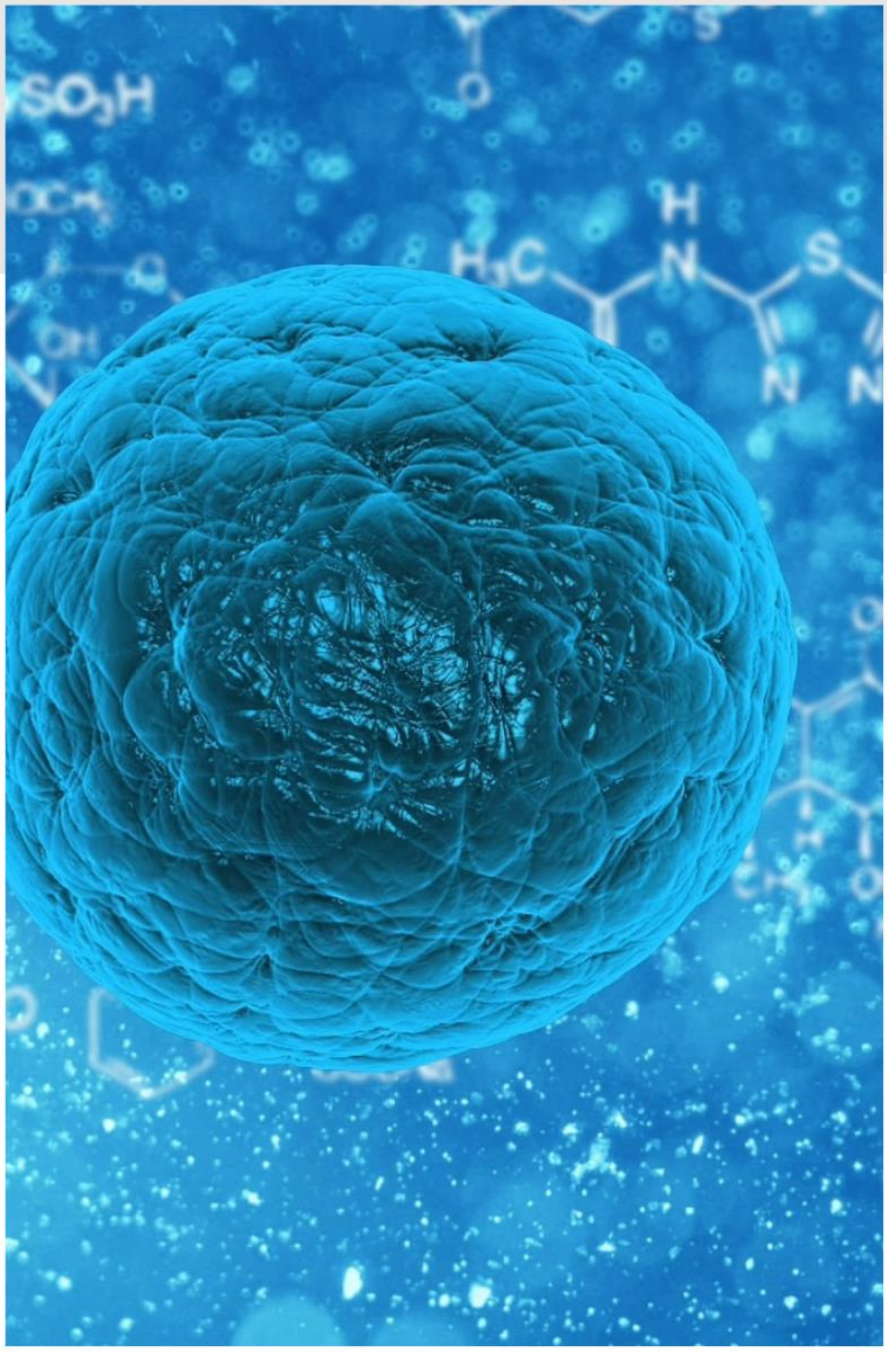
DECEMBER 2022

THE SMITH PARASITE

Machine Learning - Final Project

Prof. Roberto Henriques | Prof. Carina Albuquerque | Prof. Ricardo Santos



GROUP 7:

Daniel Branco, r20191230
Filipe Dias, r20181050
Gonçalo Lourenço, r20191097
Inês Santos, r20191184
Manuel Marreiros, r20191223

NOVA Information Management School | Universidade Nova de Lisboa
Master's Degree Program in Data Science and Advanced Analytics

Index

INTRODUCTION **2**

EXPLORATION **2**

PREPROCESSING **3**

Coherence Checking 3

Feature Engineering 3

Filling Missing Values 3

Outliers 3

Encoding and Split 4

Scaling 4

Feature Selection 5

MODELLING **6**

Ensemble models 6

1. *Bagging Classifier* *6*
2. *Random Forest* *7*
3. *Extra Trees* *7*
4. *AdaBoost* *7*
5. *Gradient Boosting* *8*
6. *Voting Classifier* *8*
7. *Stacking Classifier* *8*

ASSESSMENT **8**

DEPLOYMENT **9**

CONCLUSION **9**

REFERENCES **10**

ANNEXES **11**

Figures 11

Confusion Matrices 25

Tables 26

Introduction

With the present study of the curricular unit of Machine Learning, we intend to analyse a newly discovered disease that has been affecting thousands of people in England. This disease, discovered by Dr. Smith, is still quite understudied, with the conditions of transmission and predisposing factors for infection still being relatively unknown. For that matter, and from a public-health point of view, getting a deeper understanding of the underlying nuances associated with the propagation of this parasite is of the utmost importance.

Thus, the goal of this project is to build a **predictive mode**l that can answer the question *"Who are the people more likely to suffer from the Smith Parasite?"*, in other words, a model that, based on a patient's characteristics, is able to predict if said patient will suffer, or not, from this disease. This means that we are dealing with a **classification problem**.

Exploration

We were given two datasets, one for training, and one for testing. To get acquainted with the data, we first focused on performing basic exploratory functions on the training dataset, such as checking if there were duplicate records or undesirable characters, of which there were none, and looking for missing values.

Then, we looked into the statistics and visual analysis of the data. Starting by dividing the variables into numerical and categorical, we got a summary of their statistics. We also created a discretized version of the numerical variables *High_Cholestrol* and *Blood_Pressure*, to visually extract more meaning from the data. Most importantly, we analysed the attributes of the patients that did not contract the disease versus the ones who did, to get an initial perception of how each variable impacted the carrying of the virus.

For each categorical variable, we produced stacked bar charts. We concluded, for instance, that variables such as *Region* and *Education* did not offer great discrimination, as opposed to *Fruit_Habit*, where we observed that people who ate at least 1 piece of fruit per day and less than 6 had a lower incidence rate, *Checkup*, where the disease had a lower incidence on people who had gone more recently to the doctor, *Exercise*, where the disease infected mainly people who admitted to not exercising regularly, and *Drinking_Habit*, considering that people who consume alcohol every day are more prone to contract the disease. As for numerical variables, after plotting the features *Mental_Health* and *Physical_Health* in a Bubble Chart where the size of the points reflected the amount of patients, and the colour whether they had contracted the disease or not, we observed an agglomerate of red bubbles on the lower-right corner of the graph, indicating that patients who had reported more mental health issues and less physical health issues had been infected by the virus. Producing an equivalent graph for variable *Weight* and *Height*, we observed that, regardless of height, the disease seemed to mostly affect overweight people.

Preprocessing

Coherence Checking

Before the development of our model, it was important to make sure that our data made sense, meaning we had to look for incoherencies. For instance, the *Mental_Health* and *Physical_Health* attributes refer only to the last 30 days, so their values shouldn't be smaller than 0 nor greater than 30, which checked out.

When looking at the values of the different categorical columns, we found that there was an issue in the column *Region*, where London was written in two different ways: "London" and "LONDON". This was easily solved by converting the upper-case word to match its counterpart.

Feature Engineering

We then moved on to feature engineering, where we created a variable *Gender* based on the prefix of the name (Mr., Mrs., or Miss), as well as a variable for the body mass index, *BMI*, which resulted from a ratio between weight and height. This way, with the help of some domain-specific knowledge, we were able to merge the information of two variables into just one, which allows for a reduction of the input space, as we will see in the Feature Selection section. Lastly, we transformed the variable *Birth_Year* into *Age* for an easier interpretation, by subtracting the year in which each person was born to the current date.

Filling Missing Values

Analysing the missing values, we concluded that the only variable that presented this issue was *Education* (13 missing values – around 1.6% of the observations). For that matter, and since the variable in question was categorical, we opted to use a measure of central tendency, the mode, to fill these values. As many machine learning algorithms do not allow missing values, addressing missing data correctly during this phase is crucial because it decreases bias and increases the quality of our models.

Outliers

Regarding the outliers, we started by producing boxplots and histograms for all numeric variables. This set of plots gave us an initial idea of what outliers we had in our data, and we concluded that there were a few in *High_Cholestrol* and *Age*. After this univariate outlier analysis, we proceeded to look at multivariate (bivariate, in this case) outliers, which are a combination of unusual scores in at least two variables. To do that, we plotted the Pairwise Relationship of Numerical Variables by Disease.

For an **automatic outlier removal**, we tested several methods, namely the **Inter Quartile Range (IQR)**, **Local Outlier Factor** and **Z-Score**. We discarded the IQR because it was removing 7.5% of our data, which was excessive. The IQR method will delete everything that is 1.5 times smaller than the first quartile or 1.5 greater than that the third quartile and deleting that many records would most

likely lead to overfitting. This is because our models would become exceedingly adapted to the highly manipulated and controlled train data, which would cause them to fail when making predictions regarding unseen data. Moreover, after checking the data frames posterior to the outlier removal, the one in which we used the Local Outlier Factor still had some values that made no sense (e.g., people with 158 years of age were still present). We believe that happened because the Local Outlier Factor algorithm computes the local density deviation of a given data point relatively to its neighbours, and it considers the samples that have a considerably lower density than their neighbours as outliers. However, if there are many outliers close together, they won't be recognized as such.

All things considered, Z-score was the method that provided the best results, specifically with a cut-off set to 3, meaning that any z-score greater than +3 or smaller than -3 was considered an outlier.

Encoding and Split

To enable a better processing of Categorical Variables, we used **dummy variables** to **one-hot encode** our data, which resulted in a data frame with 38 columns. While we considered doing ordinal encoding instead, this wouldn't have produced good results, since there is no way to measure the distance in categorical variables.

For the split, we decided to divide our dataset in such a way that 80% of the records would go to the training, and the remaining 20% to the validation. We also stratified by the dependent variable to keep the proportion of 0's and 1's in both the new datasets, resulting in around 53% of 1's and 47% of 0's. This similarity in both cases' proportions is also important to avoid issues related to **imbalanced learning**, where models wind up performing worse when attempting to identify members of the minority class. By the end, we had four sets of data: *X_train* for the independent data used to train the models; *y_train* with the dependent variable correspondent to the records of *X_train*; *X_val* for the independent data used to validate the models; *y_val* with the dependent variable correspondent to the records of *X_val*.

Scaling

Immediately thereafter, we proceeded with the normalization of the data (or scaling). Our variables were measured in different scales, so it was necessary to adjust the values to a common scale, since models assume that distances in different directions of the input space have equal importance. For the effect, we used the **MinMaxScaler Method**, adjusting the variables to a scale from 0 to 1. We tried several scaling methods, but ultimately chose this one since it was the most well-adjusted to the data we had. **Standard Scaler** would have required our features to be more or less normally distributed (e.g., Gaussian with 0 mean and unit variance), which was not the case, and **Robust Scaler** wouldn't have made a big difference because we had already removed outliers.

Feature Selection

Because we understand the importance of selecting the right features for our models, feature selection was one of the steps to which we paid the most attention. We first started by examining the correlations between variables using the **Spearman Correlation**, which evaluates how well a monotonic function can explain the relationship between two variables. This way, we quickly identified clearly redundant variables, such as '*Checkup_Not sure*' and '*Checkup_More than 3 years'*, which had a strong negative correlation.

To further our analysis, we applied 5 feature selection algorithms, them being the **Logistic Regression**, **Random Forest**, **Recursive Feature Elimination (RFE)**, **Select K Best** and **Mean Absolute Diference**. Each algorithm classified the importance of each of our features, which we then converted to *True* for important features, and *False* for unimportant features. From these 5 methods, we would like to highlight the Recursive Feature Elimination. This method works by training a given estimator (Random Forest, in our case) with the entire set of features, and then recursively considering smaller and smaller sets of features, removing the least important features in each iteration. Additionally, with a specific implementation of RFE with cross-validation, in which we used a 5-fold cross validation, we were able to obtain a graph with the cross-validation score on the y axis, and the number of features in the x axis, which made us settle with 10 features.

In the end, we calculated how many times each variable was classified as important and we selected the ones that were classified as such the most, them being: '*Gender_M*', '*Fruit_Habit_Less than 1. I do not consume fruits every day*', '*Exercise_Yes*', '*Diabetes_Neither I not my immediate family have diabetes.*', '*Checkup_More than 3 years*', '*BMI*', '*Physical_Health*', '*Mental_Health*', '*High_Cholestrol*' and '*Drinking_Habit_I usually consume alcohol every day*'. All these 10 variables were **classified as important by at least 4 of our 5 models**, meaning that they are the most important of the entire set. While others had this same score, they were excluded due to being highly correlated to at least one of the selected 10. For instance, we did not include '*Checkup_Not sure*', since we already had a variable associated with the latest check-up of the patient. Another point in favour of opting for *'Checkup_More than 3 years'* was the fact that it presented a more significant correlation with the target variable.

Furthermore, we wanted to eliminate any univariate numerical variables. For that effect, we checked the variance for all our variables, concluding that none of them was univariate. In other words, there were no constant or quasi-constant features. Finally, we examined the Spearman Correlation matrix once again, to make sure we weren't including any features with a high correlation amongst them. After concluding this process, we ended up with **10 features that were relevant for our analysis**, not correlated between them, thus having a much smaller input space, which allowed for a faster and more efficient analysis of our problem, avoiding the curse of high dimensionality.

Modelling

After the pre-processing steps were finished, it was time to start modelling. In this stage, we tried using several different predictive models to draw conclusions on which one would produce the best results. We started out by looking into simpler models, and then moved on to more complex ones, finishing with ensemble models. The latter are combinations of different individual models, as we will expand upon further ahead in its own section.

The non-Ensemble models we tested were: **Logistic Regression**, **Gaussian Naïve Bayes**, **KNN**, **Decision Trees**, **Support Vector Machine** (specifically, **C-Support Vector Classification (SVC)**, **Nu-Support Vector Classification (NuSVC)**, and **Linear Support Vector Classification (LinearSVC)**), and **Neural Networks**. Throughout the modelling process, we tried to find the optimal setting of hyperparameters for each model, mostly using **Grid Search**, with the aim of obtaining the combination that would lead to the best result. While our goal was to attain the best F1-Score, we aimed to get the least amount of overfitting possible as well. For instance, Decision Trees can easily lead to overfitting, which we can fix by constraining their growth. This can be achieved by limiting their parameters, but a more effective way is by using post pruning methods, such as cost complexity pruning. Another noteworthy decision we made in this phase was to opt for **Randomized Search** in place of Grid Search when it came to the Neural Networks, since going through the entire hyperparameter space was very time consuming.

While the aforementioned methods did not generate the best results, we could extract some meaningful conclusions to our analysis. For example, using Decision Trees, we could plot feature importance, further corroborating our feature selection. Moreover, out of the various SVM methods, the one that got better performance was the SVC, nonetheless, results suggested that it was overfitting. Through this method, we inferred that our data might be impossible to properly separate in a linear way.

Ensemble models

The ensemble of classifiers is a technique that seeks better predictive performance by combining the predictions from multiple models. In fact, it is proven that an ensemble of classifiers is typically more accurate than a single classifier, which makes this a very important tool in machine learning [1]. The ensemble methods we used were the following:

1. **Bagging Classifier**

The Bagging Classifier fits base classifiers one at a time to random subsets of the original dataset, and then aggregates the individual predictions (either by voting or by averaging) to provide a final prediction. We started by comparing the F1-scores for three bagging classifiers with different base algorithms, namely **KNN**, **Decision Trees** and **Logistic Regression**. We also got the F1-scores for the base algorithms without bagging, concluding that, although the difference was not very significant, they benefited from bagging. Then, we tested some more hyperparameters individually, such as *max_samples* (the number of samples to draw from X to train each base estimator), *max_features* (the

number of features to draw from X) and *bootstrap* (whether samples are drawn with replacement or not). Once that was done, we added the best values to a parameter space and applied Grid Search.

2. Random Forest

Random Forest is an ensemble classifier model that combines multiple decision trees. A standard categorization decision tree takes several factors, converts them into rule questions, and then, given each element, either produces a judgement or takes another factor into consideration. If there are several choice rules, for instance, if the threshold to make a decision is unclear or we add new sub-factors for consideration, the decision tree's outcome may become uncertain. Random Forest, by combining numerous diverse trees, can provide a more powerful prediction rather than relying on a single tree.

In terms of parameters, we started out by testing what would be the best number of estimators, that is, the number of trees used. To do that, we used a function to plot the different F1 scores for the different number of trees, and tested it with 10, 20, 50, 100, 200 and 300 trees, concluding the last was the best option. Then, we used the same technique to conclude that it would be better to not apply bootstrap, meaning that the whole dataset is going to be used to build each tree. Concerning the max depth, the best option was to leave to the default value None, which makes it so that nodes expand until all leaves are pure or until all leaves contain less than *min_samples_split* samples. To finalize, we calculated once more the feature importance to make sure that our feature selection was the best possible, and the independent features that the model deemed to be the most important were '*Checkup_More than 3 years'*, followed by '*High_Colestrol'* and '*Mental_Health'*.

3. Extra Trees

Although Random Forest is the most well-known ensemble method based on Decision Trees, Extra Trees also has some value for classification problems, mostly because it is less computationally costly than Random Forest. The two algorithms are almost identical, and so are the parameters, with the biggest difference being the selection of cut points in order to split nodes. Random Forest chooses the optimum split, whereas Extra Trees chooses it randomly, and therefore we expected the F1 score to be slightly worse, which was, in fact, the case.

4. AdaBoost

AdaBoost is a boosting algorithm which, theoretically, can be used to significantly reduce the error of any learning algorithm that consistently generates classifiers whose performance is a little better than random guessing [3] (so called weak learners). Unlike random forest, it does not use full grown trees to make decisions, but rather stumps, which are very simple trees with just one level. Another difference is that the trees are fitted sequentially, with each one trained on, in a sense, the mistakes from the previous tree, meaning that the order in which the trees are presented is very important. AdaBoost can work with several base estimators, the default being decision trees. Therefore, we tried seeing how logistic regression would perform against the default, with the latter achieving a better F1 score. We then tested some more parameters individually, such as the number of estimators or learning rate, before adding them to a parameter space to be scanned using Grid Search.

5. **Gradient Boosting**

Gradient Boosting is another tree-based algorithm but, as opposed to Random Forest, it builds the trees one at a time, in a way that the subsequent trees try to reduce the errors of the previous ones.

6. **Voting Classifier**

A voting classifier trains on a collection of several models and forecasts an output based on the class with the highest likelihood of being the output. To predict the output class based on the highest majority of votes, it merely averages the results of each classifier that was passed into the voting classifier. In our case, we used Random Forest, Extra Trees, and Gradient Boosting, since these were some of the models that produced the best results.

7. **Stacking Classifier**

The Stacking Classifier consists of stacking the outputs of multiple individual estimators. The different performances obtained by the models are then combined to make a final prediction, often leading to an improved F1-Score performance. We created two Stacking Classifiers, one combining the two best models, which performed better, and another combining the three best models. The results produced by the two classifiers did not constitute an improvement on our current best performance, previously obtained by the Random Forest Classifier.

Assessment

The assessment of our models was mainly based on their respective confusion matrixes and individual metrics (accuracy, precision and recall), as well as the **F1 score**. This is a harmonic mean between precision and recall that uses the following formula: $F1 = \frac{2 \text{ x } precision \text{ x } recall}{precision + recall}$. A **confusion matrix**, on the other hand, reflects the outcome of a classification problem by diving the results into four possibilities: the True Positives (1's predicted correctly), True Negatives (0's predicted correctly), False Positives (1's predicted wrongly) and False Negatives (0's predicted wrongly). Then, based on these four possibilities, we can compute performance metrics, such as the ones previously mentioned.

Accuracy corresponds to the percentage of cases correctly predicted in the total number of cases. It must be used with caution because it can give faulty results when dealing with imbalanced datasets, however, as previously stated, is not our case. **Precision** reflects how many of our predicted positives were actually positive. When the cost of a false positive is very high and the cost of a false negative is low, precision is an effective evaluation metric to use. Finally, **recall** reflects how many of our true positives were identified as such. If a model does not produce false negatives, it will have a recall of 1.

Now that we understand how each of the individual metrics works, we can conclude that, on their own, they all have some limitations. For that matter, it is important to use a more robust metric to evaluate our models, and that is where the F1-score comes in, incorporating both precision and recall. With this metric, we can limit both false positives and false negatives as much as possible. Therefore,

we decided to use it as the ultimate model performance evaluator in the project, optimizing for it as much as possibly, namely on the Grid Search and Randomized Search algorithms.

Once we were done evaluating each model individually, we decided to do a final comparison between all of them, using the F1 score once again, but this time with cross-validation, a method for testing machine learning models that involves training models on different subsets of the input data and then comparing the results. Here, we applied **Repeated Stratified K-Fold** as the cross-validation splitting strategy. Then, for a visual analysis of our results, we plotted the means and the standard deviations for each of our model's F1 scores. After carefully examining the results, we found that, as anticipated, ensemble methods produced, on average, higher F1-Scores than other algorithms, both for the training dataset and the validation dataset, with the model coming out on top being the **Random Forest**.

Deployment

In order to deploy our model and make the final predictions on the unseen test data, we first had to replicate all the steps we had made on the training and validation sets. This meant performing all the transformations, creation of variables, scaling, and feature selection once again. Then, we just used the model that had the best performance, Random Forest, to make predictions for the labelling of the unseen data.

Conclusion

Through the development of this project, we were able to both consolidate the materials learned during the Machine Learning course, as well as apply knowledge obtained through self-study. Although challenging, it was enriching, allowing us to discuss and investigate certain topics more thoroughly.

The project was not, however, without its setbacks. A major limitation we faced was lack of computing power, as seen when we had to opt for Randomized Search when testing different hyperparameters in our Neural Networks model because it was taking an excessive amount of time when using Grid Search, a solution that traded off accuracy for a faster runtime. Another limitation was the lack of data in the training dataset, which only included 800 records. Although the provided dataset was balanced, with few missing values and outliers, for more complex models to make more accurate estimators, they require huge amounts of quality data. This is a noteworthy issue, since we're dealing with a model that must be able to accurately make predictions regarding people's health.

Finally, the obtained results gave us some meaningful insights. For instance, ensemble methods are deemed to produce better predictions and perform better than any single model, an assumption that was corroborated by our results, which were superior for the ensemble models used. Ultimately, we concluded that the best predictive model for identifying a patient that is likely to suffer from the Smith Parasite for this particular dataset was Random Forest, with a prediction score of 1 on Kaggle.

References

[1] Opitz, D., & Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research 11,* 169-198

[2] Geurts, P., Ernst, D. & Wehenkel, L. (2006). Extremely randomized trees. Mach Learn 63, 3–42 https://doi.org/10.1007/s10994-006-6226-1

[3] Freund, Y., & Schapire, R.E. (1996). Experiments with a New Boosting Algorithm. *International Conference on Machine Learning*.

Annexes

Figures

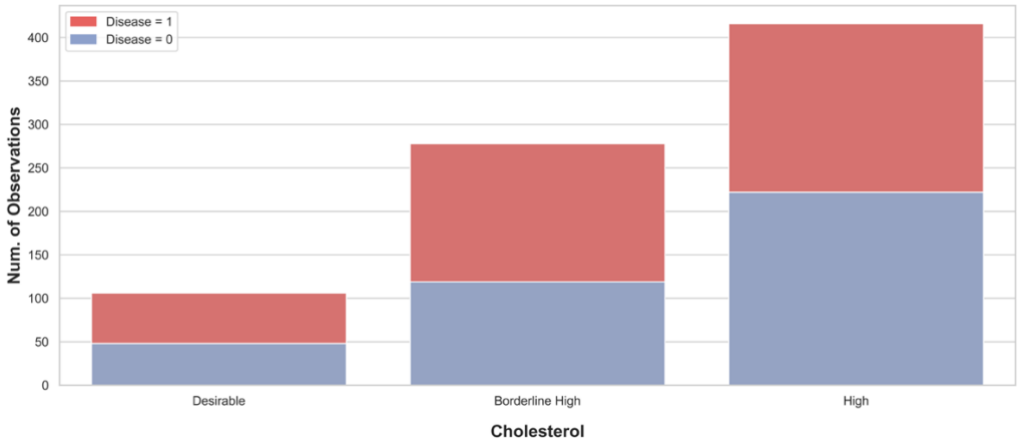


Figure 1 – Cholesterol Stacked Bar Chart

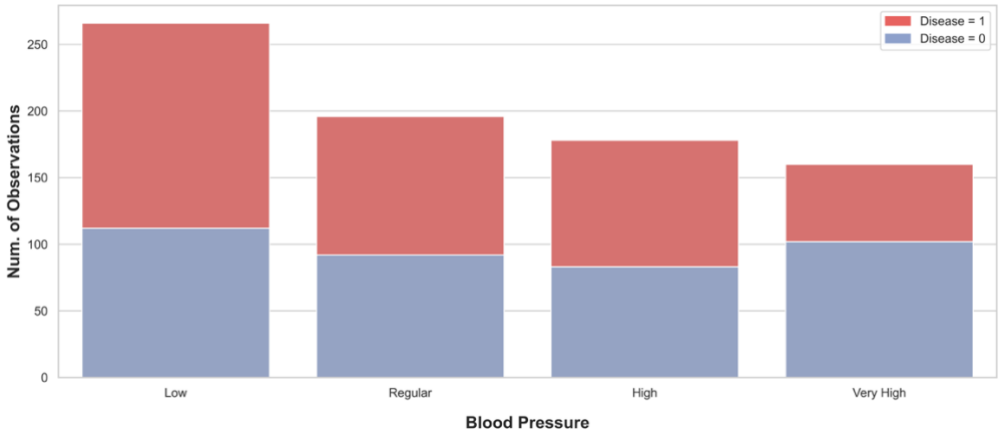


Figure 2 - Blood Pressure Stacked Bar Chart

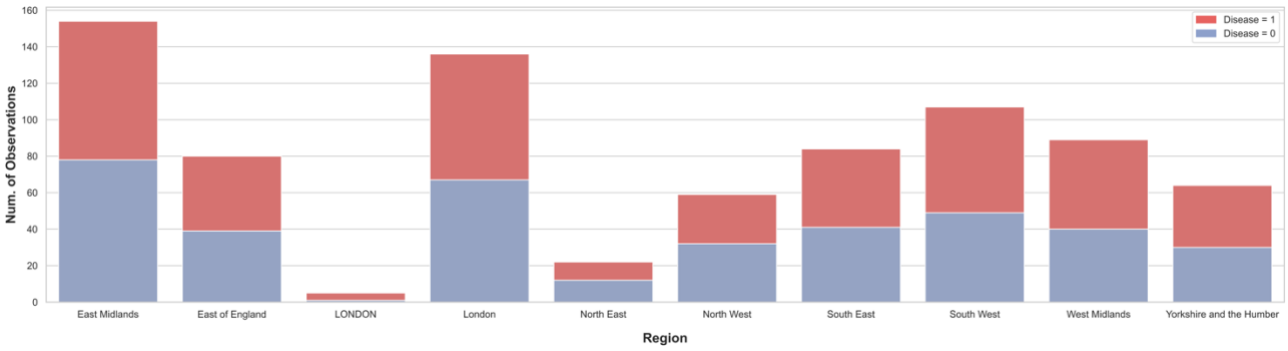


Figure 3 – Region Stacked Bar Chart

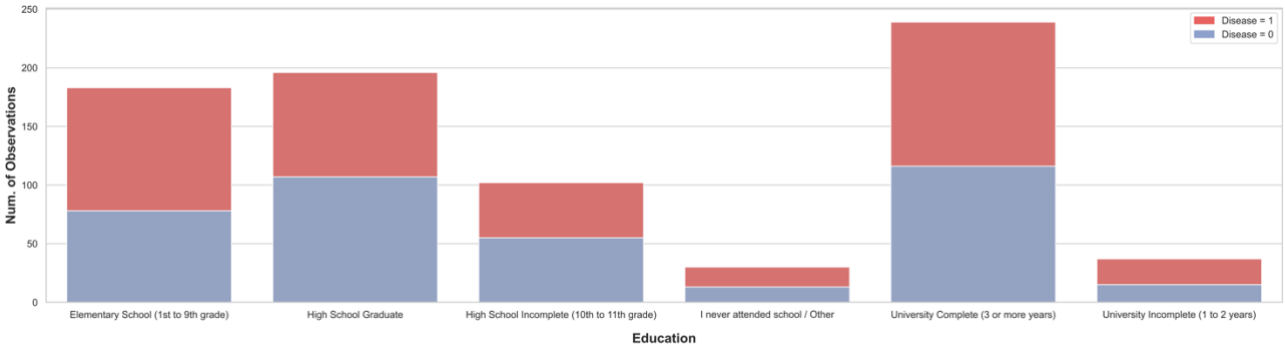


Figure 4 - Education Stacked Bar Chart

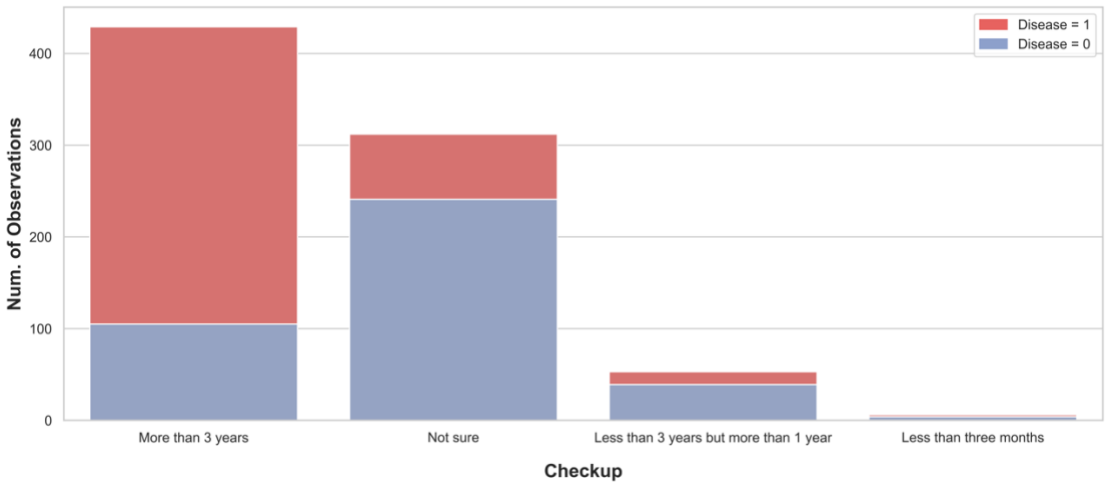


Figure 5 - Checkup Stacked Bar Chart

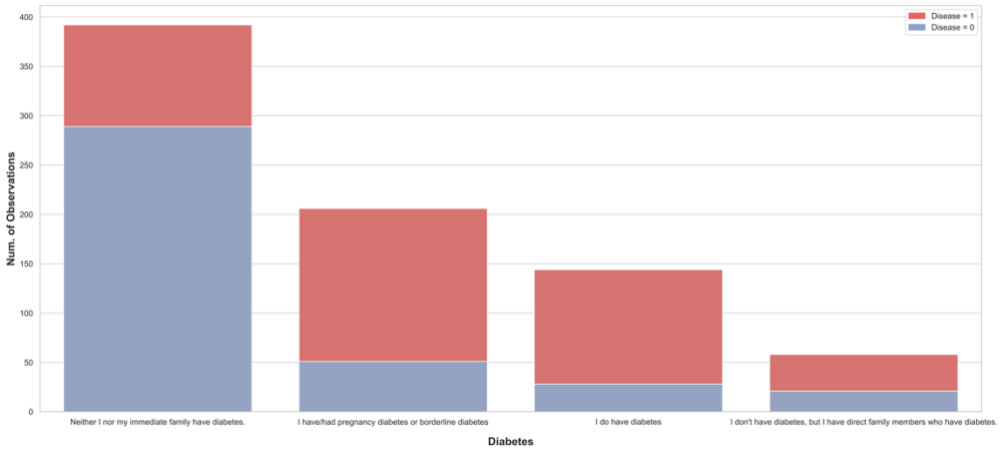


Figure 6 - Diabetes Stacked Bar Chart

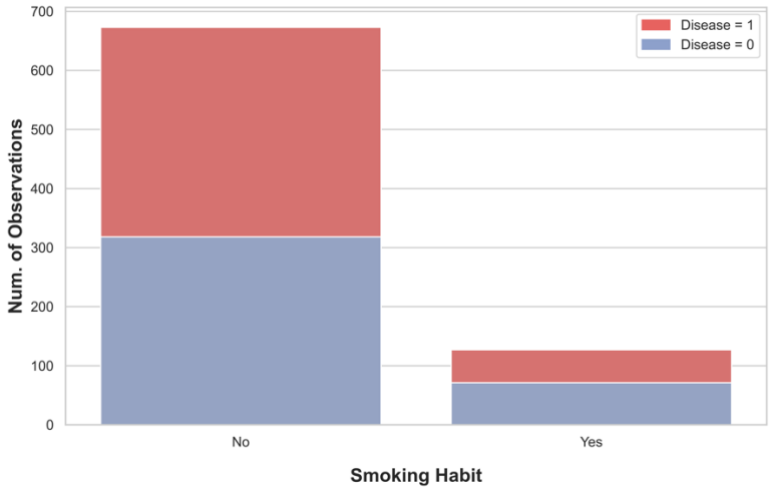


Figure 7 - Smoking Habit Stacked Bar Chart

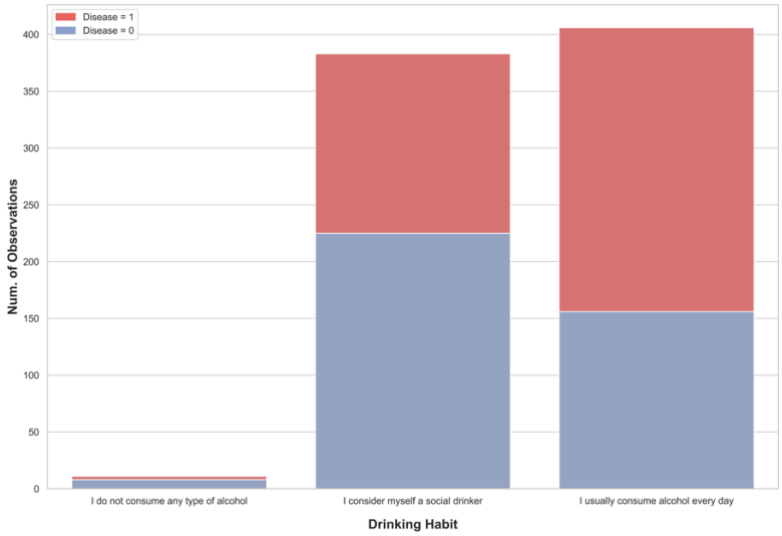


Figure 8 - Drinking Habit Stacked Bar Chart

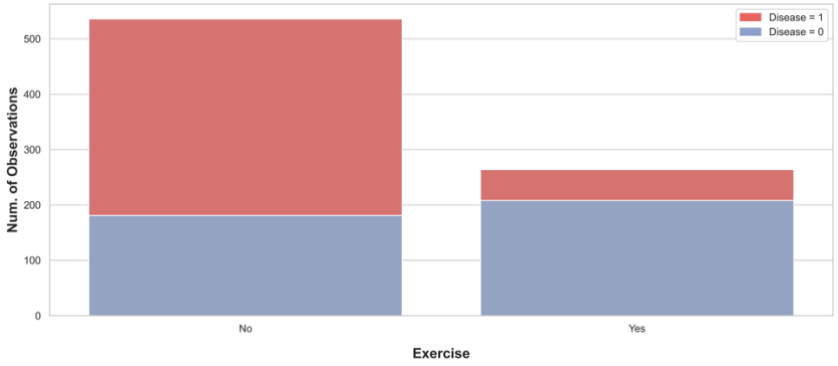


Figure 9 - Exercise Stacked Bar Chart

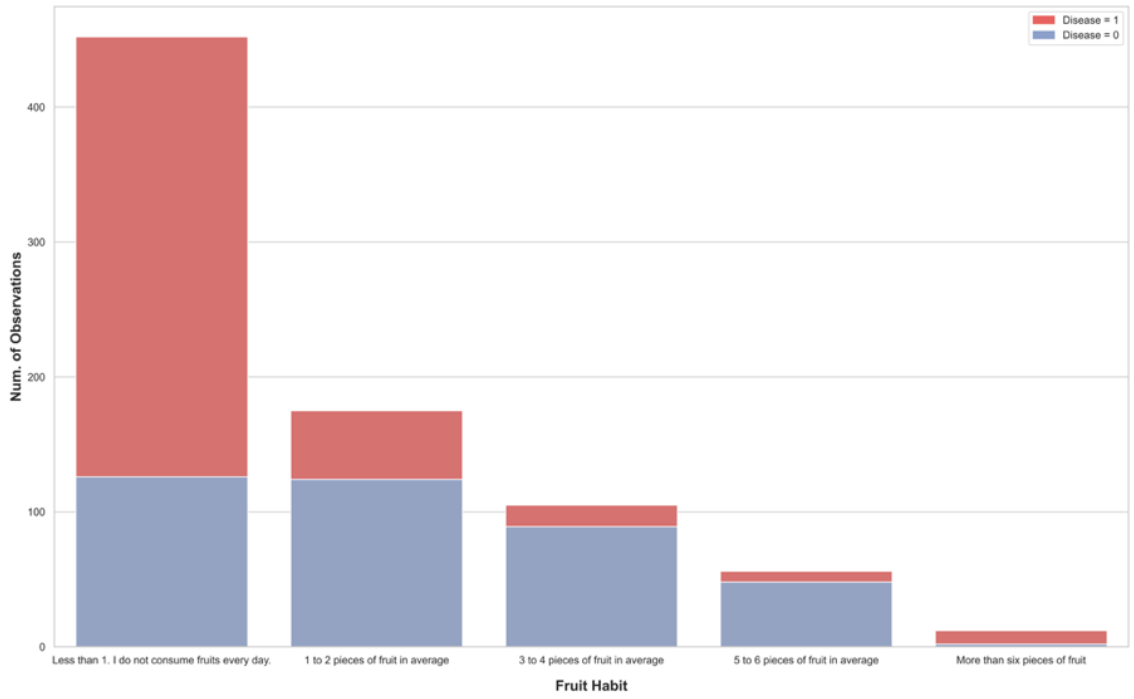


Figure 10 - Fruit Habit Stacked Bar Chart

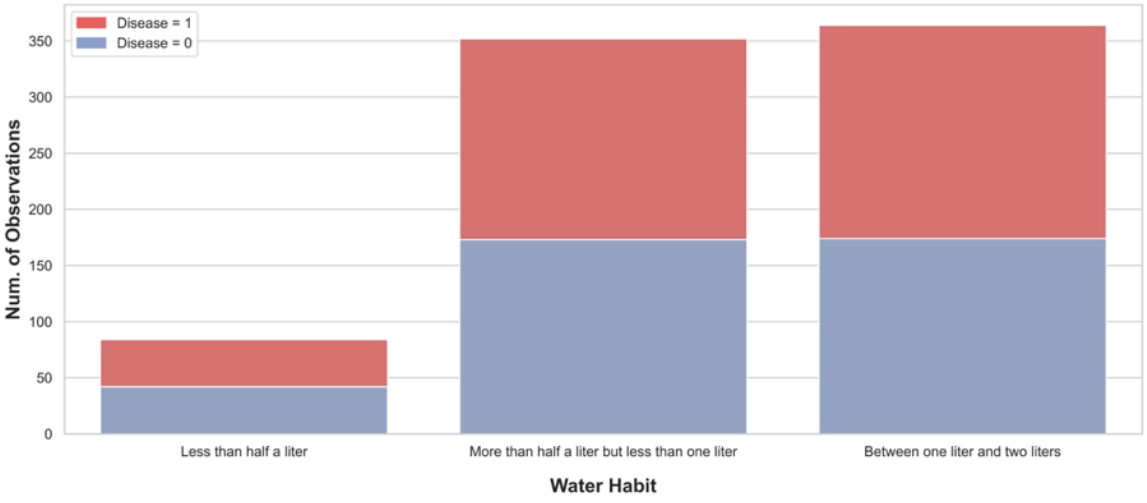


Figure 11 - Water Habit Stacked Bar Chart

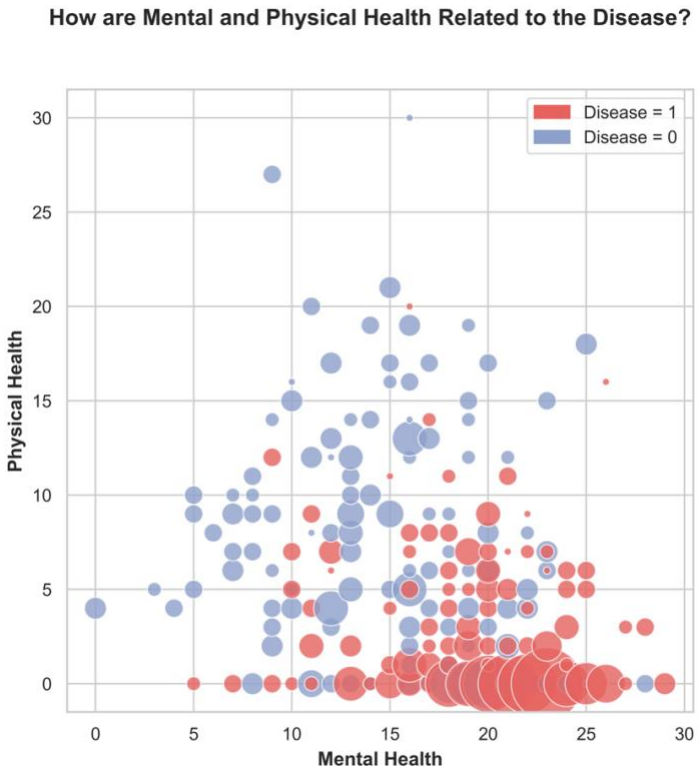


Figure 12 - Bubble Chart Reflecting How Mental and Physical Health may be Related to the Disease

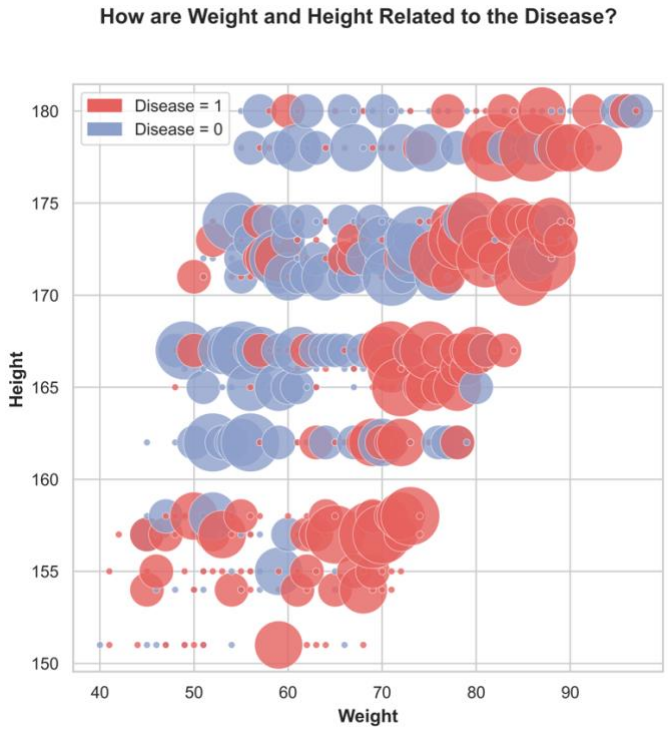


Figure 13 - Bubble Chart Reflecting How Weight and Height may be Related to the Disease

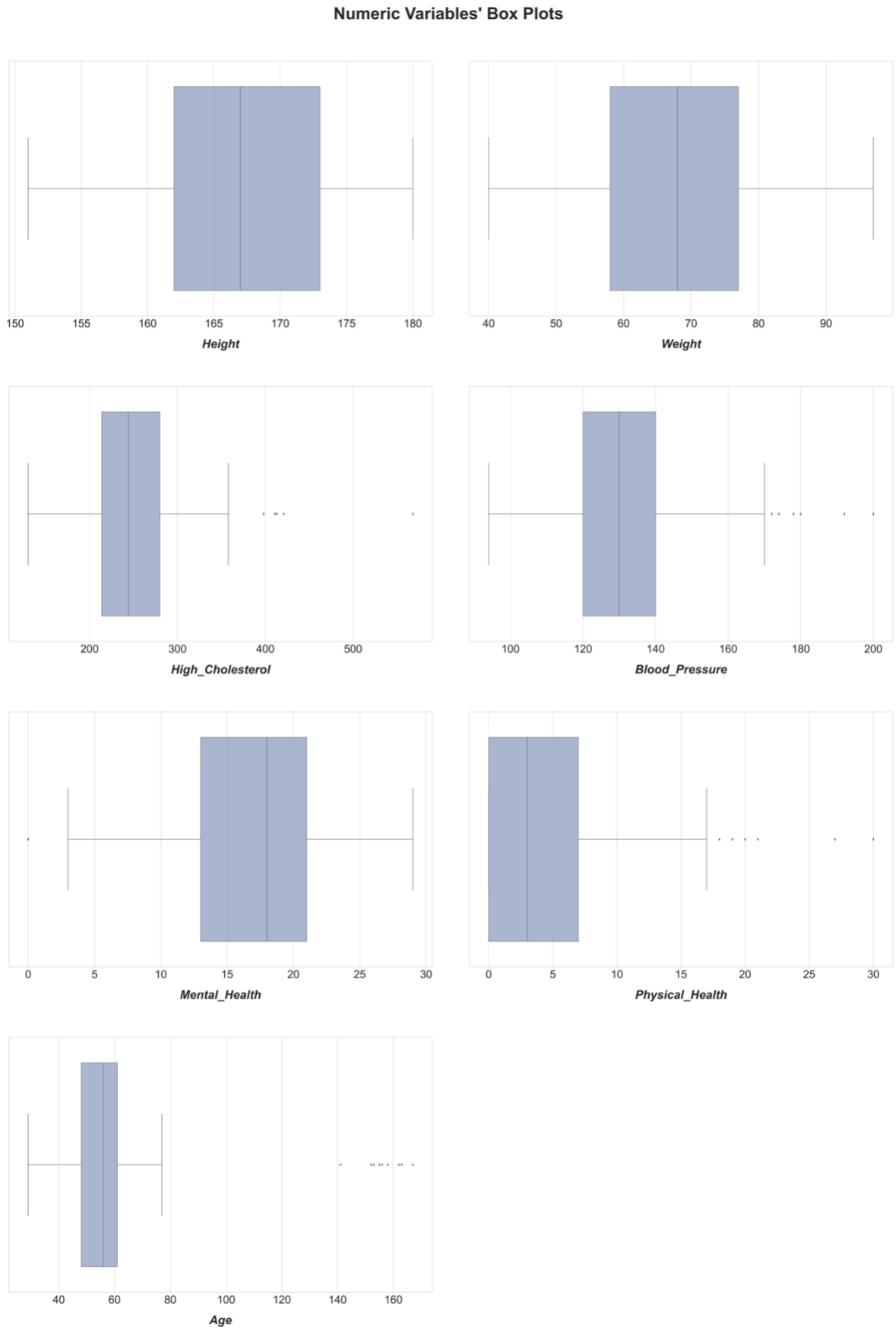


Figure 14 - Numerical Variables Boxplots

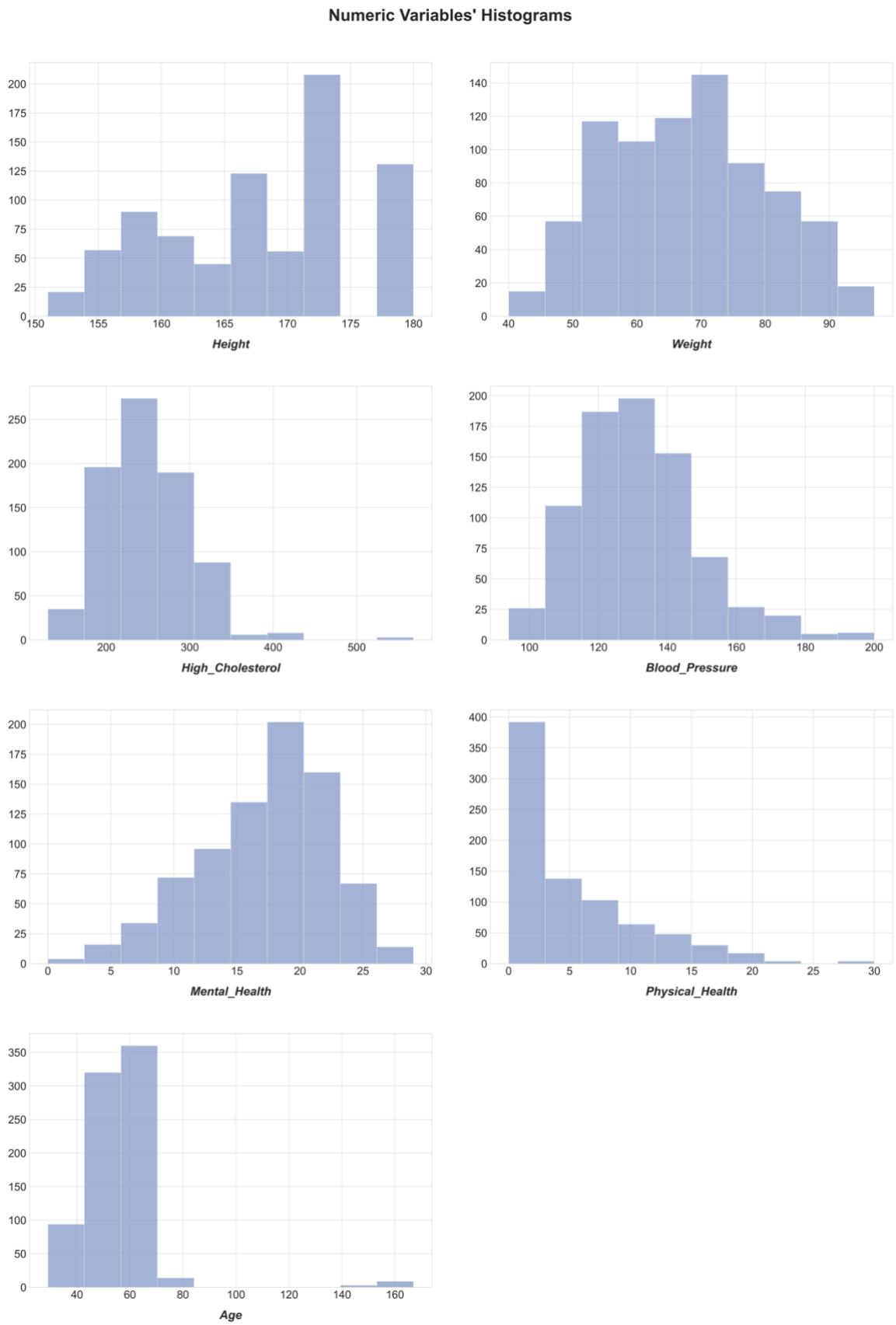


Figure 15 - Numerical Variables Histograms

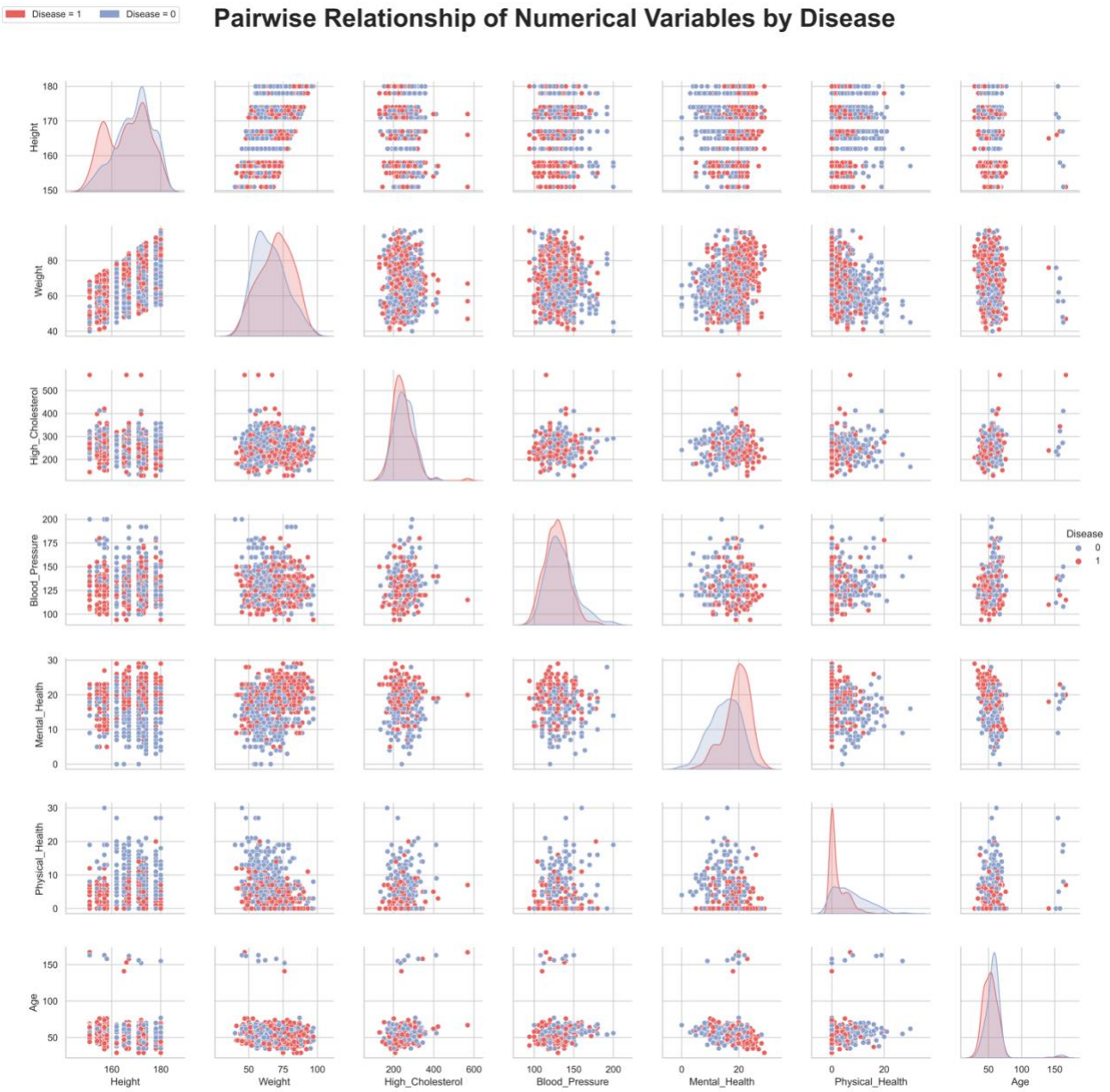


Figure 16 – Pairwise Relationship of Numerical variables By Disease

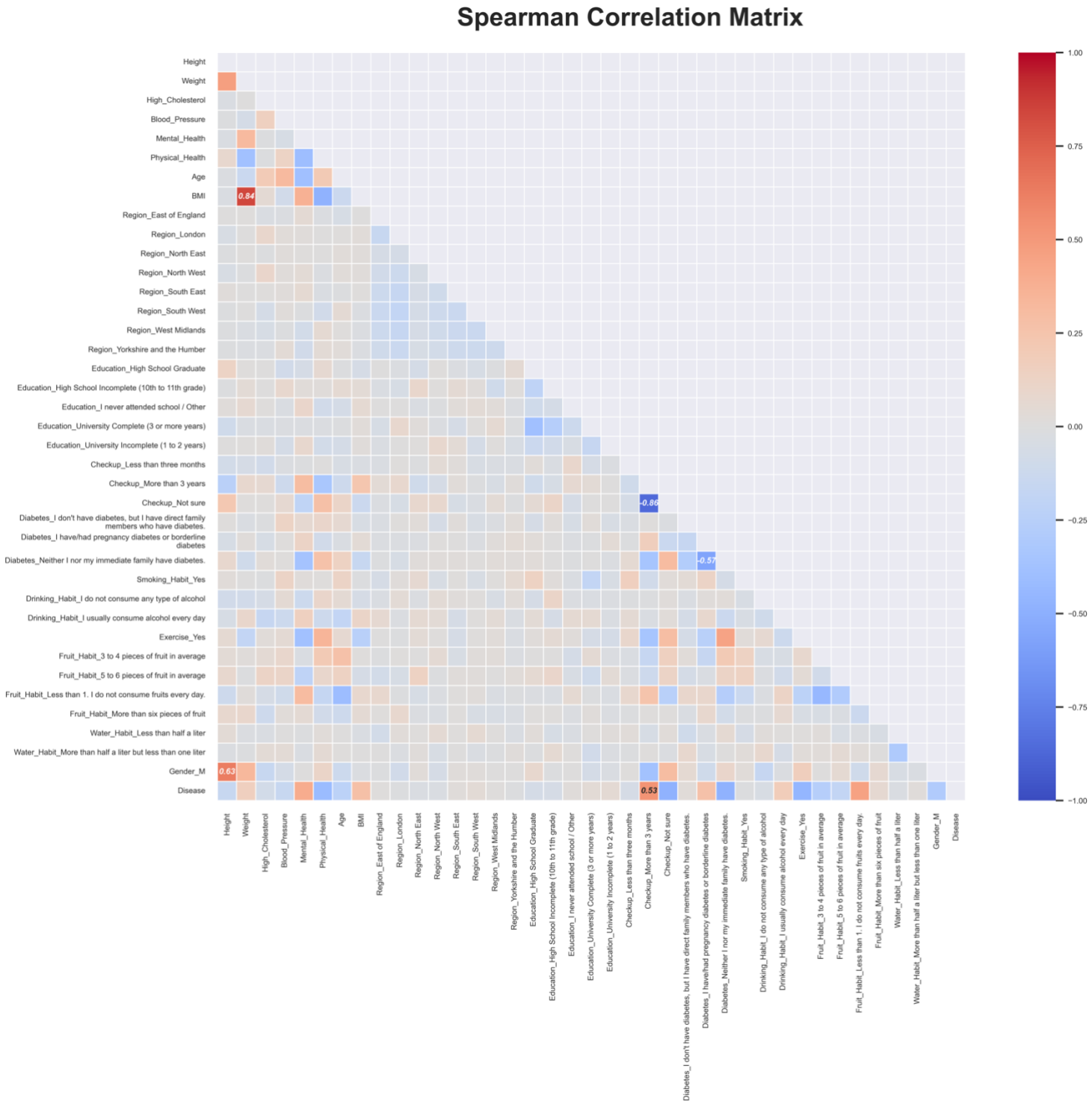


Figure 17 - Spearman Correlation Matrix Before

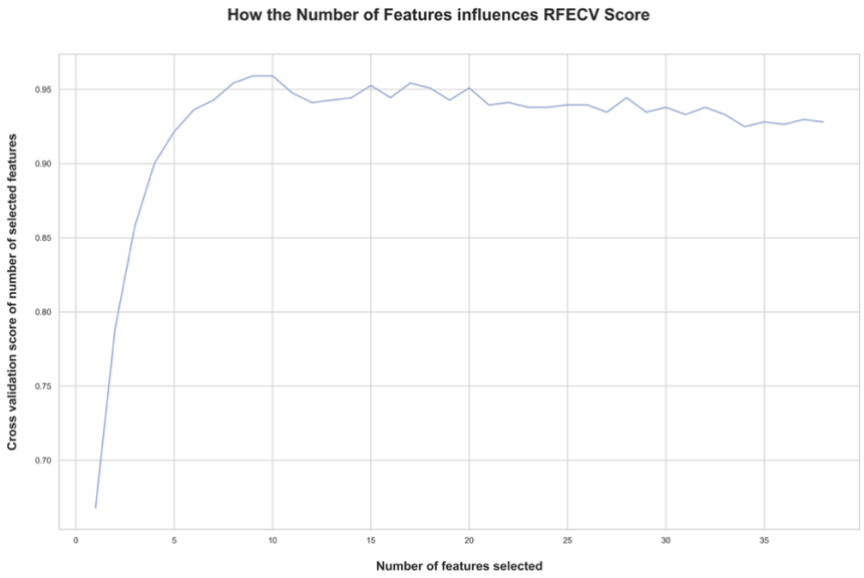


Figure 18 - How the Number of Features Influences RFECV Score

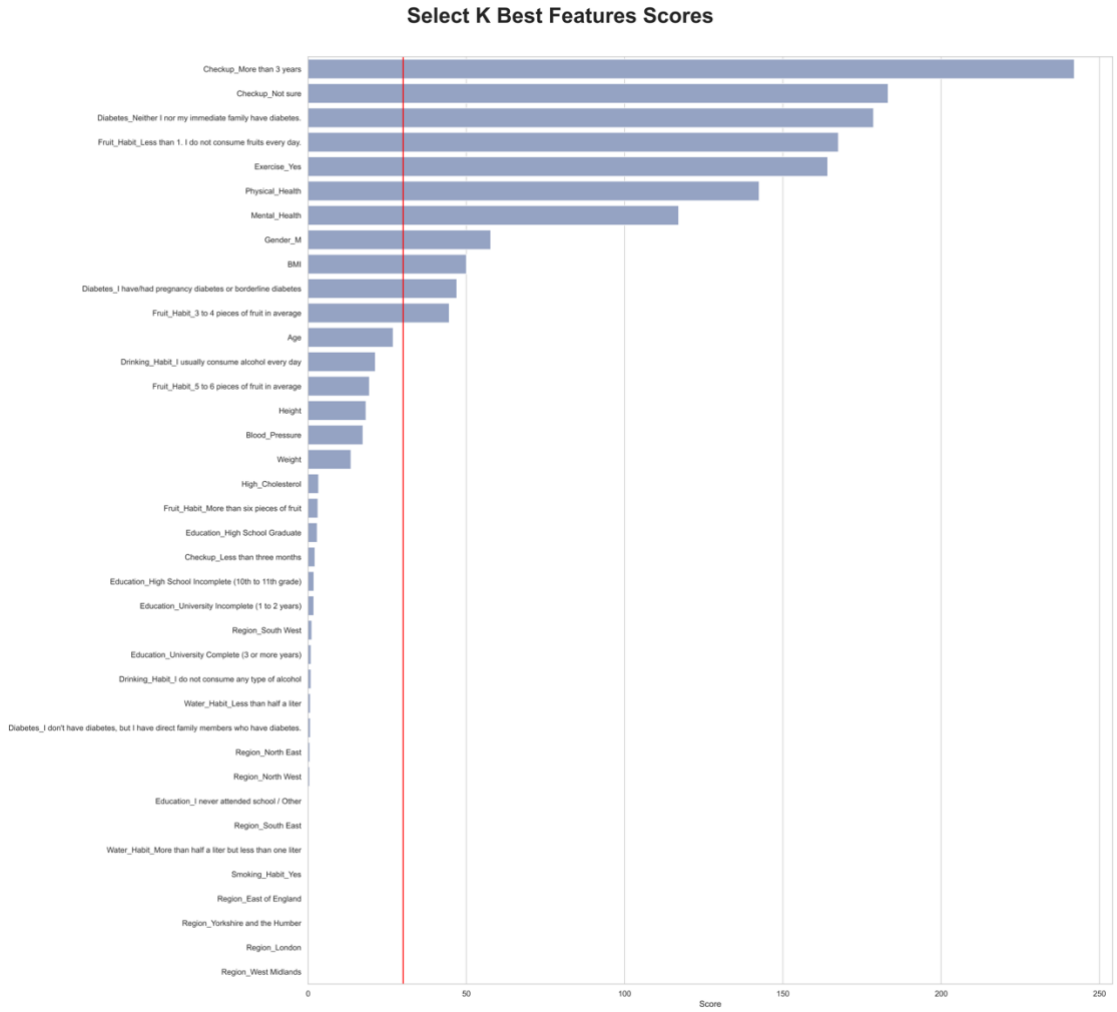


Figure 19 - Select K Best Features Scores

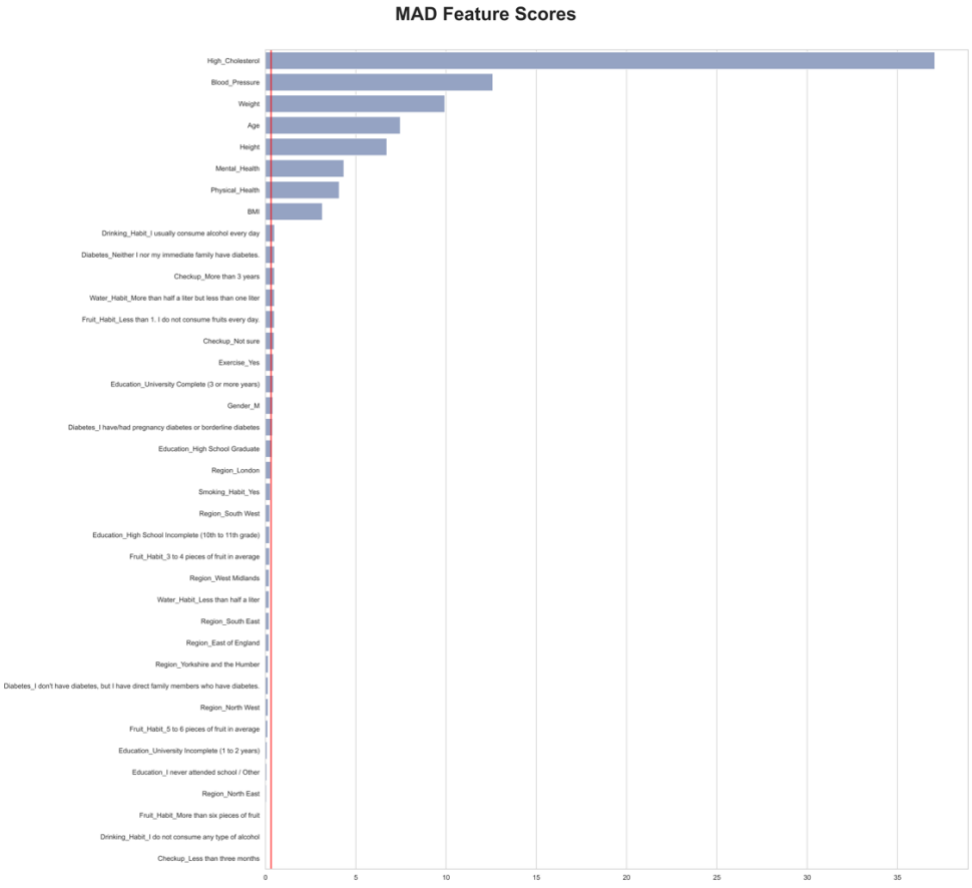


Figure 20 - MAD Feature Scores

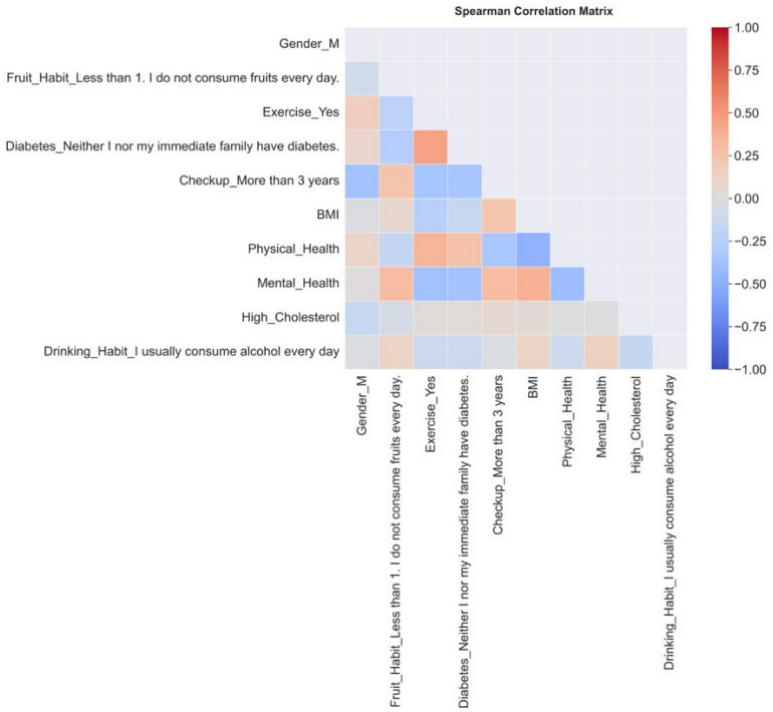


Figure 21 - Spearman Correlation Matrix After

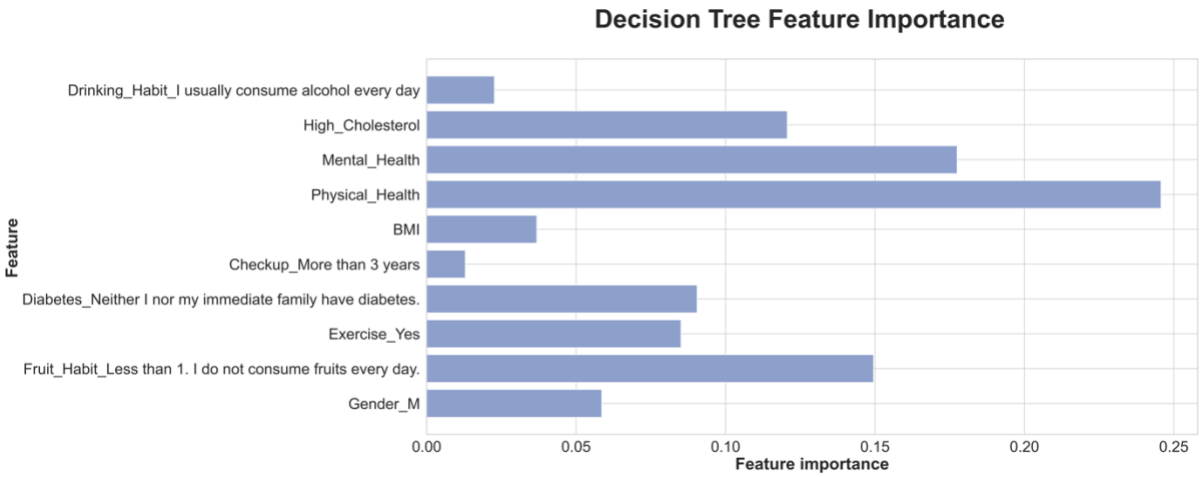


Figure 22 - Decision Tree Feature Importance Bar Chart

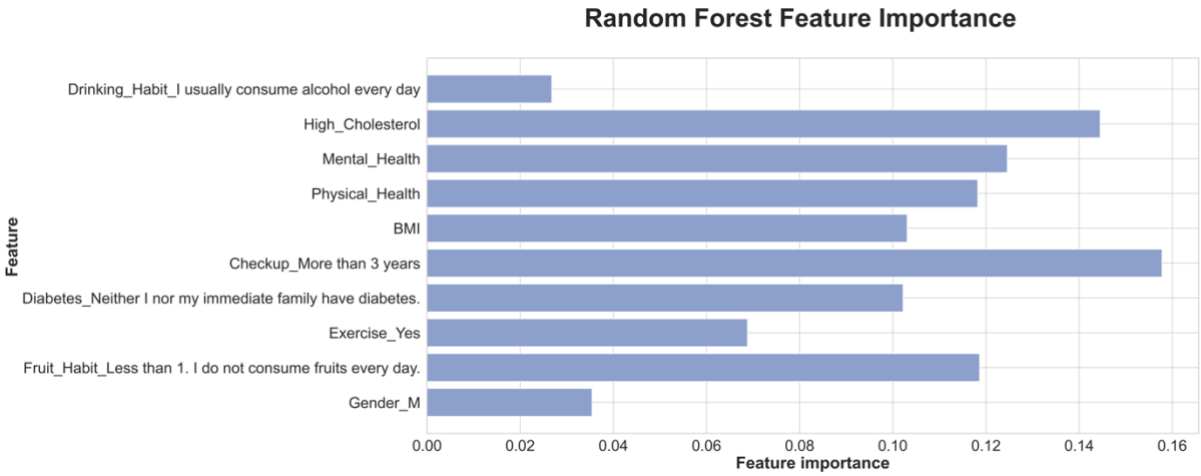


Figure 23 - Random Forest Feature Importance Bar Chart

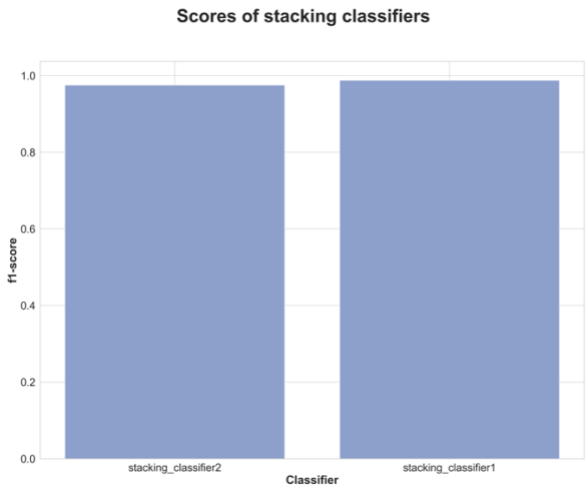


Figure 22 - Scores of Stacking Classifiers

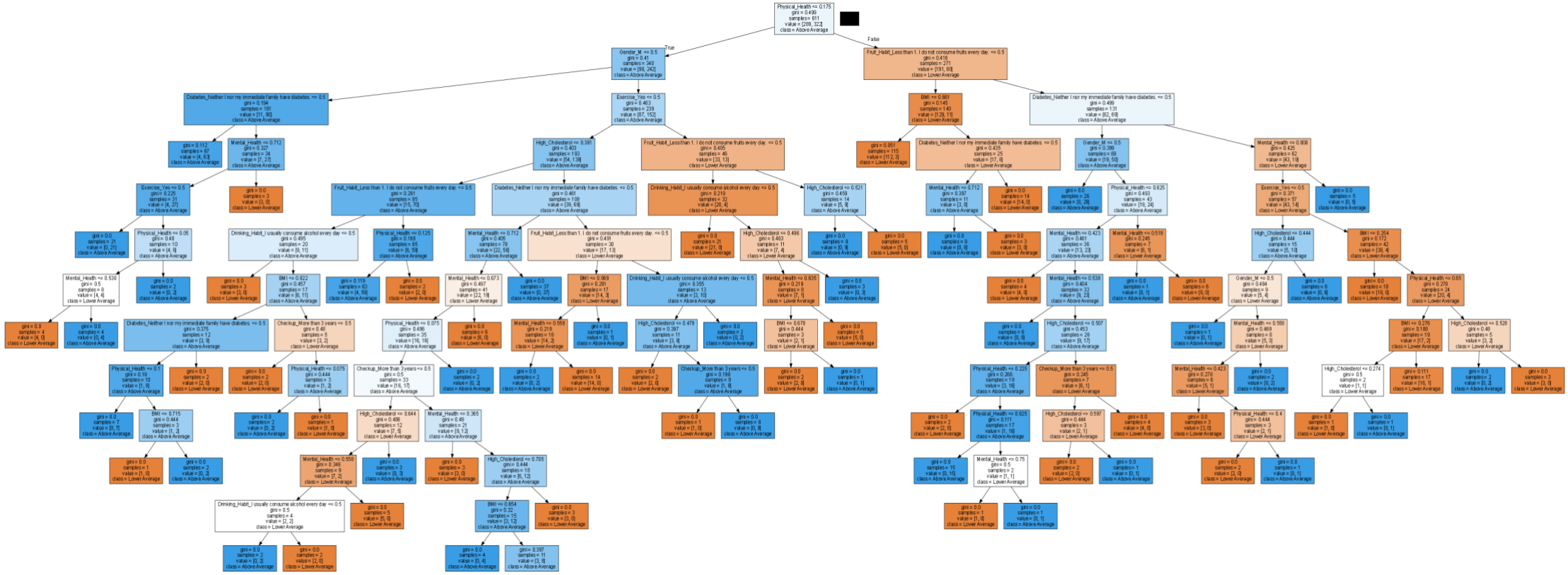


Figure 23 – Decision tree

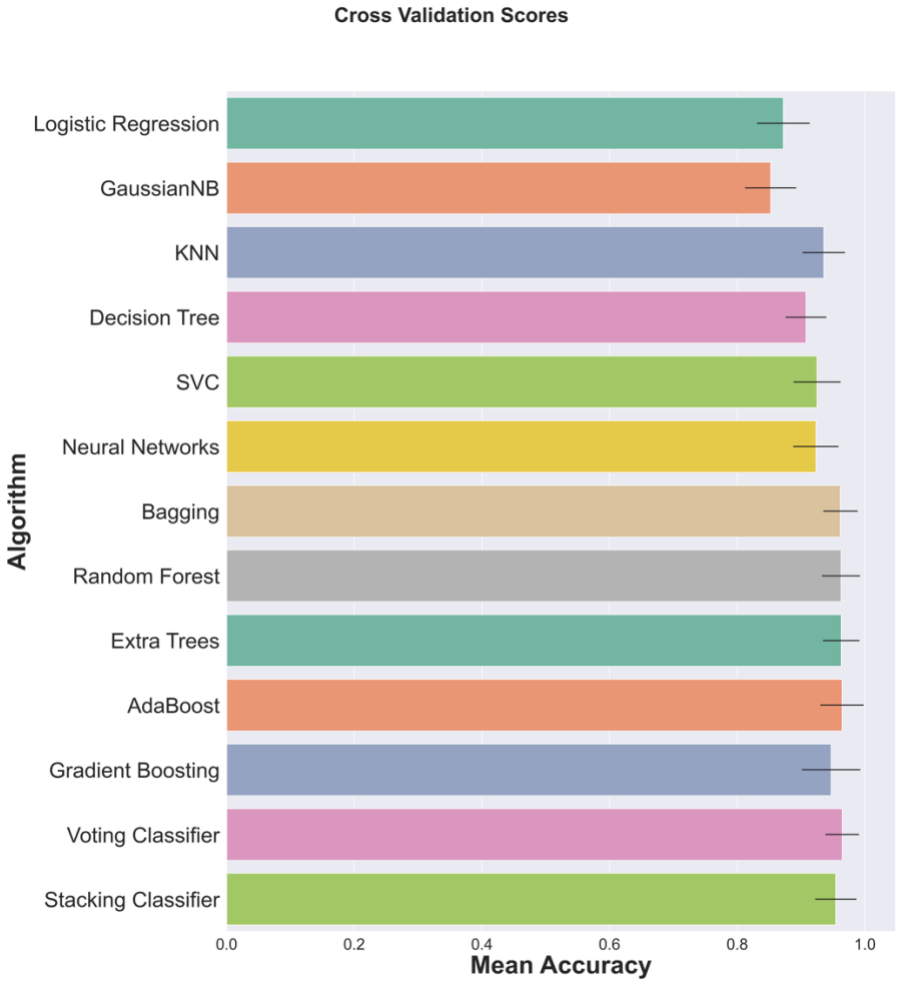
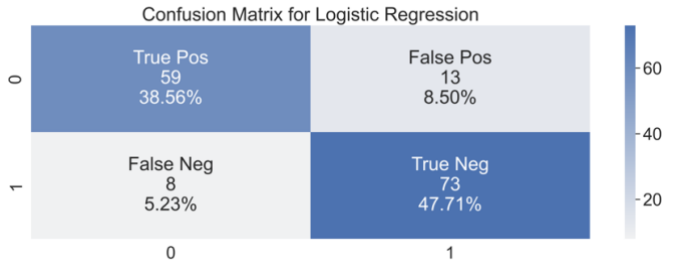
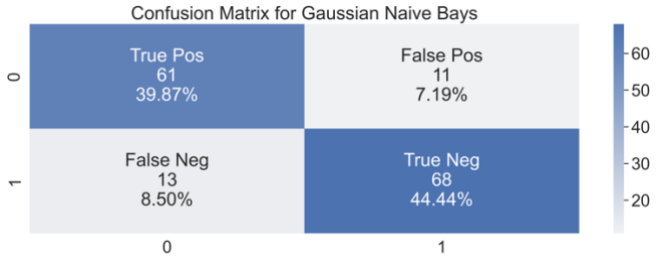
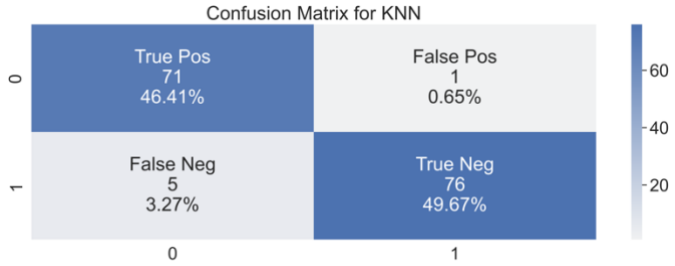


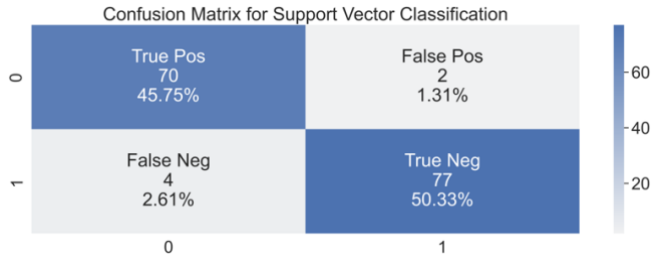
Figure 24 - Final Model Scores

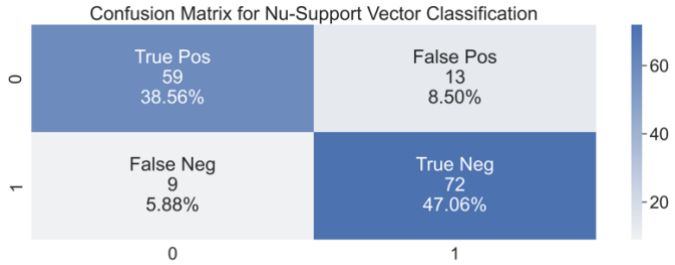
Confusion Matrices

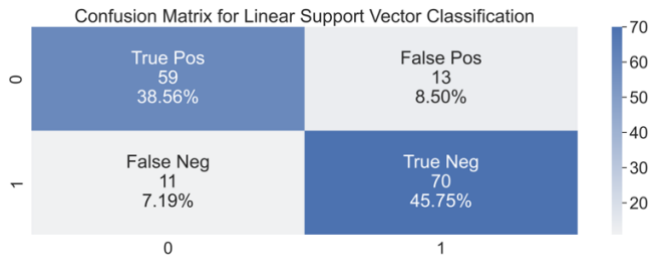


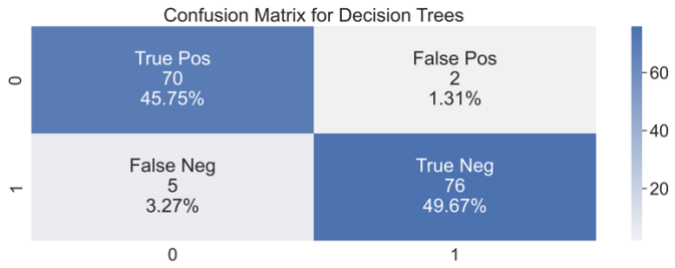












Tables

Table 1 - Feature Selection by Model

Feature	*Logistic Regression*	*Random Forest*	*RFE*	*MAD*	*KBest*	*Total*
Fruit_Habit_Less than 1. I do not consume fruits every day.	True	True	True	True	True	5
Exercise_Yes	True	True	True	True	True	5
Diabetes_Neither I nor my immediate family have diabetes.	True	True	True	True	True	5
Checkup_More than 3 years	True	True	True	True	True	5
Checkup_Not sure	True	True	True	True	True	5
Physical_Health	False	True	True	True	True	4
Mental_Health	False	True	True	True	True	4
Gender_M	True	False	True	True	True	4
Diabetes_I have/had pregnancy diabetes or borderline diabetes	True	False	True	True	True	4
BMI	False	True	True	True	True	4
Weight	False	True	True	True	False	3
High_Cholesterol	False	True	True	True	False	3
Height	False	True	True	True	False	3
Fruit_Habit_3 to 4 pieces of fruit in average	True	False	True	True	False	3
Drinking_Habit_I usually consume alcohol every day	True	False	True	True	True	3
Blood_Pressure	False	True	True	True	False	3
Age	False	True	True	True	False	3
Water_Habit_More than half a liter but less than one liter	False	False	False	True	False	1
Smoking_Habit_Yes	False	False	True	False	False	1

Education_University Complete (3 or more years)	False	False	False	True	False	1
Education_High School Graduate	False	False	False	True	False	1
Diabetes_I don't have diabetes, but I have direct family members who have diabetes.	False	False	True	False	False	1
Water_Habit_Less than half a liter	False	False	False	False	False	0
Region_Yorkshire and the Humber	False	False	False	False	False	0
Region_West Midlands	False	False	False	False	False	0
Region_South West	False	False	False	False	False	0
Region_South East	False	False	False	False	False	0
Region_North West	False	False	False	False	False	0
Region_North East	False	False	False	False	False	0
Region_London	False	False	False	False	False	0
Region_East of England	False	False	False	False	False	0
Fruit_Habit_More than six pieces of fruit	False	False	False	False	False	0
Fruit_Habit_5 to 6 pieces of fruit in averare	False	False	False	False	False	0
Education_University Incomplete (1 to 2 years)	False	False	False	False	False	0
Education_I never attended school / Other	False	False	False	False	False	0
Education_High School Incomplete (10th to 11th grade)	False	False	False	False	False	0
Drinking_Habit_I do not consume any type of alcohol	False	False	False	False	False	0
Checkup_Less than three months	False	False	False	False	False	0