NOVA IMS - INFORMATION MANAGEMENT SCHOOL
Master's Degree in Data Science and Advanced Analytics

Group Project
The Smith Parasite

COURSE
Machine Learning 2022/2023

PROFESSORS
Roberto Henriques
Carina Albuquerque
Ricardo Santos

GROUP 32
20220593 Jaime Kiyoshi Kuei
20221750 Luca Frias Loureiro
20220621 Lucas Emilio Mendes Ferreira
20221381 Maria Arbelaez
20210581 Miguel Vaz Ramos

1. Introduction

The Smith Parasite is a new disease that has recently emerged in England, affecting over 5000 individuals. The main objective of this project is to build a predictive model that can accurately identify which individuals are more likely to suffer from this disease. To accomplish this task, we will have access to the demographic, behavioural, and health data from healthy and infected patients. We will analyse and transform the data to train various machine learning models, documenting all the steps taken in the process, including data exploration, pre-processing, modelling, and assessment. The performance of the models will be evaluated using the F1 score, which is a metric that combines precision and recall.

2. Exploration

Before beginning the data exploration, the first step was to import all of the demographic, behavioural, and health data from the training and test sets, verify that they were composed of the same number of rows and “Patient IDs”, and only then did we concatenate the data from the test set and the training set. We then moved on to separating the numerical and categorical features based on the data type of the values in each column.

2.1. Categorical Features Exploration

Checking for missing values and null values on the many observations that make up the training set was the first stage in our categorical feature exploration process. The feature “Education” has 13 missing values identified, but no null values were found. Then, because there was the possibility of the same person having two different “Patient IDs”, we searched for duplicated names. In fact, we discovered a repeated name, “Mr. Gary Miller”, but when we examined the other traits, we were able to quickly determine that it was a different person.

The next step was to check the values in the “Disease” column to see if the dataset was balanced for training or if we needed to rebalance it. The dataset was already balanced for our analysis.

The subsequent phase was to investigate the unique values that each of our categorical features from the training set could take. This process entailed not only examining the representation of each class within each categorical feature, but also utilizing our custom volumetry function[1] to visualize the prevalence of the virus in each class within the features. Upon examining the classes each feature can take, we noticed several issues that will need to be addressed during the pre-processing. Specially, we observed that London is written in two different ways in the “Region” feature, and there is a low number of one of the classes in the “Checkup”, “Drinking_Habit”, and “Fruit_Habit” features. Upon examining the volumetry plots[2], it is clear that the following categorical features: "Checkup", "Fruit_Habit", "Drinking_Habit", "Exercise", and "Diabetes" will be good indicators of whether or not a person has the disease. This is because the proportion of people with the disease changes significantly based on the class of these features. On the other hand, for "Education," "Smoking_Habit", "Water_Habit", and "Region," the proportion of people with the disease does not vary significantly based on the class of these features.

Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

2.2. Numerical Features Exploration

No missing values, nor null values were identified on the numerical features that make up the training set. We analysed central tendency metrics[3] grouped by the target variable to determine if there were differences between the two groups. We found several features that seemed to differentiate the groups, including "Mental_Health", "Physical_Health", "High_Cholesterol", "Weight", and "Birth_Year". However, we caution that the data has not yet been cleaned of outliers, which could affect the accuracy of these findings.

To gain a deeper understanding of our numerical features, we created a function[4] that generates both a box plot and a histogram for each feature. These plots[5] enable us to see the distribution and range of values for each feature, as well as the prevalence of the disease or the absence of it in each bar of the bar plot. The box plots show that "Physical_Health", "Birth_Year", and “Blood_Pressure” have the most outliers, while the other features have a more even distribution of values. Based on the histograms, we can conclude that a young person who weighs more, has lower cholesterol levels, and experiences no difficulty walking but has mental health issues is more likely to have the disease. It is not possible to draw conclusions about the relationship between the disease and blood pressure or height based on this first analysis. Furthermore, a pairwise scatter plot[6] was plotted in order to identify bivariate outliers. During the pre-processing, we will address the outliers mentioned throughout this analyses. The correlation matrix was not analysed in this instance, as it will be further examined during the feature selection process using the filter method.

3. Pre-Processing

3.1. Categorical Feature Pre-Processing

As the first step in pre-processing the categorical features, we standardized the spelling of "London" in the "Region" feature to ensure that it refers to the same city. We then filled the missing values in the "Education" feature with "I never attended school / Other" since there was no discernible pattern among the observations with missing values, aside from the fact that all of these observations were from patients who were over 44 years old. Subsequently, on the feature “Checkup” we decided to group the class “Less than 3 months” with the class “Less than 3 year but more than 1 year”, as their volumetry is very similar and the former is almost not represented. On the other hand, and although “Fruit_Habit” and “Drinking_Habit” have a class that is almost not represented, we did not group these classes with others because their volumetry was significantly different.

Embarking on feature engineering[7], based on the formal title present in the “Name” column of each patient, we were able to create a feature called “Gender” that identifies if a certain patient is Male or Female. We were then able to remove the “Name” feature, as it was not adding any value to our analysis. Additionally, we decided to group the nine different regions listed in the "Region" feature into six broad categories: South, West, East, North, London, and Yorkshire and the Humber creating the feature “Region_Grouped”. This allowed us to simplify and streamline our analysis. By using our custom volumetry function we create the volumetry plots[8] and through their analysis, we found out that the

Additional notes on the appendix,
counting towards “Creativity and Other Self-Study”.

“Gender” feature will be a good predictor, as the proportion of people with the disease varies significantly based on their gender. However, we did not observe a significant difference in the proportion of people with the disease based on the “Region_Grouped” feature. We applied the feature engineering and grouping techniques to the test set, as these techniques do not require us to know the data to be effective, and have to be in consensus with the training set.

3.2. Numerical Feature Pre-Processing

As our first step in feature pre-processing, we applied feature engineering to create a new column called "Age" based on the "Birth_Year" column. We then dropped the "Birth_Year" column because it provided the same information as the newly created "Age" column. Additionally, we created a new column called "BMI" (Body Mass Index) based on the "Height" and "Weight" columns.

The second step in our process was to create a function that removes outliers from our training data set. We experimented with several thresholds and different methods, but the one that yielded the best results was the one that considered any value that distances +/- 1.5 IQR from the 1st and 3rd quantiles of a specific numeric feature to be an outlier. Using this method, we were able to remove the outliers that we identified through the boxplots and pairwise relationship plots while exploring the numerical features. This resulted in retaining 92% of the training set, which is a good percentage given the training set size. After using the previously mentioned function to remove outliers, we plotted[9] the box plot and histogram of each numerical feature again. The conclusions are similar to what we found during the initial exploration, but this time the box plots and histograms are more clear. The bivariate outliers present on the pairwise scatter plot[10] also decreased significantly. Additionally, we can see that the newly created feature "BMI" may be a good predictor, as individuals with higher BMIs tend to have a higher probability of having the disease. We applied feature engineering to the test set because this technique does not require knowledge of the data. However, we cannot remove outliers from the test set because doing so would prevent us from making predictions on the entire test set.

3.3. Methods for Feature Selection

From the previous exploration, we have identified several key features that we believe will be important to use as predictors. These include a number of categorical features such as "Checkup", "Fruit_Habit", "Drinking_Habit", "Exercise", and "Diabetes," as well as some numerical features including "Weight", "High_Cholesterol", "Mental_Health", "Physical_Health", "BMI", and "Age”. We will use feature selection methods to determine which of these identified features are the best predictors.

By applying the chi-squared independence test as a filter method to the categorical features, we found that the most important features are "Checkup", "Diabetes", "Drinking_Habit", "Exercise", "Fruit_Habit", and "Gender". Using the Spearman correlation matrix[11] as a filter method, which measures the strength and direction of a monotonic relationship between two variables, we found that the numerical variables that are most correlated with the "Disease" variable are "Mental_Health", "Physical_Health", "Age", and "BMI". Although "Height" had a strong correlation with "Disease," we decided to exclude it because it also had a high correlation with "BMI" and we wanted to avoid overfitting our model. We used Pearson correlation to measure the linear relationship between each pair of

Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

variables. These selected features will be referred to as "df_train_final_filter" in the subsequent steps of the analysis.

For the wrapper method, we decided to use a more sophisticated method than the usual recursive feature elimination because we found it to be too unstable. In each iteration, it suggested a different number of features. We came up with two different methods, the first one was a for-loop with recursive feature elimination[12], and the second one was a recursive feature elimination with cross validation[13]. The conclusion of both was to include the following numerical features: “Height”, “High_Cholesterol”, “Blood_Pressure”, “Mental_Health”, and “Physical_Health”. These added to the categorical features obtained from the filter method will be the ones referred to as “df_train_final_wrapper”. For the embedded method, we chose to use Lasso Regression[14], which increases the coefficient of a feature as it becomes more important. This method identified the following numerical features as the most important: "Mental_Health," "Physical_Health," "Height," and "High_Cholesterol." These, along with the categorical features identified by the filter method, will be referred to as "df_train_final_embedded" in the subsequent steps.

4. Modelling

In this section, we applied a range of models to the training sets that were generated during the feature selection process. We carefully evaluated each model based on its strengths and weaknesses, as well as how well it would fit our data. We considered factors such as the size of the dataset, the presence of outliers, the balance of the target variable, and whether we needed interpretable results. In addition to the models, we will also utilize scaling techniques, one hot encoding, and the ability to remove outliers on these training sets. To assess the performance of these techniques, we will use a function[15] that calculates the F1 score, using cross-validation.

4.1. Logistic Regression Classifier

After testing multiple training datasets, we found that the one created using wrapper methods and applying minimum maximum scaling (-1,1) without removing outliers performed the best when used with a default logistic regression classifier. This combination resulted in an F1 Train Score of 0.879 and an F1 Validation Score of 0.873, using the function described previously. Our results were surprising because we did not expect that a training dataset with outliers included would perform better than one without outliers, given that logistic regression is sensitive to outliers and can be negatively impacted by them.

4.2. Multinomial Naïve Bayes Classifier

The multinomial naïve bayes classifier estimates the probability of each feature occurring in each class. The classifier then determines the likelihood that a new data point belongs to each class based on the likelihoods that its features appear in each class when the classifier is presented with a new data point. The classifier then determines which class has the highest likelihood of being the class of the data point. Because it presumes that the features are independent of one another, which is

typically false in real-world data, it is known as "Naïve Bayes." We selected this model as it is simple, effective, and an efficient tool for classification. After testing multiple datasets, we found out that the dataset that included all available features (except those removed during pre-processing) and applied minimum-maximum scaling (0,1), performed the best when using a default multinomial Naïve Bayes Classifier. This dataset also included the removal of outliers. By using the combination of this dataset and the model, we got an F1 Train Score of 0.858 and F1 Validation Score of 0.853, using the F1 Function described previously.

4.3. K-Nearest Neighbours Classifier

K-Nearest Neighbours is a simple algorithm that is used to classify new cases based on similarity to previously stored cases. It is effective for high-dimensional data but can be computationally expensive and sensitive to the presence of outliers. We ran a grid search with pipeline[16] on all possible combinations of datasets[17] (feature selection method, scaling, and the presence or not of outliers), and using our F1 Score function we found that our best KNN model had an F1 Train Score of 1.0 and F1 Validation Score of 0.98. The dataset used to get this result was the one containing the features resulting from the filter method with robust scaling, without the removal of outliers. The model used has the following hyperparameters: n_neighbours=5, weights="distance", algorithm="ball_tree", and metric="manhattan". Although getting a high score in this case, it is important to note that KNN is a simple algorithm and may not always produce the most accurate results. More complex algorithms may be more effective and reliable, especially in the field of epidemiology.

4.4. Multi-Layer Perceptron Classifier

An MLP, or multi-layer perceptron, is a type of neural network used for classification tasks. It consists of multiple layers of interconnected nodes that process input data through nonlinear activation functions to produce output. While MLPs are generally effective for large datasets and complex non-linear problems, they can be challenging to tune and may overfit easily when working with small datasets and outliers. In our case, we are using a small dataset, so we need to take steps to reduce the risk of overfitting, such as using fewer hidden layers and neurons, usage of scalers, and employing cross-validation to improve the robustness of our model's performance. While interpretability may not be a concern for us, optimizing the f1 score is important, which makes the use of an MLP classifier a suitable choice for our needs. After evaluating various training datasets, we found that the one created using filtering methods and applying robust scaling without removing outliers resulted in the best performance when using a default MLP classifier with one hidden layer and 12 neurons. It is worth noting that despite not using the IQR method to remove outliers, the robust scaler can handle them effectively. To optimize the model's performance, we employed a technique using pipeline and grid search as previously described. The model that achieved the highest F1 Score had the following hyperparameters: hidden_layer_sizes=17, alpha=0.05, learning_rate='constant', activation='tanh', and solver='lbfgs'. With these hyperparameters and the training set described above, we obtained an F1 Train Score of 1.00 and an F1 Validation Score of 0.985 using the previously mentioned F1 function.

4.5. Decision Tree Classifier

Additional notes on the appendix,
counting towards "Creativity and Other
Self-Study".

We chose to test with Decision Trees, as it is a model easy to understand with a lot of interpretability. Normally, it doesn't need much pre-processing of the data, because it is not sensible to scaling nor missing values. On the other hand, it is a model prone to overfitting and sensible to outliers. After the initial experimentation, we decided to test multiple training sets, and we found that the one created using the wrapper data set without removing outliers is the one that performs the best when using a default decision tree classifier. To optimize model performance we used grid search combined with the wrapped dataset to find the best combination of hyperparameters for training. The model which delivered the highest F1 Score was the one with the following hyperparameters: ccp_alpha=0, max_depth=None, max_features=10, min_samples_leaf=1, min_samples_split=2, splitter=random. Interestingly, apart from the splitter and the maximum features, these are the parameters of the default decision tree classifier, without any type of pruning. Using these hyperparameters, combined with the training dataset described above we got an F1 Train Score of 1.0 and an F1 Validation Score of 0.974, using the F1 Function described previously. In order to maximize the F1 score, we allowed the decision tree to fully grow. However, if we wanted to gain a more detailed understanding of the data, we could have limited the depth of the tree to 2, as demonstrated in the annexes[18].

4.6. Random Forest Classifier

Random forest is an ensemble meta-estimator that fits decision trees each on random subsets of the original dataset and then aggregate their individual performance to form a final prediction. We want to induce high variability in the base classifiers, because this will allow the ensemble model to capture a wider range of patterns and trends in the data. To build a random forest model, it is generally recommended to use bootstrap samples that are the same size as the training set, to select the square root of the total number of features, to include a large number of base classifiers and to allow each tree to grow to its maximum size. These will help to increase the variance and reduce the correlation between each one of the decision trees. After testing multiple training datasets, we found that the one created using filter methods and applying standard scaling, without removing outliers performed the best when using a default random forest classifier. To optimize model performance, we used the previously described pipeline combined with grid search cross validation to find the best combination of hyperparameters for training. The model which delivered the highest F1 Score was the one with the following hyperparameters: bootstrap=False, max_depth=None, max_features=1, max_samples=None, n_estimators=50. Using these hyperparameters, combined with the training dataset described above we got an F1 Train Score of 1.00 and an F1 Validation Score of 0.989, using the F1 Function described previously. These weren't the results we were expecting as what we got was 50 decision trees, each with a single decision node based on the value of one features, which likely increased variance between the different decision trees. Nonetheless, this was the most effective solution for the training dataset described above. One disadvantage of this model is that it may be less interpretable due to the use of multiple decision trees.

4.7. Bagging with K-Neighbours Classifier (K=1)

Bagging with K-Neighbours Classifier is also an ensemble meta-estimator but the difference to Random Forest Classifier is that instead of using as the base learner Decision Trees, it uses other weak

Additional notes on the appendix,
counting towards "Creativity and Other
Self-Study".

learners, which in our case is the K-Neighbours Classifier (K=1). In bagging, to increase the variance of each base classifier, it is generally recommended to use bootstrap samples that are smaller than the full training sample, include all available features, use a large number of base classifier, and set the value of K to a low number. After testing multiple training datasets, we found that the one that created using filter methods and applying robust scaling, without removing outliers performed the best when using a default bagging classifier with k-neighbours classifier (k=1) as the estimator. To optimize model performance, the technique with pipeline and grid search described previously was used. The model which delivered the highest F1 Score was the one with the following hyperparameters: bootstrap=False, max_features=0.8, max_samples=0.8, n_estimators=300. Using these hyperparameters, combined with the training set described above we got an F1 Train Score of 1.00 and an F1 Validation Score of 0.978, using the F1 Function described previously. Our results were in line with our theoretical expectations, with the exception of not using bootstrapping and omitting some features. Using 80% of the training set and features in each base classifier increases the risk of data leakage.

4.8. Gradient Boosting Classifier

Gradient Boosting is an ensemble algorithm that works by training a series of weak models, each of which tries to correct the mistakes made by the previous model. It is resistant to overfitting, but can be computationally expensive. We used the grid search with pipeline on different combinations of data sets[19] (feature selection method, scaling, and presence or not of outliers) and found that our best model had an F1 Test Score of 1.00 and an F1 Validation Score of 0.987. The dataset used to get this result was the one containing the features resulting from the embedded method with no scaling nor remotion of outliers. The model used had the following hyperparameters: learning_rate=0.5, n_estimators=100, subsample=1.0, max_features=None.

4.9. Stacking and Voting Classifier

Stacking is a technique used to create a powerful model by combining multiple models. Each model is trained separately, and their predictions are combined and trained in another estimator to make a final prediction, that in our case is a Logistic Regression. This approach can improve performance by allowing the individual models to give complementary information and avoid overfitting, which makes the use of a Stacking classifier a suitable choice for our needs. Our best three models in the previous session were: Random Forest, MLP Classifier and Bagging with KNN, combining them with Stacking classifier and set the last estimator with a logistic regression we achieved a F1 Train Score of 1.00 and a F1 Validation Score of 0.988 using the previously mentioned F1 function. We also use another ensemble method called Voting classifier explained in the appendix[20], with this classifier we achieved an F1 Train Score of 1.00 and an F1 Validation Score of 0.991.

5. Model Assessment

The true positive rate (TPR) and false positive rate (FPR) are important to consider when evaluating classifier performance in epidemiology diagnostics. A high TPR indicates accurate identification of positive cases, while a low FPR minimizes false positives and unnecessary testing and treatment. The F1 Score is a measure that considers both precision and recall, where precision is the

Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

ratio of true positive predictions to all positive predictions, and recall is the ratio of true positive predictions to all positive cases. While it can be helpful, the F1 Score may not always be the ideal option because it does not demonstrate the TPR and FPR balance as clearly as the ROC curve and AUC do. As a result, we will consider multiple performance measures when analysing the different classifiers we tested throughout the report.

To evaluate the performance of our various models presented in 4. Modelling, we used our F1 Function. The results of our analysis are detailed in the individual explanations of each model and are also depicted in a in bar chart[21] as annexes. As expected, the model that performed the best according to the F1 Score was the Voting Classifier, a predictive model not covered in class. When we applied the F1 Score function to this model, we obtained an F1 Score Test of 1.0 and an F1 Score Validation of 0.991. Since the F1 Score is an average, it is not possible to calculate the confusion matrix or precision and recall. To resolve this issue and generate the ROC curve and AUC area, we performed a simple train-test split, using 80% of the data for training the models and 20% of the data as a validation set. We then plotted the results in a table[22] and in a bar chart[23]. The table shows the F1 scores obtained using the 80% of data in the F1 cross-validation function (left) and the simple F1 score of the 20% validation data (right). In this case, the MLP model had the highest F1 validation score of 0.988. The table also includes the precision, recall, and confusion matrix. The best models almost perfectly predict the outcomes, except for two false positives. We plotted the ROC curve[24] using the results and found that several models had an AUC (Area Under the Curve) of 1 when rounded to the second decimal place, indicating that they are highly accurate at predicting true positives and do not have a significant error rate for false positives.

Despite the fact that the MPL model performed best when using a train-test split, we only used 80% of the data for training in that case. However, when we used all of the data for training, as we previously mentioned, the voting model performed the best. Therefore, we will use the voting model to make our final predictions and submit them to Kaggle.

6. Conclusion

According to the results of our top-performing models and the analysis of categorical and numerical data during the exploration process, the profile of an individual most likely to suffer from the Smith Parasite is as follows: Women between the ages of 45 and 65 with Obesity Class I (as defined by WHO guidelines) and diabetes (including borderline, pregnancy-related, or with a relative with diabetes), low fruit consumption, medium-high alcohol consumption (either as social drinkers or daily drinkers), and a lack of exercise for more than 30 minutes at least 3 times per week. Additionally, these women have reported poor mental health for over 15 days in the past 30 days, but have not experienced physical health issues that made it difficult for them to walk. It is possible that the ability of these women to walk and leave their houses may increase their exposure to potential contagious diseases. However, this is just a hypothesis that requires further investigation.

Overall, our efforts to achieve the objectives outlined in the introduction have been successful, as demonstrated by the high F1 Score of 0.991 obtained using the Voting Classifier. This model was able to accurately predict whether a patient has the disease in question.

Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

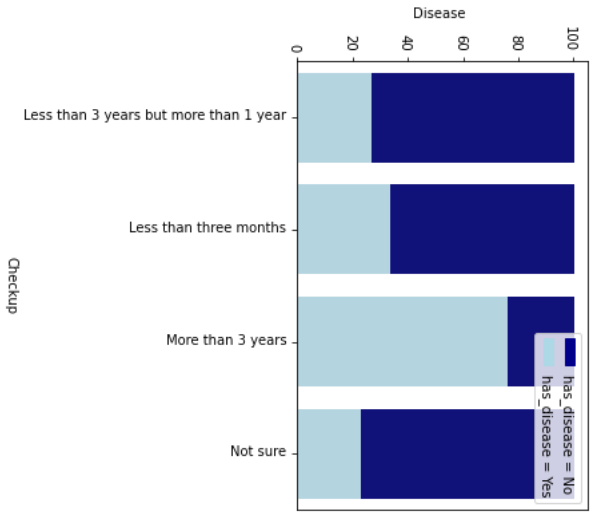
Appendix

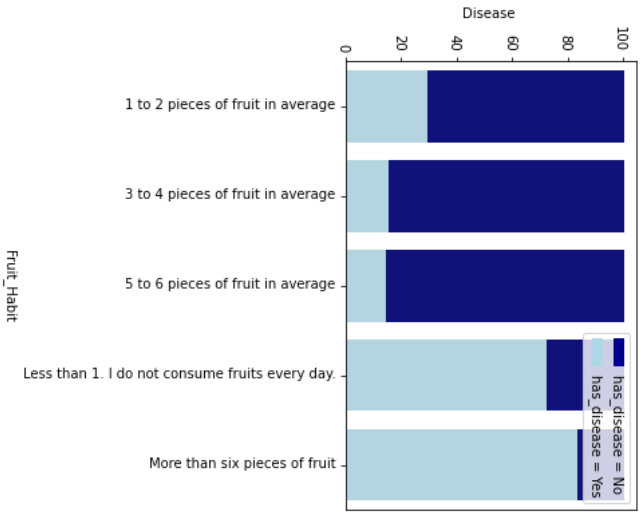
2.1. Categorical Feature Exploration

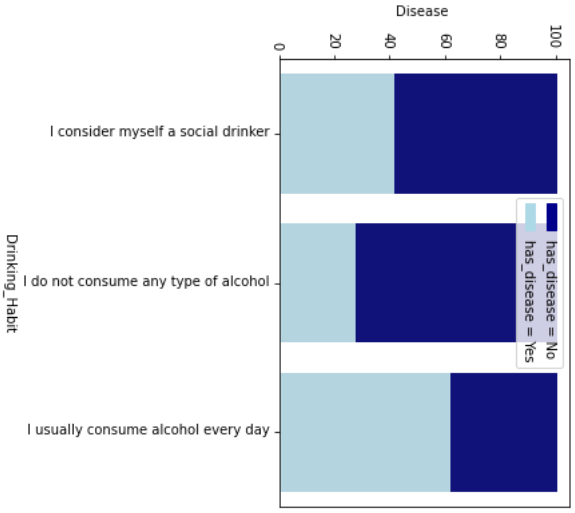
Point 1. Custom Volumetry Function:

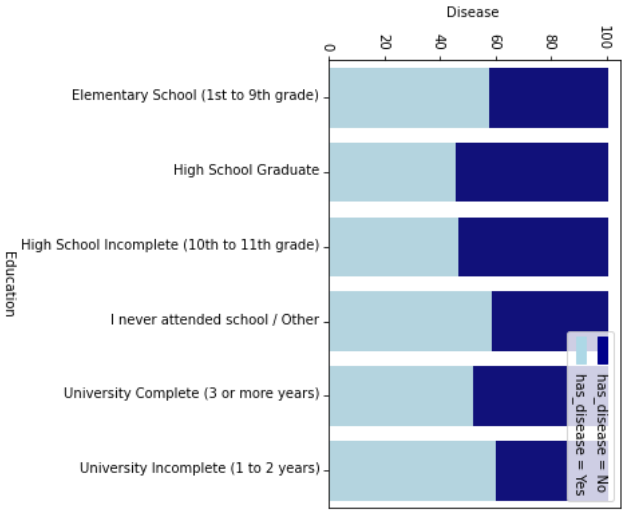
In order to help us have a better understanding of the distribution in our categorical features, we developed a function (plot_categorical_volumetry) that, given the dataset, the target variable (disease) and the variable name, we plot graphics representing the proportion between diseased and non-diseased people. For each of the possible values inside the feature, we scale the values for a percentage value and make boxplots to show the difference between our targets. The goal in these plots is to see if a determined characteristic has a different proportion from the others, so we can identify if the feature is significant to our classification.

Point 2. Volumetry Plots for each of the Categorical Features:

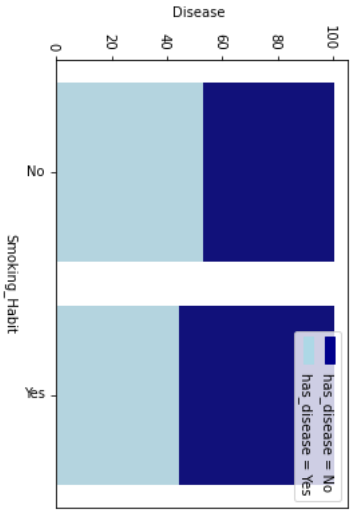


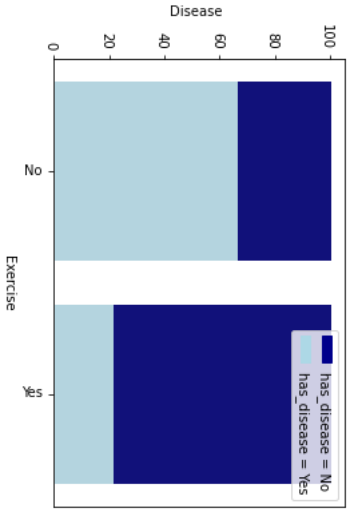


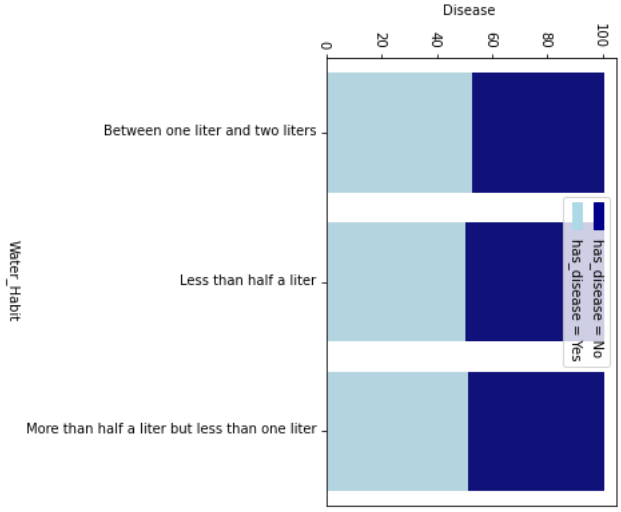


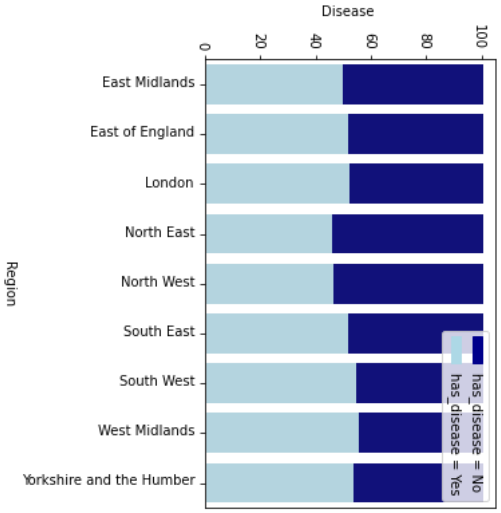


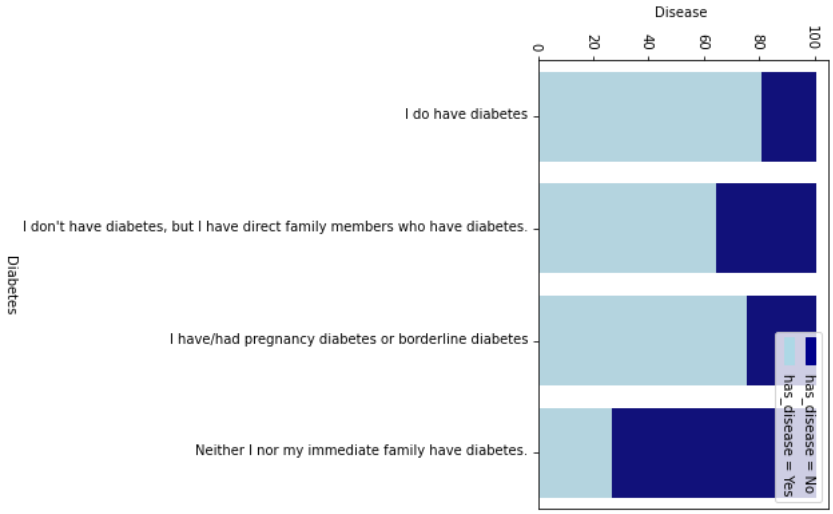
Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.











Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

2.2. Numerical Feature Exploration

Point 3. Central Tendency Metrics by Diseased or Non-Diseased

		Disease	
		0	1
Birth_Year	count	389	411
	mean	1963.52	1968.43
	std	16.52	13.91
	min	1859	1855
	25%	1960	1963
	50%	1964	1970
	75%	1970	1977
	max	1987	1993
Height	count	389	411
	mean	169.21	166.48
	std	7.34	8.33
	min	151	151
	25%	165	158
	50%	171	167
	75%	174	173
	max	180	180
Weight	count	389	411
	mean	65.65	69.89
	std	11.69	12.16
	min	40	41
	25%	56	61
	50%	64	71
	75%	73	79
	max	97	97
High_Cholesterol	count	389	411
	mean	253.31	245.55
	std	47.97	54.55
	min	135	130
	25%	220	211
	50%	252	238
	75%	287	272
	max	413	568

		Disease	
		0	1
Blood_Pressure	count	389	411
	mean	133.97	128.29
	std	18.55	15.01
	min	100	94
	25%	120	120
	50%	130	130
	75%	144	138
	max	200	180
Mental_Health	count	389	411
	mean	15.03	19.53
	std	5.30	4.48
	min	0	5
	25%	12	17
	50%	16	20
	75%	19	23
	max	28	29
Physical_Health	count	389	411
	mean	7.03	2.22
	std	6.15	3.28
	min	0	0
	25%	2	0
	50%	6	0
	75%	11	4
	max	30	20

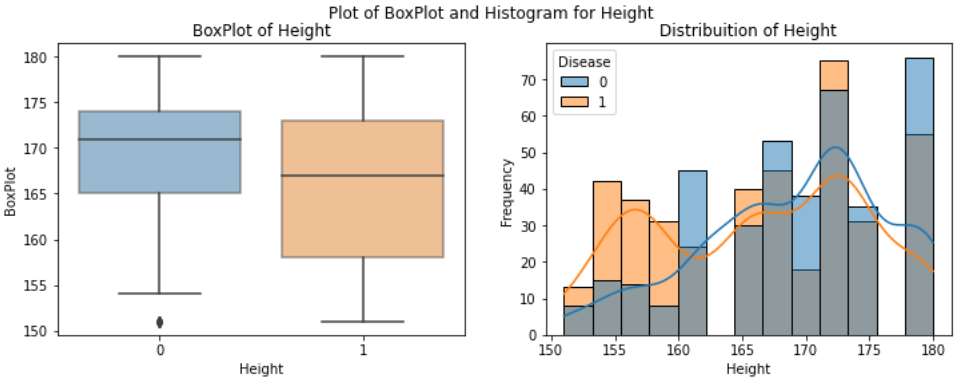
Point 4. Histogram and Boxplot Univariate Function Description:

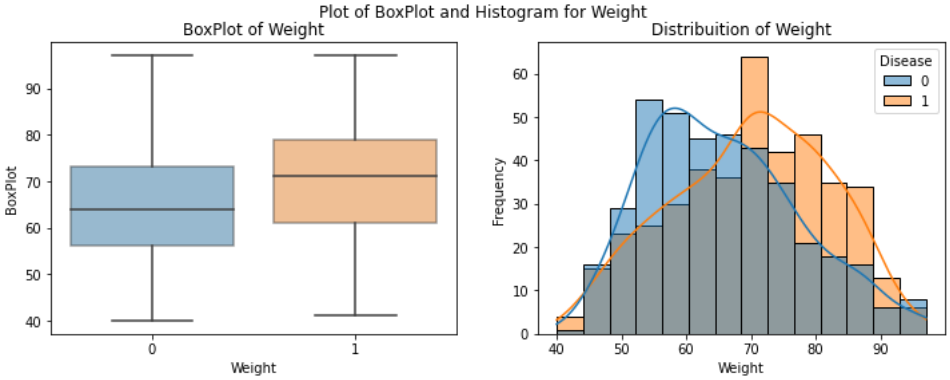
A histogram and boxplot are graphical representations that show the distribution and statistical characteristics of a variable, respectively. A histogram displays the frequency of different values of the variable, while a boxplot shows the range, central tendency, and variability of the variable.

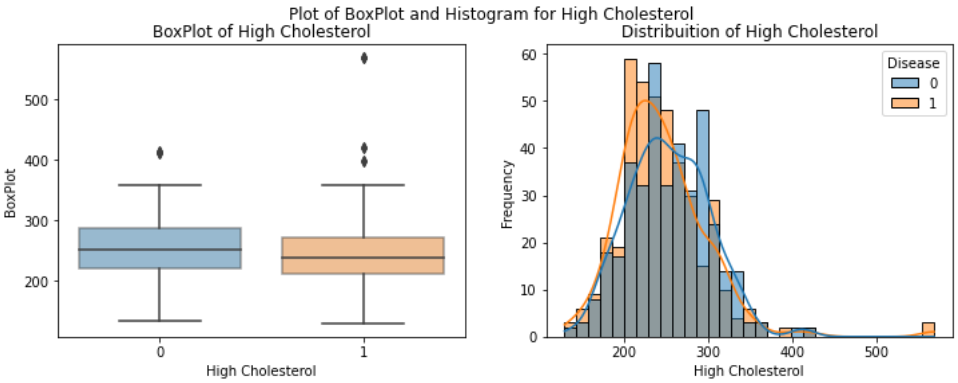
The function "plot_histogram_and_boxplot_univariative" is a visualization tool that plots the distribution and frequency of a specific variable along with the target variable Disease, in the form of a boxplot and histogram. It allows the examination of the relationship between the selected variable and the target variable. The function requires four input arguments: the dataset, the variable to be plotted, the name of the variable and the target variable Disease. The resulting plots are labelled with appropriate titles and axis labels.

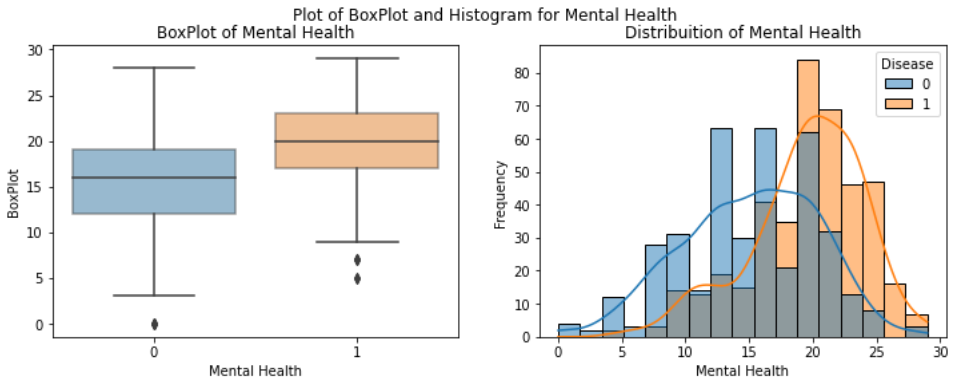
Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

Point 5. Box Plot and Histogram for each of the Numerical Features (Pre-Outlier Removal):

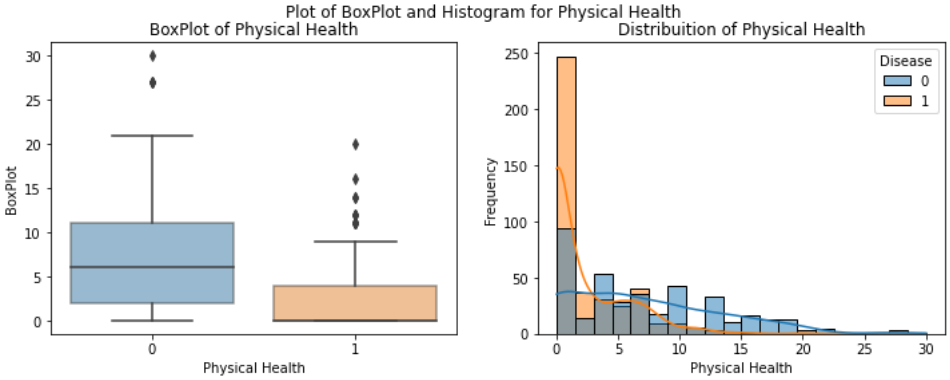




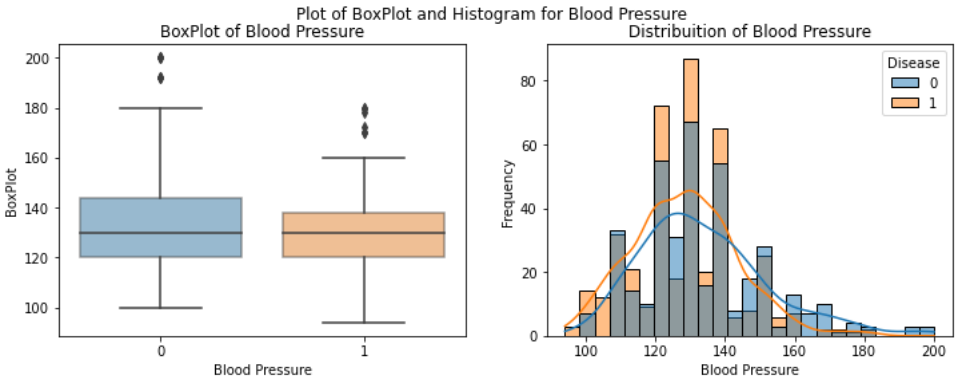




Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

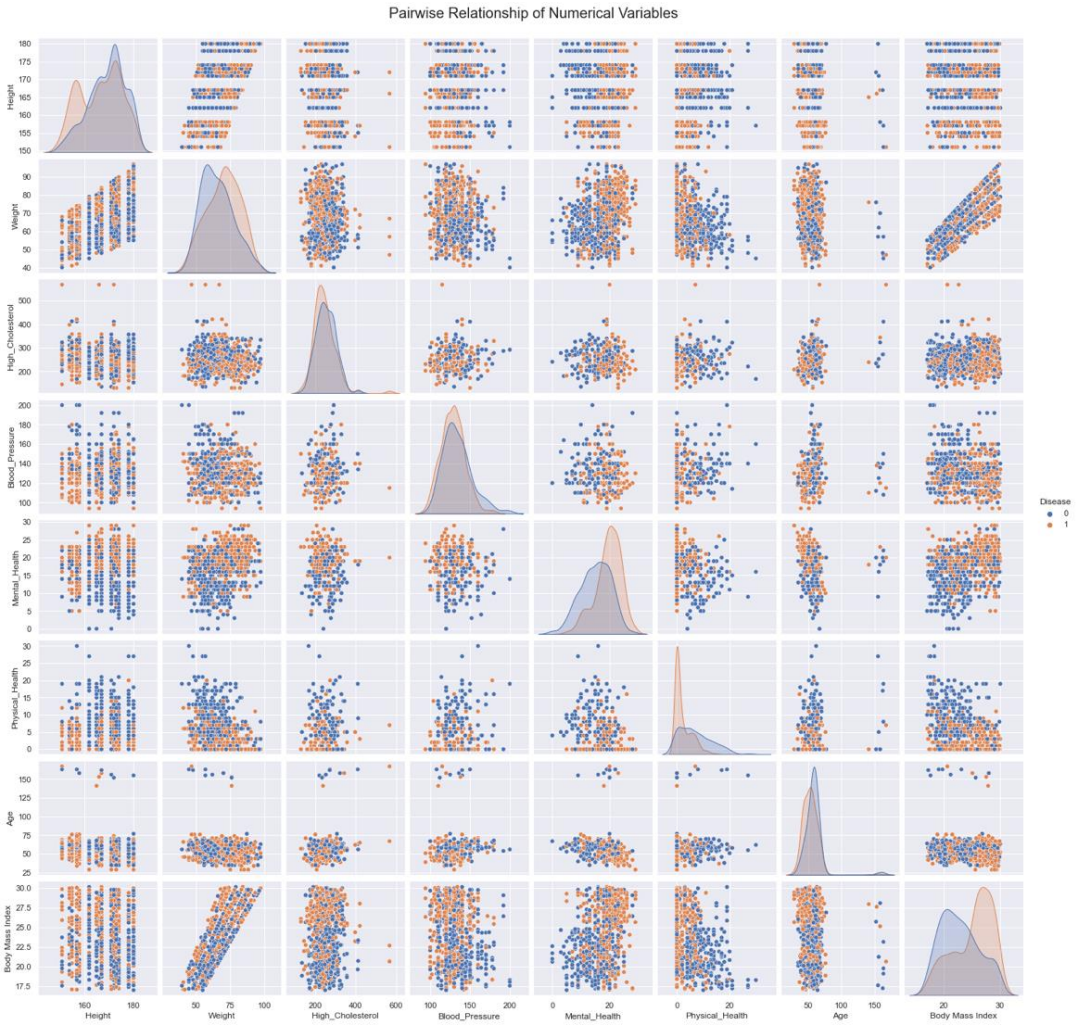


Plot of BoxPlot and Histogram also for "Birth_Year"



Additional notes on the appendix, counting towards "Creativity and Other Self-Study".

Point 6. Pairwise Scatter Plot between the Numerical Features (Pre-Outlier Removal):



3. Pre-Processing

3.1. Categorical Feature Pre-Processing

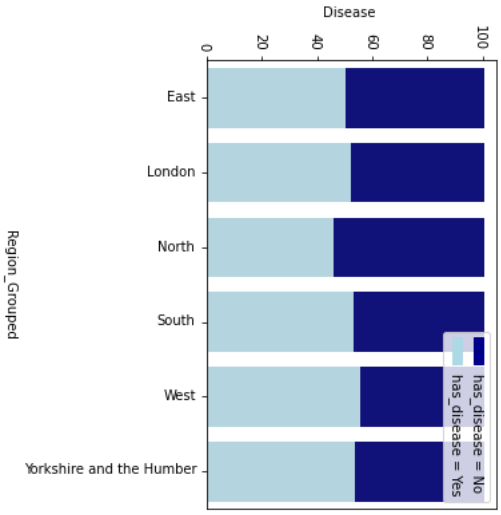
Point 7. Explanation of the new variables created during Feature Engineering:

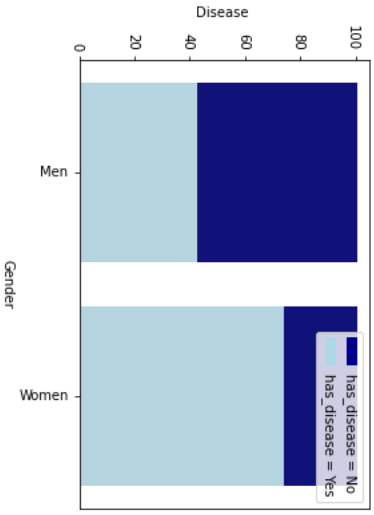
New Feature	Original Feature	Objective
Gender	Name	The original dataset did not include a 'Gender' variable, but it did include a 'Name' variable with the titles 'Mr'. and 'Mrs.' that indicated whether the patient was a man or a woman. Since diseases can sometimes affect more one gender than another, we thought it would be valuable to include this new feature for our predictions.
Region_Grouped	Region	As we did the Data Exploration, we realized the categorical feature 'Region' had high cardinality (it could take up to 9 different values). Therefore, in order to reduce it, we created 6 grouped regions based on the 4 cardinal directions, one for London and other for Yorkshire and the Humber.

Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

Age	Birth_Year	The original dataset included the birth year of the patients, but for interpretation purposes we assumed the database was from 2022 and transformed this variable into 'Age'. This way if this feature turned out to be a predictor, we could easily describe the age range of the people most likely to be infected by the parasite.
BMI	Weight and Hight	According to the WHO, Body Mass Index "is a measure for indicating nutritional status in adults. It is defined as a person's weight in kilograms divided by the square of the person's height in meters (kg/m2) …. BMI was developed as a risk indicator of disease; as BMI increases, so does the risk for some diseases." As we are trying to predict a disease, we thought this would be a more valuable indicator and could also aid with feature reduction as we would be replacing two variables with one.

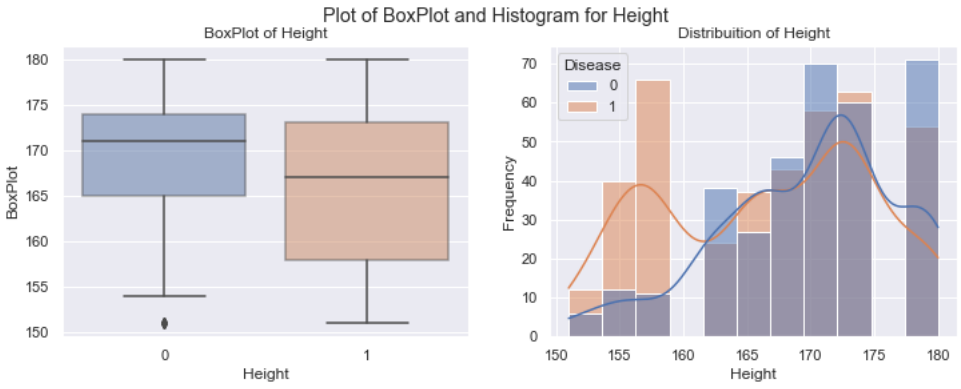
Point 8. Volumetry Plots for each of the Categorical Features:



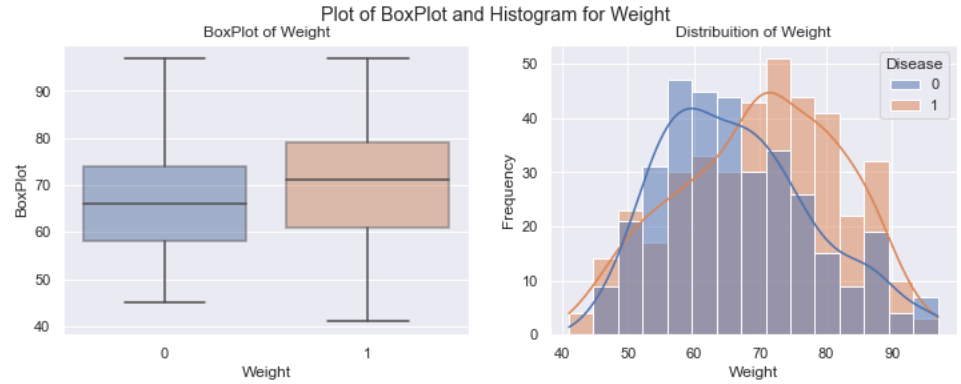


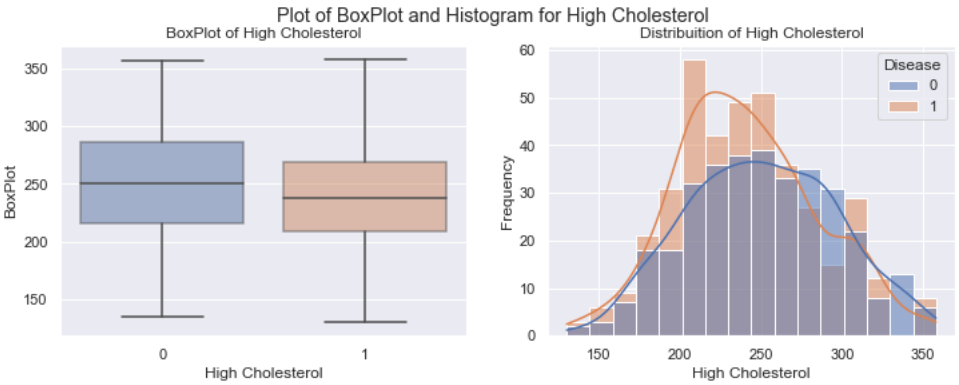
3.2. Numerical Feature Pre-Processing

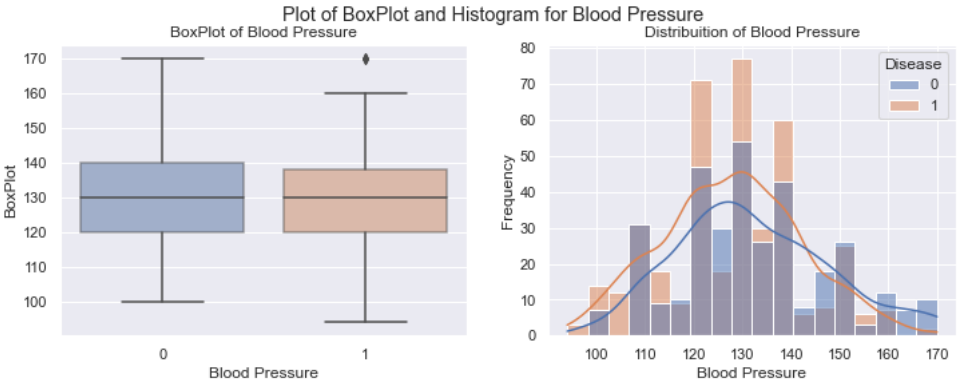
Point 9. Box Plot and Histogram for each of the Numerical Features (Post-Outlier Removal):

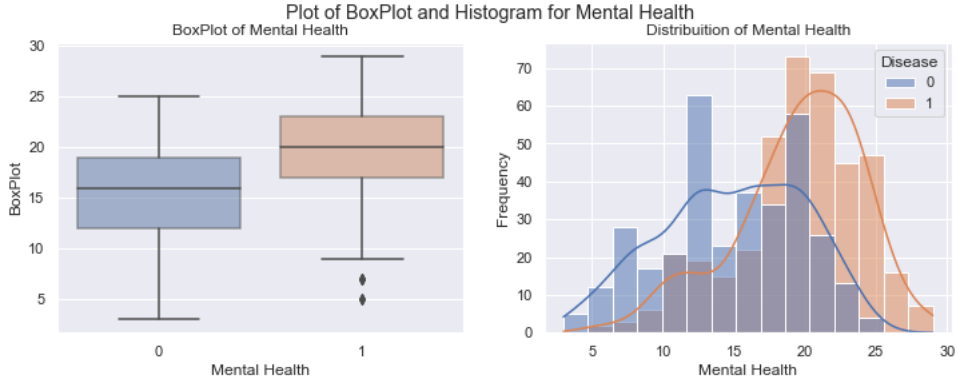


Additional notes on the appendix, counting towards "Creativity and Other Self-Study".

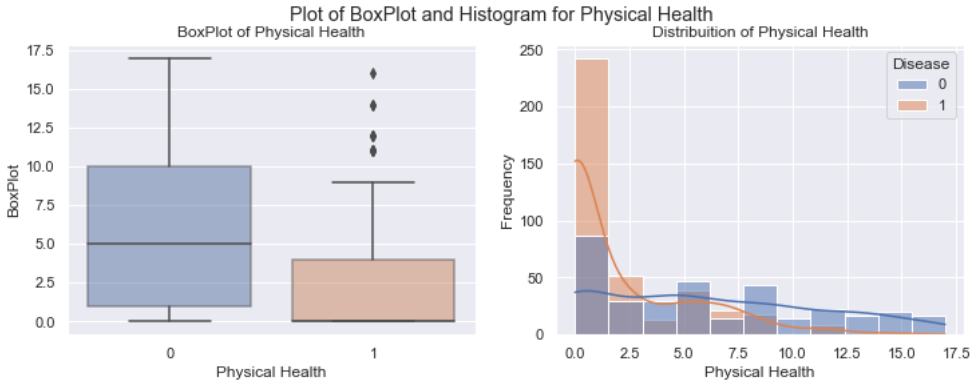


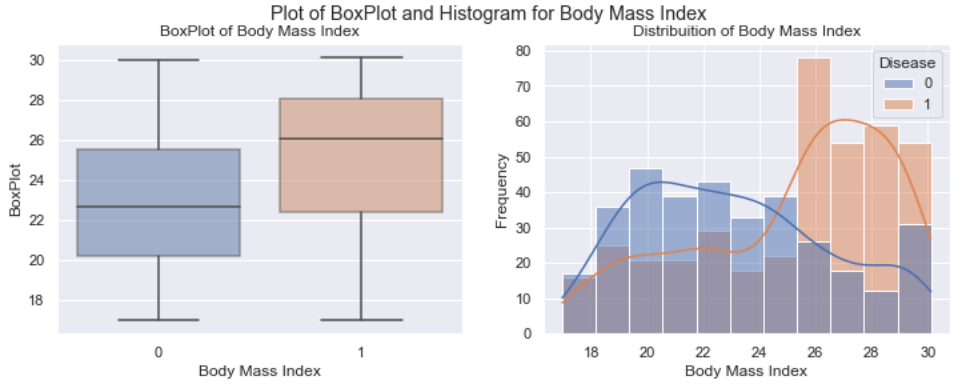


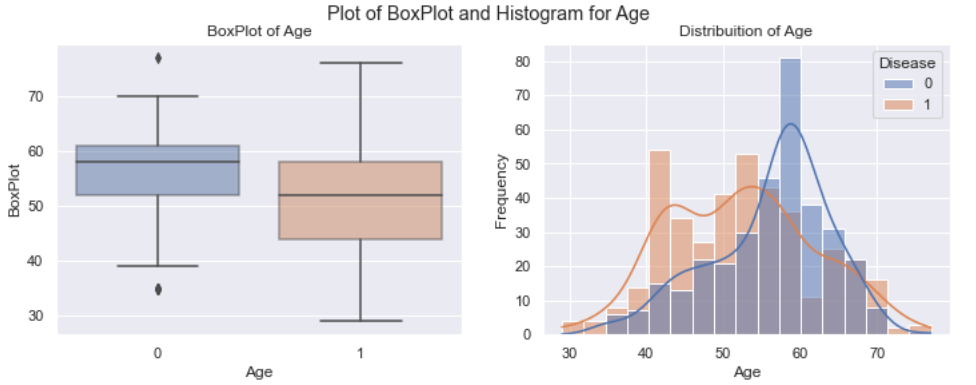




Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

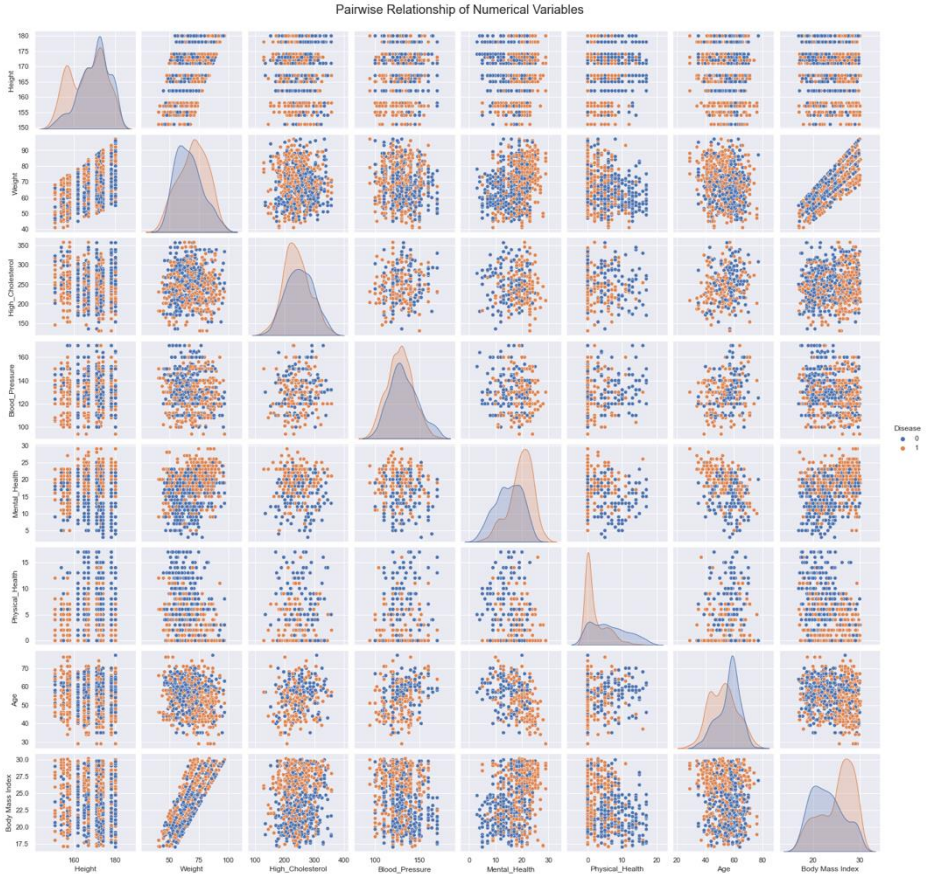






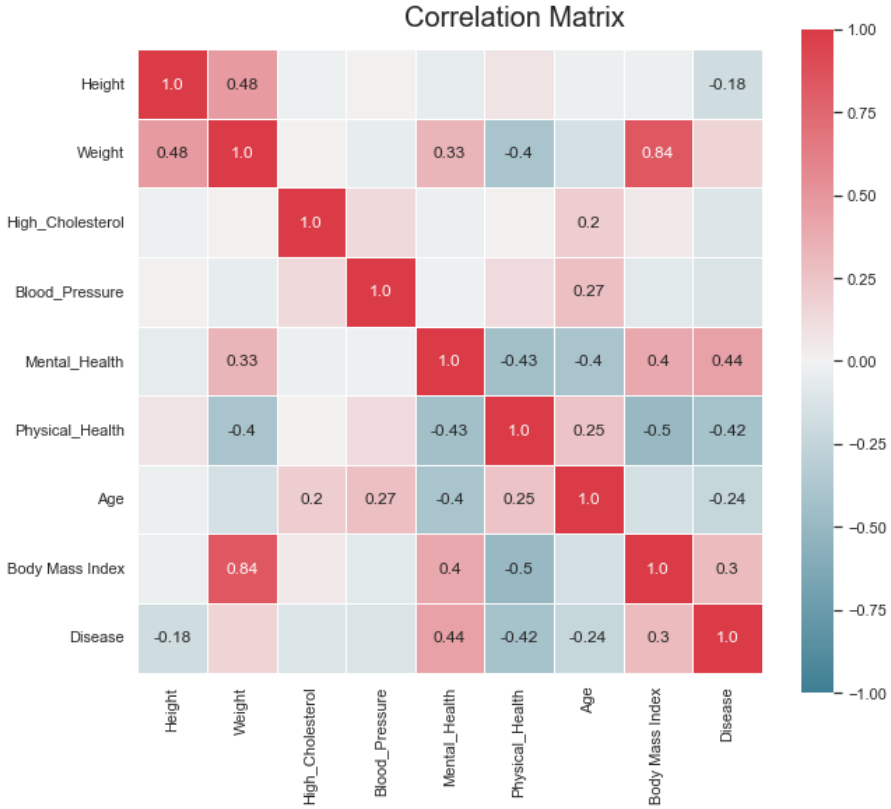
Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

Point 10. Pairwise Scatter Plot between the Numerical Features (Pre-Outlier Removal):



3.3. Methods for Feature Selection

Point 11. Spearman Correlation Matrix



Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

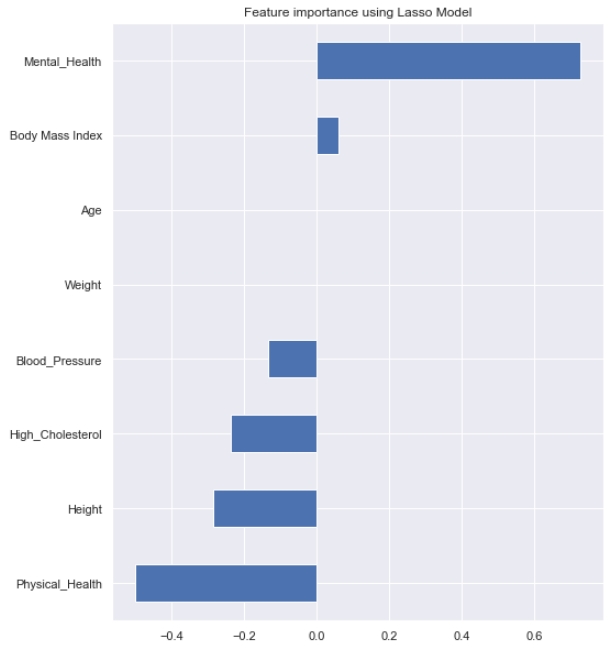
Point 12. Wrapper Method for Numerical Features, 1st Method: For-Loop with Recursive Feature Elimination:

We developed a method to improve the stability of our recursive feature elimination process for Logistic Regression. Initially, we were getting different numbers of features each time we ran the function, and while we could have addressed this issue with a random state, we believed there was a better solution. We created a for loop that determines the optimal number of features in each iteration and stores them in a list. After all the iterations are done, the algorithm identifies the number of features that was most frequently chosen. This approach gave us a more reliable and accurate result for the number of features to use.

Point 13. Wrapper Method for Numerical Features, 2nd Method: Recursive Feature Elimination with Cross Validation:

To further enhance the stability of the recursive elimination process for Logistic Regression, we discovered the function RFECV() in the sklearn library, a function not taught in class. This function performs cross validation and in each iteration of the cross validation it determines the optimal number of features for that specific folding. After all the iterations, corresponding to the number of folds, it selects the number of features that was most frequently chosen as the best number of features.

Point 14: Lasso Feature Importance



4. Modelling

Point 15. Calculate F1 Score Average Function:

The function calculate_f1_score_average calculates the F1 score using Stratified K-Fold cross-validation as a strategy. It allows users to specify certain treatments for the dataset, such as scaling, OneHotEncoding, and outlier removal, to be applied during each iteration of the cross-validation. These

Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

treatments are only applied to the training fold, while the validation fold uses the fitted versions of these treatments to avoid data leakage and produce more robust model results. It was created to make easy the evaluation of all models that we trained in the notebook, avoiding the repetition of codes for pre-processing data for each model.

4.3. K-Nearest Neighbours Classifier

Point 16. Datasets where Grid Search was applied for KNN

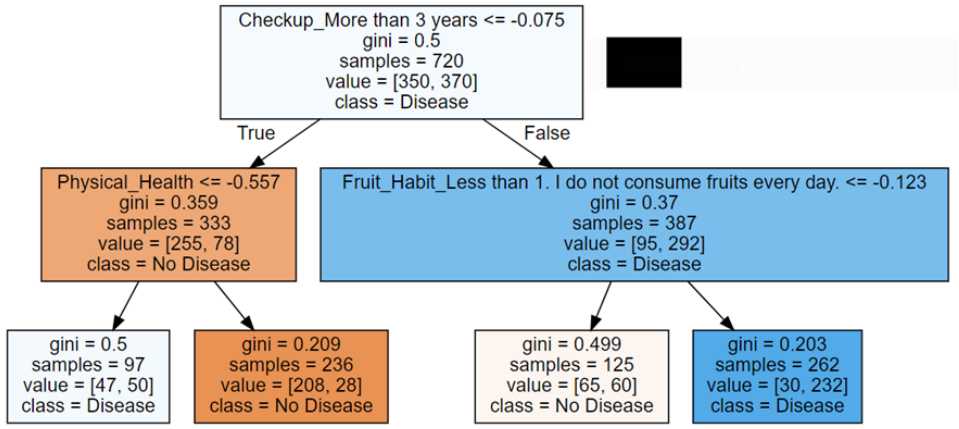
		OneHot Encoding	Scaler	Outlier Removal	Feature Selection
1	Model11	YES	MinMax(0,1)	NO	None
2	Model12	YES	MinMax(-1,1)	NO	None
3	Model13	YES	Standard	NO	None
4	Model14	YES	Robust	NO	None
5	Model15	YES	MinMax(0,1)	YES	None
6	Model16	YES	MinMax(-1,1)	YES	None
7	Model17	YES	Standard	YES	None
8	Model18	YES	Robust	YES	None
9	Model21	YES	MinMax(0,1)	NO	Filter
10	Model22	YES	MinMax(-1,1)	NO	Filter
11	Model23	YES	Standard	NO	Filter
12	Model24	YES	Robust	NO	Filter
13	Model25	YES	MinMax(0,1)	YES	Filter
14	Model26	YES	MinMax(-1,1)	YES	Filter
15	Model27	YES	Standard	YES	Filter
16	Model28	YES	Robust	YES	Filter
17	Model31	YES	MinMax(0,1)	NO	Wrapper
18	Model32	YES	MinMax(-1,1)	NO	Wrapper
19	Model33	YES	Standard	NO	Wrapper
20	Model34	YES	Robust	NO	Wrapper
21	Model35	YES	MinMax(0,1)	YES	Wrapper
22	Model36	YES	MinMax(-1,1)	YES	Wrapper
23	Model37	YES	Standard	YES	Wrapper
24	Model38	YES	Robust	YES	Wrapper
25	Model41	YES	MinMax(0,1)	NO	Embedded
26	Model42	YES	MinMax(-1,1)	NO	Embedded
27	Model43	YES	Standard	NO	Embedded
28	Model44	YES	Robust	NO	Embedded
29	Model45	YES	MinMax(0,1)	YES	Embedded
30	Model46	YES	MinMax(-1,1)	YES	Embedded
31	Model47	YES	Standard	YES	Embedded
32	Model48	YES	Robust	YES	Embedded

Point 17. Column Transformer + Pipeline + GridSearchCV() Combo Explanation:

To solve the issue of not properly scaling the data during cross-validation using grid search, we implemented a solution using two functions from the sklearn library: column transformer and pipeline. With column transformer, we defined the pre-processing techniques to apply to specific columns, such as scaling for numerical features and hot encoding for categorical features. We then created a pipeline and inserted our column transformer and the model we wanted to use in there as parameters. By using this pipeline as our model in the grid search, we ensured that the hot encoding and scaling were applied to each new combination of folds during each iteration of cross-validation, properly scaling the data, and effectively eliminating the risk of data leakage.

4.4. Decision Tree Classifier

Point 18. Decision Tree with depth of 2 for better understanding of feature importance.



4.7. Gradient Boosting Classifier

Point 19. Datasets where Grid Search was applied for Gradient Boosting

		OneHot Encoding	Scaler	Outlier Removal	Feature Selection
1	Model11	YES	NO	NO	None
2	Model12	YES	NO	YES	None
3	Model21	YES	NO	NO	Filter
4	Model22	YES	NO	YES	Filter
5	Model31	YES	NO	NO	Wrapper
6	Model32	YES	NO	YES	Wrapper
7	Model41	YES	NO	NO	Embedded
8	Model42	YES	NO	YES	Embedded

4.9. Stacking and Voting Classifier

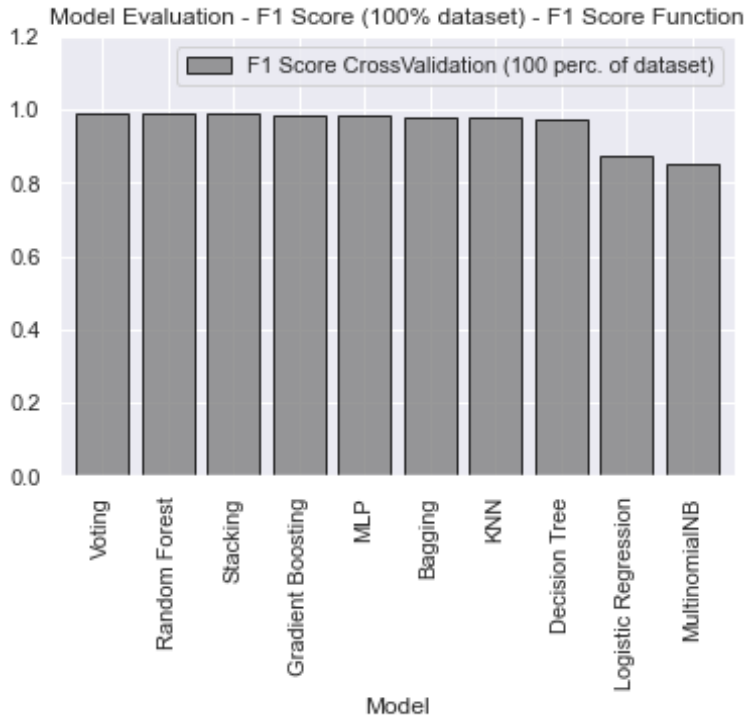
Point 20. Theoretical Explanation of the Voting Classifier:

For our final model we chose Voting Classifier, an ensemble algorithm that combined the predictions of our top three performing individual classifiers (Random Forest, Bagging and MLP) to make the final prediction. The idea behind, is that by combining models to make a prediction, we mitigate the risk of one model making an inaccurate prediction by having other models that can make the correct prediction, thus enabling the estimator to have an improved predictive performance and be more robust to overfitting. Even though it can be computationally expensive and time-consuming, Voting Classifier is relatively simple, as it involves training and using multiple classifiers rather than optimizing a single complex model. It is important to note that Voting only serves to benefit when the individual classifiers perform at similar levels, and given this was our case, it worked perfectly.

Additional notes on the appendix,
counting towards “Creativity and Other
Self-Study”.

5. Model Assessment

Point 21. Bar Chart of the F1 Score Performance of each best model



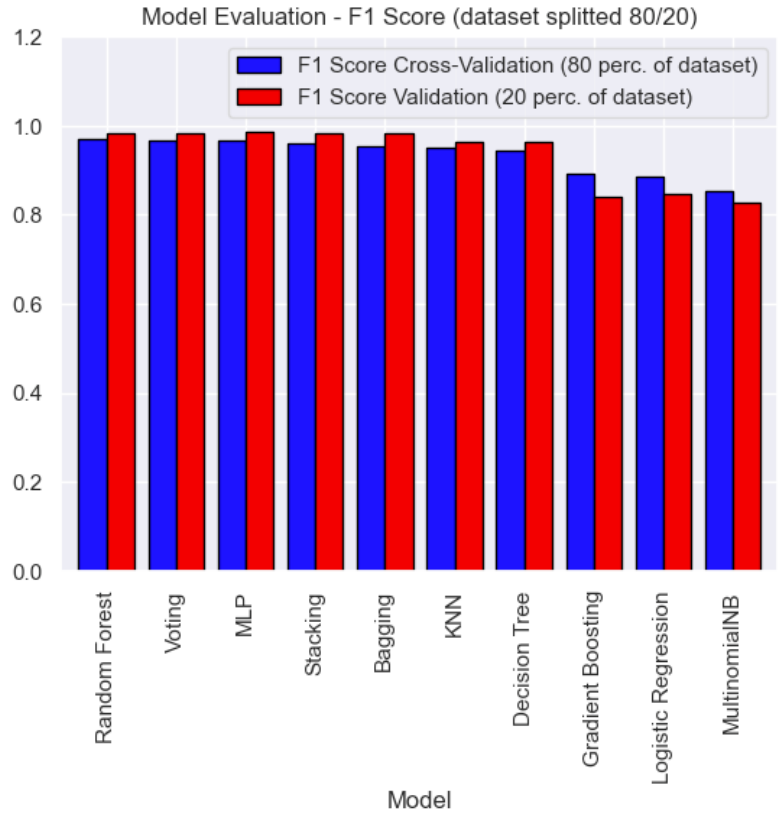
Point 22. Table with the results of the 80%/20% split

Since the VotingClassifier is parametrized as 'hard', it does not have the ability to access the predict_proba() function. As a result, it is not possible to calculate the AUC (Area Under the Curve) for this model. If we had used the 'soft' parameter, which considers the probabilities of the models, we would have been able to access the predict_proba() function, but it would have resulted in a worse performance.

Models	F1 Score Cross-Validation (80 perc. of dataset)	F1 Score Validation (20 perc. of dataset)	Precision	Recall	AUC Validation	Confusion Matrix			
						tp	fn	fp	tn
MLP	0.967	0.988	1.000	0.976	0.987	78	0	2	80
Random Forest	0.969	0.981	1.000	0.963	1.000	78	0	3	79
Bagging	0.953	0.981	1.000	0.963	0.999	78	0	3	79
Voting	0.968	0.981	1.000	0.963	-	78	0	3	79
Stacking	0.961	0.981	1.000	0.963	1.000	78	0	3	79
KNN	0.951	0.963	0.987	0.939	0.998	77	1	5	77
Decision Tree	0.945	0.962	1.000	0.927	0.963	78	0	6	76
Logistic Regressio	0.884	0.847	0.852	0.841	0.917	66	12	13	69
Gradient Boosting	0.891	0.841	0.841	0.841	0.952	65	13	13	69
MultinomialNB	0.854	0.826	0.812	0.841	0.887	62	16	13	69

Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

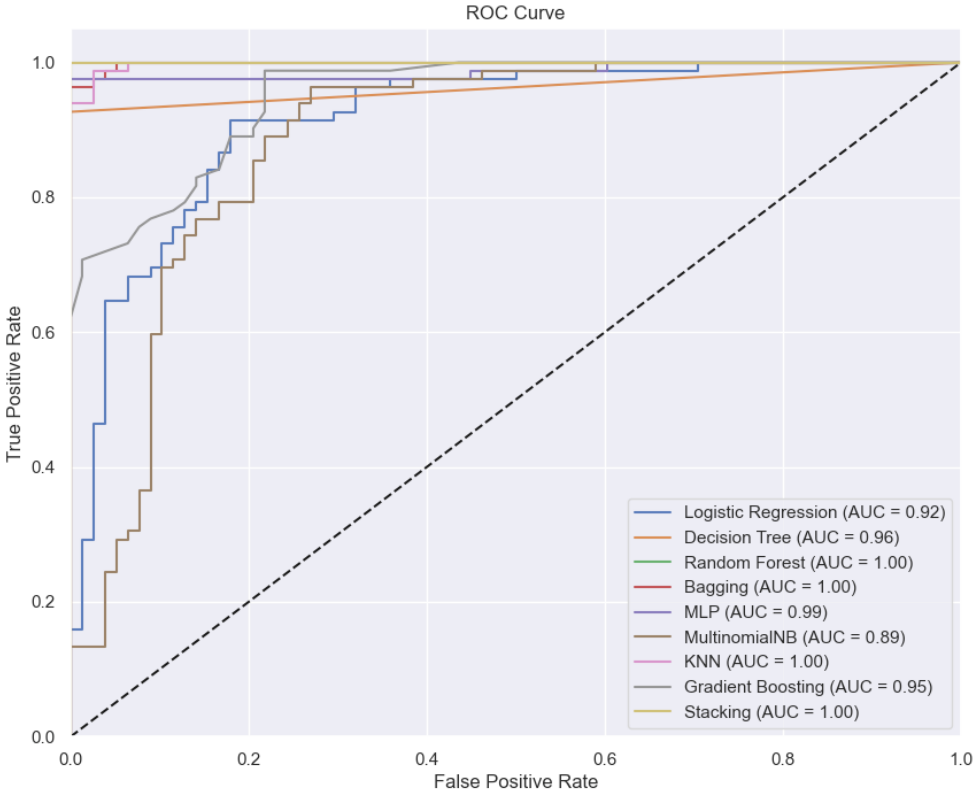
Point 23. Bar Chart with the results of the 80%/20% split



Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

Point 24. ROC Curve and respective AUC Score

Since the VotingClassifier is parametrized as 'hard', it does not have the ability to access the predict_proba() function. As a result, it is not possible to plot the ROC curve for this model. If we had used the 'soft' parameter, which considers the probabilities of the models, we would have been able to access the predict_proba() function, but it would have resulted in a worse performance.



Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.

References

- *Linear Models* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/linear_model.html
- *Naïve Bayes* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/naive_bayes.html
- *Nearest Neighbors* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/neighbors.html
- *Neural network models (supervised)* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/neural_networks_supervised.html
- *Decisions Trees* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/tree.html
- *Ensemble methods* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/ensemble.html
- *Pipelines and composite estimators* (n.d.) scikit-learn.org. https://scikit-learn.org/stable/modules/compose.html
- Nair, A. (2021, October 12). Combine Your Machine Learning Models With Voting. *Medium*. https://towardsdatascience.com/combine-your-machine-learning-models-with-voting-fa1b42790d84
- World Health Organization. (2010, May 6). *A healthy lifestyle – WHO recommendations.* www.who.int. https://www.who.int/europe/news-room/fact-sheets/item/a-healthy-lifestyle---who-recommendations

Additional notes on the appendix, counting towards “Creativity and Other Self-Study”.