Machine Learning Group Project
Techspace – E-commerce

Nuno Penim m20210998 (m20210998@novaims.unl.pt)

Diogo Tomas Peixoto m20210993 (m20210993@novaims.unl.pt)

Goncalo Gomes m20211007 (m20211007@novaims.unl.pt)

Paulo Oliveira m20211002 (m20211002@novaims.unl.pt)

I - Introduction

In the beginning of 2020, five entrepreneurs built TechScape, a start-up company to sell goods related to digital detox. TechScape sells products and services using an online store

Due to the global pandemic caused by Covid-19, in the beginning of March of 2020, the company halted their work in April of the same year. They returned in May of 2020. TechScape hired a team of Data Scientists to analyse online tendencies and the behaviour of their customers and visitors to the webpage, in order to be able to predict who will buy the products available.

This study aims to develop a predictive model, based on a given train dataset, that will predict the customers who are more likely to buy TechScape's products. The train data contains 9999 records and is about the behaviours of the customers who visited the website between February and December 2020.

Our final goal is to obtain the best possible performance on a test dataset containing 2300 unseen records, by applying different Machine Learning algorithms and by following different pipelines with different parameters and criteria applied.

II – Methodology

I - Data Exploration

As presented above, we have received two datasets summarized below.

Datasets	Number of Observations
Train	9999
Test	2300

Table 1: Datasets and number of observations

Each dataset entails 16 independent variables both numerical and categorical. The train dataset includes the binary dependent variable. All variables can be easily interpreted apart from two related to google analytics, which are explained below.

Starting with "GoogleAnalytics_BounceRate", we can consider this variable to be referring to a single-page session on our website, more specifically, bounce rate is the percentage of all sessions on the website in which users viewed only a single page and triggered only a single request to the Analytics server. Since this is an e-commerce website, it will probably make sense to consider that customers with low bounce rate values will be good candidates to buy existing products on the website.

For “GoogleAnalytics_ExitRate”, we can define it as the percentage of all page views for an individual page that made it the last page in the session. This means that, for example, on an e-commerce site, such as TechScape, if a customer has high exit rate values, we can probably assume that either something is wrong with our product page or even the product itself that causes originally interested buyers to leave after checking out the product.

As explained before, the dependent variable “Buy” is a binary variable, which takes the value 1 if a certain customer has made a purchase and 0 if he did not. Hence, our main goal is to classify the customers into two groups (buyers and non-buyers) based on classification algorithms.

All initial independent variables are presented below in accordance with its type.

Initial Independent Variables				
Numerical		**Categorical**		**Date**
Continuous	**Discrete**	**Nominal**	**Ordinal**	**-**
AccountMng_Duration	AccountMng_Pages	OS		Date
FAQ_Duration	FAQ_Pages	Browser		
Product_Duration	Product_Pages	Country		
GoogleAnalytics_BounceRate		Type_of_Traffic		
GoogleAnalytics_ExitRate		Type_of_Visitor		
GoogleAnalytics_PageValue				

Table 2: Initial Independent Variables Types

Afterwards, we have checked the existence of missing values and duplicated values in both train and test dataset. None were found.

II – Data Visualization

In this next step, we have started by plotting the categorical variables frequency. The plots showed us some trends, please refer to figure 2 in appendix, such as:

- Windows was the most used Operating System
- Most of the customers were returners
- Portugal was the country with most accesses to the website

Next, we have performed a similar process by plotting the numerical variables histograms. Please refer to figure 3 in appendix. Most of our numerical variables are highly right-skewed (as we will be able to confirm through the boxplots analysis).

The train dataset presents an unequal target class “Buy” distribution, meaning that our dataset is imbalanced. There’s approximately 8500 and 1500 customers, who have visited the website and ended up buying or not buying any product, respectively. This might pose a challenge to our project since most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class[1]. That’s the reason, we have used the SMOTE technique, as will be explained later.

III – Data Transformation and Observation

We have begun by setting the “Access_ID” as the new index for both the train and test datasets to assign a unique identification to each observation.

In addition to this, we have figured it would be interesting to extract some potential insights that could be implied in the *Date* variable, namely trying to find possible seasonal trends, creating the following ones. At this point, we had 9 categorical variables in total.

New Variables	
Month	Vales ranging from 1 to 12
Day	Values ranging from 1 to 31
Weekday	Values ranging from 1 to 7
Yearday	Values ranging from 1 to 365

Table 3: New Created Variables

Following the date variable transformations, we have analysed the categorical variables frequency and proportion bar charts crossed with the dependent variable. The most relevant plots are shown in figure 4 in appendix. In addition to the insights presented above for the categorical variables, it’s also valuable to refer:

- Fedora was the only OS whose accesses did not result in any purchase
- Portugal was by far the country with the most buys
- Returner customers were the most common type of visitors, however new accesses actually did more purchases in percentage, when compared to everyone else
- The months with the most associated purchases was May and November, and summer months were considerably worse in terms of procurement. May is probably explained by the excitement of reopening the shop after its closure in April. November is probably explained by the proximity to Christmas and due to the increase in Covid-19 cases that led people looking for more anti-stress products
- The day with less purchases was the 31st of every month, due to the fact that not every month has a 31st day
- The least productive weekday was Saturday.

Afterwards, we had to encode the categorical variables still in string format (“OS”,” Country” and “Type of Visitor”). We have considered both label encoding and one hot encoding options. We have decided to use the One Hot Encoding because the referred categorical variables are not ordinal. With label encoding we would introduce an order/hierarchy, leading to give further importance to the instances with higher values encoded when using algorithms distance based. [2]. Eventually, we have dropped the original string categorical variables.

IV - Coherence Check

We then moved on to perform a coherence check on the train dataset' numerical variables. The goal is to assure that it does not have illogical and incoherent values, which could impair the performance of our models.

All google analytical variables are not easy to interpret since they do not point out to a specific page of our website, but rather are an average of the pages visited by the user. Anyway, both "BounceRate" and "ExitRate" variables' values are low and that's a good sign for an e-commerce business. So, no further search has been done.

The remaining numerical variables have three pairs that are correlated and hence, can be checked.

We have started by checking customers that visited two or more "Account Management pages", but spent zero seconds in them, since it is highly unlikely the duration would remain zero with this condition. The second check we did was regarding the possibility that a user had not visited any page and yet the duration time was bigger than zero seconds. Observations such as these should be considered anomalies.

The same reasoning was applied to both "FAQ" and "Product" variables. The results are summarized below.

Coherence Check		
Condition Nº	**Condition**	**Number of Observations**
a)	AccountMng_Pages>=2 and AccountMng_Duration=0	3
b)	AccountMng_Pages=0 and AccountMng_Duration>0	0
c)	FAQ_Pages>=2 and FAQ_Duration=0	9
d)	FAQ_Pages>=0 and FAQ_Duration=0	0
e)	Product_Pages>=2 and Product_Duration=0	130
f)	Product_Pages>=0 and Product_Duration=0	0

Table 4: Coherence Check

We have decided to preserve these observations, but adjust its values of condition number a), c) and e). We believe that these are due to minor miscalculations and therefore remain relevant to the development of the models. For each page variable, we have calculated the respective median duration and inputted its value to the observations presented above.

V - Data Partition

Afterwards, we have split the original train dataset using the train-test-split technique from sklearn, with a split of 80%-20% of the observations for the training and validation dataset, respectively. Please note we have purposefully named training dataset, which is different from the original train dataset referred along the report until now. We have used the parameter, stratify=target, so that the proportion of each class was preserved in each dataset. We have also defined the parameter, shuffling=True, to avoid introduce potential bias.

Datasets	Number of Observations
Training	7950
Validation	2000

Table 5: Training and validation number of observations

Then, we have divided all the existing variables into categorical and numeric so that we can, for example, proceed with handling outliers in the numeric variables and by applying different feature selection algorithms to both numeric and categorical variables.

VI – Outlier Treatment

We have started by plotting the graphs of the training dataset's numeric variables, the cumulative distribution function plots and the respective boxplots, and we could confirm what we already concluded in the data visualization chapter, which is that all the numerical variables are considerably right-skewed, and therefore, might consider the existence of some outliers in the right tail of the distribution.

After this, we plotted the pairwise relationship between all numerical variables, looked for patterns that captured our attention like, for example, the positive linear relationship between "GoogleAnalytics_BounceRate" and "GoogleAnalytics_ExitRate".

Proceeding with the outlier removal, the first choice was using the Z-score Method. However, this statistical technique would remove approximately 15% of our total observations. That's a high value and we have decided to not move on with this technique.

Subsequently, we tried to use the IQR method with a lower and upper limit equal to 1.5 inter quartile range. The results were even worse when compared to the z-score method, since about 52% of the total observations would be removed.

For this reason, we have decided to manually filter the outliers by analyzing in combination the relevant pairplots and boxplots plotted before. Please see the table below and the figure 5 in appendix, which are self explanatory.

Manual Outlier Removal			
Filter Nº	**Filters**	**Figures used in Appendix**	**% Outliers removed**
a)	AccountMng_Pages>20 and AccountMng_Duration>2000	5.1	0.15
b)	FAQ_Pages>15 and FAQ_Duration>1500	5.2	0.1875
c)	Product_Pages>400 and Product_Duration>25000	5.3	0.125
d)	GoogleAnalytics_PageValue>150	5.4	0.2125
e)	GoogleAnalytics_BounceRate>0.190	5.5	5.8882
f)	GoogleAnalytics_ExitRate>0.190	5.6	5.9757

Table 6: Manual Outlier Removal

The filters a), b), c) and d) have been applied on the training dataset. Both the the filter e) and f), despite of visually representing a single point in the boxplot, would lead to a removal of about 12% our dataset, hence were not applied.

VII – Variable transformation and Standardization

Some machine learning algorithms assume the variables are normally distributed. Transforming the variables to map their distribution to a Gaussian distribution may, and often does, boost the performance of the machine learning algorithms.

We have decided to apply the Yeo-Johnson Power transformation to our numerical variables and thus creating 9 new variations of these previous variables, which were added to our training dataset.

The new transformed variables histograms' visualization shows some of them more similar to a theoretical Gaussian distribution.

Afterwards, we have applied the MinMaxScaler to the total 18 numerical variables, scaling the variables to a range value between zero and one, while also maintaining their original distribution. Scaling is a necessary step in order to be able to use some feature selection techniques presented on next chapter.

VIII – Feature Selection

After normalization, it is time to apply different methods that will allow us to select the best set of features in order to use them in our models. That's a relevant step to increase the training efficiency of our models, improve their accuracy and also reduce overfitting [3].

Dependent upon the variable is categorical or numerical, different techniques have been applied to each group.

For the categorical variables both the chi-square and the mutual information methods were used. The final categorical picked variables are ("Browser", "Type_of_Traffic", "Month", "Yearday", "x0_Windows", "x1_Germany", "x2_Returner").

The numerical feature selection techniques results are combined on table 7 in appendix. It is the results combination itself that led us to decide to choose the following 4 variables ("Product_Duration", "Product_Pages_yeo", "GoogleAnalytic_ExitRate_yeo" and "GoogleAnalytics_PageValue_yeo"). The table's figures indicate the order of importance given by each method to each variable, starting with the most important. Some threshold values were used in some techniques, and that's the reason several variables have empty cells on the table.

The plot results of some techniques are presented on figure 6 in appendix.

The following paragraphs will briefly describe some relevant insights taken from this process.

Both mutual information and chi-square methods are the ones suitable when the inputs are categorical and the outputs either, hence we have used them. Both confirm the variables "Yearday", "Month", "Type_of_Traffic" and "Browser" are relevant. The remaining variables picked mentioned above have been chosen based on the chi-square output result. Interestingly enough, both methods seem to differ on the dummy variables created by the one-hot encoding technique.

Spearman's correlation heat map shows there wasn't any dent variable highly correlated with our target, with the max value equals to 0.6. Regarding the correlation between independent variables, there's high correlations between *AccountMng_Pages* and *AccountMng_Duration* (0.9), *FAQ_Pages* and *FAQ_Duration* (1.0) and between *Product_Pages* and *Product_Duration* (0.9), as expected. We have paid attention to it during the feature selection, in order to try to prevent multicollinearity in the final variables selected. Otherwise, it could cause a problem when fitting the model and interpret the results [4].

The Recursive Feature Elimination (RFE) is not strongly dependent on its hyper parameters, so we figured we should use a Logistic Regression as the algorithm [5]. The optimal number of variables to use was eleven with a total score of 0.89.

Moving on now to embedded methods, we have used Lasso Regularization, by defining the logistic regression model, with the parameter penalty equals L1. This method shrinks the coefficients to zero, indicating the features that shall be removed.

We have also used random forest and adaboost classifier, which presents the top features that lead to bigger information gains.

IX – Model Training

With the features selected, two main pipelines have been created to train the algorithms. The first pipeline has been produced using the SMOTENC technique, applied to the training dataset that resulted from the train-test-split technique. The idea is to resample the data, in order to reduce the data unbalancing. The output dataset has 13464 observations. The second pipeline is the training dataset itself, with 7950 observations.

Since we have used classification algorithms that are sensitive to the scale of the features, we had to perform feature scaling. Normalization and standardization are both valid techniques, but we have chosen only one to decrease the number of posterior algorithm iterations. Normalization is usually used when data does not have a Gaussian Distribution, amongst other reasons [6]. In addition, we do have independent binary categorical variables created by the one-hot-encoding method. Therefore, minmax scaling technique with the values ranging from 0 to 1 was chosen. The numerical features histograms are show in figure 7, in appendix, to support our reasoning. In addition, robust scaling technique was used because it differentiates from minmax by not restricting the possible minimum and maximum values and by being robust to outliers.

Based on the above, there’s a total of six different datasets, with four scaled and two not scaled. Please see the table below.

Datasets		
Pipeline	**Training**	**Validation**
1	training - train test split	validation - train test split
	training - train test split - minmax	validation - train test split - min max
	training - train test split - robust	validation - train test split - robust
2	training SMOTE	validation - SMOTE
	training SMOTE minmax	validation SMOTE minmax
	training SMOTE robust	validation SMOTE robust

Table 7: Training-validation datasets

We do acknowledge that the algorithms that do not need feature scaling, using training datasets scaled or not scaled would provide the same results. Nevertheless, we have preferred to keep the original datasets not scaled and used them on the referred algorithms.

We have used eleven different classification algorithms. Nine of them have been lectured during the semester and the other two are the ridge classifier and the passive aggressive classifier. Further details about the last two can be found on the appendix.

A similar reasoning has been used in all algorithms. The first step is to choose the most important parameters to finetune. After, we have used the RandomizedSearchCV technique from sklearn to obtain the best hyper parameters that would lead to the highest f1 scoring. At this point, it is important to refer we have decided to optimize the f1 score, unde the parameter scoring within the RandomizedSearchCV, rather than the accuracy because the training dataset is unbalanced. More specifically, our goal is to predict the best possible the customers that purchase items on the e-commerce shop. Hence, we do want to lower as much as possible the false positives and false negatives. The f1 score is the best metric to accomplish it. In addition, we have used repeated stratified kfold under the parameter cv within RandomizedSearchCV, with 10 folds and 3 repetitions.

We have tried using the GridSearchCV instead, but not only the process time would be too long, the few times we have tried to run together with the RandomizedSearchCV, the results difference were not significant.

After the hyperparameters finetuned, the classification report and the confusion_matrix were run on the respective validation dataset, whereby the f1 scores could be visualized. Please refer to the summary results on tables 9 and 10 in appendix. As you might notice, the algorithms that not needed or needed feature scaling, 2 and 4 iterations have been performed, respectively.In addition, we have concluded the best algorithms were the support vector machines, gradient boost and random forest.

Afterwards, we have decided to use the stacking technique combining two or even all three best algorithms mentioned. The only combination that outperformed the best single algorithm alone was the combination of the three algorithms. The F1 score results are shown below for each model.

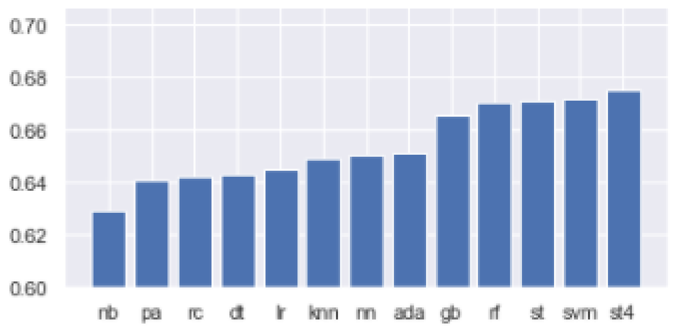


Figure 1:F1 score modes comparison

The support vector machine algorithm has provided the best result (0.6719) not considering the stacking technique. The stacking technique combining the best three algorithms, labeled as st4, returned an f1 score value equals to **0.6755**, and therefore is considered our best model.

To confirm the results, we have plotted the ROC curve for the best 5 models. Please refer to figure 8 in appendix.

REFERENCES

[1] https://machinelearningmastery.com/what-is-imbalanced-classification/

[2] https://towardsdatascience.com/choosing-the-right-encoding-method-label-vs-onehot-encoder-a4434493149b

[3] https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

[4] https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/

[5] https://machinelearningmastery.com/rfe-feature-selection-in-python/

[6] https://becominghuman.ai/what-does-feature-scaling-mean-when-to-normalize-data-and-when-to-standardize-data-c3de654405ed

Appendix

Theoretical Explanation of the algorithm/technique not given in the classes

During our analysis and report, we used several techniques and algorithms, some of which were not studied during the curricular program. This brief background aims to introduce and contextualize such techniques.

The **mutual information** technique for feature selection measures the reduction in uncertainty in variable A when variable B is known. To select variables, we are interested in the mutual information between the predictor variables and the target. Higher mutual information values, indicate little uncertainty about the target Y given the predictor X.

Still in the feature selection, we used the **Lasso Regularization** method. This method was originally formulated for Linear Regression, however it has been adapted for Logistic Regression. With this, we were able to select some features that were important in our dataset. We used a LogisticRegression with the arguments C (the step size, as explained above) as 0.3, penalty as L1, which adds a penalty to our model, limiting the size of the coefficients, the solver as liblinear, which is a library for large linear classifications and a random_state seed, so that we always obtain the same values in every execussion.

In the Variable transformation we applied a **Power Transformation**, specifically using the Yeo-Johnson Transformation. This transformation allows us to normalize the data, making it a distribution more Gaussian like.

After realizing the dataset was unbalanced, we applied the **SMOTE** technique, specifically SMOTENC, which allows us to oversample the dataset that contains both numerical and categorical variables. This technique allows us to oversample the minority class present in our data, in order to reduce the unbalance present in the Dataset.

As a search algorithm, we experimented with **RandomizedSearchCV**, as it seemed faster than GridSearch in our machines. RandomizedSearch differs from GridSearch as not all the parameters need to be tested. It takes as arguments an estimator, which here was variable, as this estimator is the classfier algorithm we are testing, param_distributions, which is a dictionary, that uses as key the parameter name, and as a value, a list of all the parameters to test. This here was also variable depending on the Classifier algorithm we were testing. It also takes n_jobs, which we used as -1, which means it will use the maximum number of threads available. The other argument is cv, which is the cross validation generator, in our case was a repeated stratified kfold. Lastly, it takes a scoring argument, which we used in all cases the f1 score, to obtain the f1 score for our execution.

As an extra, we used the **Passive-Aggressive Algorithm**. This algorithm is in a group of efficient algorithms for large-scale learning. It takes sequential data and the model is updated with each iteration. On the detection of a good prediction, this algorithm does not change the model, while that on the detection of a bad prediction, it changes the model to fit the prediction. In this algorith, we used the C, max_iter and tol arguments. C is refering to the step size, which is used for regularization. This essentially represents the learning rate of the model. Max_iter is the defined maximum number of iterations over the training data. Tol is the tolerance. It lets the model know when to stop. In this specific model it will happen when loss is superior to the subtraction between the previous loss and the tolerance. We used a randomized search function to locate the best values for our arguments, ending up with a Tolerance of 0.00001, a C of 0.3 and a Max Iteration of 500.

We also used the **Ridge Classifier Algorithm**. This algorithm is based on the Ridge Regression. It estimates the coefficients of multiple regression models in scenarios where the independent variable is highly correlated. It initially converts the target values into multiple classes, and then treats the problem as a regression task. In this algorithm we mainly use the alpha argument, which allows us to define the regularization strength. This strength has as a main objective the reduction of overfitting. Similarly, we used a randomized search function to determine the best values for our model, ending up with an alpha of 0.6.

Support Report Figures and Tables

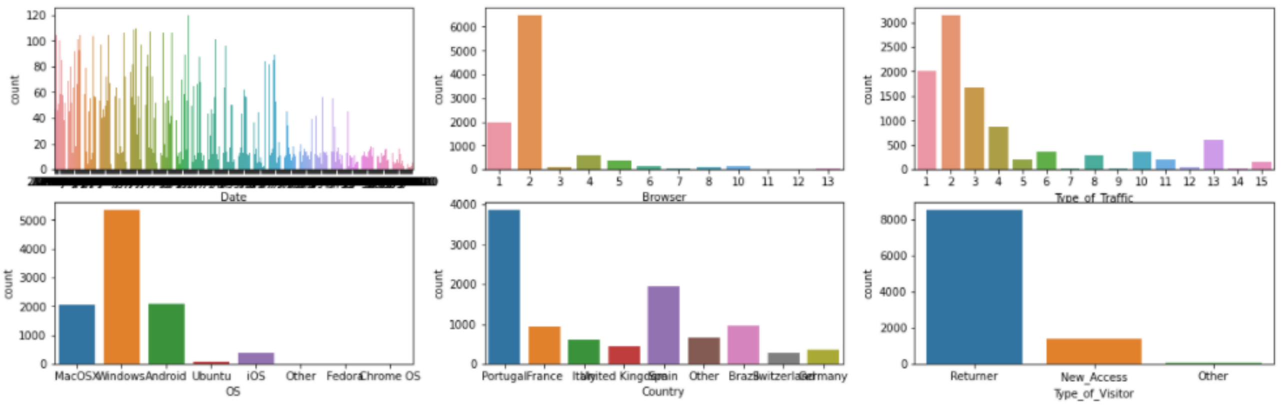


Figure 2: Frequency Categorical Variables

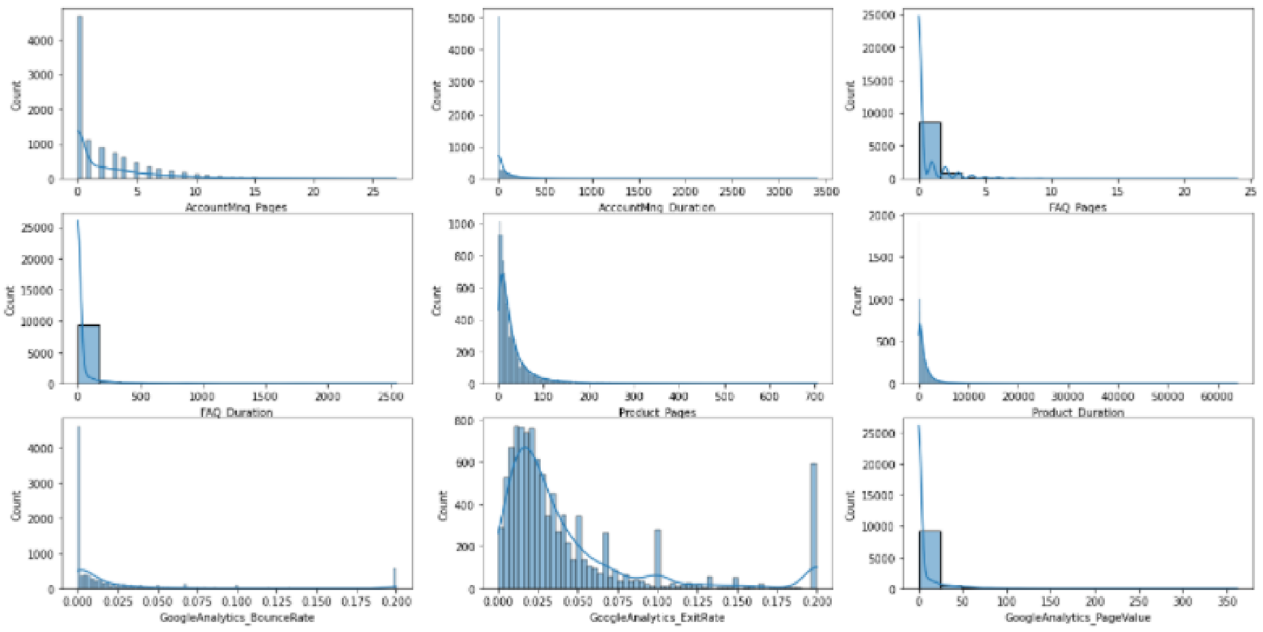
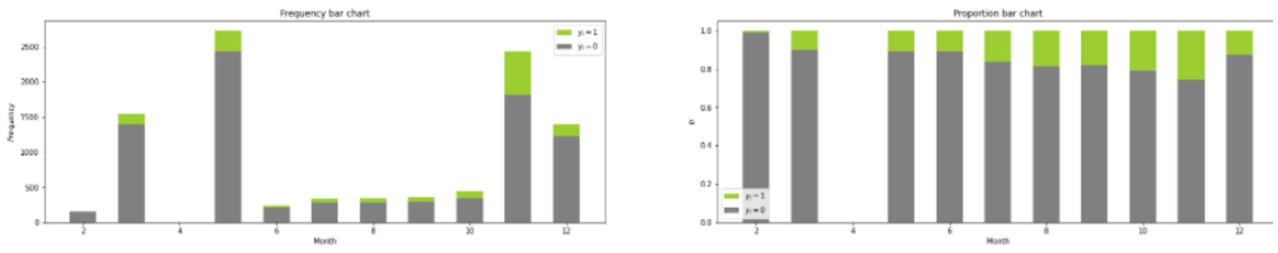


Figure 3: Numerical Variables Histogram's

Month



Yearday

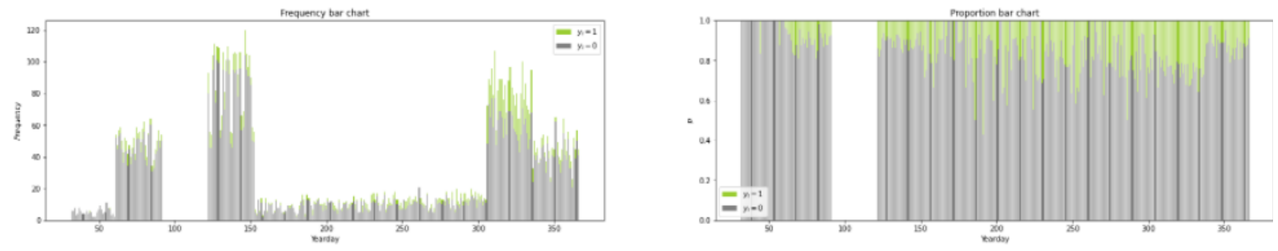
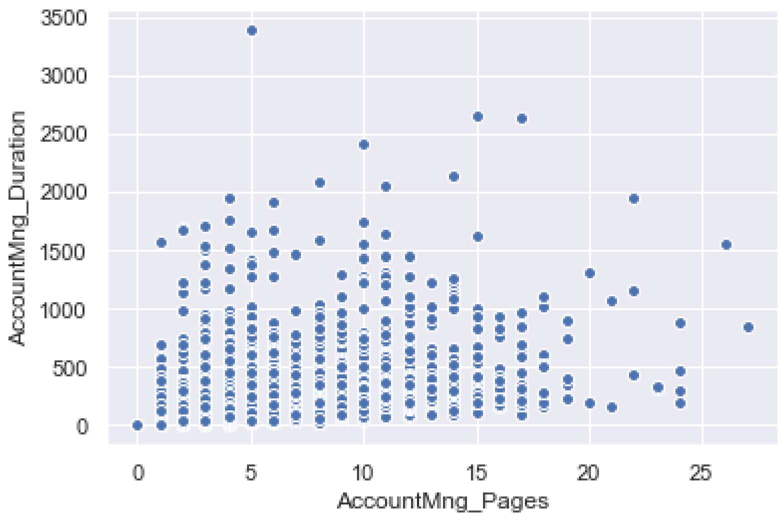
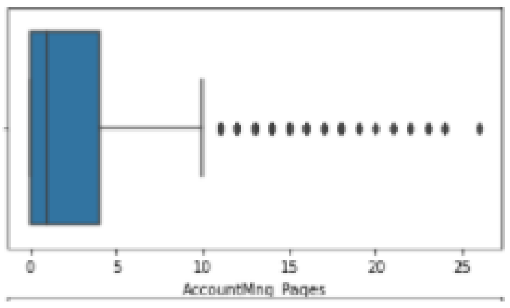


Figure 4: Frequency and proportion of categorical variables crossing with the target class

Figure 5.1





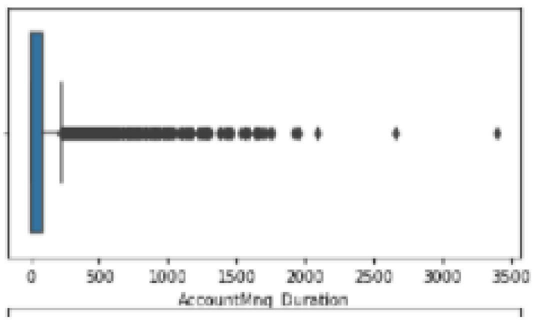


Figure 5.2

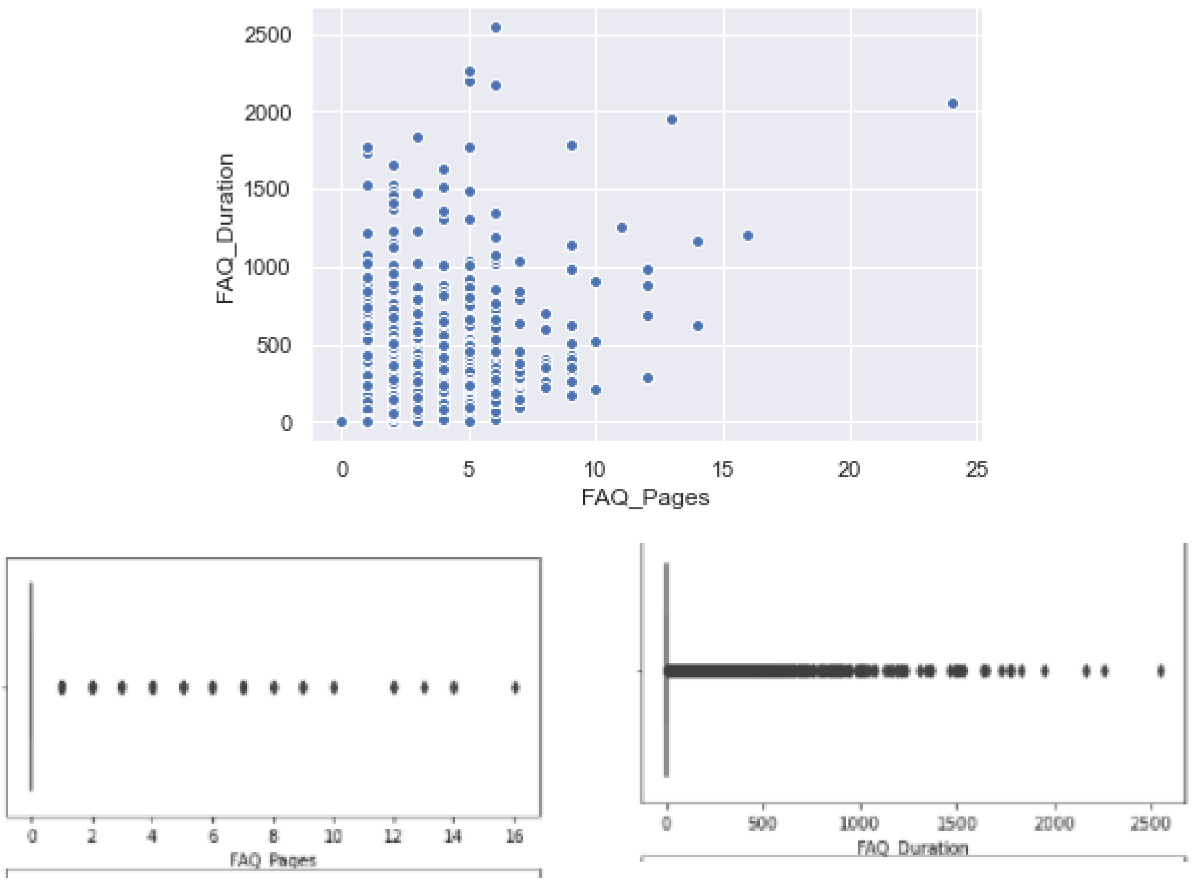


Figure 5.3

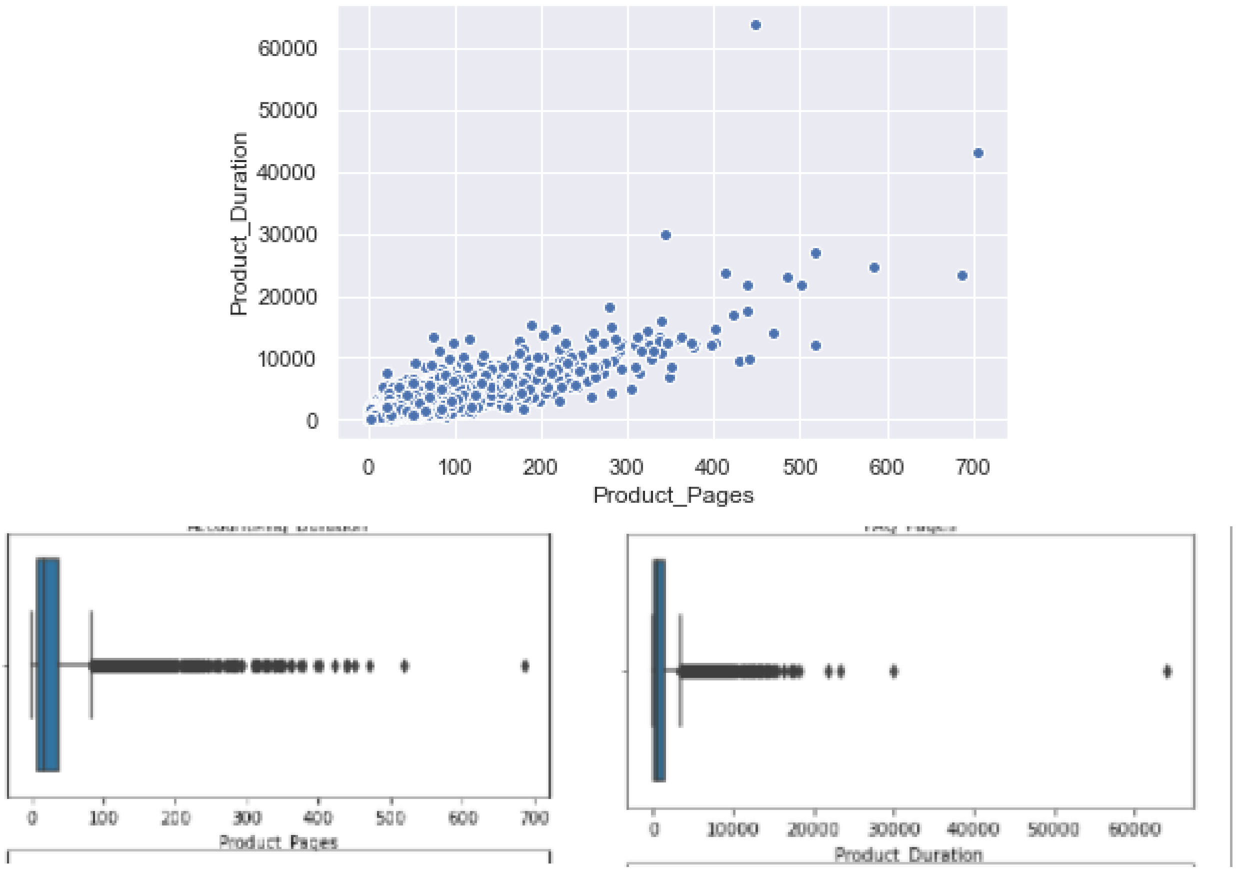


Figure 5.4

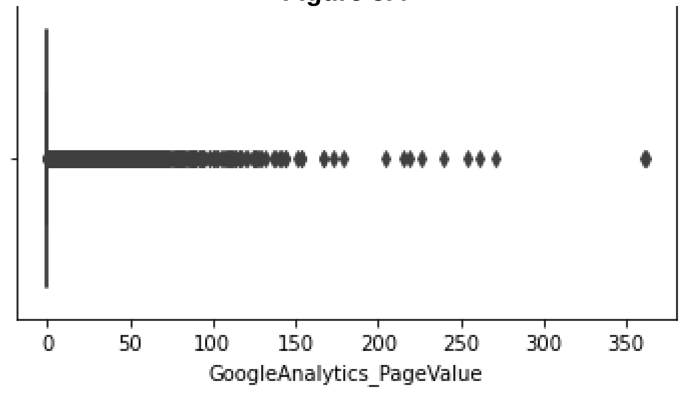


Figure 5.5

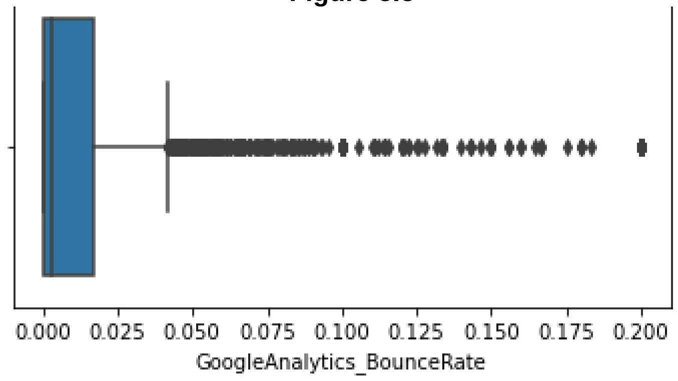


Figure 5.6

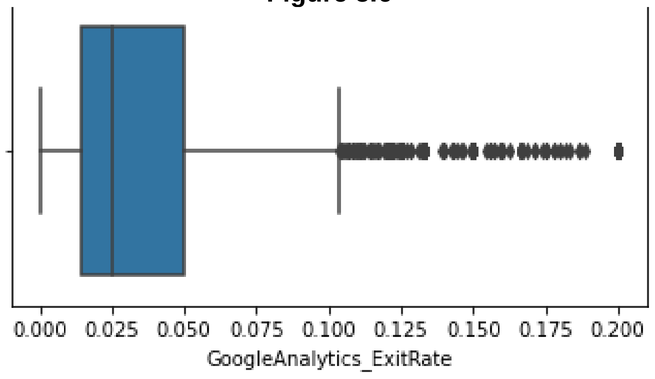
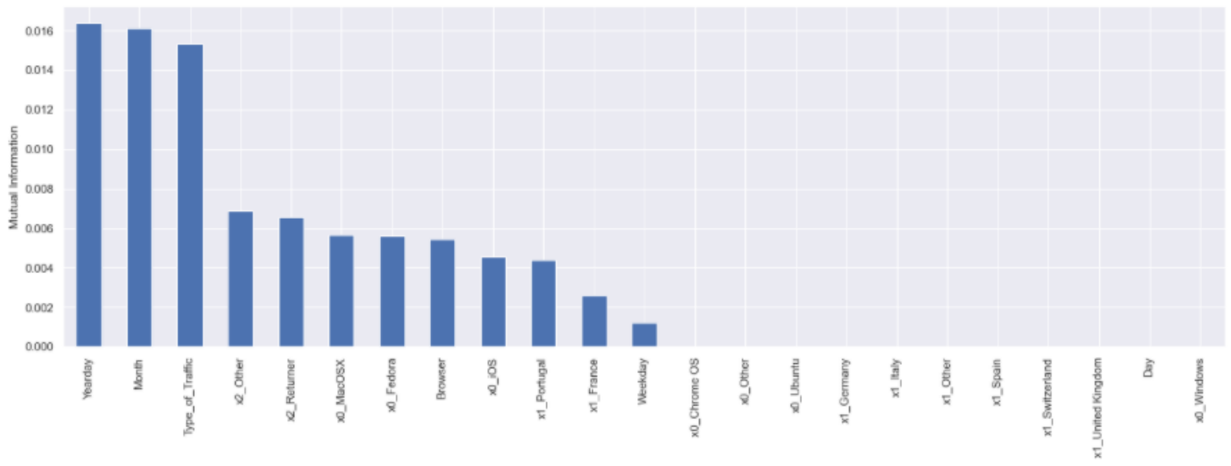
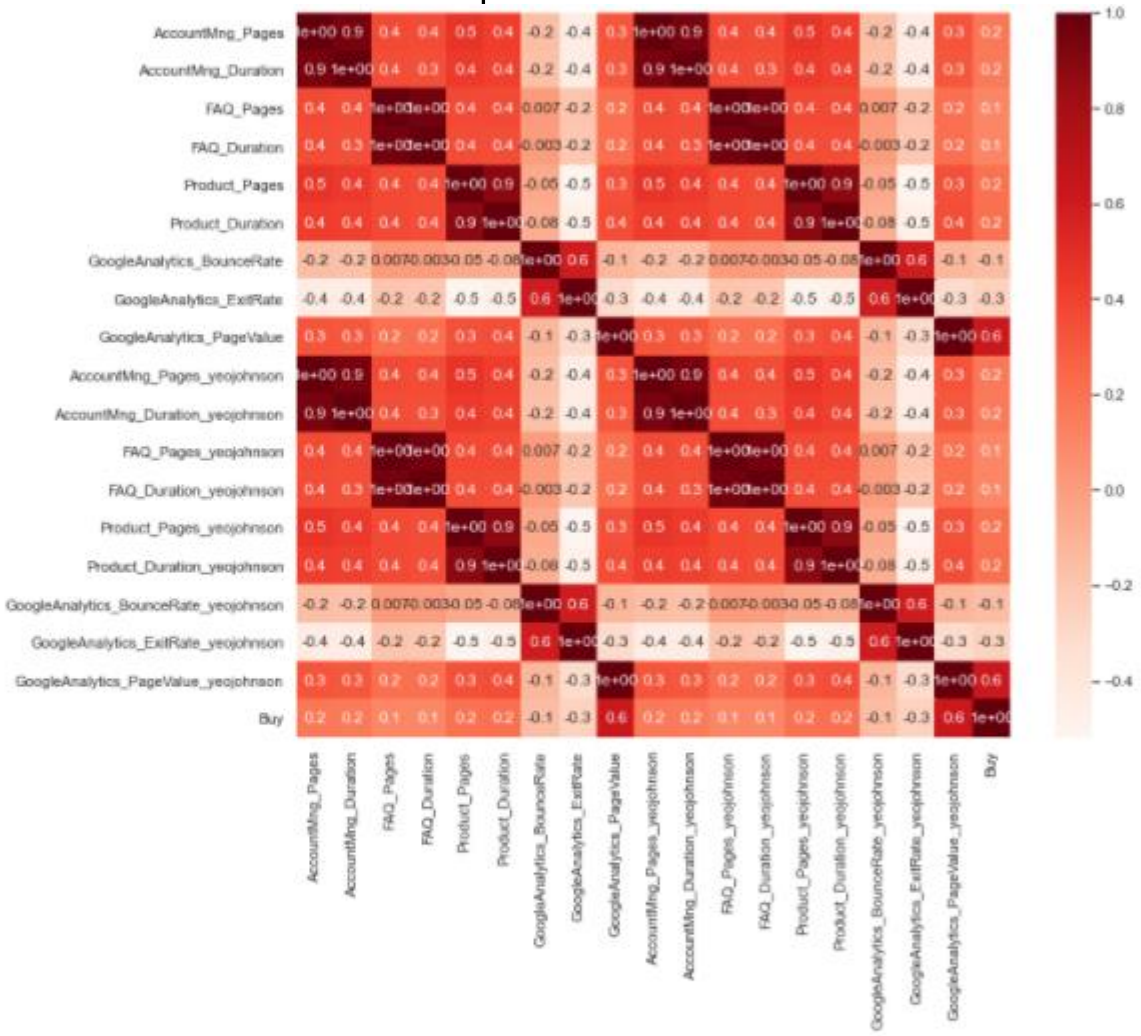


Figure 5: Pairplots and Boxplots for ouliers removal

Mutual Information



Spearman Correlation



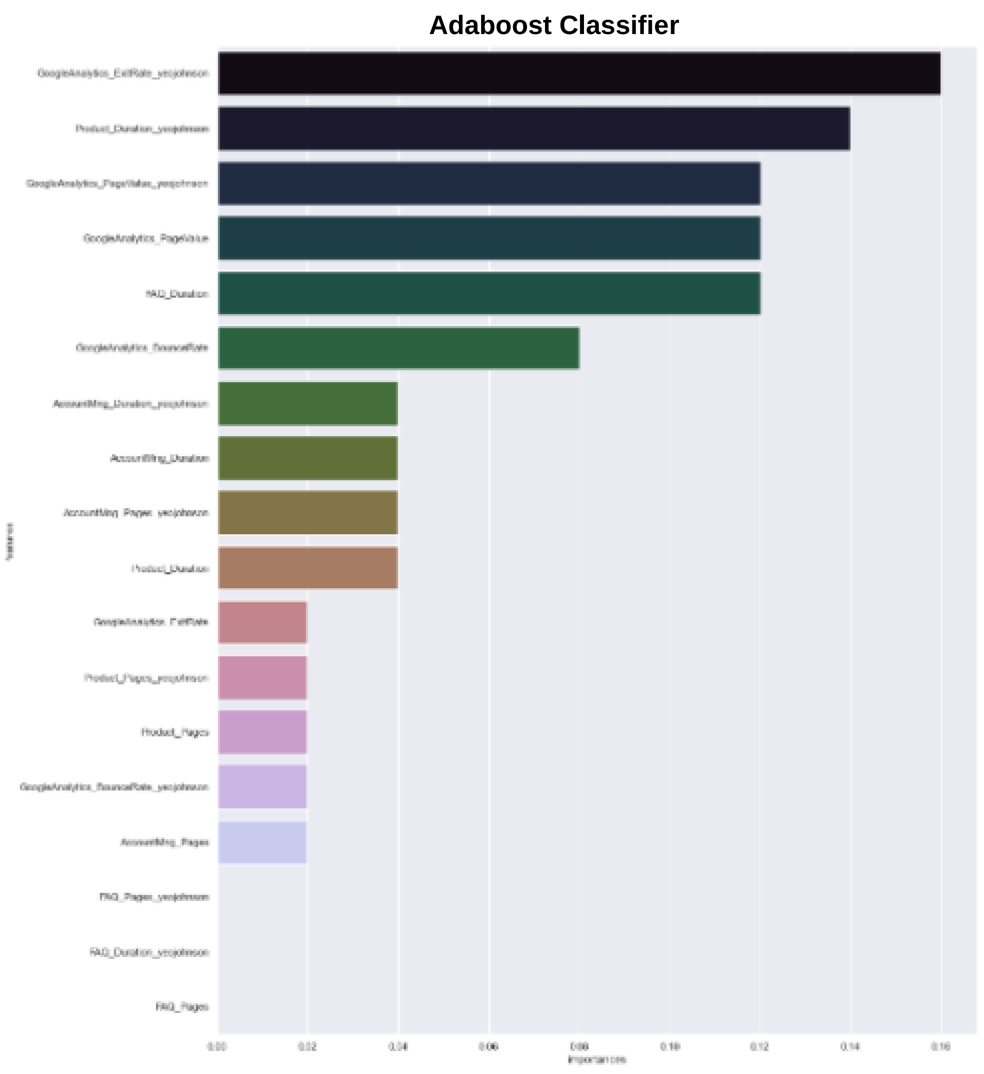


Figure 6: Feature selection technique plots

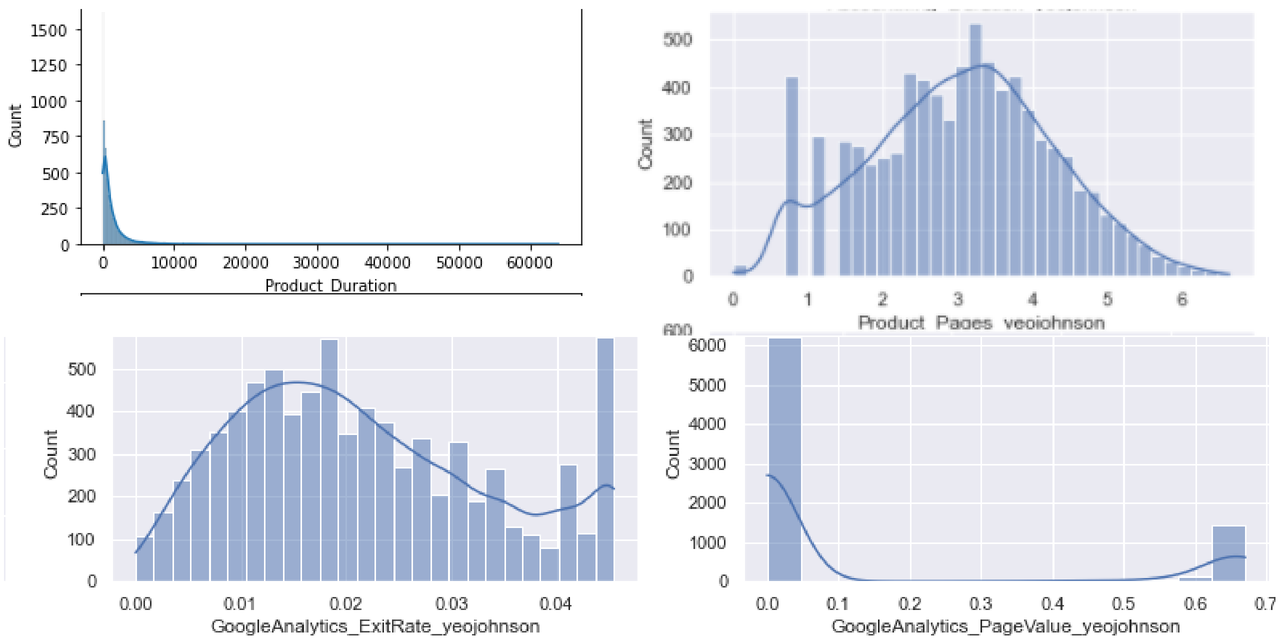


Figure 7: Final numerical variables picked to model

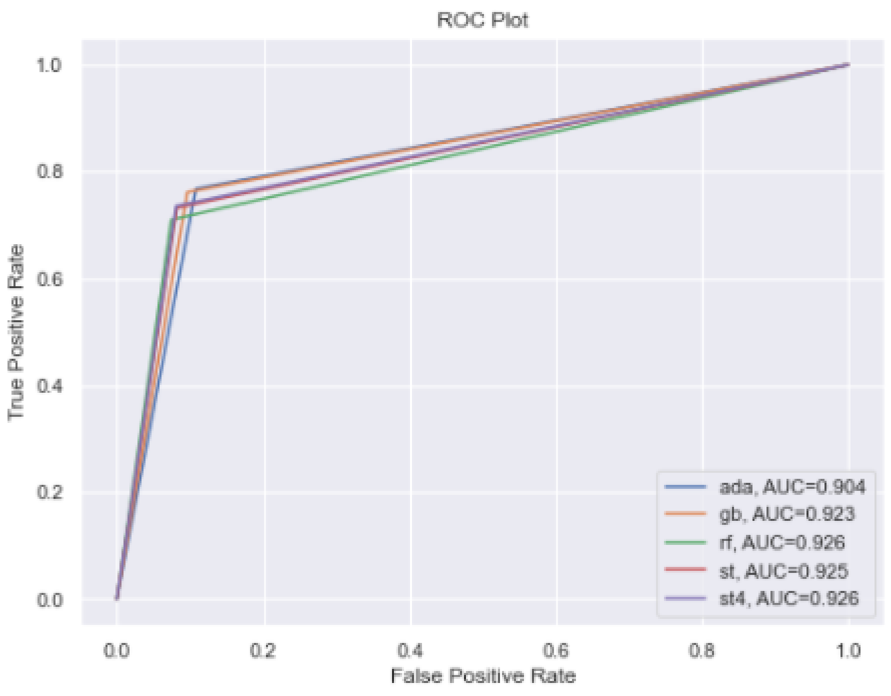


Figure 8: ROC Curve

Variables	Feature selection technique				
	Spearman	**RFE**	**Lasso Regularization**	**Random Forest (|value| >0.04)**	**Adaboost (|value| >0.03)**
AccountMng_Pages	Discard	Keep			
AccountMng_Duration	Discard		Not Keep		5
FAQ_Pages	Discard		Not Keep		
FAQ_Duration	Discard		Not Keep		3
Product_Pages	Discard	Keep	Not Keep	4	
Product_Duration	Discard	Keep		2	5
GoogleAnalytics_BounceRate	Discard		Not Keep		4
GoogleAnalytics_ExitRate	Discard	Keep		3	
GoogleAnalytics_PageValue	Discard	Keep		1	3
AccountMng_Pages_yeo	Discard		Not Keep		5
AccountMng_Duration_yeo	Discard		Not Keep		5
FAQ_Pages_yeo	Discard	Keep	Not Keep		
FAQ_Duration_yeo	Discard	Keep			
Product_Pages_yeo	Discard	Keep		4	
Product_Duration_yeo	Discard		Not Keep	2	2
GoogleAnalytics_BounceRate_yeo	Discard	Keep			
GoogleAnalytics_ExitRate_yeo	Discard	Keep		3	1
GoogleAnalytics_PageValue_yeo	Discard	Keep		1	3

Table 8: Feature Selection Numerical Independent Variables

F1_Score_Validation				
Algorithm	**SMOTE with RSK**		**Train_Test_Split with RSK**	
	Min_Max	**Robust**	**Min_Max**	**Robust**
Logistic Regression	0.64	0.64	0.65	0.65
KNN	0.61	0.62	0.65	0.56
Neural Network	0.65	0.66	0.64	0.64
Ridge Classifier	0.64	0.64	0.64	0.64
Passive Agressive	0.63	0.63	0.36	0.59
SVM	0.66	0.64	0.67	0.65

Table 9: F1 score for the algorithms that needed feature scaling

F1_Score_Validation		
Algorithm	**SMOTE with RSK**	**Train_Test_Split with RSK**
Decision Trees	0.64	0.64
Logistic Regression	0.64	0.65
Naive Bayes	0.63	0.64
KNN	0.62	0.65
Neural Network	0.66	0.64
SVM	0.66	**0.67**
Ridge Classifier	0.64	0.64
Passive Agressive	0.63	0.59
Adaboost	0.65	0.65
Gradient Boost	**0.67**	**0.67**
Random_Forest	0.65	**0.67**

Table 10: Best F1 score for all classification algorithms