Machine Learning Project

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

THE SMITH PARASITE

GROUP 09:

Cláudia Alves Fernandes Rocha (r20191249)

Susana Teresa Dias (20220198)

Inês Nascimento(r20170746)

Felix Gayer (20220320)

Felix Sami Gaber (20221385)

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Index

Machine Learning Project Report: The Smith Parasite

ABSTRACT: There's a new disease in England - found by Dr. Smith-, in which the most common symptoms are tiredness and fever, but some infected people are asymptomatic. This virus is associated with post-disease conditions such as loss of speech, confusion, chest pain and shortness of breath. This Machine Learning project was made to analyze the data of the patients, and accordingly predict if the patient will or will not be infected by The Smith Parasite. The data was explored, preprocessed and feature engineered in several different ways until we obtained satisfactory results on our baseline model using distinct classifiers. After improving and analyzing the models, the Histogram Gradient Boosting Classifier turned out to be the better performing algorithm for this problem, with a F1-Score of 1 on Kaggle.

I. Introduction

In this project, our team aims to develop a machine learning model that can accurately predict whether a patient is sick or not from the recently discovered disease. This is a crucial problem to solve, as early detection and diagnosis of a new disease can greatly improve the chances of successful treatment and recovery for the patient.

The conditions for transmission of the disease are still unknown, and there is no certainty as to what causes a patient to have the disease or not. To tackle this problem, we were provided three files for the train, and three for the test, which we combined and got one dataset for train, to use to perform training and validation in our model, and one dataset destinated to test. These files had information about the health, habits and sociodemographic characteristics of 800 and 225 patients, respectively. The model will learn from the train data to identify patterns and characteristics that are indicative of the disease and use this knowledge to make predictions on the test data.

The performance of our model will be evaluated using a variety of metrics and fine tuning it to improve its balance between recall and precision (F1) and reliability. The determining metric will be the F1 score on Kaggle.

II. Exploration

The team started the exploration part by using the method *info(),* to know the data types of the variables we had (Figure 1), as well as understanding which variables had missing values, where 13 missing observations in "Education" were found. To check this, the *isna()* method was made, and we also checked this for other types of representations ("", "?","-", " ", "null", "NK").

Next, the *describe()* method was executed for both categorical and numerical variables. With this, our team saw that the "High_Cholesterol" and "Blood_Pressure" features had some unusual values, for example, 568 and 200, so we assumed that these variables may have been subject to some measurement errors, since the typical range values for these features are [0;200] and [80;120], respectively.

Thus, “Birth_Year” has a minimum value of 1855 (distant from the value in the first quartile, 1961), which may indicate the presence of outliers, in other words, an observation that lies an abnormal distance from the quartiles. Additionally, given the Smith Parasite is a recently discovered parasite, it is expectable our patients to be alive in 2022, so we may exclude that observation. Regarding “Physical_Health”, it is possible that this variable has outliers, due to the fact that the maximum value is 30 and the third quartile is only 7, which is quite distanced. Also, there are only two numerical variables (“Height” and “Weight”) without outliers - we can state this in Figure 7, available in Annexes.
After checking, we concluded that there were no duplicates in the dataset. However, there are two people with the same name (Mr. Gary Miller) that presented very different information in other variables, leading to the conclusion that it was not a duplicate value either, but only a coincidence - see Figure 3 in Annexes.
Furthermore, the team checked that the provided train dataset is not perfectly balanced and does not have significant discrepancy – only 22 more "1” observations than "0”, corresponding to having the disease or not, respectively, in the dependent variable. While checking for inconsistencies, the word “London” was found written in different ways. In the “Education” variable, there are categories that, in our perspective, could be redundant and increase cardinality: given that a person’s educational qualifications for someone who did not graduate High School, is “Elementary School”, we believe the terms "Elementary School (1st to 9th grade)" and "High School Incomplete (10th to 11th grade)" may have same meaning. Similarly, the educational qualifications for someone who did not finish University, is equivalent to a High School Graduate.
Moving forward to data visualization was an important step, since it made the group better understand and make sense of our data, quickly identify trends, patterns and insights, and effectively check for abnormal distributions. To perform this step, and since the metric and non-metric features are visualized differently, we first started by dividing out features into metric and non-metric. For the metric features, we made histograms (figure 4), box plots (figure 5), their pairwise relationship among each other (figure 6), the box plots segmented by the target (“Disease”) in figure 7, their pairwise relationship segmented by the target (figure 8), among others. By the Pearson correlation heatmap (figure 9), we concluded that there are not highly correlated variables, but we noticed that “Height” and “Weight” have a Pearson’s correlation of 0.51. About the non-metric features, its absolute frequencies (figure 10) and bar plots with these features segmented by the target (figures 11 - 17) were made. These visualizations helped us understand a bit of the behavior of our patients: most of the patients in this study do not smoke, however, only a few patients do not drink alcohol at all. Additionally, a lot of patients do not exercise. All in all, we can find good and bad habits from patients to have or not have a risk behaviour to get the disease.

III. Pre-processing

Once analyzed our data and understood the type of variables we had, as well as what information they provided, we proceeded to the data pre-processing part, which our group decided to split into two sections: “Data Cleaning” and “Feature Engineering and Selection”. For the first, some problems found in the previous EDA[1], like outliers, missing values, and unwanted observations will be solved. Regarding the second part - "Feature Engineering and Selection" – we will create new features, encode, and scale our categorical and numerical variables, respectively, and finally select the features we want to keep for the Modelling step. We should not fail to mention that in every step of this part, a copy to the previous dataset will be made, in order not to make any permanent changes in the dataset we are working on, and to be possible, later, to test models on different versions of the dataset, and, subsequently, choose the ones that performed best.

Data Cleaning

The data cleaning part started by partitioning the data. Given that we already had a test dataset, we only had to split the train dataset into train and validation, which we defined the percentages as 85% and 15%, respectively. To do so, the *train_test_split()* method was used, and randomly split our data. It is important to emphasize that, to improve data quality, the following changes in this section will be performed in the three partitions according to the information and patterns in the train dataset, with the exception of the outlier treatment, since, in order to make the final prediction on Kaggle, this last adjustment can remove rows.

Regarding the "LONDON" and "London" values found in the EDA step, all values of the "Region" feature were capitalized, in case there are others containing the same mistake in the test dataset.

As previously mentioned, in our understanding, the feature "Education" may have some redundant categories. Due to this, the "High School Incomplete (10th to 11th grade)" observations were converted into "Elementary School (1st to 9th grade)" observations, as well as the "University Incomplete (1 to 2 years)" observations to "High School Graduate" observations.

Taking a deeper look into the "Drinking_Habit" variable and its frequencies, we recognized that there are only 11 patients with the value "I do not consume any type of alcohol". Similarly, there are only 6 patients who answered "Less than three months" on the "Checkup" question. As expected, when visualizing the bar plots for these features, we concluded that their proportion is very small when compared to the total observations of the dataset (figures 12 and 16). For that reason, and since they could add noise, those values were converted into missing values.

Through the interpretation of boxplots in Figure 18, the team could identify the presence of outliers and decided to take two approaches. On one hand, we applied a Manual Filter on the extreme values of the "Birth Year," "High Cholesterol," and "Blood Pressure" variables, since the rest of them, even though being outliers, represent real behavior. So, from the "Birth_Year" we decided to remove any values smaller than 1920 and from the variables "High_Cholesterol" and "Blood_Pressure", any values higher that 500 and 185, respectively. Although feature "Physical_Health" also presents outliers, we decided to keep these because, once again, they represent real behavior. This method removed around 2,5% of the observations of our training dataset.

On the other hand, we performed the common IQR[2] Method, that sets up a "fence" outside of the first and third quartiles, which are, respectively, the values under which 25% and 75% of the observations

[1] Exploratory Data Analysis
[2] Inter Quartile Range

fall, considering any values outside these as outliers and, subsequently, deletes them from the dataset. With this method, around 8% of the observations were removed from the training dataset. This percentage is considered a high loss of information; therefore, we proceeded this step with a dataset with the Manual Filtering Method and another without the outlier treatment.

Considering the missing values identification in prior steps, in Table 1 we can see the number of observations that are zeros and missing values, and in Figure 19 we can find the MSNO[3] matrix, a graphical representation of missing data in training. Although two of the variables contain many "0", this is a value considered a "real zero", so they should not be treated as missing observations.

Different executions were made regarding the treatment of missing values, such as deleting the rows, applying a simple imputer with mode and mean and, lastly, iterative imputer. Firstly, we deleted the missing values in a copy of the datasets, and the percentages of deleted data were around 4% for the dataset without outliers and 6,3% for the one with outliers. Secondly, another alternative was applied: a simple imputer with the mode for the non-metric features and the mean for the metric features. Although there are no missing data on metric features on the training dataset, this is unknown regarding the test dataset, so we will use the mean of the feature on the training data to impute missing value on the test dataset. Thirdly, we tried the iterative imputer strategy that models a column with the missing values - target variable - as a function of other features - predictor variables - and uses that estimate for imputation which has the advantage of avoiding possible bias.

Comparing the performed methods and taking into consideration the relatively low number of missing values we have, our group decided to proceed with the simple imputer and apply this to the train, validation, and test datasets, always using the mean and the mode of the training dataset.

Feature Engineering and Selection

Besides the original features of the dataset, and to simplify the interpretation, we formed a new variable named "Age" by subtracting the "Birth_Year" from the current year. Additionally, a new column named "BMI"[4] was made, as well as the binary column "Gender" by extracting the gender from the prefixes of the patient names. With these new features created, we dropped the original columns "Name", "Birth_Year", "Height", and "Weight" directly afterwards due to redundancy. The information about these new variables can be checked on Table 2 in the Annexes. Then, new visualizations of this cleaned data were performed, in order to check the distributions and the plots with all the pre-processing made – see figures 20 and 21 for the histogram of metric features and absolute frequencies of the non-metric.

Moving on to the feature transformation, we apply encoding through three different approaches: the Label, Ordinal and the One-Hot encoder. In fact, the Label Encoding is more adequate to features with many categories and where the order does not matter, which is not the case. Besides resorting to the Ordinal Encoder when the dataset is in the presence of ordinal features, it is also used when they do not want to have numerous binary variables just to represent one single initial variable with multiple

[3] MSNO stands for "missingno", and it is a bar plot showing the number of non-null data values in each column of the dataset

[4] BMI stands for Body Mass Index, and corresponds to the weight in Kg, divided by the Height in meters squared

categories, which is how the One-Hot Encoder performs. After applying the different types of encoding to the copies of the datasets, and taking into consideration the previous insights, the team decided to move on with the One-Hot Encoding, finding this the most appropriate.

Not only for the comparability of the different features in our models and algorithms, but also for future analysis and visualization it is beneficial to perform scaling. As previously said, machine learning algorithms just perceive numbers. As a result, they develop the implicit assumption that higher ranging numbers have some sort of superiority, and they begin to play a more significant role in the training of the model and for that reason it is needed perform scaling. We performed the Standard scaler - which assumes a Normal Distribution of the metric features, the MinMax scaler – suited for when a distribution is not Gaussian and the standard deviation is low, and, finally, the Robust scaler - which is better suited for datasets with outliers. The final decision was to proceed with the three versions in order to see the model performance, also taking into consideration the assumption that the MinMax scaler would be the most suited one.

At this point, we proceeded to feature selection using a dataset version without outliers, the metric features scaled with MinMax scaler and the categorical features with their original values, that is not encoded. We started this step with the filter methods: after checking the variance of the metric features, and the Spearman correlation of the metric features with the target, we concluded that these tests did not return any special insights for feature selection. On the other side, according to the Pearson correlation and with a threshold of 0.2, the features we should keep are: "Mental_Health", "Physical_Health", "BMI'" and "Age". The result of the K-Best (*SelectKBest* method) using the ANOVA F-value and *k=4* as a parameter so this test could return us the best 4 features, was the same as the Pearson correlation test. According to the Chi-Squared test, which calculates the chi-square among each categorical variable and the target, and selects the variables with the best results, the outcome was to discard the following features: "Region", "Education", "Smoking_Habit", "Water_Habit".

The RFE[5] with logistic regression as estimator and F1 as the scoring parameter, showed that the three most important features were "High_Cholesterol", "Mental_Health", and "Physical_Health". Finally, regarding embedded methods, we ran both Lasso and Ridge regression. Lasso regression uses a L1 norm regularization term that results in some of the regression coefficients being set to 0 during modeling, limiting the number of variables used. Ridge regression, on the other hand, uses an L2 norm regularization term that results in all regression coefficients being reduced in size, but not set to 0. Both yielded very similar results indicating that the features "Age" and "BMI", respectively, should be removed from the model – see figures 22 and 23. After we have performed all feature selection methods with partly different outcomes, we decided to keep going with two different sets of features: the ones that the Pearson Correlation and the Chi-Squared tests outcomes returned to keep (for the metric and non-metric variables, respectively); and the ones that we interpreted from the performed tests – Tables 3 and 4.

Before moving on to the Model Assessment part, we had different datasets. What varied among them was how they were scaled, the way the feature selection was performed, and whether the datasets had

[5] Recursive Feature Elimination

outliers or not. With that said, it is important to refer that we had eight different kinds of datasets. Among these eight, six of them had the feature selection done applying tests, analyzing the results and choose the features to drop (let's call this “manually”), and two of them had the feature selection done by only the Pearson Correlation and Chi-Squared tests mentioned above. These two were scaled with MinMax scaler, one dataset had outliers, and the other did not. In the six whose filter selection was done “manually”, three of them had outliers, and three of them did not, and in each of these three, one was scaled with Robust scaler, one with standard scaler and one with the MinMax scaler – Table 5.

IV. Modelling

First Model Selection Round - One Dataset, Multiple Classifier

In this first round - the Model Selection - we will only deal with the X_train_MF_OH_SS[6] dataset. We only used one dataset in the beginning in order to receive a feeling about which models performed better in general. On this dataset, 18 models were applied: DecisionTreeClassifier, RandomForestClassifier, ExtraTreesClassifier, LogisticRegression, LinearDiscriminantAnalysis, GaussianNB, GaussianProcessClassifier, KNeighborsClassifier, BaggingClassifier, AdaBoostClassifier, GradientBoostingClassifier, HistGradientBoostingClassifier, SVC, LinearSVC, RidgeClassifier, SGDClassifier, PassiveAggressiveClassifier, QuadraticDiscriminantAnalysis. In fact, the models that were previously unknown are explained in more detail in the Annexes.

We created a function that returns all parameters for the model performance evaluation separately for all mentioned models and stores them in a dataframe- Tables 6-13. These parameters included the true positive rate, the true negative rate, the AUC[7] score, the accuracy, precision, recall, and the F1 score.

AUC score is helpful in assessing the model since it measures how well the predictions are classified, rather than their absolute values – being scaling invariant. In addition, it measures the quality of the model's predictions regardless of the classification threshold chosen - being classification threshold invariant. However, these benefits can also turn to disadvantages: scale invariance is not always desirable, for example, sometimes it is really needed well-calibrated probability outputs, and AUC does not provide any information about them. Classification threshold invariance is also not always desirable, mainly in cases where the costs of false negatives and false positives vary widely, which can be critical to minimize some sort of classification error. Especially in the detection of unexplored diseases with unknown impact on long-term human health, these false classification errors should be considered (Hajian-Tilaki, 2013). This is the reason why we also consider other parameters such as true negative rate (*specificity*) in our decision making.

To actively prevent overfitting in the model, we calculated all parameters directly for the train and validation datasets. In our subsequent analysis, we were thus able to ensure that these values did not deviate too much from each other. Also, we started to plot some visualizations to get a general overview of the comparison between the different models in accuracy, as well as the F1 score. This data frame, as well as the visualizations, can also be seen in the Annexes in Figures 24 - 26.

[6] Dataset without outliers, encoded with the One-Hot encoder and scaled with the Standard scaler
[7] AUC stands for Area Under the ROC Curve

In terms of F1 score, the models who performed better were the DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, GaussianProcessClassifier, HistGradientBoostingClassifier, ExtraTreesClassifier, KNeighborsClassifier, BaggingClassifier. Since there are several models with very good scores and we are still making some adjustments in the next round, we continue not only with the best model, but with these eight best models for the second model assessment round.

Second Model Selection Round & Dataset Selection

In contrast to the First Model Assessment Round, where we only used one dataset to find the eight best models and their scores, we now consider all eight possible variations of datasets (Table 5) created in the preprocessing and combine them with only the best models from the first model selection round.
Again, a dataframe for evaluating the measures of each of the eight models is created so that we get all possible combinations between the eight different datasets – Tables 6-13. The team marked the best scores in each column and row so that identifying the good scores would be easier. What we noticed directly, for example, was that the models generally performed better when outliers are in. It is understandable that there are many combinations that returned a very good F1 score (*f1_valid*) for the model as a whole. This parameter, *f1_valid*, thus indicates the score of the model - trained with the train dataset and evaluated with the validation dataset. Additionally, there is *f1_train*, which was trained with the train dataset and evaluated with the train dataset. It is important to mention again that this value is only used to compare it to the actual *f1_valid* model value. The difference (*f1_diff*) between *f1_valid* and *f1_score* indicates how much the model was overfitted. Consequently, the higher *f1_diff* is, the more our model is overfitted. It was therefore desirable to obtain a *f1_diff* score as low as possible.

V. Assessment

Comparing the performance of the models selected before, the team realised that, although the obtained scores were already high regarding the prediction in validation dataset, these could be improved. Therefore, we took a deeper look into the datasets with higher scores in each model and other measures such as mean, deviation, overfitting score and the f1 score for each of the train and validation datasets for each of the best 18 combinations, in Table 14.
Starting with Decision Trees and Extra Trees classifiers, we performed a grid search with several parameters such as *max_depth*, which is a parameter that determines the maximum number of levels that the tree can have and can help prevent overfitting. However, the results returned predictions on validation with a significantly lower F1 score, as well as on the train datasets. Therefore, our group decided to proceed with the default ones, even at the cost of the model not being able to generalize so well.
Similarly, with the Random Forest and Gaussian Process classifier, the grid search of parameters would worsen the performance of the model regarding the train and validation datasets, so the final choice was also to proceed with the default parameters in each.
Regarding KNeighbors, we calculated the best number of neighbors to use, that resulted in *k=1* for both. This caught our attention because in this case it means that only the single closest neighbor is used to make the prediction and leads to a more complex decision boundary. In in other words, the line that

separates the different classes is non-linear and more flexible, capturing well the structure of data but also is more prone to overfitting. Nevertheless, since this was still the value of *k* that provided the best score in the three versions on the dataset, we proceeded with that one.

Lastly, with the classifiers Gradient Boosting, Hist Gradient Boosting and Bagging the use of grid search to find the best parameters was successful and we were able to get a improve the F1 score for both training and validation datasets.

For each combination of model and dataset created before, we test it with the test datasets and download each of them (downloads are in comments). With these results, we submitted them to Kaggle and checked which ones gave us the best scores. The winners were combinations 16 and 17 (Hist Gradient Boosting Classifier) with the following two datasets: X_test_MF_OH_MM and X_test_OH_MM (see table 5). Between these two models, we chose the second one, because its training dataset does not contain outliers and is, therefore, better preprocessed than the first one.

VI. Conclusion

Throughout the project, we focused on finding the most effective preprocessing steps and algorithmic approaches for our data. Rather than attempting to implement every possible technique, we took a reversed approach by first creating a baseline model for general understanding of the data and then setting up a process structure to quickly develop a model with a score on train and validation. After completing the Exploratory Data Analysis, we prioritized various techniques based on their impact and compatibility with our data. Afterwards we gradually increased the score of our model by implementing and evaluating these techniques in the process until we achieved a satisfactory result in terms of generalization and F1 score. If necessary, we would have tried even more techniques such as rebalancing, bootstrapping, neuronal approaches, other outlier handling methods, cross-validation, or different imputing techniques.

With this project, we can state that we were able to develop a Machine Learning predictive model based on the sociodemographic, health, and habits information of 800 patients, and answer the question "Who are the people more likely to suffer from the Smith Parasite?", putting into practice the concepts taught in the Machine Learning classes, as well as facing a real-world problem.

We cannot fail to highlight the data exploration and pre-processing steps. These were, indeed, the most laborious and relevant parts of the project, so that we could understand and correctly interpret the variables and the data provided. We can say that our datasets, already split into training and validation, did not suffer major changes beyond those already mentioned above, such as cardinality problems, unwanted observations, outliers, missing values, and creation of new variables.

In conclusion, and after testing and comparing the models, our best model was the Histogram Gradient Boosting Classifier, which resulted on a F1-Score of 1 in Kaggle, having predicted correctly all the patients with the disease and not falsely identifying any patient that did not caught The Smith Parasite.

Finally, we believe that this project was essential, not only to assimilate the knowledge we learned in classes, but also to gain a deeper understanding of Machine Learning techniques and grow us as future data scientists, while dealing with a meaningful real-world problem.

VII. References

3.2 - Identifying Outliers: IQR Method | STAT 200. (n.d.). PennState: Statistics Online Courses. https://online.stat.psu.edu/stat200/lesson/3/3.2

sklearn.impute.SimpleImputer. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

sklearn.impute.IterativeImputer. (n.d.-b). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html

Roy, B. (2022, August 11). All about Feature Scaling - Towards Data Science. Medium. https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

Compare the effect of different scalers on data with outliers. (n.d.). Scikit-learn. https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html

sklearn.feature_selection.f_classif. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html

Recursive Feature Elimination — Yellowbrick v1.5 documentation. (n.d.). https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html

Kumarappan, S. (2021b, December 15). Feature Selection by Lasso and Ridge Regression-Python Code Examples. Medium. https://medium.com/@sabarirajan.kumarappan/feature-selection-by-lasso-and-ridge-regression-python-code-examples-1e8ab451b94b

Chamarthy, R. (2021, December 12). Interpreting machine learning model performance measures. Medium. https://medium.com/ibm-data-ai/interpreting-machine-learning-model-performance-measures-ef2138047b96

1.1. Linear Models. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/linear_model.html

1.5. Stochastic Gradient Descent. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/sgd.html
sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis — scikit-learn 0.20.4 documentation. (n.d.). https://scikit-learn.org/0.20/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html
sklearn.ensemble.GradientBoostingClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

sklearn.ensemble.HistGradientBoostingClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html

Brownlee, J. (2017, July 13). What is the Difference Between Test and Validation Datasets? *MachineLearningMastery.Com*. https://machinelearningmastery.com/difference-test-validation-datasets/

Brownlee, J. (2019, August 11). A Tour of Machine Learning Algorithms. *MachineLearningMastery.Com*. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Hajian-Tilaki, K. (2013). Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian Journal of Internal Medicine*, *4*(2), 627–635.

Ismiguzel, I. (2022, May 12). *Imputing Missing Data with Simple and Advanced Techniques*. Medium. https://towardsdatascience.com/imputing-missing-data-with-simple-and-advanced-techniques-f5c7b157fb87

VIII. Annexes

Self- Study

Iterative Imputer

The Iterative Imputer is a multivariate approach that uses all other columns (predictor variables) to impute values in the missing value column (target variable) that we are imputing. In this way, the predictor variables are predicting the main variable (also called the target variable).

For our estimator object, we chose BayesianRidge() as the default estimator to impute the missing values.

BayesianRidge() as a predictor that will repeat all possible combinations between all columns (also known as a round robin).

This iterative process will stop until reaching the maximum number of iterations by default 10.

The n_nearest_features equal to "None" means that all columns will be used. Since we do not have a very high number of columns, all columns will be used.

Also, in the imputation_order we also used the default "ascending", since we have a small number of variables with missing values, we will start to impute the variables with fewer missing values until reach the features with more missing ones, in this way the column "Drinking_Habit" is the last one seeing your data being imputed.

As it is possible to see and as mentioned in classes, we prefer to choose the default values when performing the models.

Gradient Boost Algorithm

The Gradient Boost algorithm is a machine learning algorithm used to build prediction models based on ensemble methods. This algorithm belongs to the class of boosting algorithms, in which multiple weaker models are combined into a stronger model to improve the accuracy of predictions.

The algorithm works by iteratively training weaker models and combining them into a stronger overall model. Each model that is trained attempts to correct the errors of the previous model and improve the predictions. This is done by the algorithm calculating the errors of the previous model and training the new models to reduce those errors.

To use the Gradient Boost algorithm, a base model must first be selected to serve as the foundation for the subsequent models. This base model is then trained and used to make predictions for the data. The errors of these predictions are calculated and passed to the next model, which is trained to reduce these errors. This process continues iteratively until the overall model is trained and can be used to make predictions for new data. The Gradient Boost algorithm has proven to be very successful and is widely used in various fields, especially regression and classification.

Concerning some criterion chosen for Gradient Boosting Classifier, the team chose to try the default “friedman_mse”, but also “squared_error” to be capable of assessing a good quality of the split. For loss function, the team tested the default, but also other options like “exponential” and “deviance”, in this way we could optimize the loss function and compare the results to check the best ones. Moreover, to perform better, the team decided to adjust the max_depth from default 3 to 5, it means that we add a slightly higher number for the size of the tree's nodes. Also, as an example, the team changed min_samples_split default from 2 to 5 (also, tested 4 and 6) to have just one more sample to separate a node itself. Finally, following the previous example, min_samples_leaf from the default 1 to 5 (also, tested 1 and 4), where we obey the leaf node to have at least more than one sample as its minimum, in this case, 5. Although, we tested more options through grid search including the default ones.

Hist Gradient Boost Algorithm

The Hist Gradient Boost and the Gradient Boost work similarly, as the name suggests. One major difference between the two algorithms is how they process the data they train. The Gradient Boosting Classifier uses a batch process where the entire training data is processed at once. The Hist Gradient Boosting Classifier, on the other hand, uses a piecewise process where the training data is divided into smaller batches and processed sequentially.

This piecewise process has several advantages over the batch process. First, it allows the algorithm to train and run faster because the data is processed in smaller batches and the memory requirements are smaller. Second, it allows the algorithm to better handle large and rapidly growing data sets, since it only processes a portion of the data at a time. Overall, the Hist Gradient Boost and the Gradient Boosting differ in their implementation and training process. The Hist Gradient Boosting Classifier generally offers better performance and scalability than the Gradient Boosting Classifier but is limited to using the Light GBM package.

Concerning Histogram Gradient Boosting, with different combinations through grid search, we obtained a result with the following choices according to the final model that we chose and submitted on Kaggle. First, the learning_rate is 0.5 a little higher than the default of 0.1, in this way the model shrinks less than the default and the loss parameter is the default “log_loss”. Moreover, max_depth brings some restriction when equal to 3, which makes the model perform from the root to the last leaf (only 3 edges). Lastly, in min_samples_leaf, the model has 5, much less than the default 20, because the model was fed with a small number of rows, so in this way, we avoid trivial trees built by the model.

ExtraTreesClassifier

The ExtraTreesClassifier is a classification algorithm based on the concept of Ensemble Learning. This means that the algorithm creates a variety of classifiers (in this case trees) and combines them to improve the accuracy of the prediction.

Unlike a single decision tree that follows rules based on the value of a particular feature when classifying data points, the ExtraTreesClassifier generates many different decision trees by using randomly selected features. This leads to greater diversity among the trees generated, which in turn can help the algorithm make more Robust predictions.

However, there are drawbacks as well: Since the algorithm uses randomly generated decision trees, it can be difficult to understand the reasoning behind a particular prediction. Also, the training process can be relatively slow, especially for large datasets.

LDA

LinearDiscriminantAnalysis (LDA) is a classification algorithm used to divide data points into multiple classes. Unlike other classification algorithms, such as the k-Nearest Neighbors algorithm, which is based on the distance between data points, LDA attempts to distinguish classes by maximizing the separation between them.

LDA is based on the assumption that the data points within a class have a normal distribution and that the classes are independent of each other. Based on these assumptions, LDA can separate the classes by determining hyperplanes that maximize the separation between the data points.
One advantage of LDA is that it is able to recognize and model even complex class structures. However, it is important to note that LDA only works well if the assumptions about the distribution of data points and the independence of classes actually hold. If this is not the case, LDA can lead to inaccurate predictions.

GaussianProcessClassifier

The GaussianProcessClassifier is a classification algorithm based on the concept of Gaussian Process Regression (GPR). GPR is a technique used to estimate features from observations and make predictions for new data points.

Unlike other classification algorithms, such as the Support Vector Machine algorithm, which is based on the concept of linear separation of classes, GPR attempts to model and account for uncertainty in data points. This allows GPR to make predictions with some uncertainty, rather than just providing a single prediction.

One advantage of GPR is that it is able to model complex, nonlinear relationships in the data. However, it is important to note that GPR is typically slower than other classification algorithms, especially for large datasets. Also, GPR is prone to overfitting if the data set is too small.

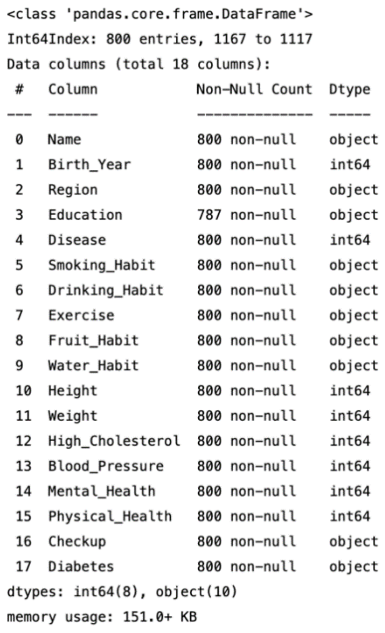
RidgeClassifier first turns the target variable to {-1,1}, thus performing it as a regression task. Also, the advantage of using RidgeClassifier() over LogisticRegression() is that it can perform faster since it only runs the projection matrix just one time.

SGDClassifier is simple to execute for linear classifiers to fit when loss functions are convex (eg, Logistic Regression) and has several ways to optimize the code. However, this classifier requires a variety of hyperparameters, including the number of iterations. Also, it varies according to feature scaling.

PassiveAggressiveClassifier, as the name indicates, this classifier is passive when it is a true classification and aggressive to any kind of blunder, where we give groups of instances over time to train it progressively.

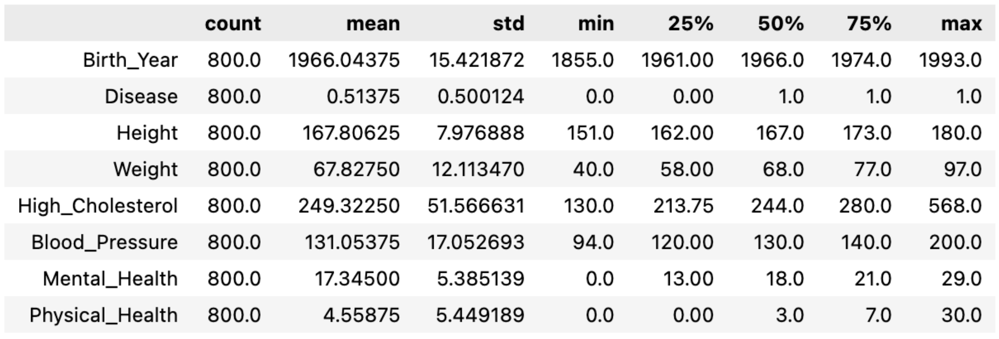
QuadraticDiscriminantAnalysis, produced by using the Bayes' rule to fit classes according to their densities to the dataset. The advantages of using this type of classifier are the fact that they do not demand any hyperparameters tuning, easy to compute solutions and they are multiclass.

Figures



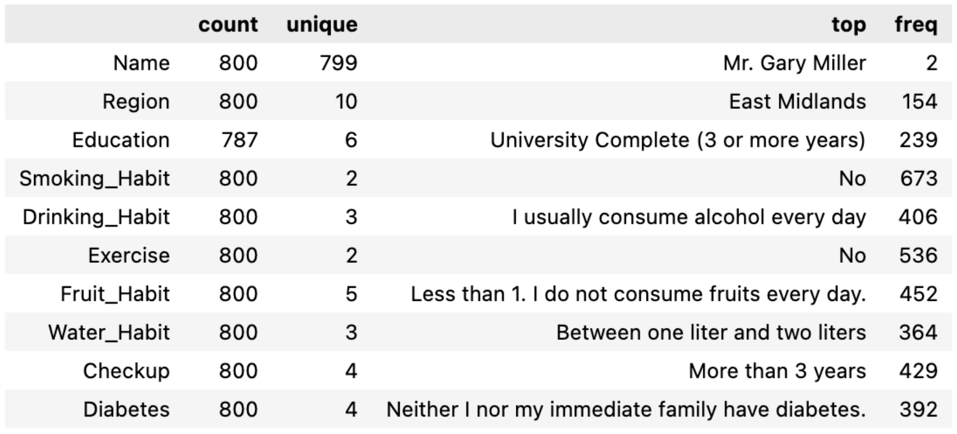
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 800 entries, 1167 to 1117
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Name             800 non-null    object
 1   Birth_Year       800 non-null    int64
 2   Region           800 non-null    object
 3   Education        787 non-null    object
 4   Disease          800 non-null    int64
 5   Smoking_Habit    800 non-null    object
 6   Drinking_Habit   800 non-null    object
 7   Exercise         800 non-null    object
 8   Fruit_Habit      800 non-null    object
 9   Water_Habit      800 non-null    object
 10  Height           800 non-null    int64
 11  Weight           800 non-null    int64
 12  High_Cholesterol 800 non-null    int64
 13  Blood_Pressure   800 non-null    int64
 14  Mental_Health    800 non-null    int64
 15  Physical_Health  800 non-null    int64
 16  Checkup          800 non-null    object
 17  Diabetes         800 non-null    object
dtypes: int64(8), object(10)
memory usage: 151.0+ KB
```

Figure 1 - *Information of the variables*



	count	mean	std	min	25%	50%	75%	max
Birth_Year	800.0	1966.04375	15.421872	1855.0	1961.00	1966.0	1974.0	1993.0
Disease	800.0	0.51375	0.500124	0.0	0.00	1.0	1.0	1.0
Height	800.0	167.80625	7.976888	151.0	162.00	167.0	173.0	180.0
Weight	800.0	67.82750	12.113470	40.0	58.00	68.0	77.0	97.0
High_Cholesterol	800.0	249.32250	51.566631	130.0	213.75	244.0	280.0	568.0
Blood_Pressure	800.0	131.05375	17.052693	94.0	120.00	130.0	140.0	200.0
Mental_Health	800.0	17.34500	5.385139	0.0	13.00	18.0	21.0	29.0
Physical_Health	800.0	4.55875	5.449189	0.0	0.00	3.0	7.0	30.0

Figure 2 - *Descriptive Statistics of the numerical variables*



	count	unique	top	freq
Name	800	799	Mr. Gary Miller	2
Region	800	10	East Midlands	154
Education	787	6	University Complete (3 or more years)	239
Smoking_Habit	800	2	No	673
Drinking_Habit	800	3	I usually consume alcohol every day	406
Exercise	800	2	No	536
Fruit_Habit	800	5	Less than 1. I do not consume fruits every day.	452
Water_Habit	800	3	Between one liter and two liters	364
Checkup	800	4	More than 3 years	429
Diabetes	800	4	Neither I nor my immediate family have diabetes.	392

Figure 3 - *Descriptive Statistics of the categorical variables*

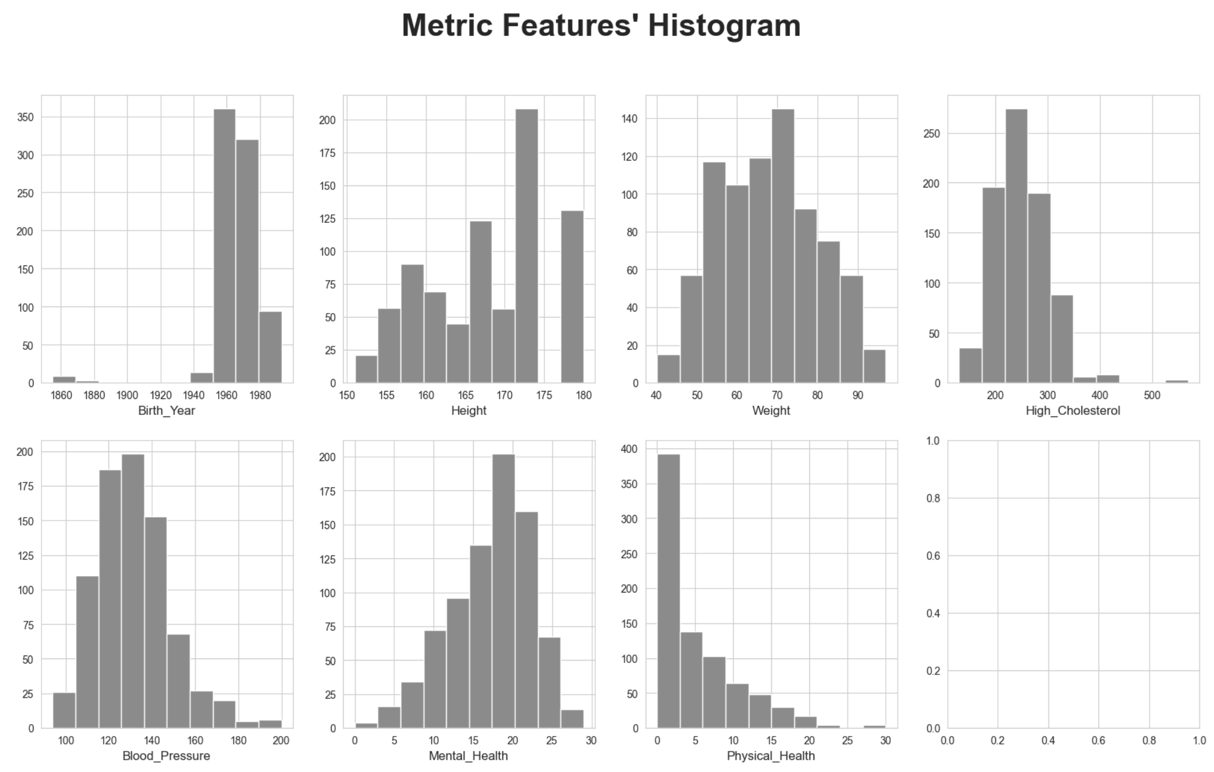


Figure 4 - *Metric Features' Histograms*

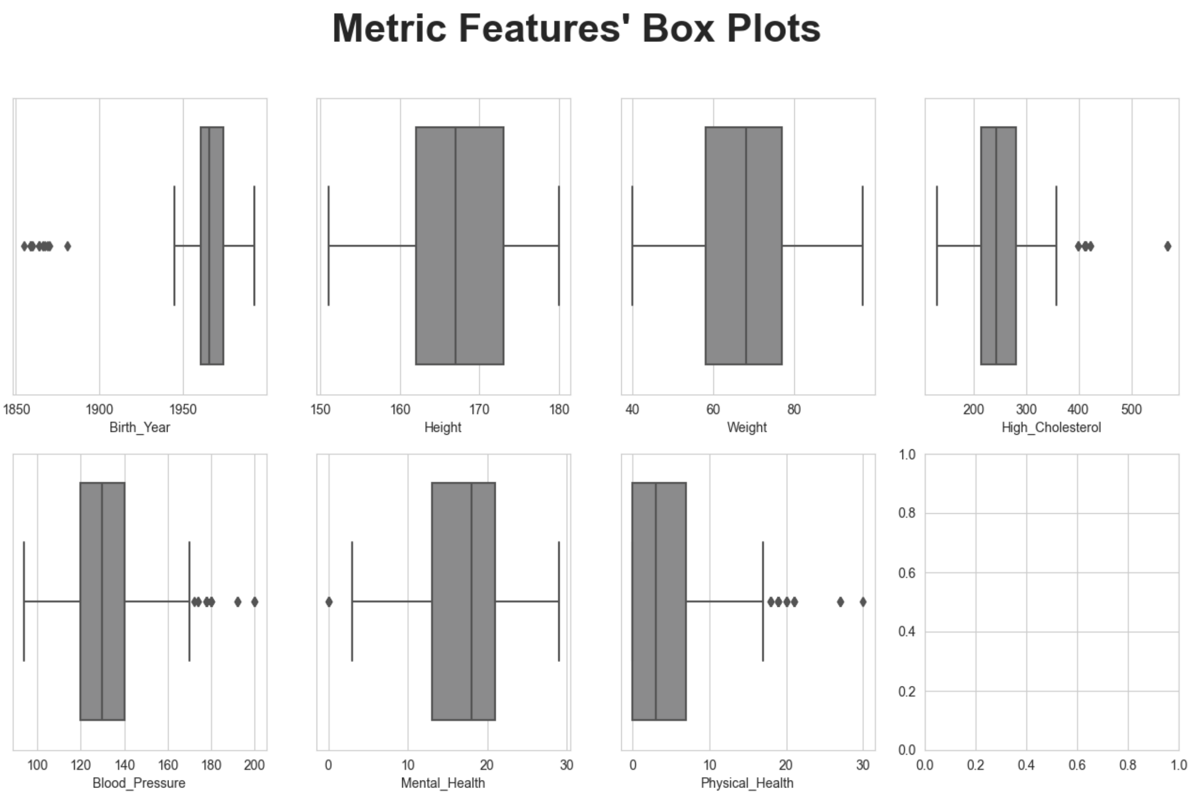


Figure 5 *- Metric Features' Box Plots*

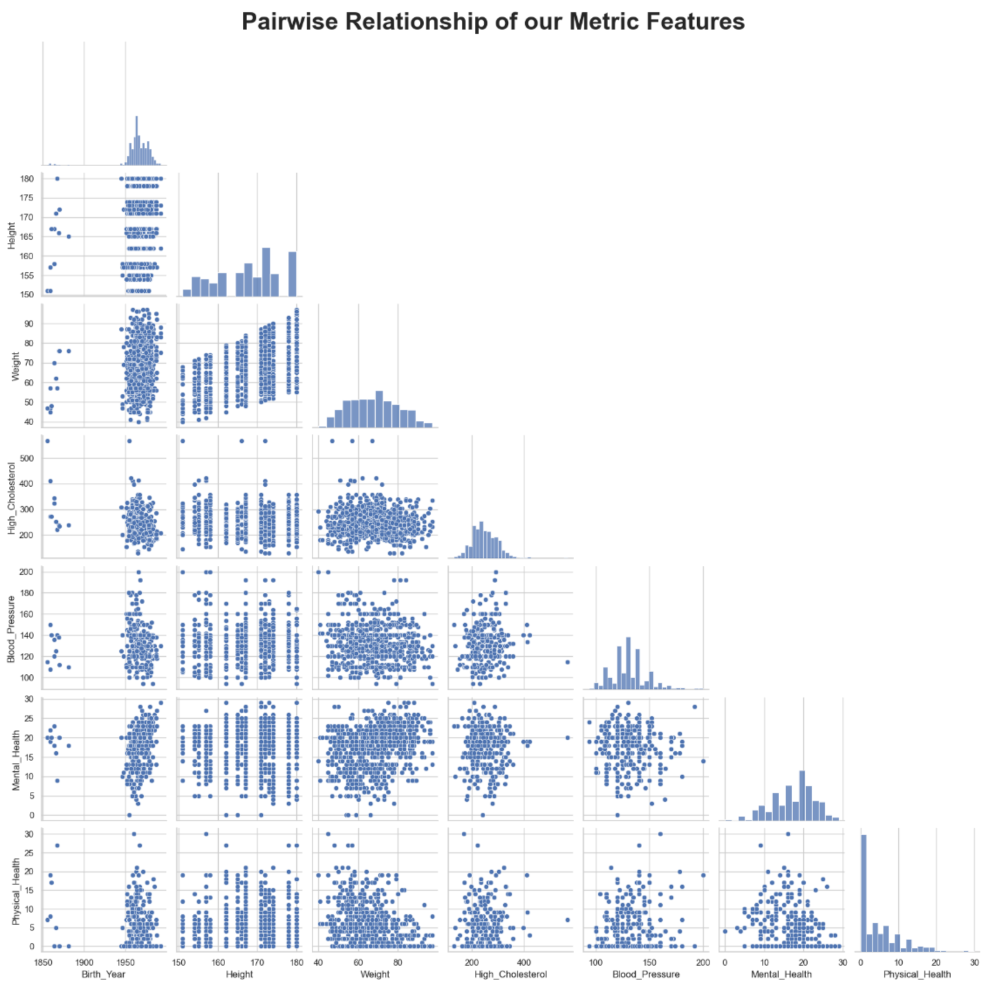


Figure 6 - *Pairwise Relationship of the Metric Features*

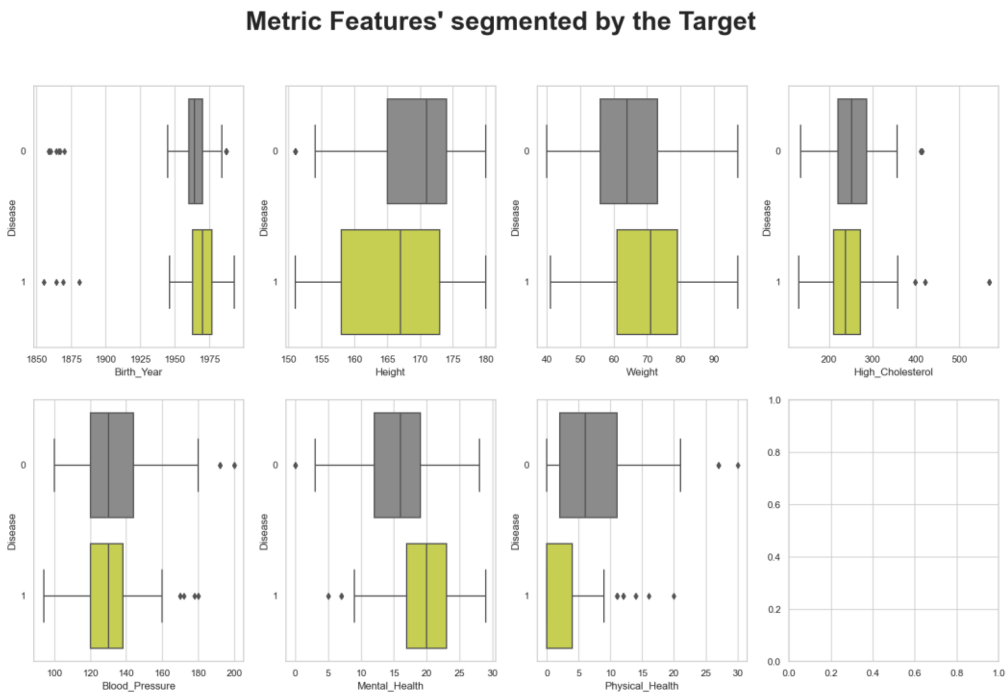


Figure 7 - *Metric Features' segmented by Disease*

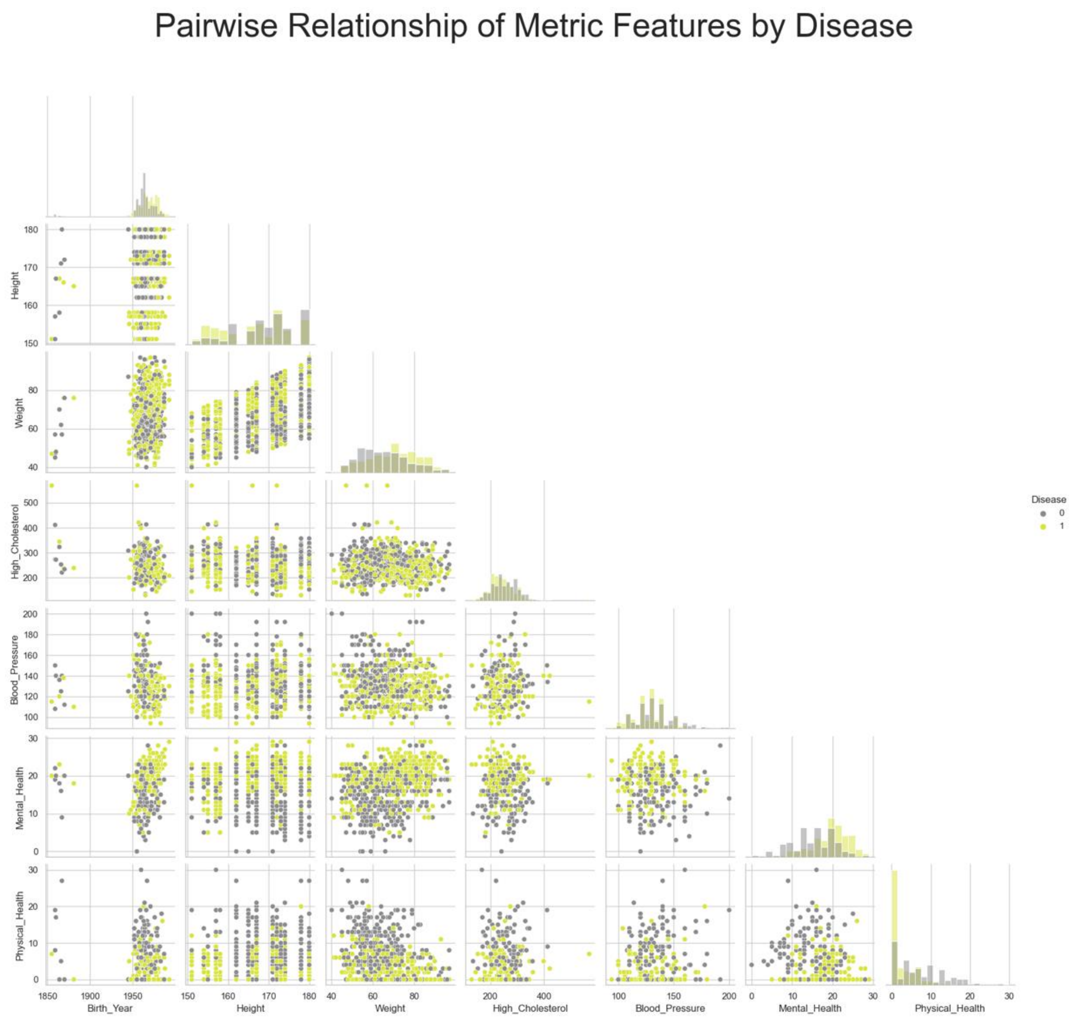


Figure 8 - *Pairwise Relationship of metric features by Disease*

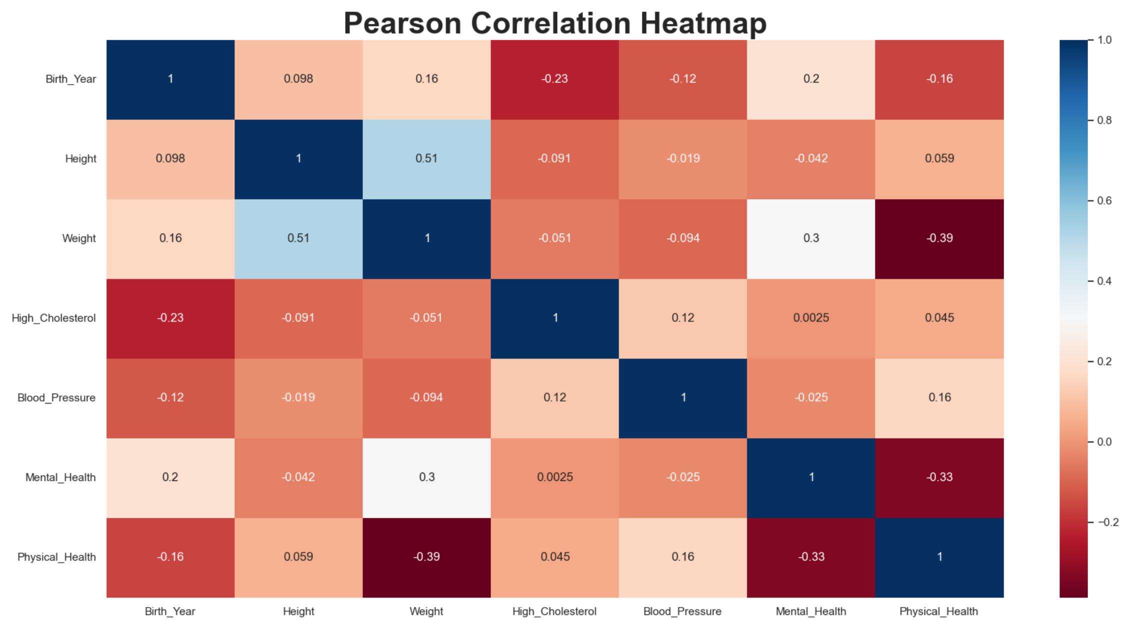


Figure 9 - *Pearson Correlation Heatmap*

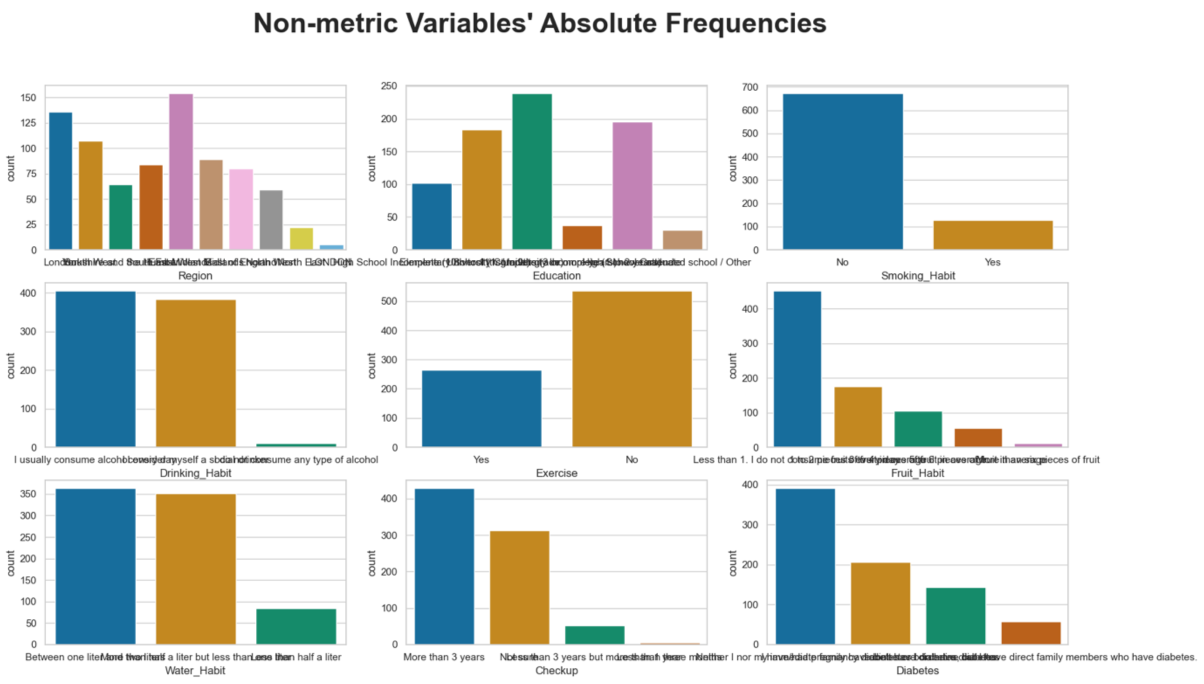


Figure 10 - *Non-metric Variable's Absolute Frequencies bar plots*

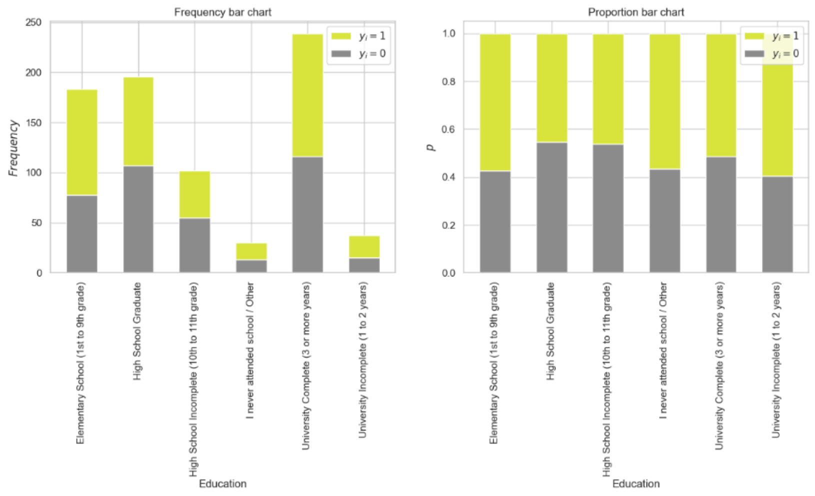


Figure 11 - *Barplots of non-metric variables segmented by the target // Education*

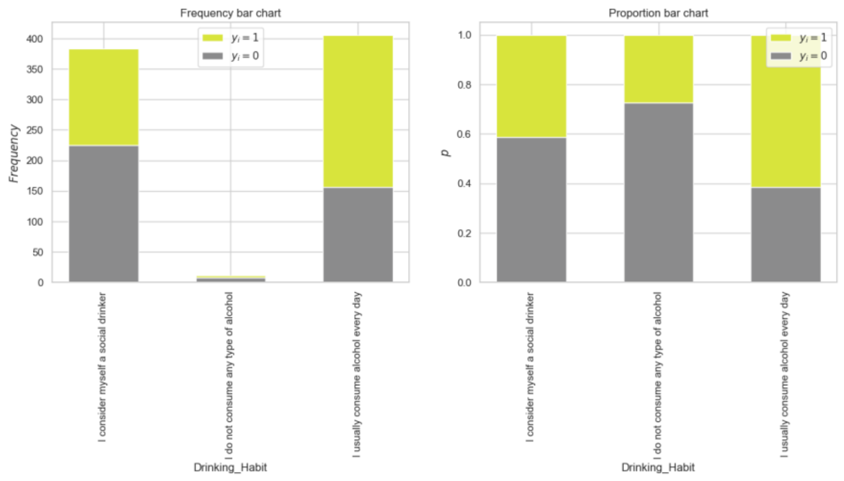


Figure 12 *- Barplots of non-metric variables segmented by the target // Drinking habit*

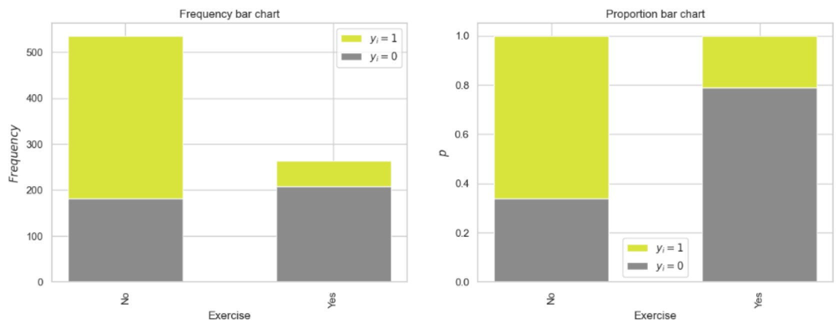


Figure 13 - *Barplots of non-metric variables segmented by the target // Exercise*

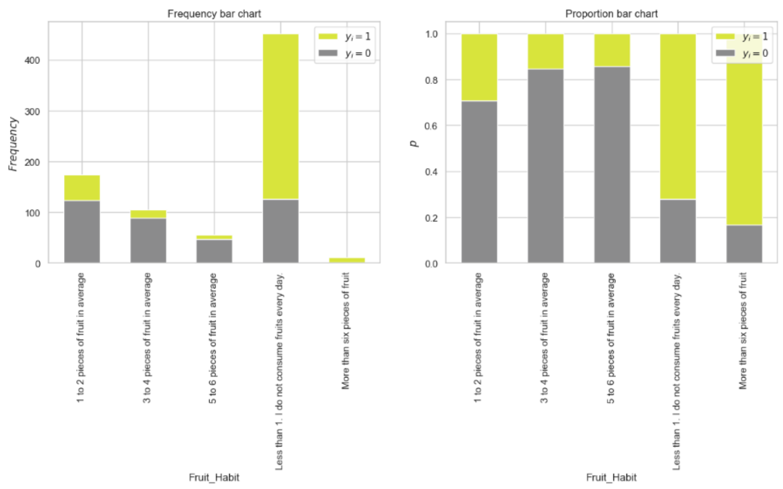


Figure 14 - *Barplots of non-metric variables segmented by the target // Fruit Habit*

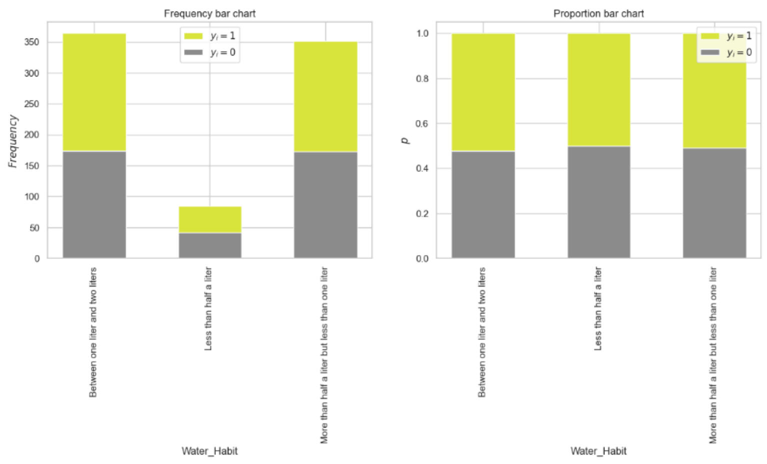


Figure 15- *Barplots of non-metric variables segmented by the target // Water Habit*

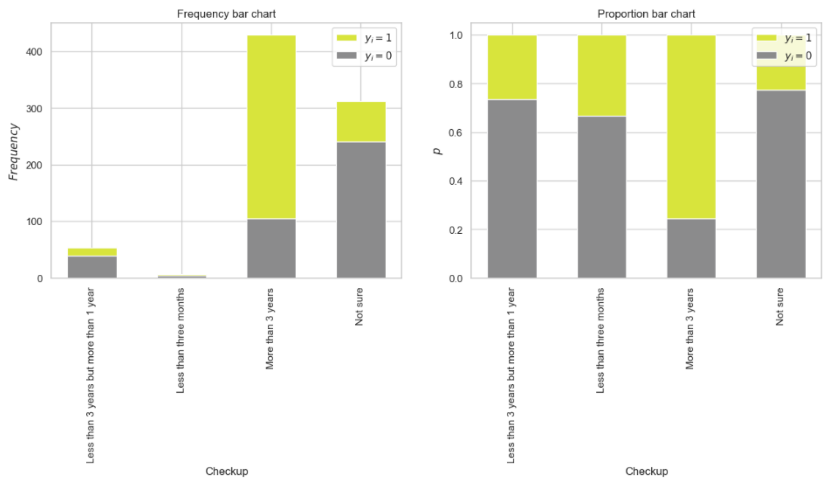


***Figure 16 -** Barplots of non-metric variables segmented by the target // Checkup*

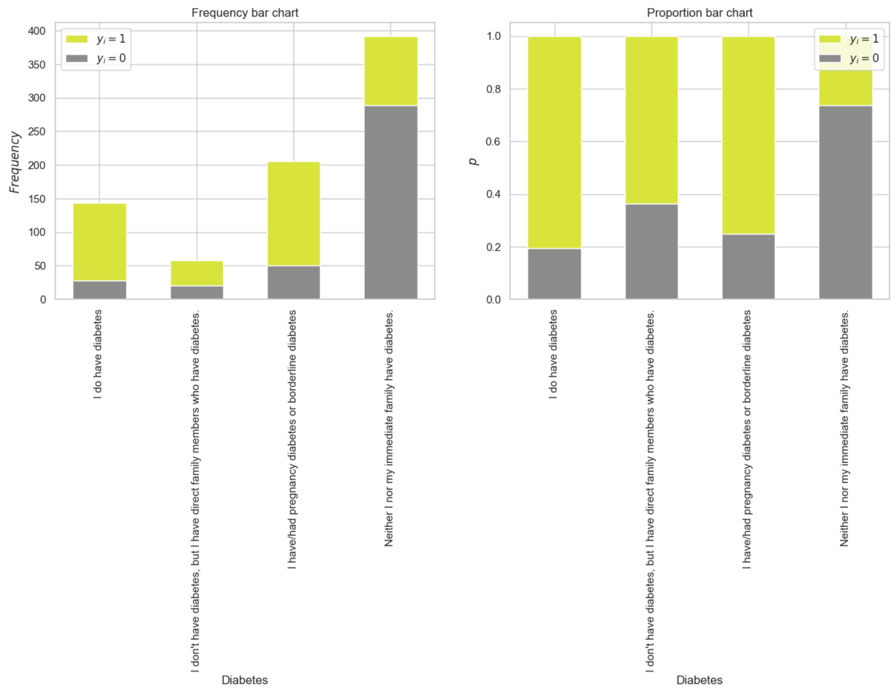


***Figure 17 -** Barplots of non-metric variables segmented by the target // Diabetes*

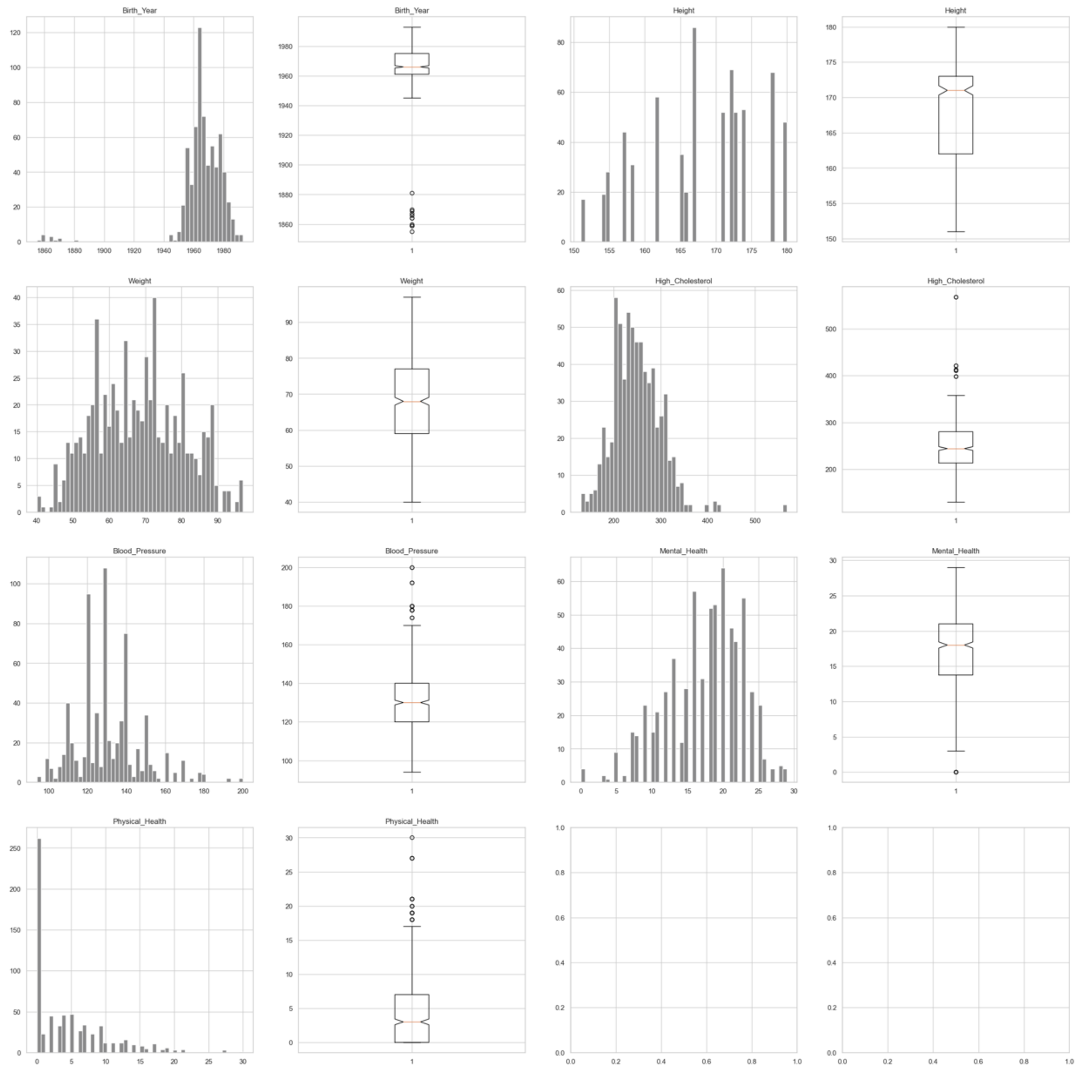


***Figure 18 -** Outliers and Box Plots for Metric Variables*

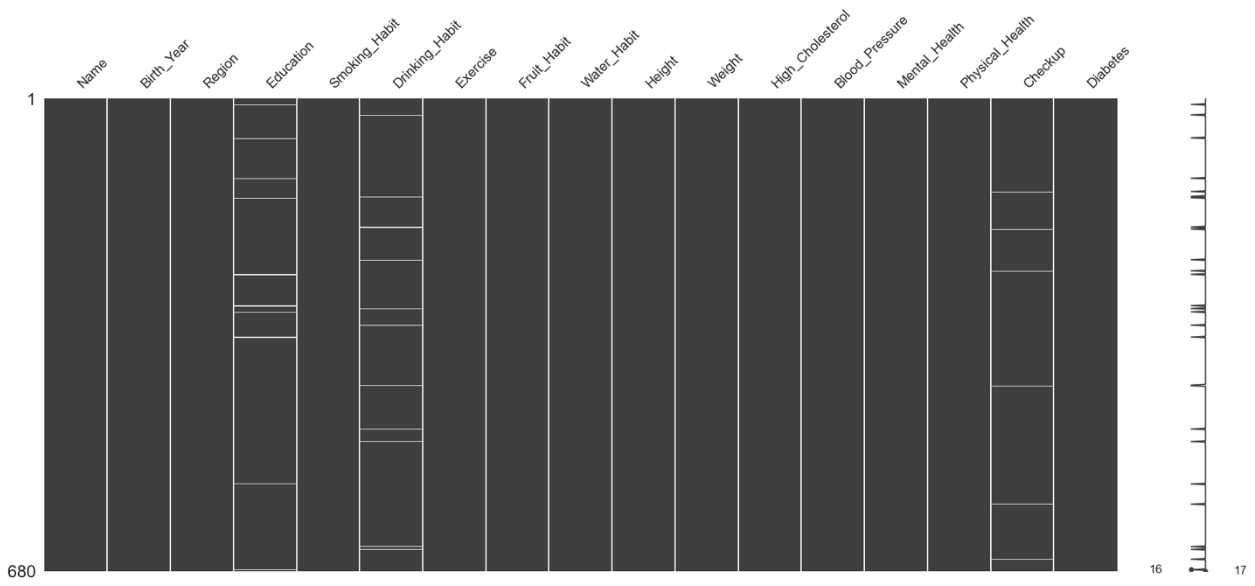


***Figure 19** - MSNO matrix for Missing Values*

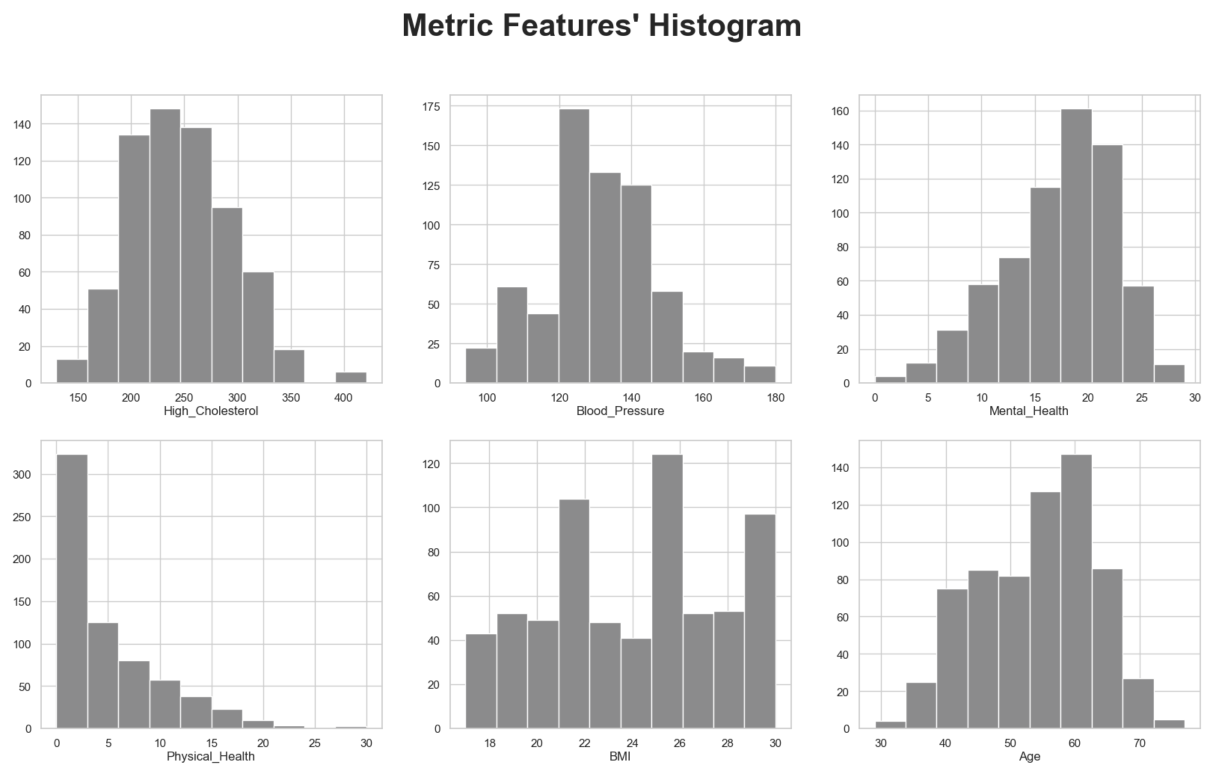


***Figure 20 -** Cleaned histograms of metric features*

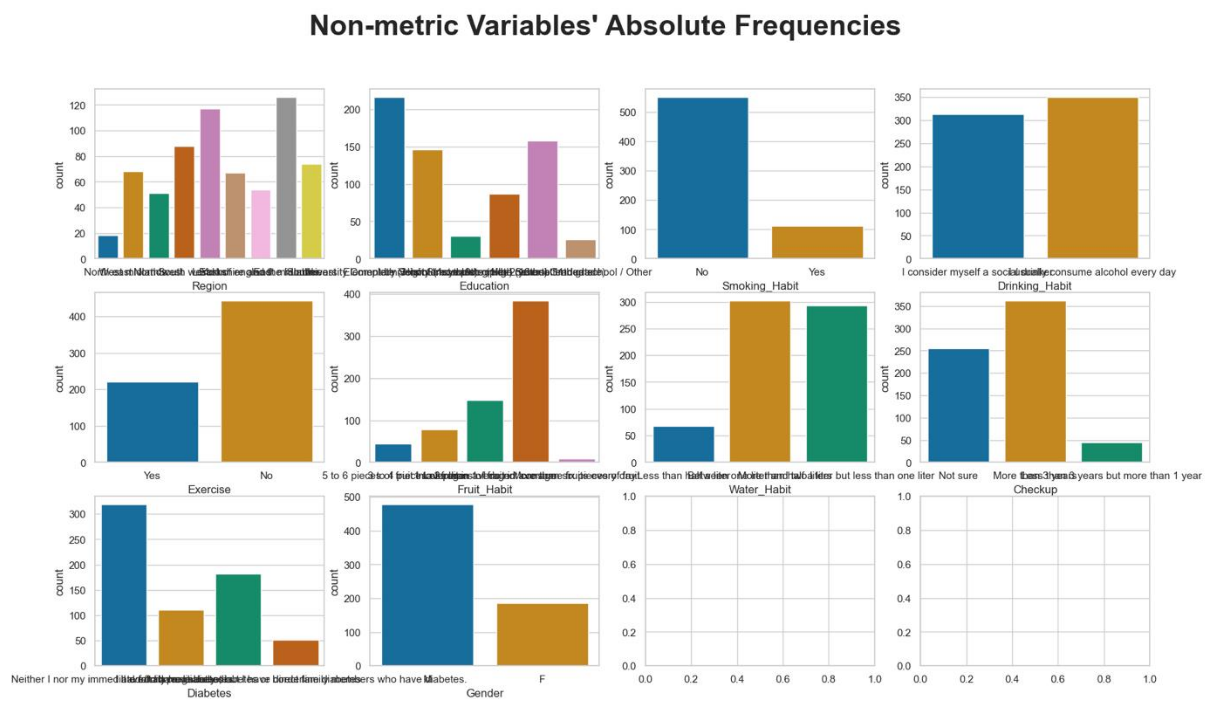


***Figure 21 -** Non-metric variables' Absolute Frequencies cleaned*

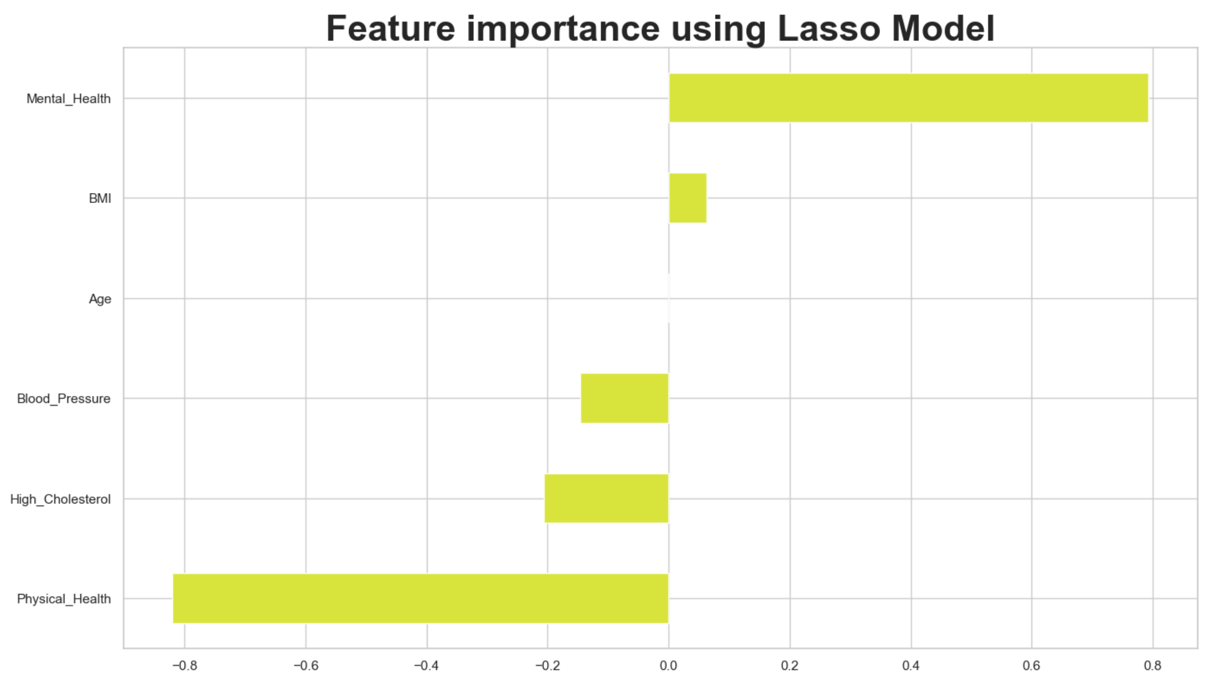


Figure 22 **-** *Feature importance using Lasso Model*

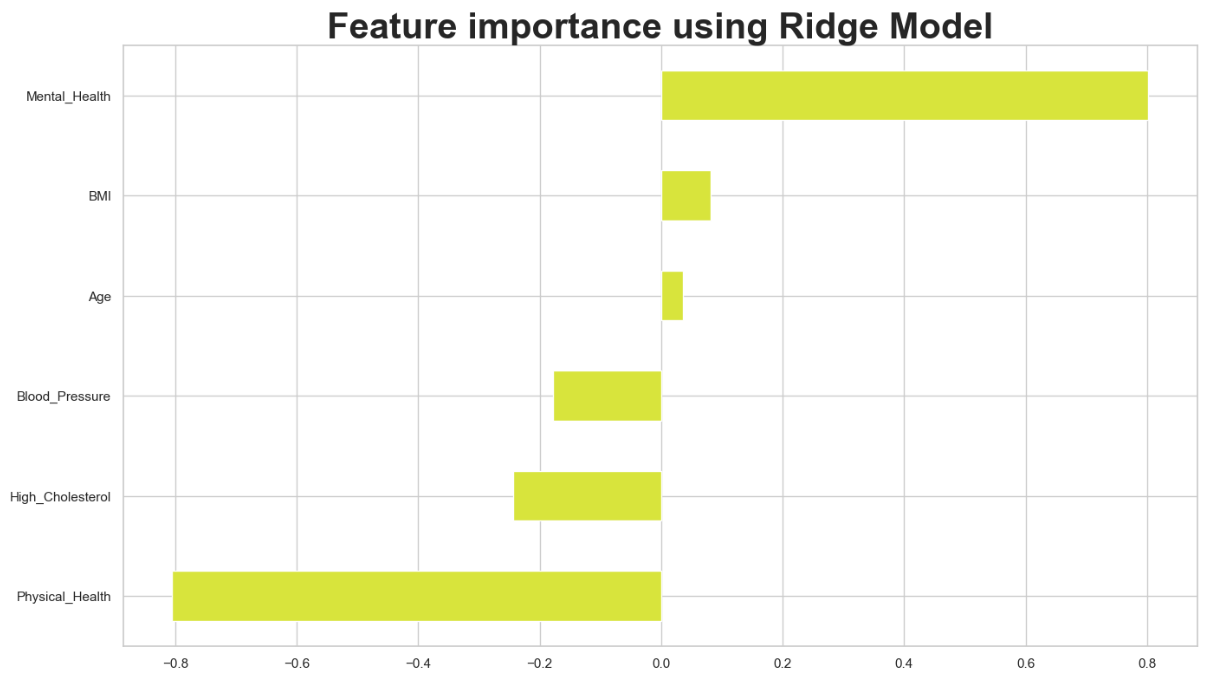


Figure 23 **-** *Feature importance using Ridge Model*

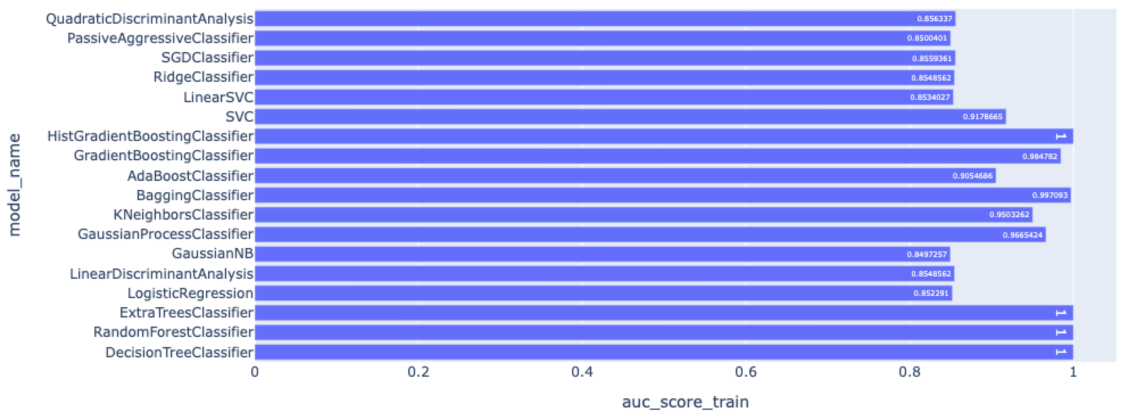


Figure 24 - *AUC Score*

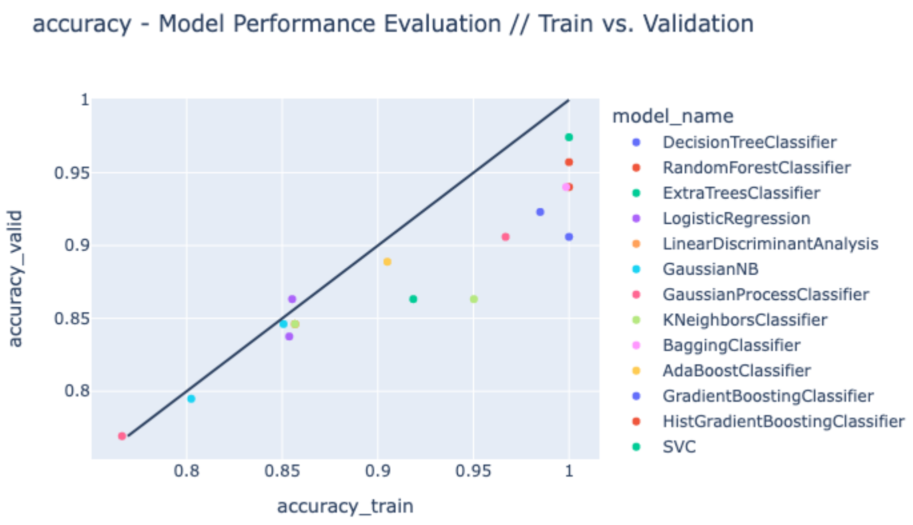


Figure 25- *Accuracy - Model Performance Evaluation // Train vs. Validation*

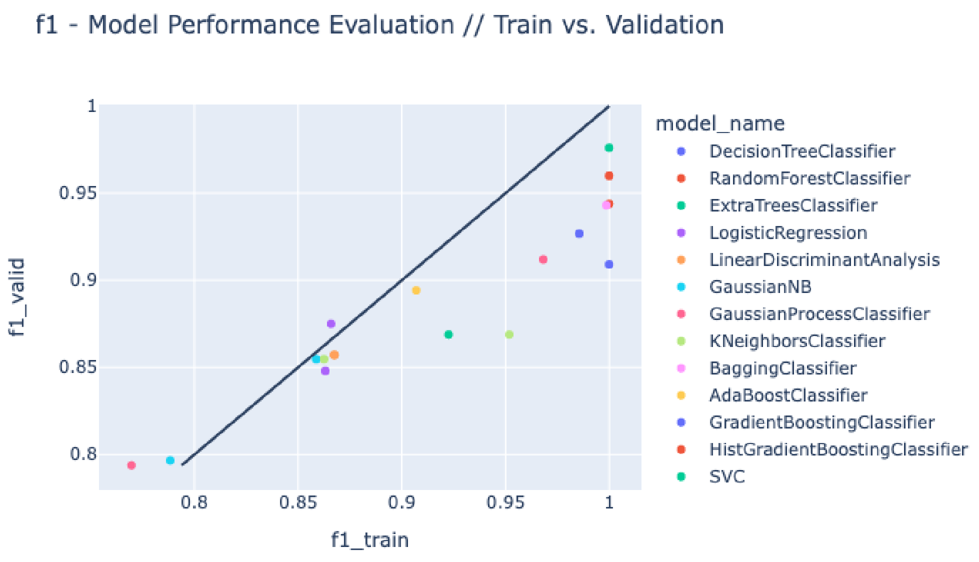
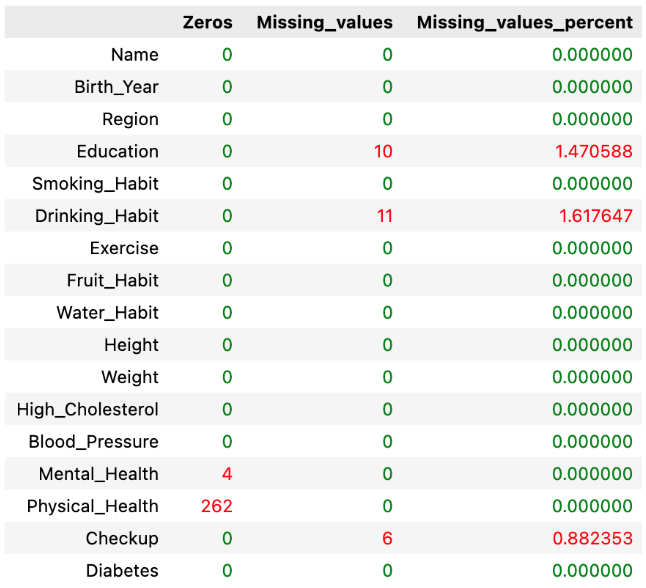


Figure 26* - *Model Performance Evaluation // Train vs. Validation

Tables



	Zeros	Missing_values	Missing_values_percent
Name	0	0	0.000000
Birth_Year	0	0	0.000000
Region	0	0	0.000000
Education	0	10	1.470588
Smoking_Habit	0	0	0.000000
Drinking_Habit	0	11	1.617647
Exercise	0	0	0.000000
Fruit_Habit	0	0	0.000000
Water_Habit	0	0	0.000000
Height	0	0	0.000000
Weight	0	0	0.000000
High_Cholesterol	0	0	0.000000
Blood_Pressure	0	0	0.000000
Mental_Health	4	0	0.000000
Physical_Health	262	0	0.000000
Checkup	0	6	0.882353
Diabetes	0	0	0.000000

***Table 1** - Missing values and the zeros of the training dataset*

Feature	Description	Method to obtain
Age	Age of the patient	Subtracted the feature "Birth_Year" to the current year
Gender	Gender of the patient	Considered "Mr" prefix for a male (M) person and the rest for a female person (F)
BMI	Body Mass Index of the patient	Performed the formula $BMI = \frac{weight\ (kg)}{height\ (m)^2}$ once having "Weight" and "Height" variables

***Table 2 -** Table with the information regarding the new features*

NUMERICAL DATA						
Predictor	**Spearman**	**Pearson**	**RFE**	**Lasso**	**ANOVA**	**What to do?**
High_Cholesterol	Keep	Discard	Keep	Keep	Discard	**Try with and without**
Blood_Pressure	Keep	Discard	Discard	Keep	Discard	**Discard**
Mental_Health	Keep	Keep	Keep	Keep	Keep	**Include**
Physical_Health	Keep	Keep	Keep	Keep	Keep	**Include**
BMI	Keep	Discard	Discard	Discard?	Keep	**Try with and without**
Age	Keep	Discard	Discard	Discard	Keep	**Discard**

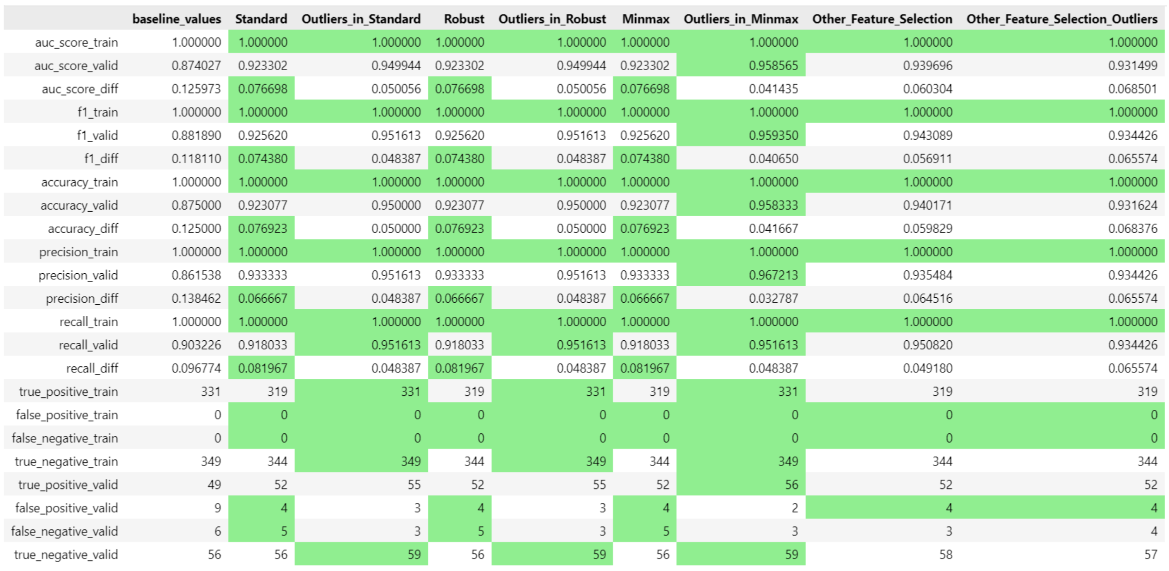
***Table 3 -** Final insights of the feature selection of the numerical data*

CATEGORICAL FEATURES	
Predictor	Chi-Square test – What to do?
Gender	Include in the model
Region	Discard
Education	Discard
Checkup	Include in the model
Diabetes	Include in the model
Smoking_Habit	Discard
Exercise	Include in the model
Water_Habit	Discard
Drinking_Habit	Include in the model

Table 4 *- Final insights on the feature selection of the categorical data*

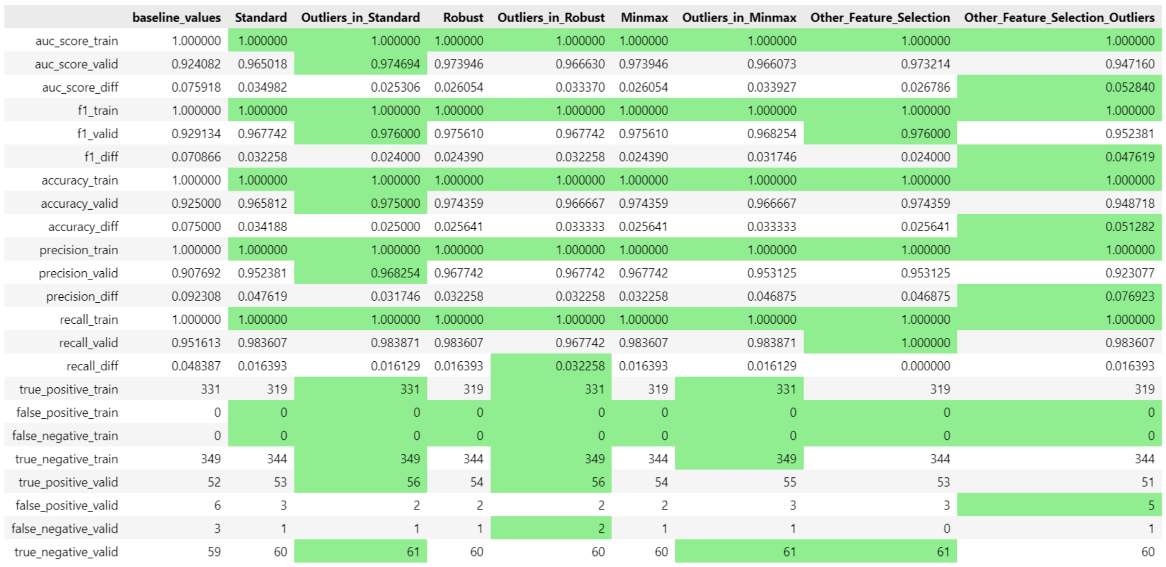
Dataset	Outliers	Encoder	Scaler	Feature Selection Method
X_train_MF_OH_SS	No - manually	One-Hot	Standard	Manually
X_train_OH_SS	Yes			
X_train_MF_OH_MM	No - manually		MinMax	
X_train_OH_MM	Yes			
X_train_MF_OH_RS	No - manually		Robust	
X_train_OH_RS	Yes			
X_train_OH_MM_FS	No - manually		MinMax	Pearson Correlation and Chi-Squared tests

Table 5 - *Labeling of the datasets*



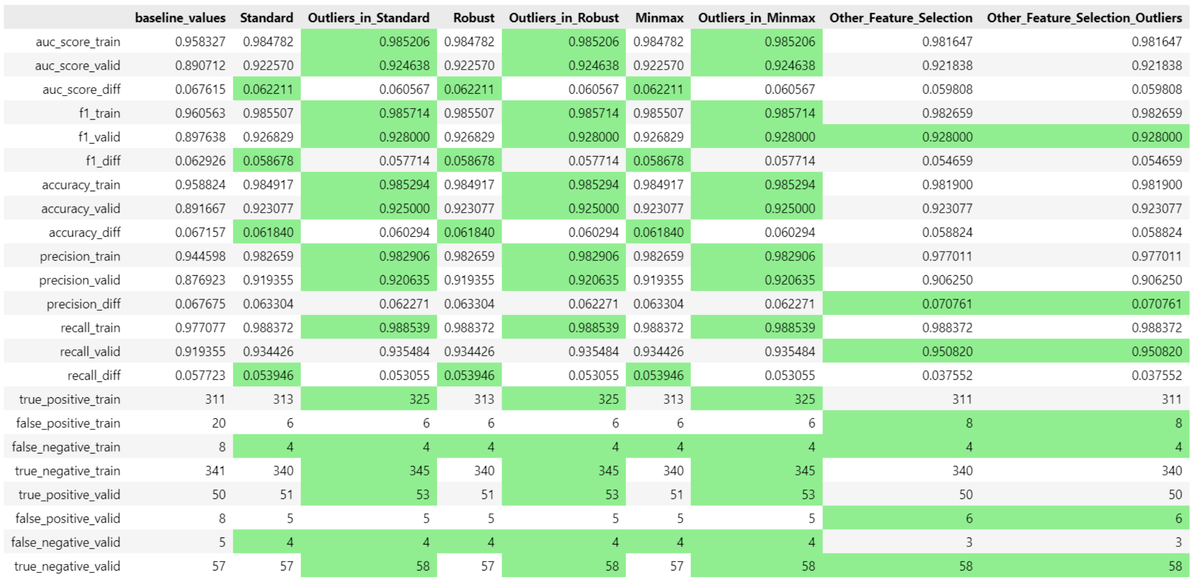
	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
auc_score_valid	0.874027	0.923302	0.949944	0.923302	0.949944	0.923302	0.958565	0.939696	0.931499
auc_score_diff	0.125973	0.076698	0.050056	0.076698	0.050056	0.076698	0.041435	0.060304	0.068501
f1_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
f1_valid	0.881890	0.925620	0.951613	0.925620	0.951613	0.925620	0.959350	0.943089	0.934426
f1_diff	0.118110	0.074380	0.048387	0.074380	0.048387	0.074380	0.040650	0.056911	0.065574
accuracy_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
accuracy_valid	0.875000	0.923077	0.950000	0.923077	0.950000	0.923077	0.958333	0.940171	0.931624
accuracy_diff	0.125000	0.076923	0.050000	0.076923	0.050000	0.076923	0.041667	0.059829	0.068376
precision_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
precision_valid	0.861538	0.933333	0.951613	0.933333	0.951613	0.933333	0.967213	0.935484	0.934426
precision_diff	0.138462	0.066667	0.048387	0.066667	0.048387	0.066667	0.032787	0.064516	0.065574
recall_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
recall_valid	0.903226	0.918033	0.951613	0.918033	0.951613	0.918033	0.951613	0.950820	0.934426
recall_diff	0.096774	0.081967	0.048387	0.081967	0.048387	0.081967	0.048387	0.049180	0.065574
true_positive_train	331	319	331	319	331	319	331	319	319
false_positive_train	0	0	0	0	0	0	0	0	0
false_negative_train	0	0	0	0	0	0	0	0	0
true_negative_train	349	344	349	344	349	344	349	344	344
true_positive_valid	49	52	55	52	55	52	56	52	52
false_positive_valid	9	4	3	4	3	4	2	4	4
false_negative_valid	6	5	3	5	3	5	3	3	4
true_negative_valid	56	56	59	56	59	56	59	58	57

Table 6 - *Decision Tree Classifier Comparison*



	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
auc_score_valid	0.924082	0.965018	0.974694	0.973946	0.966630	0.973946	0.966073	0.973214	0.947160
auc_score_diff	0.075918	0.034982	0.025306	0.026054	0.033370	0.026054	0.033927	0.026786	0.052840
f1_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
f1_valid	0.929134	0.967742	0.976000	0.975610	0.967742	0.975610	0.968254	0.976000	0.952381
f1_diff	0.070866	0.032258	0.024000	0.024390	0.032258	0.024390	0.031746	0.024000	0.047619
accuracy_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
accuracy_valid	0.925000	0.965812	0.975000	0.974359	0.966667	0.974359	0.966667	0.974359	0.948718
accuracy_diff	0.075000	0.034188	0.025000	0.025641	0.033333	0.025641	0.033333	0.025641	0.051282
precision_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
precision_valid	0.907692	0.952381	0.968254	0.967742	0.967742	0.967742	0.953125	0.953125	0.923077
precision_diff	0.092308	0.047619	0.031746	0.032258	0.032258	0.032258	0.046875	0.046875	0.076923
recall_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
recall_valid	0.951613	0.983607	0.983871	0.983607	0.967742	0.983607	0.983871	1.000000	0.983607
recall_diff	0.048387	0.016393	0.016129	0.016393	0.032258	0.016393	0.016129	0.000000	0.016393
true_positive_train	331	319	331	319	331	319	331	319	319
false_positive_train	0	0	0	0	0	0	0	0	0
false_negative_train	0	0	0	0	0	0	0	0	0
true_negative_train	349	344	349	344	349	344	349	344	344
true_positive_valid	52	53	56	54	56	54	55	53	51
false_positive_valid	6	3	2	2	2	2	3	3	5
false_negative_valid	3	1	1	1	2	1	1	0	1
true_negative_valid	59	60	61	60	60	60	61	61	60

Table 7 - *Random Forest Classifier Comparison*



	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	0.958327	0.984782	0.985206	0.984782	0.985206	0.984782	0.985206	0.981647	0.981647
auc_score_valid	0.890712	0.922570	0.924638	0.922570	0.924638	0.922570	0.924638	0.921838	0.921838
auc_score_diff	0.067615	0.062211	0.060567	0.062211	0.060567	0.062211	0.060567	0.059808	0.059808
f1_train	0.960563	0.985507	0.985714	0.985507	0.985714	0.985507	0.985714	0.982659	0.982659
f1_valid	0.897638	0.926829	0.928000	0.926829	0.928000	0.926829	0.928000	0.928000	0.928000
f1_diff	0.062926	0.058678	0.057714	0.058678	0.057714	0.058678	0.057714	0.054659	0.054659
accuracy_train	0.958824	0.984917	0.985294	0.984917	0.985294	0.984917	0.985294	0.981900	0.981900
accuracy_valid	0.891667	0.923077	0.925000	0.923077	0.925000	0.923077	0.925000	0.923077	0.923077
accuracy_diff	0.067157	0.061840	0.060294	0.061840	0.060294	0.061840	0.060294	0.058824	0.058824
precision_train	0.944598	0.982659	0.982906	0.982659	0.982906	0.982659	0.982906	0.977011	0.977011
precision_valid	0.876923	0.919355	0.920635	0.919355	0.920635	0.919355	0.920635	0.906250	0.906250
precision_diff	0.067675	0.063304	0.062271	0.063304	0.062271	0.063304	0.062271	0.070761	0.070761
recall_train	0.977077	0.988372	0.988539	0.988372	0.988539	0.988372	0.988539	0.988372	0.988372
recall_valid	0.919355	0.934426	0.935484	0.934426	0.935484	0.934426	0.935484	0.950820	0.950820
recall_diff	0.057723	0.053946	0.053055	0.053946	0.053055	0.053946	0.053055	0.037552	0.037552
true_positive_train	311	313	325	313	325	313	325	311	311
false_positive_train	20	6	6	6	6	6	6	8	8
false_negative_train	8	4	4	4	4	4	4	4	4
true_negative_train	341	340	345	340	345	340	345	340	340
true_positive_valid	50	51	53	51	53	51	53	50	50
false_positive_valid	8	5	5	5	5	5	5	6	6
false_negative_valid	5	4	4	4	4	4	4	3	3
true_negative_valid	57	57	58	57	58	57	58	58	58

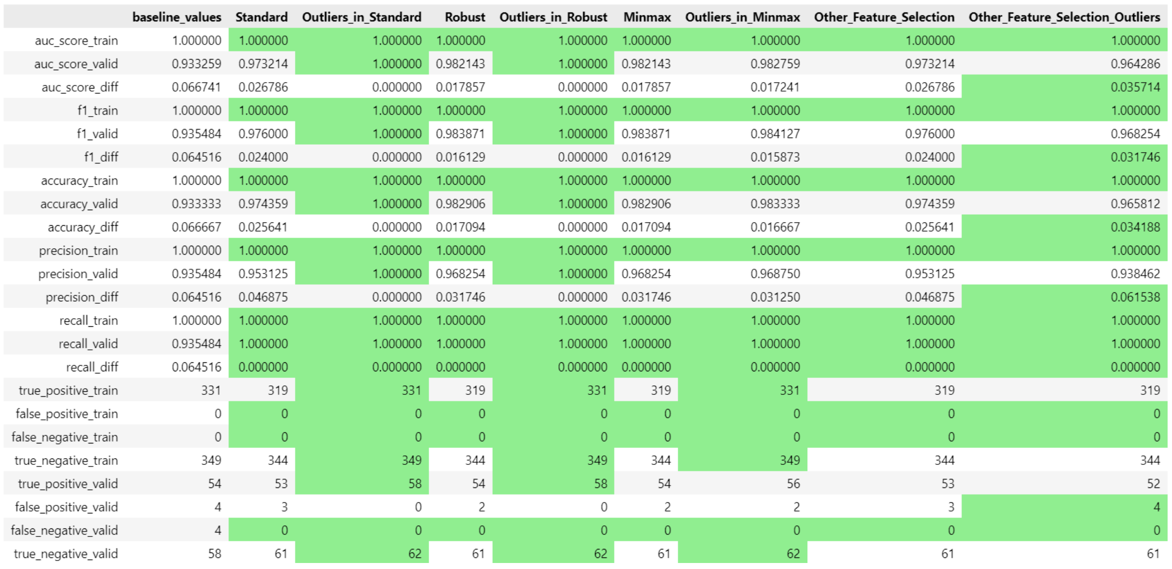
Table 8 - *Gradient Boosting Comparison*

	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
auc_score_valid	0.941324	0.956089	0.974694	0.956089	0.974694	0.956089	0.974694	0.973946	0.973946
auc_score_diff	0.058676	0.043911	0.025306	0.043911	0.025306	0.043911	0.025306	0.026054	0.026054
f1_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
f1_valid	0.944000	0.960000	0.976000	0.960000	0.976000	0.960000	0.976000	0.975610	0.975610
f1_diff	0.056000	0.040000	0.024000	0.040000	0.024000	0.040000	0.024000	0.024390	0.024390
accuracy_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
accuracy_valid	0.941667	0.957265	0.975000	0.957265	0.975000	0.957265	0.975000	0.974359	0.974359
accuracy_diff	0.058333	0.042735	0.025000	0.042735	0.025000	0.042735	0.025000	0.025641	0.025641
precision_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
precision_valid	0.936508	0.937500	0.968254	0.937500	0.968254	0.937500	0.968254	0.967742	0.967742
precision_diff	0.063492	0.062500	0.031746	0.062500	0.031746	0.062500	0.031746	0.032258	0.032258
recall_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
recall_valid	0.951613	0.983607	0.983871	0.983607	0.983871	0.983607	0.983871	0.983607	0.983607
recall_diff	0.048387	0.016393	0.016129	0.016393	0.016129	0.016393	0.016129	0.016393	0.016393
true_positive_train	331	319	331	319	331	319	331	319	319
false_positive_train	0	0	0	0	0	0	0	0	0
false_negative_train	0	0	0	0	0	0	0	0	0
true_negative_train	349	344	349	344	349	344	349	344	344
true_positive_valid	54	52	56	52	56	52	56	54	54
false_positive_valid	4	4	2	4	2	4	2	2	2
false_negative_valid	3	1	1	1	1	1	1	1	1
true_negative_valid	59	60	61	60	61	60	61	60	60

***Table 9 -** HistGradient Boosting Classifier Comparison*

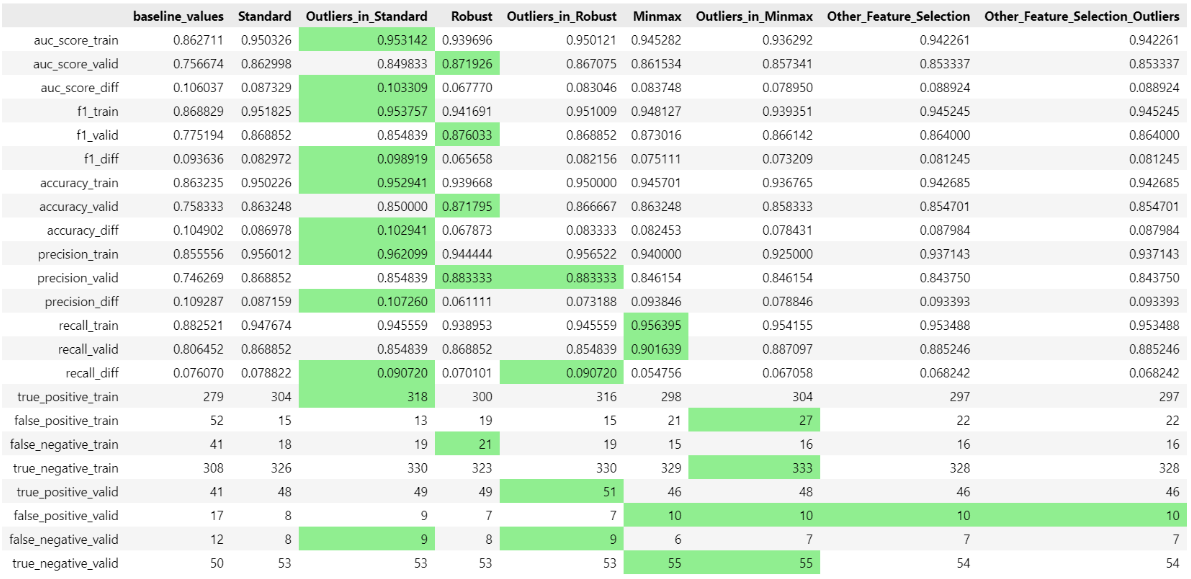
	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	1.000000	0.966542	0.967546	0.935878	0.943767	0.907236	0.914334	0.907236	0.907236
auc_score_valid	0.907953	0.904713	0.916018	0.871194	0.883204	0.844409	0.865406	0.844409	0.844409
auc_score_diff	0.092047	0.061829	0.051529	0.064684	0.060563	0.062828	0.048928	0.062828	0.062828
f1_train	1.000000	0.968208	0.968571	0.940000	0.946176	0.912732	0.917847	0.912732	0.912732
f1_valid	0.912000	0.912000	0.920635	0.878049	0.887097	0.857143	0.875000	0.857143	0.857143
f1_diff	0.088000	0.056208	0.047937	0.061951	0.059079	0.055590	0.042847	0.055590	0.055590
accuracy_train	1.000000	0.966817	0.967647	0.936652	0.944118	0.907994	0.914706	0.907994	0.907994
accuracy_valid	0.908333	0.905983	0.916667	0.871795	0.883333	0.846154	0.866667	0.846154	0.846154
accuracy_diff	0.091667	0.060835	0.050980	0.064857	0.060784	0.061840	0.048039	0.061840	0.061840
precision_train	1.000000	0.962644	0.965812	0.924157	0.935574	0.898592	0.907563	0.898592	0.898592
precision_valid	0.904762	0.890625	0.906250	0.870968	0.887097	0.830769	0.848485	0.830769	0.830769
precision_diff	0.095238	0.072019	0.059562	0.053190	0.048477	0.067822	0.059078	0.067822	0.067822
recall_train	1.000000	0.973837	0.971347	0.956395	0.957020	0.927326	0.928367	0.927326	0.927326
recall_valid	0.919355	0.934426	0.935484	0.885246	0.887097	0.885246	0.903226	0.885246	0.885246
recall_diff	0.080645	0.039411	0.035863	0.071149	0.069923	0.042080	0.025141	0.042080	0.042080
true_positive_train	331	306	319	292	308	283	298	283	283
false_positive_train	0	13	12	27	23	36	33	36	36
false_negative_train	0	9	10	15	15	25	25	25	25
true_negative_train	349	335	339	329	334	319	324	319	319
true_positive_valid	52	49	52	48	51	45	48	45	45
false_positive_valid	6	7	6	8	7	11	10	11	11
false_negative_valid	5	4	4	7	7	7	6	7	7
true_negative_valid	57	57	58	54	55	54	56	54	54

***Table 10 -** Gaussian Process Classifier Comparison*



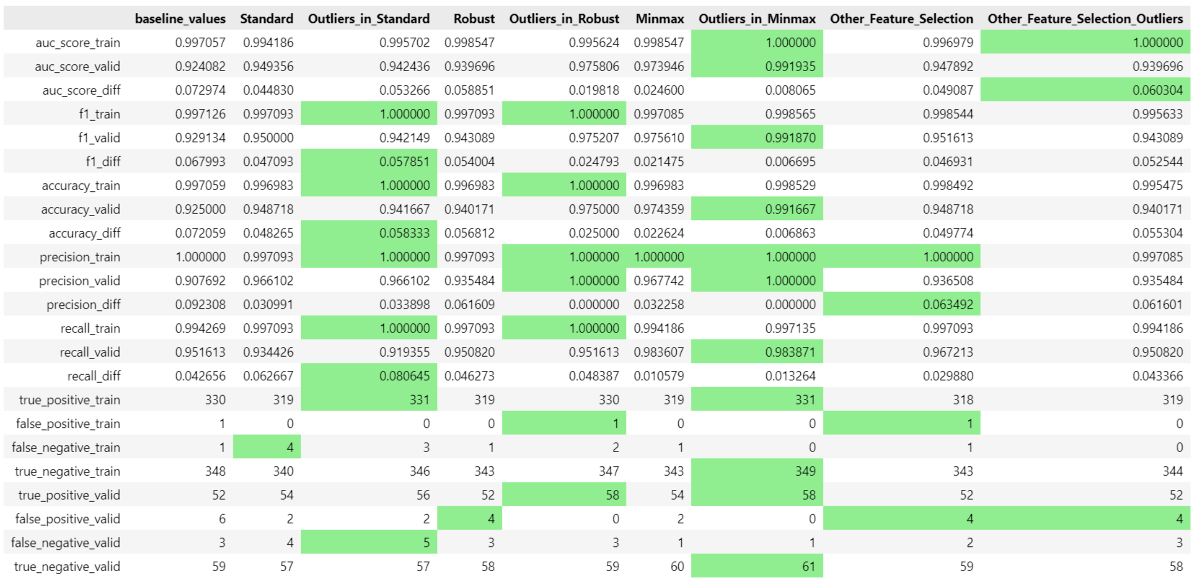
	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
auc_score_valid	0.933259	0.973214	1.000000	0.982143	1.000000	0.982143	0.982759	0.973214	0.964286
auc_score_diff	0.066741	0.026786	0.000000	0.017857	0.000000	0.017857	0.017241	0.026786	0.035714
f1_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
f1_valid	0.935484	0.976000	1.000000	0.983871	1.000000	0.983871	0.984127	0.976000	0.968254
f1_diff	0.064516	0.024000	0.000000	0.016129	0.000000	0.016129	0.015873	0.024000	0.031746
accuracy_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
accuracy_valid	0.933333	0.974359	1.000000	0.982906	1.000000	0.982906	0.983333	0.974359	0.965812
accuracy_diff	0.066667	0.025641	0.000000	0.017094	0.000000	0.017094	0.016667	0.025641	0.034188
precision_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
precision_valid	0.935484	0.953125	1.000000	0.968254	1.000000	0.968254	0.968750	0.953125	0.938462
precision_diff	0.064516	0.046875	0.000000	0.031746	0.000000	0.031746	0.031250	0.046875	0.061538
recall_train	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
recall_valid	0.935484	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
recall_diff	0.064516	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
true_positive_train	331	319	331	319	331	319	331	319	319
false_positive_train	0	0	0	0	0	0	0	0	0
false_negative_train	0	0	0	0	0	0	0	0	0
true_negative_train	349	344	349	344	349	344	349	344	344
true_positive_valid	54	53	58	54	58	54	56	53	52
false_positive_valid	4	3	0	2	0	2	2	3	4
false_negative_valid	4	0	0	0	0	0	0	0	0
true_negative_valid	58	61	62	61	62	61	62	61	61

***Table 11 -** Extra Trees Classifier Comparison*



	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	0.862711	0.950326	0.953142	0.939696	0.950121	0.945282	0.936292	0.942261	0.942261
auc_score_valid	0.756674	0.862998	0.849833	0.871926	0.867075	0.861534	0.857341	0.853337	0.853337
auc_score_diff	0.106037	0.087329	0.103309	0.067770	0.083046	0.083748	0.078950	0.088924	0.088924
f1_train	0.868829	0.951825	0.953757	0.941691	0.951009	0.948127	0.939351	0.945245	0.945245
f1_valid	0.775194	0.868852	0.854839	0.876033	0.868852	0.873016	0.866142	0.864000	0.864000
f1_diff	0.093636	0.082972	0.098919	0.065658	0.082156	0.075111	0.073209	0.081245	0.081245
accuracy_train	0.863235	0.950226	0.952941	0.939668	0.950000	0.945701	0.936765	0.942685	0.942685
accuracy_valid	0.758333	0.863248	0.850000	0.871795	0.866667	0.863248	0.858333	0.854701	0.854701
accuracy_diff	0.104902	0.086978	0.102941	0.067873	0.083333	0.082453	0.078431	0.087984	0.087984
precision_train	0.855556	0.956012	0.962099	0.944444	0.956522	0.940000	0.925000	0.937143	0.937143
precision_valid	0.746269	0.868852	0.854839	0.883333	0.883333	0.846154	0.846154	0.843750	0.843750
precision_diff	0.109287	0.087159	0.107260	0.061111	0.073188	0.093846	0.078846	0.093393	0.093393
recall_train	0.882521	0.947674	0.945559	0.938953	0.945559	0.956395	0.954155	0.953488	0.953488
recall_valid	0.806452	0.868852	0.854839	0.868852	0.854839	0.901639	0.887097	0.885246	0.885246
recall_diff	0.076070	0.078822	0.090720	0.070101	0.090720	0.054756	0.067058	0.068242	0.068242
true_positive_train	279	304	318	300	316	298	304	297	297
false_positive_train	52	15	13	19	15	21	27	22	22
false_negative_train	41	18	19	21	19	15	16	16	16
true_negative_train	308	326	330	323	330	329	333	328	328
true_positive_valid	41	48	49	49	51	46	48	46	46
false_positive_valid	17	8	9	7	7	10	10	10	10
false_negative_valid	12	8	9	8	9	6	7	7	7
true_negative_valid	50	53	53	53	53	55	55	54	54

***Table 12-** K-Neighbors Comparison*



	baseline_values	Standard	Outliers_in_Standard	Robust	Outliers_in_Robust	Minmax	Outliers_in_Minmax	Other_Feature_Selection	Other_Feature_Selection_Outliers
auc_score_train	0.997057	0.994186	0.995702	0.998547	0.995624	0.998547	1.000000	0.996979	1.000000
auc_score_valid	0.924082	0.949356	0.942436	0.939696	0.975806	0.973946	0.991935	0.947892	0.939696
auc_score_diff	0.072974	0.044830	0.053266	0.058851	0.019818	0.024600	0.008065	0.049087	0.060304
f1_train	0.997126	0.997093	1.000000	0.997093	1.000000	0.997085	0.998565	0.998544	0.995633
f1_valid	0.929134	0.950000	0.942149	0.943089	0.975207	0.975610	0.991870	0.951613	0.943089
f1_diff	0.067993	0.047093	0.057851	0.054004	0.024793	0.021475	0.006695	0.046931	0.052544
accuracy_train	0.997059	0.996983	1.000000	0.996983	1.000000	0.996983	0.998529	0.998492	0.995475
accuracy_valid	0.925000	0.948718	0.941667	0.940171	0.975000	0.974359	0.991667	0.948718	0.940171
accuracy_diff	0.072059	0.048265	0.058333	0.056812	0.025000	0.022624	0.006863	0.049774	0.055304
precision_train	1.000000	0.997093	1.000000	0.997093	1.000000	1.000000	1.000000	1.000000	0.997085
precision_valid	0.907692	0.966102	0.966102	0.935484	1.000000	0.967742	1.000000	0.936508	0.935484
precision_diff	0.092308	0.030991	0.033898	0.061609	0.000000	0.032258	0.000000	0.063492	0.061601
recall_train	0.994269	0.997093	1.000000	0.997093	1.000000	0.994186	0.997135	0.997093	0.994186
recall_valid	0.951613	0.934426	0.919355	0.950820	0.951613	0.983607	0.983871	0.967213	0.950820
recall_diff	0.042656	0.062667	0.080645	0.046273	0.048387	0.010579	0.013264	0.029880	0.043366
true_positive_train	330	319	331	319	330	319	331	318	319
false_positive_train	1	0	0	0	1	0	0	1	0
false_negative_train	1	4	3	1	2	1	0	1	0
true_negative_train	348	340	346	343	347	343	349	343	344
true_positive_valid	52	54	56	52	58	54	58	52	52
false_positive_valid	6	2	2	4	0	2	0	4	4
false_negative_valid	3	4	5	3	3	1	1	2	3
true_negative_valid	59	57	57	58	59	60	61	59	58

Table 13 - *Bagging Classifier Comparison*

Model	Train Dataset			Validation Dataset			Overfitting	Kaggle Score
	Mean	std	f1	Mean	std	f1		
X_train_MF_OH_MM	1	0	1	0.95	0.02161	0.92562	0.05	0.98924
X_train_MF_OH_MM_FS	1	0	1	0.94551	0.02197	0.93548	0.05449	0.98924
X_train_MF_OH_SS	1	0	1	0.947436	0.02318	0.92562	0.05256	0.98924
X_train_OH_RS	1	0	1	0.955625	0.01619	0.93548	0.04438	0.97872
X_train_OH_SS	0.96177	0.00525	0.96857	0.91781	0.01828	0.92063	0.04396	0.90109
X_train_OH_MM	1	0	1	0.948125	0.02064	0.94574	0.05188	0.96774
X_train_OH_SS	1	0	1	0.95016	0.02259	0.95161	0.04984	0.96774
X_train_MF_OH_SS	1	0	1	0.96827	0.01669	0.98361	0.03173	0.97826
X_train_MF_OH_RS	1	0	1	0.96779	0.01832	0.99187	0.03221	0.97826
X_train_MF_OH_MM	1	0	1	0.96410	0.02055	0.976	0.0359	0.96842
X_train_OH_MM	1	0	1	0.97047	0.012207	0.98413	0.02953	0.97826
X_train_OH_SS	1	0	1	0.97172	0.01519	0.98413	0.02828	0.97872
X_train_OH_RS	1	0	1	0.97078	0.01431	0.98413	0.02922	0.97826
X_train_OH_MM_FS	1	0	1	0.96715	0.01705	0.96774	0.03285	0.97826
X_train_MF_OH_MM_FS	1	0	1	0.96458	0.01727	0.99187	0.03542	0.98924
X_train_MF_OH_SS	0.99995	0.0003	1	0.96362	0.02158	0.96825	0.03632	0.96842
X_train_MF_OH_MM	0.99995	0.0003	1	0.9641	0.02003	0.96825	0.03584	0.98924
X_train_OH_MM	1	0	1	0.97219	0.01305	0.976	0.02781	0.97826
X_train_OH_MM	1	0	1	0.97547	0.01366	0.98413	0.02453	1
X_train_MF_OH_MM	1	0	1	0.96747	0.01904	0.98387	0.03253	1
X_train_OH_MM	0.99729	0.00194	0.99857	0.954375	0.00194	0.95081	0.04292	0.94382

***Table 14 -** Comparison of the different Datasets*