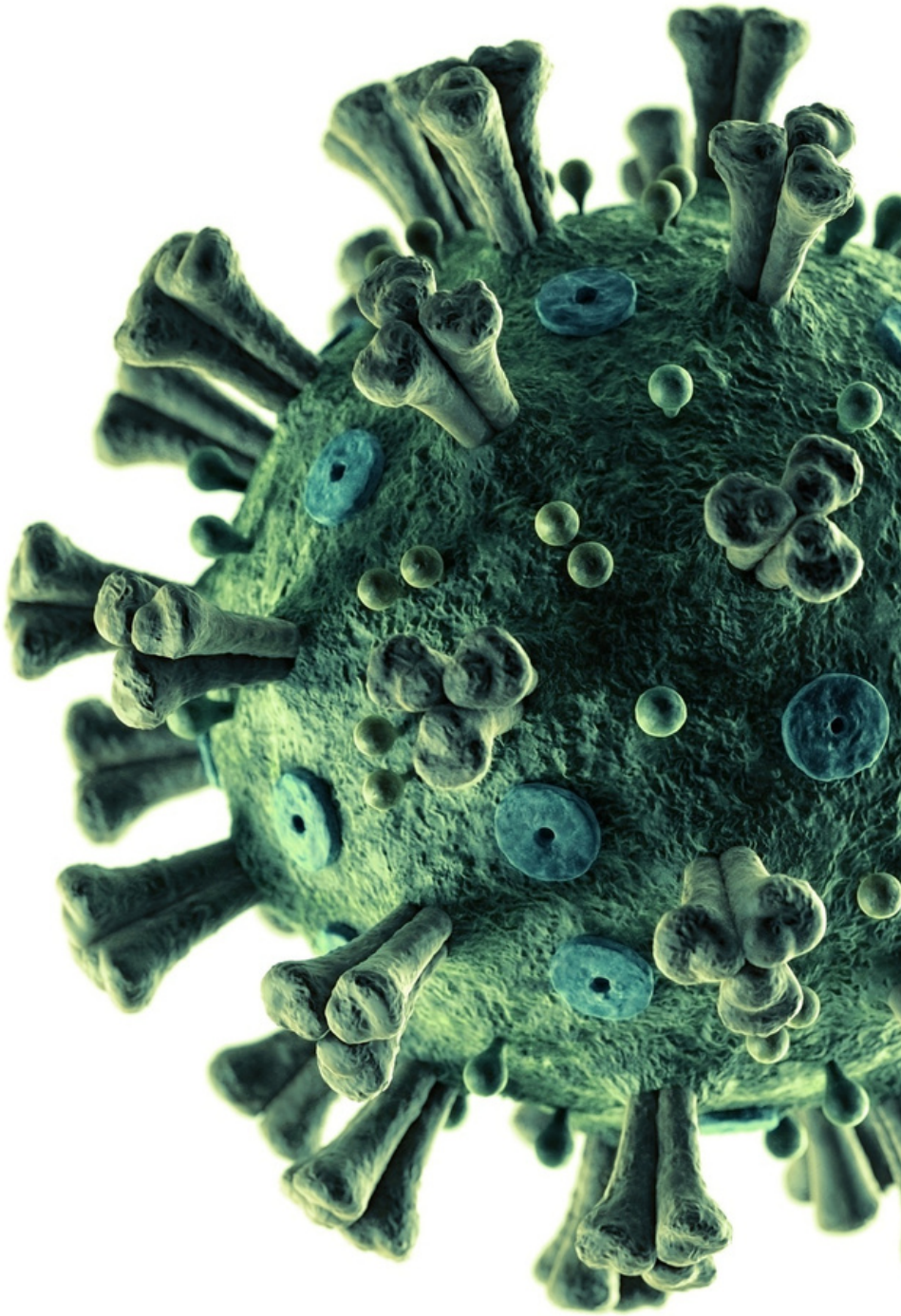
# The Smith Parasite



**Team 22:**
Luís Gonçalves 20220624
Margarida Ferreira 20220677
Mariana Água 20220704
Nádia Carvalho 20220700

Master in Data Science and Advanced Analytics with specialization in Business Analytics

Machine Learning

# INDEX

## INTRODUCTION

In England, a new disease has been recently discovered by Dr. Smith and has affected more than 5000 people with no apparent connection between them. The conditions of the transmission of the disease are still unknown and there are no guarantees about what causes a patient to suffer or not from it.

Hence, the following study is intended for the Machine Learning Curriculum Unit and the aim of this report is to answer the question “Who are the people more likely to suffer from the Smith Parasite?” by building a predictive model for a dependent variable (Disease) and be able to forecast if a patient will suffer, or not, from the Smith Disease.

With that in mind, we will analyse and transform the data available to apply different models to answer the previously defined question with more accuracy. As a matter of fact, we will base our research and modelling on a data set with 800 observations that provide us a truthful representation of England’s population and be able to take the most precise conclusions. Several processes were applied, including outliers’ detection, data normalization and feature selection moreover the election of the best model based on Logistic Regression, Gaussian NB, KNN, Neural Networks, Decision Trees, Random Forest, Gradient Boosting, Support Vector Machines, AdaBoost, XGBoost.and StackingClassifier.

This report documents all the steps executed in Jupyter Notebook, from data exploration to the construction of the predictive model and contains our conclusions about the project. Finally, the methodology used was literature research along with the subject matter taught throughout the semester.

## EXPLORATION

First of all, in order to better understand the dataset that we were dealing with, our group decided to conduct an explanatory study by generating some statistics and observing graphics. This step enabled us to understand how our data behaved and to try to spot any trends or issues that may needed to be rectified.

As a starting point and based on our notebook, we verified that our problem is a *Binary Classification* case, so we are facing a *Supervised Learning*. This dataset had 800 observations and 19 variables (numerical and categorical data) and in terms of the dataset’s content, we gained some insight into the challenges that we could face, not only during the data treatment phase but also while applying prediction techniques.

Afterwards, the following step was to generate a statistical analysis to acquire a better insight of the dataset. We checked for duplicated observations (result: no duplicated values), checked for any missing values (result: only the variable “Education” had missing values), changed the index (it made more sense to set PatientID as the index of the data frame) and plotted the *Descriptive Statistic* [1] of the dataset, allowing us to analyse the mean, median, standard deviation, the maximums, the minimums and the unique values for categorical variables (the variable “Name” had one repeated value – Mr. Gary Miller – however, it corresponded to two different people; the minimum of the variable “Birth Year” was 1855 remitting to an incoherence; and through the quartiles in comparison with each maximum and minimum, we were able to have an important glance of the dataset’s outliers – the bigger the distance between, for example, the 3$^{rd}$ quartile and its respective maximum, the higher the probability of existing outliers). Additionally, we did a coherence check for the variables “Birth Year”, “High Cholesterol” and “Blood Pressure” and an inconsistency test for the variable “Region” (which will be treated later).

In case of the Coherence check, we verified that our dataset had birth years below 1922, meaning that, there were people with more than one hundred years; the cholesterol levels had values too high in comparison to standard levels and the same for the variable “Blood Pressure”.

Following this analysis, as a means to gain a better insight of each variable, to check the relationships between the dependent variable “Disease” and the independent variables and the outliers, we created some visualizations (attached in annexes). Firstly, in order to have an overview of the dataset, we started by plotting the *Numeric variables’ Histograms* [2] in order to analyse the general characteristics of the population. With this, we observed that, on average, the individuals in our data had been born between 1950-1975, had a hight around 175cm, a weigh between 60-80, the cholesterol level varied between 200-300, the blood pressure level assumed values between 120-140, the struggle with mental health days around 20 days and the physical nearby 0. Secondly, we depicted the data in a *Seaborn Pairplot diagram* [3], being able to take out some conclusions about which numerical variables are influenced or not by the dependent variable (Disease) by looking at its diagonal’s distributions. As a matter of fact, regarding the variable “Weight”, we observed a skewed-right distribution comparing people affected by the disease (sick people) and the ones that are not affected by it (healthy people) – meaning overweight people tend to be more likely to have the disease. Moreover, by analysing the variable “Mental Health” distribution, we noticed that people with the disease have a higher propensity to last longer with poor physical or mental health, keeping them from doing their usual activities. On the contrary, variables such as “Birth Year”, “Height”, “Cholesterol”, “Blood Pressure” can be not directly influenced by “Disease”, but we noticed that they can be correlated within each other, and their results still be significant (we will analyse them in a row).

Thirdly, due to the output of the previous graphics, we found interest in analysing and comparing the relation between the Disease and one or more independent variables. Based on the *individual’s box plot* for each relevant variable, for the variables “Weigh” [5] and “Mental Health” [7] we were able to confirm our initial hypothesis (the weight average and the day’s average of the people with the disease is much higher in comparison with healthy people). Furthermore, we found the variable “Height” relevant [6] (the height average of the people with disease is much lower in comparison with healthy people). Afterwards, we found pertinent to analyse the relation between two independent variables and the Disease. In this graphic [9], for the same range of cholesterol (between 150-500), we noticed that overweight people have more propensity in acquiring a disease. In [11], we observed that there was a correlation between the variables “High Cholesterol” and “Height” affecting the individual’s health – for ill people above 165cm, the cholesterol average is lightly lower comparing with non-sick people. In [12], we concluded that the blood pressure directly affects people that have a height below 165cm considering the variable disease (since the blood pressure average is much lower in this case compared to the one that appears for people above 165cm). Finally, in [13], the main conclusion is that overweight people tend to struggle more with mental health if they are sick, as shown by a higher weight average in the range of 15-29 days.

Fourthly, we observed the *Metric Features’ Correlation Matrix* [4] and the result revealed poor levels of correlation between our variables, with the exception of weight and height, which are the only ones that are related.

Lastly, continuing our analysis, we decided to look at the *Numeric Features’ Box Plots* [14], to have a better look at the outliers that were present in our dataset in order to be able to later fix these issues and to have a more accurate result in our conclusions. Indeed, the variables “Birth Year”, “High Cholesterol” and “Blood Pressure” are the ones that are more evident in having outliers.

## PREPROCESSING

Pre-processing is a technique (preliminary processing) that allows data analysts to convert raw data, which usually includes missing or inconsistent values, into the data prepared to model (primary processing) and into a format that can be easily interpreted. This process includes data cleaning, data transformation and data reduction. In fact, it is essential because it ensures the integrity of the results.

***Missing Values***

A missing value occurs when there is no data value stored in an observation, leading us to incorrect conclusions. In that way, once they are detected, we must correct them by either eliminating them, replacing them with a constant, mean, or mode, or applying a predictive model to them.

Hence, due to the fact that the only variable that had missing values was the categorical variable "Education", we replaced them using the Mode (the most frequent value), as a means to solve the issue and keep those observations (having as much information as possible is important).

***Outliers***

Outliers are data records that deviate significantly from the rest. Although they may be useful in some analysis, they have a considerable impact on data distributions and statistical methods, biasing our results.

Hence, as previously mentioned, when checking outliers through box plots and histograms [14], we identified some values as possible outliers in the variables: "Birth Year", "High Cholesterol" and "Blood Pressure".

In order to correct the situation and not drop the outliers (doing so would interfere with the test's dataset), we decided to apply the Inter-Quartile Range (IQR) method. Thus, we substituted all the values we considered as outliers and that were above the 75 quantile (3$^{rd}$ quartile) and under the 25 quantile (1$^{st}$ quartile) by NaN (not a number) and we used the KNN imputation to replace them.

However, we noticed that, by applying the IQR method, we could be removing valuable information (under the 1$^{st}$ quartile or above the 3$^{rd}$ quartile) in our dataset. For example, if we have a patient with a cholesterol level equal to 290, with the IQR method, we will consider it as an outlier since it is above the 3$^{rd}$ quartile (the benchmark is equal to 280) [1].

Additionally, based on the *CUF* site, having a cholesterol value equal to 290, it is indeed within the standard parameters and cannot be considered as an outlier. So, in order to avoid removing information that may be relevant, we decided to replace the outliers with NaN, but use a manual filter, based on hospital websites.

***Feature Engineering***

The process of extracting, transforming, or creating new features from the original variables to increase the predictive performance of machine learning models is known as Feature Engineering.

To simplify our code and avoid future errors, we decided to modify certain variable names: the variable "Birth Year" to "Age" (obtained by subtracting the dates of birth from the current year), the variable "Name" to "Sex" through the abbreviations "Mrs" and "Ms" (being a binary column that identifies whether the person is a "male" – if equal to 0 – or a "female" – if equal to 1) and the variable "Region" we divided into three sub-regions [22] (Region Central, Region North and Region South – because the disease can be more dominant in certain regions and enable us to transform them into dummies).

Additionally, we created a new feature that we considered important. By analysing the variables of the dataset, we noticed that it presented detailed information about the height and weight of each person. However, there was no variable that related these variables and taking into consideration that it might come in handy later on in the analysis, we created the variable "IMC".

Lastly, due to the fact that the algorithm only process numbers and that it is unable to distinguish whether it is written in text format or not, it cannot differentiate which value is better or worst. Hence, we decided to rearrange all the categorical variables: the variables "Exercise" and "Smoking Habit" were transformed into a binary type and the others in degrees and in ascending order (ordinal encoding) because they have more than one option. In case of the variables "Region Central,

Region North and Region South" we put them into dummies because we were unable to order them by an importance order (they are not ordained within each other and are equally important).

***Data Scaling***

The variables tend to differ in magnitude, units and range and many machine learning algorithms perform better when all input variables are scaled to fit within a standard range bearing in mind that variables with higher magnitudes will weight more than others in data analysis.

Hence, in order to fix this, we used the *MinMax Normalization* technique. This method scales each input variable separately to the range 0 – 1 (range for floating values) [15], constraining our data attribute to a particular container to develop a correlation among different data points. However, this technique only accepts numerical values, so to proceed with the scaling without any additional problems, previously we had to transform the categorical variables into dummy variables (the regions variables).

It should also be noted that, although we chose the Min-Max scaler as our scaling algorithm, we tried to use the robust scaler, since this is the algorithm that is used when we have the presence of outliers. This scaler removes the median and scales the data according to the quantile range (default: IQR). The IQR is the range between the 1$^{st}$ quartile (25$^{th}$ quantile) and the 3$^{rd}$ quartile (75$^{th}$ quantile). However, after the application, we were not successful, meaning we did not have an increase in the performance of our model. We believed that this happened because it might be related to the fact that we were disregarding information that may be relevant, as we mentioned previously.

***Feature Selection***

While building a machine learning model, we come across a lot of features in the dataset, and not all of these features are essential on every occasion. Indeed, adding unnecessary features during the training phase reduces the overall accuracy of the model, increases the model's complexity, reduces the model's generalization capabilities, and biases the model.

In that way, with Feature Selection we aim to find and select the most significant and relevant variables that shows to be important for our analysis and to build our machine learning model, removing the ones that do not. So, Feature Selection helps in finding the set of features that results in training a machine learning algorithm faster, building a model with better prediction power, and reducing over-fitting.

It is important to notice that we decided to do the split after the Feature Selection enabling us to do the analyse of the untouched data frame (without division) and to lead us to more reliable results.

Firstly, to measure the degree of dependence between numerical variables, we used *Spearman Correlation* [16] and plotted it in a Heat Map. Based on our dataset, we noticed that there are a pair of variables that presented a fairly high correlation (larger than 0,7), consequently suggesting dropping the variable "Weight" (it is highly correlated with the variables "IMC" and "Height", promoting redundant information).

Secondly, for the numerical input and categorical output, we applied the *ANOVA* test, which allows us to compare the means of more than two groups, to complement our overview of which features have higher importance. Having this in mind, the output was to discard three variables out of eight, maintaining the variables "Age", "Weight", "Mental Health", "Physical Health" and "IMC". Furthermore, we computed the *Kendall non-parametric correlation* [19] that assesses statistical associations based on the ranks of the data, which suggested to keep all the variables (we were unable to extract valuable information). Thirdly, we computed the *Decision Tree algorithm* [18] removing the variables "Weight" and "Height", and the *Gini/Entropy* [17], concluding that the variables "Weight", "Height" and "IMC" are not relevant for the analysis.

Additionally, we computed the *Recursive Feature Elimination* (RFE) enabling to select features in a training dataset that are more relevant in predicting the target variable (removes the weakest features – eliminated all the variables except "Mental Health" and "Physical Health") and the *Ridge Regression* [20] that by adding a regularized term that forces the learning algorithm to fit the data, allows to reduce the variance of the estimates (recommending to drop the variables "Age" and "Blood Pressure") . Last, for the categorical input to categorical output, we processed the *Chi Square* that identifies the best features for a given dataset by determining the features on which the output class label is most dependent on: if the chi-squared value is high (result: keep the variables "Sex", "Drinking Habit", "Fruit Habit", "Exercise", "Check-up" and "Diabetes") and the *Mutual Statics Information*, keeping the same variables as the chi-squared method plus "North Region", "Education" and "Water Habit".

Finally, we applied the *Random Forest algorithm* [35], which is useful for both numerical and categorical variables, which allows us to eliminate irrelevant variables and improve the accuracy as well as the performance of classification. In fact, each tree of this algorithm calculates the importance of a feature according to its ability to increase the pureness of the leaves, suggesting in keeping the variables "Age", "High Cholesterol", "Blood Pressure", "Mental Health", "Physical Health", "IMC", "Fruit Habit", "Checkup" and "Diabetes".

In that way, after we had done all the feature selection models, we decided that we would keep all the variables that the output result was "keep" in all the models or had only one "discard". And for the variables that had two "discards", we decided to make combinations between them (try with and without those variables), until we have achieved the feature selection that gave us the best output.

As a result, we concluded that some of the features improved the model and others made it worse. Therefore, we decided to use the variables "Sex", "Age", "Exercise", "Fruit Habit", "Mental Health", "Physical Health", "Checkup", "Diabetes", "Drinking Habit", "Blood Pressure" and "High Cholesterol" that were chosen as selection criterion [21].

## MODELLING

Before we started applying the models to our training dataset, we decided to make a small adjustment by creating the function *"processing_data".* This function allowed us to incorporate all the changes previously made to our data into one function because we thought that would be pertinent and it would make our job easier on the test dataset.

The process of creating machine learning algorithms is divided into two parts: training and testing, so in order to test our models, we proceeded to partition the dataset of 70/30 (it is a small dataset) so as to be able to do the Training phase in one and the Validation phase on the other. In that way, we were able to use the train set (equal to 0.7) to build the model and the validation data (equal to 0.3) to check how good our developed model was. To do so, we applied the *Train Test Split* and *the K-Fold*. This method allowed us to split the dataset into K (in this case K=10) number of folds and each time this model runs, one fold is considered for validation and the remaining are used for training (split the dataset in 10 consecutive folds where 9 is for the training set and 1 for the validation set), being repeated successively through iterations (during the full iterations, each fold will be used once as a test). By applying this method, we are able to understand how the data is spread in a consistent way, improving the score accuracy and reducing overfitting.

It is also important to mention that, as requested, we will use the **F1-score** as the metric to analyse our results. This metric can be used in classification models, which is the case, and it is considered one of the most important evaluation metrics in machine learning. The F1-score combines the precision and recall of a classifier into a single metric.

Hence, our modelling approach began with models learnt in class and also with self-studied methods like the XG-Boost (we decided to test a model outside the sklearn library simply to visualize the results of a different model), so considering this, we did not include it in our selection even though it had a good score.

Firstly, we decided to compute the *Logistic Regression* [23], an algorithm that predicts the probability of an event's occurrence based on a given dataset of independent variables by fitting that data into a logistic curve/function. Then, we decide to apply the *Gaussian Nb* [24], (an algorithm based on applying Bayes' theorem with the 'naïve' assumption of conditional independence between every pair of features given the value of the class variable), the *K-Nearest Neighbour* [25] (KNN – is a non-parametric algorithm, meaning that does not make any assumption on underlying data, which stores the dataset and at the time of classification, it performs an action on the dataset), the *Neural Networks* [26] (a system that learns how to make forecasts by taking the input data, making a prediction, comparing the prediction to the desired output and adjusting its internal state to foresee correctly next time). As it was also discussed in class, we use the *Decision Tree* [27] (able to calculate the potential success of different series of decisions made to achieve a goal), the *Gradient Boosting* [28] (the algorithm builds an addictive model in forward stage-wise fashion – allows for the optimization of arbitrary differentiable loss functions), the *Support Vector Machines* [29] (the algorithm draw lines to separate the groups according to patterns – creates the best line that can segregate n-dimensional space into classes and choose the best vectors that helps in creating that line), the *AdaBoost* [30] (combines multiple "weak classifiers" into a single "strong classifier"), and finally, the *Random Forest* [31] (which combines the output of multiple decision trees to reach a single result).

Regarding the *XG-Boost model* [31], despite the fact that we were not able to use it as a final solution, we thought that it could be a good result, since it is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.

## ASSESSMENT

Finally, we performed a rigorous analysis against the results extracted from each model in order to select a model or a set of models that guarantee us the highest results and performance.

Initially we started by changing the parameters of the models manually. However, due to the huge number of parameters and consequently the huge number of iterations between them, we decided to apply a function in some models that combined the various parameters of the respective model (seen in the *sklearn library*), giving us the best results (the best F1-scores and the best parameters).

First, we started by exposing the first model analysed in our notebook, which is the **Logistic Regression** [23]. In order to be able to understand which were the parameters that we could investigate, we consulted the *sklearn* documentation and after this research, we selected the penalty and solver parameters in our test, with the aim to see which of them enhance the model. As a matter of fact, looking at the results we verified that the score is not the most desired one (0.845 for train and 0.834 for validation) but presented a little overfitting of 0.011, with the I1 and *saga* as the best parameter.

Next, we evaluated **Gaussian NB** [24] with the var smoothing parameter from the *sklearn* documentation. With this parameter at 0.0001 we were able to maximize our results, with a score for train of 0.845 and for validation of 0.824. The conclusion of this model was the same as the Logistic model: the score was not the best, however overfitting was short.

To continue our analysis, we tested the **KNN** [25] model, where we used the parameters to find the optimal number of neighbours, weight, and the algorithm (in this case it was, 49, *distance* and *brute*, respectively). In view of the results, we realized that the higher the score is, the higher the overfitting is as well, so these results translated into an F1-score for train

of 1 and 0.971 for validation, adding up to an overfitting of 0.029. It is also important to mention that this model is the one that presents the best performance in the test and validation.

We also included the model of **Neural Networks** [26] (MLP Classifier) in our analysis with the parameters confirmed on the *sklearn* documentation. In fact, we decided to use the *hidden layers* sizes (that represent the number of neurons in the hidden layer), the *solver* for weight and the *activation* optimization. Hence, by analysing the graph, we noticed that there were points that had no overfitting and even underfitting, but in those moments the train and validation performance were bad. According to the output, we realized that the best scenario was with the hidden layers at 100, activation *logistic* and solver *lbfgs*, presenting an F1-score for the train of 1 and 0.964 for the validation.

Next, we assessed the **Decision Trees** [27] model, with the parameters max leaf nodes, random state and criterion (gini and entropy). By doing this, we were able to see that the best criterion was the *gini*, with a maximum *number of nodes* of 71, together with a *random state* of 82. This reflected on a F1-score for the train of 1 and 0.956 for validation. Additionally, processing this model we got 19208 possible observations of the various parameters.

Apart from all these models, we also decided to evaluate the **Random Forest** [32], a robust and widely used model for this type of problem. Again, we went to the *sklearn* documentation to find the parameters that this model is built in and decided to work on the number of trees in the forest (n estimators) and the criteria (gini and entropy). In that way, by analysing the results, we realized that the best parameters were the combination of *100* of the *estimators* and the *entropy* as the criterion, achieving an F1-score for train of 1 and 0.963 for validation. With these results, we realized that it was the model that guaranteed us a good performance given that the score presented is high, together with an unconcerned overfitting of 0.037.

We also evaluated the **Gradient Boosting** [28] with the following parameters: the number of boosting stages to perform (n_estimators); criteria (friedman_mse, squared_error); learning rate and loss (deviation, exponential). Thus, when we looked at the results, we saw that this model presented particularly good results with 1 for train and 0.966 for validation, with 75 of estimators, the *friedman_mse* as the criterion, 0.5 of learning rate and *deviance* as loss.

Regarding the **Support Vector Machines** [29], we decided to use the kernel parameter (linear, poly, rbf and sigmoid). This decision was made, because it was a simple model with few parameters, but it was the model that gave us one of our worst scores, deciding that the best decision was to discard this option. Despite this, the best criteria was obtained by defining the kernel as *linear*, achieving a score of 0.831 for validation and 0.847 for train.

The last scikit learn model used was **Adaboost** [30] and we decided to use the parameters estimators and the learning rate, as the main changing factors. The values for the parameters were decided manually, after successive observations of the graph. With *estimators* equal to 500 and *learning rate* equal to 1, we attained a F1-score of 1 for the train, and 0.955 for the validation.

To finish, we used the **XGBoost** [31] model. With *estimators* equal to 100, we obtained a F1-score of 1 in train and 0.96 in validation. In general, despite the high values for training obtained with this model, when we submitted it to Kaggle, the score was never higher than 0.85. However, as mentioned earlier, we will not consider this model.

Hence, in order to be able to find out which model was the best to predict, we made a table with the scores [33] of the different models

We tried to change several things in the jupyter notebook, from the way we did the outliers, using different features, different ways of scaling the data, but so far, we had **not** managed to exceed the Kaggle score of **0.98924**.

In a last attempt to get the maximum score in Kaggle, we decided to use the **Stacking Classifier**. A Stacking Classifier is an ensemble learning method that combines multiple classification models to create one "super" model. This can often lead to improved performance since the combined model can learn from the strengths of each individual model.

Initially, we decided to test the Stacking Classifier with the combinations of our three best models (KNN, Neural Networks and Gradient Boosting), and with its best parameters, respectively, but the score did not increase. However, we continued testing, but this time with different models and with different parameters.

Through the combination of Random Forest (estimators = 100 and entropy as criterion), KNN (neighbors = 100 and weights = distance) and Logistic Regression, we were able to obtain, in the Stacking Classifier, a score of 1 in training and 0.967 in validation.

After building the different models, and in a way of optimization the models, we tested all the models with the optimal parameter, and putted them into a table [34], finding all the validation scores of each model.

Using the previous table, we saw that the tree best models are the **Neural Networks**, **Stacking Classifier** and the **Gradient Boosting**, with a score of 0.9774, 0.9729 and 0.9689, respectively. Hence, we decided to test the tree models in Kaggle, to observe the score. Subsequently, we tested the tree best models with the best validation's parameters.

Once the best model, the one that guarantees the best results, was selected, we moved on to the last step of the project: apply the results in the **test** dataset. Initially we started by doing a brief similar analysis to the train's dataset where we verified that it is composed by 225 rows. Next, we applied the processing data function, created earlier, as well as the chosen features. It is important to notice that we did not analyse the duplicated variables nor change the data fulfilment (we do not analyse the outliers, incoherencies, etc) to guarantee the same number of rows.

Using Gradient Boosting and Neural Networks on the test dataset, the Kaggle score ranges between 0.89 and 0.91, respectively. With **Stacking Classifier,** we were able to get the **maximum score**, 1. In that way, given these results, we believed that the difference regarding the Gradient Boosting and the Neural Networks might be related to the existence of overfitting in both models.

## CONCLUSION

In our study, we began by analysing the dataframe to gain a broad understanding of the impact that the provided variables of the dataset might have had on the target variable ("Disease"). We started by exploring (Data Exploration) and to treating (Pre-processing) our data set and we found quite challenging to know which variables were good information and what weren't.

In that way, after meticulously trimming our dataset into ideal features ("Sex", "Age", "Exercise", "Fruit Habit", "Mental Health", "Physical Health", "Checkup", "Diabetes", "Drinking Habit" and "Blood Pressure"), we proceeded to model and adjust the hyperparameters using several partition methods (Train Test Split and the K-Fold). Hence, based on the Stacking Classifier and the combination of tree models (Random Forest, KNN and Logistic Regression), we obtained on Kaggle a score of 1. Towards their results we are able to answer the question "Who are the people more likely to suffer from the Smith Parasite" and we can state that 389 will have the disease and 411 will not.

This work was indispensable to understand the materials taught throughout the classes. It has helped us to have a better insight about the importance and versatility of Machine Learning in the labour world and its contribution to society.

# REFERENCES


"Sklearn.ensemble.GradientBoostingClassifier"scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html. Accessed 29 Nov. 2022.

"Sklearn.neural_network.MLPClassifier"scikit.,Available at:Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html Accessed 29 Nov. 2022.

Pressão Arterial: Perceba Se Tem os Valores Elevados (no date) CUF. Available at: https://www.cuf.pt/mais saude/pressao-arterial-elevada (Accessed: December 23, 2022)

Colesterol CUF. Available at: https://www.cuf.pt/saude-a-z/colesterol (Accessed: December 23, 2022).

"Chi-Square Test for Feature Selection - Mathematical Explanation." GeeksforGeeks, 19 July 2019, Available at: www.geeksforgeeks.org/chi-square-test-for-feature-selection-mathematical-explanation/. Accessed 23 Dec. 2022.

"Gini Impurity and Entropy in Decision Tree - ML." GeeksforGeeks, 14 July 2020, Available at: www.geeksforgeeks.org/gini-impurity-and-entropy-in-decision-tree-ml/. Accessed 23 Dec. 2022.

"How to Do Train Test Split Using Sklearn in Python." GeeksforGeeks, 25 June 2022, Available at: www.geeksforgeeks.org/how-to-do-train-test-split-using-sklearn-in-python/. Accessed 23 Dec. 2022.

"ML | Ridge Regressor Using Sklearn." GeeksforGeeks, 6 Sept. 2019, Available at: www.geeksforgeeks.org/ml-ridge-regressor-using-sklearn/. Accessed 23 Dec. 2022.

"Neural Networks | a Beginners Guide." GeeksforGeeks, 17 Jan. 2019, Available at: www.geeksforgeeks.org/neural-networks-a-beginners-guide/. Accessed 23 Dec. 2022.

"Python | Kendall Rank Correlation Coefficient." GeeksforGeeks, 15 May 2020, www.geeksforgeeks.org/python-kendall-rank-correlation-coefficient/. Accessed 23 Dec. 2022.

"Spearman's Rank Correlation." GeeksforGeeks, 15 Aug. 2020, www.geeksforgeeks.org/spearmans-rank-correlation/. Accessed 23 Dec. 2022.

"XGBoost." GeeksforGeeks, 18 Sept. 2021, Available at: www.geeksforgeeks.org/xgboost/. Accessed 23 Dec. 2022.

JavaTpoint. "K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint." Available at: Www.javatpoint.com, www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. Accessed 23 Dec. 2022.

Bedre, Renesh. "ANOVA Using Python (with Examples)." Data Science Blog, 22 Oct. 2018, Available at: www.reneshbedre.com/blog/anova.html. Accessed 23 Dec. 2022.

"Python Machine Learning - Logistic Regression." Available at: www.w3schools.com, www.w3schools.com/python/python_ml_logistic_regression.asp. Accessed 23 Dec. 2022.

"Sklearn.linear_model.LogisticRegression" (scikit. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 29 Nov. 2022.

"Sklearn.naive_bayes.GaussianNB" scikit. Available at: https://scikit learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. Accessed 29 Nov. 2022.

"Sklearn.tree.DecisionTreeClassifier" scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. Accessed 29 Nov. 2022.

"Sklearn.ensemble.RandomForestClassifier scikit. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed 29 Nov. 2022.

"Sklearn.ensemble.AdaBoostClassifier" scikit. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html. Accessed 29 Nov. 2022.

"Sklearn.neighbors.KNeighborsClassifier" scikit. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed 29 Nov. 2022.

"Sklearn.ensemble.StackingClassifier" scikit. Available at: https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html. Accessed 29 Nov. 2022.

"Sklearn.svm.SVC" scikit. Available at: https://scikit learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed 29 Nov. 2022.

# ANNEXES

| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Name** | 800 | 799 | Mr. Gary Miller | 2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Birth_Year** | 800.0 | NaN | NaN | NaN | 1966.04375 | 15.421872 | 1855.0 | 1961.0 | 1966.0 | 1974.0 | 1993.0 |
| **Region** | 800 | 10 | East Midlands | 154 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Education** | 787 | 6 | University Complete (3 or more years) | 239 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Disease** | 800.0 | NaN | NaN | NaN | 0.51375 | 0.500124 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| **Smoking_Habit** | 800 | 2 | No | 673 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Drinking_Habit** | 800 | 3 | I usually consume alcohol every day | 406 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Exercise** | 800 | 2 | No | 536 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Fruit_Habit** | 800 | 5 | Less than 1. I do not consume fruits every day. | 452 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Water_Habit** | 800 | 3 | Between one liter and two liters | 364 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Height** | 800.0 | NaN | NaN | NaN | 167.80625 | 7.976888 | 151.0 | 162.0 | 167.0 | 173.0 | 180.0 |
| **Weight** | 800.0 | NaN | NaN | NaN | 67.8275 | 12.11347 | 40.0 | 58.0 | 68.0 | 77.0 | 97.0 |
| **High_Cholesterol** | 800.0 | NaN | NaN | NaN | 249.3225 | 51.566631 | 130.0 | 213.75 | 244.0 | 280.0 | 568.0 |
| **Blood_Pressure** | 800.0 | NaN | NaN | NaN | 131.05375 | 17.052693 | 94.0 | 120.0 | 130.0 | 140.0 | 200.0 |
| **Mental_Health** | 800.0 | NaN | NaN | NaN | 17.345 | 5.385139 | 0.0 | 13.0 | 18.0 | 21.0 | 29.0 |
| **Physical_Health** | 800.0 | NaN | NaN | NaN | 4.55875 | 5.449189 | 0.0 | 0.0 | 3.0 | 7.0 | 30.0 |
| **Checkup** | 800 | 4 | More than 3 years | 429 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Diabetes** | 800 | 4 | Neither I nor my immediate family have diabetes. | 392 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

FIGURE 1 – DESCRIPTIVE STATISTICS

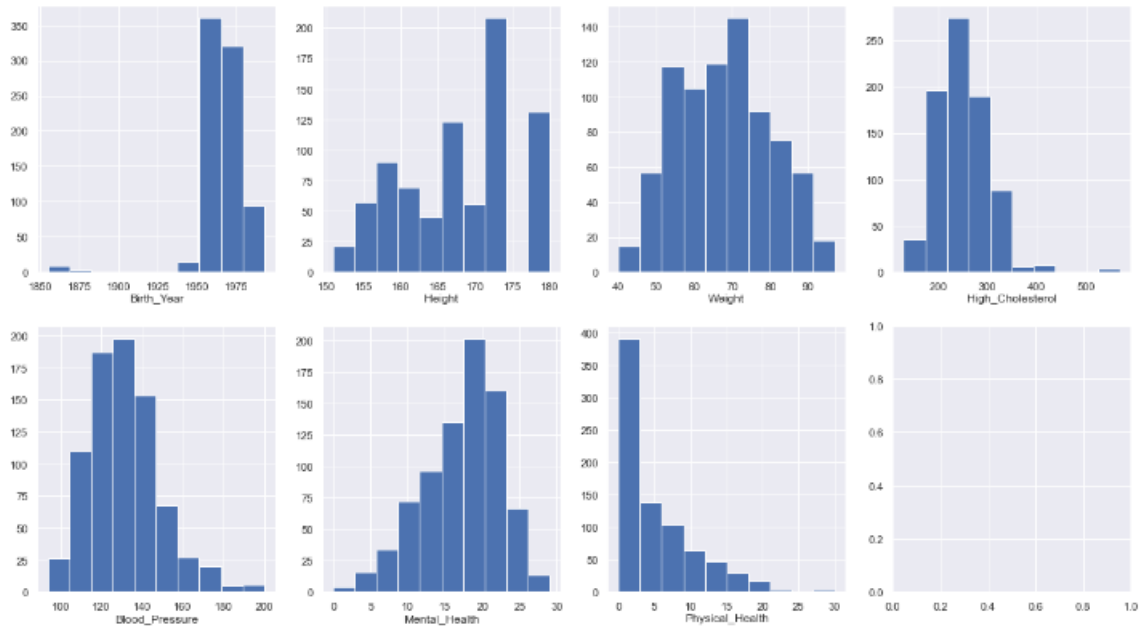



FIGURE 2 – NUMERIC VARIABLES' HISTOGRAMS

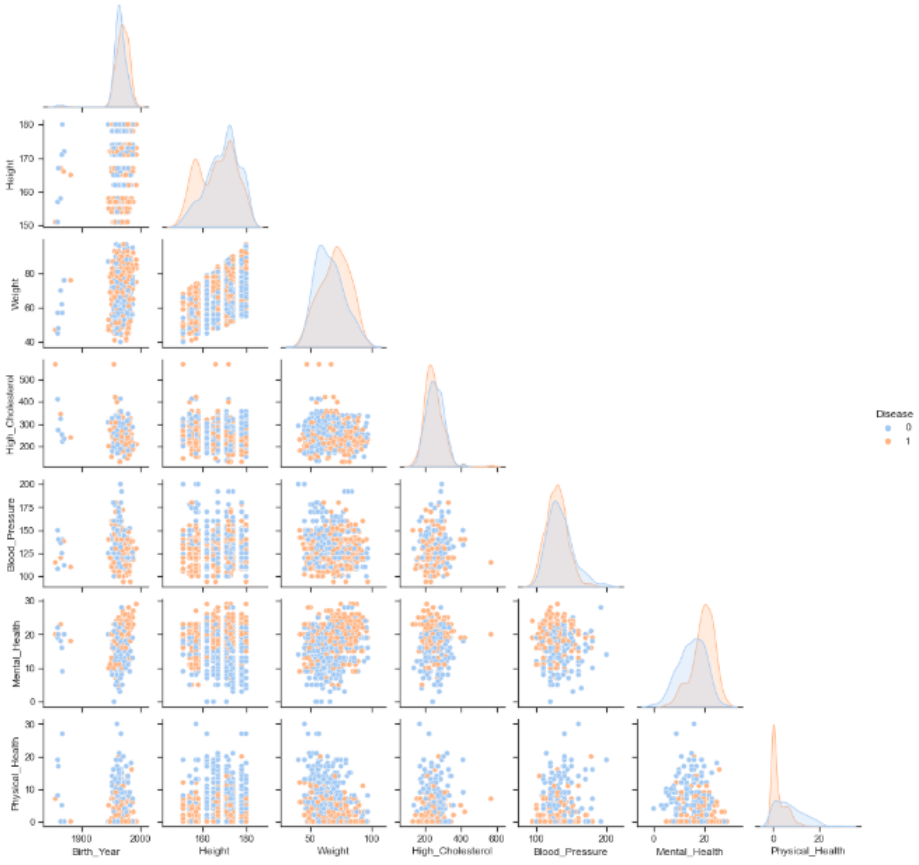



FIGURE 3 – PAIRPLOT

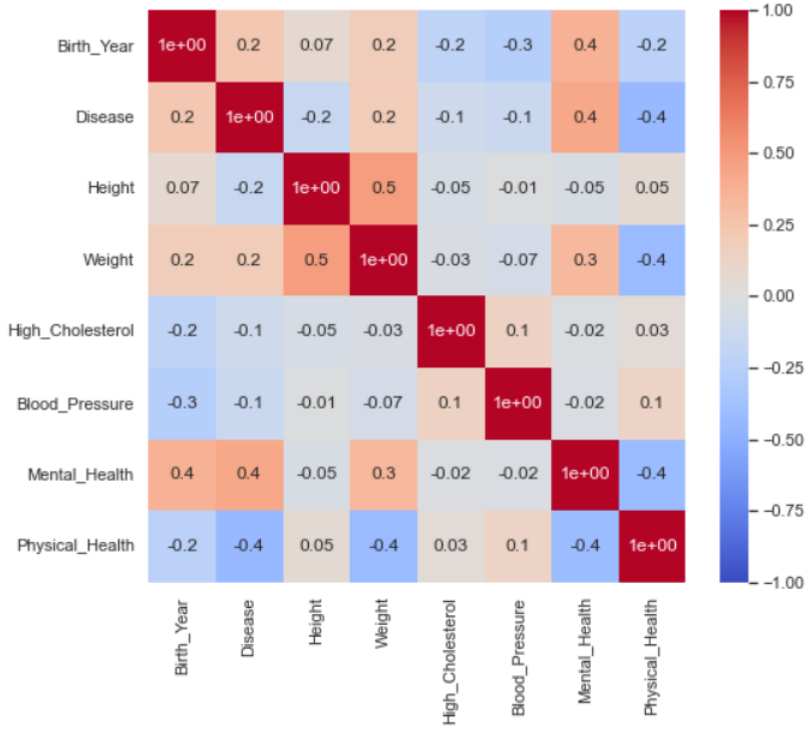



FIGURE 4 – CORRELATION MATRIX

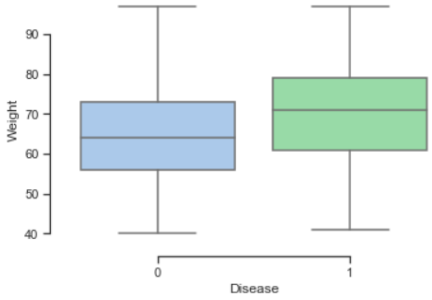



FIGURE 5 – WEIGHT BOXPLOT

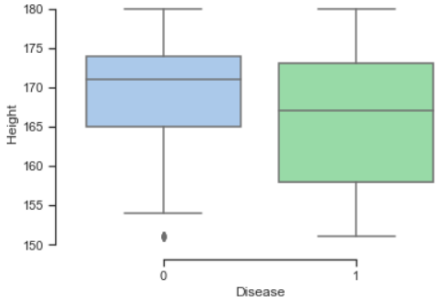



FIGURE 6 – HEIGHT BOXPLOT

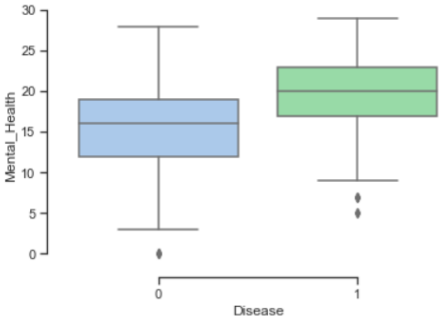



FIGURE 7 - MENTAL_HEALTH BOXPLOT

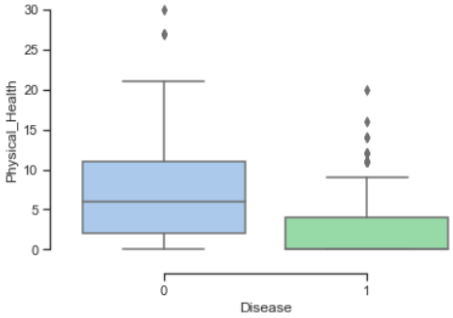



FIGURE 8 – PHYSICAL_HEALTH BOXPLOT

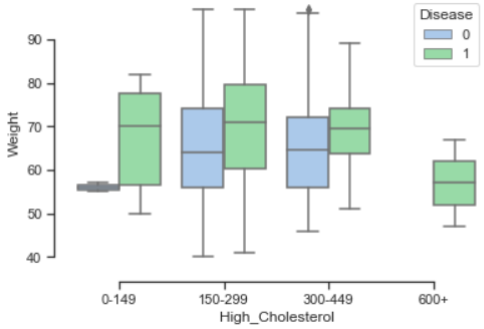



FIGURE 9 - THREE VARIABLE BOXPLOT

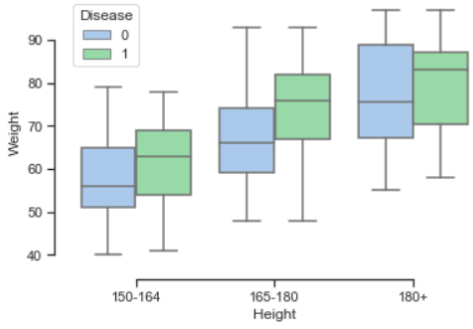



FIGURE 10 - THREE VARIABLE BOXPLOT

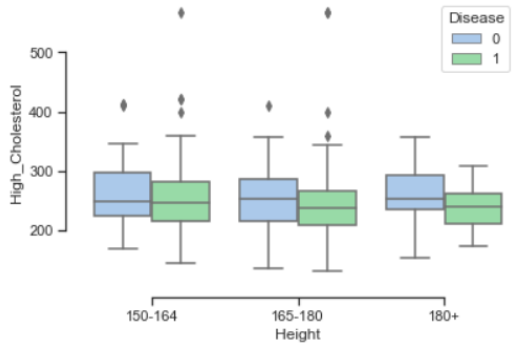



FIGURE 11 - THREE VARIABLE BOXPLOT

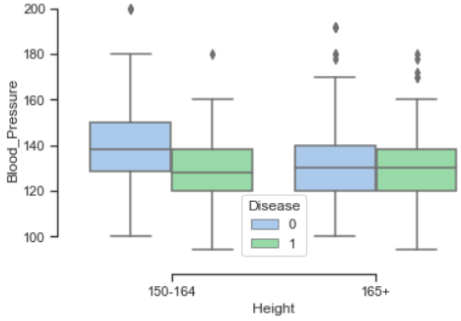



FIGURE 12 - THREE VARIABLE BOXPLOT

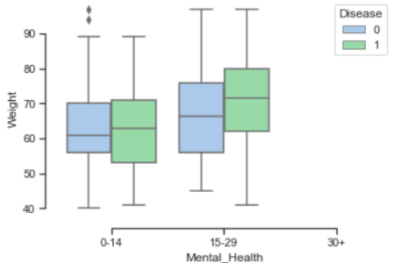



FIGURE 13 - THREE VARIABLE BOXPLOTS

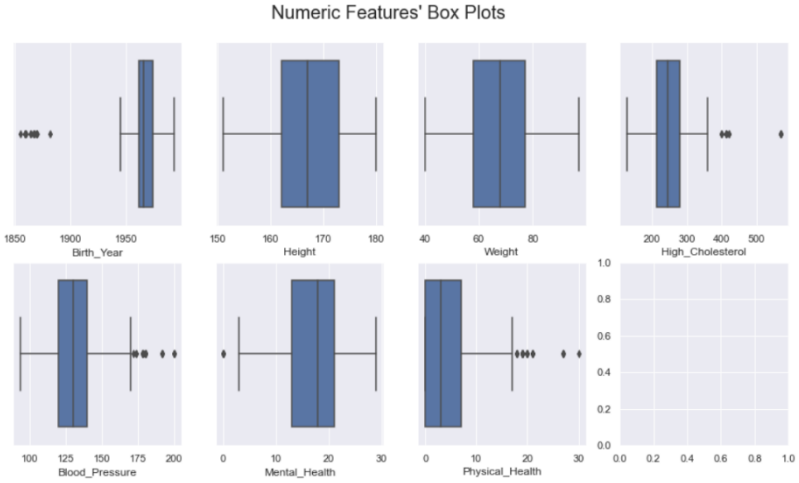



FIGURE 14 – NUMERIC FEATURES' BOXPLOT

| PatientID | Sex | Education | Smoking_Habit | Water_Habit | Drinking_Habit | Fruit_Habit | Exercise | Checkup | Region_Central | Region_North | Region_South | Diabetes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1167** | 1.0 | 0.4 | 0.0 | 1.0 | 0.0 | 0.00 | 1.0 | 0.333333 | 0.0 | 0.0 | 1.0 | 1.000000 |
| **1805** | 0.0 | 0.4 | 0.0 | 1.0 | 0.5 | 0.00 | 1.0 | 0.000000 | 0.0 | 0.0 | 1.0 | 1.000000 |
| **1557** | 0.0 | 0.2 | 0.0 | 0.5 | 0.5 | 0.00 | 0.0 | 0.333333 | 0.0 | 1.0 | 0.0 | 1.000000 |
| **1658** | 0.0 | 1.0 | 0.0 | 0.5 | 0.0 | 0.00 | 1.0 | 0.000000 | 0.0 | 0.0 | 1.0 | 0.333333 |
| **1544** | 0.0 | 0.8 | 0.0 | 0.5 | 0.5 | 0.25 | 0.0 | 0.333333 | 0.0 | 0.0 | 1.0 | 0.333333 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1909** | 0.0 | 0.4 | 0.0 | 1.0 | 0.5 | 0.00 | 1.0 | 0.000000 | 1.0 | 0.0 | 0.0 | 1.000000 |
| **1386** | 1.0 | 0.2 | 0.0 | 1.0 | 0.0 | 0.00 | 0.0 | 0.333333 | 0.0 | 1.0 | 0.0 | 0.333333 |
| **1088** | 1.0 | 0.2 | 0.0 | 0.5 | 0.5 | 0.50 | 0.0 | 0.333333 | 1.0 | 0.0 | 0.0 | 1.000000 |
| **1662** | 0.0 | 0.2 | 0.0 | 0.5 | 0.0 | 0.00 | 0.0 | 0.333333 | 1.0 | 0.0 | 0.0 | 1.000000 |
| **1117** | 0.0 | 0.2 | 0.0 | 0.0 | 0.5 | 0.00 | 1.0 | 0.000000 | 0.0 | 0.0 | 1.0 | 1.000000 |

| PatientID | Age | Height | Weight | High_Cholesterol | Blood_Pressure | Mental_Health | Physical_Health | IMC |
|---|---|---|---|---|---|---|---|---|
| **1167** | 0.583333 | 0.137931 | 0.473684 | 0.783505 | 0.342105 | 0.724138 | 0.066667 | 0.830289 |
| **1805** | 0.500000 | 0.758621 | 0.842105 | 0.343643 | 0.631579 | 0.310345 | 0.000000 | 0.945205 |
| **1557** | 0.395833 | 0.379310 | 0.491228 | 0.329897 | 0.368421 | 0.896552 | 0.000000 | 0.679604 |
| **1658** | 0.729167 | 1.000000 | 0.456140 | 0.628866 | 0.407895 | 0.448276 | 0.266667 | 0.257991 |
| **1544** | 0.520833 | 1.000000 | 0.315789 | 0.505155 | 0.407895 | 0.620690 | 0.066667 | 0.070015 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **1909** | 0.437500 | 0.931034 | 0.368421 | 0.254296 | 0.657895 | 0.413793 | 0.133333 | 0.172755 |
| **1386** | 0.270833 | 0.206897 | 0.368421 | 0.285223 | 0.342105 | 0.793103 | 0.000000 | 0.591324 |
| **1088** | 0.666667 | 0.551724 | 0.140351 | 0.487973 | 0.605263 | 0.689655 | 0.566667 | 0.017504 |
| **1662** | 0.375000 | 0.482759 | 0.614035 | 0.268041 | 0.236842 | 0.551724 | 0.000000 | 0.804414 |
| **1117** | 0.291667 | 0.758621 | 0.526316 | 0.175258 | 0.342105 | 0.379310 | 0.400000 | 0.487823 |

FIGURE 15 – DATA SCALED


FIGURE 16 – SPEARMAN METHOD

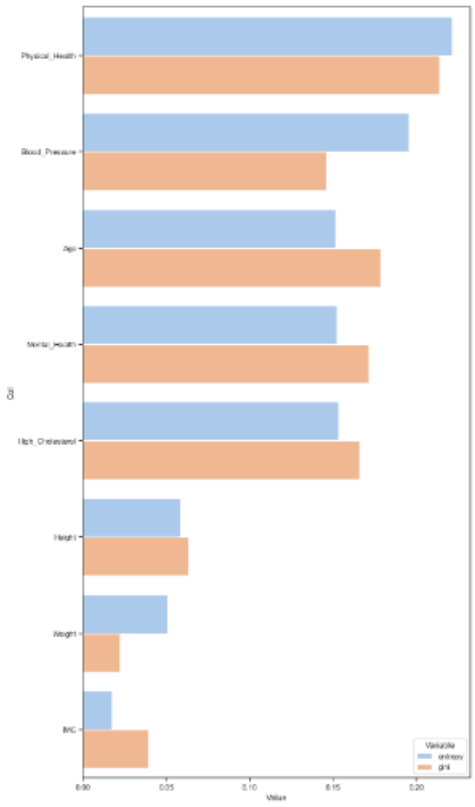



FIGURE 17 – GINI METHOD

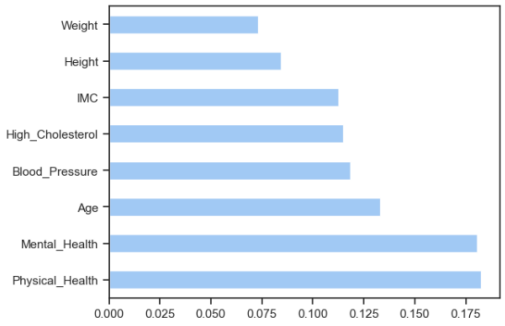



FIGURE 18 – DECISION TREE CLASSIFIER

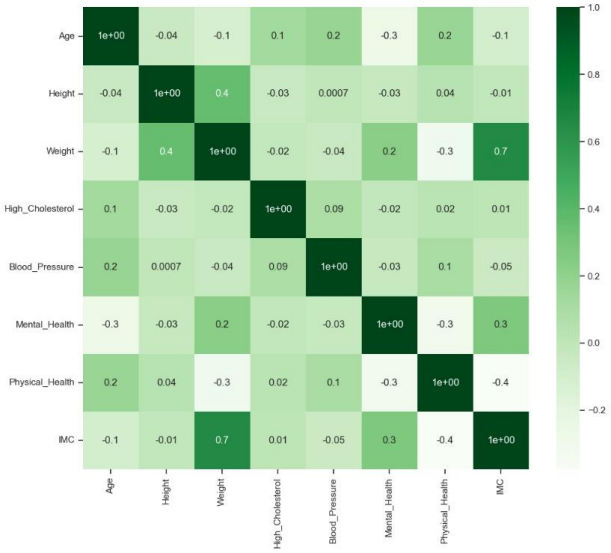



FIGURE 19 – KENDALL METHODS

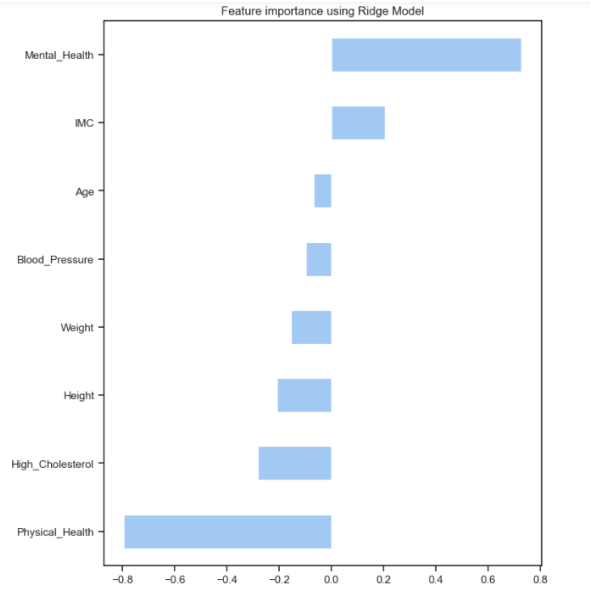



FIGURE 20 – RIDGE REGRESSION

### Numerical Variables

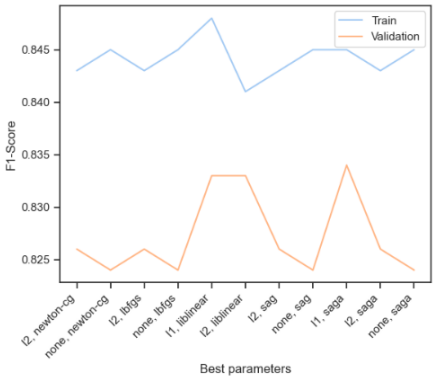
| Predictor | Spearman | ANOVA | Gini/Entropy | Decision Tree | RFE | Kendall | Ridge | Random Forest | What to do? (One possible way to "solve") |
|---|---|---|---|---|---|---|---|---|---|
| Age | Keep | Keep | Keep | Keep | Discard | Keep | Discard | Keep | Try |
| Height | Keep | Discard | Discard | Discard | Discard | Keep | Keep | Discard | Discard |
| Weight | Discard | Keep | Discard | Discard | Discard | Keep | Keep | Discard | Discard |
| High_Cholesterol | Keep | Discard | Keep | Keep | Discard | Keep | Keep | Keep | Try |
| Blood_Pressure | Keep | Discard | Keep | Keep | Discard | Keep | Discard | Keep | Try |
| Mental_Health | Keep | Keep | Keep | Keep | Keep | Keep | Keep | Keep | Keep |
| Physical_Health | Keep | Keep | Keep | Keep | Keep | Keep | Keep | Keep | Keep |
| IMC | Keep | Keep | Discard | Keep | Discard | Keep | Keep | Keep | Try |

### Categorical Data

| Predictor | Chi-Square | Mutual Statistics | Random Forest | What to do? (One possible way to "solve") |
|---|---|---|---|---|
| Sex | Keep | Keep | Discard | Keep |
| Drinking_Habit | Keep | Keep | Discard | Keep |
| Exercise | Keep | Keep | Keep | Keep |
| Fruit_Habit | Keep | Keep | Keep | Keep |
| Checkup | Keep | Keep | Keep | Keep |
| Diabetes | Keep | Keep | Keep | Keep |
| Central Region | Discard | Discard | Discard | Discard |
| North Region | Discard | Keep | Discard | Discard |
| South Region | Discard | Discard | Discard | Discard |
| Education | Discard | Discard | Discard | Discard |
| Smoking_Habit | Discard | Keep | Discard | Discard |
| Water_Habit | Discard | Keep | Discard | Discard |

FIGURE 21 – FEATURE SELECTION SUMMARY


FIGURE 22 – REGION DIVISION




```
Penalty:  l2 Solver:  newton-cg
Penalty:  none Solver:  newton-cg
Penalty:  l2 Solver:  lbfgs
Penalty:  none Solver:  lbfgs
Penalty:  l1 Solver:  liblinear
Penalty:  l2 Solver:  liblinear
Penalty:  l2 Solver:  sag
Penalty:  none Solver:  sag
Penalty:  l1 Solver:  saga
Penalty:  l2 Solver:  saga
Penalty:  none Solver:  saga
```

FIGURE 23 – LOGISTIC REGRESSION

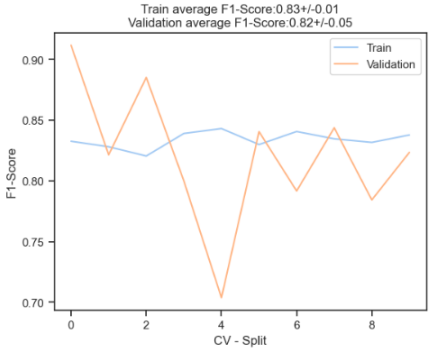
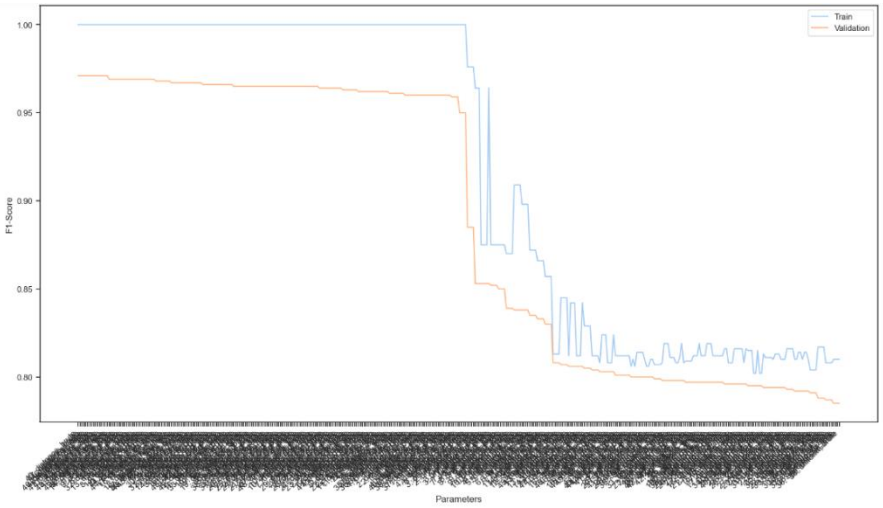
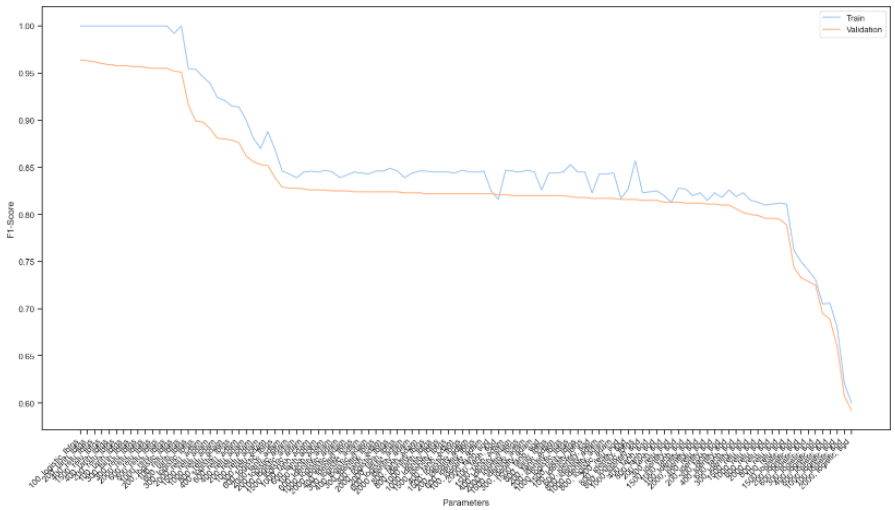



FIGURE 24 – GAUSSIAN NB



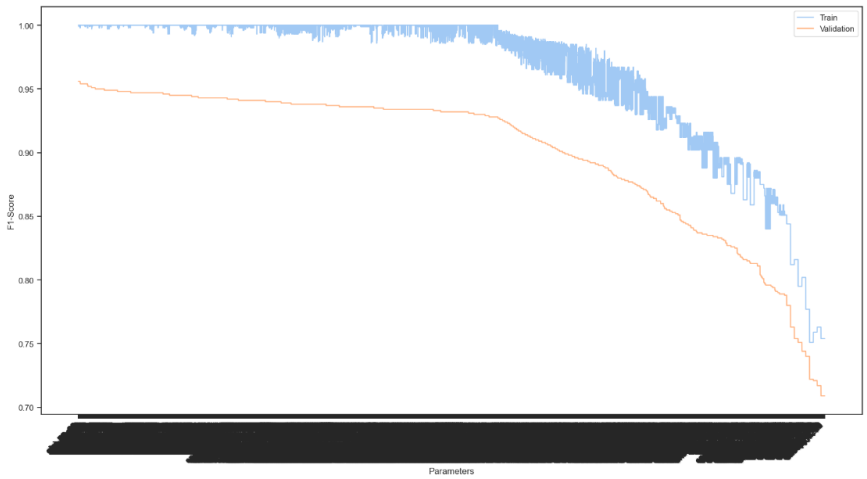

| | Parameters | Train | Validation |
|---|---|---|---|
| **391** | 49, distance, brute | 1.000 | 0.971 |
| **375** | 47, distance, brute | 1.000 | 0.971 |
| **364** | 46, distance, auto | 1.000 | 0.971 |
| **365** | 46, distance, ball_tree | 1.000 | 0.971 |
| **366** | 46, distance, kd_tree | 1.000 | 0.971 |
| **...** | ... | ... | ... |
| **288** | 37, uniform, auto | 0.808 | 0.787 |
| **282** | 36, uniform, kd_tree | 0.810 | 0.785 |
| **281** | 36, uniform, ball_tree | 0.810 | 0.785 |
| **280** | 36, uniform, auto | 0.810 | 0.785 |
| **283** | 36, uniform, brute | 0.810 | 0.785 |

FIGURE 25 – KNN



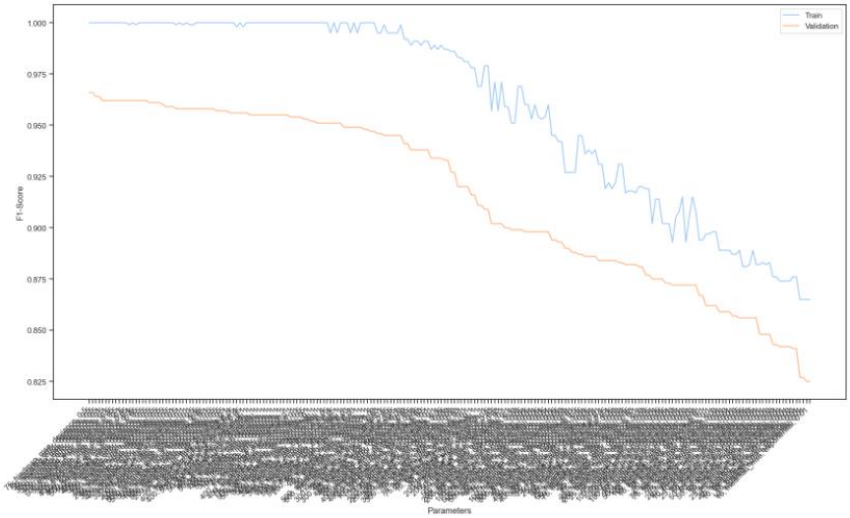

| | Parameters | Train | Validation |
|---|---|---|---|
| **3** | 100, logistic, lbfgs | 1.000 | 0.964 |
| **18** | 200, tanh, lbfgs | 1.000 | 0.963 |
| **45** | 400, relu, lbfgs | 1.000 | 0.962 |
| **93** | 1500, relu, lbfgs | 1.000 | 0.960 |
| **42** | 400, tanh, lbfgs | 1.000 | 0.959 |
| **...** | ... | ... | ... |
| **40** | 400, logistic, sgd | 0.705 | 0.695 |
| **64** | 800, logistic, sgd | 0.706 | 0.689 |
| **76** | 1000, logistic, sgd | 0.681 | 0.660 |
| **4** | 100, logistic, sgd | 0.621 | 0.608 |
| **100** | 2000, logistic, sgd | 0.600 | 0.592 |

FIGURE 26 – NEURAL NETWORKS




| | Parameters | Train | Validation |
|---|---|---|---|
| **6842** | gini, 71, 82 | 1.000 | 0.956 |
| **5862** | gini, 61, 82 | 1.000 | 0.956 |
| **7332** | gini, 76, 82 | 1.000 | 0.956 |
| **5274** | gini, 55, 82 | 1.000 | 0.956 |
| **6254** | gini, 65, 82 | 1.000 | 0.956 |
| **...** | ... | ... | ... |
| **9633** | entropy, 2, 31 | 0.754 | 0.709 |
| **9632** | entropy, 2, 30 | 0.754 | 0.709 |
| **9631** | entropy, 2, 29 | 0.754 | 0.709 |
| **9630** | entropy, 2, 28 | 0.754 | 0.709 |
| **9604** | entropy, 2, 2 | 0.754 | 0.709 |

FIGURE 27 – DECISION TREE




| | Parameters | Train | Validation |
|---|---|---|---|
| **76** | 75, friedman_mse, deviance, 0.5 | 1.000 | 0.966 |
| **88** | 75, squared_error, deviance, 0.5 | 1.000 | 0.966 |
| **64** | 50, squared_error, deviance, 0.5 | 1.000 | 0.964 |
| **52** | 50, friedman_mse, deviance, 0.5 | 1.000 | 0.964 |
| **159** | 300, squared_error, deviance, 0.1 | 1.000 | 0.962 |
| **...** | ... | ... | ... |
| **7** | 10, friedman_mse, exponential, 0.03 | 0.876 | 0.841 |
| **12** | 10, squared_error, deviance, 0.01 | 0.865 | 0.827 |
| **0** | 10, friedman_mse, deviance, 0.01 | 0.865 | 0.827 |
| **18** | 10, squared_error, exponential, 0.01 | 0.865 | 0.825 |
| **6** | 10, friedman_mse, exponential, 0.01 | 0.865 | 0.825 |

FIGURE 28 – GRADIENT BOOSTING

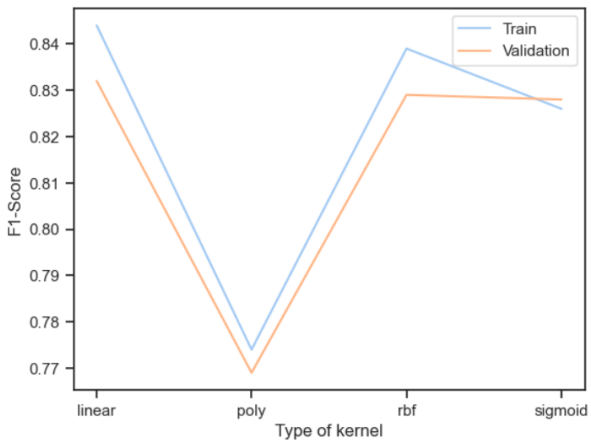
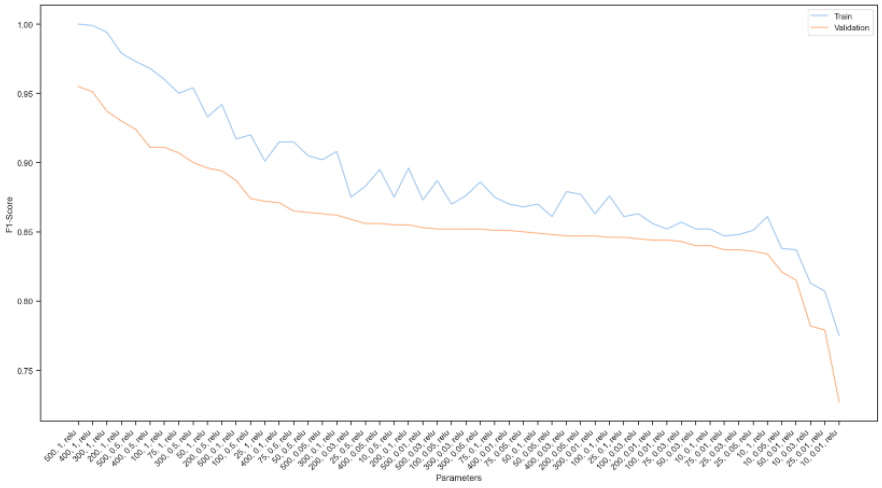



FIGURE 29 – SUPPORT VECTOR MACHINE




| | Parameters | Train | Validation |
|---|---|---|---|
| **53** | 500, 1, relu | 1.000 | 0.955 |
| **47** | 400, 1, relu | 0.999 | 0.951 |
| **41** | 300, 1, relu | 0.994 | 0.937 |
| **35** | 200, 1, relu | 0.979 | 0.930 |
| **52** | 500, 0.5, relu | 0.973 | 0.924 |
| **46** | 400, 0.5, relu | 0.968 | 0.911 |
| **29** | 100, 1, relu | 0.960 | 0.911 |
| **23** | 75, 1, relu | 0.950 | 0.907 |
| **40** | 300, 0.5, relu | 0.954 | 0.900 |
| **17** | 50, 1, relu | 0.933 | 0.896 |
| **34** | 200, 0.5, relu | 0.942 | 0.894 |
| **51** | 500, 0.1, relu | 0.917 | 0.887 |
| **28** | 100, 0.5, relu | 0.920 | 0.874 |
| **11** | 25, 1, relu | 0.901 | 0.872 |
| **45** | 400, 0.1, relu | 0.915 | 0.871 |
| **22** | 75, 0.5, relu | 0.915 | 0.865 |
| **16** | 50, 0.5, relu | 0.905 | 0.864 |
| **50** | 500, 0.05, relu | 0.902 | 0.863 |
| **39** | 300, 0.1, relu | 0.908 | 0.862 |

FIGURE 30 - ADABOOST

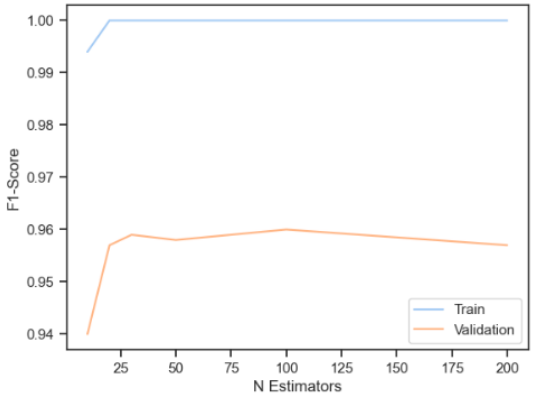
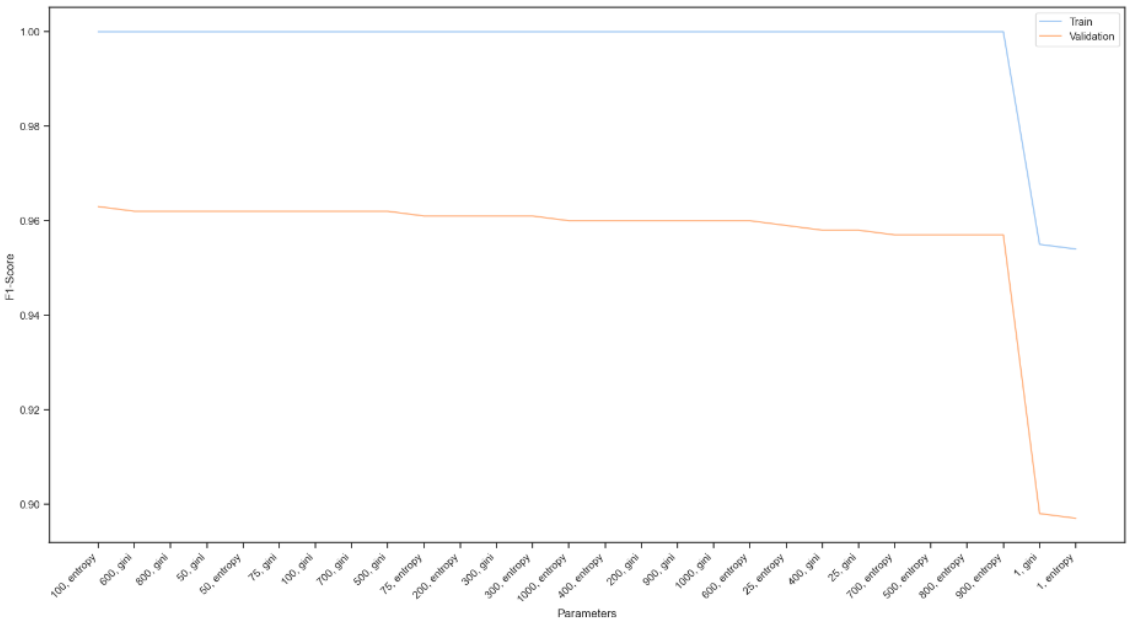



FIGURE 31 – XG BOOST




| | Parameters | Train | Validation |
|---|---|---|---|
| 9 | 100, entropy | 1.000 | 0.963 |
| 18 | 600, gini | 1.000 | 0.962 |
| 22 | 800, gini | 1.000 | 0.962 |
| 4 | 50, gini | 1.000 | 0.962 |
| 5 | 50, entropy | 1.000 | 0.962 |
| 6 | 75, gini | 1.000 | 0.962 |
| 8 | 100, gini | 1.000 | 0.962 |
| 20 | 700, gini | 1.000 | 0.962 |
| 16 | 500, gini | 1.000 | 0.962 |
| 7 | 75, entropy | 1.000 | 0.961 |
| 11 | 200, entropy | 1.000 | 0.961 |
| 12 | 300, gini | 1.000 | 0.961 |
| 13 | 300, entropy | 1.000 | 0.961 |
| 27 | 1000, entropy | 1.000 | 0.960 |
| 15 | 400, entropy | 1.000 | 0.960 |
| 10 | 200, gini | 1.000 | 0.960 |
| 24 | 900, gini | 1.000 | 0.960 |
| 26 | 1000, gini | 1.000 | 0.960 |
| 19 | 600, entropy | 1.000 | 0.960 |
| 3 | 25, entropy | 1.000 | 0.959 |
| 14 | 400, gini | 1.000 | 0.958 |
| 2 | 25, gini | 1.000 | 0.958 |
| 21 | 700, entropy | 1.000 | 0.957 |
| 17 | 500, entropy | 1.000 | 0.957 |
| 23 | 800, entropy | 1.000 | 0.957 |
| 25 | 900, entropy | 1.000 | 0.957 |
| 0 | 1, gini | 0.955 | 0.898 |
| 1 | 1, entropy | 0.954 | 0.897 |

FIGURE 32 – RANDOM FOREST

| Modelo | Parâmetro | Score |
|---|---|---|
| LogisticRegression | l1 and saga | 0.834 |
| GaussianNB | 0.0001 | 0.824 |
| KNN | 49, distance, brute | 0,971 |
| NeuralNetworks | 100, logistic, lbfgs | 0.964 |
| DecisionTrees | gini, 71, 82 | 0,956 |
| RandomForest | 100, entropy | 0,963 |
| GradientBoosting | 75, friedman_mse, deviance, 0.5 | 0.966 |
| SVM | linear | 0.831 |
| AdaBoost | 500, 1, relu | 0.955 |
| XGBoost | 100 | 0.96 |
| Stacking Classifier | knn, rf & lr | 0.967 |

FIGURE 33 - BEST PARAMETER FOR EACH MODEL

| Modelo | F1-Score(Validation) |
|---|---|
| LogisticRegression | 0.8300395256916996 |
| GaussianNB | 0.816 |
| KNN | 0.9501915708812262 |
| NeuralNetworks | 0.9774436090225563 |
| DecisionTrees | 0.9545454545454547 |
| RandomForest | 0.9649805447470817 |
| GradientBoosting | 0.9729729729729729 |
| SVM | 0.8346456692913387 |
| AdaBoost | 0.937984496124031 |
| XGBoost | 0.9694656488549619 |
| Stacking Classifier | 0.9689922480620154 |

FIGURE 34 - SUMMARY

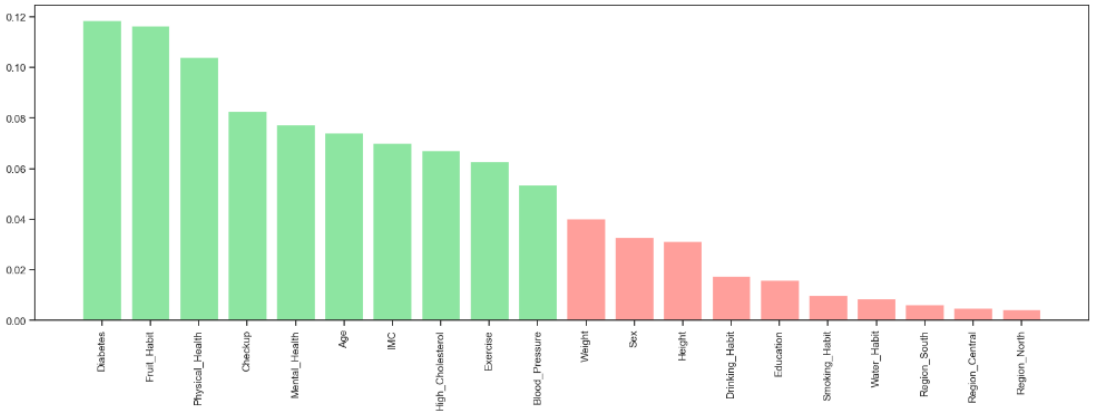



FIGURE 35 - RANDOM FOREST ALGORITHM