Techscape | E-commerce Project Report

Machine Learning 20/21

Group 40

Ana Vera Prada (m20210755) | Débora Carreira (m20210757)

Francisco Costa (m20211022) | Hugo Fernandes (m20210682) | Martim Figueira (m20210686)

Abstract

This report is focused on explaining all the steps, techniques, methodologies, and conclusions of a predictive model built to predict if a customer will buy a product (1) or not (0). Based on the online behavior of the customers, the goal of the project is to answer this main question for Techscape, a startup that is selling goods related to digital detox via e-commerce.

In the following report, we will approach the techniques used to perform the mentioned predictive model to be evaluated with F1-Score and built with both numerical and categorical data. So, this project will go through topics as Data Cleaning, Data Visualization, Outliers, Feature Engineering, Feature Selection, Feature Decomposition, Feature Enconding and Feature Transformation in the context of data treatment with the purpose of pursuing the best predictive model.

Allong the report, we will also dig into some models and techniques as K-Nearest Neighbors, Random Forest, Ada-Boost, Gradient Boost, HistGradientBoost, GridSearchCV, Voting Classifier, Stacking Classifier, Logistic Regression, Gaussian Naive Bayes, Categorical Naive Bayes and some others.

Keywords: Machine Learning, Predictive Model, Classification System, Binary Classification, Supervised Learning.

1.Introduction

Techscape is a startup which business core is selling goods related to digital detox, in order to reduce the nowadays people's biggest addiction: spending the days in social media, affecting the daily productivity, mental health and life in general. With some financial problems due to Covid-19, Techscape felt the need to overcome and increase online sales – a commercial channel with a major potential in this pandemic period. To achieve that, it was necessary to analyze the customers' online behavior to predict which customers with a certain behavior would by the products. And this is the main goal of this project. Having those patterns of behavior recognized, we can predict the buyers and some actions in terms of marketing or management strategies can be made by the startup.

With the dataset that contains all the customers' data (9999 observations), our goal is to build a classification system, that is able to predict if a customer will buy (1) or not (0) at least one product from Techscape. In that sense, our target variable

is “Buy” and will depend on other sixteen features regarding the customer online behavior and some other demographic and technical data.

We aim to achieve a steady predictive model, that can perform and predict in the test dataset, making use of all important techniques within scikit-learn package.

2.Exploratory Data Analysis

2.1. Data Cleaning

It’s vitally important to start the problem by checking the data and evaluate if data cleaning is needed, in order to get better results. Regarding the data types of the features, we decided to split the ‘Date’ feature in ‘Month’ and ‘Weekday’ to simplify the analysis. Additionally, we created a column ‘Weekend’ which dictates if the access occurred on a weekend (1) or not (0). We checked for missing values in the dataset as well. In this case, there were no missing values to replace, and we were able to proceed with the dataset.

2.2. Data Visualization

We began our analysis by checking the correlation matrix between the numerical features(Fig 1) showing us that **AccountMng_Pages** has a moderated correlation with **AccountMng_Duration** so does **FAQ_Pages** with **FAQ_Duration**. There are some strong correlations between **Product_Pages** and **Product_Duration** and between **GoogleAnalytics_BounceRate** and **GoogleAnalytics_ExitRate**. Looking to the target variable Buy, we can see that only **GoogleAnalytics_PagesValue** has a moderated correlation with it, the rest doesn’t seem to have any or has a very weak correlation.

Knowing the correlation, we decided to check the skewness and kurtosis to start understanding some data behavior. The skew gave results greater then 1 so all the features are skewed positively, and the kurtosis only gave positives results, creating the idea that the data is mainly concentrated near the origin, and it is spread along the positive axis, this might be a sign for possible outliers. Applying a boxplot (Fig 2) and histogram (Fig 3) to each variable, we can see that in all the cases there are a lot of data points over the right whisker, and the 75% from the data is relatively close to the origin.

After this, we decided to start looking to the target variable Buy, applying a pie chart (Fig 4) in order to get our dataset buying rate, which is around 16% and that makes our dataset unbalanced, so we decided to do some analysis to check if this rate is kept along the features. Looking at the categorical features countplots(Fig 5, Fig 6 and Fig 7) the majority of the features’ values, seem to preserve this buying rate. However, there are some values, from example the month of November, where the buying rate assumes a different behavior and they might be used later when creating the model.

Looking at the numeric features, applying a several scatterplots, the points representing the buyers and the non-buyers are mixed across all the distribution in all features, except in the **GoogleAnalytics_PageValues** where, for values greater than zero, we can see that most of the observations are representing buyers.

2.3. Outliers

Regarding outliers, we had to evaluate if we would drop, replace those values or keep them.

We checked for outliers in each feature, observing the percentage of outliers in the total of the data for each feature. In the end, we decided not to remove or replace the outliers because it would lead to information loss. For instance, a high Product Duration, could mean that a user let the page open and didn't end up purchasing, so that could be an outlier but could also be an important observation.

3.Data Preparation

3.1. Feature Engineering

After we did all the exploration analysis, we though it could make sense create some new variables from the previous ones. As a result, we had created all the Duration variables divided over the Pages variables, to get the average time rate that the user had spent in each page. After running the feature selection, we realized that these features that we created were not important, so we decided instead of making that division, applying a product between Pages and Duration Variables. So then, as these ones were not discarded by feature selection algorithms, we decided to keep them.

3.2. Feature Selection

The feature selection was applied to one of the 2 models that were developed in parallel, that will be explained in detail below. These models are practically the same, with the difference that one of them uses only the variables selected by the methods/techniques and the other does not.

We started by testing the significance of our variables in relation to the target variable using the Chi-Square test for categorical variables. For numeric variables, we used the Shapiro-Wilk when the variable follows a normal distribution, and otherwise, we used the nonparametric Mann Whitney U test. After that, we applied the Variance Inflator Factor method to measure how much the behavior (variance) of an independent variable is influenced, or inflated, by its interaction/correlation with the other independent variables. Using the good practice rule of thumb of a 10 VIF score, we would iteratively remove features until the remaining ones had a smaller score than this threshold, but as all the variables had scores under 10, we kept them all. The last feature selection approach was implementing Recursive Feature Elimination with Cross Validation (RFECV). This method fits a model (Random Forest was the chosen model for us, in this case), and removes the weakest features recursively until the optimal number of features is reached, according to the score of the algorithm. This method has the advantage of automatically selecting the number of variables. However, with this method, we also can lose some control over the extraction.

After all these techniques were applied and tested, we compared the result of each one. Given the results, we chose to firstly remove the most correlated features with VIF method and then applied the RF-RFE to select the most important features automatically, within 10 folds of stratified cross-validation, 3 times. We ended up with all the variables except for FAQ Duration, FAQ pages and Country variables.

3.2. Feature Decomposition

After our feature selection analysis, it was time to test a feature decomposition technique. Unlike feature selection, the goal of feature decomposition is to decompose the original set of features into several subsets. The main goal of the attempt of applying this technique was to try to reduce the noise and try to have different components (like variables) completely independent of each other. As we had mixed different data types, and PCA can only work with continuous variables, we tried to apply FAMD – Factor Analysis Mixed Data. We ended up not following this approach because we did not manage to get a number of components smaller than the number of variables that all together would explain at least 80% percent of total variance.

3.3. Feature Enconding

As almost all the algorithms cannot support categorical variables, we had to encode all these categorical variables. As a result, we used One Hot Encoding for this purpose. The variables that were encoded were Browser, OS, Month, Type_of_Traffic and Type_of_Visitor.

3.4. Feature Transformation

An assumption often taken by Gaussian Naive Bayes algorithm is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. Following this idea, we checked the skewness of our variables and we realized that some of them were highly skewed. As a result, we've applied a **Box-Cox transformation** to all numerical variables. It is important to emphasize that we only applied this transformation to data that was strictly used to this algorithm, and for all others we kept raw distributions.

4.Models

After having clean and prepared data to perform a good model, we began by exploring several models and how they scored across F1-Score. A big decision we had to make was to balance or not the training, since this one was highly imbalanced. To perform this, we resorted to SMOTESVM package that creates artificial samples of the data it receives using the K-Nearest Neighbors algorithm. It is important to notice that, to evaluate and tune the parameters of the models, we use cross validation that, in each fold, splits a part of the training set as validation. So, if we generated artificial samples on the entire training set, we would be biasing the data that would later be selected as validation, and that would cause the overfitting of the model. To counter this effect, we had to build a pipeline that receives each model and only oversamples the fold selected as training, leaving the validation intact.

Following the idea mentioned above we split our model performance exploration in 2 parts:

- Data with all the variables. (1)

- Data with the features selected by our feature selection process. (2)

Comparing the performance of those several methods across (1) and (2) we considered that the results didn't differ so much and the results with (1) were slightly higher, so we kept going our process applying hyper-parameter optimization in the algorithms that gave best results when the initial exploration was done on the (1). This hyper-parameter optimization was done by looking for possible great parameters' values (one by one) and then, after got restricted the range of possible values for each parameter, applying a GridSearchCV. The algorithms were: Random Forest, Gradient Boost, Ada-Boost and an algorithm that was not lectured in class and will be explained in annex, HistGradientBoost.

After that, we applied these best models tunned on smoted data, we checked that this class imbalance technique did not improve our models' performance, so we established to keep our data as itself.

Then, in order to improve the model, we decided to apply a Voting Classifier to our data with our best models tunned. In addition, we applied a Stacking Classifier with the best models tunned as well and lastly the meta classifier, a Logistic Regression.

In the end, after we did some adjustments on the voting classifier weights, we ended up choosing the Stacking Classifier as the final one. This choice came up since this classifier was being strongly consistent and was giving the best results.

4.1. Final Model

Models used by the Stacking	Models' best parameters
Ada Boost Classifier	-algorithm = SAMME-base_estimator = DecisionTreeClassifier(max_depth=1) -learning_rate = 0.001-n_estimators = 17
Random Forest Classifier	-class_weight = { 0 = 0.1 , 1 = 0.25 }-criterion = entropy-max_depth = 11-max_features = 0.8-max_samples = 0.35-min_samples_leaf = 6-min_samples_split = 28-n_estimators = 19
Grandient Boosting Classifier	-learning_rate = 0.1-loss = deviance-max_depth = 4-max_features = 22-min_samples_leaf = 51-min_samples_split = 20-n_estimators = 44-subsample = 1
Histogram Gradient Boosting Classifier	-l2_regularization = 0-learning_rate = 0.1-loss = auto-max_bins = 15-max_depth = 4-max_iter = 40-max_leaf_nodes = 14-min_samples_leaf = 160

Using the Stacking Classifier with these models optimized, we were got:

- F1-Score for train dataset = 0.7018
- F1-Score for validation dataset = 0.6853

5.Other Approaches

After computing the feature importance from one of the models that was giving better results (Random Forest), we verified that the GoogleAnalytics_PageValue variable was dominate. Moreover, in 77% of the data, this variable had a value of 0, and only 3% corresponded to accesses that had a PageValue of 0 and had concluded the purchase. That seemed to be a good discrimination between the accesses that presented a PageValue of 0 and those which presented a value higher than 0, so we decided to split the dataset in 2: GoogleAnalytics_PageValue = 0 and GoogleAnalytics_PageValue > 0, hoping that in each one of the subsets would be other variables that would gain a greater importance and that would somehow, help to better predict. We developed a model for each of the subsets. We made the predictions and we ended up joining them. We decided not to follow this approach because none of the models had an acceptable performance in predicting the subset corresponding to GoogleAnalytics_PageValue = 0.

When we were looking to each model, we thought that would be a good idea to, after applying the Random Forest Classifier, find a range in prediction probabilities where the predict sucess rate is lower than 50%, and try to train a model to specifically predict these observations. In our case that range was from 50% to 80%. These were the probabilities associated with accesses that ended up purchasing (Buyers). We tried other classifiers as Random Forest, MLP and KNN but none of them gave any standout results, even with the parameters tunned by RandomSearchedCV, so we didn't go through with this approach.

After applying the Box Cox transformation, we started building our Gaussian Naive Bayes but we had some categorical variables that we did not want to ignore, so we did a Naïve Bayes for both numeric and categorical features. In this approach we decided not to encode our categorical features because, as we used one-hot encoding, we were basically dealing with boolean features. In other words, each feature is following a Bernoulli distribution. In that sense, we applied the Gaussian Naive Bayes to the numerical data and the Categorical Naïve Bayes to the categorical data, aiming to get the predictions probabilities accordingly to the numerical data and different probabilities accordingly to the categorical data. To each observation, we just saved the 'not buying' probabilities for both numerical and categorical data and created a new Gaussian Naive Bayes in order to predict accordingly these two probabilities. Unfortunately, the F1-Score in the validation data was too low, that we had to discard this model.

6.References

Histogram-Based Gradient Boosting Ensembles in Python

https://machinelearningmastery.com/histogram-based-gradient-boosting-ensembles/ , Accessed on 14/12/2021

Ordinal and One-Hot Encodings for Categorical Data (machinelearningmastery.com)

https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/ , Accessed on 14/12/2021

Factor Analysis of Mixed Data. Use of FAMD for data having continuous… | by Md Sohel Mahmood | Towards Data Science

https://towardsdatascience.com/factor-analysis-of-mixed-data-5ad5ce98663c , Accessed on 16/12/2021

The right way of using SMOTE with Cross-validation | by KSV Muralidhar | Towards Data Science

https://towardsdatascience.com/the-right-way-of-using-smote-with-cross-validation-92a8d09d00c7 , Accessed on 16/12/2021

7.Annex

7.1. Histogram-based Gradient Boosting Classification Tree

Histogram-based Gradient Boosting Classification Tree is a different Gradient boosting technique, where the numerical data is discretized or binned into a fixed number of bins. This allows the decision tree to operate upon the ordinal bucket (an integer) instead of specific values in the training dataset.

In this model, beyond the existent parameters in Gradient Boosting, we have:

- max_iter - The maximum number of iterations of the boosting process, i.e. the maximum number of trees for binary classification.
- l2_regularization - Regularizer on the loss function and corresponds to the regularization rate (lambda) in equation (2) of XGBoost.
- max bins - The maximum number of bins to use for non-missing values. Before training, each feature of the input array X is binned into integer-valued bins, which allows a much faster training stage. In addition to the max_bins bins, one more bin is always reserved for missing values. Must be no larger than 255.

7.2. Data Visualization

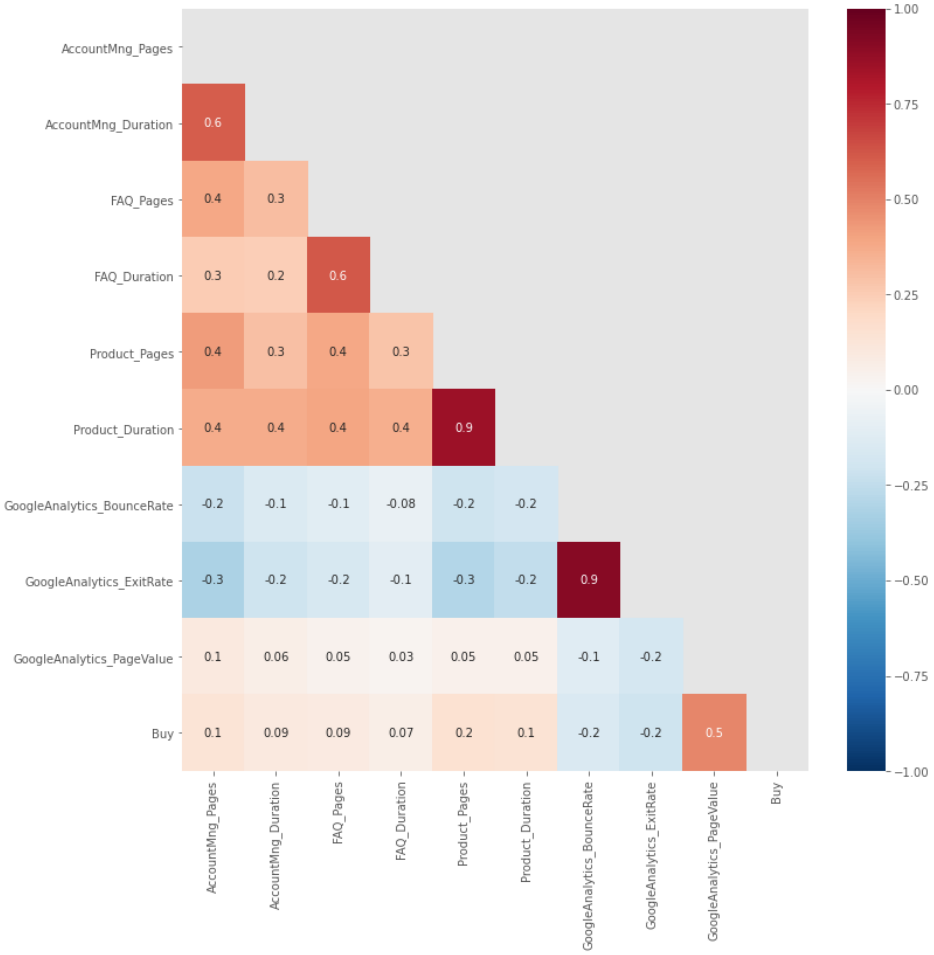

Fig 1 – Correlation Matrix

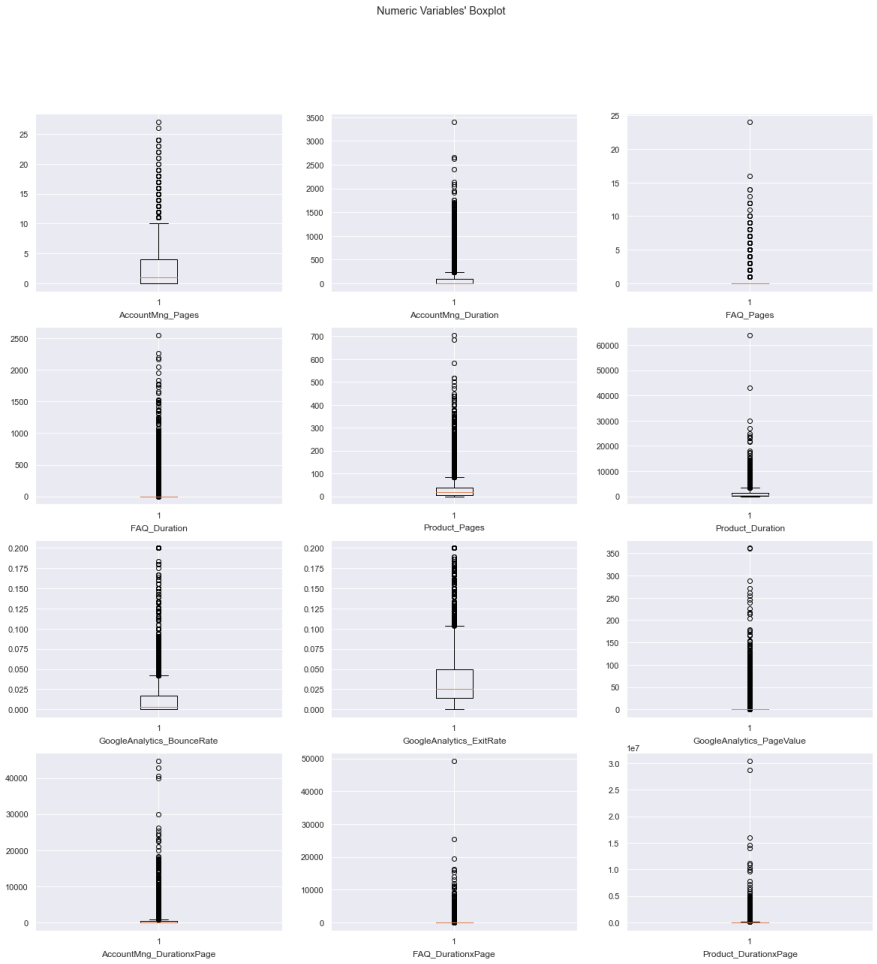

Fig 2 –Numerical features Boxplot

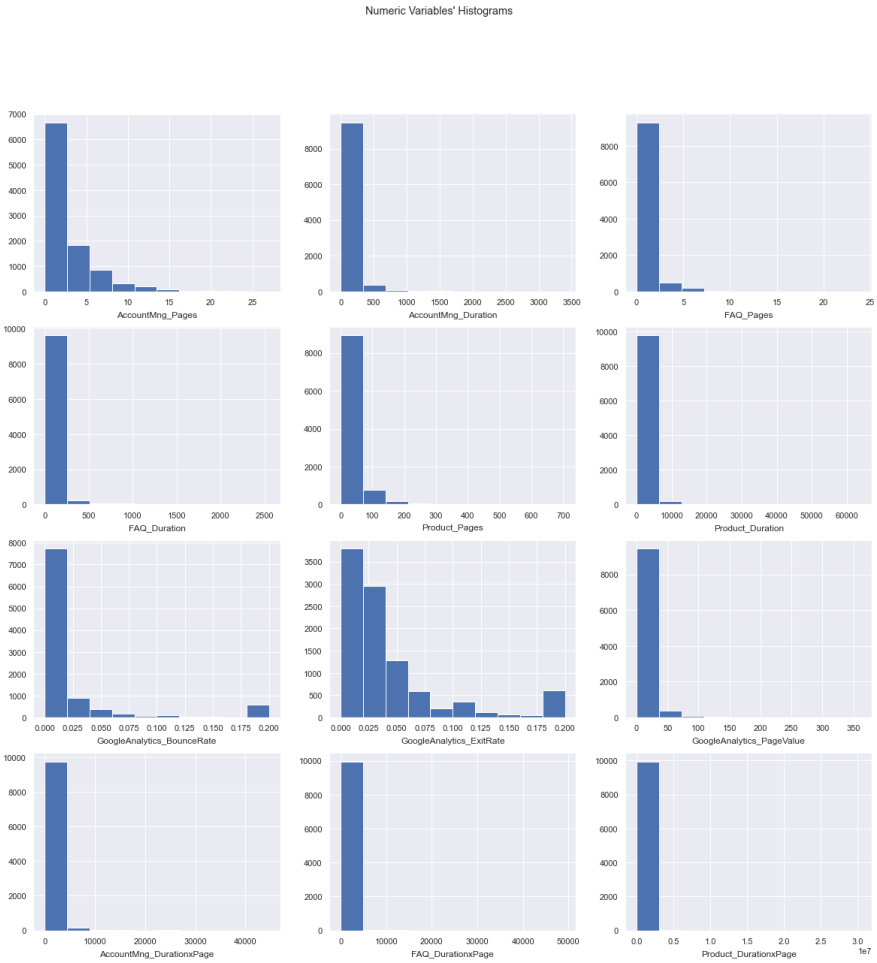

Fig 3 - Numerical features histogram

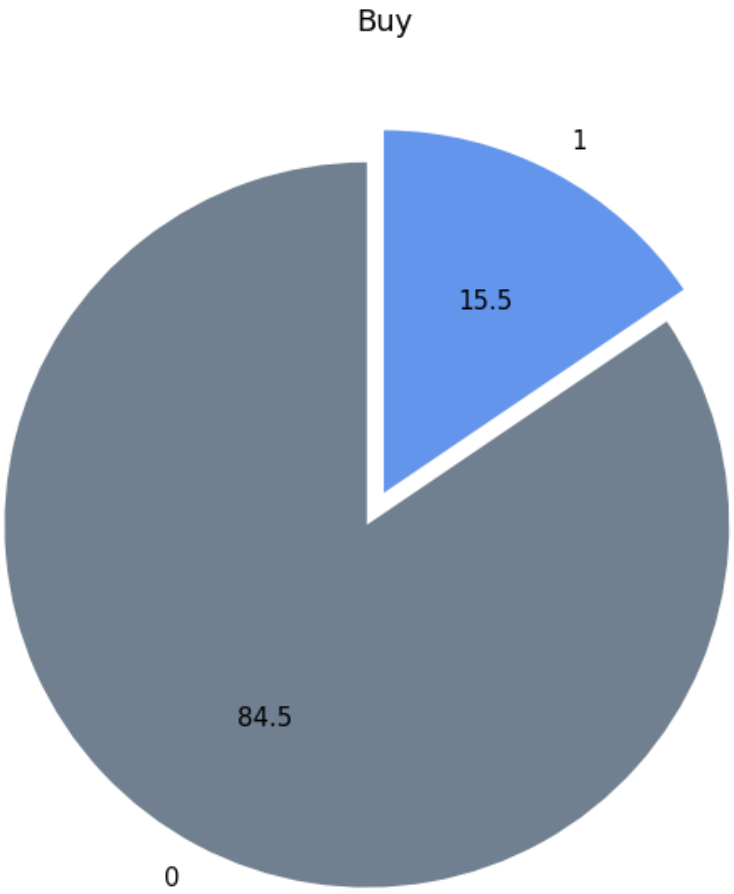

Fig 4 – Buy variable piechart

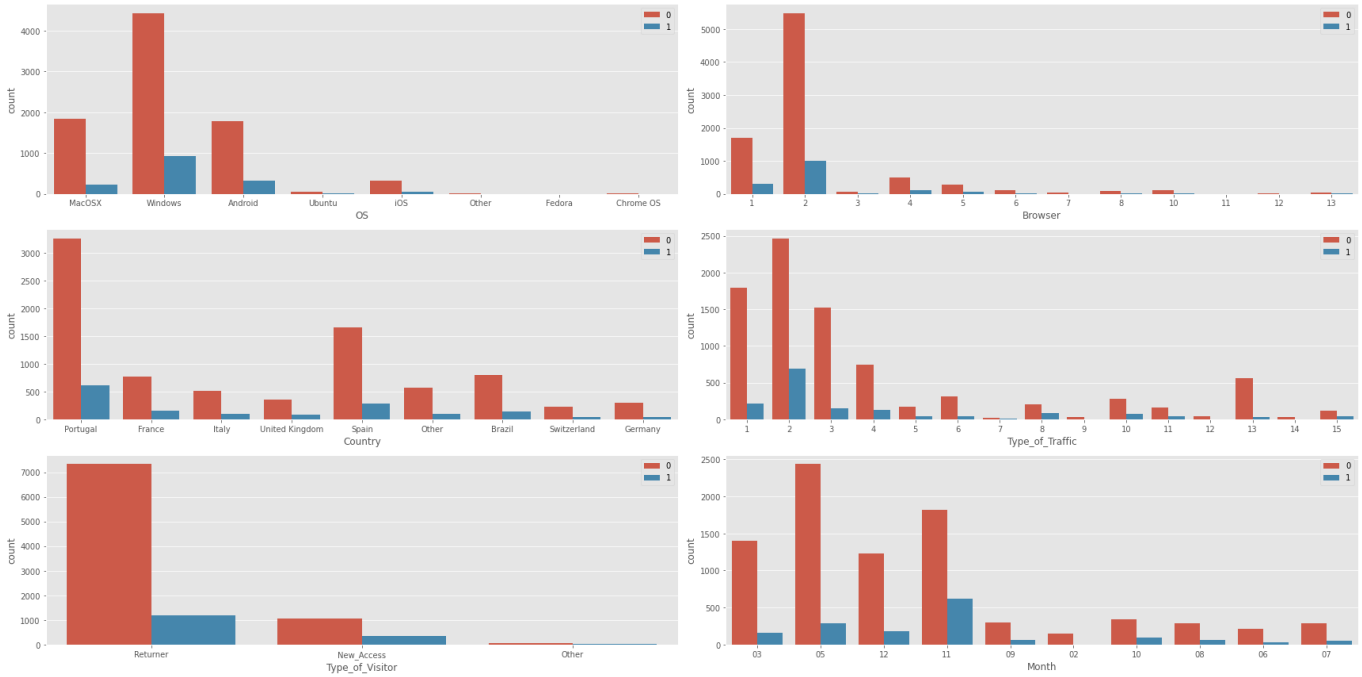

Fig 5 – Countplot for categorical features

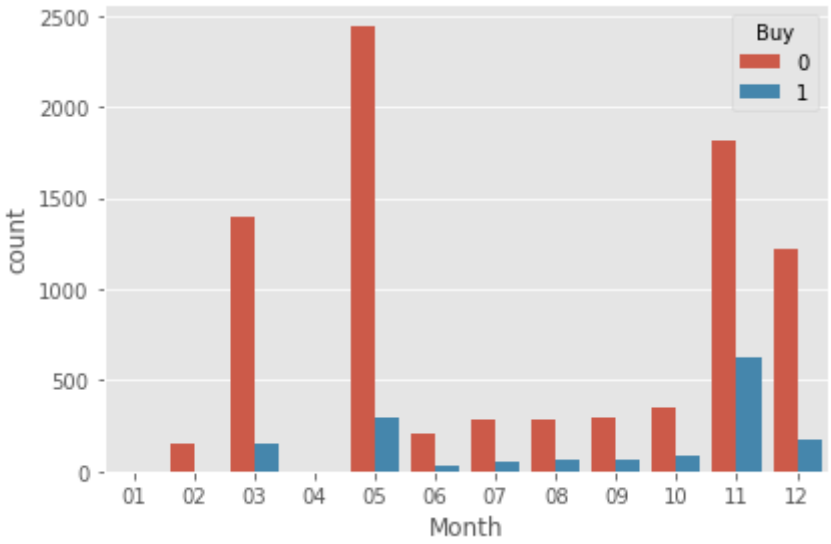

Fig 6 – Countplot for feature Month

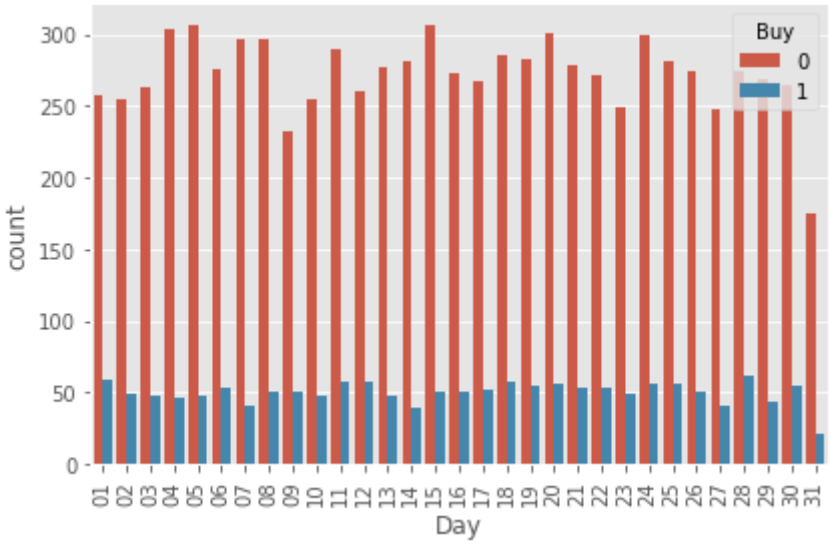

Fig 7 – Countplot for feature Day

Fig 8 – Pairplot with scatterplots for numerical features