Predictive Modelling for *TechScape*

Machine Learning | MDSAA 2021/22

Group 60
António Fonseca - r20181154
João Carvalho - r20181122
Lucas Lopes - r20181111
Mariana Ferreira - r20181071
Tiago Oom - r20181077

Professors: Roberto Henriques, Carina Albuquerque and Lara Oliveira

Index

1. Introduction

This report addresses the final project in the Machine Learning course of the Master Degree Program in Data Science and Advanced Analytics of Nova IMS, which focuses on creating a predictive model to help TechScape, a Portuguese startup, sell their goods online.

For the development of the project the group impersonated a team of data scientists, given the task of analyzing the online behavior of TechScape's customers and to predict which ones have a high probability of acquiring their products depending on their online actions.

In the following pages, we present and discuss the entire process of the project development including data treatment, models created, decisions made, and major findings.

2. Initial Data

To begin with, we imported all the needed libraries for our project. After this, we used pandas to import 'train.csv' (which we gave the name techscape) and we checked the first records of the dataset to ensure that everything went well. Table 1, in the annex, shows all the 17 variables we have on our dataset along a brief description of its meaning.

3. Dataset exploration

To have better insights regarding the initial dataset we have, we perform some analysis concerning the statistics of the variables, their graphic visualizations and data types, the coherence of the data in the context of this problem and finally some cleaning of outliers.

3.1. Descriptive statistics

Firstly, we changed the types of some variables from object into strings, such as Access_ID, OS, Country, Type_of_Visitor, Type_of_Traffic, and Browser. Since we want to perform the exact same procedure to the test dataset when we deploy our final model in the end of our project, we created a function to do this step called change_data_types. After this, we inspected the descriptive statistics of all the variables (table 2 of the annex) and found that there are some discrepancies, meaning there is also high variance on the values of Account_Mng_Durantion, FAQ_Duration, Product_Pages, Product_Durantion, GoogleAnalytics_PageValue, which can indicate that there are some unidimensional outliers on theses variables to be solved later. Moreover, we can also notice by this table that there are no negative values on the numerical variables which makes sense in this context.

3.2. Missing values

Then, we checked for missing data and realized that no null values existed in the 9999 existing records. This extremely facilitated the preprocessing stage of our dataset.

3.3. Data visualization

To complement our quantitative analysis of the dataset, we performed some visualizations of both numerical and categorical variables, by doing histograms and bar plots according to the values of the target, respectively, and also a pairwise comparison of numerical variables to assess the presence of possible multidimensional outliers.

By performing these visualizations, we noticed that almost all the numerical variables are very skewed to the right meaning that is not usual that our clients spend a lot of time navigating on the website and it is really plausible that we have outliers on our dataset. In the context of this problem, we can also highlight that the time that customers spend on the website is not directly proportional to their probability of buying a product since the proportion of clients that buy regarding the values of each variable is relatively constant. Moreover, we can also see that more visits and purchases are from people who use Windows, and Android is also preferable for buying in comparison with Mac. The visits and purchases come mainly from browser 1 and 2, and the visits are usually from the traffic sources 1, 2 and 3, being the proportion of new clients that buy something almost 50%, a much higher value than this same proportion for returner clients. Lastly, we can also see that the majority of the clients are from the Iberian Peninsula but most of the visits are made by customers from Portugal.

From the pairwise comparison between numerical variables, we can conclude that there are various possible multidimensional outliers because there are a lot of observations that are far away from the usual cloud of points of the plots, and we can also conclude about some visual patterns on the scatter plots, namely that Product_Duration and Product_Pages are linearly correlated as well as GoogleAnalytics_ExitRate and GoogleAnalytics_BounceRate. This means that the others are not, so, later on, we should evaluate correlations with Spearman coefficient that accounts for non-linear relationships. In the context of the problem, we can also see that GoogleAnalytics_PageValue has a big influence on the possibility of buying products but GoogleAnalytics_BounceRate and GoogleAnalytics_BounceRate have not.

3.4. Coherence checking

As for the incoherencies in techscape, we looked at the columns of number of Pages and Duration for each type of category (AccountMng, FAQ, and Product). We corrected the values of these columns in the following way: if we have a Duration>0 but no Pages were seen, we set Pages equals to 1; if we have Pages>0, but no Duration exists, we set Duration equals to 1. Similar to the data types, we built a function called inchorencies_check to do this.

3.5. Outliers

Outliers should only be removed from the training data: since the test data may or may not have outliers, they should not be removed from the validation data too. Since our split of the data is a complex process, the outlier detection is explained later in this paper.

4. Feature engineering

Besides the original variables of the dataset, we created 53 new variables by combining or discretizing the initial ones to increase the number of possible independent variables that influence our target. This feature engineering process also included some adjustments of incoherent values to allow correct calculations, namely on the new variables Total_Pages, Total_Duration, AvgTimeOnAccountMng, AvgTimeOnFAQ and AvgTimeOnProduct. These new variables and the description of their engineering process are presented in table 3 of the appendices.

Furthermore, we also performed a PCA analysis to check how they would perform in our models. We concluded by the Kaiser´s, Pearson and Scree plot that the best number of components to retain was 3.

5. Feature selection

5.1. Correlation matrix

For the feature selection section of the project, we first decided to check for any univariate variables since it would be pointless to keep one in the model as it would not provide any relevant information and we concluded there was none.

Then we resorted to the Spearman matrix of correlations (figure 1 of the annex) to search for any pair/subset of variables which were too correlated with each other since keeping these variables in a model could harm its performance. We were able to identify some problematic pairs of variables and took that in consideration further in the project when estimating the models

Note that this step was only performed on numerical variables as it is not adequate for categorical variables.

5.2. Stratified k-folds for numerical data

Next, we proceeded to a more objective process of feature selection. More concretely we performed seven feature selection algorithms: Lasso Regression, Decision Tree (Gini), Decision tree (Entropy), Regressive Feature Elimination (RFE), Backward Regression, Forward Selection, and Select From Model. Note that these algorithms were also only performed on numerical variables (they are not suitable for categorical ones).

For those algorithms, we decided to use stratified 10-folds to split our data into train and validation and performed each one of those using 10 different train and validation sets. This way we were able to avoid the potential bias caused by using only one regular split of the data. In the end, we were able to visualize which variables were constantly identified as relevant for our model. The results' overview is displayed in the annex on table 4.

5.3. Stratified k-folds for categorical data

For the categorical features, we performed the exact same analysis but with only one algorithm, the *Chi-Square test for categorical data*. The results overview is in the annex on table 5. To better understand the importance of the categorical variables which we found to be relevant, we created a visualization that is displayed in the annex on figure 2.

At the end of this feature selection section, we were able to identify which variables are plausible candidates to be included in the final model. However, the final results are just a guideline as we try multiple combinations of variables and evaluate their performance.

6. Model training

6.1. Models

In our demand for the best possible model, we tried several algorithms: Decision Tree, Neural Network, K-nearest neighbours, Naïve Bayes, Support Vector Machine, Random Forest, Bootstrap Aggregation, Extremely Randomized Trees, Ada Boost, Gradient Boost and Histogram based Gradient Boost. In the end, we also tried Stacking on some of our best models.
The procedure on how we modelled each one of them will now be explained:

1. Grid Search

To start things of, we split the data into train and validation using a usual train test split. The train data could then be scaled and treated for outliers (note that both the scaling and the removal of outliers are defined as parameters of a function, so we can do only one, both or neither one of these two steps). Then, we performed a grid search only on the training data with the intention of finding the best parameters for each model. This grid search uses a stratified k-fold of its own so only a subset of the training data would be used in every iteration to estimate the models. This process was facilitated with the creation of the function run_grid that would receive the following arguments: the data to be split, the random state, the algorithm, the parameters, the option to remove outliers, the option to scale the data and the scaling method to be used (if needed). In the end of this step we would get, at least, a plausible model for further experiencing.

2. Validation

In this second step, we started by splitting the data into train and validation using the same random state as above. Once we had a plausible model from the grid, we trained it on the train set and evaluated its performance on the validation set (that was left out previously for the grid search). The main objective of this step was to evaluate the performance of the model on new data to be aware of how it performs in terms of overfitting. To evaluate the performance, we resourced to the f-1 scores from the training and validation data, plotted the ROC curve and other performance metrics like accuracy, support, precision, and recall, as well as the confusion matrix.

3. Stratified K-fold for performance evaluation

Since the results from the previous steps could vary a lot depending on some minor changes, (like the random state of the train test split, for example) we decided to also evaluate the performance of the models using a stratified 5-fold on the whole dataset. This way, it was possible to not only train but also evaluate our predictive algorithms on different portions of the whole dataset to avoid a mis-representative performance of the models. Similar to the grid search step, each one of the different 5 training sets created would have the options of being scaled and treated for outliers. At the end of this step, we were able to evaluate the performance of the model (with f1-score) on every iteration of the k-folds and also the average score for the different training and validation sets. The main purpose of this process was not only to evaluate the performance of a model but also its consistency using different data in order to test, once again, the presence of overfitting. This whole process was facilitated with the creation of the function avg_score that would receive the following arguments: the data to split, the splitting method, the algorithm, the option to remove outliers, the option to scale the data and the scaling method to be used (if needed).

4. Final Model

Lastly, the final model is trained using all the available data for the estimation. We decided to do this because once an algorithm has been tuned to achieve the best performance, we are better off using all the data we have to estimate it. Nevertheless, once we had the final submission, we would still check if it would perform as expected in the 30% of the test set available in Kaggle. By “as expected” we mean that the score available in Kaggle is very close to the one obtained when fitting our model to the entire dataset.

6.2. Data preprocessing

There are some data pre-processing steps that should be performed only on the training data. However, as explained above, since we used stratified K-fold to evaluate the performance of a model, the training data set was not always the same and so, it was a more complex task. The steps are explained further.

6.2.1. *Outlier detection and removal*

We performed outlier detection and removal while performing all the steps explained above. On the Grid Search, since we used a regular train test split, the outliers would simply be removed from the training data set before the data was passed to the Grid Search function. On the Stratified k-fold, the outliers would be removed only on each one of the 5 training data sets that would be used (we do not want to remove outliers from the validation data set because they can be present in the test data and we do not want our evaluation to biased by the non-presence of outliers). We tried different methods for univariate and multivariate outliers, which are better explained later in the annex: Manual Approach, Standard Deviation, K-means for outlier detection and Isolation Forest and evaluated their performances and we concluded that main decisive factor to choose the best one of them would be the percentage of observations removed and not the method used. But, in the end, we concluded that to not remove outliers at all is also a valid approach. Theoretically, this can be justified by the fact that the test data will have outliers and by keeping them in the training data, our model would become more robust, and they would not harm its performance as much.

6.2.2. Scaling

The second step was to scale the data. Once again, the scaler must be fit only to the training data because we want to transform our entire data based only on our training data. We tried 4 different scaling methods: MinMaxScaler ([0,1]), MinMaxScaler ([-1,1]), StandardScaler, RobustScaler and evaluated the performance of our models with each one of them.

7. Model deployment

In the deployment phase, we want to apply our best model to the test dataset and predict if that customers will or will not finalized their actions on the website with a transaction. To do so, we start by importing the 'test.csv' file and change the types of the variables with the function change_data_types. We can then decide if we want to correct our incoherencies or not. Finally, we create all our new variables, also with the use of a function created previously (create_new_variables).

In the subsequent part of the code, we give the option of scaling or not the test data (since the models that work with Decision Trees, for example, do not need scaling). We then use our best model to predict the test data and add a new column called 'Buy' with the results obtained (each record is classified by either 0-not buy or 1-buy). Because Kaggle is expecting our submission to be in a specific format, we extract from this dataset just the two needed columns (*Access_ID* and *Buy*) and transform them into a csv file. Moreover, in order to keep track of the submission number and the submission details (such as selected features, scaling used, and the parameters of the final model), we export a text file to store all that needed information.

8. Discussion

We tried our best to perform all the best pre-processing steps, the most precise feature selection methods and feature engineering ideas. In the modelling part we explored a large amount of machine learning algorithms with different combinations of features to be used, different types of incoherencies to be corrected, different methods of outliers' removal, different scaling techniques, different sets of algorithm parameters and different metrics for performance evaluation.

By comparing all the models, we discovered that decision trees-based algorithms usually outperform the other models as we obtain higher training scores. Additionally, we explored more in-depth boosting algorithms, in which base estimators are built sequentially in order to reduce the bias. Having this in mind, we tried to find a decision trees-based boosting algorithm.

We came across two main algorithms that fulfil the above criterion: gradient boost and histogram gradient boost. After a long and extensive analysis trying to have the best score possible while reducing as much as possible the overfitting, we concluded that the histogram gradient boost algorithm controls better the overfitting. Some performance measures were taken into account for this analysis such as: the ROC Curve (figure 3 in the annex), and the confusion matrix (figure 4 in the annex). Moreover, we applied stacking between a histogram gradient boost and several other algorithms to see if its performance increases.

In the end, the final and best algorithm regarding our analysis was a Histogram Gradient Boost with the following parameters: learning_rate = 0.1700000000000001, min_samples_leaf = 100, max_leaf_nodes = 11, warm_start = True, loss='auto', max_iter= 71, and max_depth=6. These specific values were obtained using a for loop that would iterate over several possible values to obtain the most accurate tuning. The model was estimated without the removal of outliers and incoherencies, since it was the combination that led to better results. Note that, since it is a decision trees-based algorithm, we can use all the features and we do not need to perform scaling on the dataset.

This final model was the one that got the overall best results both in our analysis and in the 30% of the test data available in Kaggle. It scored an average of 0.72907 for the train data and 0.72897 for the validation data in the stratified 5-fold for performance evaluation. The score obtained for the train dataset is 0.72908, and the Kaggle score is 0.71186 (overfitting= 0.01722).

9. Conclusions

We tried to make the jupyter notebook as practical as possible to turn the process of exploring all these machine learning solutions and procedures into a feasible process. In the end, the most challenging task was to make sure that our final model would be the best performer not only on our data and the 30% of the test data available in Kaggle, but also on the most important data, the 70% test set which is unknown.

Therefore, there were some limitations along with the development of the project such as the small number of observations that we had both for training our model (9999 records) and for testing it in the end. Since we only had about 2300 records available to test the performance of our final model, we were constantly having a big variance and inconsistency in the obtained results, which also led to some difficulties when trying to control the overfitting of the models. Regarding the context of the problem, some limitations could affect later suggestions to the company, namely the non-detailed description of variables such as Type_of_Traffic and Browser, and also the presence of bad indicative labels on some records such as the country of a client that can be registered as Other.

Besides this, overall, we achieved our main objective of deploying a robust machine learning model to predict the customers with a high probability of buying TechScape products based on their online behaviour.

10. References

[1] Sklearn.ensemble.BaggingClassifier. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

[2] Sklearn.svm.SVC. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn/stable/modules/generated/sklearn.svm.SVC.html

[3] Sklearn.naive_bayes.GaussianNB. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

[4] Sklearn.neighbors.KNeighborsClassifier. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[5] Example gallery—Seaborn 0.11.2 documentation. (n.d.). Retrieved 26 December 2021, from https://seaborn.pydata.org/examples/index.html

[6] Sklearn.linear_model.LinearRegression. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[7] Brownlee, J. (2018, May 22). A Gentle Introduction to k-fold Cross-Validation. Machine Learning Mastery. https://machinelearningmastery.com/k-fold-cross-validation/

[8] Sklearn.preprocessing.StandardScaler. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[9] Akshara_416. (2021, July 26). Isolation Forest | Anomaly Detection with Isolation Forest. Analytics Vidhya https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/

[10] Analytics Vidhya. (2019, June 24). Feature Selection methods with example (Variable selection methods). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/

[11] Wijaya, C. Y. (2021, October 12). 5 Feature Selection Method from Scikit-Learn you should know. Medium. https://towardsdatascience.com/5-feature-selection-method-from-scikit-learn-you-should-know-ed4d116e4172

[12] sklearn.feature_selection.SelectFromModel. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html

[13] Ensemble methods. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn.org/stable/modules/ensemble.html

[14] sklearn.ensemble.ExtraTreesClassifier. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier

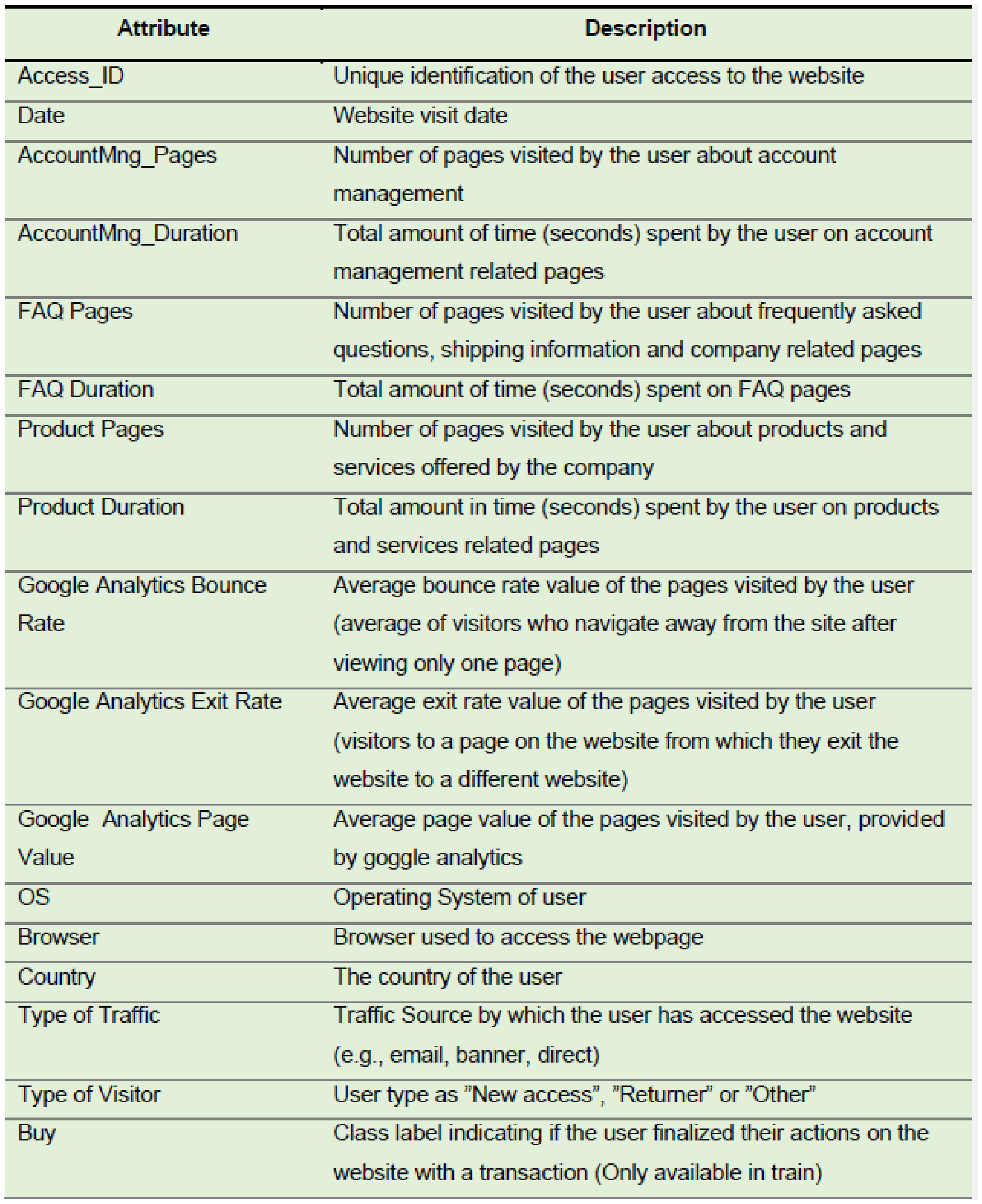
[15] sklearn.ensemble.HistGradientBoostingClassifier. (n.d.). Scikit-Learn. Retrieved 26 December 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html#sklearn.ensemble.HistGradientBoostingClassifier

[16] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

[17] A Simple Explanation of K-Means Clustering and its Adavantages. (2020, October 4). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/

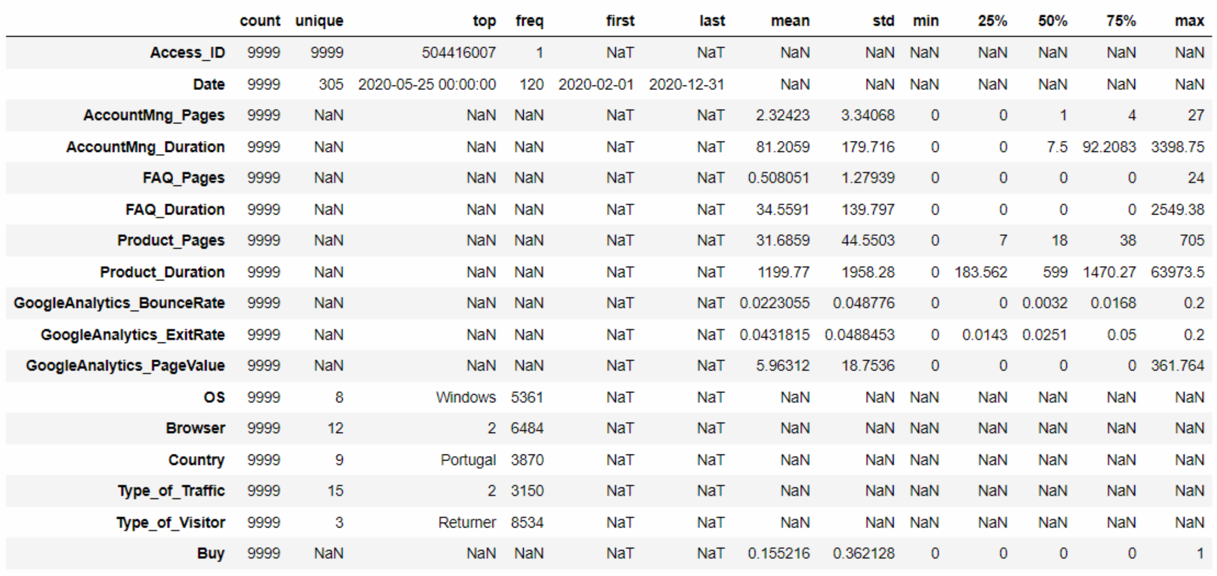
11. Appendices

11.1. Tables and figures



Attribute	Description
Access_ID	Unique identification of the user access to the website
Date	Website visit date
AccountMng_Pages	Number of pages visited by the user about account management
AccountMng_Duration	Total amount of time (seconds) spent by the user on account management related pages
FAQ Pages	Number of pages visited by the user about frequently asked questions, shipping information and company related pages
FAQ Duration	Total amount of time (seconds) spent on FAQ pages
Product Pages	Number of pages visited by the user about products and services offered by the company
Product Duration	Total amount in time (seconds) spent by the user on products and services related pages
Google Analytics Bounce Rate	Average bounce rate value of the pages visited by the user (average of visitors who navigate away from the site after viewing only one page)
Google Analytics Exit Rate	Average exit rate value of the pages visited by the user (visitors to a page on the website from which they exit the website to a different website)
Google Analytics Page Value	Average page value of the pages visited by the user, provided by goggle analytics
OS	Operating System of user
Browser	Browser used to access the webpage
Country	The country of the user
Type of Traffic	Traffic Source by which the user has accessed the website (e.g., email, banner, direct)
Type of Visitor	User type as "New access", "Returner" or "Other"
Buy	Class label indicating if the user finalized their actions on the website with a transaction (Only available in train)

Table 1 *- Initial Variables.*



	count	unique	top	freq	first	last	mean	std	min	25%	50%	75%	max
Access_ID	9999	9999	504416007	1	NaT	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Date	9999	305	2020-05-25 00:00:00	120	2020-02-01	2020-12-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
AccountMng_Pages	9999	NaN	NaN	NaN	NaT	NaT	2.32423	3.34068	0	0	1	4	27
AccountMng_Duration	9999	NaN	NaN	NaN	NaT	NaT	81.2059	179.716	0	0	7.5	92.2083	3398.75
FAQ_Pages	9999	NaN	NaN	NaN	NaT	NaT	0.508051	1.27939	0	0	0	0	24
FAQ_Duration	9999	NaN	NaN	NaN	NaT	NaT	34.5591	139.797	0	0	0	0	2549.38
Product_Pages	9999	NaN	NaN	NaN	NaT	NaT	31.6859	44.5503	0	7	18	38	705
Product_Duration	9999	NaN	NaN	NaN	NaT	NaT	1199.77	1958.28	0	183.562	599	1470.27	63973.5
GoogleAnalytics_BounceRate	9999	NaN	NaN	NaN	NaT	NaT	0.0223055	0.048776	0	0	0.0032	0.0168	0.2
GoogleAnalytics_ExitRate	9999	NaN	NaN	NaN	NaT	NaT	0.0431815	0.0488453	0	0.0143	0.0251	0.05	0.2
GoogleAnalytics_PageValue	9999	NaN	NaN	NaN	NaT	NaT	5.96312	18.7536	0	0	0	0	361.764
OS	9999	8	Windows	5361	NaT	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Browser	9999	12	2	6484	NaT	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Country	9999	9	Portugal	3870	NaT	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Type_of_Traffic	9999	15	2	3150	NaT	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Type_of_Visitor	9999	3	Returner	8534	NaT	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Buy	9999	NaN	NaN	NaN	NaT	NaT	0.155216	0.362128	0	0	0	0	1

Table 2 *- Descriptive statistics.*

New variable	Description
Apple	1 if OS equals MacOSX or iOS, 0 otherwise
Windows	1 if OS equals Windows, 0 otherwise
Android	1 if OS equals Android, 0 otherwise
Returner	1 if Type_Of_Visitor equals Returner, 0 otherwise
New_Access	1 if Type_Of_Visitor equals New_Access, 0 otherwise
Total_Duration	AccountMng_Duration + FAQ_Duration + Product_Duration
Total_Pages	AccountMng_Pages + FAQ_Pages + Product_Pages
PropTime_AccountMng	AccountMng_Duration/Total_Duration
PropTime_FAQ	FAQ_Duration/Total_Duration
PropTime_Product	Product_Duration/Total_Duration
PropPages_AccountMng	AccountMng_Pages/Total_Pages
PropPages_FAQ	FAQ_Pages/Total_Pages
PropPages_Product	Product_Pages/Total_Pages
AvgTimePerPage	Total_Duration/Total_Pages
AvgTimeOnAccountMng	AccountMng_Duration/AccountMng_Pages
AvgTimeOnFAQ	FAQ_Duration/FAQ_Pages
AvgTimeOnProduct	Product_Duration/Product_Pages
Month	Month extraction from Date
Covid	1 if Month is greater than 3, 0 otherwise
Browser	One new binary variable per each Browser type
Country	One new binary variable per each Country

Table 3 *- New Variables.*

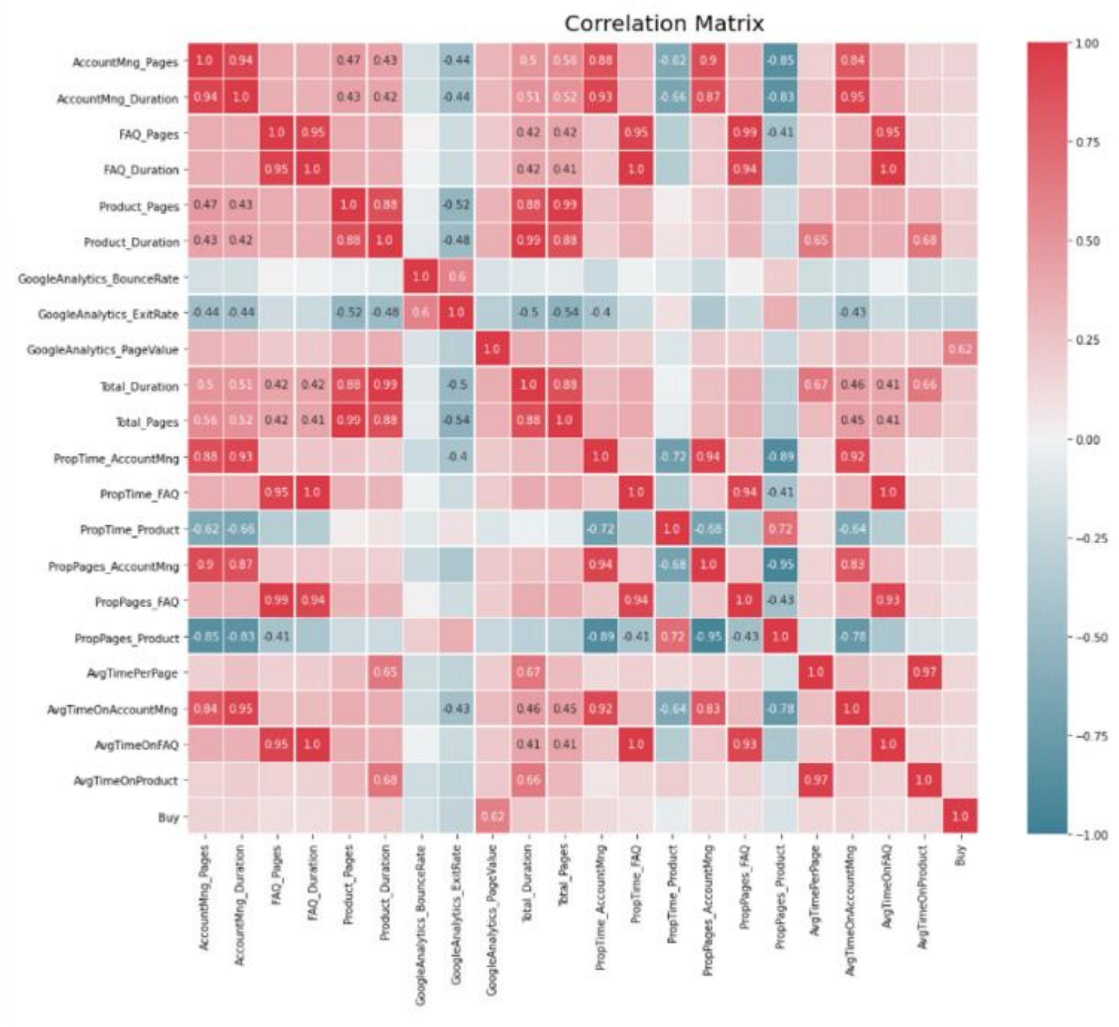
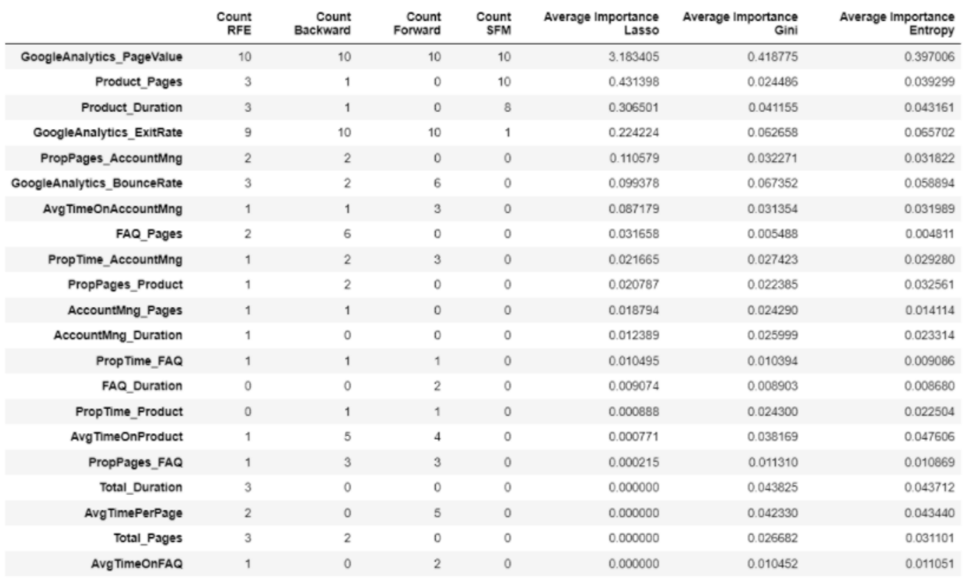
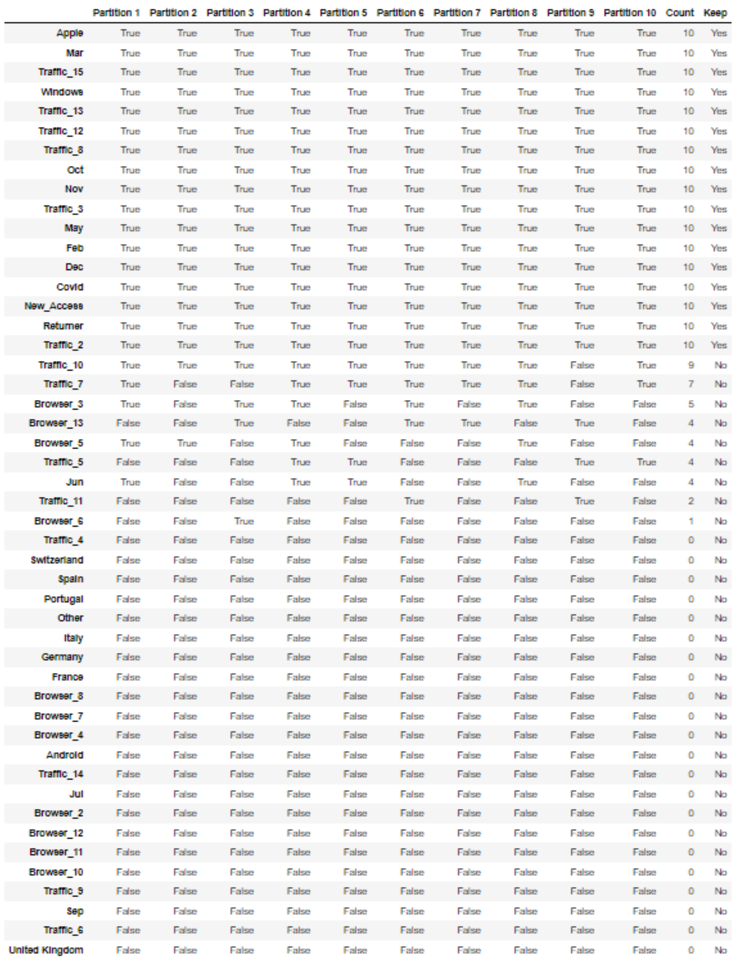


Figure 1 - *Spearman correlation matrix.*



	Count RFE	Count Backward	Count Forward	Count SFM	Average Importance Lasso	Average Importance Gini	Average Importance Entropy
GoogleAnalytics_PageValue	10	10	10	10	3.183405	0.418775	0.397006
Product_Pages	3	1	0	10	0.431398	0.024486	0.039299
Product_Duration	3	1	0	8	0.306501	0.041155	0.043161
GoogleAnalytics_ExitRate	9	10	10	1	0.224224	0.062658	0.065702
PropPages_AccountMng	2	2	0	0	0.110579	0.032271	0.031822
GoogleAnalytics_BounceRate	3	2	6	0	0.099378	0.067352	0.058894
AvgTimeOnAccountMng	1	1	3	0	0.087179	0.031354	0.031989
FAQ_Pages	2	6	0	0	0.031658	0.005488	0.004811
PropTime_AccountMng	1	2	3	0	0.021665	0.027423	0.029280
PropPages_Product	1	2	0	0	0.020787	0.022385	0.032561
AccountMng_Pages	1	1	0	0	0.018794	0.024290	0.014114
AccountMng_Duration	1	0	0	0	0.012389	0.025999	0.023314
PropTime_FAQ	1	1	1	0	0.010495	0.010394	0.009086
FAQ_Duration	0	0	2	0	0.009074	0.008903	0.008680
PropTime_Product	0	1	1	0	0.000888	0.024300	0.022504
AvgTimeOnProduct	1	5	4	0	0.000771	0.038169	0.047606
PropPages_FAQ	1	3	3	0	0.000215	0.011310	0.010869
Total_Duration	3	0	0	0	0.000000	0.043825	0.043712
AvgTimePerPage	2	0	5	0	0.000000	0.042330	0.043440
Total_Pages	3	2	0	0	0.000000	0.026682	0.031101
AvgTimeOnFAQ	1	0	2	0	0.000000	0.010452	0.011051

Table 4 - *Stratified k-fold for numerical features selection.*



	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5	Partition 6	Partition 7	Partition 8	Partition 9	Partition 10	Count	Keep
Apple	True	True	True	True	True	True	True	True	True	True	10	Yes
Mar	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_15	True	True	True	True	True	True	True	True	True	True	10	Yes
Windows	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_13	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_12	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_8	True	True	True	True	True	True	True	True	True	True	10	Yes
Oct	True	True	True	True	True	True	True	True	True	True	10	Yes
Nov	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_3	True	True	True	True	True	True	True	True	True	True	10	Yes
May	True	True	True	True	True	True	True	True	True	True	10	Yes
Feb	True	True	True	True	True	True	True	True	True	True	10	Yes
Dec	True	True	True	True	True	True	True	True	True	True	10	Yes
Covid	True	True	True	True	True	True	True	True	True	True	10	Yes
New_Access	True	True	True	True	True	True	True	True	True	True	10	Yes
Returner	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_2	True	True	True	True	True	True	True	True	True	True	10	Yes
Traffic_10	True	True	True	True	True	True	True	True	False	True	9	No
Traffic_7	True	False	False	True	True	True	True	True	False	True	7	No
Browser_3	True	False	True	True	False	True	False	True	False	False	5	No
Browser_13	False	False	True	False	False	True	True	False	True	False	4	No
Browser_5	True	True	False	True	False	False	False	True	False	False	4	No
Traffic_5	False	False	False	True	True	False	False	False	True	True	4	No
Jun	True	False	False	True	True	False	False	True	False	False	4	No
Traffic_11	False	False	False	False	False	True	False	False	True	False	2	No
Browser_6	False	False	True	False	False	False	False	False	False	False	1	No
Traffic_4	False	False	False	False	False	False	False	False	False	False	0	No
Switzerland	False	False	False	False	False	False	False	False	False	False	0	No
Spain	False	False	False	False	False	False	False	False	False	False	0	No
Portugal	False	False	False	False	False	False	False	False	False	False	0	No
Other	False	False	False	False	False	False	False	False	False	False	0	No
Italy	False	False	False	False	False	False	False	False	False	False	0	No
Germany	False	False	False	False	False	False	False	False	False	False	0	No
France	False	False	False	False	False	False	False	False	False	False	0	No
Browser_8	False	False	False	False	False	False	False	False	False	False	0	No
Browser_7	False	False	False	False	False	False	False	False	False	False	0	No
Browser_4	False	False	False	False	False	False	False	False	False	False	0	No
Android	False	False	False	False	False	False	False	False	False	False	0	No
Traffic_14	False	False	False	False	False	False	False	False	False	False	0	No
Jul	False	False	False	False	False	False	False	False	False	False	0	No
Browser_2	False	False	False	False	False	False	False	False	False	False	0	No
Browser_12	False	False	False	False	False	False	False	False	False	False	0	No
Browser_11	False	False	False	False	False	False	False	False	False	False	0	No
Browser_10	False	False	False	False	False	False	False	False	False	False	0	No
Traffic_9	False	False	False	False	False	False	False	False	False	False	0	No
Sep	False	False	False	False	False	False	False	False	False	False	0	No
Traffic_6	False	False	False	False	False	False	False	False	False	False	0	No
United Kingdom	False	False	False	False	False	False	False	False	False	False	0	No

Table 5 *- Stratified k-fold for categorical feature selection.*

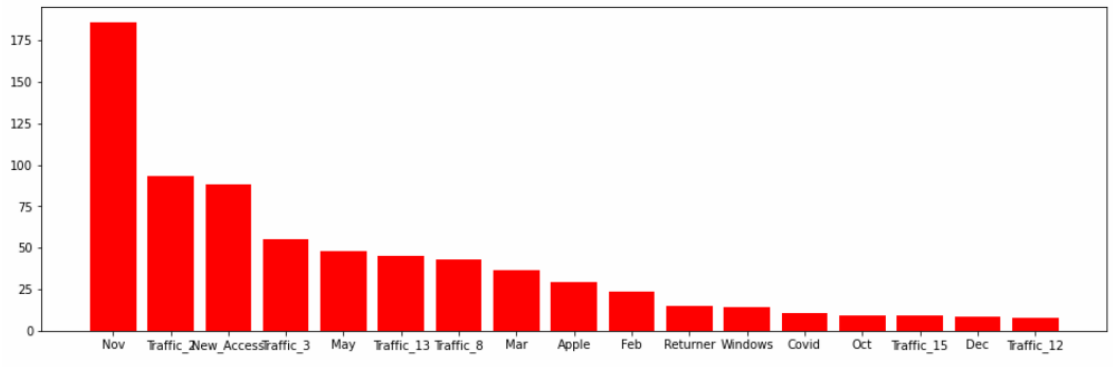


Figure 2 *- Categorical features importance visualization.*

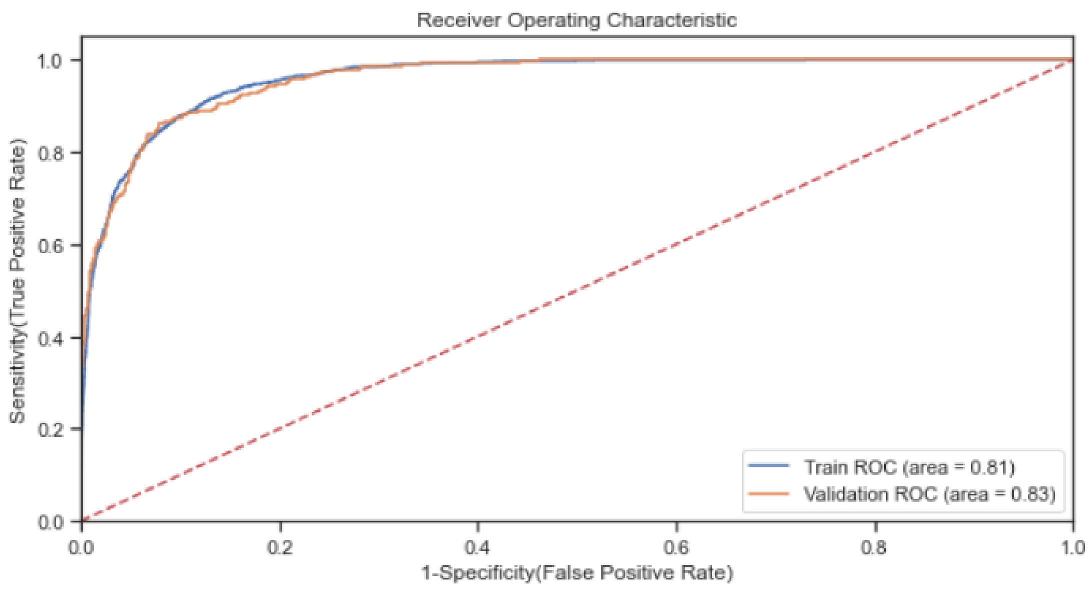


Figure 3 *- ROC Curve.*

```
_______________________________________________________
                          TRAIN
-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.94      0.97      0.96      6757
           1       0.82      0.65      0.73      1242

    accuracy                           0.92      7999
   macro avg       0.88      0.81      0.84      7999
weighted avg       0.92      0.92      0.92      7999

[[6578  179]
 [ 430  812]]
_______________________________________________________
                        VALIDATION
-------------------------------------------------------
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      1690
           1       0.78      0.70      0.74       310

    accuracy                           0.92      2000
   macro avg       0.86      0.83      0.85      2000
weighted avg       0.92      0.92      0.92      2000

[[1629   61]
 [  94  216]]
```

Figure 4 *- Confusion Matrix.*

11.2. Self-study

<u>Outlier detection methods</u>

(1) Univariate outliers

→ Mannual Approach: We evaluated the histogram and boxplots of the numerical variables and identified possible outliers.

→ Standard Deviation: After standardizing the numerical variables, this method identifies as outliers the observations that below in the tails of the normal distribution. The threshold is a function of the standard deviation and we tried different ones that would identify less or more observations as outliers.

(2) Multivariate outliers

→ K-means: clustering method that when use for outlier detection will identify the observations from each cluster that are the most different from the cluster´s neighbours.

→ Isolation Forest: unsupervised outlier detection technique, based on decision trees, like Random Forest, and built on the assumption that anomalies are the data points that are "few and different". In an Isolation Forest, randomly sub-sampled data is processed in a tree structure based on randomly selected features. The samples that travel deeper into the tree are less likely to be anomalies as they required more cuts to isolate them. Similarly, the samples which end up in shorter branches indicate anomalies as it was easier for the tree to separate them from other observations.

<u>Feature engineering process - PCA</u>

Furthermore, to choose the most relevant features, we decided to perform PCA to reduce dimensionality and retain the best combinations of the most important variables.

→ Principal Component Analysis is a dimensionality reduction method that combines all the existing variables in a set of new uncorrelated variables called the principal components. The first one is obtained storing the maximum variance possible of the original variables and each succeeding component in turn has the highest variance.

Feature selection methods

For the Feature Selection process, we decided to try more three methods: Forward Selection, Backward Elimination and Select from model

- → Forward selection is an iterative method which starts with no features in a regression and by each iteration, if the feature is identified as relevant, it is added. Otherwise, it is left out. The estimator used was a logistic regression.
- → Backward selection is an iterative method which starts with all the features in a regression and by each iteration, a feature is removed if it is identified as irrelevant for the model. The estimator used was a logistic regression.
- → Select From Model, just like RFE, is based on a Machine Learning Model estimation for selecting features. The difference is that SFM feature selection is based on the importance attribute threshold, which by default, is the mean. The features are considered unimportant and removed if the corresponding values are below the provided threshold parameter.

Models

→ Extremely Randomized Trees

Extremely Randomized Trees is an ensemble method (averaging method), therefore, its main purpose is to combine several estimators in order to improve robustness. Similar to the Random Forest Model, it creates random uncorrelated trees that form the forest.

The major difference between these two Forests of randomized trees is that Extremely Randomized Trees does not select the most discriminative thresholds to perform the splitting rule, thresholds are selected at random and the best one is chosen to perform the splitting rule, consequently, it often reduces the variance of the model.

→ Histogram based Gradient Boost

Histogram-Based Gradient Boosting is an ensemble method (boosting method), therefore, its objective is to estimate several models which are sequentially created where the following one tries to reduce the bias of the previous model.

The advantages of this model compared to the gradient boosting method is that it is faster for larger datasets since when it estimates this boosting method starts to allocate the input samples into the bins (default value - 255) and as a consequence, it reduces the number of splits which leads to a faster approach of the dataset. Additionally, it estimates even if the dataset has missing values.