Group Project

“The Smith Parasite”

Machine Learning 2022/2023

MSc. Data Science and Advanced Analytics

Professors:

- ***Roberto Henriques***
- ***Carina Albuquerque***
- ***Ricardo Santos***

Group №: 18

Members:

- ***Duarte Reis Pinto Girão (20220670@novaims.unl.pt)***
- ***Sabeen Mubashar (20220726@novaims.unl.pt)***
- ***Vladislav Abramov (20220729@novaims.unl.pt)***
- ***Wai Kong Ng (20221384@novaims.unl.pt)***

Table of Content

Abstract

Our research aims to develop a best predictive model for estimating the probability of an individual contracting the imaginary disease "*Smith Parasite*", based on the given datasets of 18 variables and a population of 1025 records. We performed preprocessing techniques, including the removal of outliers, One Hot Encoding, Correlation Checks, and the creation of new features. We then, tested a subset of 10 predicting models: "*Logistic Regression*", "*SVM*", "*Decision Tree*", "*MLP Classifier*", "*Random Forest*", and "*Gradient Boosting*", "*AdaBoost*" "*Stacking Classifier*", "CatBoost" and "XGBoost" (the latter two are not part of the "sklearn" library). We used "*F1 score*" as our preferred metric for assessing the model performance and we applied "K-Fold" Cross Validation with 10 splits and "GridSearch" to find out the best parameter combination for each model. As a result, we found that "*Gradient Boosting*" outperformed the other models introduced in "sklearn" library in terms of "*F1 score*", with a final score of **~0.9871** and "CatBoost" is the leader among all models we've researched with F1 score equal to **~0.9894** .

Keywords: Machine Learning, Supervised Learning, Predictive Models, Data Pre-processing, Feature Engineering, Evaluation Metrics, CatBoost and Gradient Boosting.

1. Introduction

A new disease called "*Smith Parasite*" has already affected five 5,000 people in England, with no apparent link between them. The most noticeable symptoms include fatigue and fever, as well as other unknown symptoms.

At the moment, the scientific knowledge about the virus is still very limited. The conditions of the transmission of the virus and the long-term consequences are two good examples of the parameters we know very few about.

Therefore, as data scientists, it is our goal to identify which individual characteristics make someone more likely to contract the virus and to create a prediction model based on these findings to prevent the number of infections from growing exponentially. In summary, our research aims to answer the following question: "*What are the main causes of the Smith Parasite?*"

2. Exploration

2.1. Initial Data Sets Combination

The data for the project was provided in a fragmented form, including three files containing the training datasets and three files containing the test datasets. Separate files containing either demographic, health or habits data, were merged into one DataFrame. This aggregation procedure was performed for both the training and test samples, whose final dimensions were 800 x 18 and 225 x 17 respectively.

2.1.1 Key Note About our Approach on Project

During the external data analysis phase, looking for insights, outliers, correlations, and other statistical indicators, it was decided to combine the test and training parts together in order to get a better understanding of data behaviour, creating the dataset called "*data*" in our notebook. Although there is no such possibility in real-life scenarios, our team was aiming to obtain the highest score in the Kaggle competition. That's why, in future steps we obtained outlier boundaries with respect to the whole dataset. (It was also discussed with the teacher)

2.2. Variables

The merged dataset we are working with called " *data*" includes 18 descriptive variables of each patient and one dependent variable ("*Disease*" whether there is a disease or not).

We divided descriptive variables into 3 groups: continuous (in numeric format), which includes "*Birth_Year*", "Blood_Pressure", "*Height*", "*High_Cholesterol*", "*Mental_Health*", "*Physical_Health*" and

“*Weight*”; categorical (in text format), which includes “*Exercise*”, “*Region*”, “*Name*” and “*Smoking_Habit*”; ordinal (in text format) which includes “*Checkup*”, “*Diabetes*” “*Drinking_Habit*”, “*Education*”, “*Fruit_Habit*” and “*Water_Habit*”. See **Table 1**.

For each group of explanatory variables, we have adopted a specific approach, including performing various checks, transformations, and generating new features that are tailored to the characteristics of that group (Sex, Age etc). Additionally, "PatientID" is a unique identifier for each individual in the dataset which was not included in training procedures, but was used for the submission.

We also checked the target variable “Disease” for class balance, and as we can see on distribution blots classes are mainly balanced.

2.3. Distributions and Outliers Detection

As shown in **Graph 1***,* most of the continuous variables are relatively normally distributed. However, we observed that "High_Cholesterol" and "Blood_Pressure" are slightly left-skewed, "Mental_Health"

is slightly right-skewed, and "Physical_Health" is strongly left-skewed. **Table 2** presents the means, standard deviations, minimums, lower quartiles (Q1), medians, upper quartiles (Q3), and maximums of each continuous feature, as well as the number of records with missing values (**Table 1**).

Based on the histograms in **Graph 1***,* we suspect the presence of some outliers in the “*Birth_Year*”, “Blood_Pressure”, “*High_Cholesterol*” and “Mental_Health”. To confirm this, we also plotted BoxPlots, represented on **Graph 2***,* which clearly show the presence of outliers.

2.4. Missing Values

For each predictor variable, we conducted a check for missing values using the "*info()*" method, which is represented in **Table 2**. The column "*Non-Null Count*" indicates the number of non-null values for each variable. As we can observe, fortunately, all explanatory variables except “*Education*” were fully populated. There are 13 missing values in the column which explains the level of education. Regarding “Disease”, the missing values correspond to test records, so there is no need to make any actions on them.

3. Preprocessing

3.1 Feature Creation and Feature Removal

In this step we created two new variables using given data: “*Age*” and “*Sex*”. To create "*Age*", we calculated the difference between the year of the report (2022) and the patient's birth year, using the "*Birth_Year*" variable. To create "*Sex*", we used the prefixes "*Mr.*" and "*Mrs.*" in the "*Name*" variable to

create a binary variable indicating the patient's gender. "*Mr.*" indicates a male patient, while "*Mrs.*" indicates a female patient.

After creating these two new predictors, we **removed** original ones, such as "*Birth_Year*" and "*Name*", in order to reduce and clean the dataset. Additionally, we used the **Random Forest Feature Importance** to assess the relative importance of each variable in the model (see **Graph 3**). From this interpretation, we concluded that the variables "*Education*" and "*Region*" were the least relevant features in our predictive model compared to the remaining ones (less than 0.01). Therefore, we also removed all variables generated "*Education*" and "*Region*" with use of One Hot Encoding procedure. We did the Random Forest Map of Feature Importance one more time to get an overall view of relevancy of the new dataset features (see **Graph 4**).

3.2 Feature Transformation

To gain a deeper understanding of our dataset, we decided to transform our ordinal group of variables from text to numeric format. In doing so, we assigned values to variables according to the likelihood of each person contracting an illness in practical terms. The more likely for the individual to contract the disease, the higher the value it received. For example, the variable "*Drinking_Habit*" has three unique values: "*I do not consume any type of alcohol*", "*I consider myself a social drinker*" and "*I usually consume alcohol every day*". We replaced these values with "*0*", "*1*", "*2*", respectively. By applying our personal knowledge on top of our research, we concluded that daily alcohol consumption is likely to have a negative effect on a person's health, increasing their likelihood of contracting "*Smith Parasit*" disease. Therefore, we applied this reasoning for all the ordinal variables, except "*Education*": "*Checkup*", "*Diabetes*", "*Drinking_Habit*", "*Fruit_Habit*" and "*Water_Habit*".

In addition, we also decided to transform our discrete group of variables, by replacing the binary variables "*Exercise*" and "*Smoking_Habit*" into either "*0*" or "*1*". For these procedures, we used the same rationale as we did for the ordinal variables, assigning the value "1" to the higher probability of contracting "*Smith Parasit*" disease. Finally, the "*Region*" variable has only 10 equally significant unique values (see **Table 3**), which cannot be sequenced in any way, but can be transformed into 9 new binary predictors. In this sense, we used the One Hot Encoding procedure.

Exception:

Although Education may also be assumed as an Ordinal group, as long as it has missing values, we also applied the One Hot Encoding, once it will automatically show us null values as zeros in all meaningful values of the "Education" variable.

3.3 Outliers replacement

As we previously observed during the data exploration phase (see **section 2.2**), there were obvious outliers at least in the variables '*Age*', 'Blood_Pressure', '*Height*', '*High_Cholesterol*', '*Weight*'. In order

to clean our dataset, we decided to replace all the outliers with the first (**Q1**) and third (**Q3**) quartiles values using the **Interquartile Range** (**IQR**) method. This process allowed us to discard the outliers and remove long tails in the distributions of the variables on which the procedure was performed, resulting in a cleaner and more accurate dataset.

3.4 Missing values solution

As mentioned above, missing values were found in the "*Education*" variable only, 13 items in total. Although "*Education*" also belongs to the Ordinal group, as long as it has missing values, we also applied an elegant solution, the One Hot Encoding. It automatically fills all the null values as zeros in meaningful column-values. So all values of "*Education*" carrying information (non empty) were transformed into binary columns, and in case a value was missing, all variables were equal to 0. Thus we do not need any nan-filling/vanishing procedures.

4. Modelling

In order to find the best model, we tried and tested several different methodologies: regression, support vector machine, trees, ensembles of trees and gradient boosting. In addition to that we compared the efficiency of the best models with the stacking classifier model, which is actually an ensemble of various classifiers.

Due to project limitations, most of the presented ML algorithms were taken from sklearn Python package, nevertheless, the report contains two unexplored on the course models both based on gradient boosting approach: **CatBoost** and **XGBoost**. For each model, the Stratified K-Folds method was applied to find the optimal intervals of key hyperparameters' values. Then GridSearch was performed based on the obtained lists of values in order to find the best combinations of hyperparameters.

4.1 Logistic Regression (Baseline Model)

For the baseline model it was decided to take an ordinary logistic regression. We tested the quality of work on both scaled and unscaled data and compared how the model behaves with Ridge and Lasso regularizations. We obtained the best grid of hyperparameters where regularisation is Ridge (L2) and solver is 'lbfgs' and it is better to use an unscaled dataset. The F1 measure showed a **~0.834** score on average, which is not very bad, but we have a lot of potential to grow.

4.2 Support Vector Machine (SVM)

This model is based on the support vectors, which are the lines that divide the several classes of the dataset. Typically, the SVM model is effective in high dimensional spaces and is very adaptive, since we can apply different "*kernel*" functions for the decision function. As well as for Logistic Regression we tested data scaling on the given data and for the SVM model it was really needed. We achieved **F1= ~0.891** for the default SVM with scaled data against **~0.713**, which proves the necessity of rescaling during Support vector machine impementation. With use of stratified K-Folds we obtained most suitable ranges of hyperparameters which were passed to the GridSearch procedure. The best parameters are: *'C' = 10, 'kernel' = "rbf"*.The F1 measure showed a **~0.946** score on average, which is a huge boost, compared with Logistic Regression and even though we have not tried tree-based algorithms yet, which are assumed to perform even better.

4.3 Decision Tree Classifier

As mentioned above, we also resorted to tree-based algorithms to solve the classification problem. For this purpose, we took a decision tree as a baseline of tree-models, found the most effective hyperparameters (**Graph 7**), and obtained a quality increase of F1 up to **~0.970**. The best parameters are: *'criterion' = "entropy", 'max_depth' = 12*. We also compared our model with the default one, which comes with the "sklearn" package, so in the result the metrics of the obtained model are higher than the default model gives (**~0.962** for F1). So, on this positive trend we moved to another model, based on the decision trees.

4.4 Random Forest

We continued the theme of trees using a random forest model. It consists in using several decision trees at once, which in theory should lead to an improvement of the predicting model performance. This is what happened, but all the variants of the hyperparameter set we examined could not surpass the default set in terms of all metrics we used (Accuracy, Precision, Recall, F1). Thus, the default values of parameters were selected: *'max_depth' = None, 'n_estimators' = 100*. And obtained an F1 score using K-Folds method with cross validation equal to **~0.978 (Graphs 8-9)**. In addition to that, the predictions of the default Random Forest based on the test set were submitted into Kaggle competition and we achieved a score equal to **1**. But we wouldn't be us if we didn't keep looking for a better model.

4.5 MLPClassifier

In the process of searching for the best model, we also explored neural network approaches. One of them is MLPClassifier which has a Multi-layer Perceptron structure. However, no matter how we tried to find the best hyperparameters for this model, it could not overcome even the decision tree score, outperforming only the logistic regression model. The best obtained parameters are: *'activation' = 'tanh', 'hidden_layer_sizes' = 78, 'learning_rate_init' = 0.001*. And the corresponding F1 score is **~0.856**. The default MLPClassifier classifier has performed much worse: F1 = **~0.816**.

4.6 Gradient Boosting

We have made a separate emphasis on the gradient boosting model. The first and as it turned out the best was Gradient Boosting Classifier. As with all models, we used K-Folds Cross Validation on the entire training sample **(Graphs 10-11)**. First, we found the most appropriate intervals of parameter values, and then with the GridSearch procedure we found the best combination which is: *'learning_rate' = 0.1, 'max_depth' = 6, 'n_estimators' = 500*. With use of these parameters we achieved an F1 score equal to **~0.987** which was the best result of all the models provided by the "sklearn" library. In addition to Gradient Boosting we also examined the AdaBoost model and two beyond the course models such as XGBoost and Catboost which will be described in the following paragraphs. This model brought us the final solution we submitted as the best one. (**We achieved a score equal to 1 in Kaggle**).

4.7 AdaBoost Classifier

For this model on the first step we decided to find the best baseline mode used in AdaBoost. Having all hyperparameters fixed we found that RandomForest classifier is the most suitable baseline model compared with Decision Tree and Logistic Regression (**~0.98**1 against **~0.967** and **~0.834** respectively for F1 measure). After this step, we found the best maximum depth for the baseline model (Random Forest). So, the achieved F1 score is **~0.981** which is less than the Gradient Boosting model provides.

4.8 XGBoost Classifier (additional model beyond "sklearn")

In addition to the basic knowledge gained in the lessons, we decided to try a few more models of gradient boosting, one of them is XGBoost. XGBoost is a well optimised gradient boosting model designed to be highly efficient in classification and regression tasks. XGBoost provides a parallel tree boosting (GBDT, GBM) that may solve many data science problems with high accuracy. The model requires a high variety of hyperparameters, so we tested the main ones: **"n_estimators"** - number of gradient boosted trees, **"max_depth"** - maximum depth of those trees, **"learning_rate"** and **"reg_alpha"** & **"reg_lambda"** as regularisation (L1 & L2) parameters. The best obtained hyperparameter combination is: *'learning_rate' = 0.5, 'max_depth' = 7, 'n_estimators' = 550* and corresponding F1 score is **~0.985** which is better than AdaBoost but still is lower then Gradient Boosting Classifier.

4.9 CatBoost Classifier (additional model beyond "sklearn")

CatBoost is an open-source software library-model developed by Yandex (Russian search engine IT corporation). It provides a gradient boosting framework which among other features attempts to solve

for categorical features using a permutation driven alternative compared to the classical algorithm. Compared with all other models it performed the best results, unfortunately we are not able to use it due to competition rules. Nevertheless, we ran a familiar parameter selection pipeline **(Graphs 12-14)** and found the best one: *'depth' = 8, 'iterations' = 250, 'learning_rate' = 0.2*. CatBoost with these parameters showed the best results on all metrics at once: **F1 = ~ 0.9894**, **Accuracy = ~0.9891**, **Recall = ~0.9894** and **Precision = ~0.9897**

4.10 Stacking Classifier

After comparing metrics of all models we selected several combinations of the best ones and most variegated ones at the same time. So, we've tried combinations: Gradient Boosting + Random Forest and final estimator as Random Forest (F1 = **~0.984**), Gradient Boosting + Random Forest + SVM and final estimator as Random Forest (F1 = **~0.989**) and Gradient Boosting + Random Forest + SVM + AdaBoost and final estimator as Random Forest (F1 = **~0.989**). But none of them overcome Gradient Boosting (the best performing model).

5. Assessment

In this section, we will compare the performance of the models we have used in our report based on their ability to correctly predict the probability of having the disease "Smith Parasite". After providing a brief context for the models, we will evaluate their performance using various metrics.

5.1 Models

As previously stated in part 4, we have used ten predicting models: eicht from "sklearn" package, "*Logistic Regression*", "*SVM*", "*Decision Tree*", "*MLP Classifier*", "*Random Forest*", *"AdaBoost"* "*Gradient Boosting*" and "*Stacking Classifier*", as well as two additional models "*CatBoost*" and "*XGBoost*" that are not part of the "sklearn" library.

5.2 Parameters and Metrics Used

For each model, we explored the impact of various parameters on its performance. By varying these parameters and keeping the remaining ones constant, we were able to gain a deeper understanding of how the model's score varied with changes to the parameters. First, we examined how the model's score was affected by the main parameters in order to narrow down the range of values to include in our "*Grid Search*". After this initial analysis, we used the "*Grid Search*" method to find the best combination among these parameters, rather than relying on a trial-and-error approach, as there are hundreds of potential combinations to consider. By using "*Grid Search*", we were able to find the best combination of parameters for each model based on the following four metrics: "*accuracy*", "*f1 score*", "*recall*" and "*precision*". "*Accuracy*", obtained through the formula (TN + TN) / (TP + FN + TN + FP),

tells us the percentage of correct predictions (either they are positive or negative). "*Recall*" obtained through the formula (TP) / (TP + FP), tells us the fraction of actual positive cases that were correctly predicted. "*Precision*" obtained through the formula (TP) / (TP + FN), tells us is the fraction of predicted positive cases that were truly positive. And finally, "*f1 score*" obtained through the formula 2x[(Precision x Recall)/(Precision + Recall)], which can be interpreted as the model's balanced ability to both capture positive cases (*recall*) and be accurate with the cases it does capture (*precision*).

Note: TP (total positives), TN (total negatives), FN (false negatives) and FP (false positives).

5.3 Assessment Comparisons

As we can see in **Table. 4**, in all four metrics for all models, the "CatBoost" classifier with obtained parameters performed the best, with an F1 score of **~0.987**. This means that the model's ability to both positively identify cases and be accurate with the cases it does identify is **~0.989**, which, in our opinion, is a very good performance. This prediction model was followed by "Gradient Boosting" under GridSearch, "XGBoost", "AdaBoost" and "RandomForest (with default parameters)", taking second, third, and fourth place, respectively. The models that performed the worst were "Logistic Regression" and "MLP Classifier", with F1 scores of **~0.834** and and **~0.855**, respectively.

5.3.1 Parameters Gradient Boosting

Among the best parameters we found for "*Gradient Boosting*" model, we would like to highlight a few: "*learning_rate*", which is set to "0.1" and represents the step size at which the optimizer makes updates to the model parameters, "*max_depth*", which is "6"; "*number of estimators*", which is set to "*500*" and represents the quantity of trees in the forest and "*random_state*", which is set to "*42*".

6. Conclusion

The goal is to build a predictive model that answers the question, "Who are the people more likely to suffer from the Smith Parasite?" even though the dataset was in good condition. There were missing values in the 'Education' variable (13 values).

Moreover, During feature creation, two variables were created: Age and Sex using 'Name' and 'Birth_Year'. We also used the Random Forest Map of Importance to assess the relative importance of each variable in the model and drop the less important variables which include Education. For the maximum benefit of our dataset, we have performed feature transformation. In addition to that the dataset has some outliers as it can negatively affect statistical analysis and could result in low accuracy, to deal with outliers is important which were replaced by the IQR method, and no outlier has been removed. We performed 'heatmap' (**Graph 5**) for the correlation check and no highly correlated variable was found, except Sex and Height (~ -0.6).

To find the best model with testing different algorithms which include Logistic Regression, Support Vector Machine, Decision tree, MLP Classifier, ensemble models like Random Forest, Gradient Boosting, AdaBoost. Then for each model the GridSearch procedure was performed to find the best combinations of hyperparameters. In addition, we also implemented **CatBoost** and **XGBoost** as additional (unknown on the course) models for prediction. Finally, we compared the efficiency of the best models with the **Stacking Classifier** model, which is actually an ensemble of various classifiers and in result **Gradient Boosting Classifier** is the best model from "sklearn" with **~0.987** F1 score while **CatBoost Classifier** has the best performance overall with **~0.989** F1 score. However, we have to keep the limitations of the Kaggle competition in mind and we choose the **Gradient Boosting Classifier model** for our final submission for the Kaggle competition and successfully achieve accuracy equal to 1 **and become leaders in the competition**.

7. References

[1] sklearn´s documentation (2022), sklearn.model_selection.GridSeachCV

Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[2] sklearn´s documentation (2022), sklearn.model_selection.RandomForestClassifier

Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] sklearn´s documentation (2022), sklearn.model_selection.GradientBoostingClassifier

Available at: :https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[4] sklearn´s documentation (2022), sklearn.model_selection.f1_score

Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[5] LT, Z (2021). *Essentials Things You Need to Know About F1-Score*

Available at: https://towardsdatascience.com/essential-things-you-need-to-know-about-f1-score-dbd973bf1a3

[6] Kanstrén, T (2020). *A Look at Precision, Recall, and F1-Score*

Available at: https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

[7] Allwright, S (2022). *How to interpret F1 score (simply explained)*

Available at: https://stephenallwright.com/interpret-f1-score/

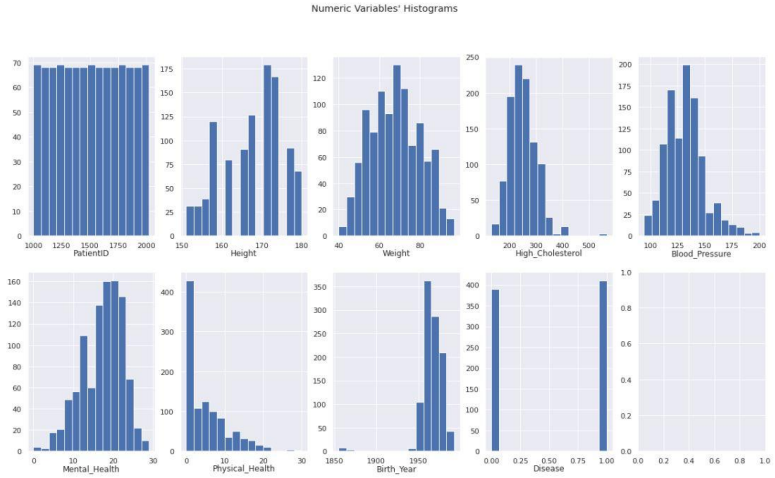
[8] Reference for CATBoost and XGBoost

CatBoost: https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier

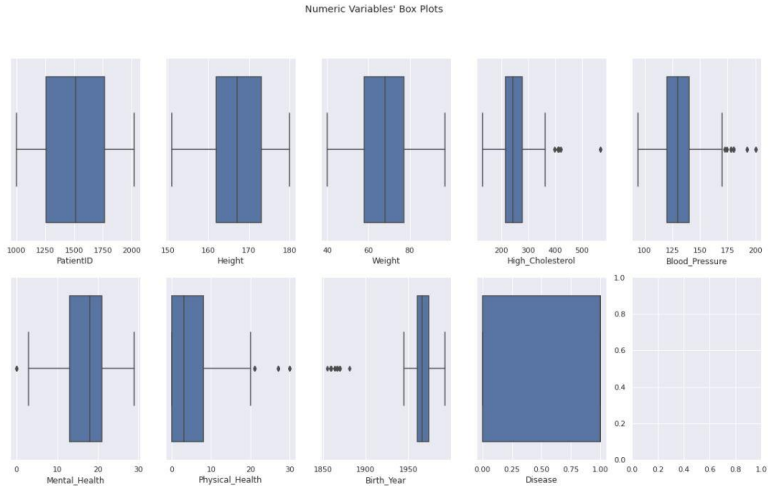
XGBoost: https://xgboost.readthedocs.io/en/stable/python/python_api.html

8. Annexes

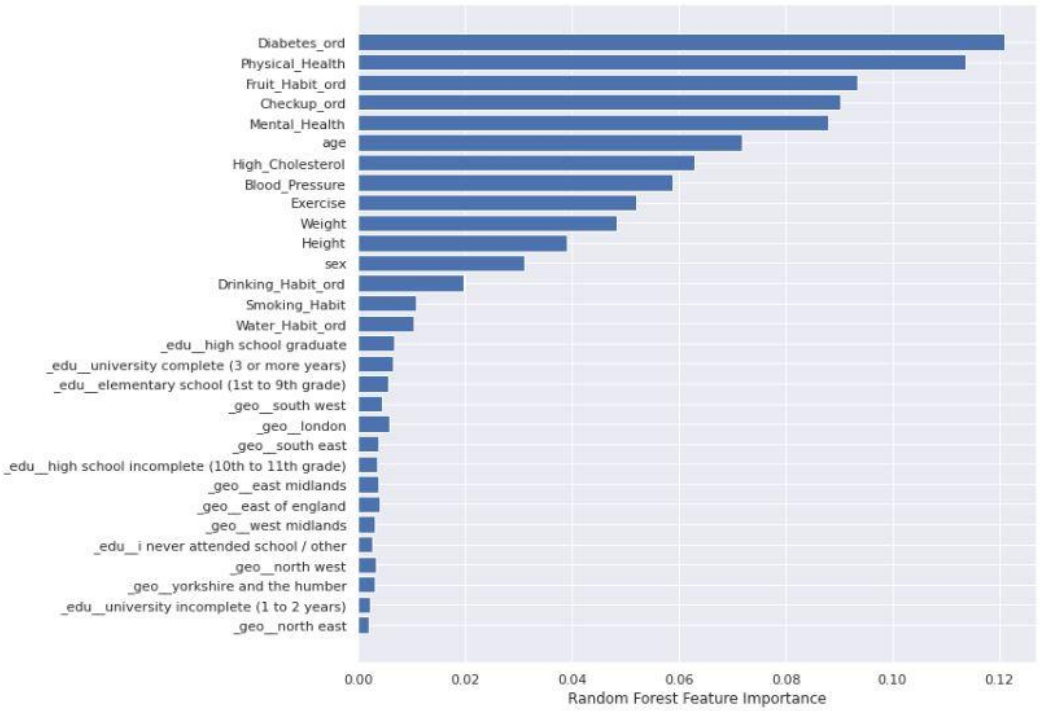
8.1. Graphs



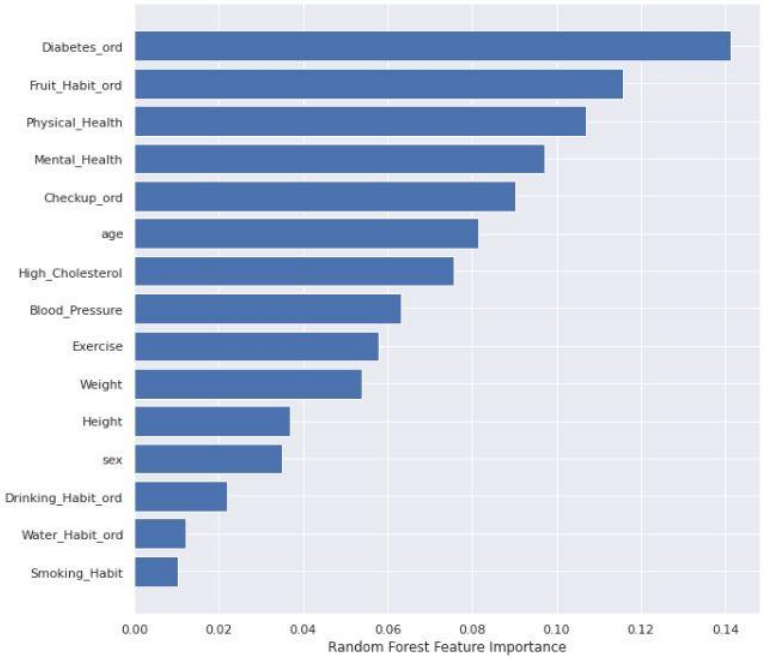
Graph 1 – "*Initial Numerical variables Histograms (Distributions)*"



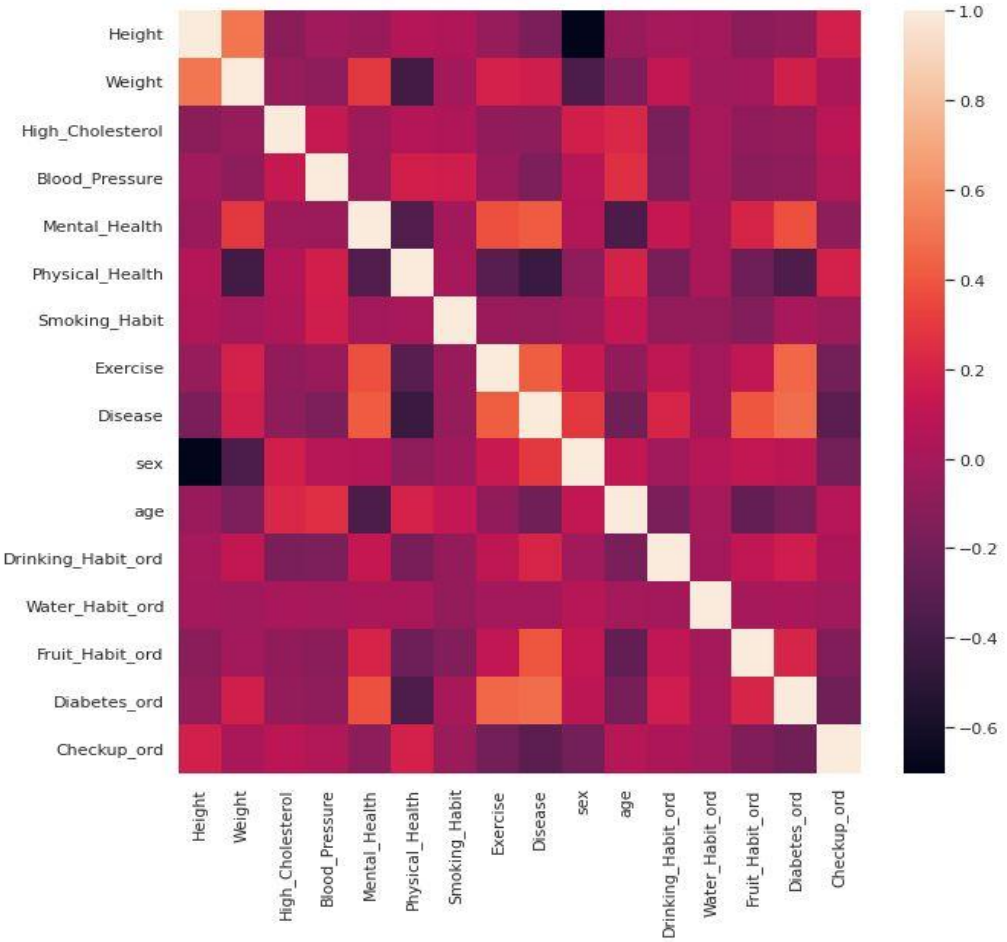
Graph 2 - "*Numerical variables Box Plots (Distributions)*"



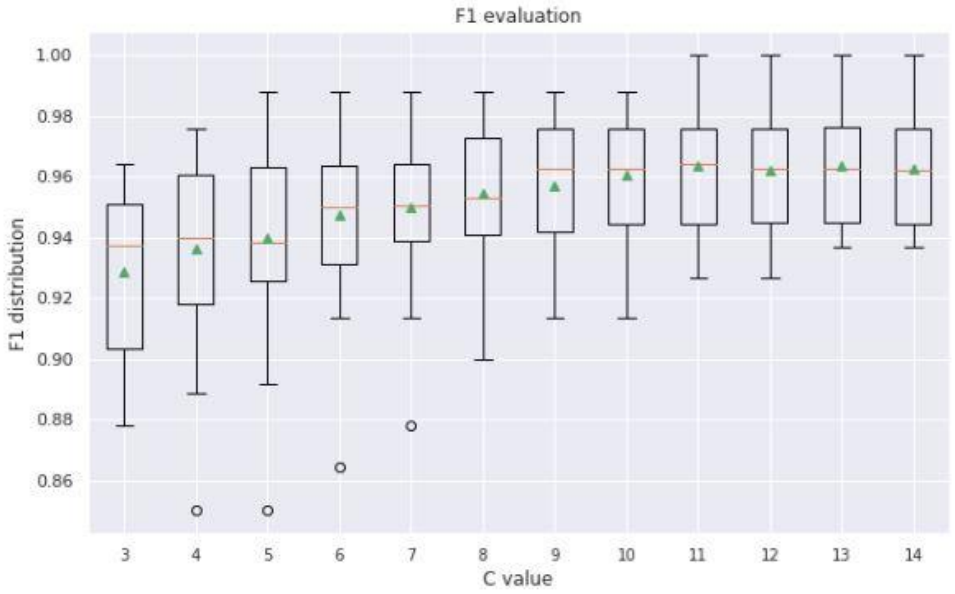
Graph 3 - "*RandomForest Feature Importance Before FeatureRemoval*"



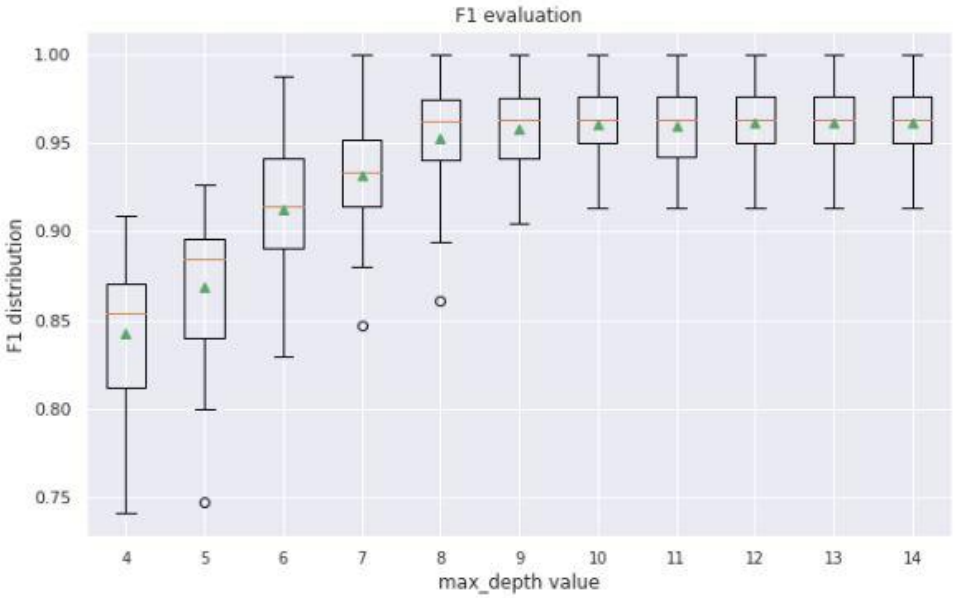
Graph 4 - "*RandomForest Feature Importance after Feature Removal*"



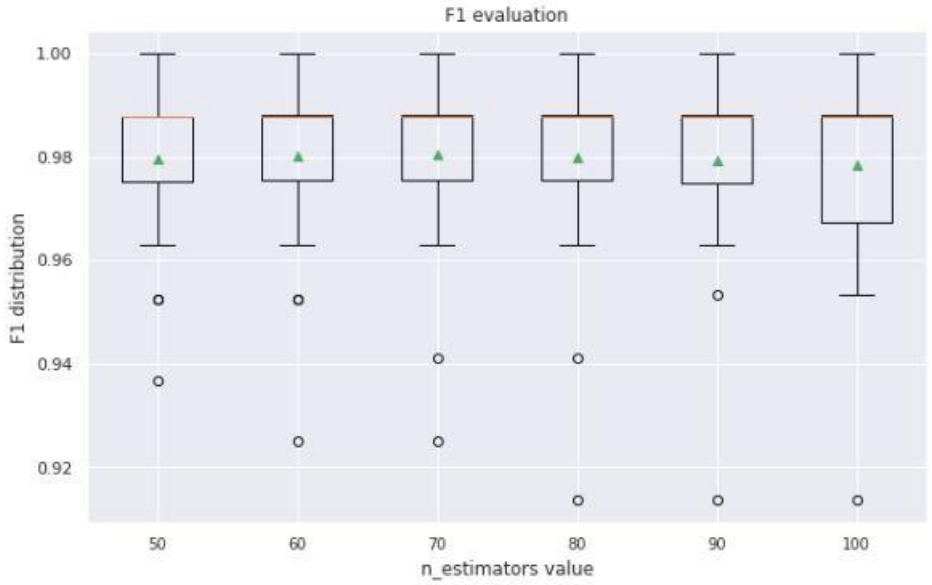
Graph 5 - "*Correlation Check Matrix*"



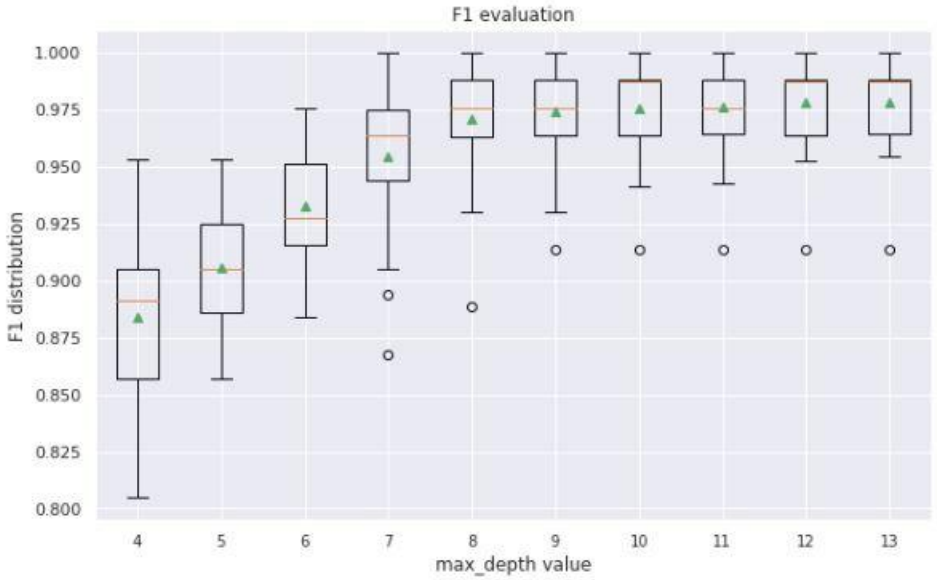
Graph 6 - "*SVM: F1 score per C-Value*"



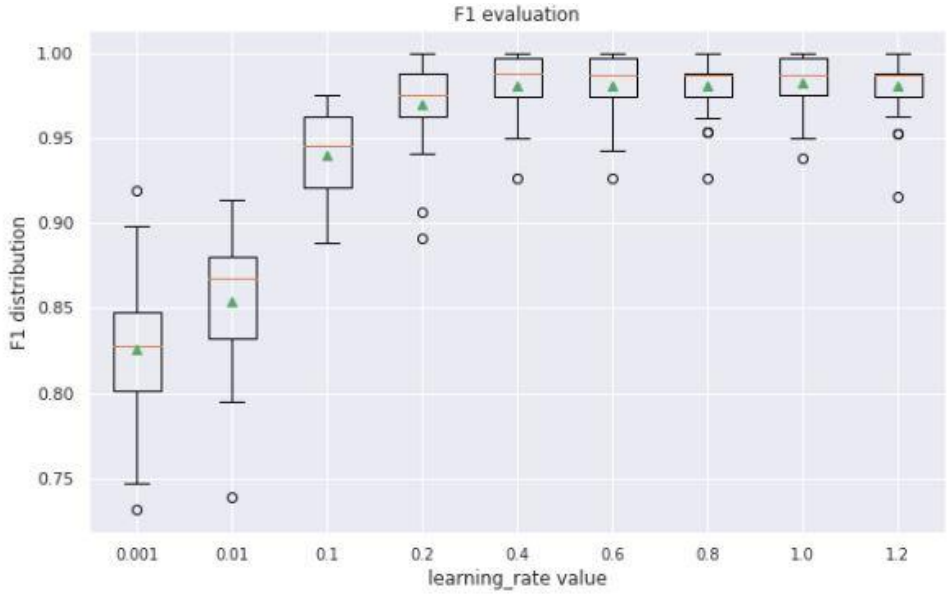
Graph 7 - “Decision Tree: F1 score per maximum depth value”



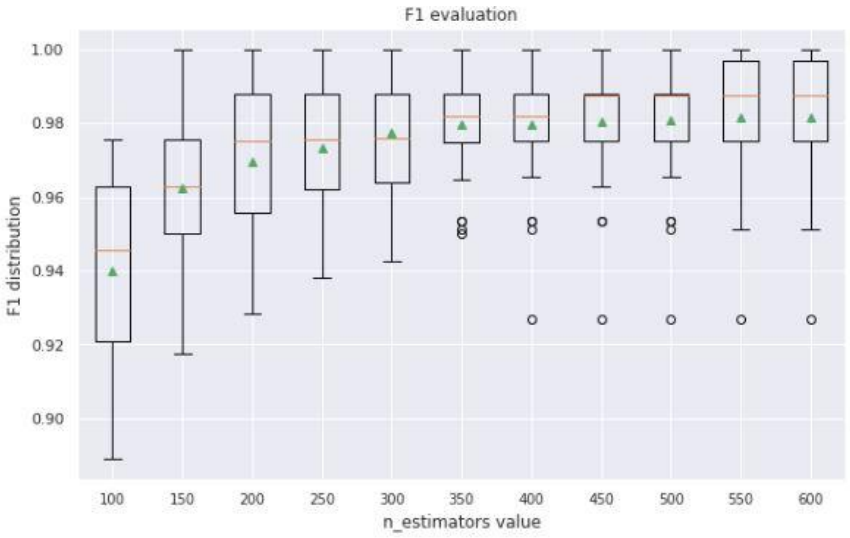
Graph 8 - “Random Forest: F1 score per n_estimators value”



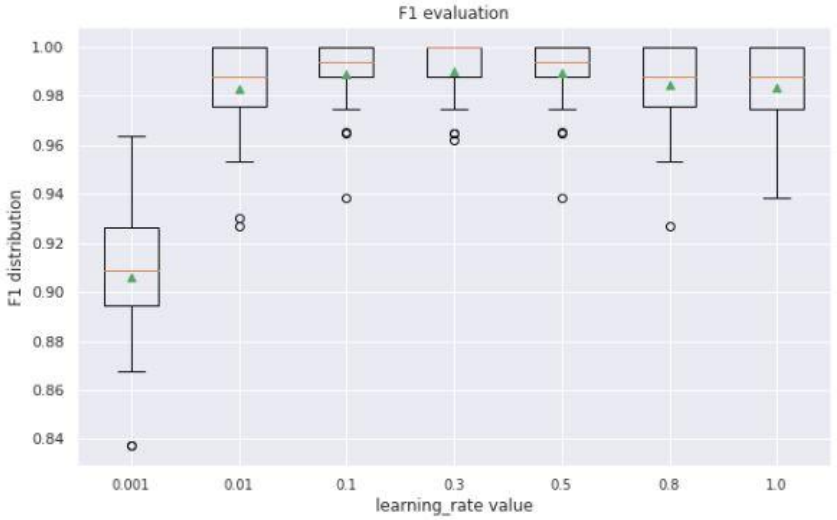
Graph 9 - "Random Forest: F1 score per maximum depth value"



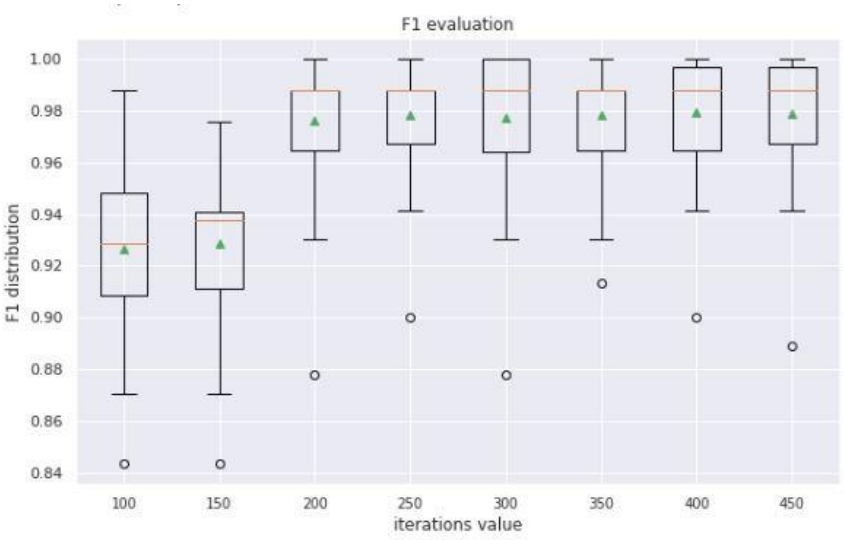
Graph 10 - "Gradient Boosting: F1 score per learning rate value"



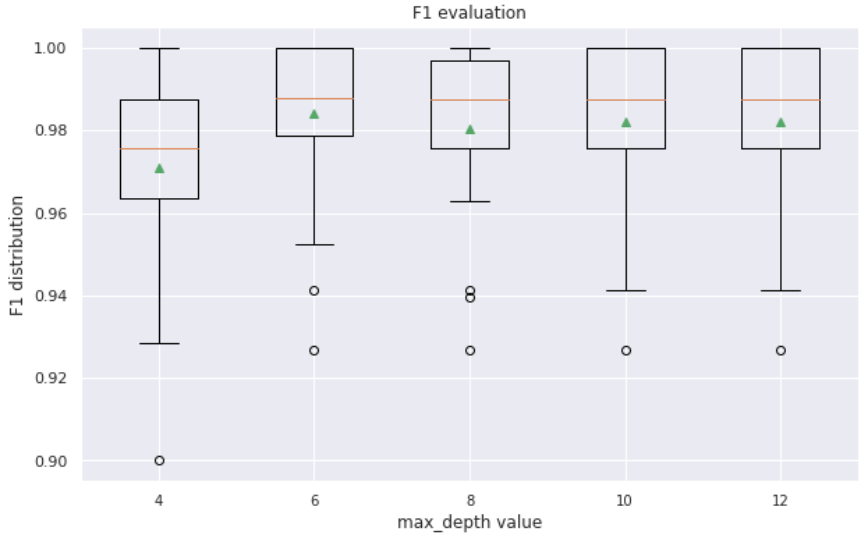
Graph 11 - “Gradient Boosting: F1 score per number of estimators”



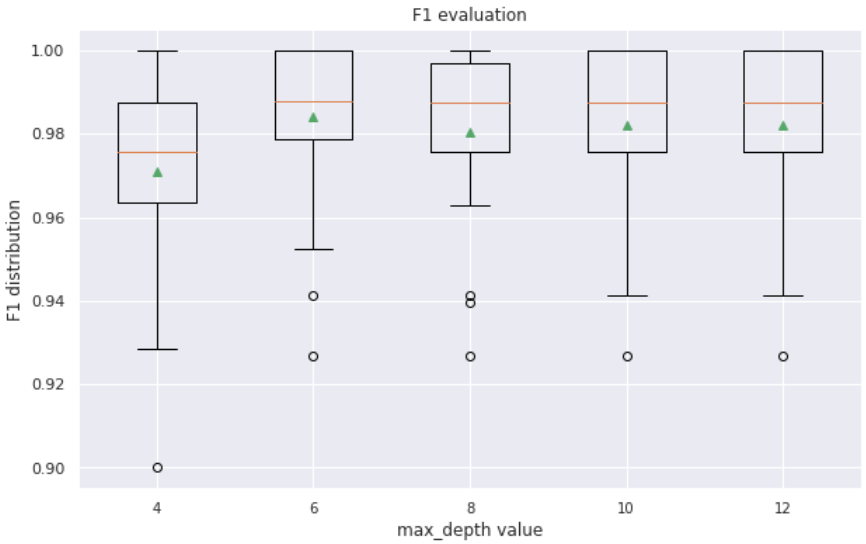
Graph 12 - “CatBoost: F1 score per learning rate value”



Graph 13 - *"CatBoost: F1 score per iterations"*



Graph 14 - *"CatBoost: F1 score per maximum depth value"*



Graph 15 - “XGBoost: F1 score per max_depth”

Graph 16- “Comparison of models (F1 score)”

8.2. Tables

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   PatientID         1025 non-null   int64
 1   Name              1025 non-null   object
 2   Birth_Year        1025 non-null   int64
 3   Region            1025 non-null   object
 4   Education         1012 non-null   object
 5   Disease           800 non-null    float64
 6   Height            1025 non-null   int64
 7   Weight            1025 non-null   int64
 8   High_Cholesterol  1025 non-null   int64
 9   Blood_Pressure    1025 non-null   int64
 10  Mental_Health     1025 non-null   int64
 11  Physical_Health   1025 non-null   int64
 12  Checkup           1025 non-null   object
 13  Diabetes          1025 non-null   object
 14  Smoking_Habit     1025 non-null   object
 15  Drinking_Habit    1025 non-null   object
 16  Exercise          1025 non-null   object
 17  Fruit_Habit       1025 non-null   object
 18  Water_Habit       1025 non-null   object
dtypes: float64(1), int64(8), object(10)
memory usage: 152.3+ KB
```

Table 1 – “Entire Data General Information”

	PatientID	Birth_Year	Disease	Height	Weight	High_Cholesterol	Blood_Pressure	Mental_Health	Physical_Health
count	1025.000000	1025.000000	800.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000
mean	1512.000000	1966.395122	0.513750	167.721951	67.821463	250.00000	131.611707	17.389268	4.738537
std	296.036315	14.335373	0.500124	7.982877	12.251687	51.59251	17.516718	5.281313	5.596449
min	1000.000000	1855.000000	0.000000	151.000000	40.000000	130.00000	94.000000	0.000000	0.000000
25%	1256.000000	1961.000000	0.000000	162.000000	58.000000	215.00000	120.000000	13.000000	0.000000
50%	1512.000000	1966.000000	1.000000	167.000000	68.000000	244.00000	130.000000	18.000000	3.000000
75%	1768.000000	1974.000000	1.000000	173.000000	77.000000	279.00000	140.000000	21.000000	8.000000
max	2024.000000	1993.000000	1.000000	180.000000	97.000000	568.00000	200.000000	29.000000	30.000000

Table 2 – “Entire Data Main Statistics per Variable”

```
[ ] # Get numbers of unique observations
    for col in data.columns:
      print(f'{col}: {data[col].nunique()} unique values')
```

```
PatientID: 1025 unique values
Height: 15 unique values
Weight: 58 unique values
High_Cholesterol: 152 unique values
Blood_Pressure: 49 unique values
Mental_Health: 28 unique values
Physical_Health: 24 unique values
Checkup: 4 unique values
Diabetes: 4 unique values
Smoking_Habit: 2 unique values
Drinking_Habit: 3 unique values
Exercise: 2 unique values
Fruit_Habit: 5 unique values
Water_Habit: 3 unique values
Name: 1023 unique values
Birth_Year: 50 unique values
Region: 10 unique values
Education: 6 unique values
Disease: 2 unique values
```

Table 3 – *"Number of unique values per variable"*

Model	F1	Accuracy	Recall	Precision
Logistic Regression	0.833707	0.826250	0.847522	0.823877
MLPClassifier	0.855733	0.848750	0.871874	0.843274
Support Vector Machine	0.960587	0.959583	0.958614	0.963536
DecisionTree	0.970472	0.970000	0.961092	0.980962
Random Forest	0.978531	0.977917	0.978126	0.979343
AdaBoost	0.980721	0.980000	0.983779	0.978233
XGBoost	0.985083	0.984583	0.985424	0.985166
Gradient Boosting	0.987121	0.986667	0.990263	0.984440
CatBoost	0.989466	0.989167	0.989450	0.989745

T**able 4** – "Model Comparison"