MACHINE LEARNING – *TECHSCAPE'S* PREDICTIVE MODEL

António Cymbron (r20181059), **Gonçalo Silva** (r20181085), **Maria João Marques** (r20181119), **Pedro Sequeira** (r20181097)
MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS, WITH A SPECIALIZATION IN DATA SCIENCE
NOVA INFORMATION MANAGEMENT SCHOOL

ABSTRACT: During the emergence of COVID-19 in early 2020, *TechScape*, a start-up that sells goods related to digital detox, started to face financial difficulties. This Machine Learning study was made to analyze the online behaviour of their customers and accordingly predict the probability of buying their products. The goal was to create a predictive model which identifies the customers as one of the two classes: buyer or non-buyer. The data was treated and tested in several different ways – missing values, duplicates, outliers and incoherencies were handled, and features were selected, transformed and created, among others -, until reaching the final model made with Random Forest, which turned out to be the better performing algorithm for this specific problem, with a F1-Score of 0,69364, on Kaggle.

KEYWORDS: Machine Learning, Predictive Modelling, F1-Score, Classification Problem, Ensemble Algorithms, Supervised Learning, RandomForest.

1. INTRODUCTION

TechScape is a start-up company founded at the beginning of 2020 which sells digital detox related products through an online store. Due to COVID-19 emerging in March 2020, the start-up faced some financial difficulties. Now that they have restarted their activities, and are looking to increase their sales, they want to know which characteristics of the customers' online behaviour are most related to them buying (or not buying) a product.

The goal of this study is to build a predictive model that can accurately predict what type of customer will buy products from *TechScape*. The two given datasets – train, and test - are strictly composed of features that represent the actions of the customer on the website or features with information about the types of technologies they are using. The target variable is "Buy" and can either be 1 – the customer bought something –, or 0 – the customer did not buy anything. The training data, composed of 9999 observations, will be the dataset used to train the models and the test dataset (2300 observations) will be used to assess how well the model performs on unseen data. Knowing all this, we will be moved by the goal of understanding what are the most important variables to predict "Buy", and what is the best model to apply to this specific problem.

2. BACKGROUND

TechScape is owned by five Portuguese entrepreneurs and sells products and services which will allow their customers to stay focused on the most important things and improve the balance with technology use in their lives, such as meditation kits, books, stress balls, "dumb" phones, retreats, among others. Focused on increasing their sales and using the small quantity of data[1] accessible from the customers database that contains general information about the customers and their behaviour in the website from February 2020 till December 2020 (excluding

April), predictions will be made to know which customers have a high probability of buying their products depending on their online actions.

3. METHODOLOGY

i. Data Exploration:

When exploring the available data, we started by checking duplicates. Duplicates are occurrences that if not properly handled, may cause that, while training, the model learns from the same instances twice. Hence this may cause overfitting, since the model will overlearn for the instances that are duplicated more in comparison to the others in the dataset and, as consequence, the model might end up not generalising well. When looking for these, 14 duplicated records were revealed. Instead of deleting these observations outright, the 28 lines were analyzed and compared. It was concluded that it was reasonable to think that these data points, although the same in terms of values, could reasonably be different accesses to the website that coincidently were similar enough to not be differentiated and represent normal usage. Therefore, these 28 observations were kept.

Moving on to the unveiling of missing values that could be present in the datasets, after looking for them in its "typical form" – as *NaN* -, we checked if there could be some on several other types of representations ("", "?", "-", " ", "null", "NK"). However, we concluded that neither of the given datasets had missing values. The handling of missing data is very important during this phase as many machine learning algorithms do not support missing values, it reduces bias, and is one-step closer to produce powerful suitable models.

Another takeout was that we were in the presence of an imbalanced dataset. The training dataset had 9999 observations, 8447 of which had "0" on the target variable "Buy" and 1552 had "1". This big discrepancy between the number of zeros and ones in the dataset means that, most likely, the models will have a poor performance when learning more about the minority class. One approach to addressing imbalanced datasets is to oversample the minority class. This can be done by synthesizing new observations from existing ones - data augmentation. For this purpose, we used SMOTE - Synthetic Minority Oversampling Technique[2] to create new datasets with artificial observations, where a random sample from the minority class is chosen and *k* of the nearest neighbours for that example are found. A randomly selected neighbour is chosen, and a synthetic example is created at a randomly selected point between the two examples in the feature space. After applying SMOTE, we generated a dataset with the exact same amount of zeros and ones. Both the datasets, with and without SMOTE, were kept to test which one would be more effective to train the predictive models.

Moving on to the visualization of the data in hands, good exploratory data analysis combined with relevant data visualization is essential for pinpointing the right direction to take. It both shortens the machine learning process and provides more accuracy for its outcome. Therefore, we started by dividing the dataset into metric (numerical) and non-metric (categorical) features as they require different methods for visualization. For the first group, histograms were plotted to check distributions, skewness – which made us consider future *log(10)* transformations

on the variables -, and try to pre-emptively identify outliers, also resorting to the use of Box-Plots. By analyzing them, we concluded that most of the variables presented outliers. Afterwards, the relationships between features were checked through a pairwise relationship plot having the *Buy* variable in account. With these plots we concluded that there are some highly correlated pairs of features (*Product_Pages*/*Product_Duration, GoogleAnalytics_ExitRate* /*GoogleAnalytics_BounceRate*, for instance*)*. Due to this graph, some bivariate outliers were also identified. The variables were then plotted again in Box-Plot graphs, but this time aggregated by *Buy*, which helped conclude that outliers were also observable when taking into account the dependent variable. With regard to the categorical variables, these were plotted as absolute frequencies by *Buy*. The higher cardinalities observed were in the *Browser* and *Type_of_Traffic* variables.

ii. Data Pre-Processing:

Feature engineering and selection are the methods used for creating a dataset that is optimised to maximise the information density on our data. With this in mind, we started by transforming some variables[3].

To enable better processing of Categoric Variables we used One Hot Encoding[4]. This process began with the transformation of some variables into categorical ones. Next, by checking the distribution of all variables belonging to this group of variables, we understood possible groupings based on the frequencies of each class of each variable. Finally, applying One Hot Encoding to all categorical variables, the resulting dataset ended up with 52 features since this technique creates, for each affected variable, N-1 binary columns, where N is the cardinality of the categorical variable at stake.

Moving on to the next technique applied, partitioning was used in order to select a model for the data at hand from a broad set of models. The basic idea of data partitioning is to keep a subset of available data out of analysis, and to use it later for verification of the model. A good principle to apply is to split our dataset before processing the data. This is due to the fact that, if we don't do it, we may occur in data leakage (by transforming too much the data that will be predicted, the test data will become too similar to the data that will be used to train the model, which will result in overfitting and bad predictions of observations that the model has never seen before, according to Weiran, S. (2019)). Thus, at this stage of our project, we randomly split our training dataset into 2 datasets: the training dataset, and the validation dataset. Throughout our work, each of these two datasets served a specific purpose: the training dataset (corresponding to 70% of the observations) was used to train the models we were developing; on the other hand, the validation dataset (corresponding to 30% of the observations) was used to evaluate the performance of the models we were creating and to ensure that we were not overfitting them to our training data. It is important to note that, as we wanted to keep the same percentage of 0's and 1's (in the dependent variable) in both datasets, the stratification was performed with the variable *Buy* as target.

Next, incoherencies were unveiled. Once more, just like outliers, these are occurrences that must be handled since it is important that the model does not learn with instances that, many times, are not real. Several cases were checked, and for each that was present in our dataset, a decision was made. In most cases, records were corrected since deleting them would mean to destroy a high percentage of records, way above the maximum 3% suggested for these purposes. One of checked cases, was to see if there were pages visited of a certain topic, and no time devoted to that on that access. On these cases, the value assigned to pages was set to zero. The inverted scenario was also checked, and the same decision was made, but for the second's value. At this stage, some assumptions were made: One can directly go to the cart and buy any product in one access, without having visited a product page; the website may have more than the indicated group of pages, allowing an access to have 0 seconds in total assigned to it.

Regarding outliers, three approaches were considered – manual, automatic and IQR method. Priorly to any action, the graphs of the variables were once more analyzed, to check its distributions, to understand which records represented outliers, using Box-Plots and Histograms. Using the IQR method, or Interquartile Range, a lot more that 3% of observations would be removed, which isn't sustainable for the continuation of the study, since most of the dataset would be eliminated. Next, concerning the Automatic approach[5], we used IsolationForest, an anomaly detection algorithm, to help identify and remove outliers. However, even with the contamination parameter equal to 3(%), this method didn't show results as satisfactory as the Manual approach, that consisted in removing observations by checking graphs, feature by feature, in order to decide the best threshold for each one. Using this, about 0,5% and 0,6% of rows were deleted on the train dataset, without and with SMOTE, respectively.

Next, Standardization is an extremely important step to take when the data that one is working with presents different scales. For example, having data with different scales would result in huge problems when using models that take into account distances. With this in mind, two different standardization techniques were applied to the data. The first used technique was MinMaxScaler between 0 and 1. This technique was applied to the versions of data where the outliers had been removed. Thus, in the observations for these datasets, all values in non-binary numeric columns were now between 0 and 1. The second applied technique was RobustScaler, and it was applied to the versions of data where the outliers had not been removed. This was because RobustScaler is robust to outliers (it removes the median from the column being calculated and then scales the data using the IQR). After this, the number of datasets to work with duplicated. The decision on which of the two standardization methods was best was made later when the models were created, and their results were measured.

Finally, selecting the best features for the modelling phase was now possible. Firstly, we wanted to drop any numerical variables that were univariate (variance equal to zero). Then, we analyzed the correlations between variables, using the Spearman Correlation, that assesses how well the relationship between two variables can be described using a monotonic function (whether linear or not). Complementary to this, the Mutual Information Coefficient[6] between variables was also taken into account and consists in the amount of information that one gains about a random

variable by observing the value of the other. Being so, we were able to keep in mind what pairs of variables were highly correlated (>0,8), with the first technique, and to know which variables had high MIC values, which could possibly be kept. Exclusively to categorical variables – now binary features -, we also analyzed which features were important to explain the dependent one. Moving on to Wrapper Methods, we did a Recursive Feature Elimination – where, with a Logistic Regression as estimator, we had as return an optimal number of features to keep, and respective score. Still in this scope, we also tried Forward, Backward and Stepwise Selections[7], were we also got the features that each method considered best to keep improving the classification made by the future model. In the first scenario, new features are added with the aim to improve the score; on the second, having all features, some are eliminated with the same goal; and, finally, on the last one, features may be added and deleted interactively, also with the aim to achieve the best score possible. Last but not least, regarding embedded methods, we used Lasso Regression, were we dropped feature with the Lasso Coefficient of 0 – meaning they don't seem to have importance. Additionally, with Ridge Regression[8] we had a group of selected features that were also considered. Finally, Decision Trees for Feature Importance were also used. Here, Gini Coefficient and Entropy were employed to understand which features were most important to keep for a successful model. All these methods were employed for all candidate datasets, culminating on two final sets for each dataset, that represented the features that had a higher ranking, having, and not having, correlation among variables into account, respectively.

iii. Modelling:

Given the conclusions we drew from the feature selection and all the previous explained steps, we arrived at the moment of modelling with 8 possible combinations of datasets and features used (2 feature sets for each of the 4 datasets – a baseline dataset, a baseline dataset with SMOTE applied to it, a dataset with incoherencies and outliers handled, and a dataset with incoherencies and outliers handled and with SMOTE applied to it)[9] [10] [11] [12].

First, models that do not belong to the Ensemble models' category were tested. Thus, we began by testing the following models with several specifications: Logistic Regression, Multi-Layer Perceptron, Decision Tree, K-Nearest Neighbors, K-Nearest Centroids[13], Passive Aggressive Classifier[14], Gaussian Naive Bayes, Support Vector Machine, Linear Discriminant Analysis, and Quadratic Discriminant Analysis[15]. During the modelling process, for the different algorithms we experimented with and for the different parameter specifications of each of them, we tried not only to get the best F1-Scores, but also to have models with little or no overfitting because, if it existed, the predictions of that model for never-before-seen data (like Kaggle's) would not be the most correct. The F1-Score[16] is used in classification problems when one wants to seek a balance between Precision – that shows how accurate a model is by showing, out of those predicted positive, how many of them are actual positive. -, and Recall - that calculates how many of the actual positives a model is able to capture through labelling it as positive -, and if there is an uneven class distribution (large number of Actual Negatives), which was our case.

Next, in the hope of improving our results, we resorted to Ensemble techniques. These techniques generally produce better results, as they combine different simpler models, intending

to create more accurate models. Thus, in this phase, the following Ensemble techniques were applied with different specifications (and using the best models obtained previously): Bagging Classifier, Random Forest, AdaBoost Classifier, Voting Classifier[17], Gradient Boosting and Stacking Classifier. As with models that do not belong to the Ensemble category, for each of these techniques and the respective specifications, after creating the model and training it, we proceed to predict the dependent variable and to calculate the F1-Scores (both for the training and validation datasets), with the goal of finding a model without overfitting and with good results.

4. Results & Discussion

As previously mentioned, we went into modelling with eight different datasets. Every algorithm was tested with each of the datasets, and only the models with the best results were kept. We experimented with different parameters on almost every model to get the best performance possible, including using grid search to find the specific values that would get us the highest F1-Score. It is worth mentioning that all the models built used the parameter *random_state = 42* so that each time we ran the code the results would be the same. The non-ensemble algorithms mentioned before were the first ones to be tested. The average scores[1] for the train and validation datasets of the best models obtained can be analyzed in the annexes[18].

As we can observe, in terms of score, we seem to have good results with little (in some models) to almost no overfitting. SVM2 (Support Machine Vector model number 2) does have a considerably lower score compared to the other ones as shown in the graph but is still present on the best models' category since it did have one of the best F1-Scores of the non-ensemble algorithms (check table of F1-Scores in appendix for more information)[19].

The ensemble algorithms, also mentioned earlier, were only tested after the non-ensemble algorithms were complete. This is because most of them have the parameter base_estimator which is the parameter that determines the estimator used for the model, and we could use previously created models as estimators. Of the used ensemble algorithms, only Random Forest and Gradient Boosting can't have their estimator changed. We tested the other algorithms with all possible estimators, using grid search to find which one would give the best F1-Score, as well as the best parameters. The average scores of the models obtained can be analyzed on the annexes[20].

Once again, in terms of accuracy the models got good scores. Although these may seem good results, there are some factors to be considered. Some of these best models were obtained using SMOTE datasets. When checking F1-Scores for train and validation on these models, we found out that they overfitted. Thus, even if they presented a good F1-Score, we immediately discarded them as a potential candidate for the final model.

After carefully analysing the results obtained, we noticed that, as expected, ensemble algorithms obtained, on average, higher F1-Scores (both for the training dataset and the validation dataset) than algorithms that do not belong to this category. It is important to highlight the performance of two models, regarding the quality of the predictions made for the validation

[1] Accuracy, or the ratio of the number of correct predictions to the total number of input samples.

dataset: Random Forest (0.6778) and Gradient Boosting - Version 2 (0.6730). On Kaggle, these models scored 0.69364 and 0.68639, respectively. Both these models were trained on the dataset whose data had been standardised with the *MinMax* method and where outliers had been removed and inconsistencies treated. Because the original dataset was quite imbalanced, we expected that by applying SMOTE, the predictions would be more correct, as the model would not only be able to predict the 0s (the majority class), but also the 1s. However, contrary to expectations, the datasets where SMOTE was applied produced worse results, producing overfitting.

Regarding the performance of our models in Kaggle, we were able, in the vast majority of the final algorithms, to obtain results quite similar to the F1-Scores we obtained in the notebook for the validation data, which proves that we were able to generate models with little overfitting. However, we would like to highlight the K-Nearest Neighbours, because it obtained an excellent score in Kaggle, above 0,7. Nevertheless, and as it can be seen in the Figure 3 chart[21] and in the Table 7[22] – present in the annexes -, this model had overfitting problems, so it was discarded. Thus, from all the models we created, we chose Random Forest as our final model, since it was the one with the highest F1-Score, while presenting no overfitting problems. Observing the metrics in Figure 4[23], we can conclude that Random Forest is much better at predicting 0s than predicting 1s, which was not unexpected, due to the fact that it was trained with imbalanced data.

5. CONCLUSION

During the development of this project, we faced a complex set of problems that required Machine Learning knowledge acquired both through self-study and during classes. The process was not without its setbacks as we weren't satisfied with the results initially reached and had to rework most of the project to eliminate the problems identified.

We got to experience various Data Exploration, Pre-Processing and Modelling Methods, and reach the conclusion that what works best is specific for each data, and mostly a combination of several methods. The methodologies employed are very situation specific.

The fact that the dataset was so heavily imbalanced also presented some challenges that we thought would be overcome with an oversampling technique, which proved an incorrect assumption as the use of SMOTE led to generally higher levels of overfitting when training the models. Also, other limitations that we faced during this project and that we believe ended up limiting the results we obtained, were related to the fact that some of our choices were limited by the lack of computing power (for instance, we intended to try one of our models created with Support Vector Machines in AdaBoost, but this was not possible). Another limitation we would like to mention is that, after carefully analyzing the data, we concluded that we had a huge number of possible outliers and that, in order not to exceed the recommended limit of 3% of removed data, we ended up treating less data than the ones that really seemed to escape the so-called normal distribution of the data.

The models with the best results were Ensemble algorithms that combine several models (weak-learners) into a single one. This reenforces the hypothesis that in machine learning the best approach is a case specific combination of different methods that build upon each other.

With that said, the group concluded that the best method to predict if a customer will make a purchase for this specific Dataset is a Random Forest with a prediction score of 0,69 on Kaggle.

6. REFERENCES

Analytics Vidhya. (2020, October 24). *Feature Selection using Wrapper Method - Python Implementation*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/

Banerjee, P. (2019). *AdaBoost Classifier Tutorial*. Kaggle.com. https://www.kaggle.com/prashant111/adaboost-classifier-tutorial

Brownlee, J. (2016, April 5). *Linear Discriminant Analysis for Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learning/

Brownlee, J. (2020a, January 16). *SMOTE for Imbalanced Classification with Python*. Machine Learning Mastery. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Brownlee, J. (2020b, April 9). *Stacking Ensemble Machine Learning With Python*. Machine Learning Mastery. https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/

Brownlee, J. (2020c, April 30). *How to Develop an AdaBoost Ensemble in Python*. Machine Learning Mastery. https://machinelearningmastery.com/adaboost-ensemble-in-python/

Brownlee, J. (2020d, June 21). *How to Avoid Data Leakage When Performing Data Preparation*. Machine Learning Mastery. https://machinelearningmastery.com/data-preparation-without-data-leakage/

Brownlee, J. (2020e, October 13). *Nearest Shrunken Centroids With Python*. Machine Learning Mastery. https://machinelearningmastery.com/nearest-shrunken-centroids-with-python/

Bunce, D. (2020, March 28). *Correlation vs Mutual Information*. Mathemafrica. http://www.mathemafrica.org/?p=16127

Clarke, M. (2021, March 14). *How to use the Isolation Forest model for outlier detection*. Practicaldatascience.co.uk. https://practicaldatascience.co.uk/machine-learning/how-to-use-the-isolation-forest-model-for-outlier-detection

Crammer, K., Keshet, J., Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2006). Online Passive-Aggressive Algorithms Ofer Dekel Shai Shalev-Shwartz. Journal of Machine Learning Research, 7, 551–585. https://jmlr.csail.mit.edu/papers/volume7/crammer06a/crammer06a.pdf

Goyal, C. (2021, April 27). *Missing Values | Treat Missing Values in Categorical Variables*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/04/how-to-handle-missing-values-of-categorical-variables/

Jing, H. (2020, November 27). *Why Linear Regression is not suitable for Binary Classification*. Medium. https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

Kumar, S. (2020, August 2). *7 Ways to Handle Missing Values in Machine Learning*. Medium. https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e

Kumarappan, S. (2020, August 16). *Feature Selection by Lasso and Ridge Regression-Python Code Examples*. Medium. https://medium.com/@sabarirajan.kumarappan/feature-selection-by-lasso-and-ridge-regression-python-code-examples-1e8ab451b94b

Lutins, E. (2017, August 2). *Ensemble Methods in Machine Learning: What are They and Why Use Them?* Towards Data Science; Towards Data Science. https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

Patrick, H. (2019, February 19). *The Importance of Feature Engineering and Selection*. Rittman Mead. https://www.rittmanmead.com/blog/2019/02/the-importance-of-feature-engineering-and-selection/

Python Examples of sklearn.ensemble.BaggingClassifier. (n.d.). Www.programcreek.com. Retrieved December 24, 2021, from https://www.programcreek.com/python/example/86713/sklearn.ensemble.BaggingClassifier

scikit-learn developers. (n.d.). *sklearn.ensemble.IsolationForest — scikit-learn 0.23.1 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Shaikh, R. (2018, November 9). *Choosing the right Encoding method-Label vs OneHot Encoder*. Medium. https://towardsdatascience.com/choosing-the-right-encoding-method-label-vs-onehot-encoder-a4434493149b

Wei, H. (2019, August 27). *Feature Selection Methods with Code Examples*. Analytics Vidhya. https://medium.com/analytics-vidhya/feature-selection-methods-with-code-examples-a78439477cd4

Weiran, S. (2019, October 22). *Avoid Data Leakage — Split Your Data Before Processing*. Medium. https://towardsdatascience.com/avoid-data-leakage-split-your-data-before-processing-a7f172632b00

Wu, J. (2019, October 31). *Why Data Visualization is Essential in Every Step of ML*. Www.welcometothejungle.com. https://www.welcometothejungle.com/en/articles/btc-data-visualization-machine-learning

Yıldırım, S. (2020, April 29). *Data Leakage in Machine Learning*. Medium. https://towardsdatascience.com/data-leakage-in-machine-learning-6161c167e8ba

Zhu, A. (2021, June 29). *Select Features for Machine Learning Model with Mutual Information*. Medium. https://towardsdatascience.com/select-features-for-machine-learning-model-with-mutual-information-534fe387d5c8

APPENDIX

1. IMAGES, TABLES & FORMULAS

Table 1 *- Techscape available attributes and respective description*

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 by the user on FAQ pages
Product_Pages	Number of pages visited by the user about products and services offered by the company
Product_Duration	Total amount of time (seconds) spent by the user on products and services related pages
GoogleAnalytics_BounceRate	Average bounce rate value of the pages visited by the user, provided by Google Analytics
GoogleAnalytics_ExitRate	Average exit rate value of the pages visited by the user, provided by Google Analytics
GoogleAnalytics_PageValue	Average page value of the pages visited by the user, provided by Google Analytics
OS	Operating System of the 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 in the website with a transaction (only available in the train dataset)

Table 2 *– Transformation, Elimination and Creation of New Variables*

Name	Description
VARIABLES TRANSFORMED	
Date	*Date*’s type was changed to datetime
Browser	*Browser*’s type was changed in order to be a categorical variable
Type_Of_Traffic	*Type_of_Traffic*’s type was changed in order to be a categorical variable
NEW VARIABLES	
Month	Month was extracted from the *Date* variable
Weekend	Date was filtered and a “1” was assigned if the access was during a weekend, and “0” otherwise
Total_Duration	The sum of all 3 *X_Duration* variables gave us the total time spent in a certain access
Total_Pages	The sum of all 3 *X_Pages* variables gave us the total number of pages visited in a certain access
lg10_AccountMng_Duration	*log(10)* transformation of the variable *AccountMng_Duration*, due to its skewness
lg10_FAQ_Duration	*log(10)* transformation of the variable *FAQ_Duration*, due to its skewness
lg10_Product_Duration	*log(10)* transformation of the variable *Product_Duration*, due to its skewness
lg10_GoogleAnalytics_BounceRate	*log(10)* transformation of the variable *GoogleAnalytics_BounceRate*, due to its skewness
lg10_GoogleAnalytics_ExitRate	*log(10)* transformation of the variable *GoogleAnalytics_ExitRate*, due to its skewness
lg10_GoogleAnalytics_PageValue	*log(10)* transformation of the variable *GoogleAnalytics_PageValue*, due to its skewness
lg10_Total_Duration	*log(10)* transformation of the variable *Total_Duration*, due to its skewness

VARIABLES DROPPED	
Date	Since it would be redundant to have this variable after some of the previous transformations, *Date* was dropped

Equation 1 *- F1-Score Equation*

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Table 3 *– Feature Selection for Dataset 1 - Baseline*

	UNBALANCED TRAIN COM OUTLIERS E INCOERÊNCIAS = BASELINE										
	Lasso	Ridge	RFE	Decision Tree	MIC	Forward Selection	Backward Selection	Stepwise Selection	Weights	FINAL DECISION I (Óbvias) = SET 1	FINAL DECISION I (Óbvias s/Corr) -> Retiradas com base na Decision Tree = SET 2
AccountMng_Pages	X			X	X						
AccountMng_Duration	X			X	X						
FAQ_Pages					X						
FAQ_Duration	X				X						
Product_Pages	X		X	X	X	X	X	X		X	
Product_Duration	X			X	X		X				
GoogleAnalytics_BounceRate				X	X						
GoogleAnalytics_ExitRate	X	X	X	X	X						
GoogleAnalytics_PageValue	X			X	X	X	X	X		X	X
Month	X	X	X	X	X	X	X	X		X	X
Weekend											
Total_Duration				X	X		X				
Total_Pages			X	X	X	X	X	X		X	X
x0_France											
x0_Germany											
x0_Italy											
x0_Other											
x0_Portugal											
x0_Spain											
x0_Switzerland											
x0_UnitedKingdom											
x1_MacOSX					X				X		
x1_Other		X									
x1_Windows					X						
x2_2											
x2_4											
x2_5		X									
x2_Other											
x3_10		X	X			X	X	X			
x3_11		X	X			X	X	X			
x3_12		X	X								
x3_13					X				X		
x3_14											
x3_15		X							X		
x3_2	X	X			X	X	X	X	X	X	X
x3_3					X				X		
x3_4											
x3_5		X	X			X		X	X		
x3_6											
x3_7											
x3_8		X	X		X	X	X	X	X	X	X
x3_9											
x4_Other											
x4_Returner	X	X			X	X	X	X	X	X	X
log10_AccountMng_Duration				X	X						
lg10_FAQ_Duration					X						
lg10_Product_Duration	X	X	X	X	X		X			X	
lg10_GoogleAnalytics_BounceRate				X	X						
lg10_GoogleAnalytics_ExitRate	X			X	X	X	X	X		X	X
lg10_GoogleAnalytics_PageValue	X	X	X	X	X	X	X	X		X	
lg10_Total_Duration		X	X	X	X		X				
TOTAL	13	15	12	16	26	12	14	12	8	10	7

Table 4 *– Feature Selection for Dataset 2 – Baseline with SMOTE*

	BALANCED TRAIN COM OUTLIERS E INCOERÊNCIAS = BASELINE_SM										
	Lasso	Ridge	RFE	Decision Tree	MIC	Forward Selection	Backward Selection	Stepwise Selection	Weights	FINAL DECISION I (Óbvias)	FINAL DECISION I (Óbvias s/Corr) -> Retiradas com base na Decision Tree
AccountMng_Pages	X	X		X	X	X	X	X		X	
AccountMng_Duration	X			X	X	X	X	X		X	X
FAQ_Pages	X				X	X	X	X			
FAQ_Duration	X				X	X	X	X			
Product_Pages		X		X	X	X	X	X		X	
Product_Duration	X		X	X	X	X	X	X		X	
GoogleAnalytics_BounceRate				X	X	X	X	X			
GoogleAnalytics_ExitRate	X			X	X	X	X	X		X	X
GoogleAnalytics_PageValue	X			X	X	X	X	X		X	X
Month	X	X		X	X	X	X	X		X	X
Weekend	X	X			X	X	X	X			
Total_Duration			X		X						
Total_Pages	X	X		X	X	X	X	X		X	
x0_France					X		X				
x0_Germany					X						
x0_Italy					X		X				
x0_Other					X		X				
x0_Portugal				X	X		X				
x0_Spain					X		X				
x0_Switzerland					X						
x0_UnitedKingdom					X	X		X			
x1_MacOSX				X	X						
x1_Other					X	X	X	X			
x1_Windows					X						
x2_2					X						
x2_4					X						
x2_5					X						
x2_Other					X						
x3_10					X	X	X	X			
x3_11		X			X	X	X	X			
x3_12		X			X						
x3_13					X	X	X	X			
x3_14					X						
x3_15		X			X	X	X	X			
x3_2	X			X	X	X	X	X			
x3_3					X						
x3_4					X						
x3_5					X	X	X	X			
x3_6					X						
x3_7					X						
x3_8	X	X			X	X	X	X			
x3_9					X						
x4_Other					X						
x4_Returner	X			X	X	X	X	X			
log10_AccountMng_Duration	X			X	X	X	X	X		X	
lg10_FAQ_Duration	X				X						
lg10_Product_Duration	X	X	X	X	X	X	X	X		X	X
lg10_GoogleAnalytics_BounceRate	X			X	X	X	X	X		X	X
lg10_GoogleAnalytics_ExitRate				X	X						
lg10_GoogleAnalytics_PageValue	X	X	X		X	X	X	X		X	
lg10_Total_Duration		X	X	X	X	X	X	X		X	
TOTAL	18	12	5	18	51	27	31	27	0	13	6

Table 5 *– Feature Selection for Dataset 3 – Outliers and Incoherencies Removed*

	UNBALANCED TRAIN SEM OUTLIERS NEM INCOERÊNCIAS = OUT+CC										
	Lasso	Ridge	RFE	Decision Tree	MIC	Forward Selection	Backward Selection	Stepwise Selection	Weights	FINAL DECISION I (Óbvias)	FINAL DECISION I (Óbvias s/Corr) -> Retiradas com base na Decision Tree
AccountMng_Pages	X	X	X	X	X	X	X	X		X	X
AccountMng_Duration	X			X	X						
FAQ_Pages	X				X						
FAQ_Duration	X				X						
Product_Pages	X		X	X	X						
Product_Duration	X	X	X	X	X	X	X	X		X	
GoogleAnalytics_BounceRate	X	X	X	X	X						
GoogleAnalytics_ExitRate	X	X	X	X	X						
GoogleAnalytics_PageValue	X		X	X	X	X	X	X		X	X
Month	X	X	X	X	X	X	X	X		X	X
Weekend	X										
Total_Duration		X	X	X	X	X	X	X		X	
Total_Pages			X	X	X						
x0_France	X										
x0_Germany	X										
x0_Italy	X										
x0_Other	X										
x0_Portugal	X										
x0_Spain	X										
x0_Switzerland											
x0_UnitedKingdom	X										
x1_MacOSX	X				X				X		
x1_Other	X										
x1_Windows	X				X						
x2_2	X										
x2_4	X										
x2_5	X					X	X	X			
x2_Other	X										
x3_10	X					X	X	X			
x3_11	X	X	X			X	X	X			
x3_12			X								
x3_13	X				X				X		
x3_14											
x3_15	X								X		
x3_2	X				X	X	X	X	X		
x3_3	X				X				X		
x3_4	X										
x3_5	X								X		
x3_6											
x3_7											
x3_8	X	X	X		X	X	X	X	X	X	X
x3_9											
x4_Other	X		X								
x4_Returner	X		X		X	X	X	X	X		
log10_AccountMng_Duration	X			X	X						
lg10_FAQ_Duration	X				X						
lg10_Product_Duration	X	X	X	X	X	X	X	X		X	X
lg10_GoogleAnalytics_BounceRate	X			X	X						
lg10_GoogleAnalytics_ExitRate	X	X	X	X	X	X	X	X		X	X
lg10_GoogleAnalytics_PageValue	X	X	X		X	X	X	X		X	
lg10_Total_Duration	X	X	X	X	X	X	X	X		X	
TOTAL	43	12	18	15	26	15	15	15	8	10	6

Table 6 *– Feature Selection for Dataset 4 – Outliers and Incoherencies Removed with SMOTE*

	BALANCED TRAIN SEM OUTLIERS E INCOERÊNCIAS = OUT+CC_SM										
	Lasso	Ridge	RFE	Decision Tree	MIC	Forward Selection	Backward Selection	Stepwise Selection	Weights	FINAL DECISION I (Óbvias)	FINAL DECISION I (Óbvias s/Corr) -> Retiradas com base na Decision Tree
AccountMng_Pages	X	X	X	X	X	X	X	X		X	X
AccountMng_Duration	X	X		X	X		X				
FAQ_Pages	X	X	X		X	X	X	X		X	X
FAQ_Duration	X	X			X		X				
Product_Pages				X	X	X	X	X			
Product_Duration	X	X	X	X	X	X	X	X		X	X
GoogleAnalytics_BounceRate	X			X	X		X				
GoogleAnalytics_ExitRate	X	X	X	X	X	X	X	X		X	X
GoogleAnalytics_PageValue	X	X	X	X	X	X	X	X		X	X
Month	X	X	X	X	X	X	X	X		X	X
Weekend	X	X			X	X	X	X			
Total_Duration		X	X	X	X	X	X				
Total_Pages	X	X	X	X	X	X	X	X		X	
x0_France	X				X						
x0_Germany	X				X						
x0_Italy	X				X						
x0_Other	X				X						
x0_Portugal	X			X	X						
x0_Spain	X				X						
x0_Switzerland	X				X						
x0_UnitedKingdom	X				X	X	X	X			
x1_MacOSX				X	X						
x1_Other	X				X	X	X	X			
x1_Windows	X			X	X						
x2_2	X				X	X	X	X			
x2_4	X				X						
x2_5	X				X						
x2_Other	X				X						
x3_10	X				X	X	X	X			
x3_11	X				X	X	X	X			
x3_12	X				X						
x3_13	X				X	X	X	X			
x3_14	X				X						
x3_15	X				X	X	X	X			
x3_2	X			X	X	X	X	X			
x3_3	X				X						
x3_4	X				X						
x3_5	X				X	X	X	X			
x3_6	X				X						
x3_7					X						
x3_8	X	X	X		X	X	X	X			
x3_9	X				X						
x4_Other	X				X						
x4_Returner	X			X	X	X	X	X			
log10_AccountMng_Duration	X			X	X	X		X			
lg10_FAQ_Duration	X				X	X					
lg10_Product_Duration	X	X	X	X	X	X	X	X		X	
lg10_GoogleAnalytics_BounceRate	X			X	X	X	X				
lg10_GoogleAnalytics_ExitRate	X			X	X	X		X			
lg10_GoogleAnalytics_PageValue	X	X	X		X	X	X	X		X	
lg10_Total_Duration	X	X	X	X	X	X	X	X		X	
TOTAL	47	15	12	20	51	28	28	24	0	10	6

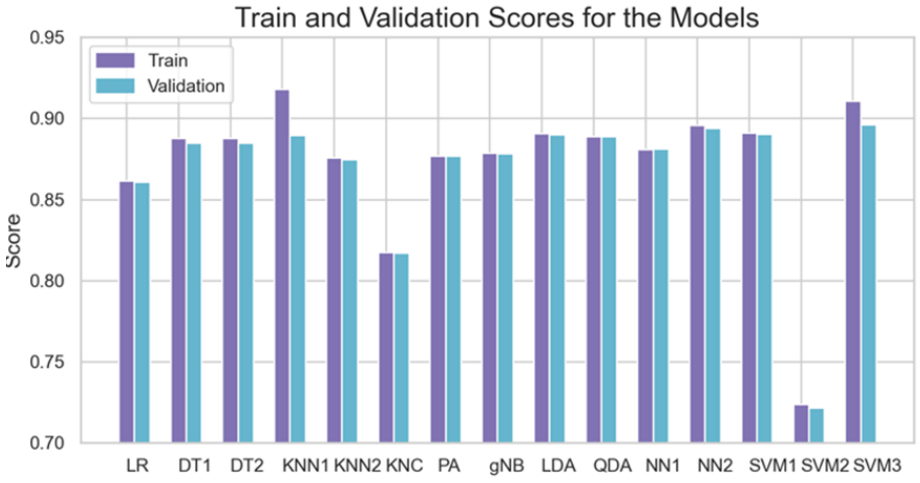

Figure 1 *- Train and Validation Scores for Non-Ensemble Algorithms*

Table 7 *– F1-Scores for Train, Validation and Kaggle by Algorithm*

Algorithm	Dataset + Set of variables used	F1 Score Train	F1 Score Validation	F1 Score Kaggle
Logistic Regression	Outliers + CC + SM Set 1	0,8531	0,6627	0,66666
Multi-Layer Perceptron/Neural Network version 1	Baseline 1 Set 1	0,6583	0,6597	0,66382
Multi-Layer Perceptron/Neural Network version 2	Outliers + CC + SM Set 1	0,9046	0,6636	0,65454
Decision Tree version 1	Baseline Set 1	0,6651	0,6595	0,61818
Decision Tree version 2	Baseline Set 1	0,6651	0,6595	0,62275
K-Nearest Neighbors version 1	Outliers + CC + SM Set 1	0,8805	0,6492	0,70046
K-Nearest Neighbors version 2	Outliers + CC Set 1	0,7150	0,6402	0,59649
K-Nearest Centroids	Outliers + CC + SM Set 1	0,8209	0,6096	0,632
Passive Aggressive Classifier	Baseline Set 1	0,6473	0,6465	0,65656
GaussianNB	Baseline Set 1	0,6112	0,6125	0,67336
Support Vector Machines version 1	Baseline Set 1	0,6506	0,6459	0,68
Support Vector Machines version 2	Baseline + SM Set 2	0,8184	0,6584	0,66382
Support Vector Machines version 3	Outliers + CC Set 1	0,6873	0,6414	0,64804
Linear Discriminant Analysis	Baseline Set 1	0,6516	0,6532	0,67
Quadratic Discriminant Analysis	Baseline Set 1	0,6410	0,6413	0,65284
Bagging	Outliers + CC + SM Set 1	0,8414	0,6694	0,67
Random Forest	Outliers + CC Set 2	0,7423	0,6778	0,69364
AdaBoost	Outliers + CC Set 1	0,6636	0,6607	0,66666
Voting classifier version 1	Baseline Set 1	0,6516	0,6546	0,67
Voting classifier version 2	Outliers + CC + SM Set 1	0,8671	0,6568	0,69683
Gradient Boosting version 1	Outliers + CC Set 1	0,7364	0,6667	0,6506
Gradient Boosting version 2	Outliers + CC Set 2	0,7337	0,6730	0,68639
Gradient Boosting version 3	Outliers + CC + SM Set 1	0,9999	0,6698	0,69811
Stacking version 1	Baseline Set 1	0,6469	0,6462	0,65517
Stacking version 2	Outliers + CC Set 1	0,7115	0,6382	0,65822

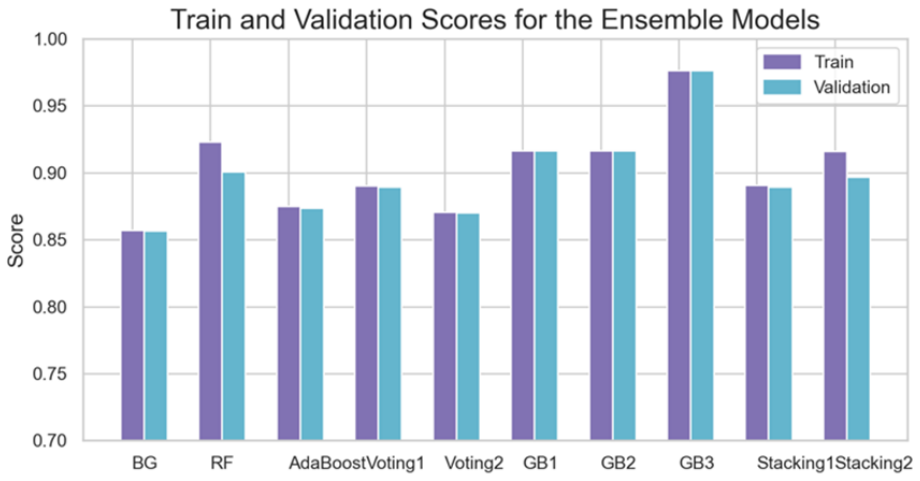

Figure 2 *– Train and Validation Score for Ensemble Algorithms*

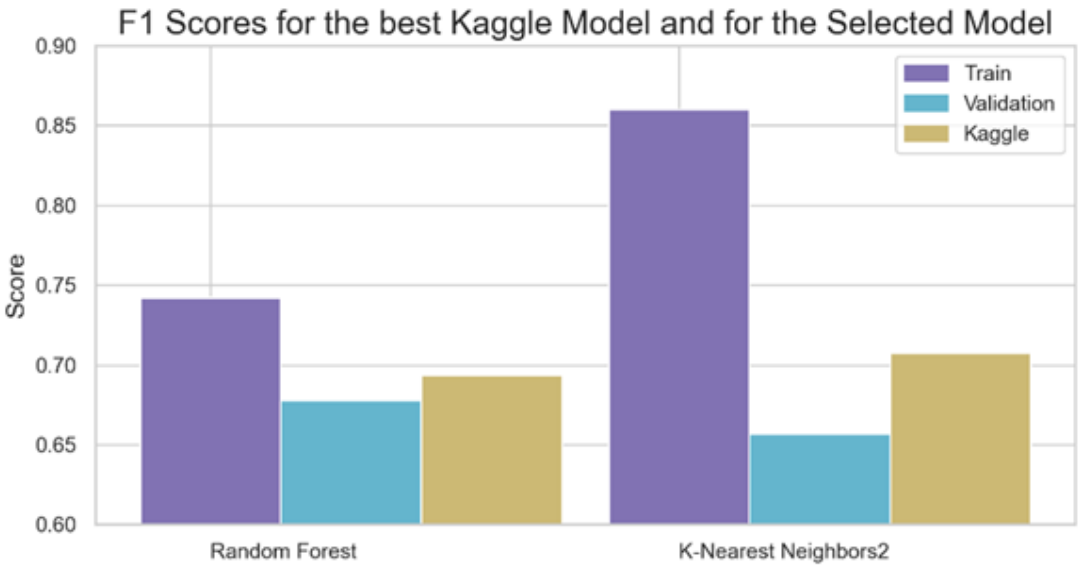

Figure 3 *– F1-Scores for the best Kaggle Model (KNN) and for the Selected Model (RandomForest)*

TRAIN

	precision	recall	f1-score	support
0	0.95	0.97	0.96	5893
1	0.80	0.69	0.74	1071
accuracy			0.93	6964
macro avg	0.87	0.83	0.85	6964
weighted avg	0.92	0.93	0.92	6964

```
[[5712  181]
 [ 332  739]]
```

VALIDATION

	precision	recall	f1-score	support
0	0.94	0.93	0.94	2534
1	0.66	0.70	0.68	466
accuracy			0.90	3000
macro avg	0.80	0.81	0.81	3000
weighted avg	0.90	0.90	0.90	3000

```
[[2368  166]
 [ 142  324]]
```

Figure 4 *– Metric output, with Cross-Validation Information (the matrixes below stand for the FN, FP, TN, TP)*

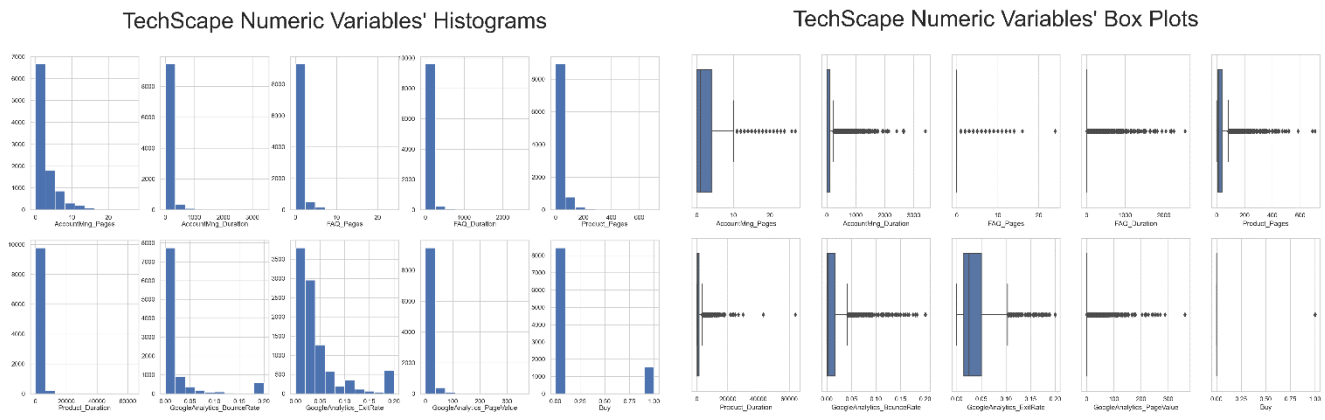

Figure 5 *– Other Jupyter Notebook Outputs: Data Visualization – Data Skewness and Possible Outliers*

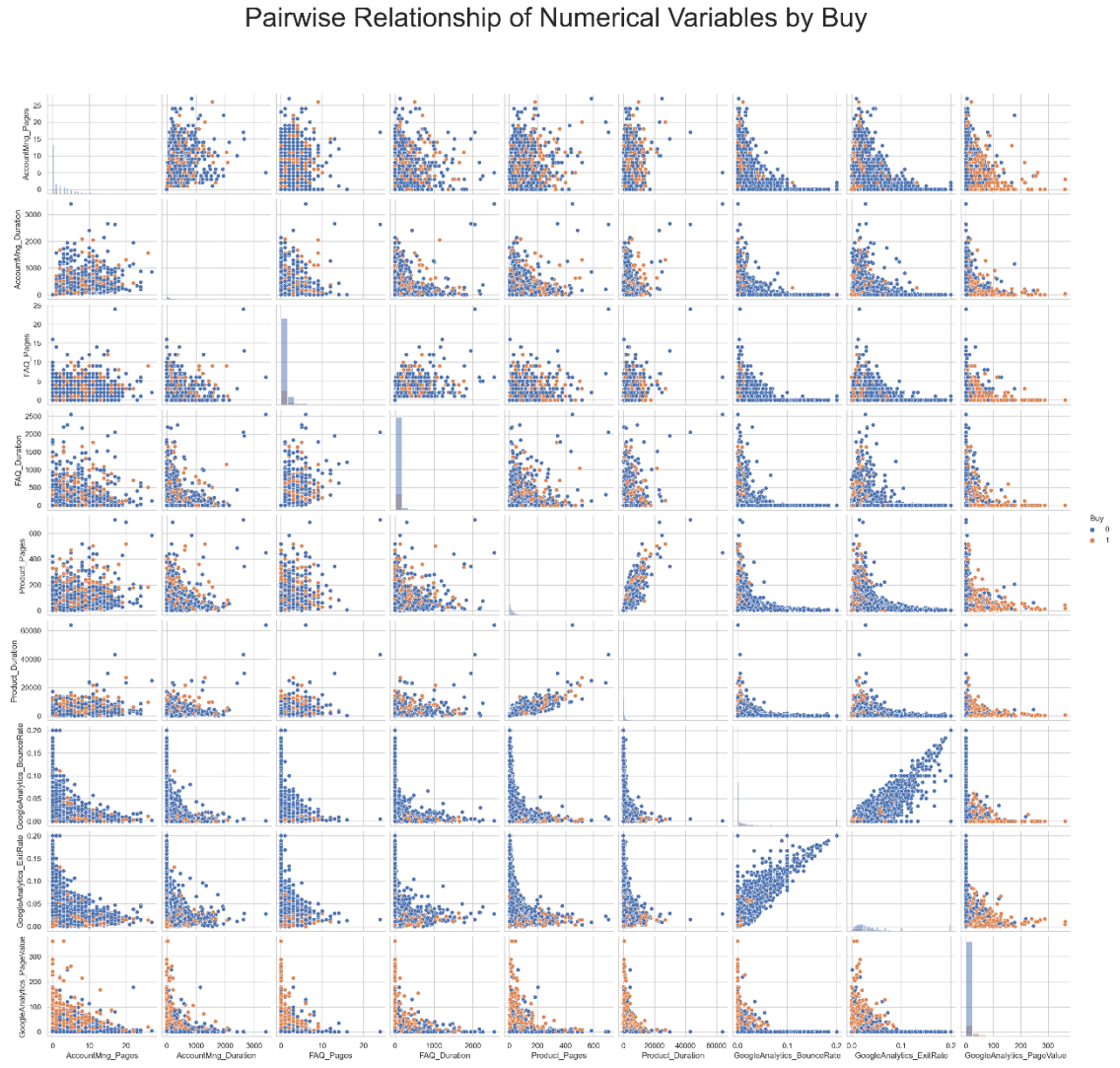

Figure 6 *– Other Jupyter Notebook Outputs: Data Visualization – Relations amongst Variables and Possible Bivariate Outliers*

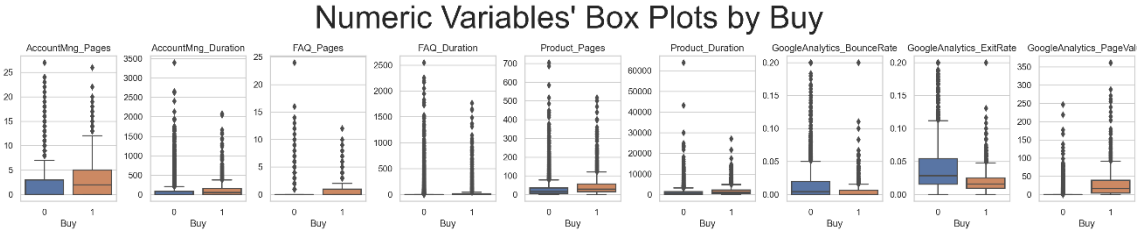

Figure 7 *– Other Jupyter Notebook Outputs: Data Visualization – Possible Outliers when Looking at Data by the Dependent Variable*

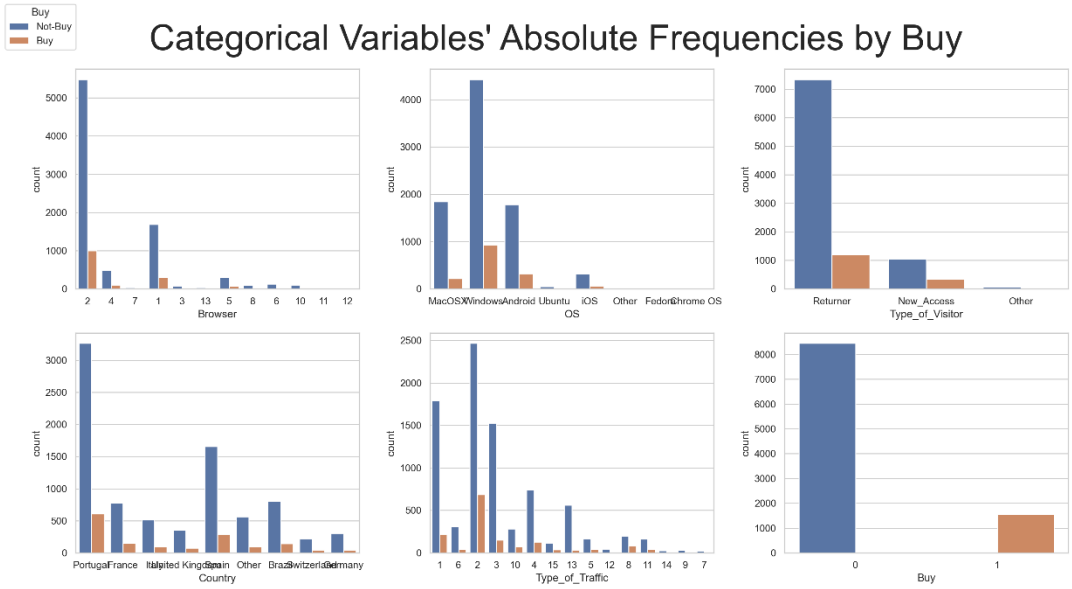

Figure 8 *– Other Jupyter Notebook Outputs: Data Visualization – Exploration of Categorical Variables and its Cardinalities, and Possible Outliers*

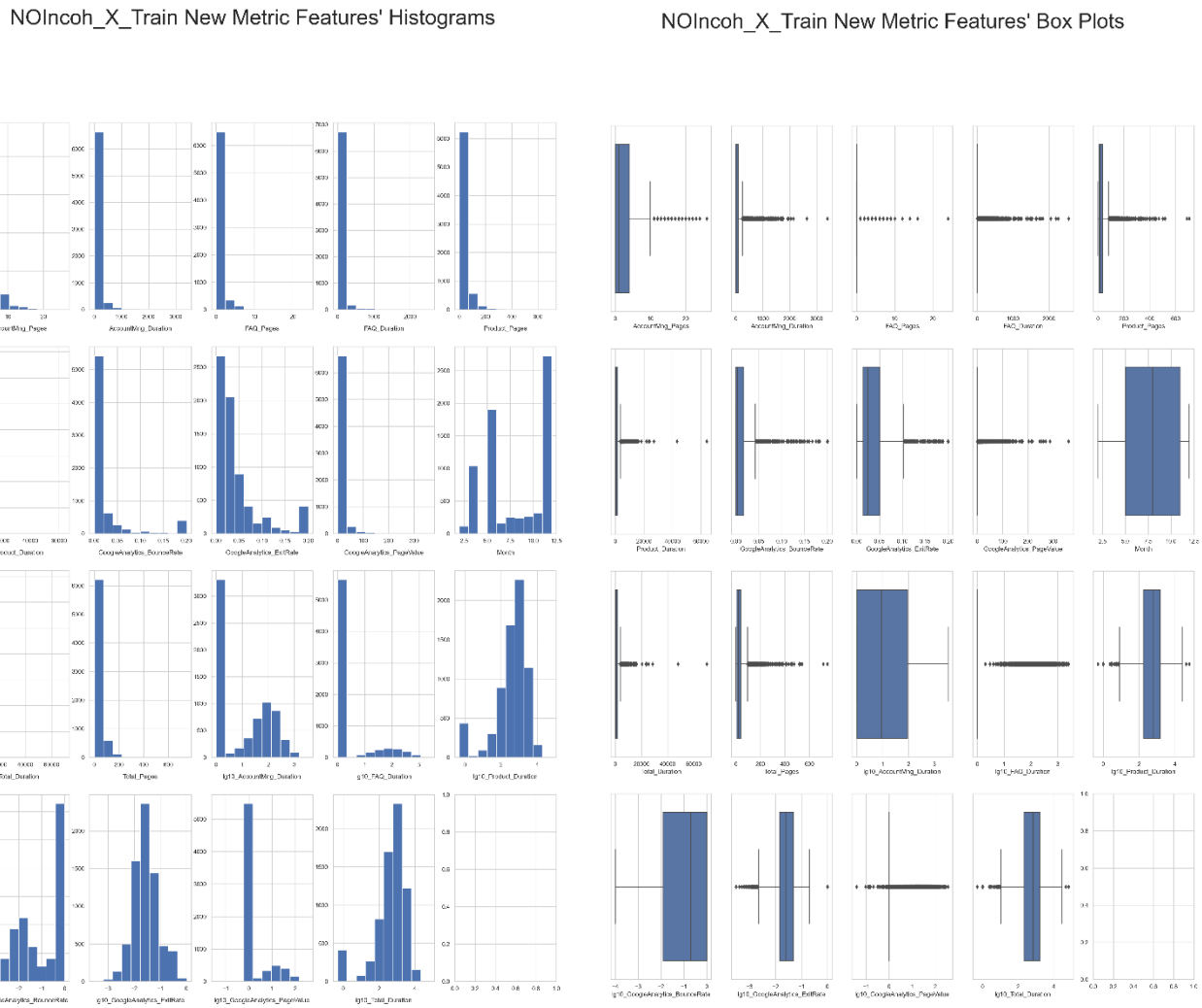

Figure 9 *– Other Jupyter Notebook Outputs: Data Processing – Exploration of Features in the Dataset for Outliers Removal*

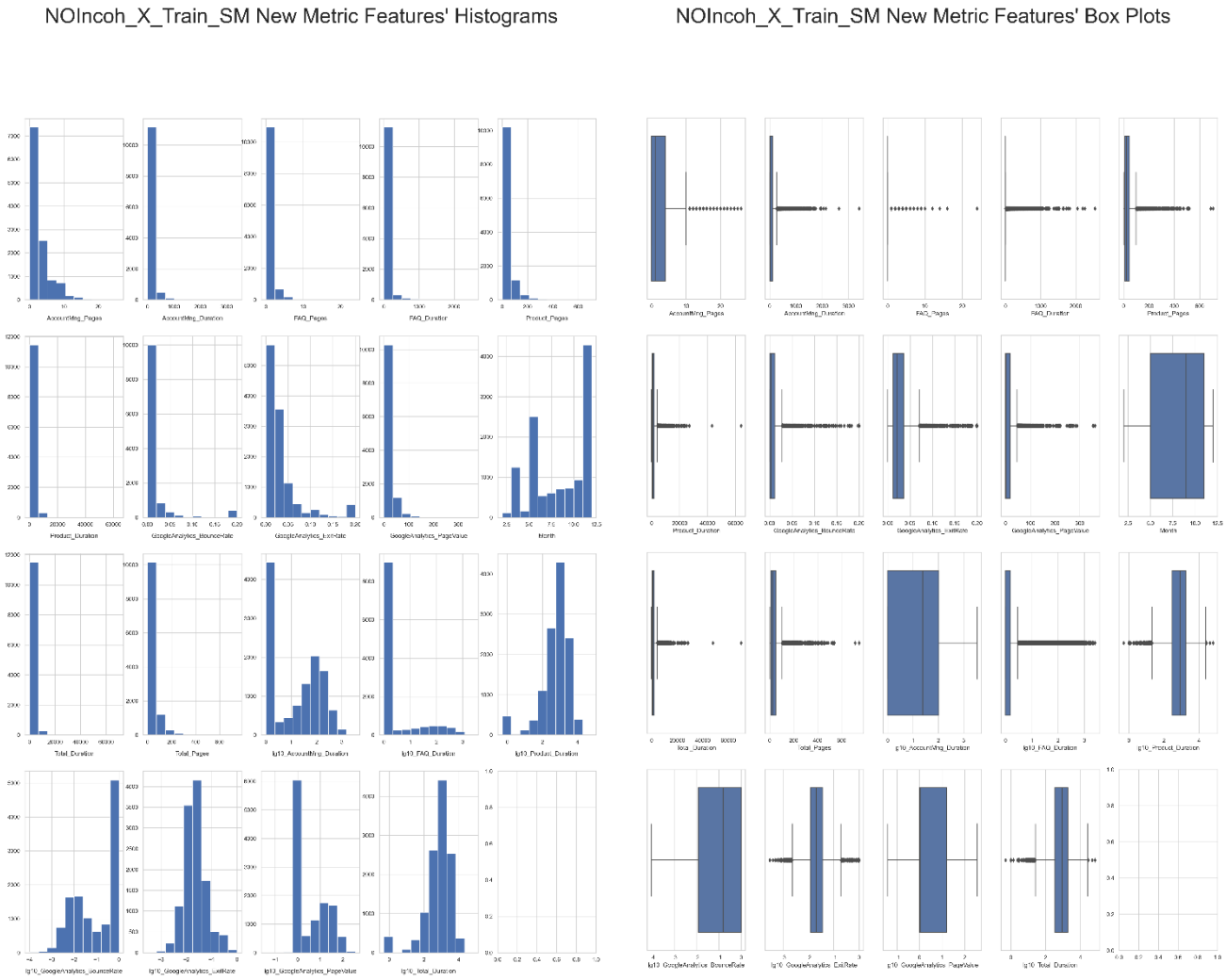

Figure 10 *– Other Jupyter Notebook Outputs: Data Processing – Exploration of Features in the Dataset with SMOTE for Outliers Removal*

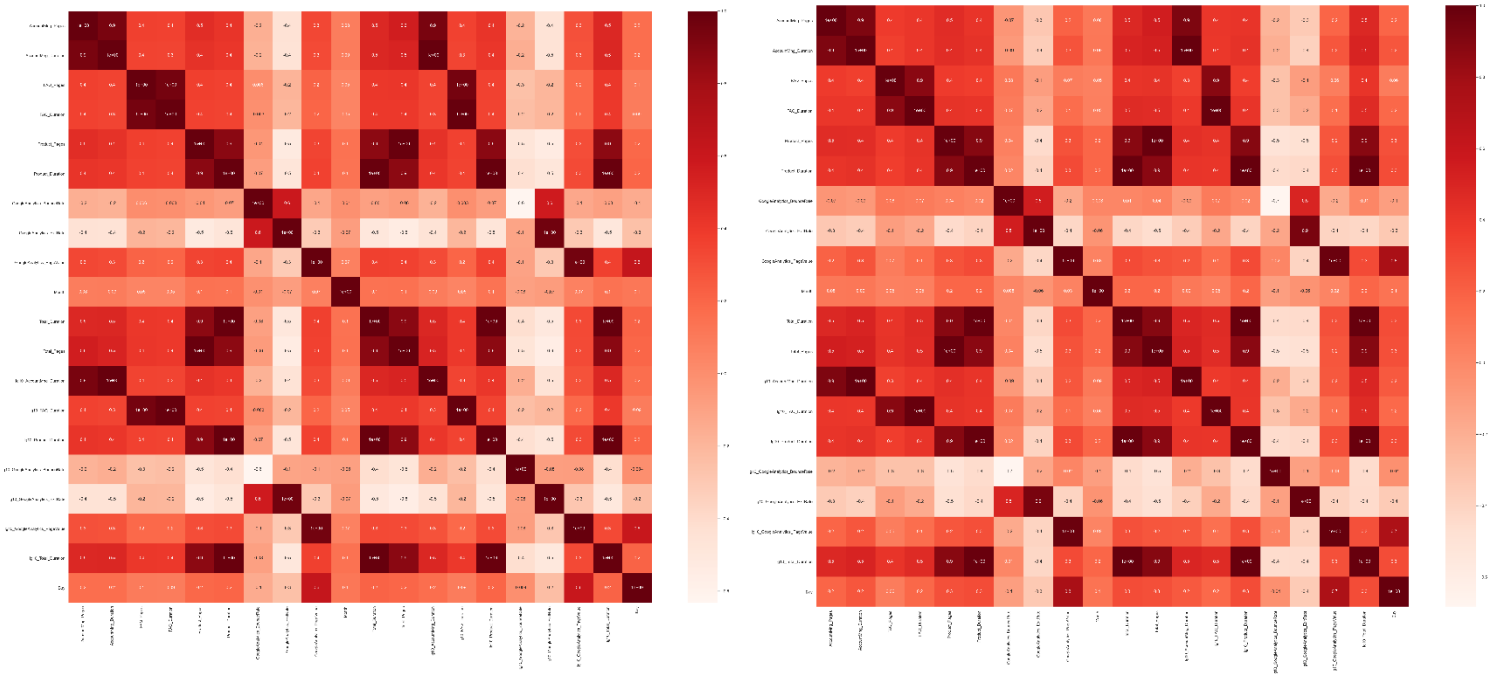

Figure 11 *– Other Jupyter Notebook Outputs: Feature Selection – Spearman Correlation Heatmap for Baseline and Baseline with SMOTE datasets*

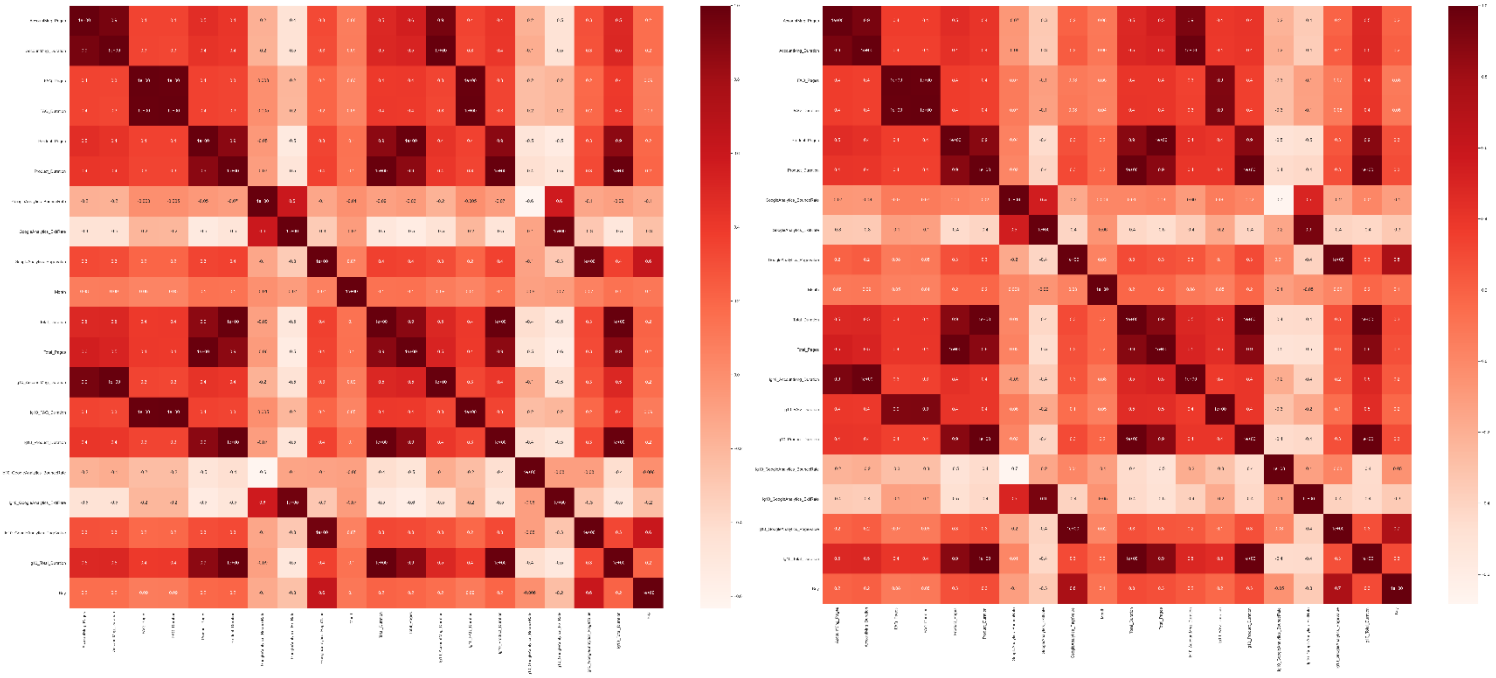

Figure 12 *– Other Jupyter Notebook Outputs: Feature Selection – Spearman Correlation Heatmap for Outliers and Incoherencies removed and for Outliers and Incoherencies removed with SMOTE datasets*

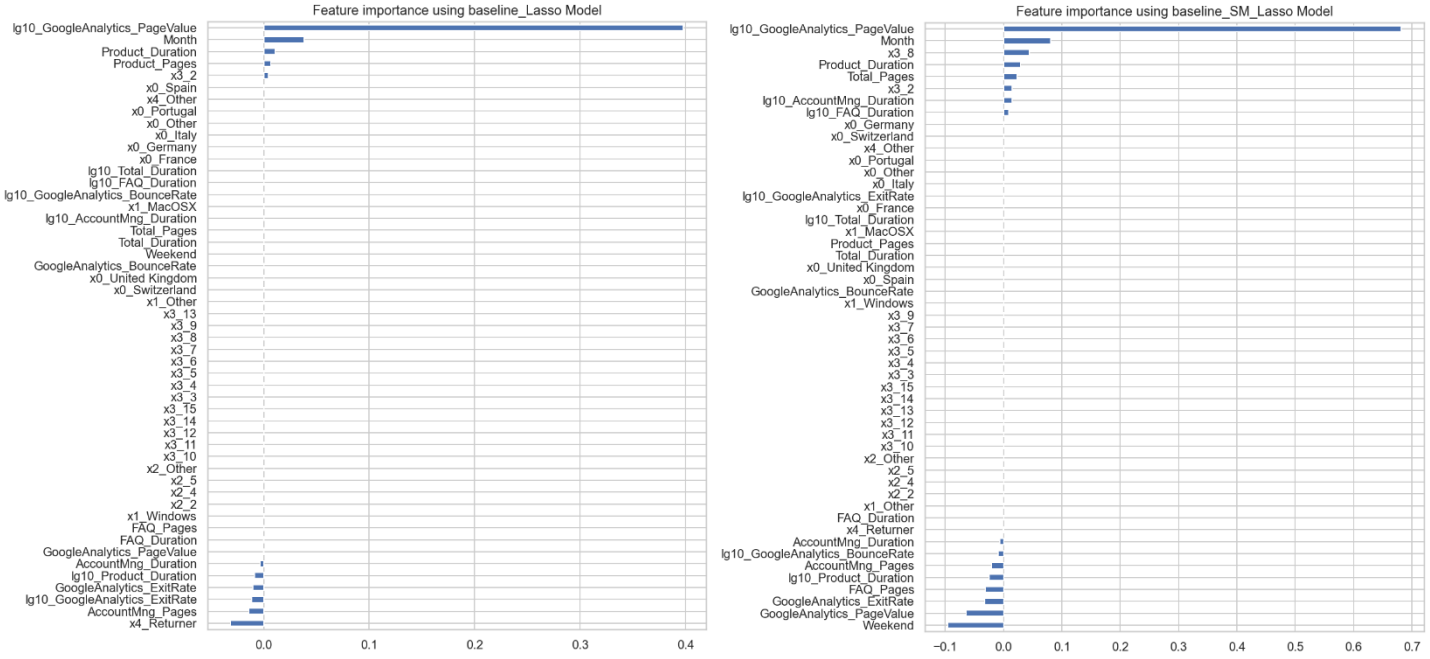

Figure 13 *– Other Jupyter Notebook Outputs: Feature Selection – Feature Importance according to Lasso for Baseline and Baseline with SMOTE datasets*

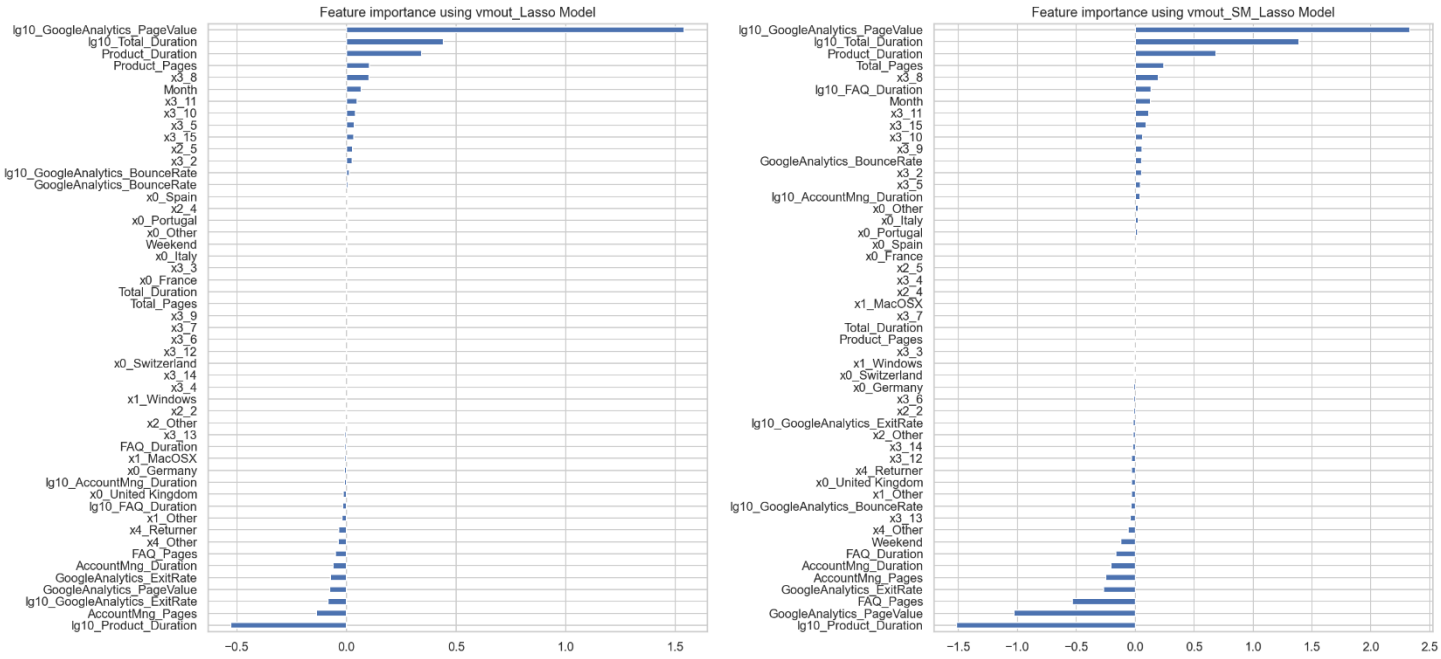

Figure 14 *– Other Jupyter Notebook Outputs: Feature Selection – Feature Importance according to Lasso for Outliers and Incoherencies removed and for Outliers and Incoherencies removed with SMOTE datasets*

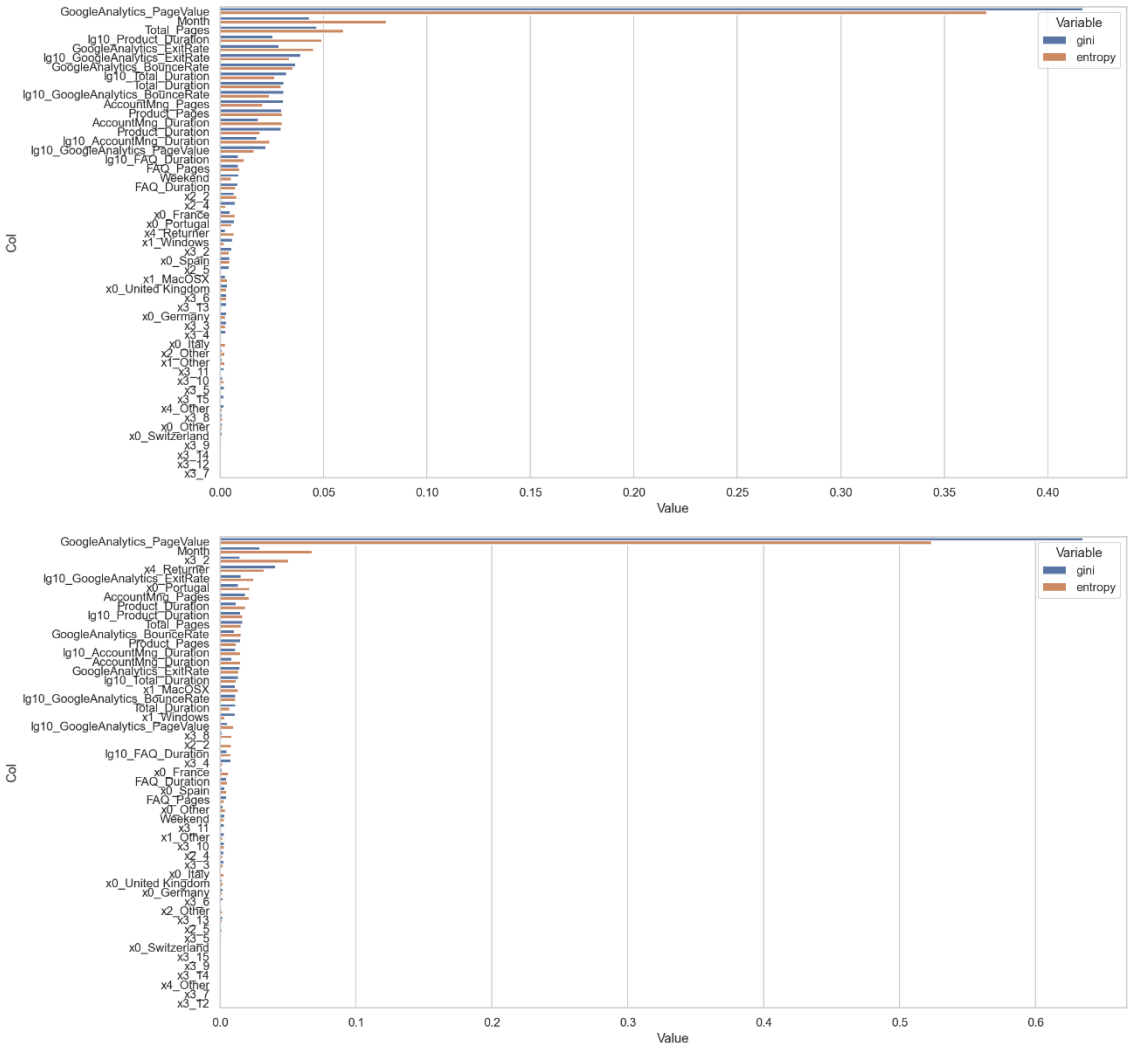

Figure 15 *– Other Jupyter Notebook Outputs: Feature Selection – Feature Importance according to Decision Trees for Baseline and Baseline with SMOTE datasets*

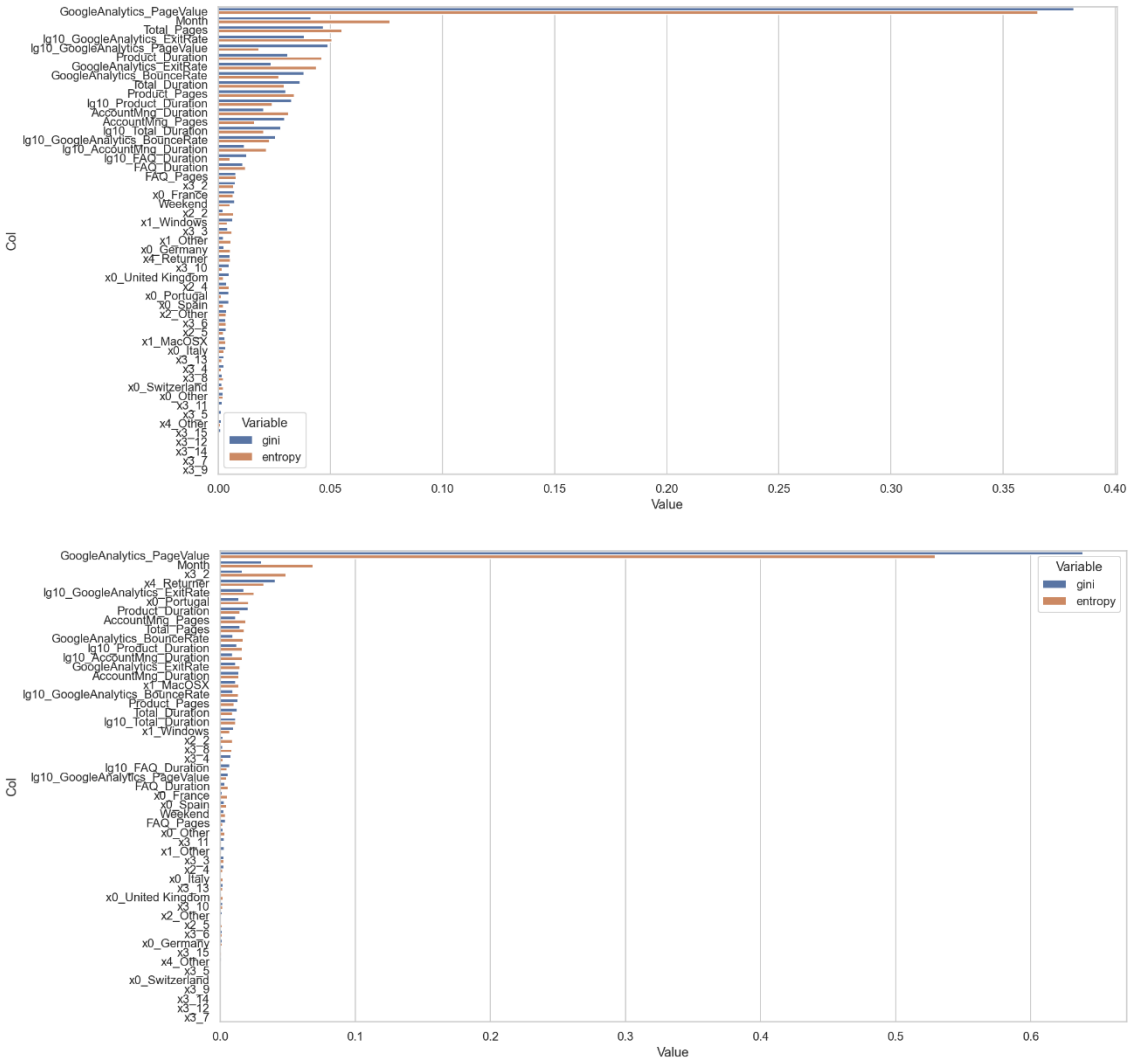

Figure 16 *– Other Jupyter Notebook Outputs: Feature Selection – Feature Importance according to Decision Trees for Outliers and Incoherencies removed and for Outliers and Incoherencies removed with SMOTE datasets*

3. CREATIVITY AND OTHER SELF-STUDY

i. One-Hot Encoding

One-hot encoding is one machine learning techniques used to deal with categorical data. Some algorithms can work with categorical data directly, but others can't. Thus, one-hot encoding is a way to transform categorical data to numerical data. It transforms each unique value of a categorical feature into a new binary variable. As an example, if we have the variable "colour" that can have one of the values (green, red, blue), then one-hot encoding will transform that feature into three new variables: green, red, blue – that will have value 0 or 1 depending on the previous value for the variable "colour" for that same observation.

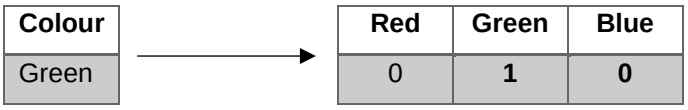

Figure 17 *– Schematic Example for One-Hot Encoding Creating of Binary Variables*

ii. Synthetic Minority Oversampling Technique – SMOTE

SMOTE or Synthetic Minority Oversampling Technique is an oversampling technique used in Machine Learning to balance imbalanced datasets. An imbalanced dataset is one where there is a big discrepancy between the number of observations in the target classes, for example, a big number of zeros (majority class) and a very small number of ones (minority class). SMOTE will, in this case, generate artificial observations with target class equal to one until the number of zeros and ones in the dataset is close or the same.

It works by synthesizing new observations from existing ones (data augmentation). A random sample from the minority class is chosen. Then, *k* of the nearest neighbours for that example are found. A randomly selected neighbour is chosen, and a synthetic example is created at a randomly selected point between the two examples in feature space.

iii. Automatic Removal of Outliers

Being in the presence of a dataset that brought out some doubts regarding the handling of outliers, treating them in an automatic manner was considered. For that, we used the Isolation Forest algorithm. As the name suggests, the Isolation Forest is a tree-based anomaly detection algorithm. It uses an unsupervised learning approach to detect unusual data points which can then be removed from the training data, since it returns the anomaly score of each sample and 'isolates' observations with its random partitioning that produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies. As for hyperparameter tuning to generate the best results, the "contamination" parameter - the proportion of outliers in the dataset -, was placed at 0.03, since it would stand for the 3% maximum value destined for removing rows that can be outliers.

Moving on to the results, although this method provided a higher percentage of rows removed that were seen by the Isolation Forest algorithm as outliers, the distribution of the

affected features and its descriptive statistics showed that the manual method was a more effective way of doing this crucial step of data pre-processing.

iv. Mutual Information Coefficient (MIC)

During the Feature Selection stage, we also used the Mutual Information Coefficient, thus complementing some of the analyses that we had already performed, helping us to understand which variables should be selected when it came to modelling. Thus, we tried to understand the relationship between each independent variable and our dependent variable. The Mutual Information Coefficient between a feature and the target is always between 0 and 1, and the higher this value is, the higher the relationship between the feature and the target. Thus, we gave priority to the features which presented a higher MIC because the higher this coefficient, the greater the information we obtained about the dependent variable through the information we have about the independent variable. This analysis complements the one we had performed through Spearman's correlation because, while correlation measures the relationship between the dependent variable and each of the independent variables, Mutual Information is more general and gives us the reduction of uncertainty in the dependent variable after we know the independent variable.

v. Forward, Backward and Stepwise Selection

In what concerned specifying how independent variables entered the analysis, three related methods were used: Forward, Backward and Stepwise Selection. Forward Selection takes the whole feature set as input, but starts with an empty set, adding a feature at each iteration until it finds the one that maximizes the criterion function. When it had evaluated the score with all the features, the process stops keeping the features responsible for the bigger score. Backward Selection also takes the whole feature set as input, and starts with all variables, removing one at each iteration: the one that, when removed, is associated with the best score leads to the stop of the process. Finally, Stepwise selection is a hybrid of the two previous processes and starts similarly to the Forward selection method - with no regressors. Features are then selected as described in forward feature selection, but after each step, regressors are checked for elimination as per backward elimination. The hope is that as new variables are entered that are better at explaining the dependent variable, variables already included may become redundant. For all four datasets, we used the independent train variables as the data to be used, and the dependent train variable as the target, and the OLS, or Ordinary Least Squares, were used to maximize the R^2 score.

vi. Ridge Regression

When it came to Feature Selection, we also chose to resort to Ridge Regression, instead of using only the insights extracted through Lasso Regression. In order to understand the Ridge Regression, and to be able to compare it with the Lasso Regression, we will use an example.

Let's assume that we are interested in modelling the relationship between people's weight and height. So, let's fit a line to the training data that we have, using Least Squares (that is, let's find the line that minimizes the sum of the squared residuals). This line will have 2 parameters:

the *y*-intercept and the slope. However, by fitting this new line only to training data, we may be overfitting. The fundamental idea of Ridge Regression is that we will try to find a line that does not fit so well to the training data, as we introduce a small bias, to reduce overfitting to the training data. So, this line will be less fit for the training data but will have a good fit for the test data. The Ridge Regression, when determining the values for the above parameters, minimizes not only the sum of the squared residuals, but also takes into account a *lambda* that is multiplied by the squared slope. Thus, the squared slope ends up introducing a penalty to the Least Squares method and the lambda determines how large this penalty is. In our project, we applied the Ridge Regression together with a Logistic Regression. Thus, in our example we will now consider using weight, but for predicting the existence of heart disease (a binary variable, which indicates whether or not the person has heart disease). When applied to a Logistic Regression, the Ridge Regression will optimize the sum of the likelihoods instead of the squared residuals, because the Logistic Regression is solved using Maximum Likelihood. So, we will try to minimize not only the sum of the likelihoods but also a lambda that is multiplied by the squared slope.

In Ridge Regression, as we increase the value of lambda, the optimal slope (which minimizes the terms mentioned above) will get closer and closer to 0 but will never be 0. The main difference between the Ridge Regression and the Lasso Regression, is that the lambda, instead of being multiplied by the squared slope, is multiplied by the absolute value of the slope. Thus, in the Lasso Regression, when we increase the value of the lambda, the optimal value approaches 0, and the optimal value of the slope may even be 0. Thus, while in the Lasso Regression we eliminate features whose optimal slope is 0, in the Ridge Regression we eliminate features whose optimal slope is very close to 0 (because when the slope of a feature is close to 0, predictions for the dependent variable are very little influenced by changes in that feature, making it irrelevant). It is also important to note that when running the Ridge Regression, we set *C* (the inverse of regularization strength) to 1, and that the smaller this value is, the stronger the regularization will be (as mentioned above, since lambda will increase, and the optimal slope will approach 0).

vii. Graphics for Model Assessment & Decision-Making

In the course of this project, we chose on several occasions to use graphics to make better decisions or to extract valuable insights. Thus, it is important to highlight several moments when this happened, throughout the work.

During the Data Visualization stage, we created histograms for our numeric variables, which served two purposes: identifying possible outliers (which could have been done only through boxplots, but we chose to use these histograms as well); and analysing the distribution of variables, to draw conclusions that could be applied when creating new variables. Thus, by using these histograms, we not only identified possible outliers, but also concluded that all variables had a positive skew. Thus, the insights obtained through these histograms were later applied when dealing with outliers (by combining them with the observation of boxplots and tables with statistical data) and when creating new variables (by creating log transformations).

Another time when we used graphs to gain valuable insights was when comparing the results of the models we used. Here, we created bar charts where, for each of the models, we

had the score for the training dataset and the validation dataset. Thus, through a visual inspection, the comparison of the models became easier, as we could understand which models had the best score for the training dataset, which models had the best score for the validation dataset, and, above all, which models were behaving better by predicting accurately for both datasets, thus minimizing overfitting. This way, we were able to make a more correct decision, by choosing a model that maximized the scores and minimized the overfitting.

viii. Training the Top 10 Best Models with All the Data

At the end of creating the models and training them, we selected our top 10 models. This selection was made not only based on the F1-Score that the models had obtained in Kaggle, but also on the F1-Scores for the training and validation datasets. Thus, to try to improve the results that we were obtaining in Kaggle, we decided to train the same models, with the same specifications, but with more data. We made this decision because by training the models with more data, the model would have more inputs and would be better at making predictions. For this, we concatenated the training and validation datasets that we were previously using (this concatenation was done both for the independent variables and the dependent variable). After training the models in the aforementioned way, we calculated the training and validation scores and submitted the predictions of each model to Kaggle.

Thus, we could conclude that, for most models, the score for the training dataset remained quite similar (and sometimes even decreased) in relation to what we had previously obtained. In what concerns the score for the validation dataset, it registered considerable increases. However, this may be due to the fact that the data with which we were validating the models had already been used to train them, this being one of the main weaknesses of this technique we implemented. Finally, regarding the results obtained in Kaggle, we obtained different behaviours for the 10 models. However, we found that there were no major improvements in the performance of the models in Kaggle (even though there were, at times, marginal increases in the scores obtained).

4. OTHER PREDICTIVE MODELS

i. K-Nearest Centroids

Nearest Centroids is a linear classification machine learning algorithm. It is arguably the simplest classification algorithm in Machine Learning. A centroid is the geometric centre of a data distribution. This algorithm predicts a class label for new observations based on which centroid it is closest to from the centroids of the training dataset. Given a data point, the Nearest Centroid classifier will assign it the label (class) of the training sample whose mean or centroid is closest to it. This algorithm works as follows. The centroid for each target class is computed while training. After training, given any point, the distances between that point and each centroid are calculated. Out of all the calculated distances, the minimum distance is picked. The centroid to which the given point's distance is minimum, its class is assigned to the given point.

As for the parameters used to get the final model, we stuck to the default ones. Although several experiments were held with other parameters, we got the best results when using the default parameters for the model.

ii. Passive-Agressive Classifier

Passive-Aggressive algorithms are generally used for large-scale learning and are from the online-learning algorithms' family. In online-learning machine algorithms, the input data comes in sequential order and the machine learning model is updated sequentially, as opposed to conventional batch learning, where the entire training dataset is used at once. Being so, this type of algorithm will get a training example, update the classifier, and then does not consider again that same example. Passive Aggressive algorithm are particularly useful in situations where there is a huge amount of data, and it is computationally infeasible to train the entire dataset.

Regarding the parameters used, in what concerns the aggressiveness parameter (*C*), that stands for the maximum step size – regularization -, values of 0,001 and 1 were tested, since this parameter ranges from 0 to 1, and controls the influence of the slack term on the objective function. Specifically, larger values of *C* imply a more aggressive update step. Moving on to the *fit_intercept* parameter, it stands for the possibility of estimating the intercept, such that when it is set to *False*, the algorithm assumes that the data is already centered. Due to this, this parameter was set to *True* – its default value. For the *tol* parameter, or the stopping criterion, three values were considered: 1×10^{-3}, 1×10^{-2}, and *None* – meaning that the algorithm's iterations stop when the loss is bigger than the difference between the previous loss and the value set for the *tol* parameter – except when it is set as *None*. Finally, as for the *loss* parameter, a loss function must be chosen between *hinge* and *squared-hinge*, depending if it is PA-I or PA-II variant, respectively, where the update of the function changes among the two cases.

iii. Linear Discriminant Analysis & Quadratic Discriminant Analysis

Linear Discriminant Analysis starts by making two assumptions: the observations follow a normal distribution; the categories have the same covariance matrix. Through Linear Discriminant Analysis we are interested in maximizing the separability between the two categories (the 0's and 1's, in the "Buy" variable, in the case of our project), in order to be able to make better decisions. For example, if we are working with data where there are 2 independent variables and 1 dependent variable, we can represent this data in a 2-D graph. In this graph, each axis will be one of the independent variables and each observation will have a colour, depending on its value for the dependent variable. Through Linear Discriminant Analysis, a new axis will be created and the data we are working with will be projected onto this new axis, in order to maximize the separation between the two categories. This new axis is created having in mind 2 objectives: we want to maximize the distance between the means of the two categories; we want to minimize the variance within each category. After creating this new axis, a boundary is defined, which is the threshold that separates the two categories. If we are working with more than 2 dimensions (as was our case during this project, since we had more than 2 independent variables), the process will be similar: we will create an axis that maximises the distance between the means of

both categories (in our case, the accesses with "Buy" equal to 1, and the accesses with "Buy" equal to 0) while trying to minimise the variance within each of the two categories. Similarly, after the axis is created, a threshold will then be defined to separate the categories on this axis.

During this process, we may conclude that there are more important variables to perform this separation, and thus a weight will be assigned to each independent variable (the higher the separation capacity of that variable, the higher weight it will be assigned). After training the model with the training data and then defining the boundary, we can proceed to the classification of new observations. Thus, when we receive new observations (like those from the validation or test dataset), these will be inserted in the computed model and we will observe if this new observation is below or above the cut-off value and, thus, we will finally assign it to one of the two categories, finalizing our prediction for that new observation.

Quadratic Discriminant Analysis is quite similar to Linear Discriminant Analysis. However, it's important to highlight two differences between them, which make them behave differently: unlike Linear Discriminant Analysis, Quadratic Discriminant Analysis assumes that each class has its own covariance matrix; furthermore, the boundary in QDA, instead of being linear, as in LDA, is quadratic. So, if we observe the scatter plot below, we can conclude that the orange line is the boundary defined by a Linear Discriminant Analysis and the green line is the boundary defined by a Quadratic Discriminant Analysis.

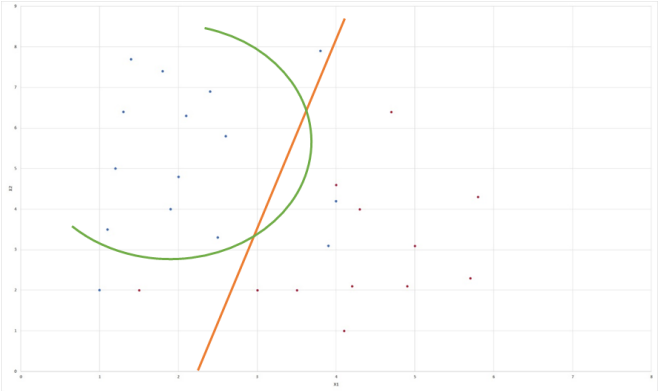

Figure 18 *– Linear VS Quadratic boundary comparison*

As for the parameters chosen for the Linear Discriminant Analysis, they resulted from a grid search, in which we tried to obtain the best specifications for the parameters "shrinkage" and "solver". Thus, except for the "solver" parameter, our Linear Discriminant Analysis presents all parameters with default values. By specifying the "solver" as "lsqr", instead of the "solver" being done through Singular Value Decomposition (the default "solver"), it was done with a Least Squares solution, which could be combined with other parameters that the default "solver" doesn't allow. However, the grid search we ran indicated that the remaining parameters should not be changed. Thus, the results we obtained with our Linear Discriminant Analysis, with the "solver" being "lsqr" did not show overfitting or underfitting. The same happened for the Quadratic Discriminant Analysis, where we tried to obtain the best value for the parameter "reg_param", through a grid search. As in the Linear Discriminant Analysis, all parameters except for the "reg_param" parameter were the default ones. By specifying "reg_param" with a value of 0.99, the covariance of each category will be regularized. Thus, the covariance estimate for each category will be (1 - 0.99) * Sigma + 0.99 * M (where M is an identity matrix, with the number of rows equal to the number of features present in the data). This way, we try to reduce overfitting

(which was accomplished, since we obtained almost identical F1-Scores not only for the training and validation datasets but also for the data submitted on Kaggle).

iv. Voting Classifier

Voting Classifier is an Ensemble machine learning model that joins other predictive models and uses them to “vote” on what the best prediction is. The fact that it makes used of various other models makes Voting Classifier a Meta-Model, or a collection of algorithms working synergistically to make better predictions than the ones that could be made individually.

There are two kinds of voting classifiers, hard votes and soft votes. Hard vote uses the final prediction of each of the chosen models to decide, based on the majority of the votes, what prediction is the most correct. Soft vote uses the sum of probabilities given by the contributing models to predict the target feature.

This makes hard and soft voting fundamentally different in what they predict best, making hard voting the better option for class label prediction and soft the better at predicting class membership probabilities. In our case, hard voting was used.

Predictions can also be affected by weight, as it can be specified for each of the contributing models. If no weight is specified, all the models have equal weight, which can prove problematic if their individual predictions have significant diversions in accuracy. With this said, voting works best when models in the ensemble generally agree on the prediction and have similarly good performance.

Although the objective is to achieve better results with Voting ensembles, this is not always the case. Therefore, it should be opted for when the performance is better than any of the individual models in the ensemble and also yields a lower variance then them. Also, to increase effectiveness, the best-case scenario for voting classifiers are ensembles models trained stochastic learning algorithms and with slightly hyperparameters.