TechScape Group Project Report

Machine Learning Course

Master's in Data Science and Advanced Analytics at NOVA IMS, Lisbon

Group 21:

Peter Hamori - Ali Sabbir - Nicola Andreatta - Foazul Islam - Elsa Camuamba

Abstract

TechScape is a portuguese startup company founded in 2020. They sell goods related to digital detox via their online store. TechScape registered the activity of its customers in their online platform and provided all this information to us. The aim of our project is to analyse the online behaviour of the customers of TechScape and to build a predictive model which is able to differentiate between customers that are likely to buy the products and those who are not, based on their online activity.

In order to achieve this goal, some machine learning techniques were implemented. First, the data was preprocessed by checking for incoherences, identifying potential outlying observations, building new meaningful features based on the original ones and transforming their values to the same comparable scale. Then, a subset of the most important features was selected with respect to the target variable. Lastly, using the preprocessed training set, several predictive models were built, including simple predictors such as logistic regression, decision tree and K-Means classifier, as well as more advanced methods like neural networks, boosting methods and ensemble models.

The final solution was obtained with Random Forest classifier, which achieved a 0.72 binary F1 score on the training set and 0.65 score on the validation set.

I. Introduction

The purpose of the current work is to predict whether a customer is likely to buy a product knowing some detailed information about its research on the online store. The outcome of a customer's research is provided to us in the training set, indicated by the dummy target variable 'Buy'. The training set consists of 9999 records which we are using to train our predictive models on. The final goal of the project is to carry out predictions for the 2300 customers of the test set. The features of each set hold general information about each customer (e.g.: *Country*) as well as specific information regarding their activity on the online platform (e.g.: *FAQ_Pages*). In order to come up with an optimal solution, we build and train several predictors on the preprocessed training set. We compare their performance in terms of both F1 score and overfitting and in the end select the best one for predicting the binary labels for the test set.

II. Background

In the project we used some techniques to preprocess the data, select the most important features and create predictions that were not part of the course. A detailed explanation of these are reported in the Appendix.

III. Methodology

III.1 Data Exploration and Preprocessing

Prior to beginning the analysis, a preliminary exploration of the raw dataset was conducted to assess its validity (and quality), and to prepare it for model building and training.

While no duplicates, missing values or empty strings were found, inconsistencies in the data types for *Browser* and *Type_of_Traffic* were corrected to accurately represent their categorical values. Similarly, the variable *Date* was converted to a datetime object to allow for easy manipulation and extraction of useful date-time features such as *Month*. This way, we started our analysis with 9 independent metric features and 5 categorical ones.

Next, a review of the distribution of the target feature *Buy* revealed a class imbalance in which the minority class denoted as 1 and corresponding to users who purchased at least one product, account for 15% of the training dataset. This uneven representation in the target class labels by definition makes it harder for a model to learn the characteristics of the observations from the minority class.

Based on the descriptive statistics, boxplots and histograms of metric features *(1, 2),* we became aware of the existence of outliers. These are observations that are different from the majority of the records and thus may bias our predictors. However, we argue that in the test set there can also be outliers and so we can only expect our models to classify them correctly if we also train them on some outliers. Some of the models are robust to outliers and so they might produce good results. For this reason, we maintained a filtered and a non-filtered version of the dataset and experimented with both in the modelling. Based on the metric features' values, our filter identifies roughly 2.5 percent of the observations as outliers, which is inside the 3 percent critical limit.

Regarding categorical features, outliers can be considered to be the records of extremely underrepresented categories *(3)*. In our case, *Browser* and *Traffic* both have some categories with only a few instances. A possible approach could be to merge these categories (e.g.: *Browser7 + Browser11 + Browser13 -> Browser7_11_13*). In the end we did not exploit this option.

Further Exploratory Analysis Using Time-Indexed Values to Understand Customer Behavior

Adding a more meaningful temporal dimension can help to contextualise and measure TechScape's

web visitors' actions. To that effect, a series of time-indexed visual explorations were performed. For this, both the average and median values of selected features are plotted side by side to help gauge possible bias in the analyses due to the sensitivity of the mean to potential outliers in the data *(4, 5, 6, 7, 8, 9)*.

Among the main patterns identified, we observe that the average amount of time users spent viewing product and services related pages reveals an upward trend, and is far superior in comparison with time spent on account management and FAQ pages *(10, 11)*. This analysis also shows that 'Returners' make up the majority of users who buy at least one product throughout all the months in which TechScape has been in operation, with the highest number of buyers registered in November and May, respectively.

Feature Extraction and Engineering

To produce a more salient set of features for our model, several transformations were performed to the existing input features. A description of the newly created features are presented below:

New feature	**Description**
Total_Pages_Visited	Based on the combination of *AccountMng_Pages, FAQ_Pages* and *Product_Pages*
Total_Duration	Based on the combination of *AccountMng_Duration, FAQ_Duration* and *Product_Duration*
Month (12)	Based on the decomposition of *Date*
Weekend (13)	Binary, 1 for weekend. Based on the decomposition of *Date*
Day (14)	Based on the decomposition of *Date*
Clusters (15)	Based on the application of KMeans algorithm on select metric features [3,4]
PageValue_Month	Based on the positive interaction found between *GoogleAnalytics_PageValue* and *Month.* This feature was built dividing the first one by the second one

We identified three problems with our features and tried to solve each of these.

Firstly, to overcome the challenges given by metric features with skewed distributions, we transformed each skewed metric feature to a logarithmic scale, this way creating new features *(16)*. Another option to mitigate this problem could have been to perform square-root transformation.

Secondly, the dataset is heavily imbalanced, which makes the job of predictors hard. A common approach is to resample the dataset in order to even out the number of observations of the two

classes. Following the idea of recent papers, we oversampled the minority class and undersampled the majority. Oversampling was done with SMOTE[1], while for undersampling we opted for the random selection[2]. We experimented with several parameters and resulting ratios, but in the end we found that resampling did not help the predictions.

Thirdly, we argue that the one-hot encoded categorical features get greater importance than the scaled metric features, because the expected difference between two arbitrary observations along a dummy dimension (assuming that the given attribute is balanced) is 0.5, which is higher than that of a metric feature of scale [0; 1]. A solution we came up with is to discretize each metric feature and then one-hot encode each. This way we only had one-hot features and so all of them would be taken into account with equal importance. However, this approach raises another problem, namely the curse of dimensionality. For this reason, we decided not to include it in our final solution.

Assessment of the target discriminatory power

Before going into the feature selection we explored the discriminatory power of the attributes with respect to the target variable *(17, 18, 19, 20)*. We first plotted categorical features showing the percentage of buyers for each layer of the attribute. Secondly, we plotted the percentage of buyers given the metric features grouped by discrete intervals of values, in order to be able to observe if the proportion of buyers was depending on the selected intervals. In this last analysis, we found an interesting outcome in the case of log-transformed features, which showed a linear interaction between the proportion of buyers and intervals of the log-transformed features *(21)*.

III.2 Feature Selection

To decide upon which features to use in the models, we considered first of all a smaller random portion of the training set to test the relevance of the features on the remaining part of the data used as a validation set, preserving the same stratification of the target variable. Later on, we decided to improve the features selection methods by introducing cross validation, with the purpose of reducing the possible bias given by the same initial random instance.

To not bias the selection, we also transformed the values of each metric feature to the [-1; 1] scale using MinMax Scaler. We would like to mention that we also experimented with the Standard Scaler and the Robust Scaler, although the differences in the later stages were minimal.

Then, we structured the selection of the metric features in the following way:

- First, we examined the relation between each two features using the Pearson and Spearman correlation matrices *(22, 23)*, in order to filter out redundant attributes and attributes with low correlation with respect to the target variable. As expected, we found a strong positive correlation between each pair of attributes “number of pages visited” - “time spent on the

pages" belonging to the same section of the website (namely Account Management, FAQ and products). This means that in the data, the time spent in the pages grows roughly pairwise with the number of pages visited. Hence we considered the exclusion of the duration feature for each pair to avoid redundancy in the information. For what concerns the correlation with the target variable, among the attributes built in the previous feature engineering stage, only *Month*, *Clusters*, and *PageValue_Month* showed some significance. Instead, in the original ones we found *GoogleAnalytics_PageValue* to be the most significant, with Pearson's coefficient value 0.48. Since the correlation found among most of the attributes and *Buy* in Spearman matrix was higher than the ones in Pearson's, we deduced that the relation was not linear. Hence, also for this reason we thought to transform the data to improve this relationship. In fact, with the log-transformed dataframe, the value of *GoogleAnalytics_PageValue* improved to 0.62. The conclusion was that models more sensible to the linearity of data would have improved with the log-transformed dataframe.

We also analysed the importance of the features measured with models such as Logistic regression, Random forest classifier and XGBoost [5,6], setting a threshold to detect any attribute with a weak discriminatory power *(24)*.

- Secondly, we thought to use wrapper methods such as Recursive Feature Elimination to fit the models on any subset of features and choose the one that provides the best classification on the validation set, in terms of f1 score *(25)*.
- Moreover, we implemented two methods that could identify the features that contribute the most to improve the prediction and the ones that cause overfitting, namely LASSO regression and Ridge regression *(26)*. In the latter we introduced a low threshold to exclude features that possibly are not important.

To gain an understanding of the relationship between each categorical feature and the target, we used the chi-squared independence test *(27)*. We concluded that *Country* is not an important predictor and thus needs to be excluded.

At the end of the feature selection process, we summarised in a grid the features supposed to be discarded and the ones to be kept for all the methods we implemented *(28)*. The features that resulted as important for the majority of the methods were included in the final subset of selected features.

Finally, to prepare the categorical features for modelling, we applied one hot encoding (OHE) as all of the categorical features in the train dataset are nominal, i.e., they do not assume a natural rank ordering between the categories. By encoding the features in this way, each category is assigned a binary value of 1 or 0 otherwise. We note that some of the challenges of this approach include, on one hand, the increase in the feature space and on the other hand, the possibility of a mismatch with respect to the number of categories and/or the frequency distribution of the categories present in the train set and the test set.

Alternative approach

In the modelling stage, we realised that this strategy was not the most efficient, since each of the models responds differently to the feature selection methods.

Thus, after a first phase of modelling, we decided to set up the feature selection for each specific model. The common selection technique we used is structured in two main steps: assessment of the most important features for the model and selection of the best subset of features through wrapper methods. The only exception was logistic regression, since it provided the best score with the features selected with LASSO regression.

III.3 Modeling

Considering the nature of a classification problem, the models that we explored and implemented are the following: logistic regression, random forest classifier, gradient boosting classifier, adaptive boosting classifier, multilayer perceptron classifier, k-nearest neighbours and random k-nearest neighbours classifiers [9]. In addition, we tried some ensembles of these predictors to further improve the prediction. As mentioned in the previous section, we measured the feature importance for each model and then we applied two different algorithms, namely recursive feature elimination (RFE) and sequential feature selection [7] (SFS). Although both of them select the subset of features sequentially, the first one chooses the next feature based on the feature importance, while the second selects the following feature based on the model performance. For this reason SFS requires more time to determine the best feature at each iteration.

Once we found the best set of features, we explored the possible parameters for each model to find the setting providing the best f1 score. We first tested many different values for each of the parameters, but since this could lead to a local optimum we implemented an hyperparameter optimization through the grid search with cross validation: this algorithm returns the maximum average score among all the possible combinations of the parameters specified in the input. Since it is an exhaustive search, the computational time grows exponentially as the amount of values given per parameter as input.

IV.6 Results for the finding of parameters of the different models

IV.6.i Results for Grid Search for MLP classifier

We have used Grid Search for MLP classifier for our model selection. Here, we have implemented all the steps necessary for the model, i.e finding best Hidden layer size, activation function, solver, LR, LR type, Max iteration sequentially. Hence, we have found 2 hidden layers with 7 neurons was optimum, following on the progress reLU is the best activation function along with the best solver is ibfgs. ibfgs solver not only took a small time but only with better accuracy.

we have also used GridSearchCV to find the best possible parameters, which includes fitting 3 folds for each of 432 candidates, totaling 1236 fits.

Hence, the resulting best parameters are:
{'activation': 'tanh', 'alpha': 0.05, 'hidden_layer_sizes': (6,), 'learning_rate': 'constant', 'learning_rate_init': 0.1, 'solver': 'sgd'}.
We have also used k-fold cross validation to find a better model and appended the result. The results using the GridSearch seemed to extract better result and reached convergence. We have used those as the optimal parameters for the rest of the models.

IV.6.ii Results for parameters for Gradient Boosting Classifier

We used different learning rates (such as 0.1, 0.3 and 0.5) to the model and then we checked the result. Right after we have applied different numbers of estimators (50, 100, 150, 300) for the learning rate 0.1. After that we used different kinds of minimum number of samples to split the model with learning rates 0.5 . With learning rate 0.1, we used the different types of minimum sample leaves (for instance, 1, 2, 5), different numbers of subsamples and various kinds of maximum features (such as 3, 7, None). Finally we got the best parameters for this classifier. At the end, we applied k-fold cross validation by taking all the best parameters of Gradient Boosting Classifier.

Adaboost Classifier

For our purposes, we have used adaboost classification to improve the performance of a decision tree[8]. We have incorporated an Adaboost classifier in our model using all default parameters, in general that includes estimators, LR, algorithm, and random_state. Primarily, the model fitted with 3 folds for each 24 candidates, totalling 72 fits.
In the stratified k-fold cross validation, using k-fold average score, we have tried to find a better result of the adaboost model and finally appended the result with the previous model.

III.2.iii Random Forest Classifier

In this model we have tried to find the best parameters directly by running a GridSearchCV. We took the best combo of features for RF with any different values of each parameter and later used the random grid to search for the hyperparameters, which fitted with 3 folds for each 100 candidates, totalling 300 fits. Our final best parameter set for RF is: Random Forest Grid Search CV Parameters : {'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 6, 'max_features': 'sqrt', 'max_depth': 45, 'bootstrap': False}.
Finally, in stratified k-fold cross validation, using k-fold average score, we have appended the result with the previous score.

SVC (Support Vector Machine Classifier)

In this case, we used grid search to find the best parameters for SVC (Support Vector Machine Classifier). We have implemented parameter space and this precisely checked it. Afterwards, we have fitted the model with 3 folds for each 12 candidates, which totals 36 fits. Then, we also initialised the model using the best parameter to check the result in a dataframe. The scores of SVC by taking the best parameters and similarly we have applied for cross validation as well.

Finally, the best parameters are: {'C': 1, 'gamma': 'auto', 'kernel': 'linear'}.

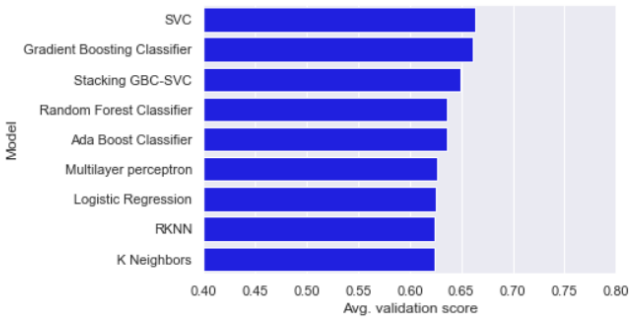
KNN (K Nearest Neighbours Classifier)

Here, we have defined the parameters spaces (such as 3, 5, 7, 9, 11, 13, 15, 17) and then we integrated grid search instances with the set of parameters that fitted 3 folds for each of 8 candidates, totaling 24 fits. Here we got 13 as our best space. Using the best parameters, we initialised the model and then checked the results in a dataframe. We applied it for cross validation by taking the same parameter.

Random KNN trees:

III.3 Model Selection and Parameter Tuning

IV. Results



The goal of this project was to provide the best prediction for the test set based on the f1 binary score. We trained the models with a cross validation in order to obtain a more consistent and robust prediction. The average validation score for each model is reported in the following picture. Overall, the score was not much different among the models *(29)*. The performance on the test provided on kaggle was slightly higher for most of the models and the best was found with the random forest classifier with a score of 0.72.

V. Discussion

Although we conducted a deep exploratory analysis and we tried to improve many times some of the stages to get the best prediction, the final score has not improved much more from the basic models without any optimization that we tested in the beginning. This issue was probably due to the small size of the dataframe and mostly to the low amount of features available. We expected to have a better result with some of the most advanced techniques and probably the information contained in the data frame was not enough, perhaps some information regarding the demographics of the customer would have helped to identify some patterns in the purchases.

Another issue was handling many nominal categorical features: most of them turned out to contain a great discriminatory power, however their encoding and dimensionality complicated the feature selection since the relevant information was sparse among the layers of each feature and the inclusion of one-hot encoded features in the models may lead to create noise for some models because of the binary discrete values they can assume. We tried many resolutions, such as Principal Component Analysis, merging layers with same discriminatory power with respect to the target variable, discretize the metric features in order to put all the feaures on the same level. We discarded the first because the reduction of the dimensionality through PCA among the independent attributes didn't take into account the correlation between them and the target variable, while the other two did not bring much of an improvement.

Finally, much work was carried out to figure out how to obtain the same score on the validation set and on the test set. However some models performed better on the test set and since the score on the test set provided to us corresponds to 30% of the total dataframe i.e. 690 records, this may lead to some discrepancies with the score on the overall test set.

VI. Conclusion

This project aimed to build a prediction to identify the customers of TechScape that are most likely to purchase goods from their platform. We explored the dataframe and we found many insights to build some new features and transformations to improve the discriminatory power of them. We implemented many different features selection methods and finally we built up a specific structure for each model. After an optimization of the parameters for all the models we compared the score for all of them and we obtained the best on the test set with a random forest classifier. We found some limitations in the initial size of the dataframe and the number of features and further research should be conducted to improve the selection of categorical features.

VII. REFERENCES

[1] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research 16 (2002) 321–357*

[2] Random undersampling: https://www.site.uottawa.ca/~nat/Workshop2003/jzhang.pdf

[3] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys, Vol. 31* (Issue 3), 265 - 323

[4] sklearn's documentation (2021), Clustering, Available at: https://scikit-learn.org/stable/modules/clustering.html, Accessed on 9/12/2021

[5] XGBOOST: https://machinelearningmastery.com/calculate-feature-importance-with-python/

[6] Bingyue Pan (2018). Application of XGBoost algorithm in hourly PM2.5 concentration prediction. *IOP Conf. Ser.: Earth Environ. Sci. 113 012127*

[7] SFS: https://analyticsindiamag.com/a-complete-guide-to-sequential-feature-selection/

[8] Adaboost Classifier: https://www.mygreatlearning.com/blog/adaboost-algorithm/#How%20Does%20AdaBoost%20Work

[9] Random KNN: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.330.6112&rep=rep1&type=pdf

VIII. APPENDIX

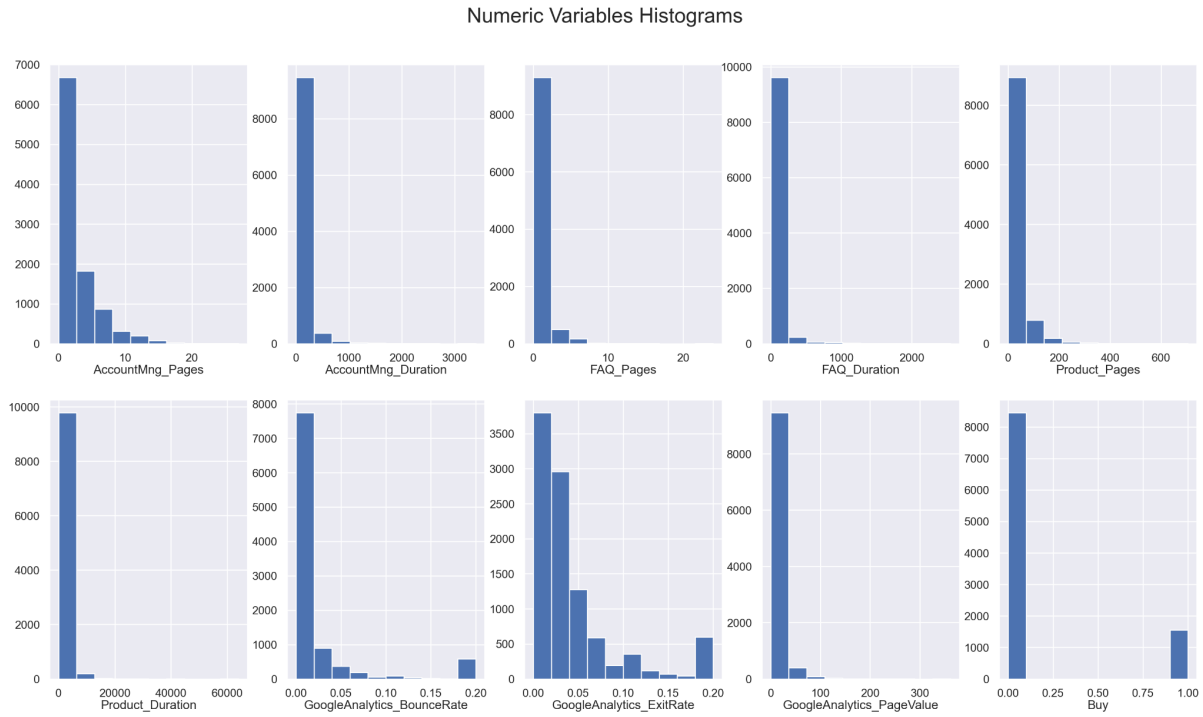


Figure 1: Histograms of Numeric Variables

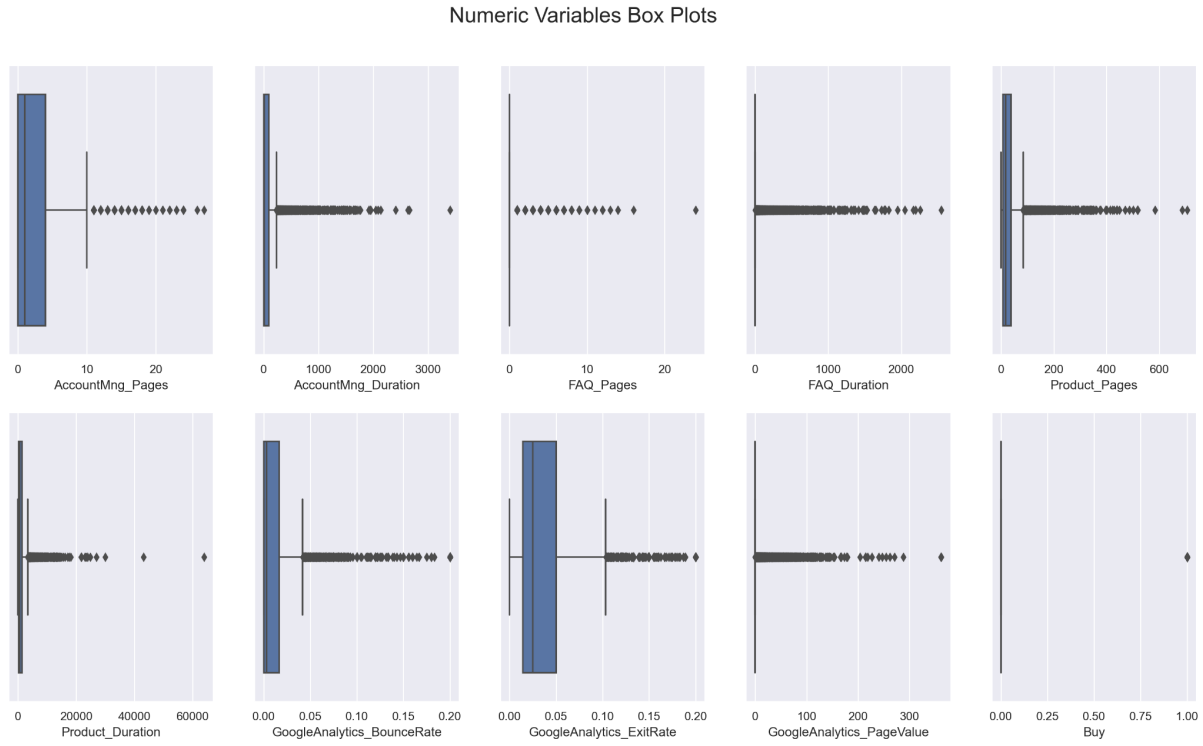


Figure 2: Boxplots of Numeric Variables

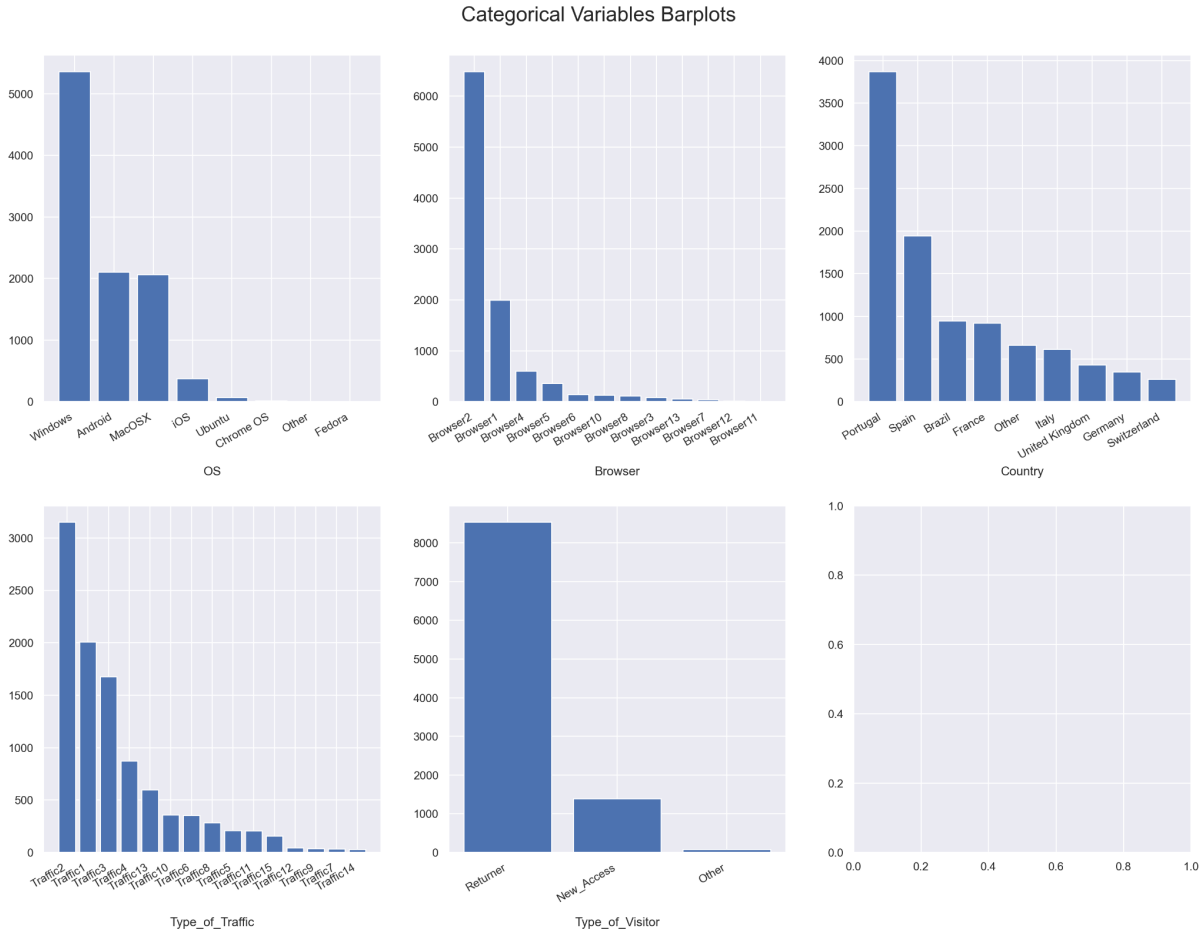


Figure 3: Barplots of Categorical Variables

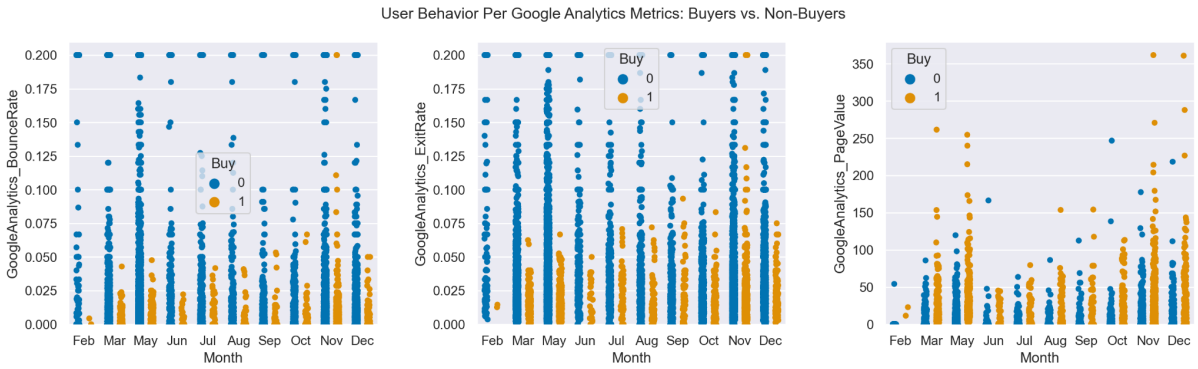


Figure 4: User Behaviour per Google Analytics Metrics for Buyers and Non-Buyers

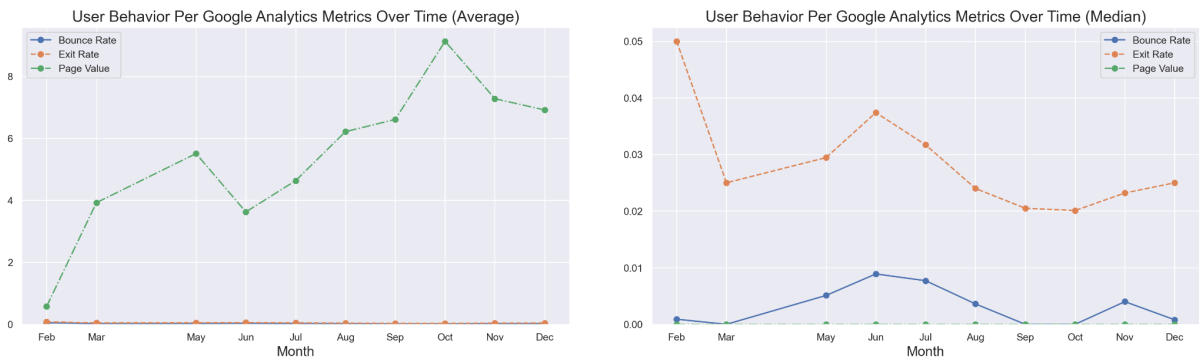


Figure 5: Average and Median User Behaviour per Google Analytics Metrics over Time

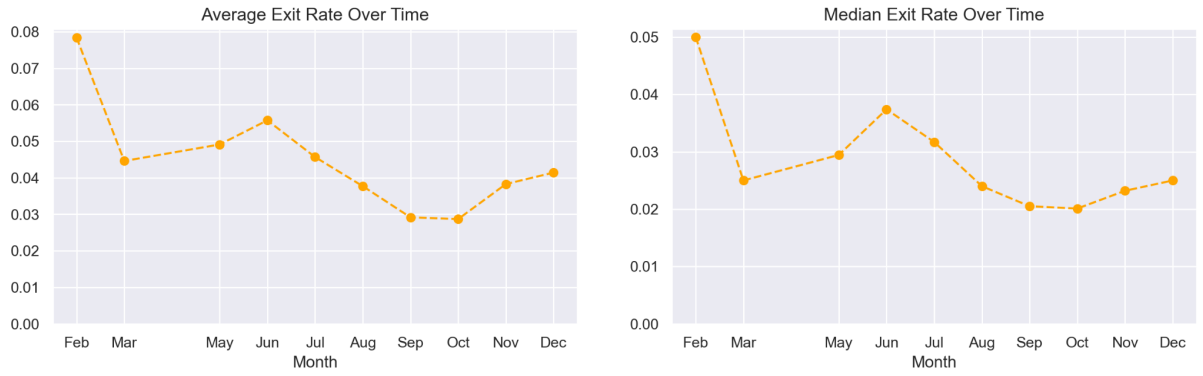


Figure 6: Average and Median Exit Rate over Time

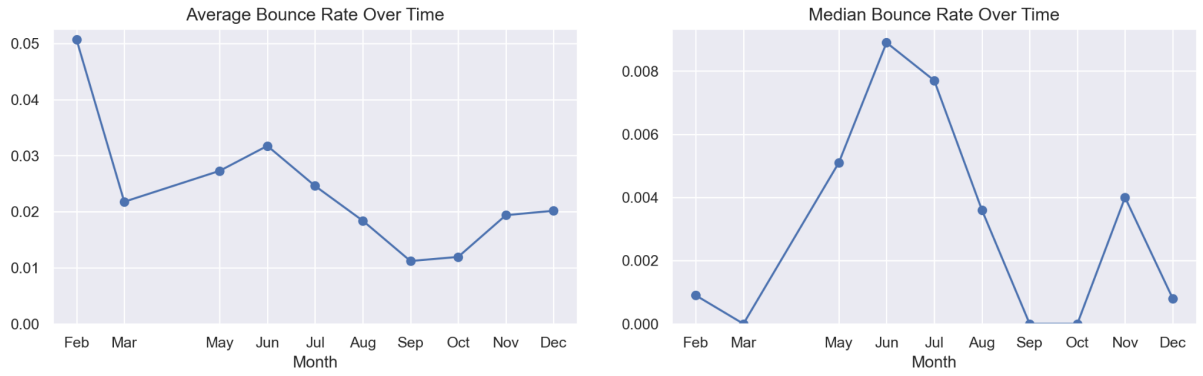


Figure 7: Average and Median Bounce Rate over Time

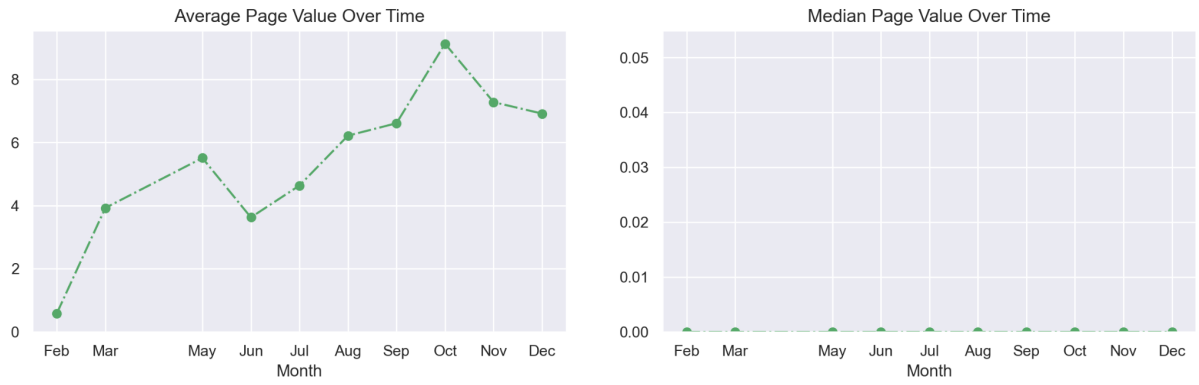


Figure 8: Average and Median Page Value over Time

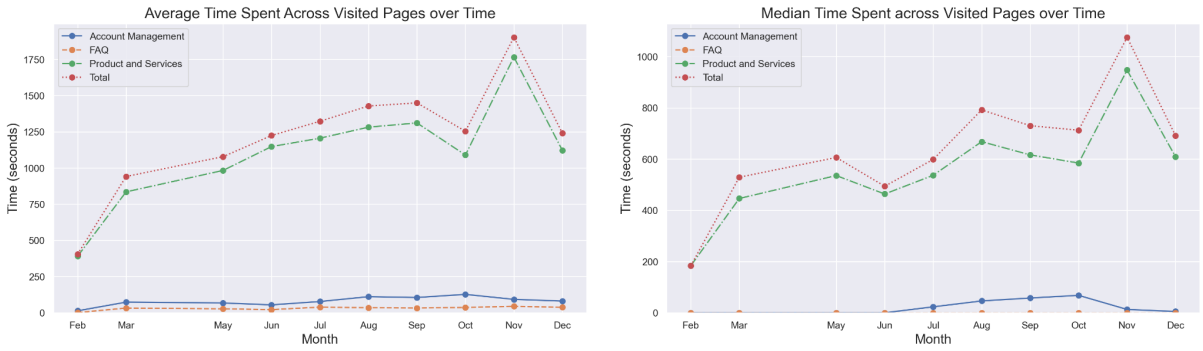


Figure 9: Average and Median Time Spent Across Visited Pages over Time

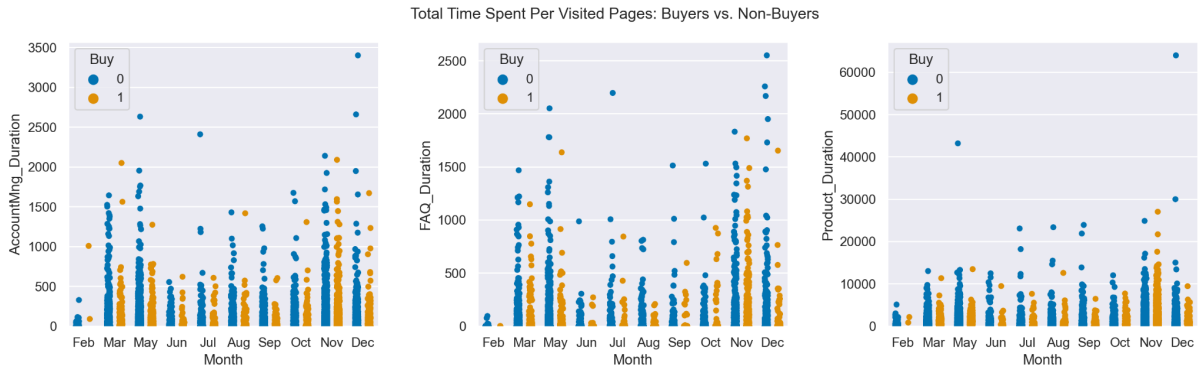


Figure 10: Total Time Spent Per Visited Pages for Buyers and Non-Buyers

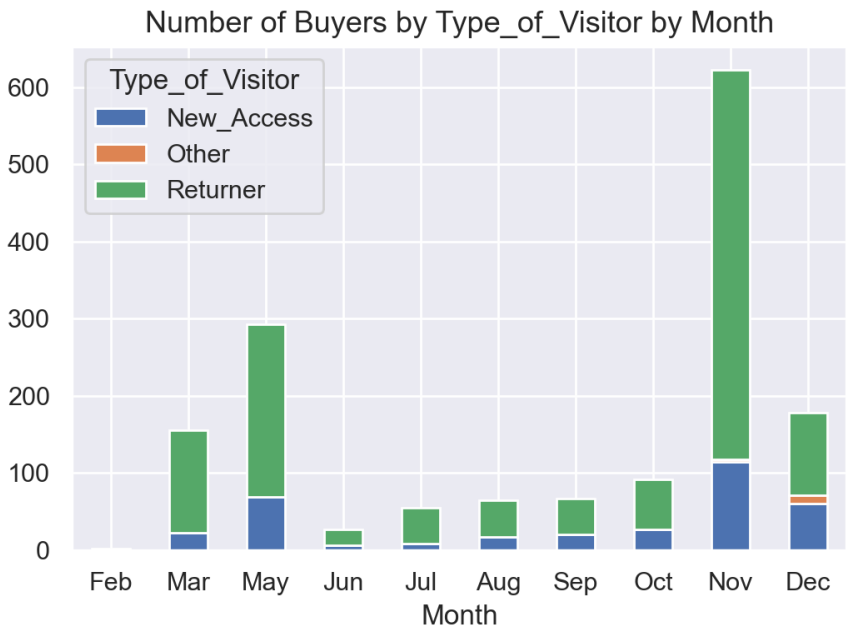


Figure 11: Number of Buyers by Type of Visitor by Month

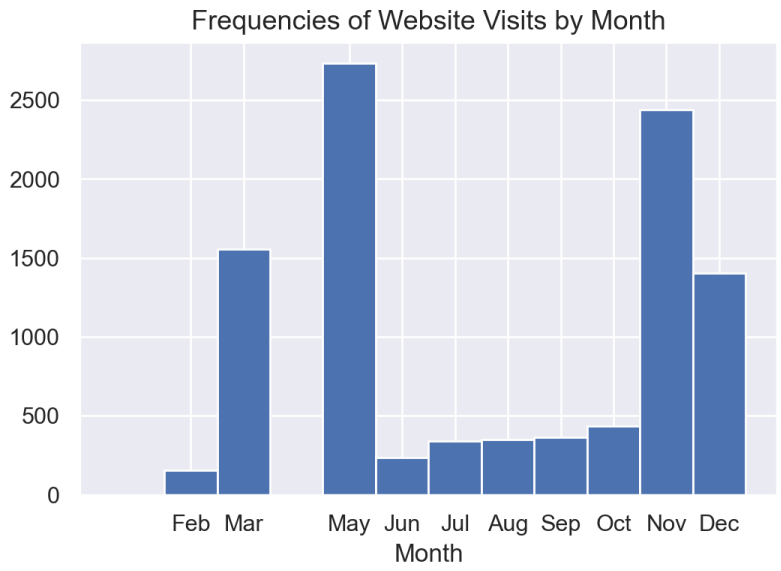


Figure 12: Month Feature (Frequencies of website visits)

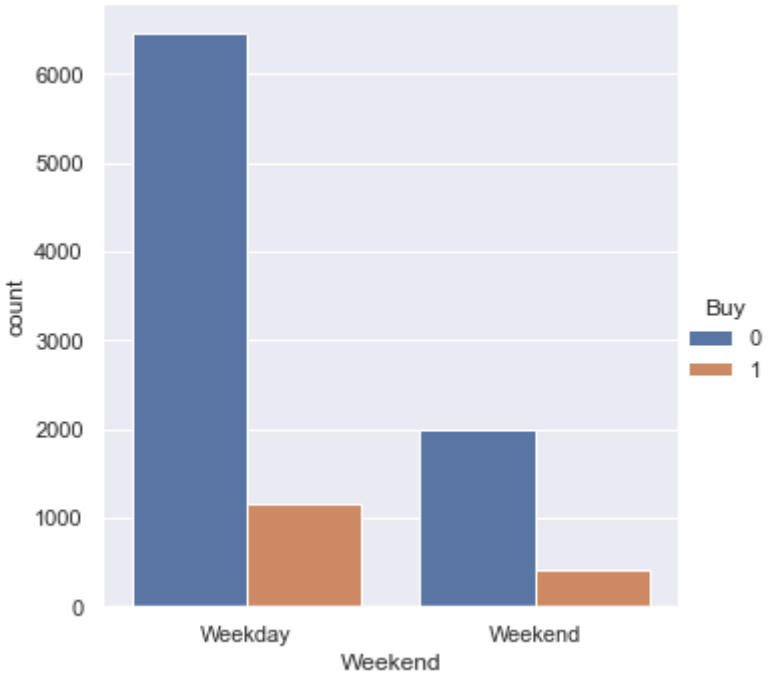


Figure 13: Weekend Feature (Frequencies of website visits)

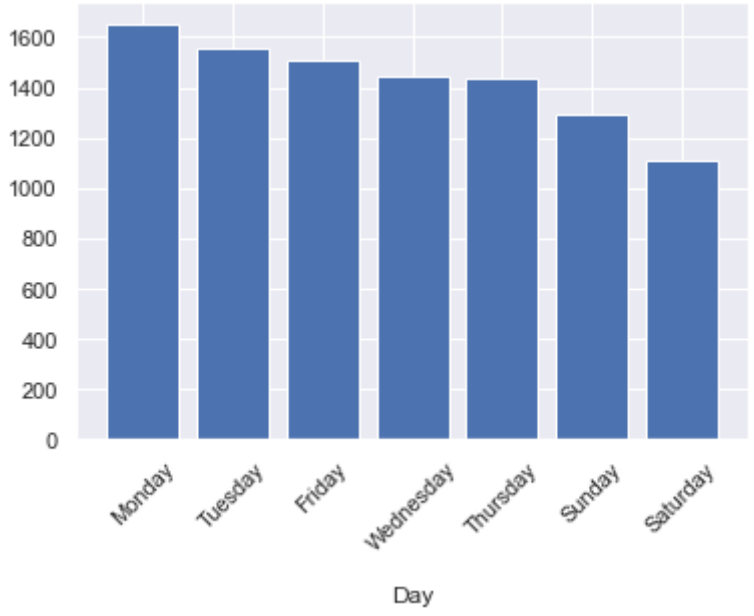


Figure 14: Day Feature (Frequencies of website visits)

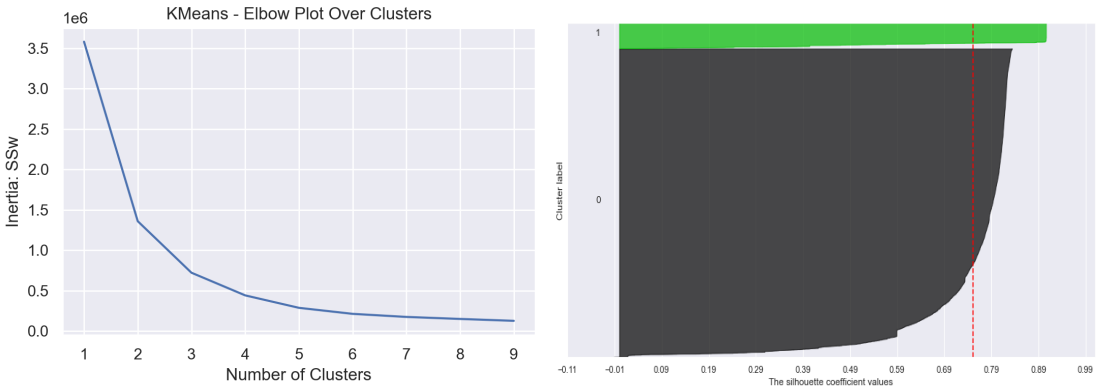


Figure 15: K-Means Elbow Plot and Inertia Plot for the Optimal Number of Clusters

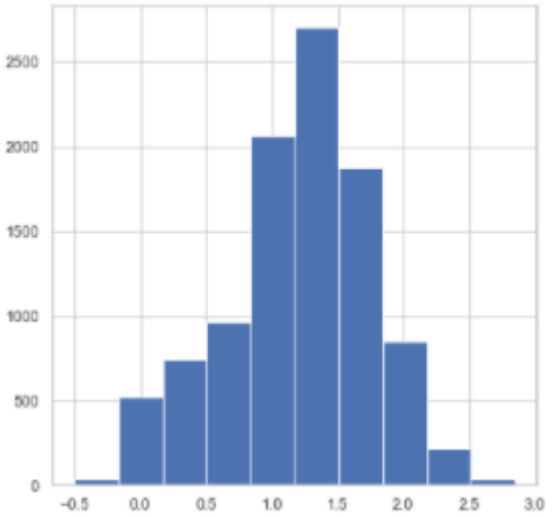


Figure 16: Distribution of Values of the Log(Product Pages) Feature

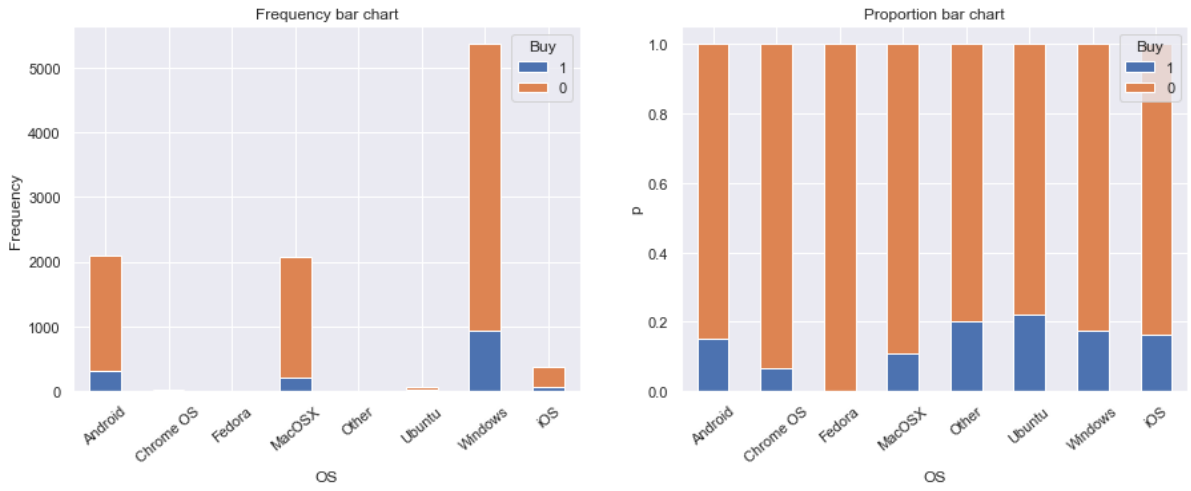


Figure 17: Discriminatory Power of the OS Feature (absolute and proportional values)

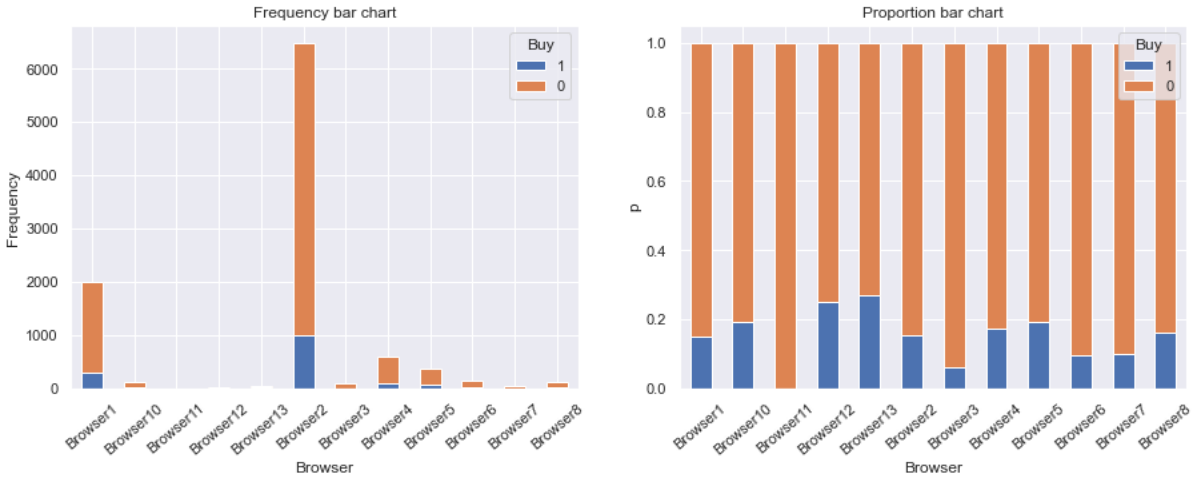


Figure 18: Discriminatory Power of the Browser Feature (absolute and proportional values)

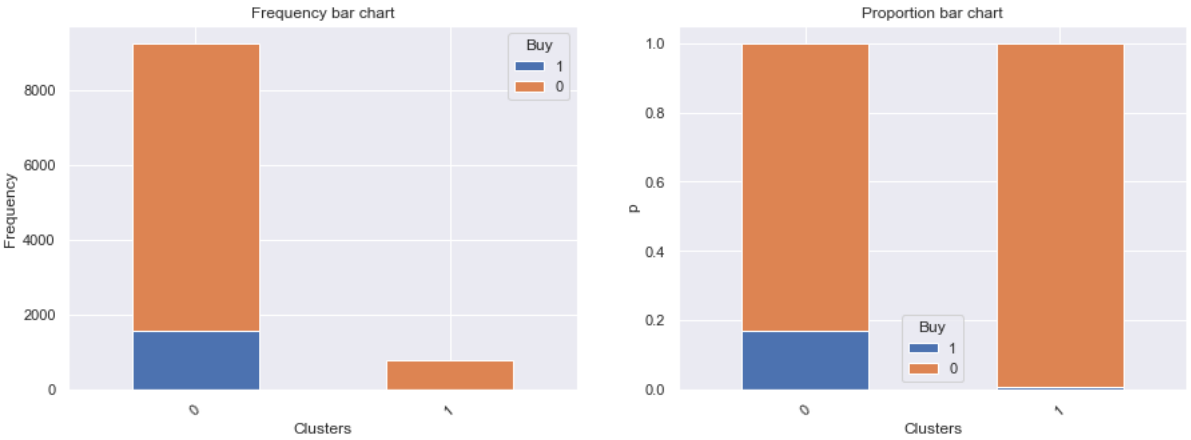


Figure 19: Discriminatory Power of the Country Feature (absolute and proportional values)

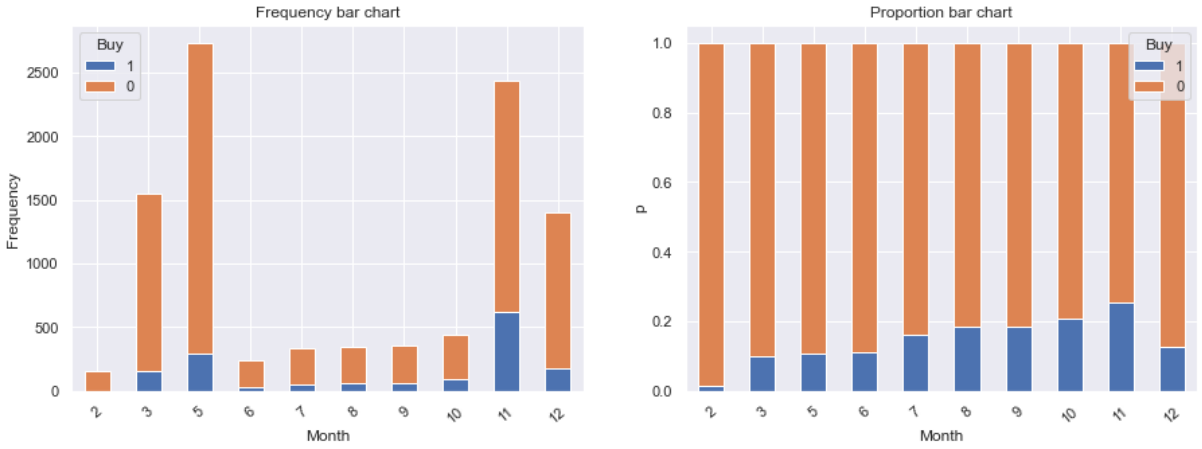


Figure 20: Discriminatory Power of the Month Feature (absolute and proportional values)

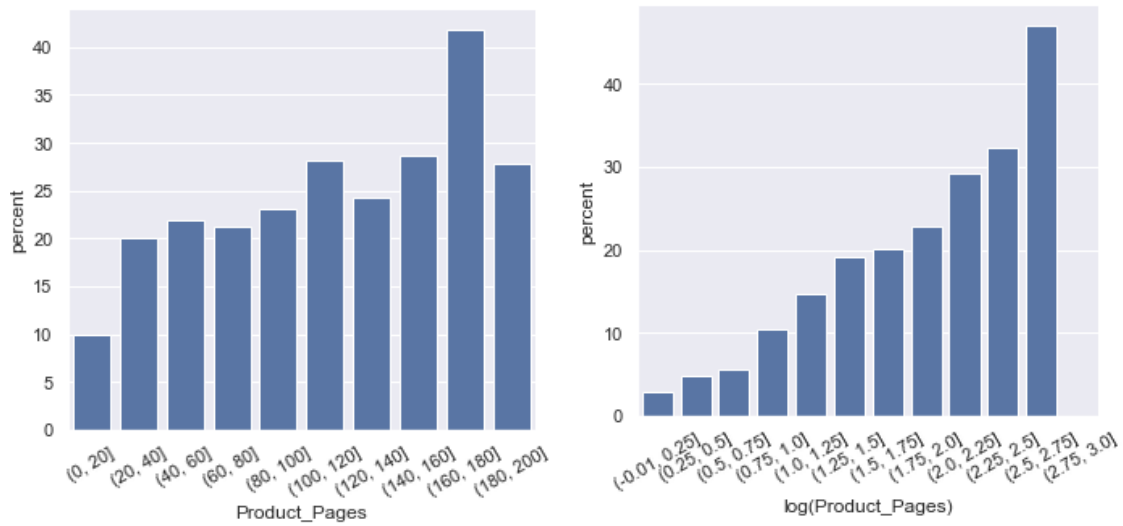


Figure 21: Proportion of Buyers per Intervals for Product_Pages and Log(Product_Pages)

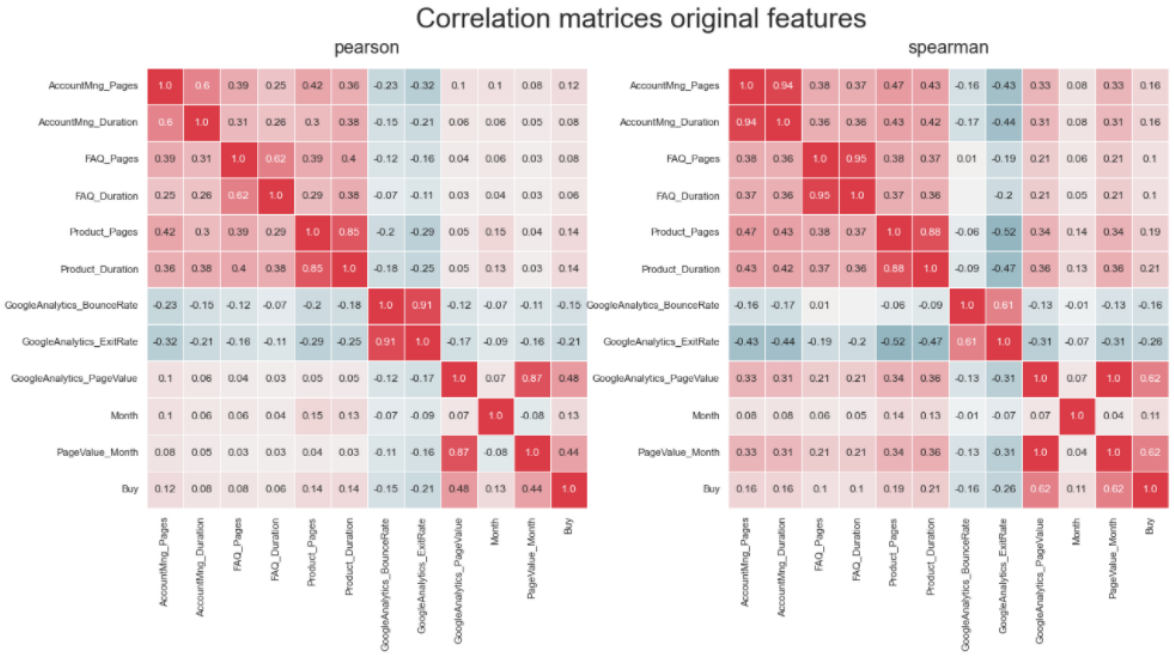


Figure 22: Pearson and Spearman Correlation matrices of the Original Features

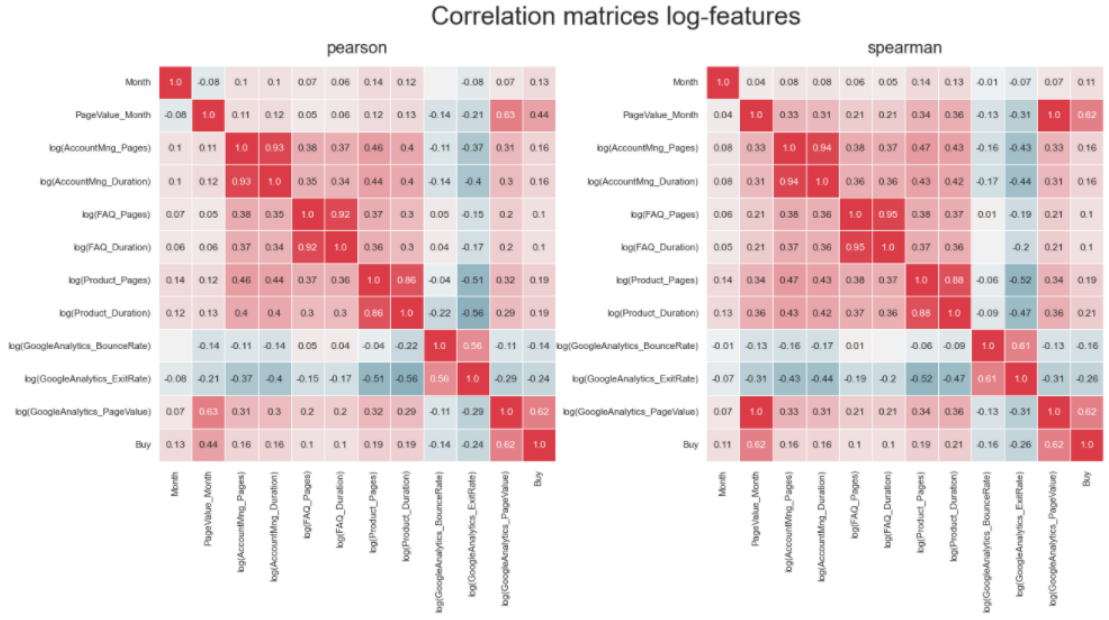


Figure 23: Pearson and Spearman Correlation Matrices of the Log-Transformed Features

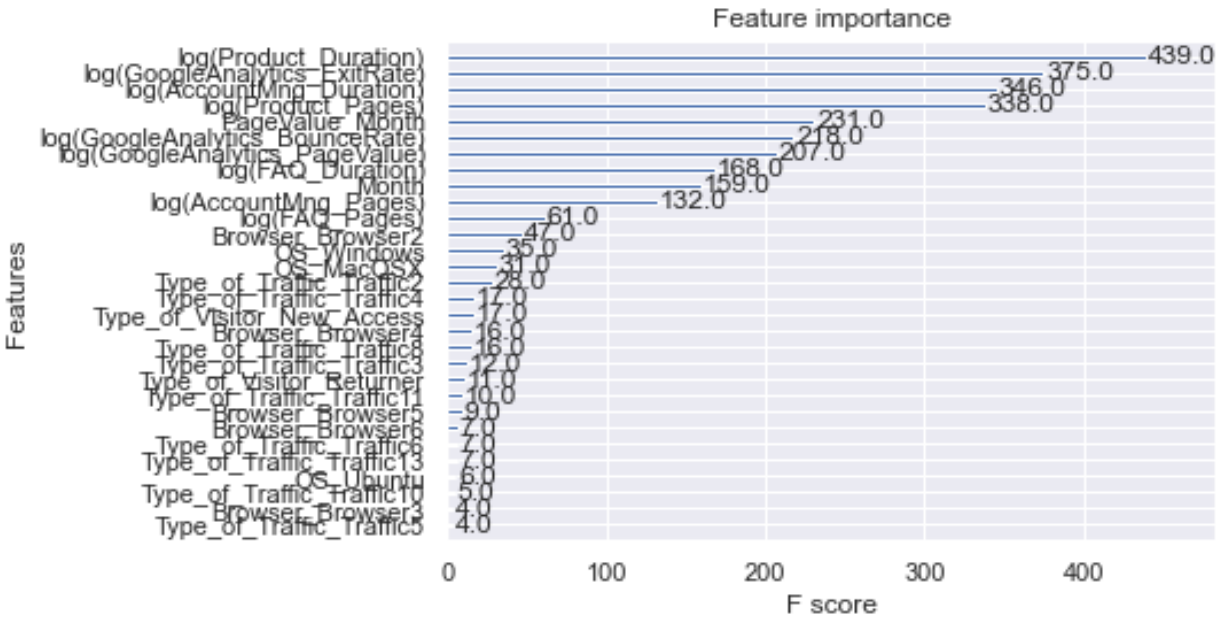


Figure 24: Assessment of Feature Importance with XGBoost

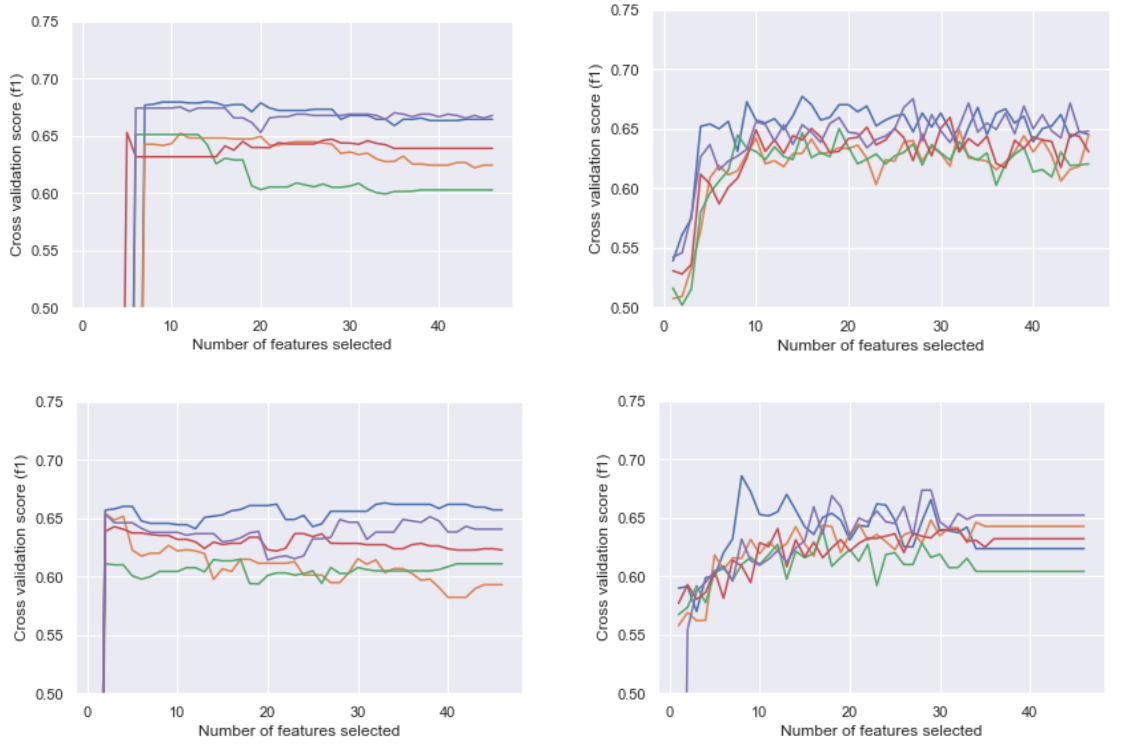


Figure 25: Assessment of Feature Importance using RFE with SVC, Random Forest, Logistic Regression, and XGBoost

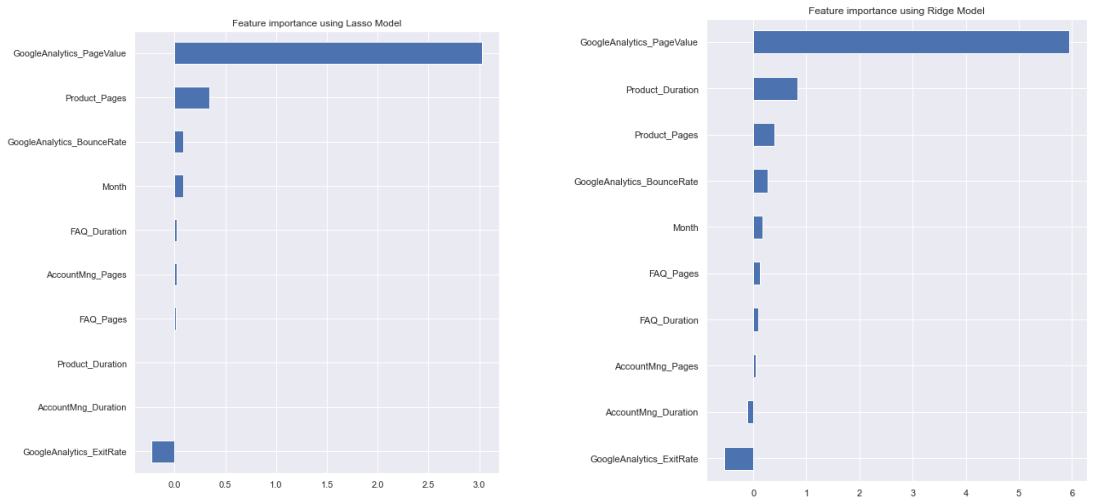


Figure 26: Assessment of Feature Importance with Lasso and Ridge Regressionű

Categorical Data

Predictor	Chi-Square
OS	Keep
Browser	Keep
Country	Discard
Type_of_Traffic	Keep
Type_of_Visitor	Keep
Clusters	Keep

Figure 27: Assessment of the Importance of Categorical Features with Chi-squared test

	Corr	RFE	Lasso_reg	Ridge_reg	XGboost
AccountMng_Pages	True	False	False	False	True
AccountMng_Duration	False	False	False	False	False
FAQ_Pages	True	False	False	False	False
FAQ_Duration	False	False	False	False	False
Product_Pages	True	True	True	False	False
Product_Duration	False	True	True	False	False
GoogleAnalytics_BounceRate	False	True	True	True	True
GoogleAnalytics_ExitRate	True	True	True	True	False
GoogleAnalytics_PageValue	True	True	True	True	True
Month	True	True	True	True	True
PageValue_Month	False	True	True	True	True

Figure 28: Comparison of Feature Selection Techniques

	Time	Train	Test
knn	0.003+/-0.0	0.667+/-0.0	0.619+/-0.03
svc	0.889+/-0.04	0.663+/-0.0	0.662+/-0.03
random forest	9.903+/-0.83	0.818+/-0.0	0.658+/-0.03
AdaBoost	1.631+/-0.05	0.661+/-0.0	0.639+/-0.02
GradientBoostingC	1.781+/-0.06	0.725+/-0.0	0.664+/-0.02

Figure 29: Grid Search Results

VIII.2. Background

This chapter discusses the theoretical background of the techniques that have not been covered in the practical classes, but are utilized in this project.

Discretization:

The goal of this technique is to transform a metric feature into a categorical. We define a desired number of bins which represent distinct intervals in a way that their union gives the interval of the

metric feature. Then, a categorical variable is created, with each of its categories corresponding to exactly one bin. Finally, each observation is assigned to a category based on the value of the selected metric feature.

Two common approaches for selecting the bins can be either to specify the number of bins and then divide the whole interval into equally-spaced parts, or to set the beginning and endpoint of each interval manually in a way that the resulting bins would have a similar number of observations.

Random Sampling:

This technique is used when dealing with an imbalanced dataset, i.e. the target variable is categorical and the categories are not equally represented in the dataset. When dealing with a binary target, the dataset can be balanced by either increasing the observations of the minority class, or decreasing the observations of the majority class. The former is called oversampling, the latter is undersampling. Usually the goal is to have 50% of observations belonging to one class and the other 50% to the other class. However, the preferred ratio of the two classes can be specified manually.

In the process of random oversampling, one observation of the minority class is duplicated in each iteration drawn randomly from the original set with replacement until the desired ratio is achieved.

In random undersampling, we consider the observations of the majority class as a population and draw a sample (a subset) from it randomly without replacement.

Under - and oversampling techniques are oftentimes combined, introducing stochasticity in both the minority and the majority class.

SMOTE Sampling:

SMOTE is an advanced oversampling technique. Instead of randomly repeating a number of observations of the minority class, SMOTE generates new synthetic samples that are similar to the original observations of the minority class, enlarging the size of the set this way. In each iteration one observation of the minority class is selected randomly. Then, we look for the K nearest minority neighbours of the given observation. Synthetic observations are introduced as points of the lines connecting the K neighbours to each other.

This process is continued until the deserved number of new observations is reached.

Sequential Feature Selection (SFS):

Sequential feature selection algorithms are basically part of the wrapper methods where it adds and removes features from the dataset sequentially. Sometimes it evaluates each feature separately and selects M features from N features on the basis of individual scores; this method is called naive sequential feature selection. It works very rarely because it does not account for feature dependence.

In a proper technique, the algorithm selects multiple features from the set of features and evaluates them for model iterate number between the different sets with reducing and improving the number of

features so that the model can meet the optimal performance and results.

Mathematically these algorithms are used for the reduction of initial N features to M features where M<N. and the M features are optimised for the performance of the model.

The sequential feature selection method has two components:

An objective function:

The method finds to minimize the number of overall features in a subset from the set of all features. So the results can be enhanced. It can be called the criterion where the mean squared error is a criterion for regression models and the misclassification rate is a criterion for the classification model.

A sequential search algorithm:

This searching algorithm adds or removes the feature candidate from the candidate subset while evaluating the objective function or criterion. Sequential searches follow only one direction: either it increases the number of features in the subset or reduces the number of features in the candidate feature subset.

XGBoost:

Extreme Gradient Boosting is a gradient boosting algorithm which can be used for both feature selection and classification (or regression). In our project, we only applied it to do feature selection. Doing so, the output of the algorithm is a score for each feature. The higher the score, the more important the feature. XGBoost generates a base estimator at each iteration and minimises the objective function with regards to the direction of the gradient. In the case of XGBoost, the weak learner is a decision tree.

K-means Clustering Algorithm:

Clustering is the unsupervised classification of patterns (or data points) into groups. K-means is considered one of the simplest algorithms that uses a squared error criterion that works well at maximising data similarity within a cluster and minimising the similarity between clusters. We apply K-means as a feature engineering technique. The optimal number of clusters is evaluated using the computed inertia values (represented graphically by the elbow plot) and supplemented by the silhouette coefficient method. The higher the silhouette coefficient in positive terms, the better-defined the clusters. For the particular set of features used, 2 clusters were identified.

Random KNN Classifier:

This method is a bagging classifier. It is based on the same idea as Random Forest, but instead of using a set of trees, we build many KNN classifiers and train each of them on a randomly selected subset of the records and features of the whole training set. The binary prediction is then obtained by

majority voting of the base classifiers. Even though a sklearn implementation of this begging classifier is available, it lacks some key parameters so for this reason we created our own implementation which enables introducing more variance than the sklearn model. In our implementation, it is possible to include stochasticity in the selection of each parameter. The number of predictors, the number of neighbours to consider for each base predictor, the number of records and the number of features are all sampled randomly and independently for each KNN classifier instance. Finally, in the sklearn method there is no option to optimise for binary F1 score, which is also possible in our version.
As a further implementational challenge, we needed to create our own cross-validation and grid search methods to find the optimal parameters for the Random KNN Classifier.

Support Vector Classification:

Support vector machines (SVMs) are supervised machine learning algorithms. It can be used for regression and classification. The latter is the case in this project. Thus, the algorithm used here is called support vector classifier. In support vector machines, the goal is to define a hyperplane in the n-dimensional space containing the training points, which separates the different classes within the target variable. In case of a linearly separable dataset, meaning that a hyperplane can be defined such that if all observations from one class are on one side and all observations from the other class are on the other side of the hyperplane, the hyperplane simply represents the border that separates the two classes and that has the maximum distance to each and every point in the dataset (measured individually), which means that only the distances to the points on each side that are closest to the hyperplane are relevant. These closest points are referred to as the “support vectors” since they support the position of the hyperplane. Because the area around the hyperplane could be imagined as a channel or street separating the dataset, this is also called the “widest street approach”. In case of a non-linearly separable dataset, a penalty can be added during the calculation of the optimal solution, such that points that are “on the wrong side” of the hyperplane are penalised in a way that is adequate for the application. This is called the “Soft Margin” approach (see chapter 7.1.1 in [3]). In applications, where the dataset contains patterns where a hyperplane is not able to create a satisfactory division, such as when a ring of points of one class surrounds a cluster of points of the other class, the n-dimensional space of the data points can be mapped to a higher-dimensional space in order to achieve a better separability. This is called the “Kernel Trick” (see chapter 6 in [3]).

Adaboost Classifier:

Adaptive Boosting (AdaBoost) is a Boosting technique used in Machine Learning as an Ensemble Method. This is called Adaptive Boosting because the weights are re-assigned to each instance, with higher weights assigned to incorrectly classified instances. Boosting is a technique used to reduce bias and variance in supervised learning. This technique is based on the principle of sequential growth. Every successive learner is sourced from a previously grown learner, with the exception of the first. In other words, weak learners become stronger. AdaBoost functions similarly to boosting, but

a little differently.

It makes ‘n’ number of decision trees during the data training period. When the first decision tree/model is made, the incorrectly classified record in the first model is given priority. Only these records are used as input to the second tree/model. The process continues until we specify how many base learners we want to create. Remember, repetition of records is allowed with all boosting techniques.