NOVA IMS
Information Management School

"THE SMITH PARASITE" MACHINE LEARNING REPORT

DECEMBER 2022

Supervised Models
Master Degree in Data Science and Advanced Analytics

Group 11
Daila Alexandre | r20191182
Diogo Silva | 20221393
Isabel Dias | r20191215
Joana Sousa | r20191205
Tiago Santos | r20191263

Teaching Staff
Roberto André Pereira Henriques
Carina Isabel Andrade Albuquerque
Ricardo Miguel Costa Santos

Index

Introduction

The purpose of this project is to solve “The Smith Parasite” Kaggle case using Machine Learning knowledge. This case regards a new disease that already spread to over 5000 people and the investigator group aims to determine who is more prone to catch the disease, so that people are more aware of the disease’s symptoms and consequences.

In order to do this, there’s available sociodemographic, health and behavioural data regarding 800 patients, with the ground truth information regarding if they contracted the disease or not. As it is expected that the provided variables influence the propensity to catch the disease, this data will be used to create a predictive model that will be applied to 225 patients where the values of the target variable - having caught the disease or not - are unknown.

Data

To start this project, the 3 datasets were imported, the variable *Patient ID* was set as index since there are no duplicates and merged them into one. Since all the datasets before merging have the same number of observations as the merged dataset, it was a perfect correspondence between them.

The sociodemographic dataset has 4 explanatory variables and the target variable. The health dataset has 5 explanatory variables, and the habits dataset has 8 explanatory variables. In total, there are 17 explanatory variables. The 18 variables are presented in the Annex [Table 1].

Exploration and Visualisation

The group started by getting a quick insight of the data by using the method info. The dataset has 800 observations, of which 13 have missing values in the variable *Education* (1,5% of the observations) and it was also verified that there were no duplicated rows.

The group then checked the main descriptive statistics, with *describe()*, for categorical and numerical variables separately. It came to attention that someone born in 1855 should not be alive nowadays, and that at least 75% of the patients have a high Cholesterol, with the max value being abnormally high. There is one repeated name but, since there are no duplicated rows, it means that there are 2 different people with the same name. Names appear to not be relevant to the analysis, considering the high unique value count (799 out of 800).

After comparing the descriptive statistics of the dataframe and of the rows that have the 13 missing values no pattern of the missing values was found.

The values’ frequencies were analysed for each variable. It was concluded that the dataset was balanced (approximately half of the patients form the sample had the disease), there were 7 patients that were born before 1920, and that the *Region* has "London" duplicated, but with different cases, this was solved by putting all values from the variable as lower case.

Additionally, to check for patterns, the percentage of diseased people was calculated by category for each variable. In order to better visualize the previous results, the density and frequencies of patients with and without the disease were plotted for each variable using a *displot*. (Figure 1)

The conclusion was that for the numerical features *Birth_Year* and *Weight* people with the disease seem to have a higher values, but the difference isn't significant; The *Height* feature has a spike in the diseased people on lower heights, but otherwise the distribution is similar for both outputs; The features *High_Cholesterol* and *Blood_Pressure* seem to have a slightly higher distribution for people without the disease as they have a heavier right tail, except for an outlier of the variable *High_Cholesterol* that has the disease; The feature *Mental_Health* has a heavier left tail on the healthy patients;

For the Categorical Features the variables *Region, Education, Smoking_Habit, Water_Habit* seem to be unrelated to the output and the variables *Drinking_Habit, Exercise, Fruit_habit, Height, Weight, Checkup* and *Diabetes* have some degree of correlation to the output. (Figure 2)

Still, the importance of the variables will be explored further in the *Feature Selection* section.

Afterwards, the outliers were searched for through box plots in the numeric variables and they were present in *High_Cholesterol, Blood_Pressure, Physical_Health* and *Mental_Health, and Birth_Year* - which were probably mistakes as no one born before 1882 should still be alive*. (*Figure 3*)*

In order to better visualize the severity of the outliers, the values were graphed as histograms. Since the outliers of *Birth_Year, High_Cholesterol* and *Physical_Health* are significantly distant from the other values, and there are few of these values, it was decided to remove them while keeping in mind that those values in *Birth_year* and *High_Cholesterol* should be impossible (Birth year before 1882 and a Cholesterol above 500). (Figure 4)

Finally, the potential relationships were checked with a pairplot, and then several scatter plots were built to further visualize any correlations that might exist. From the graphs, (Figure 5), the most evident correlations were between *Height* and *Weight* and *High_Cholesterol* and *Birth_Year*, even though this latter has a small incline. Other pairs were also checked, namely, *Weight* and *Mental_Health*, *Birth_Year* and *Mental_Health*, *Birth_Year* and *Blood_Pressure*, and *Birth_Year* and *High_Cholesterol*. However, the best fitting line did not seem to adjust well to the data, as there was a lot of error.

Data Cleaning and Wrangling

After saving the original dataset the cleaning starts with the previously mentioned outliers being removed – these observations corresponded to 2.22% of the data set, which is less than 3% and within the acceptable range to be removed.

The group created the variable *Female* (with value 1 if the patient is a female and 0 if it's a male) from the prefix "Mr." (male) and "Mrs." (female) from the column *Name*. The variable *Birth_Year* was converted to *Age* since it was deemed important to keep in mind the scale of the variation of the feature and to make it more interpretable – it was created by subtracting the variable *Birth_Year* from the current

date. To reduce dimensionality, a new variable *BMI*- Body Mass Index- was created from dividing the weight with the square of the height. In the variable *Checkup*, the categories “Less than 3 months” and “Less than 3 years but more than 1 year” were grouped into “Less than 3 years” since there were only 6 observations in the first mentioned category. The variables *High_Cholesterol* and *Blood_Pressure* were binned according to sub-groups that are applied in the medical field; the binned features were created while keeping the discrete features in order to later see which will be more important. In the *Fruit_Habit* variable the group realized that people with extreme behaviour had a significant larger percentage of diseased people so, in an effort to facilitate the models’ job of recognizing patterns, a binary feature stating if it's an extreme fruit habit (1) or not (0) was created, *Fruit_Extremes*. Afterwards, the variable *Region*, where ranks were not applicable, since there is no logical order, dummy variables were created, for n-1 values that the variable can take. The rest of the categorical features were converted to ranks. After this it wouldn’t be an issue for models that can only work with numerical variables.

Then the density and frequencies of patients with and without the disease were plotted for each value of each new feature. (Figure 6) From this output, most of the variables do not have a strong predictive power. However, there are exceptions like *Fruit_Extremes, Female,* and *BMI*. Still, it is not enough to discard these variables, as sometimes the information that the variables give even if it has a small informative power can be complementary to the information provided by other variables. This will be explored further in the Feature Selection section.

After this, the 12 missing values remained in the variable *Education*. The percentage of missing values is small – 1.5% - so it should be safe to remove them without causing a big impact on the dataset, since the total of the removed observations would be slightly more than the 3% previously mentioned. There is also the option of imputing the missing values. Both were tried and, comparing the performance between the models using the 2 methods, the KNN Imputer led to the best performances.

The dataset was then separated into categorical and numerical variables and then split the dataset in data for training (80%) and for validation (20%). A dataframe without the dummies and the original *Region* feature was kept in order to use in the Feature Selection section.

Finally, the best scaling was looked into by comparing the scores of 3 models while using 4 scaling methods and the unscaled data. (Figure 7)

All models regardless of scaling seem to have very similar performances. The scaling selected was MinMaxScaler, because it was important that all the variables were comparable. Since the distribution of the explanatory variables do not have a skewed distribution (or many outliers), the Robust Scaler would have had a similar result. So, in the case of 2 equal solutions, the simpler one is always the best.

Feature Selection

The dataset at this point has 30 variables which may lead to worse performances because of the high dimensionality. To solve this problem, several methods were evaluated to select the most relevant features. The selection was divided in three parts Numeric, Categorical and Numerical & Categorical.

Numeric Features

Here only the numeric features were evaluated, so dataset with only the following features was created: *Age, Height, Weight, High Cholesterol, Blood Pressure, Mental Health, Physical Health*, and *BMI*.

Filter Methods

Univariate Variables

Checked if any variable was univariate, i.e., with variance equal to 0. Since no variables were univariate, none were deleted.

Kendall's Correlation

Then the correlation matrix was checked. Kendall's correlation was chosen because it's the most adequate for the data, since the input data is numerical, and the output is categorical. The correlation between *Weight* and *BMI is* the highest in the matrix (0.657) which was expected as the latter is directly proportional to the *Weight*. There's an interesting negative correlation between the *BMI* and *Physical_Health* (-0.371), but the correlation isn't high enough to create a multicollinearity problem in the models.

Wrapper Methods

RFE

Moving away from statistical methods, RFE was performed using the GradientBoostingClassifier to evaluate which variables are the most important to explain the target. The optimum features are: *Age. High_Cholesterol, Blood_Pressure, Mental_Health, Physical_Health, BMI.*

Embedded Methods

Lasso Regression

In the Lasso Regression, the *Weight* coefficient is 0, meaning that it is not a very important variable. The *Age* and *BMI* variable were also considered not important due to their small coefficients. The conclusions for this test were shown in a plot created through the function "plot_importance" that receives the coefficient values and the method name and plots the results sorted descending.

PCA

Principal Components Analysis was also carried out, not to select features, but to reduce dimensions. After analysing the Scree Plot, 4 Principal Components (PCs) were chosen as they reached 80% of the total variance in the dataset. Looking at the composition of each one of the four PCs, no interpretation was found for them. Because of that and since the PCs did not yield good scores, the results of the analysis were not used for the model.

Decision Numerical Data

Through Table 2 and after trial and error, through the graph of cross-validation scores of the numerical models (Figure 8), the best selection of features was: *Height, Mental_Health, Physical_Health, Blood_Pressure, High_Cholesterol, BMI, Age*.

Categorical Data

Similarly, to numerical variables, a dataset with only categorical variables was created to assess the features' importance.

Filter Methods

Chi-Square

The Chi-Squared test was performed with a 5% significance level, and it returned 9 important variables: *Drinking_Habit, Exercise, Fruit_Habit, Checkup, Diabetes, Category_BP, Category_HC, Fruit_Extremes* and *Female*. This test was made through the function "TestIndependence", that receives the variables for which test will be performed and the p-value and returns "Yes" or "No" according with the dependence between the two tested variables.

Mutual Information

Mutual Information was executed, by running the command *selectKBest* with the parameter *score_func* set to *mutual_info_classif*. The threshold used was 0.05 to choose the categories that were more dependent on the target. Since it is a stochastic method, the results are not completely constant, however, in general, the most important were the following: *Exercise, Fruit_Habit, Checkup, Diabetes, Fruit_Extremes* and *Category_HC.*

Decision Categorical Data

Through Table 3 and after trial and error, through the graph of cross-validation scores of categorical models (Figure 9), the best selection of features was: *Category_HC, Fruit_Extremes, Checkup, Diabetes, Exercise, Female, Category_BP.*

Categorical and Numerical Data

Feature Importance

In this section all variables were evaluated by assessing and plotting their feature importance for Random Forest with 500 number of estimators, ExtraTreesClassifer with the default parameters, Gradient Boosting with 500 number of estimators, 0.5 learning rate and random state 0 and Decision Tree with the default parameters. A table was made with an average placement to see which features were most important as seen in Table 4. (Figure 10)

Decision Numerical and Categorical

Through Table 3 and after trial and error, through the graph of cross-validation scores of categorical and numerical models (Figure 11), the best selection of features was: *Height, Mental_Health, Physical_Health, Blood_Pressure, High_Cholesterol, BMI, Age.*

Comparing the features selected in this section with the features of the previous sections, there is not much of a difference regarding the features chosen, with exception to the height feature, which is not very relevant according to the features' importance. Also, the discrete values of *Blood_Pressure* and *High_Cholesterol* are more important than their corresponding binning.

Modelling

There are 3 moments of Modelling and Assessment alternated. The **First** part of the modelling was done looking into the models applicable for the numeric data, categorical data, and both at the same time. The models applied to numeric data were: Logistic Regression, K-Nearest Neighbours, Multilayer Perception, Support Vector Classification [1], Gaussian Naïve Bayes and Gradient Boosting Classifier. The models applied to categorical data were Logistic Regression and Categorical Naïve Bayes. The models applied to numeric and categorical data were Logistic Regression, K-Nearest Neighbours, Decision Tree, Random Forest, Extra Trees, Multilayer Perception, Gradient Boosting and Histogram Gradient Boosting [2].

For the numeric and categorical data, the pairwise relationship was plotted to see if there was a discernible pattern that separated the 2 outputs, and it was found that for the numerical data there was some distinction, but not a clear separation. (Figure 12)

The models were then used with parameters that were iteratively improved, through trial and error, since, to assess the models, the default parameters of the methods may not do them justice when applied to the data.

The **Second** part of the modelling is focused on getting the best parameters for the models chosen in the first part. In order to achieve the optimal parameters, Grid Search was performed focusing on getting the best f1_score possible, while performing 7-fold Cross-Validation, as the goal is to arrive at the best possible model that can predict the testing data, and not the training data.

In order to better evaluate the ranges of numeric parameters to use in the Grid Search, the variation of the f1-score depending on all the numeric parameters was looked into for all models. The variation was checked by applying a range of intervals to the parameter under evaluation, while the other parameters were set to the default. This was made through the function *calculate_f1* that plots the f1-score (calculated through a 10-fold CV) for the selected model within a range of values in a defined parameter, allowing the team to visually select the best value for the parameter. If for the same parameter, there was the same variation on a previous model, the graph for the parameter was deleted.

A high *max_depth* is usually correlated with a high performance in the training, but lower on the test as it increases the risk of overfitting. From Figure 13 there does not seem to be overfitting for values above 20, as both f1-scores continuously increase, but the higher the values the lower the robustness of the model. There is a clear case of underfitting for *max_depth* of 2, so the range chosen is [3,20].

For *min_samples_split* and *min_samples_leaf,* as seen in the Figure 14 and Figure 15, the f1-scores get progressively worse, but the first values don't differ too much from the 1st result, so the range chosen is [1,4] for *min_samples_leaf and* [2,6] for *min_samples_split.*

Due to the small variations of *max_features* all values will be included in the analysis. [Figure 16]

As seen in the Figure 17 the f1-score continuously increases no *max_leaf_nodes* will be chosen.

With the increase of the *ccp_alpha* the f1-score continuously decreases so the value will be set to the default, 0. [Figure 18]

In Figure 19 and Figure 20, it is clear that the f1-score changes very little due to the *n_estimators*. The higher the number the more robust the model would be, but it would take progressively longer to run the Grid_Search Cross-Validation, the value chosen was 400.

As the *Max_iter* value increases so does the f1-score, as seen in Figure 21. But to save computational power, 200 was chosen as there was significant increase after it.

The f1-score seems to first increase and after a certain point decrease continuously with the variation of *learning_rate,* as seen in Figure 22, so the range chosen was *np.linspace(0.7,1.3,40).*

The **Third** part of the modelling consists of trying to create a model that can compete with the ExtraTreesClassifier model, reaching an f1-score of 1 while being more robust. For this, the group tried to use Voting [3] and Stacking Classifiers.

The models have high performance, meeting one of the criteria to apply Voting and Stacking Classifiers. To meet the second criteria as they must also not have high correlation between them. Since the performance is so high on the models that had their parameters tuned, it is likely that the correlation is high as well, since not much information that can be new. The KNN and the Logistic Regression were added to the possible models for Voting and Stacking to have models with smaller levels of performance. From the correlation matrix in Figure 25 the models chosen were Decision Trees, Gradient Boosting, Logistic Regression and KNN.

Assessment

In the **First Assessment** the methods were evaluated through the 10-fold cross-validation accuracies and the 5 models chosen to be further explored were Decision Tree Classifier, Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Classifier, and Histogram Gradient Boosting Classifier. These were chosen with the aid of boxplots, available in Figure 8, Figure 9, Figure 11.

The next 2 assessments use the following methods:

Evaluation of the models using a table that contains the metrics accuracy, precision, f1-score and recall. In order to check for overfitting and underfitting the table also presents 'Min Cv', 'Max Cv', 'Mean Cv'. This table was created by calling the function *results_table* that receives as parameters the model's

name, the model, the train data, the predictions and the validation data, and returns a table with the previous measures.

The robustness [4] of the models is also checked through the Jitter test [5], by adding random points to the model to see its response - the slower/smaller the response, the more robust the model is. The test was made through the function Jitter, that receives has parameters the train dataset and maximum value of noise to be added to the dataset, and the function *JitterTest* that receives the model, the independent variables, the target variable and the range of noise to test and returns the accuracy of the predictions to the different levels of noise.

In the **Second Assessment** the models are evaluated by the ROC curve and AUC [Figure 23], which suggests that the 5 methods all have a very high level of performance. Looking at the Metrics' table, table 5, it is evident that the 5 models perform at similar levels, which would make choosing 1 difficult, so their robustness is also checked. In Figure 2828 it is obvious that the most robust model is Extra Trees, that also has the highest f1-score and other metrics, as seen before, so it is the best model. There shouldn´t be overfitting in any of the models since the CV-scores are very high.

In the **Third Assessment** from Table 6 and Figure 2727 it is summarized that the two new models have a slightly worse performance as the Extra Trees model, and less robust as well, despite them being more robust than the base models and have the same metrics as the best base model. Despite this, both models have a significant improvement on their robustness compared to the base models and slightly worse performance than the model with the best metrics.

Taking all the previous information into account, the final model chosen was the Extra trees classifier, with the following parameters: *criterion*='entropy', *max_depth*=14, *max_features*=4*, n_estimators*=400.

Model Deployment

To be able to deploy the model, the test data was cleaned in the same way as the training data. The datasets were merged, new variables were created, the data was scaled and features selected. Finally, the extra trees model was used to predict the model, and with the estimated values, a pie chart was created to evaluate the distribution of the predicted. The final score with 40% of the test data was 1.0.

Conclusion

As expected, the ensemble classifiers obtained better results than single base models. Moreover, the methods involving trees seem to be the most suitable for the dataset, clearly outperforming the others in every metric. The final model submitted by the team was the Extra Trees Classifier with the parameters: *criterion* = 'entropy', *max_depth*=19, *max_features*=4 and *n_estimators* = 40. It returned a 1.0 provisory Kaggle f1-score upon submission which confirms the model's good metrics and robustness, tested throughout the assessment. This means that the group created a model successfully able to predict who are the patients with the "Smith Parasite".

References

[1] Stecanella, B. (2017). Support Vector Machines (SVM) Algorithm Explained. Retrieved 4 December 2022 from https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[2] Brownlee, J. (2020). Histogram-Based Gradient Boosting Ensembles in Python. Retrieved 2 December 2022 from https://machinelearningmastery.com/histogram-based-gradient-boosting-ensembles/

[3] Towards Data Science. Kumar S., Use Voting Classifier to improve the performance of your ML model. Retrieved 9 November 2022 from https://towardsdatascience.com/use-voting-classifier-to-improve-the-performance-of-your-ml-model-805345f9de0e

[4] Neumayer, E., & Plümper, T. (2017). A Typorology of Robustness Tests. In Robustness Tests for Quantitative Research (Methodological Tools in the Social Sciences, pp. 34-51). Retrieved 28 November 2022 from http://www.polsci.org/robustness/robustness.pdf

[5] Winder. Testing Model Robustness With Jitter. Retrieved 1 December 2022 from https://winder.ai/testing-model-robustness-with-jitter/

[6] Richardson, M. (2009, May). Principal Component Analysis. Retrieved 24 November 2022 from http://aurora.troja.mff.cuni.cz/nemec/idl/09bonus/pca.pdf

[7] Rahimi, K. (2022, August). Why you should not use correlation. Retrieved 6 December 2022 from https://www.kaggle.com/code/khashayarrahimi94/why-you-should-not-use-correlation

[8] Scikit Learn. sklearn.feature_selection.mutual_info_classif. Retrieved 6 December 2022 from https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html

[9] Thankachan, K. (2022, August 9). What? When? How?: ExtraTrees Classifier - Towards Data Science. Medium. Retrieved 23 November 2022 from https://towardsdatascience.com/what-when-how-extratrees-classifier-c939f905851c

[10] Dietterich, Thomas G. Ensemble Methods in Machine Learning. Retrieved 5 November 2022 from https://link.springer.com/chapter/10.1007/3-540-45014-9_1

[11] IBM. What is Bagging? Retrieved 6 November 2022 from https://www.ibm.com/cloud/learn/bagging

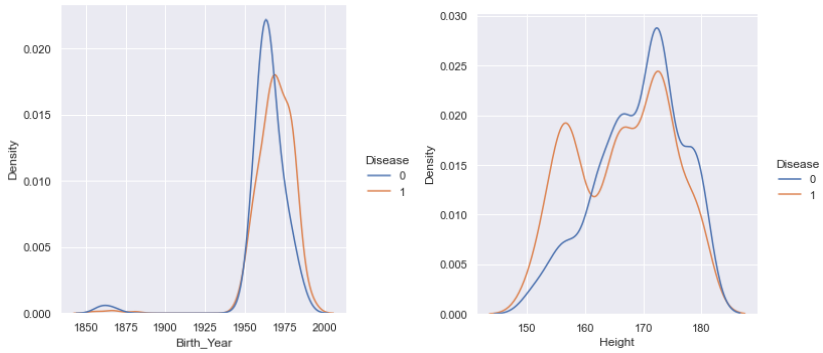
[12] GeeksforGeeks. (2019, November 25). ML | Voting Classifier using Sklearn. Retrieved 22 December 2022 from https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/

Annex

Tables and Figures

Variable Name	Data Type	Variable Name	Data Type
PatientID	Integer (Nominal)	Highcolestrol	Integer (Ratio)
Birth_Year	Integer (Interval)	Bloodpressure	Integer (Ratio)
Name	String (Nominal)	Mental Health	Integer (Ratio)
Region	String (Nominal)	Physical Health	Integer (Ratio)
Education	String (Ordinal)	Smoking_Habit	String (Binary)
Height	Integer (Ratio)	Drinking_Habit	String (Ordinal)
Weight	Integer (Ratio)	Exercise	String (Binary)
Checkup	String (Ordinal)	Fruit_Habit	String (Ordinal)
SDiabetes	String (Ordinal)	Water_Habit	String (Ordinal)

Table 1. ID and explanatory variables description



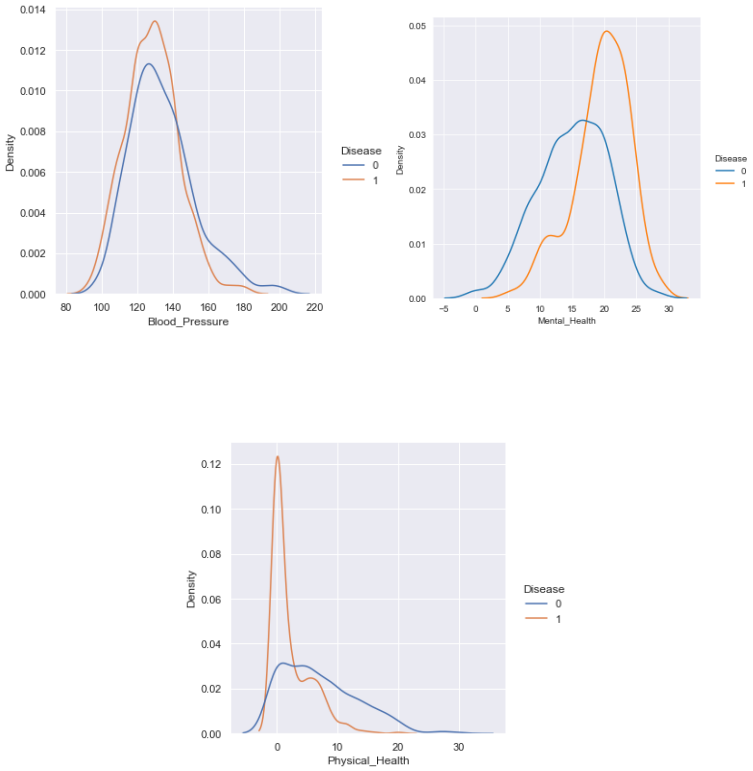
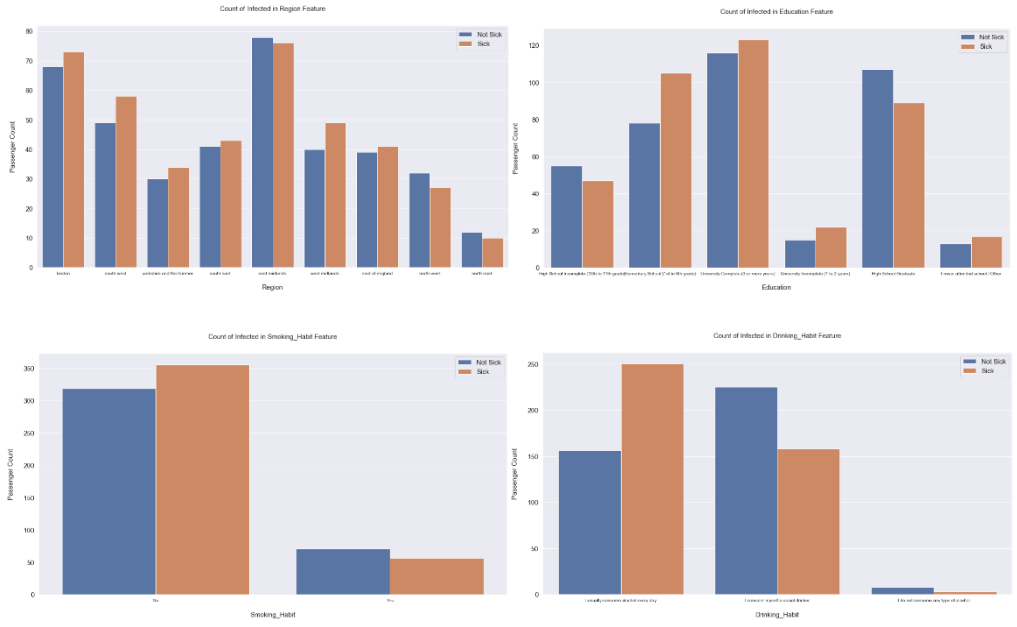


Figure 1. Visualization of the densities and frequencies of patients with and without the disease for Numerical Features.



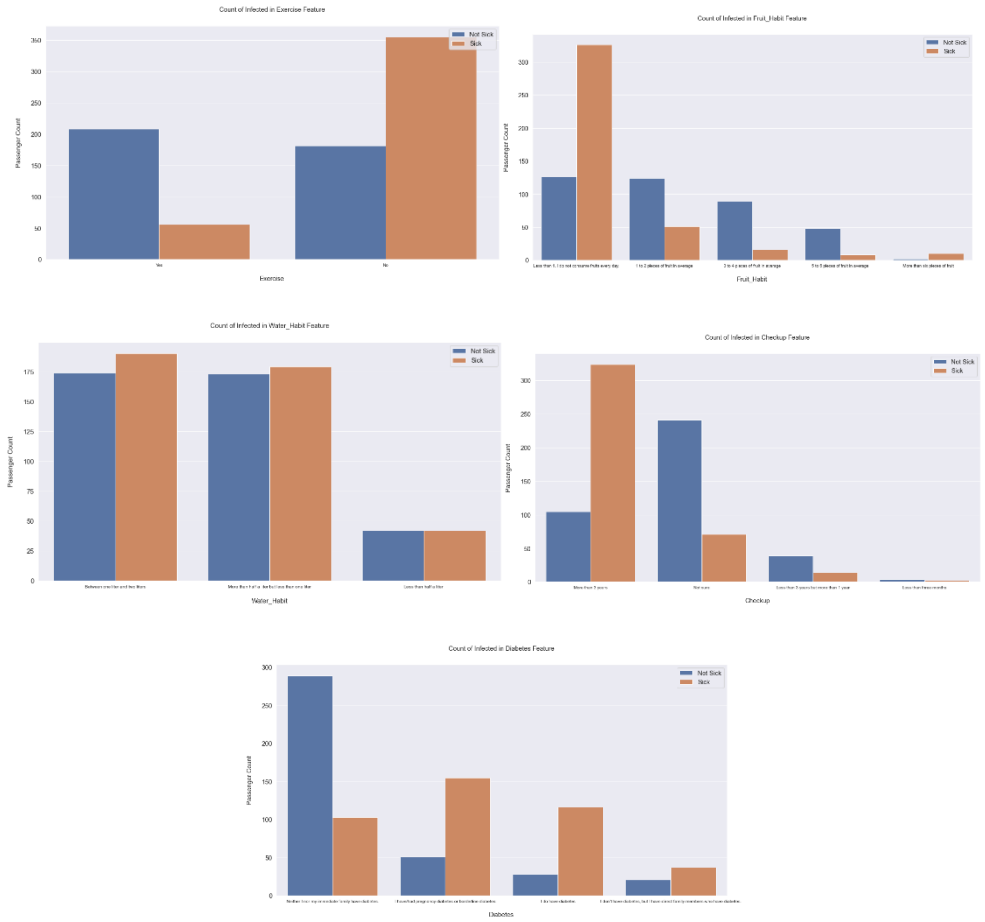


Figure 2. Visualization of the densities and frequencies of patients with and without the disease for Categorical Features.

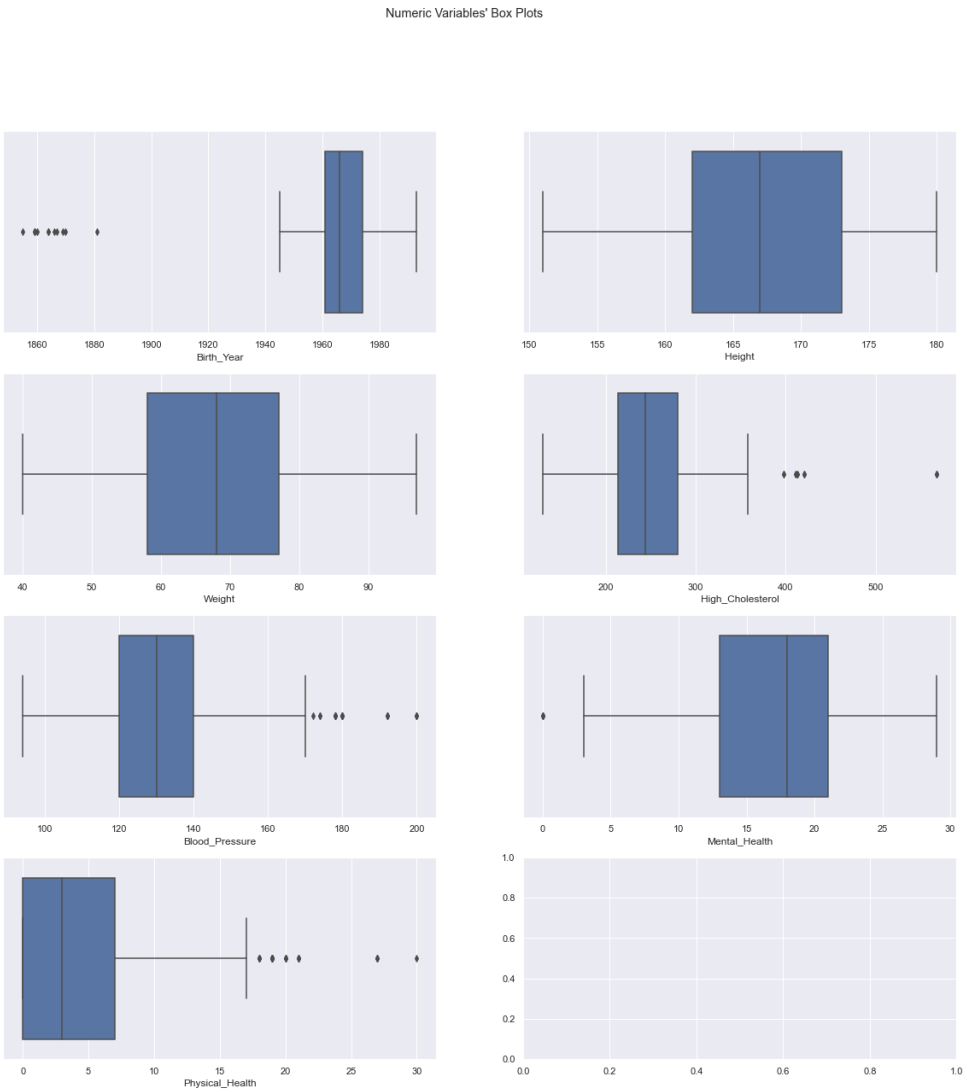


Figure 3. Outliers through boxplot

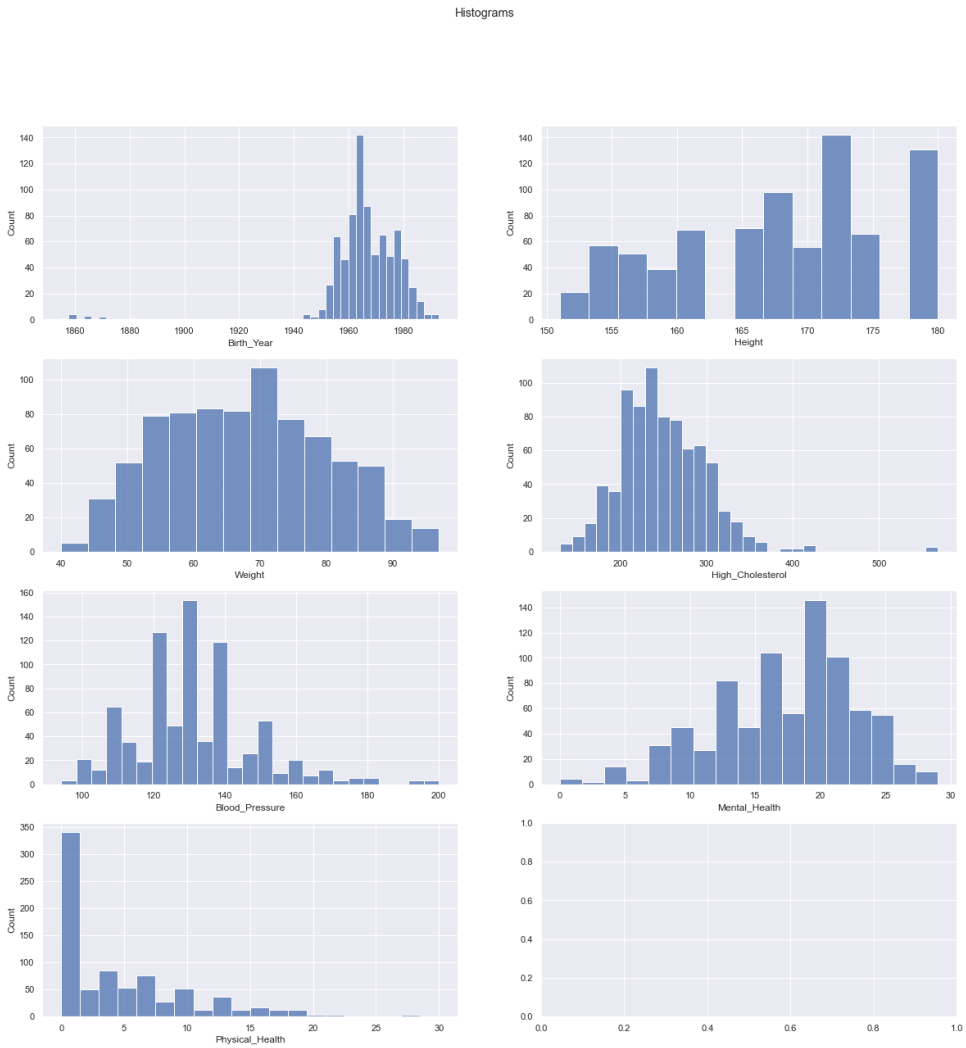
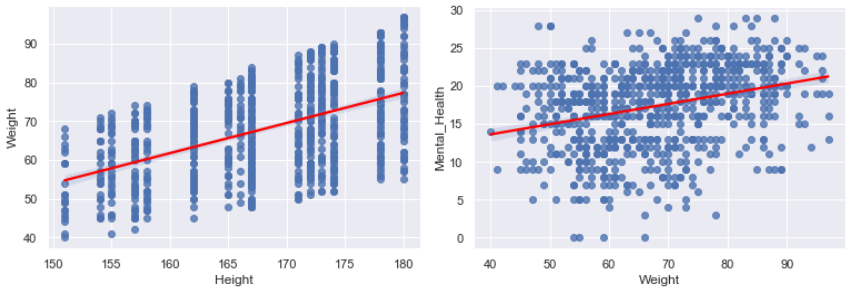


Figure 4. Histogram of Numeric Variables



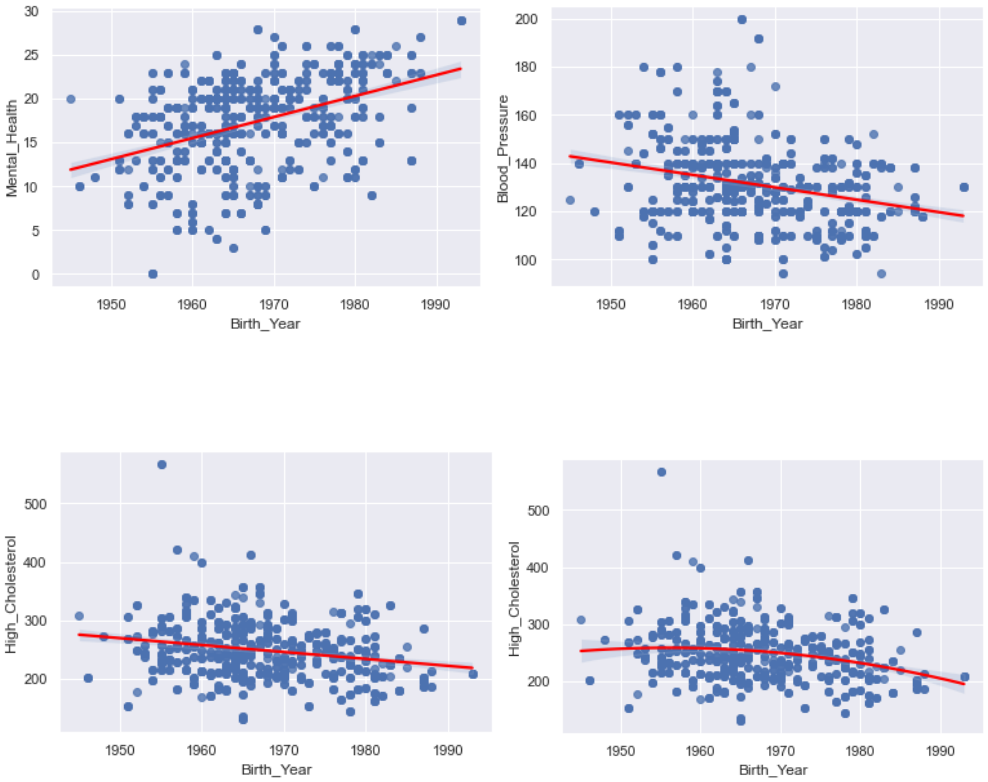
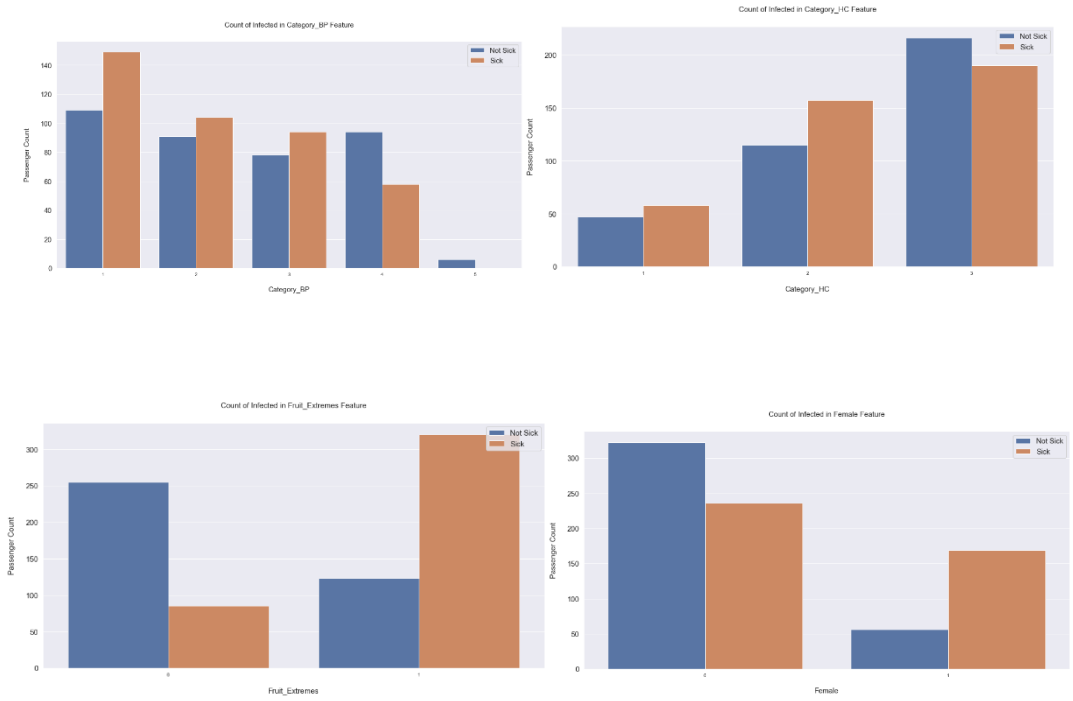


Figure 5. Scatterplots between variables with fitted line



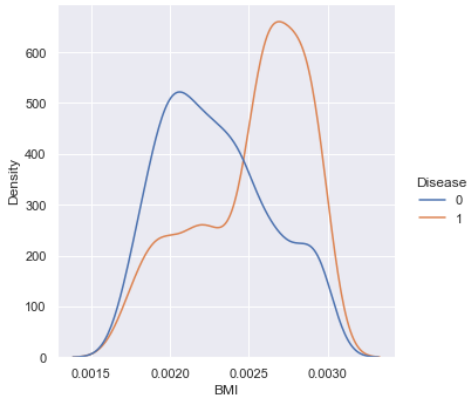


Figure 6. Visualization of the densities and frequencies of patients with and without the disease for New Features

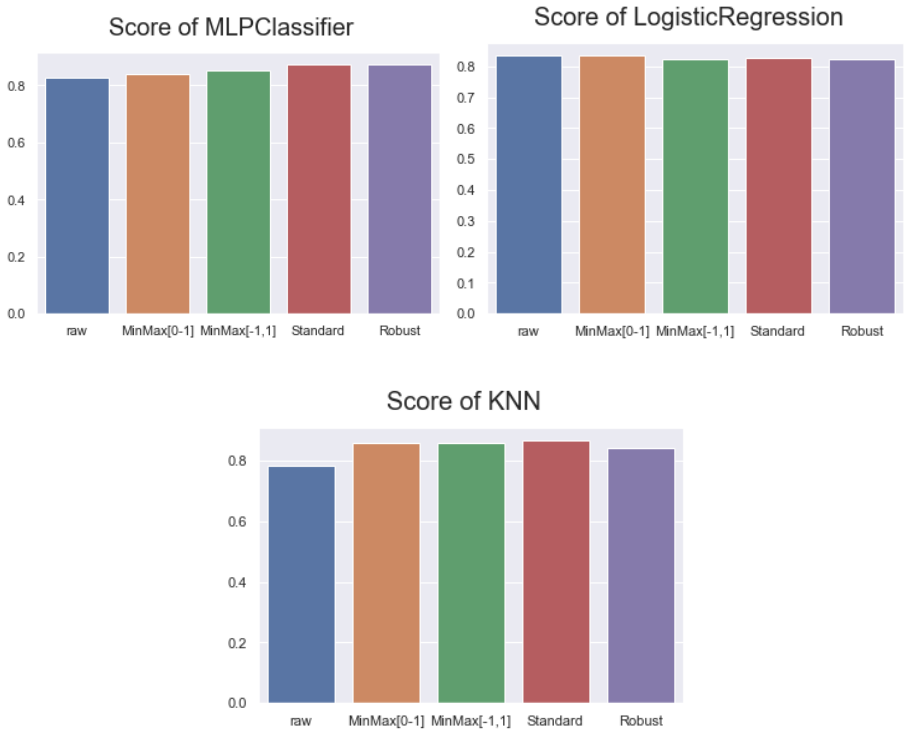


Figure 7. Comparison of different methods of scaling

Predictor	Kendall	RFE	Lasso	Decision
Age	Discard	Keep	Discard	Try with and without
Height	Discard	Keep?	Keep	Keep
Weight	Discard	Discard	Discard	Discard
High_Cholesterol	Keep	Keep?	Keep?	Try with and without
Blood_Pressure	Keep	Discard	Keep	Try with and without
Mental_Health	Keep	Keep	*Keep*	Keep
Physical_Health	Keep	Keep	*Keep*	Keep
IMC	Discard	Keep	Discard	Try with and Without

Table 2. Decision for Numerical Data

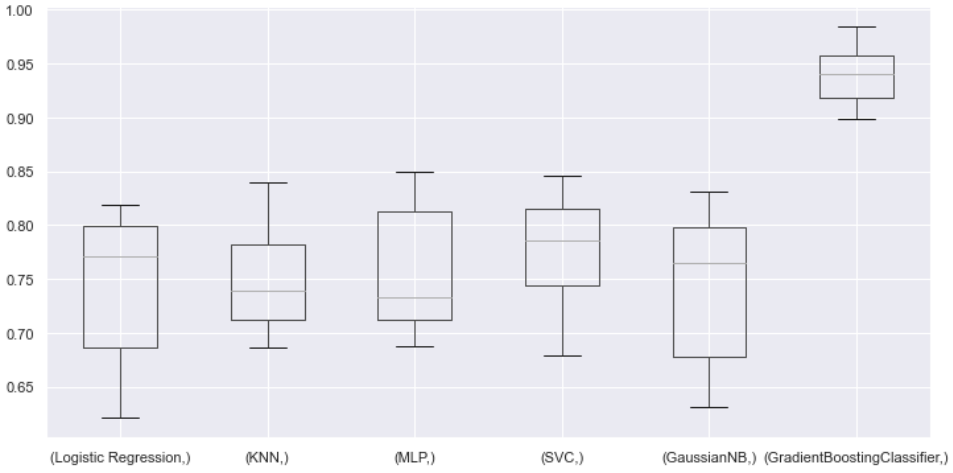


Figure 8. Graph of cross-validation scores of Numerical Data

Predictor	Chi-squared	Mutual Information	Decision
Checkup	Keep	Keep	Keep
Diabetes	Keep	Keep	Keep
Fruit_Extremes	Keep	Keep	Keep?
Fruit_Habit	Keep	Keep	Keep?
Exercise	Keep	Keep	Keep
Category_HC	Keep	Keep	Keep
Category_BP	Keep	Discard	Try with and Without
Female	Keep	Discard	Try with and without
Drinking_Habit	Keep	Discard	Try With and Without
Smoking_Habit	Discard	Discard	Discard
...	...	...	Discard

Table 3. Decision for Categorical Data

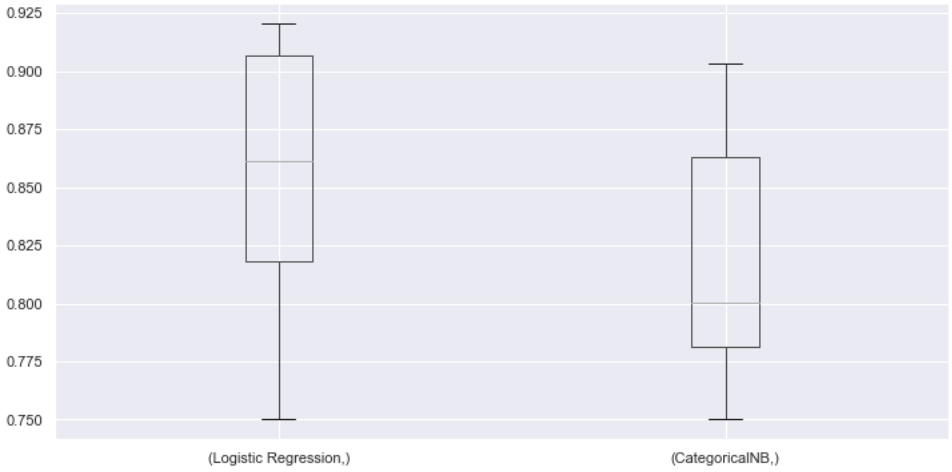


Figure 9. Graph of cross-validation scores of Categorical Data

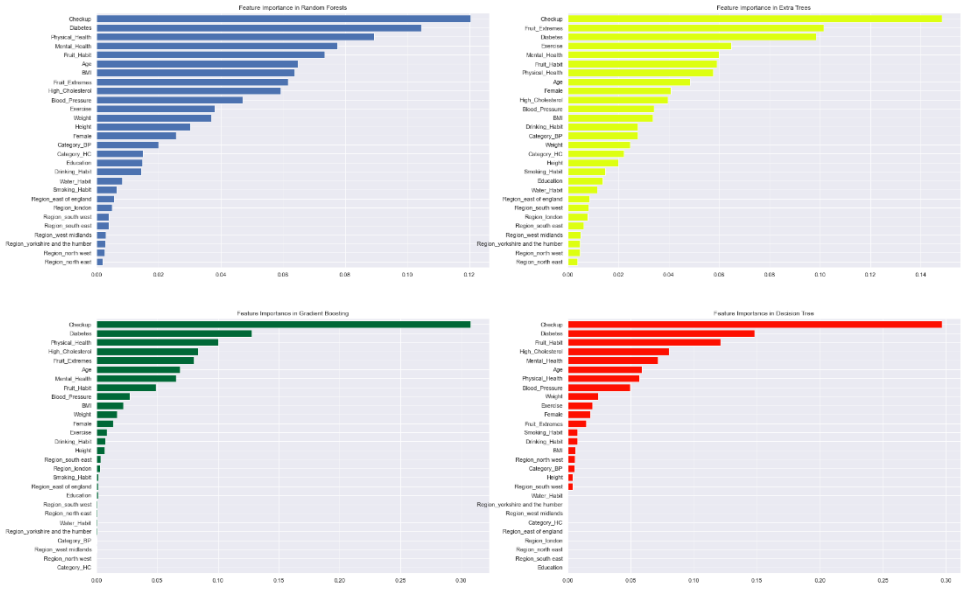


Figure 10. Features' importance

Predictor	Random Forest	Adaboost	Gradient Boosting	Decision Trees	Average Rank
Diabetes	2	2	2	2	2
Checkup	1	6	1	1	2.25
Physical_Health	3	1	3	8	3.75
High_Cholosterol	9	3	5	3	5
Fruit_Extremes	6	7	4	4	5.25
Mental Health	5	5	7	7	6
Fruit_Habit	4	8	8	6	6.5
Age	7	12	6	5	7.5
Female	14	4	10	10	9.5
Blood_Pressure	10	9	11	9	9.75
IMC	8	13	9	14	11
Exercise	11	11	12	12	11.5
Weight	12	17	13	11	13.25
Height	13	14	15	15	14.25
Drinking_Habit	16	10	14	18	14.5
...	...	...	...	...	Discard

Table 4. Features' Rankings

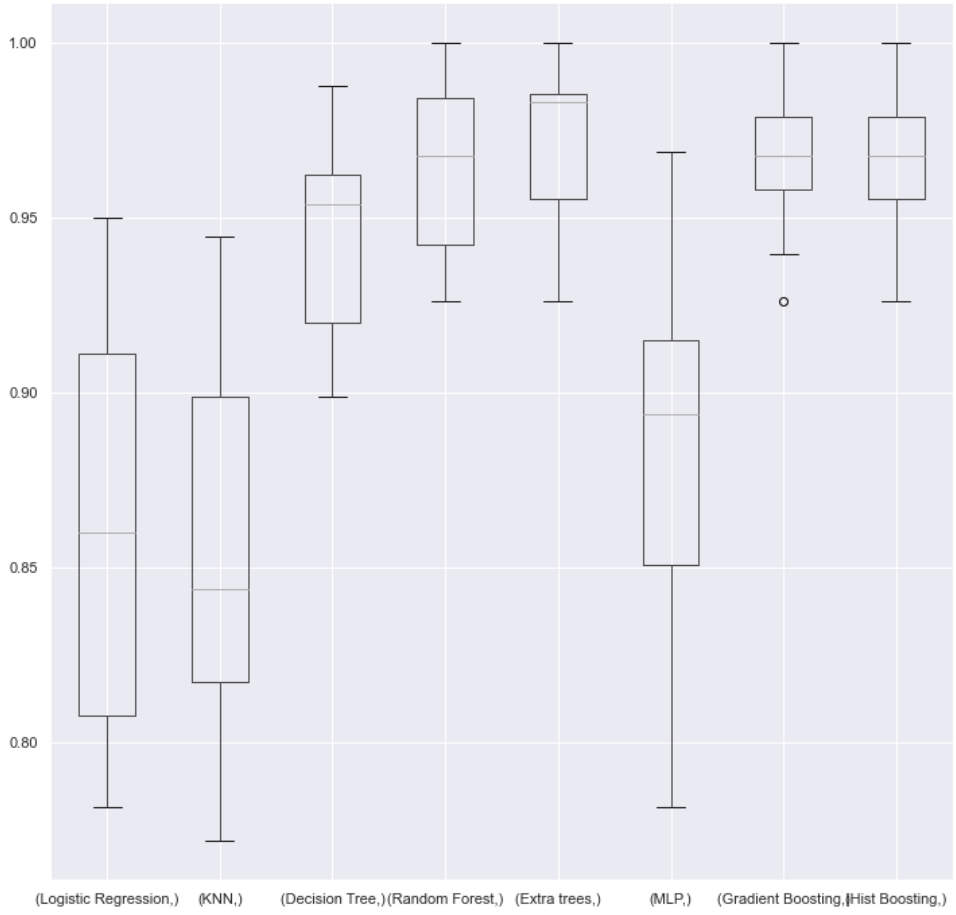


Figure 11. Graph of cross-validation scores of Categorical and Numerical Data

Figure 12. Pair plot

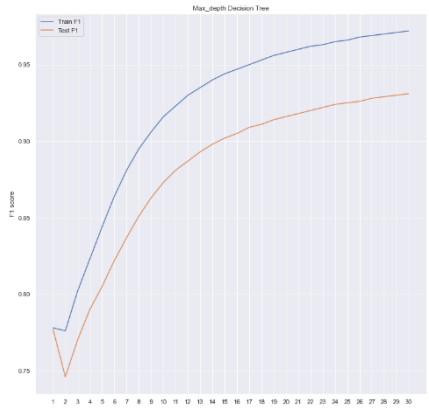


Figure 13. Max_depth in Decision Tree

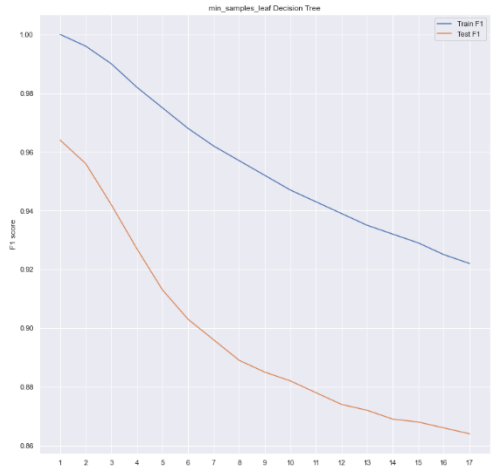


Figure 14. min_samples_leaf in Decision Tree

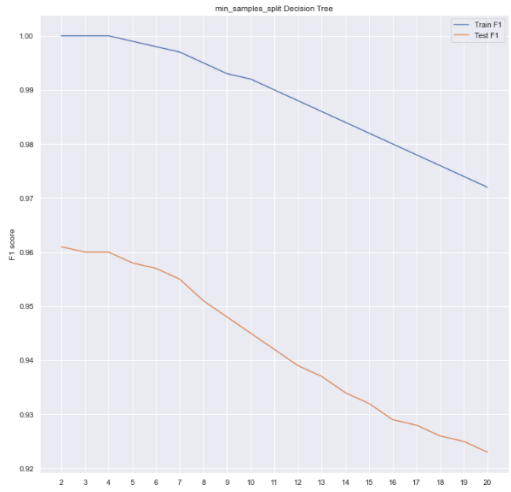


Figure 15. min_samples_split in Decision Tree

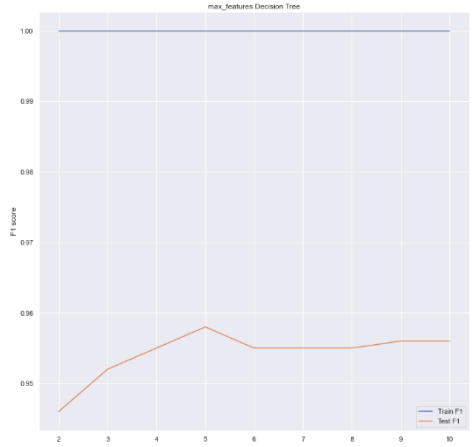


Figure 16. max_features in Decision Tree

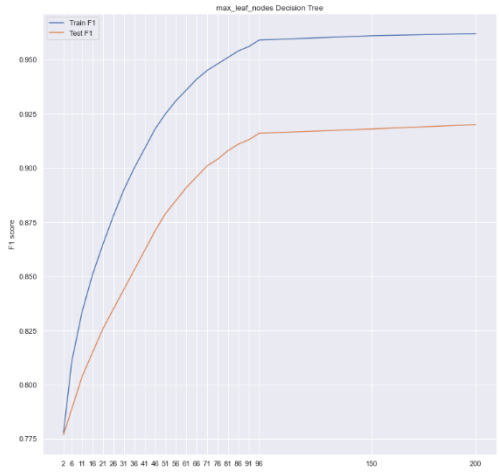


Figure 17. Max_leaf_Nodes in Decision Tree

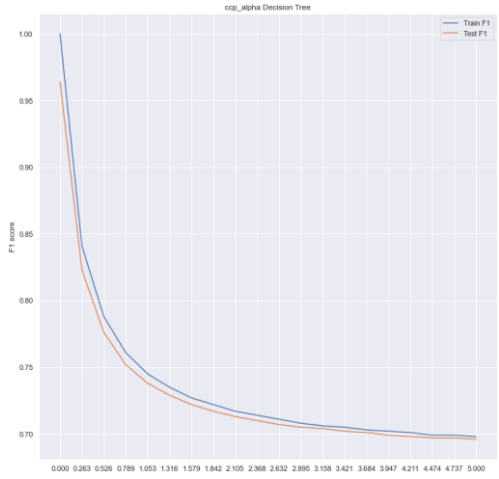


Figure 18. ccp_alpha in Decision Tree

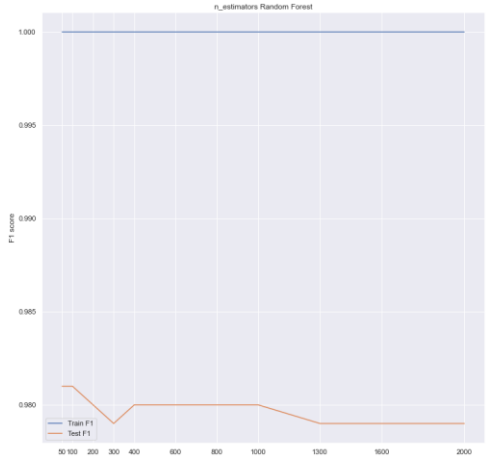


Figure 19. n_estimators in Random Forest

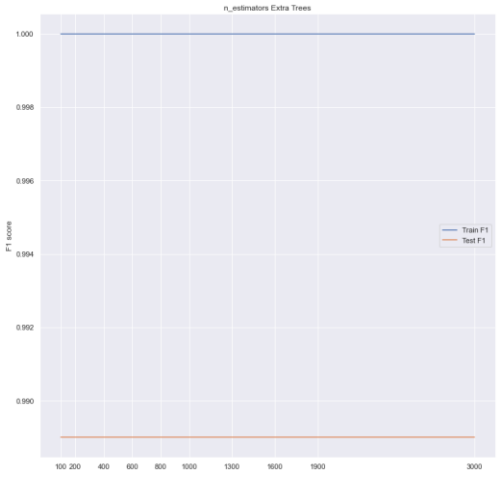


Figure 20. n_estimators in Extra Trees

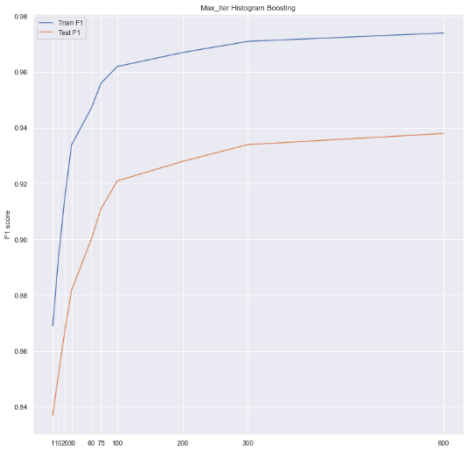


Figure 21. Max_iter in Histogram Boosting

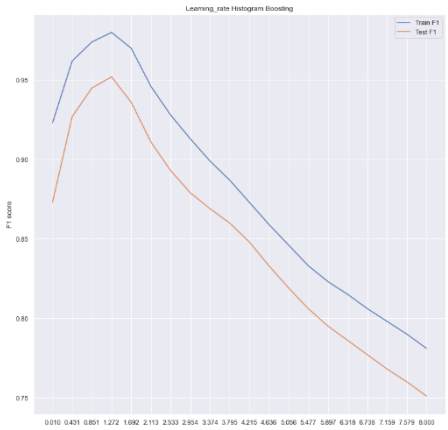
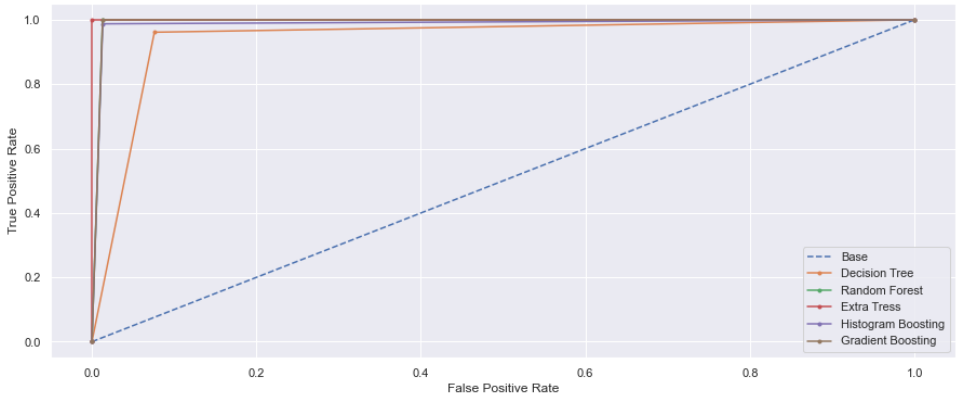


Figure 22. Learning Rate in Histogram Boosting




```
  AUC
0.943:       Decision Tree
0.994:       Random Forest
1.000:          Extra Tree
0.987:  Histogram Boosting
0.994:   Gradient Boosting
```

Figure 23. AUC Curve

	Min CV	Max Cv	Mean Cv	accuracy	precision	recall	f1-score
Extra trees Classifier	0.936508	1.000000	0.971249	1.000000	1.000000	1.000000	1.000000
Random Forest	0.936508	1.000000	0.966513	0.987261	0.987654	0.987654	0.987654
Histogram Boosting	0.920635	0.984127	0.955325	0.987261	0.987654	0.987654	0.987654
Gradient Boosting	0.935484	1.000000	0.968049	0.980892	0.975610	0.987654	0.981595
Decision Tree	0.887097	0.984127	0.947312	0.942675	0.950000	0.938272	0.944099

Table 5. Performance of best models

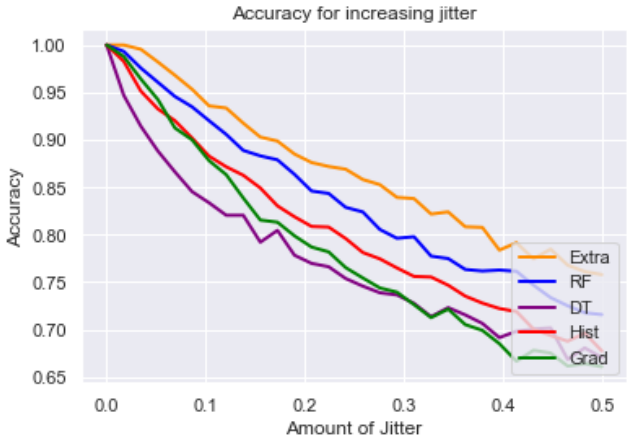


Figure 24. Accuracy of models with increasing jitter

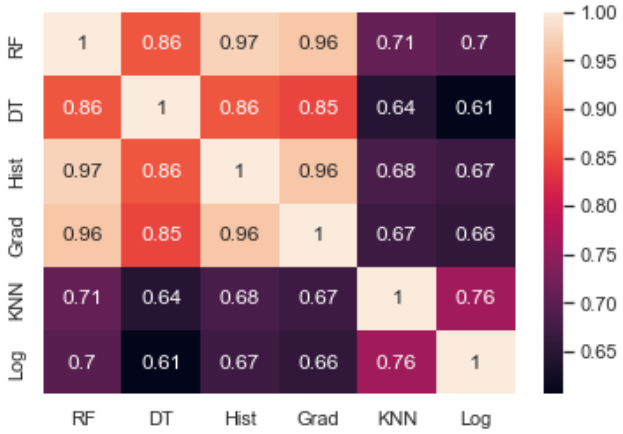


Figure 25. Heatmap showing correlation between models

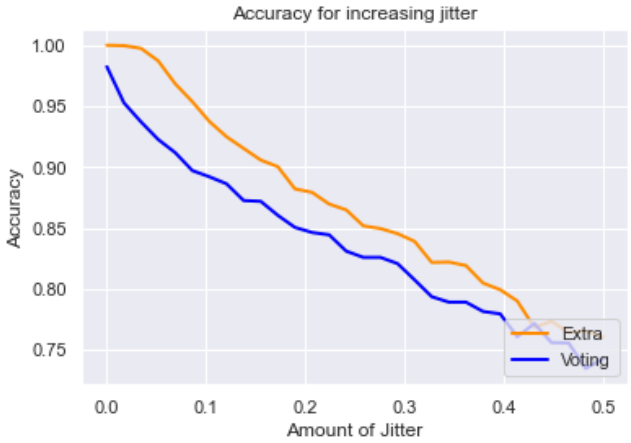


Figure 26. Accuracy of increasing jitter comparing Voting and Extra Trees Classifiers

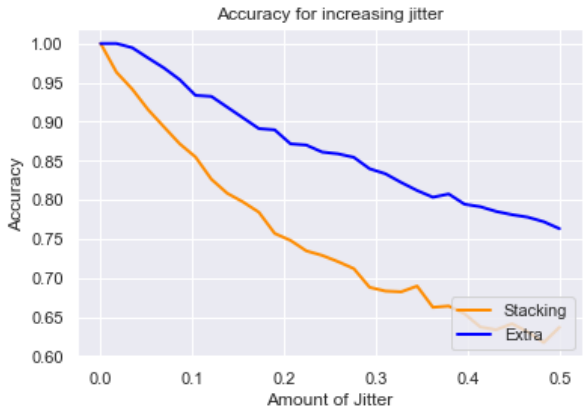


Figure 2727. Accuracy of increasing jitter comparing Stacking and Extra Trees Classifiers

	Min CV	Max Cv	Mean Cv	accuracy	precision	recall	f1-score
Extra trees	0.936508	1.000000	0.972862	1.000000	1.000000	1.000000	1.000000
Random Forest	0.920635	1.000000	0.964926	0.987261	0.987654	0.987654	0.987654
Histogram Boosting	0.920635	0.984127	0.955325	0.987261	0.987654	0.987654	0.987654
Voting	0.825397	0.967742	0.912238	0.974522	0.975309	0.975309	0.975309
Stacking	0.904762	0.984127	0.947261	0.961783	0.962963	0.962963	0.962963
Decision Tree	0.870968	0.984127	0.947158	0.942675	0.950000	0.938272	0.944099

Table 6. Comparison of performance of best models with ensemble models

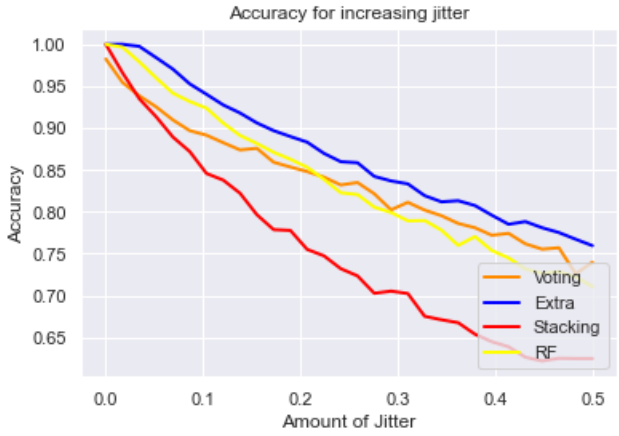


Figure 2828. Accuracy of increasing jitter in voting classifier

PCA

Principal Components Analysis (PCA) [6] is a dimensionality reduction technique. It can be especially useful when there are a lot of variables to consider in the feature selection process, although it can also be useful when using decision tree models, as this method transforms the variables into orthogonal new variables. This way, linear functions of the best partition in a decision tree can be transformed into a horizontal or vertical line. As mentioned above, this technique forms new uncorrelated variables called Principal Components (PC), which are linear composites of the original ones. They can be thought of as the variables corresponding to the best fitting line that goes through the origin, perpendicular to the other Principal Components.

PCA should be done with standardized variables, as it is important that all the variables have the same weight when creating the new components (otherwise variables with lower variance would be penalized in relation to the ones with higher variance). Moreover, the variables should be numeric, as this technique is reliant on the variance of the variable using distance measures between points, which is not a characteristic present in categorical variables (at least not in the same way).

Firstly, the line that retains the maximum variance is computed, which is equivalent to running a simple regression (ordinary least squares) on the data. This will be the first principal component. The others are computed by getting orthogonal projections of the first one, until the n dimensions are reached. These principal components are calculated in descending order of the variance they can account for in the data.

Where accounts for the maximum variance in the data, accounts for the maximum variance that has not been accounted for by the and so on. From the equations above, the form the eigenvectors, which are vectors with the weights of each variable. It is also possible to derive the eigenvalues: the variance of each Principal Component (Sum of Squared distances between projected points and the origin).

An important issue in the PCA is the number of components to extract. There are several possible methods, but all of them take into account how much information can be sacrificed, i.e., unaccounted variance. The method used in this project was the **Scree plot**. It consists of plotting the percent of the variance accounted for each PC and choose the variables until the first big elbow. The rationale behind it is that after the drop stops decreasing dramatically, the gain in explained variance from including another principal component does not justify the handling of a new variable.

After computing the Principal Components and their suitable number, it is necessary to assign each variable to the new variable that is the most suited to it. For that, the loadings are used. They are the correlation between each PC and the original variable, which means they show the extent to which the variable is important to the PC. The higher the correlation, in absolute terms, the more important it is to explain the PC. If the variable has a lower loading than 0.5 for all the components, then either more components need to be added or variables need to be deleted, because it means that that variable is not being represented by the current results.

Mutual Information

During the Feature Selection process, one way of deciding which variables should be removed, is by checking the how much information (in terms of entropy) two random variables share, and thus being better at explaining the target. Mutual information [7] is one of such measures, with the advantage of being able to detect non-linear dependencies between variables. The formula is given by:

$$I(x_i, y) = \int_{x_i} \int_{y} p(x_i, y)\, log \frac{p(x_i, y)}{p(x_i)\, p(y)}\, dxdy$$

The *sklearn* [8] version used takes as input discrete variables (can either be categorical, discretized variables or equivalent) and relies on entropy estimation from k-nearest neighbour distances. According to this measure, higher mutual information indicates a large reduction in uncertainty, i.e., there is a relationship between the variables. If the mutual information is zero, the variables are independent. It is worth mentioning that the units of this measure are the same as the variables.

Jitter Test

Since it is impossible to know the true data generating process, it is naturally improbable that the model created conceives the right assumptions about the data, capturing the parameter perfectly. Robustness testing explores if the baseline model (the one created by the researcher) holds when the assumptions are tested. This can be done by replacing the model specifications by plausible alternatives systematically, then observe how the model behaves. The baseline model's estimates are supported if there are higher levels of robustness, while low levels of robustness show less confidence in the model's predictions.

The Jitter Test is a method used to test the robustness of a model. It does so by adding noise to the data and see how the performance of the model changes with the increase in the magnitude of the noise. In this case, a normal distributed error was created, even though other distributions were possible. Then, the model is run by adding the randomly generated error to the independent variables, increasing standard deviations by each iteration. In the end, the results were plotted so that the accuracy of the models could be compared with the evolution of the jitter in the data. This test mainly checks whether the model can handle observations that are not in the same general pattern, or if a small change in the data will have a significant impact on the predictions that it produces. In the latter case, it can be a sign of over-fitting, as the model is not learning the pattern of the data, but simply memorizing it.

Histogram Gradient Boosting

In order to speed up the Gradient Boosting process, the number of input values can be reduced by binning the continuous variables, i.e., aggregate the values that at in a certain interval which can be defined in many ways. This way, the decision tree will work on the bins and not on every single value, accelerating the process. This type of Gradient Boosting is called Histogram based.

ExtraTrees

ExtraTrees [9] are similar to Random Forests, as they are an ensemble [10] approach to Machine Learning that will train multiple decision trees and join the outputs from them to make a prediction. The difference between Random Forests and ExtraTrees is that Random Forests, to do the multiple decision trees, will perform bagging [11] to make sure the decision trees are different from each other. ExtraTrees however uses all of the values of a dataset to train the decision trees by default, so to make sure the decision trees are different, it does not greedy search to decide where to split a feature. Instead, ExtraTrees randomly select a value to split a feature to create child nodes.

While using the whole dataset can reduce bias, randomly selecting where to split can increase bias and variance, since it most likely will include features that are irrelevant, so doing a good feature selection is very important when using ExtraTrees.

Voting Classifier

Voting Classifier [12] is an ensemble model that combines the predictions of multiple base models to make a final prediction, with the goal of improving the performance of the model. It can combine the predictions by hard voting, which means the most frequent value is the final one or by soft voting, which uses the average of the predictions of the base models.

Voting can be useful to improve the performance when the models used as base are diverse and complementary to each other. However, if the models are very similar, the voting classifier does not actually improve the performance of the model. So, the performance of the voting classifier must be evaluated carefully to make sure it is improving the performance.