were the models with the greatest potential. Among these, we performed a deeper tuning of the first three. We decided not to go further with the *BaggingClassifier* because the *RandomForestClassifier* is a specific type of a bagging model and showed better results. Also, a Random Forest is always more robust and at least as accurate as a simple Decision Tree, so

we also decided not to go into further analysis of the latest. Regarding the models with poorer performance, it is likely that linear models, such as linear regression, do not work well because the data is not linearly separable. For these models to work, we would have to do some feature engineering, but this task is really difficult to do 'by hand' and there are models that already do this, such as MLP and SVM. We can see this by looking at the slightly higher score of an SVM with *rbf* as kernel and also the MLP score, which is much higher.

The fine tuning steps are split into two phases: the first where individual parameters are tested and the second, where the already pre searched ranges of parameters are fed into a Grid Search in order to find the best combination of them. For the first phase, we compare the average results of a K-Fold CV for each model with different parameters and the preprocessing steps are, once more, done inside each fold. All of these steps are listed below for each model.

5.1. Random Forest

Random Forests are a type of ensemble method used for classification that operate by constructing a multitude of decision trees and outputting the class that is the mode of the classes of the individual trees. All the results of the fine tuning can be seen in Table 7.

First, we tested the ***n_estimators*** parameter. Increasing this number will generally improve the performance, but it also increases the training time. A good rule of thumb is to start with a small number and gradually increase it until the performance plateaus, and that's exactly what we did. The val score stabilized at 100, so we considered the values *100, 150, 200* to later use in the grid search.

The ***bootstrap*** parameter determines whether or not the trees are constructed using samples drawn with replacement from the original dataset. If bootstrap is set to false, the whole dataset is used by each tree. We found out that using bootstrap results in better performance. Using this value as default from then on, we could tune the ***max_samples*** parameter, the maximum number of samples drawn from the dataset when constructing a tree. For this parameter, *None* proved to be the optimal value, therefore we decided to also use it as default in the *GridSearchCV*.

Next, we tested different values of ***max_depth,*** which determines the number of nodes from the root node of the tree to the farthest leaf node. The maximum possible depth of a tree is the number of features of the dataset, and given that we have 10 features, we trained models with this parameter set from 4 to None, which resulted in the *9* and *None* as the best ones.

Finally, the ***min_samples_split*** determines the minimum number of samples required to split an internal node in a tree. Given the small size of the dataset, 2 ended up being the best value.

With the previous parameters explored, we performed a *GridSearchCV* using a *RandomForestClassifier* with the before-chosen default parameters in the base model. The parameters tuned were then the ***criterion****,* with possible values *gini* and *entropy* and the ***max_features*** with *sqrt, log2* and *None.* The ***n_estimators*** and ***max_depth*** were fine tuned with the possible values mentioned above. The search revealed that the best parameters were *sqrt* as the *max_features, 1*50 *estimators*, max_depth equal to *9* and *gini* as the criterion. These are the results of a *GridSearchCV* with average validation score of 0.9804 and this model then showed an OOB Score of 0.9949.

5.2. Gradient Boosting

Gradient boosting is an ensemble technique, which combines a sequence of weak models, with each focusing on the errors of the previous models.

The methodology applied was similar to the previously explained. First, we looked into the ***learning_rate*** and ***n_estimators***. The *learning_rate* controls the shrinkage of the contribution of each tree on the final prediction and the number of estimators is the number of weak learners. These parameters are highly correlated, so we tested them at the same time and in two different ways. First, we set a smaller *learning rate* of 0.5 and we got the highest score with 150 estimators and then a larger value of 1 and we got the best score with 100. This makes sense given that the higher the *learning_rate,* the less estimators are needed. These results can be found in Table 8.

Next, we tested the ***subsample*** parameter, which determines the fraction of the training data that is used for each tree in the model. Using a *subsample* can help to prevent overfitting by training each tree on a smaller portion of the data. We obtained the highest score with a subsample set to 0.8. The ***max_features*** parameter specifies the number of features to consider when looking for the best split and the optimal value found was 2. Finally, the ***max_depth*** parameter specifies the maximum depth of the individual decision trees in the model and the highest score was reached around 5.

To identify the best combination of hyperparameters we used a *RandomizedGridSearchCV*. We decided to use this instead of a regular grid search because the *learning_rate* is a continuous value and therefore it would be time consuming to include a large set of values for this parameter. We found that the best overall model was a *GradientBoostingClassifier* with *learning_rate* of 0.907, 100 estimators, *max_depth* of 7, using ‘log2’ of features and all samples, achieving 0.994 validation score.

5.3. MLP

The MLP is a type of ANN that performs a series of linear followed by non linear transformations in the input features, being one of the most powerful models used nowadays.

The first parameter to take into consideration was the **solver**. This parameter defines the optimization algorithm used to adjust the weights in order to minimize the error. The solver that showed best performance was the LBFGS by a significant margin. Having in mind that this solver usually performs well in small sized data sets, the result makes sense and was expected. We then searched for the best ranges of other parameters having this solver chosen as the final one.

The next parameter to be tested was the **activation function**, which is the nonlinearity applied to the output of each neuron, and *tanh* was the best one, but only with a small difference compared with the others. Therefore we decided to test all of them again in our grid search. After this, we concluded that the optimal **architecture** of the network would be one hidden layer with around 50 neurons. Finally, we reached an ideal value of around 0.001 for the regularization factor ***alpha***. This parameter is the L2 regularization term and is used to prevent overfitting. These results are in Table 9.

All of these conclusions were then fed into a *RandomizedGridSearch* with cross validation in order to find the optimal combination of parameters. The best model found was an MLP with one hidden layer with 50 neurons, using sigmoid activation function, optimized with the LBFGS solver and using a regularization factor of 0.0014. It achieved a 0.994 validation score on the grid search.

6. Assessment and Final Model Choice

Once we finished fine tunning the 3 predictive models, we opted to use multiple performance metrics to help us choose which model was the best. For that, we compared F1-scores with CV and then the decision boundaries, ROC and AUC, and confusion matrices using a regular *train_test_split*.

The **F1-score with CV** can give a more reliable estimate of the model's performance, as it takes into account the variability of the results and helps reduce the risk of evaluating the model on a particularly easy or difficult test set. In this case, our results showed that the *RandomForestClassifier* was the one with the best performance by a small difference [Table 10].

Then, we observed the **decision boundaries**. A model with well-defined decision boundaries is likely to be able to generalize well to new data. We plotted them using a Voronoi Tessellation and visually assessed the results [Fig 7]. The detailed explanation of this method can be found in Annex 2. We noted that the separation lines were almost the same, meaning all models were capturing the information that distinguishes the classes. We also observed the non-existence of straight lines, concluding that they all well capture non linearities. Finally, the MLP showed a bit wider red area, which can mean better or worse generalization that would only be properly justified with more data.

Thirdly, we plotted the **ROC curve** and the respective **AUC** [Fig 8]. The receiver operating characteristic curve (ROC) is a graphical plot that shows the performance of a binary classifier as the classification threshold varies. The AUC quantifies the overall performance of a classifier and ranges from 0 to 1, with 0.5 corresponding to a classifier no better than random guessing, and 1 to a perfect classifier. Analysis revealed that the Random Forest and the Gradient Boosting achieved a flawless score, while the MLP displayed a slightly lower level of performance as it scored less on the TPR.

Finally we analyzed the **confusion matrix** [Fig 9]. The values in the matrix allow the calculation of various metrics, such as accuracy, precision and recall, which can help understand the strengths and weaknesses of the model. This metric once again showed that all models are highly similar to one another. Given that we are predicting if a person has a disease, it is most important to reduce the FN and we were once more able to see that the Random Forest and the Gradient Boosting were the best ones since they got the least False Negatives.

In conclusion, we have to bear in mind that the latter 3 methods were performed on a simple *train_test_split* and the results may be biased by the *random_state* used. With all these aspects taken into consideration, we ended up concluding that our best model was the *RandomForestClassifier*.

7. Conclusion

Although it was a challenge that carries great importance for world health, our group can confidently declare that it is possible to understand who are the people more likely to suffer from the Smith Parasite with the help of our predictive model. After intense research, careful methodology and parameters adjustment, our result used a *RandomForestClassifier* and reached a perfect training F1-score of 1.0+/-0.0 and an almost perfect score of 0.993+/-0.01 in validation with CV. Although the shortage of data was a limitation, these values translated into a very positive performance evaluation for the achieved predictions. It is safe to affirm that this was a crucial step to eradicate this parasite and, thankfully due to our work, more **known** than ever.

References

- [1] *Norma nº 054/2011 DE 27/12/2011* (no date) *Direção-Geral da Saúde*. Available at: https://www.dgs.pt/directrizes-da-dgs/normas-e-circulares-normativas/norma-n-0542011-de-27122011.aspx (Accessed: December 23, 2022);
- Berrar, D. (2018) *Cross-validation - researchgate*. Available at: https://www.researchgate.net/profile/Daniel-Berrar/publication/324701535_Cross-Validation/links/5cb4209c92851c8d22ec4349/Cross-Validation.pdf (Accessed: December 23, 2022);
- Brownlee, J. (2020) *A gentle introduction to k-fold cross-validation*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/k-fold-cross-validation/ (Accessed: December 22, 2022);
- *Disease* (no date) *Cambridge Dictionary*. Available at: https://dictionary.cambridge.org/dictionary/english-portuguese/disease (Accessed: December 22, 2022);
- *Feature selection* (no date) *How Does Feature Selection Benefit Machine Learning Tasks?* Available at: https://h2o.ai/wiki/feature-selection/ (Accessed: December 22, 2022);
- *Supervised learning* (no date) *scikit*. Available at: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning (Accessed: December 23, 2022);
- *Model selection and evaluation* (no date) *scikit*. Available at: https://scikit-learn.org/stable/model_selection.html (Accessed: December 23, 2022);
- *9.2.8 - quadratic discriminant analysis (QDA): Stat 508* (no date) *PennState: Statistics Online Courses*. Available at: https://online.stat.psu.edu/stat508/lesson/9/9.2/9.2.8 (Accessed: December 23, 2022);
- Migut, M.A.; Worring, M.; Veenman, C.J. (2015) *Visualizing Multi-Dimensional Decision Boundaries in 2D.* Available at: https://pure.uva.nl/ws/files/2110683/164710_431596.pdf (Accessed: December 23, 2022).

Annexes

1. Another predictive models

QDA: Quadratic Discriminant Analysis

In order to explain the QDA algorithm, we first need to see in detail how a simpler version of it, the LDA (Linear Discriminant Analysis), works.

The LDA is a technique used in machine learning to find a linear combination of features that best separate observations from different classes. The algorithm tries to maximize the separability between classes so the best decision can be made. To accomplish this, LDA works like PCA by reducing dimensions in data. However, these algorithms work differently because of the criterion used to perform the dimensionality reduction. In PCA, data is projected into dimensions that preserve the greatest variance in an unsupervised way, whereas LDA finds the directions that maximize the group separation.

But how does LDA do these projections? Firstly, there are some assumptions made by the algorithm: that the density of each feature X, given every class *k*, follows a Gaussian distribution and, for different *k,* the covariance matrix is identical, becoming this way a linear classifier. With these assumptions kept in mind, the algorithm maximizes the distance between the means of the observations of the classes and minimizes the variation within each class. In summary, it finds the axis that maximize the formula $\frac{(\mu_1-\mu_2)^2}{s_1^2+s_2^2}$, known as the Fisher's criterion.

QDA is an extension of LDA, which relaxes its assumptions by allowing different covariances within each class, making it possible to find non linear decision boundaries, thus being a more flexible algorithm. However, this flexibility comes at cost, given that QDA needs more data to estimate the parameters accurately and can become prone to overfitting in case the number of features is relatively small compared with the number of observations.

This model might be a good option to be tested given that it captures non linearities and is specialized in better separating classes.

2. Visualizing high dimensional decision boundaries in 2D

A Voronoi diagram is a mathematical concept used to partition a plane into regions based on the distance to a given set of points. The plane is divided into cells, one around each point, being the region of the plane nearer to that point than any other. In machine learning, Voronoi tessellation can be used to visualize the decision boundaries of a classifier. The main idea is to approximate these boundaries in 2D by overlaying Voronoi tessellations on the two-dimensional transformed data.

Seeing that we have a high dimensional space of 10 features, and given our limitation of only being able to observe graphics up to 3 dimensions, we first needed to reduce the dimensionality. We used the TSNE technique to map the validation data into a 2D space, and then performed a Voronoi tessellation to divide the reduced feature space into two regions: one for each class. In practice, a

KNNClassifier is used to approximate the original model's predictive behavior on the original data and apply the same technique in the 2D transformed data. For this, a *1NNClassifier* was fitted to the validation data, using the already predicted target values for this set, in order to extract the regions of space that would be classified with each class. The decision boundaries are the lines that divide the two regions

3. Figures and Tables

	Height	*Weight*	*High_Cholesterol*	*Blood_Pressure*	*Mental_Health*	*Physical_Health*
count	800.0	800.0	800.0	800.0	800.0	800.0
mean	167.80	67.8	249.32	131.05	17.35	4.56
std	7.97	12.1	51.57	17.05	5.39	5.45
min	151.0	40.0	130.00	94.00	0.00	0.00
25%	162.00	58.0	213.75	120.00	13.00	0.00
50%	167.0	68.0	244.00	130.00	18.00	3.00
75%	173.00	77.0	280.00	140.00	21.00	7.00
max	180.00	97.0	568.00	200.00	29.00	30.00

	Checkup	*Diabetes*
count	800	800
unique	4	4
top	More than 3 years	Neither I nor my immediate family have diabetes.
freq	429	392

Table 1

	count	*unique*	*top*	*freq*
Smoking_Habit	*800*	*2*	*No*	*673*
Drinking_Habit	*800*	*3*	*I usually consume alcohol every day*	*406*
Exercise	*800*	*2*	*No*	*536*
Fruit_Habit	*800*	*5*	*Less than 1. I do not consume fruits every day.*	*452*
Water_Habit	*800*	*3*	*Between one liter and two liters*	*364*

Table 2

	Birth_Year	Disease
count	800.00	800.00
mean	1966.04	0.51
std	15.42	0.50
min	1855.00	0.00
25%	1961.00	0.00
50%	1966.00	1.00
75%	1974.00	1.00«
max	1993.00	1.00

	Name	Region	Education
count	800	800	787
unique	799	10	6
top	Mr. Gary Miller	East Midlands	University Complete (3 or more years)

Table 3

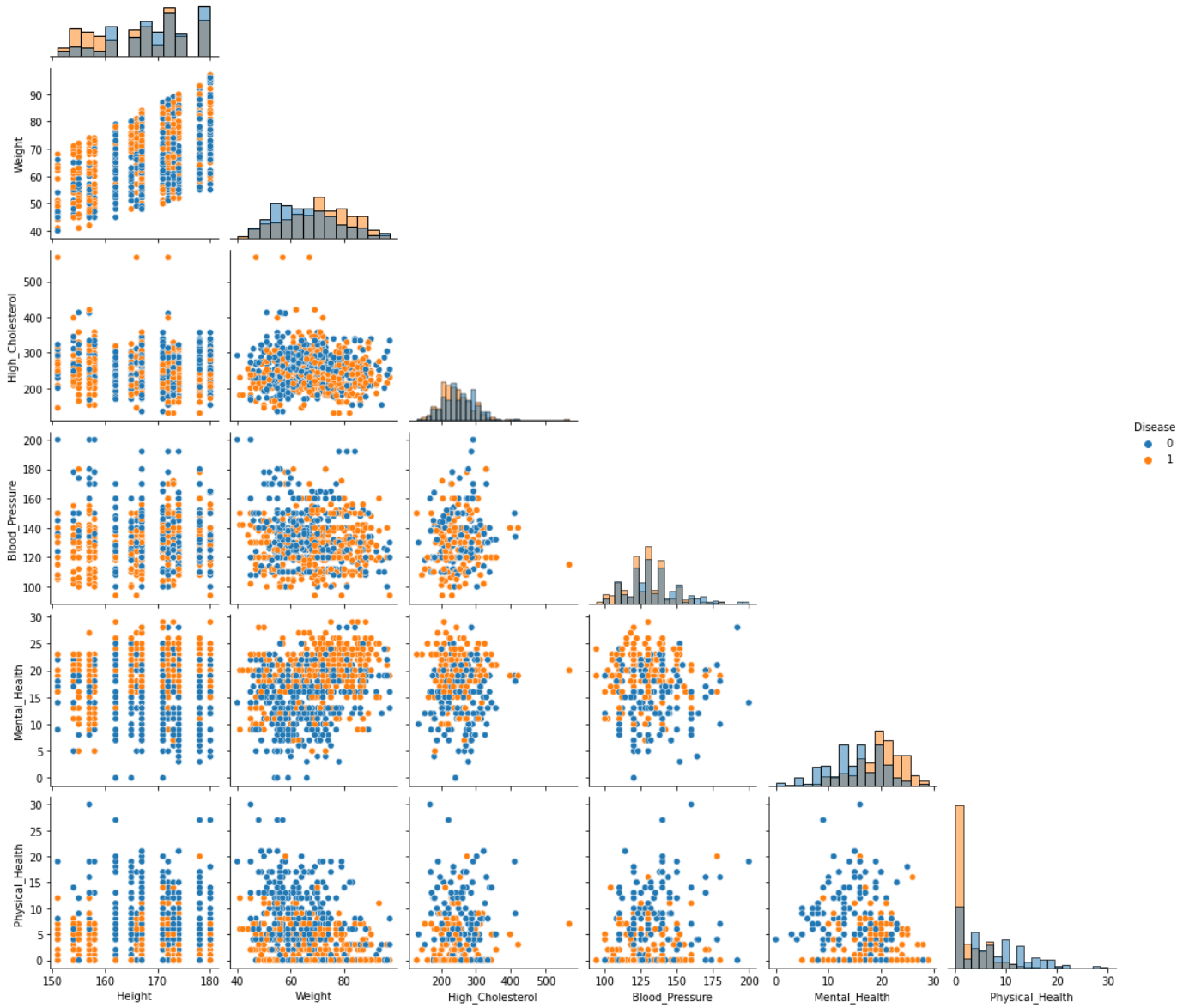


Figure1

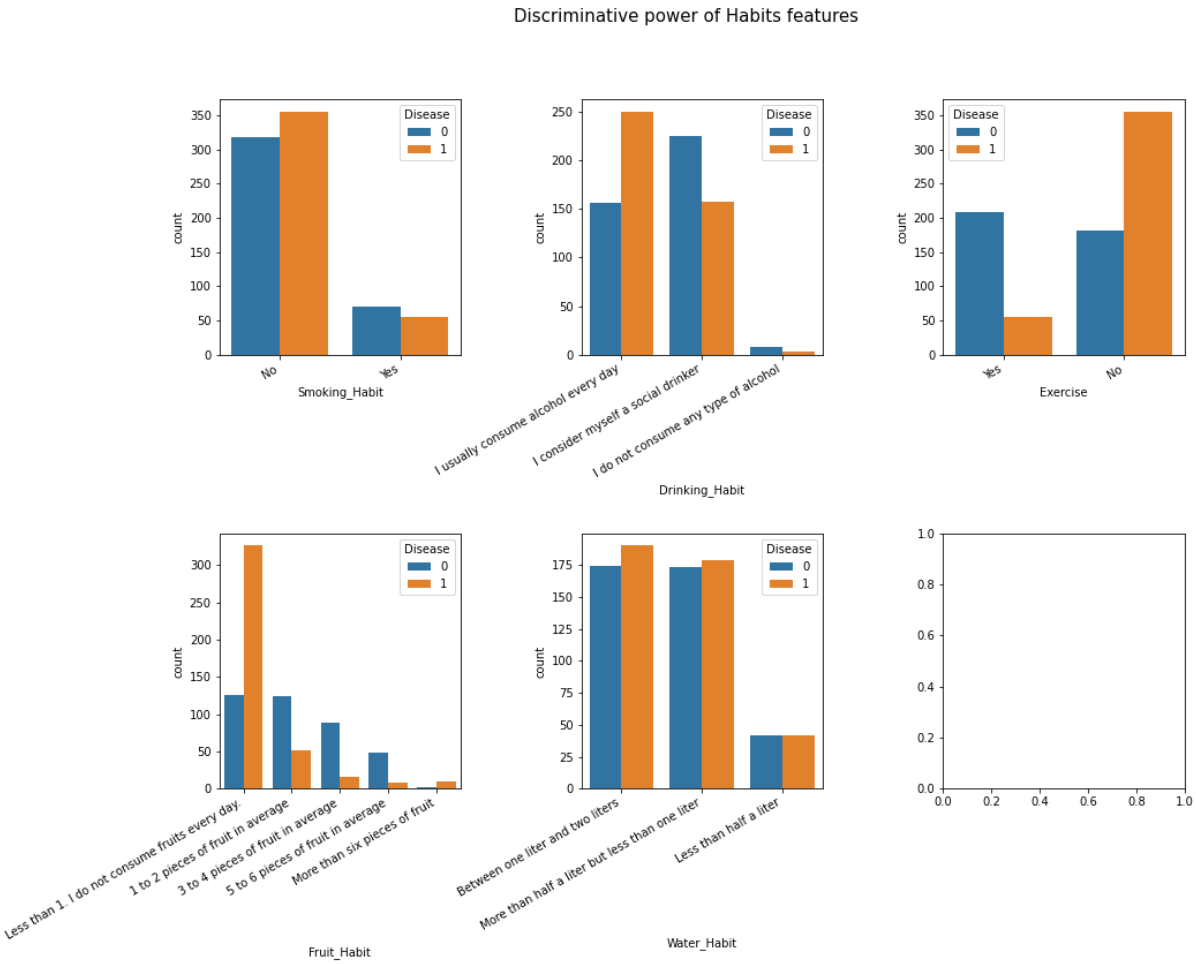


Figure 2

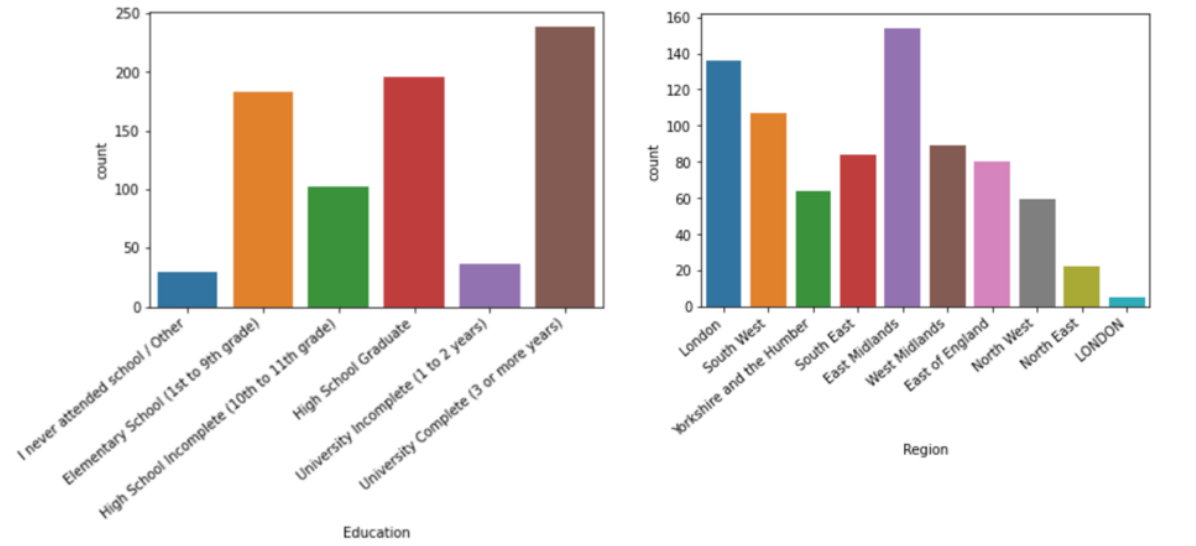


Figure 3

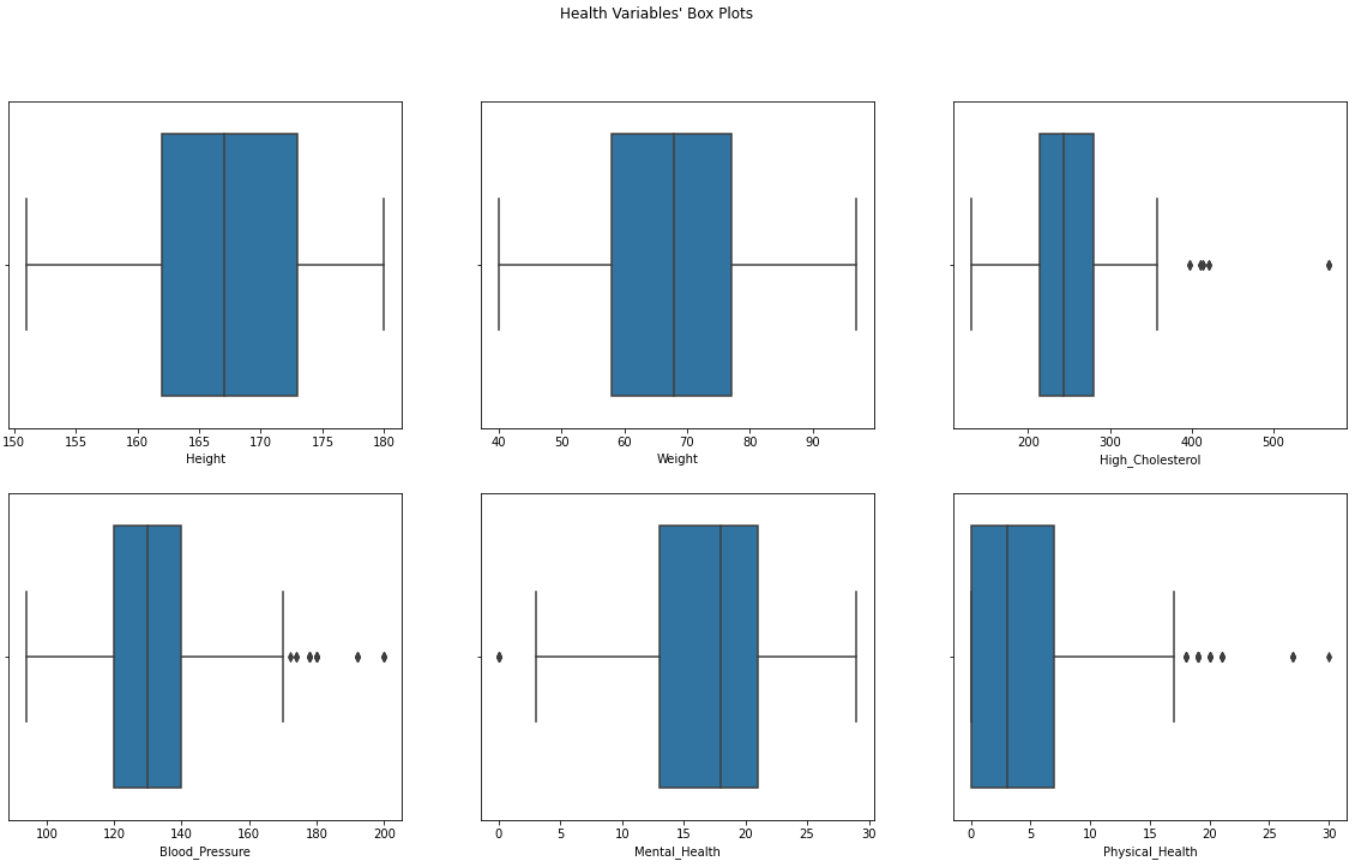


Figure 4

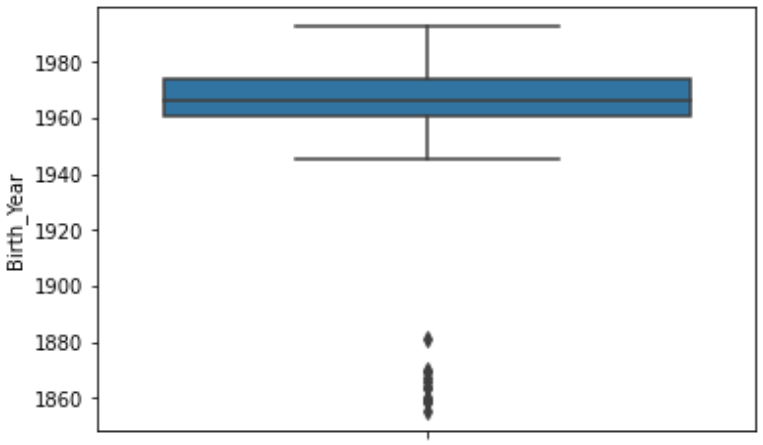


Figure 5

Feature	Old Value	New value
Education	*I never attended school / Other*	0
	Elementary school (1st to 9th grade)	9
	High School Incomplete (10th to 11th grade)	10.5
	High School Graduate	12
	University Incomplete (1 to 2 years)	13.5
	University Complete (3 or more years)	15
Fruit_Habit	*Less than 1. I do not consume fruits every day.*	0.5
	1 to 2 pieces of fruit in average	1.5
	3 to 4 pieces of fruit in average	3.5
	5 to 6 pieces of fruit in average	5.5
	More than six pieces of fruit	6
Water_Habit	*Less than half a liter*	0.25
	More than half a litter but less than one litter	0.75
	Between one liter and two liters	1.5
Drinking_Habit	*I usually consume alcohol every day*	2
	I consider myself a social drinker	1
	I do not consume any type of alcohol	0

Table 4

Female	Disease	Percentage
True	Yes	57.86%
	No	42.14%
False	Yes	75%
	No	25%

Table 5

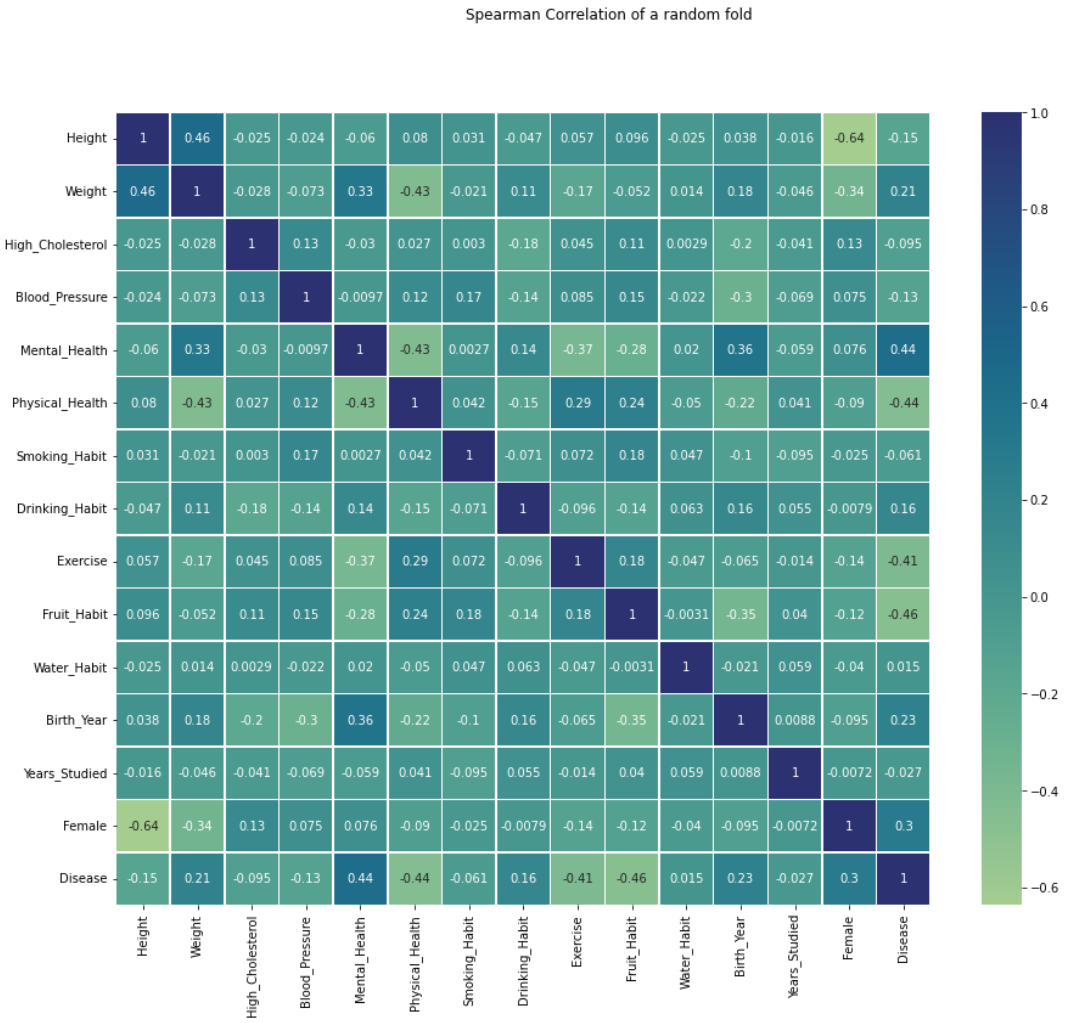


Figure 6

Model	Train	Val
LogisticRegression	0.854+/-0.01	0.838+/-0.02
KNNClassifier	0.929+/-0.01	0.851+/-0.03
DecisionTree	1.0+/-0.0	0.965+/-0.02
SVM	0.882+/-0.01	0.857+/-0.03
GaussianNB	0.847+/-0.01	0.842+/-0.01
QDA (Annex 1)	0.855+/-0.01	0.842+/-0.03
BaggingClassifier	0.998+/-0.0	0.976+/-0.01
RandomForestClassifier	1.0+/-0.0	0.989+/-0.01
AdaBoostClassifier	0.921+/-0.01	0.888+/-0.02
GradientBoostingClassifier	0.984+/-0.0	0.943+/-0.02
MLPClassifier	0.988+/-0.01	0.969+/-0.01

Table 6

Parameter	Options	Train	Val
Number of estimators	**10**	0.999+/-0.0	0.987+/-0.01
	50	1.0+/-0.0	0.99+/-0.01
	100	1.0+/-0.0	0.993+/-0.01
	200	1.0+/-0.0	0.993+/-0.01
	300	1.0+/-0.0	0.993+/-0.01
	500	1.0+/-0.0	0.993+/-0.01
Bootstrap	**True**	1.0+/-0.0	0.993+/-0.01
	False	1.0+/-0.0	0.991+/-0.01
Max samples	**0.4**	0.994+/-0.0	0.968+/-0.03
	0.6	0.998+/-0.0	0.984+/-0.01
	0.8	1.0+/-0.0	0.99+/-0.01
	None	1.0+/-0.0	0.993+/-0.01
Max depth	**4**	0.916+/-0.0	0.893+/-0.03
	5	0.951+/-0.01	0.911+/-0.02
	7	0.994+/-0.0	0.973+/-0.02
	8	0.999+/-0.0	0.989+/-0.01
	9	1.0+/-0.0	0.993+/-0.01
	None	1.0+/-0.0	0.993+/-0.01
Min samples to split	**2**	1.0+/-0.0	0.993+/-0.01
	4	1.0+/-0.0	0.99+/-0.01
	6	0.998+/-0.0	0.983+/-0.01
	8	0.994+/-0.0	0.965+/-0.02
	10	0.987+/-0.01	0.948+/-0.03

Table 7

Parameter	Options	Train	Val
Learning Rate (0.5) and *Number of estimators*	**30**	0.996+/-0.0	0.966+/-0.02
	50	1.0+/-0.0	0.984+/-0.01
	100	1.0+/-0.0	0.981+/-0.01
	150	1.0+/-0.0	0.984+/-0.01
	200	1.0+/-0.0	0.984+/-0.01
	400	1.0+/-0.0	0.981+/-0.01
Learning Rate (1) and *Number of estimators*	**20**	0.998+/-0.0	0.974+/-0.01
	30	1.0+/-0.0	0.981+/-0.01
	50	1.0+/-0.0	0.99+/-0.01
	100	1.0+/-0.0	0.991+/-0.01
	150	1.0+/-0.0	0.991+/-0.01
	200	1.0+/-0.0	0.986+/-0.01
	500	1.0+/-0.0	0.986+/-0.01
Subsamples	**0.2**	0.587+/-0.09	0.575+/-0.11
	0.4	0.94+/-0.05	0.922+/-0.06
	0.6	1.0+/-0.00	0.985+/-0.01
	0.8	1.0+/-0.01	0.993+/-0.01
	1.0	1.0+/-0.0	0.991+/-0.01
Max features	**2**	1.0+/-0.0	0.993+/-0.0
	0.5	1.0+/-0.0	0.984+/-0.01
	sqrt	1.0+/-0.0	0.989+/-0.01
	log2	1.0+/-0.0	0.989+/-0.01
	None	1.0+/-0.0	0.991+/-0.01
Max depth	**1**	0.881+/-0.01	0.869+/-0.03
	2	0.94+/-0.01	0.905+/-0.04
	5	1.0+/-0.0	0.989+/-0.01
	7	1.0+/-0.0	0.982+/-0.01
	9	1.0+/-0.0	0.978+/-0.01
	None	1.0+/-0.0	0.978+/-0.01

Table 8

Parameter	Options	Train	Val	Iterations
Solver	***sgd***	0.846+/-0.01	0.837+/-0.03	698.6+/-34.15
	lbfgs	1.0+/-0.0	0.978+/-0.01	207.6+/-46.97
	adam	0.981+/-0.0	0.944+/-0.02	1261.4+/-148.78
Activation function	***relu***	1.0+/-0.0	0.978+/-0.01	207.6+/-46.97
	logistic	1.0+/-0.0	0.987+/-0.01	187.6+/-23.53
	tanh	1.0+/-0.0	0.983+/-0.01	124.2+/-18.13
Number of hidden units	**(50,)**	1.0+/-0.0	0.988+/-0.01	225.2+/-43.26
	(100,)	1.0+/-0.0	0.978+/-0.01	207.6+/-46.97
	(200,)	1.0+/-0.0	0.984+/-0.01	190.8+/-62.54
Nr of layers	**(200,)**	1.0+/-0.0	0.988+/-0.01	225.2+/-43.26
	(200, 200)	1.0+/-0.0	0.978+/-0.01	207.6+/-46.97
	(200, 200, 200)	1.0+/-0.0	0.982+/-0.01	159.6+/-13.81
Alpha	**0.0001**	1.0+/-0.0	0.978+/-0.01	207.6+/-46.97
	0.001	1.0+/-0.0	0.988+/-0.02	497.4+/-54.8
	0.005	1.0+/-0.0	0.988+/-0.01	1660.4+/-418.76
	0.01	1.0+/-0.0	0.987+/-0.01	1917.4+/-605.34

Table 9

Models	Train	Val
RandomForestClassifier fine tuned	1.0+/-0.0	0.993+/-0.001
GradientBoostingClassifier fine tuned	1.0+/-0.0	0.991+/-0.01
MLP fine tuned	1.0+/-0.0	0.986+/-0.01

Table 10

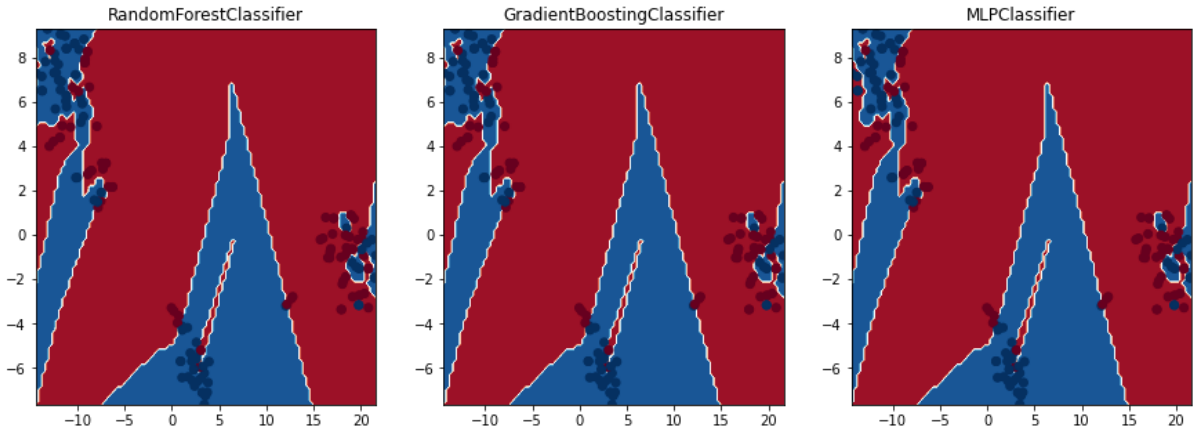


Figure 7

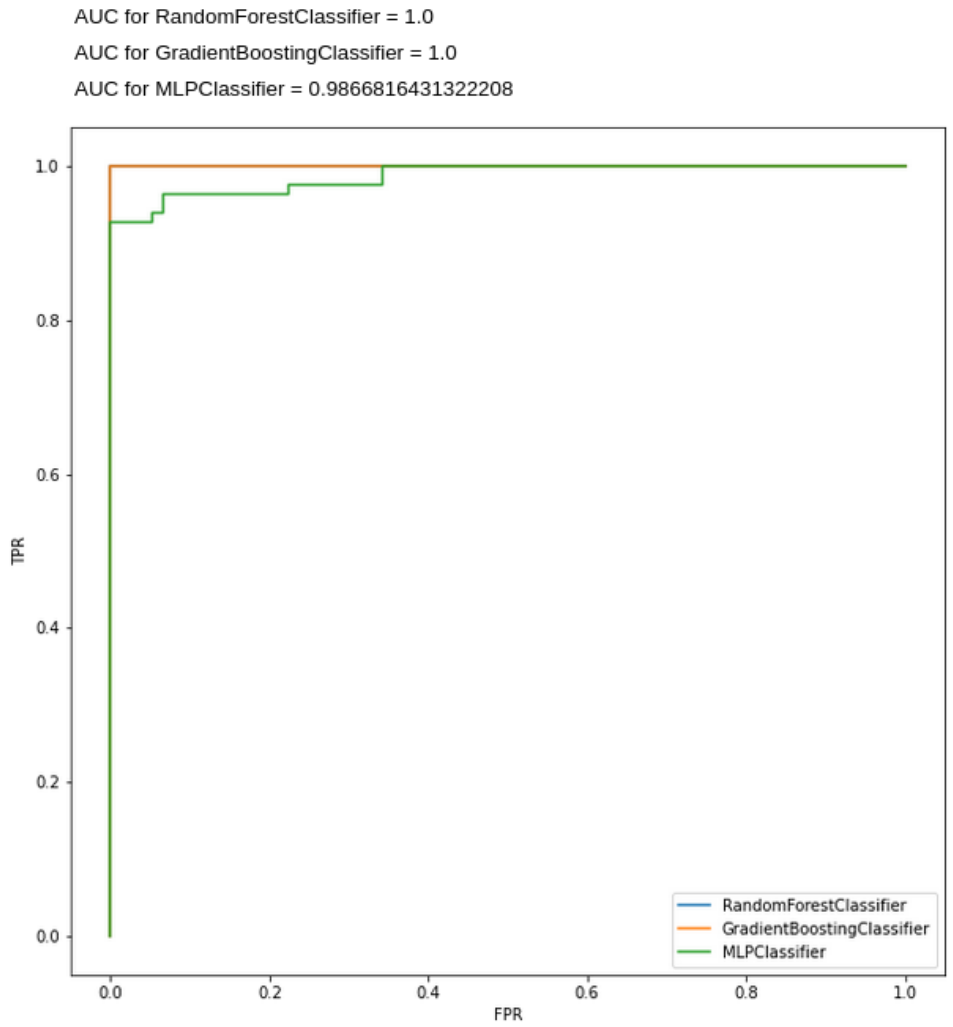


Figure 8

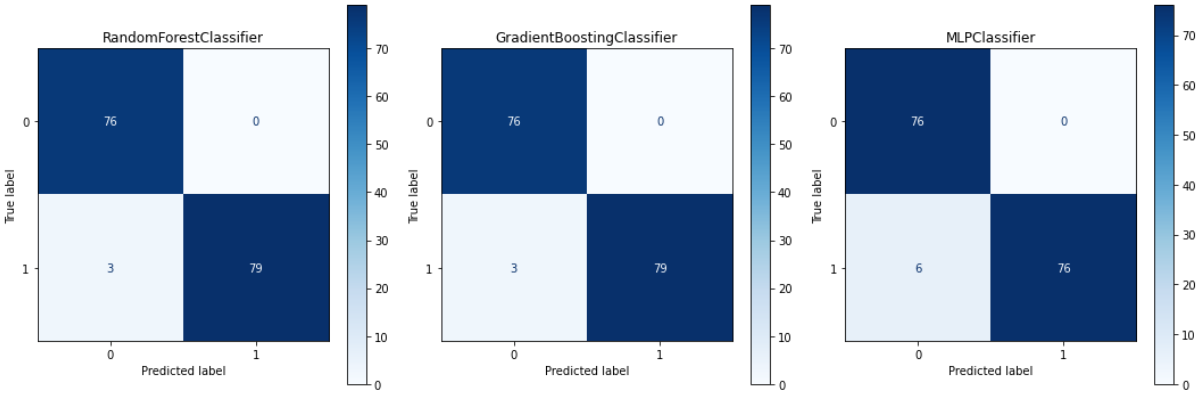

RandomForestClassifier
GradientBoostingClassifier
MLPClassifier
True label
Predicted label
76
0
3
79
6

Figure 9

END