Project II: Classification and Clustering with Bank Customer Data

Student Name: Ioannis Triantafyllakis

Student Id: f2822115

Course: Statistics for Business Analytics II

Semester: Winter Semester

M.Sc.: Master of Science in Business Analytics (Full-time)

Athens 2022

TABLE OF CONTENTS

Introduction

In this assignment, I used a dataset containing bank customers' data which was collected through phone calls. The data includes various variables about each customer and whether they Subscribed to a bank's service or not. The aim of this assignment is to try 4 different classification methods in order to predict whether a customer will subscribe or not, taking into consideration the data about them. The classification methods I used are the following: K-Nearest Neighbor, Random Forest Classification, Naïve Bayes Classifier, and Support Vector Machines. To compare each method, the accuracy metric will be used. In the end of the assignment, I also attempt to cluster the customers by taking into account specific variables about them. To cluster the customers, I will use the Partitioning Around Medoids method, I will also attempt to characterize each cluster by its demographics and average customer profile.

Input variables

As I already mentioned, the data relate to telemarketing phone calls whose aim was to sell a retail bank service (long-term deposits). The input variables and their explanations are the following.

Input variables**:**
bank client data:
1 - **age** (numeric)
2 - **job** : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - **marital** : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - **education** (categorical: basic.4y','basic.6y','basic.9y', 'high.school','illiterate','professional.course', 'university.degree','unknown')
5 - **default**: has credit in default? (categorical: 'no','yes','unknown')
6 - **housing**: has housing loan? (categorical: 'no','yes','unknown')
7 - **loan**: has personal loan? (categorical: 'no','yes','unknown')

related with the last contact of the current campaign:
8 - **contact**: contact communication type (categorical: 'cellular','telephone')
9 - **month**: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - **day_of_week**: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - **duration**: last contact duration, in seconds (numeric).

other attributes:
12 - **campaign**: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - **pdays**: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - **previous**: number of contacts performed before this campaign and for this client (numeric)
15 - **poutcome**: outcome of the previous marketing campaign (categorical:

'failure','nonexistent','success')
social and economic context attributes
16 - **emp.var.rate**: employment variation rate - quarterly indicator (numeric)
17 - **cons.price.idx**: consumer price index - monthly indicator (numeric)
18 - **cons.conf.idx**: consumer confidence index - monthly indicator (numeric)
19 - **euribor3m**: euribor 3 month rate - daily indicator (numeric)
20 - **nr.employed**: number of employees - quarterly indicator (numeric)

Output variable (desired target):
21 - **SUBSCRIBED** - has the client subscribed a term deposit? (binary: 'yes','no')

Exploratory Data Analysis

After importing the dataset in RStudio, the first step is to use the "str()" function of R to take a look at the data. It can be seen, that our variables are either of "int" or "chr" or "num" data type. After that, I use the "is.na()" and "is.null()" functions of R to see if any null values or NA's exist, only to find out there are not any of them. Next, I convert all the variables of "int" data type to "num" and all the variables of "chr" data type to "factor" data type.

Then, I use the "summary()" function of R to see some summary statistics for the data. At first, I can notice that "duration", "campaign" and "previous" appear to have some outliers and are quite asymmetric (their means and medians have a decent difference). To further understand the variables, I will construct their frequency distribution bar plots. Before doing so, I will separate the data to 2 different dataframes, one containing the "numeric" variables only (data_num), and one containing the "factor" variables only (data_cat).

In *figure 1*, the bar plots of variables "age", "duration", "campaign" and "pdays" can be seen. It is visible that "pdays" variable has some outliers and it is not symmetric at all. Also, "duration" and "campaign" variables are not symmetric at all too and are quite far from approaching a normal distribution, as we already found out by using the "summary()". The variable "age" appears to be much more symmetrical, and as we found out by using "summary()" once again, the mean and median value of "age" are not far from each other.

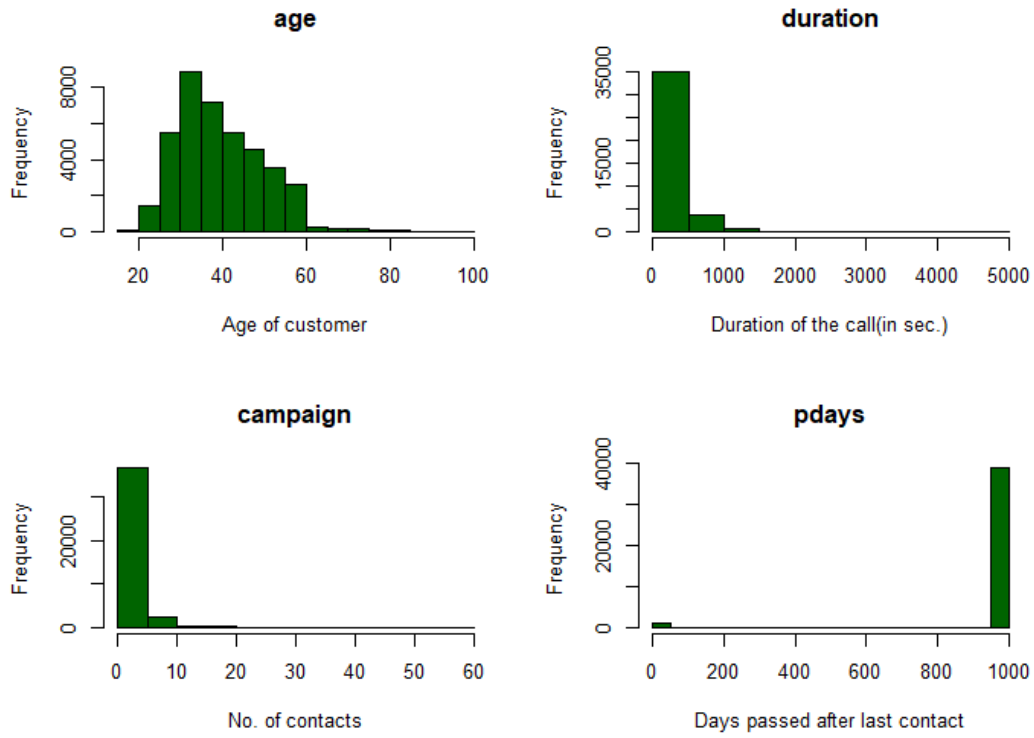

Figure 1: Frequency Distribution Bar Plots of variables "age", "duration", "campaign", and "pdays".

In *figure 2*, we can see the bar plots of variables "previous", "emp.var.rate", "cons.price.idx", and "cons.conf.idx".

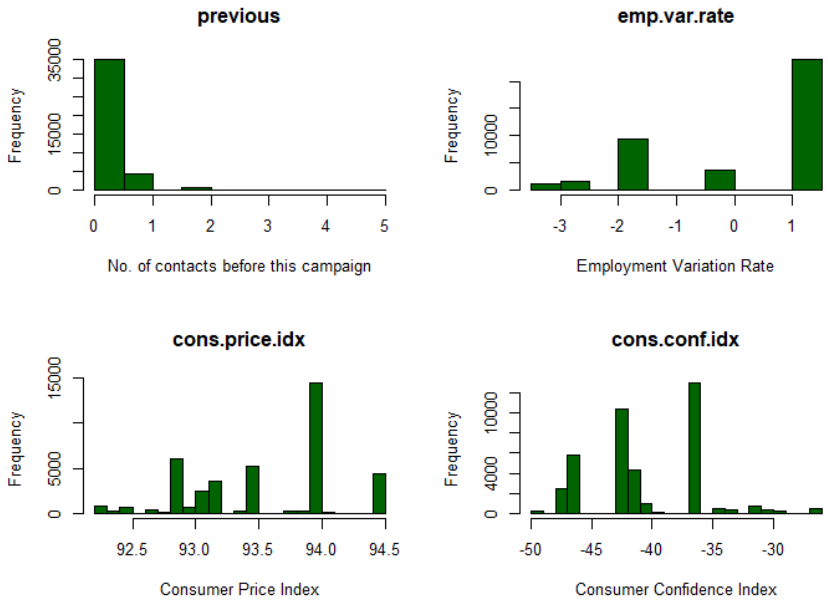

Figure 2: Frequency Distribution Barplots of variables "previous", "emp.var.rate", "cons.price.idx" and "cons.conf.idx".

In *figure 2*, we can clearly see that variables “emp.var.rate”, “cons.price.idx” and “cons.conf.idx” are quite dispersed and do not appear to follow the pattern of any specifically known distribution. They also appear to be asymmetric, and we can also assume that no outliers exist. Regarding the “previous” variable, it seems that values are quite concentrated, and no outliers can be detected.

In *figure 3* we can see the distribution bar plots of variables “euribor3m” and “nr.employed”.

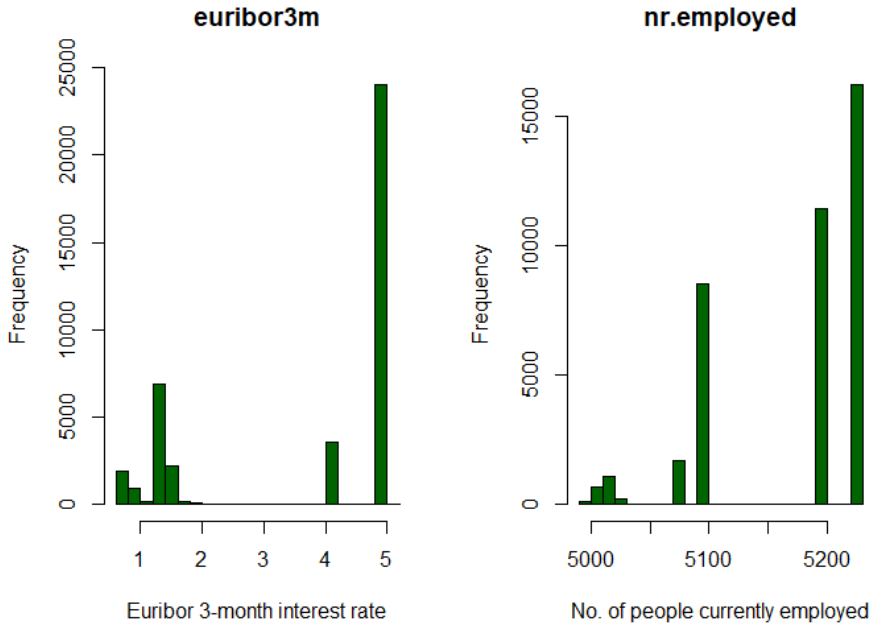

Figure 3: Distribution Frequency barplots of variables “euribor3m”, and “nr.employed”.

It is visible that both variables in *figure 3* are dispersed, asymmetric, and no outliers can be detected.

In *figure 4* we can see the distribution bar plots of variables “job” and “marital”.

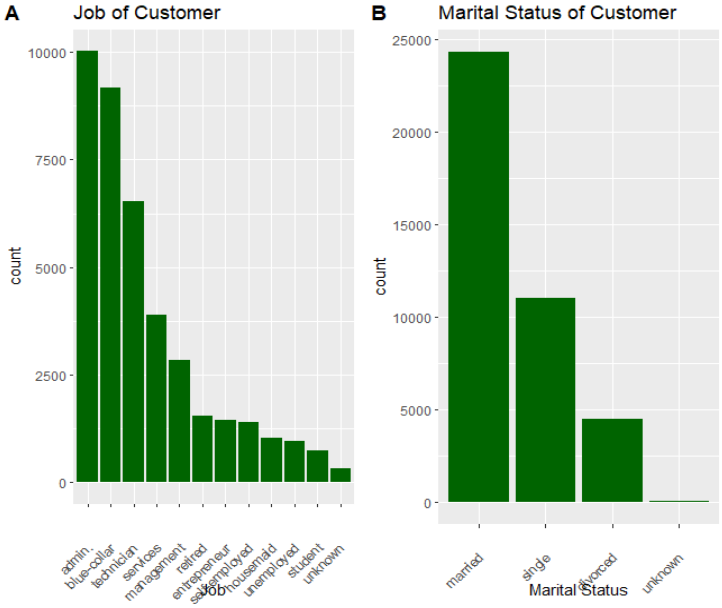

Figure 4: Distribution bar plots of variables “job” and “marital”.

In *figure 4* we can see that "job" variable has some jobs that are heavily represented (e.g. "admin") while some others are under-represented (e.g. "student"). Additionally, some jobs could belong to more than one category. For example, we do not know if some entrepreneurs chose the option "entrepreneur" while some other entrepreneurs chose the option "self-employed", and both options could technically be correct.

In the following figure (*figure 5*), we can see the distribution bar plots of variables "education" and "default".

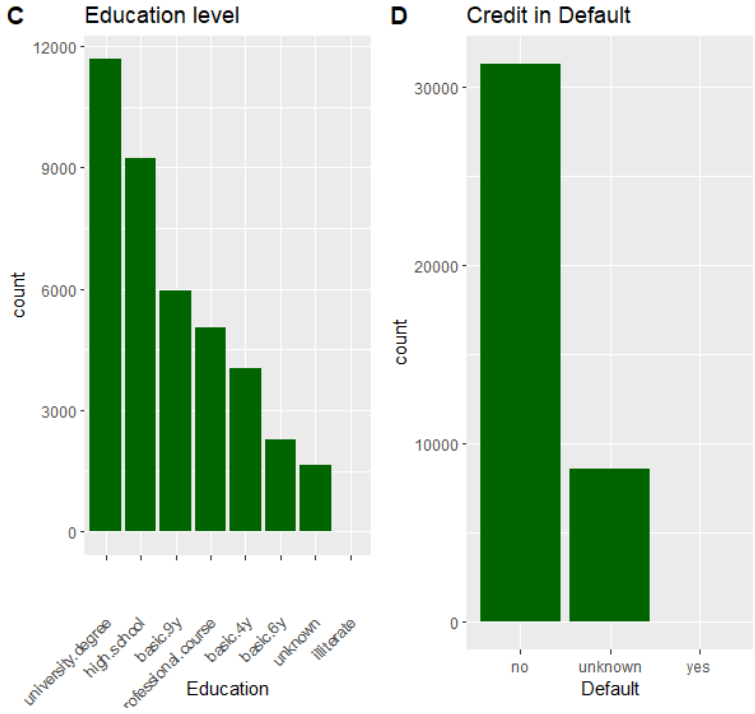

Figure 5: Distribution Bar Plots of the variables "education" and "default".

In the figure above (*figure 5*), we can see that in the "education" variable, some values are slightly over-represented once again, but not on an extreme degree. Regarding the "default" variable, it is noticeable that there are zero responders to the "yes" option. This could either mean that records with the "Yes" option were somehow lost, or that the telemarketing calls targeted only customers that did not have any credit in default on purpose.

In the following figure (*figure 6*) we can see the distribution bar plots of the variables "housing" and "loan". It is easy to observe that in both variables, the "unknown" choice is underrepresented. Additionally, the "yes" option in the "loan" variable is slightly under-represented. That could indicate that telemarketing call agents targeted potential customers that do not have a loan.

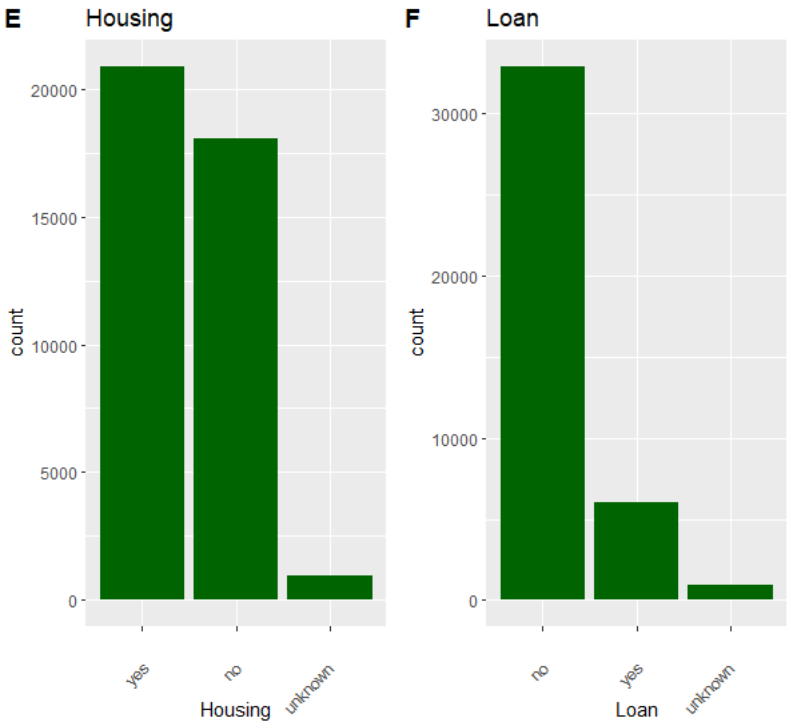

Figure 6: Distribution Bar Plots of variables "Housing" and "Loan".

In the next figure (*figure 7*) we can see the distribution bar plots of the variables "contact" and "month".

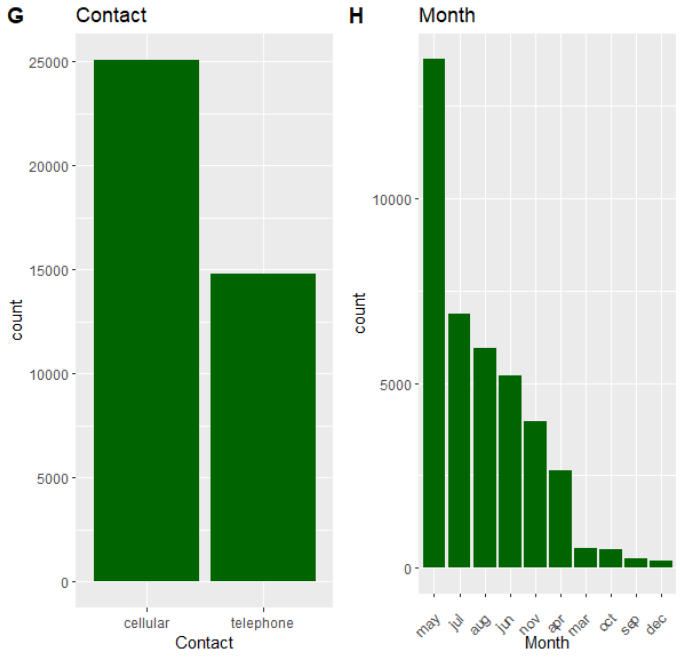

Figure 7: Distribution bar plots of the variables "contact" and "month"

In *figure 7* we can see that the "month" variable has 4 values that are severely under-represented and that the majority of calls took place in "may". Regarding the "contact" variable, both options are well represented.

In the following figure (*figure 8*), the distribution bar plots of variables "day_of_week" and "poutcome" can be seen.

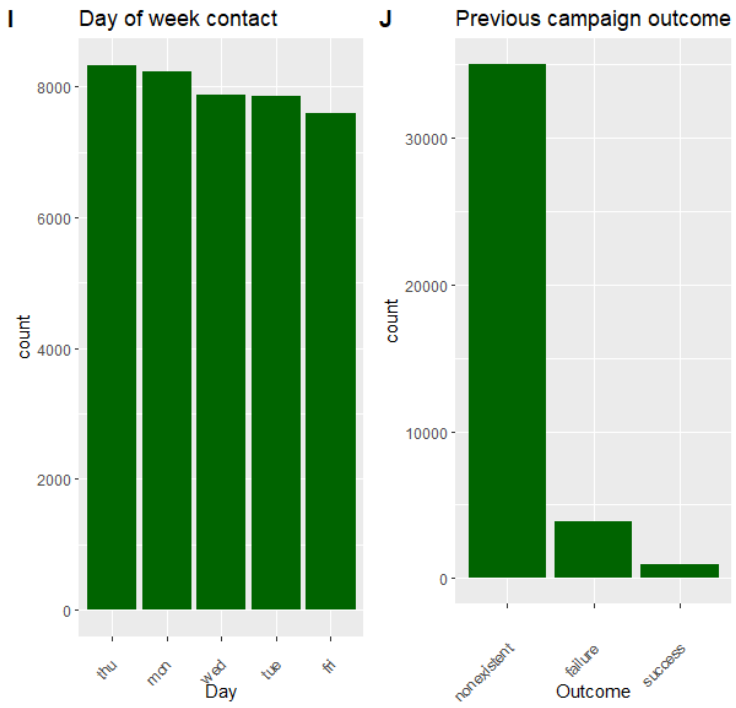

Figure 8: Distribution Barplots of variables "day_of_week" and "poutcome".

In figure 8, we can see that "day_of_week" variable's values are well represented, and we can assume that this variable's distribution approaches the Uniform Discrete Distribution. Regarding the "poutcome" variable, we see that some values of it are under-represented.

At last, in the following figure (*figure 9*) we can see the distribution bar plot of the variable "SUBSCRIBED".

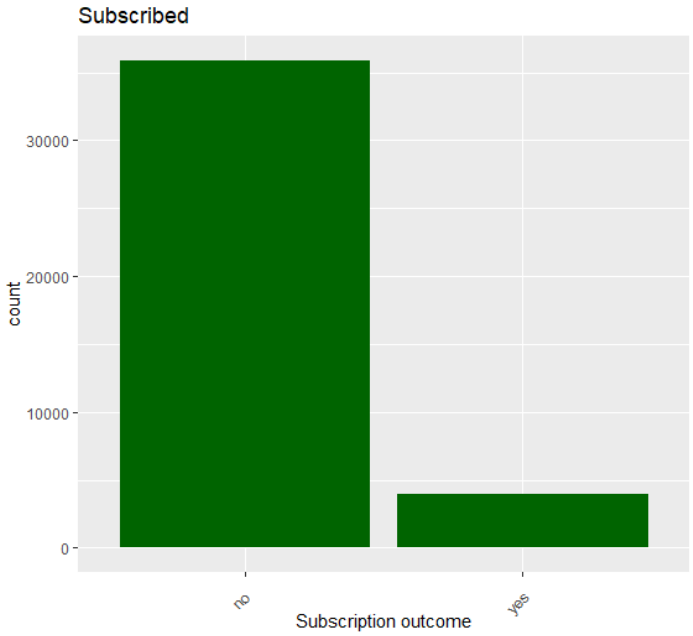

Figure 9: Distribution Bar Plot of the variable "SUBSCRIBED".

In the figure above (*figure 9*), we can clearly see that the "yes" option is quite under-represented compared to the "no" value of the variable "SUBSCRIBED".

Random Forest Classification

When using an algorithm that implements classification with a random forest, we actually construct many binary decision trees that create a "forest", take one prediction from each tree, and combine all these predictions together in order to get a final generic prediction. Each tree gets a sample with replacement from the "training" dataset, and only some of the variables are being taken into consideration (that are chosen randomly) for each tree. Since the predictions are "averaged" the random forest algorithm is safer than a single decision tree would be from possible outliers, and that is the reason I chose a random forest instead of a single decision tree (as we saw in the Exploratory Data Analysis section of the assignment, some variables have outliers).

To implement a random forest classification, we use the R function "randomForest()" from the package "randomForest". This algorithm is the algorithm of Breiman for random forests, which means that node splitting is based on the Gini Gain formula. Also, the algorithm uses the averages of 500 trees by default. These 500 trees "vote" for the prediction value of each row of the "test" and "valid" dataset, and for each row, a dataframe with 2 columns is constructed. The first column shows the percentage of "SUBSCRIBED" = "yes" votes while the second column shows the percentage of "SUBSCRIBED" = "no" votes, and the sum of each row is equal to 1 obviously. This dataframe can be seen in R with the command rf_model$votes, where "rf_model" is the name of the random forest model. When called, this command's dataframe output will show us that the voting is "clear" for some observations (for example all the trees voted for "SUBSCRIBED" = "yes") while for some other observations, votes from the 500 trees are not fully agreeing (for example 45% of the 500 trees may voted for "SUBSCRBED" = "yes" and 55% of them voted for "SUBSCRIBED" = "no"). The algorithm uses a threshold to classify the observations by using the voting results, and this threshold is by default equal to 0.5, meaning that the prediction value of an observation will be the one with the majority of votes. In the 45% and 55% example mentioned above, the algorithm will by default classify this observation as "SUBSCRIBED" = "no", since 0.55 is greater than than 0.5.

At last, we make predictions by using the R function "predict()", first with the "test" data and then with the "valid" data. Then we make a confusion matrix for each one of these 2 prediction dataframes, and finally we get the in-sample accuracy (from predictions with "test" dataset), and the out-of sample accuracy (from the predictions from "valid" dataset). The in-sample accuracy is equal to 91.98% and the out-of sample accuracy is equal to 92.28%.

K-Nearest Neighbors Classification

The basic idea when using K-nearest Neighbors for Classification is that when we want to classify a new observation, we use observations similar to it, and we use as predictors their values or function of them. In our case, we can use the mode of the k-closest observations of every observation for which we want to make a prediction. The prediction is based on *formula 1*. In this formula, *y-hat* is the prediction of the new observation *x, k* is the closest neighboring observations of the *x,* and *Nk(x)* is the set of the *k* closest neighbors of *x.*

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{Nk(x)} yi \qquad formula\ 1.$$

To train the K-Nearest Neighbor model, we will use the library “caret” of R. By default, the distances between observations are found by using Euclidean distances. Before training the model, we specify the object “trControl” from the “caret” package and it’s arguments. In “trControl”, when the argument "method" is equal to "repeatedcv", the algorithm conducts repeated K-Fold cross-validation, and when "repeats" argument is equal to 3, it means that this K-Fold Cross-Validation will occur 3 times. Next, we train the model with the “training” data and “SUBSCRIBED” variable as a response variable. In the “tuneGrid” argument, we specify different values of “k” nearest neighbors, to conduct some basic hyperparameter tuning. More specifically, the algorithm is getting executed for 10, 30, 40, 60, 80, 100, and 200 nearest neighbors. In the "preProc" argument, I Standardize the data, meaning that every variable now will have a mean value of 0 and a standard deviation of 1. Rescaling reduces the influence of extreme values on the K-Nearest Neighbor distance function. The model (which is named “knn_model” in our script) returns to us in the console that accuracy was used as a metric to evaluate each execution with a different number of k, and that the optimal number of k is k=30. This means that for every new observation we want to classify, the algorithm checks the 30 closest observations to it, and the mode of their “SUBSCRIBED” value is chosen as a prediction for the new observation.

An important limitation in this K-Nearest Neighbor is that as it is mentioned above, the distances are by default calculated with the Euclidean distance formula. In our dataset though, we have both numeric and categorical variables, so the ideal option would be to use Gower distances, but the packages we used do not support dissimilarity matrices. This means that the algorithm we chose, will correctly interpret numerical and binary variables, it will “understand” categorical variables with 3 or more different values. This will lead our algorithm to lose a lot of information when it is getting executed.

At last, we conduct predictions with both the “test” and “valid” datasets, and a confusion matrix is produced for the predictions of each dataset. The accuracy of the K-Nearest Neighbor model in the predictions of the “test” dataset is 90.90% and the accuracy in the predictions of the “valid” dataset is equal 91.12%

Naïve Bayes Classification

A naïve Bayes classification algorithm is based on the Bayes Theorem in order to classify observations. Let us suppose for simplicity reasons, that our datasets contains only the following variables: “SUBSCRIBED” (target variable), “marital”, and “education”. According to Bayes Theorem, the probability that a person will subscribe while having “marital” =1 and “education” = 3 is given by the following formula (*formula 2*):

$$formula\ 2.$$

$$P(SUBSCRIBED = Yes\ |marital = 1, education = 3)$$

$$= \frac{P(marital = 1, education = 1|SUBSCRIBED = Yes)P(SUBSCRIBED = Yes)}{P(marital = 1, education = 3)}$$

The frequencies this formula needs, can be calculated by creating 2 frequency tables. In the first one, there will be 4 rows, one for each different value of the “marital” variable, and 2 columns, one for each value of the “SUBSCRIBED” variable (yes and no in our case). The

second frequency table will be made accordingly: it will have 7 rows (each for every different value of the "education" variable, and 2 column each for every different value of the "SUBSCRIBED" variable (yes and no in our case).

The algorithm though, is called "naïve" because it uses the assumption that variables are independent even though many times they are not. After this strong assumption, the formula above (*formula 2*) changes, and for a new observation that must be classified, two formulas are now used. These 2 formulas (*formula 3* and *formula 4* can be seen below):

$$formula\ 3.$$

$$P(SUBSCRIBED = Yes\ |marital = 1, education = 3)$$
$$= \frac{P(marital = 1, |SUBSCRIBED = Yes)P(SUBSCRIBED = Yes) * P(education = 3|SUBSCRIBED = Yes)p(SUBSCRIBED = Yes)}{P(marital = 1, education = 3)}$$

$$formula\ 4.$$

$$P(SUBSCRIBED = No\ |marital = 1, education = 3)$$
$$= \frac{P(marital = 1, |SUBSCRIBED = No)P(SUBSCRIBED = No) * P(education = 3|SUBSCRIBED = No)p(SUBSCRIBED = No)}{P(marital = 1, education = 3)}$$

When for example a new observation must be classified (let's say that this observation has "marital"=1 and "education"=3), then the 2 formulas above will be used, in order to find the possibility that a person with such characteristics subscribed, and the possibility that a person with these characteristics did not subscribe, all by using past data ("training" dataset in our case).

The problem with *formula 3* and *formula 4* is that if any term (for example P(marital=1|SUBSCRIBED=Yes) is equal to zero (because there were no people with "marital"=1 who subscribed) then the division will result to zero. Since it can be natural that a certain group of people with a specific attribute may have never subscribed for various reasons, we will use "Laplace Smoothing" when training the Naïve Bayes Classification model, meaning that we add a small number, usually "1" so the division will never result to zero.

To train the naïve bayes model ("nb_model") we will use the "naivebayes()" function from the package "e1071". The target variable will be the "SUBSCRIBED" variable obviously, the data used in the training will be the "training" dataset, and the argument "laplace=1" will be added. When plotting the "nb_model", R will return density plots for each explanatory variable like the following one that can be seen in *figure 10*. In this figure, we can see the density plot of "marital" against each possible outcome of the "SUBSCRIBED" variable, and the density plot of the "education" variable for every possible outcome of the "SUBSCRIBED" variable. Density plots for the rest of the independent variables will not be displayed here for space saving reasons. As it can be clearly seen in *figure 10* the density plots of every possible outcome of the "SUBSCRIBED" variable, overlap with each other at both plots. This could mean that variables "education" and "marital" may not be independent after all.

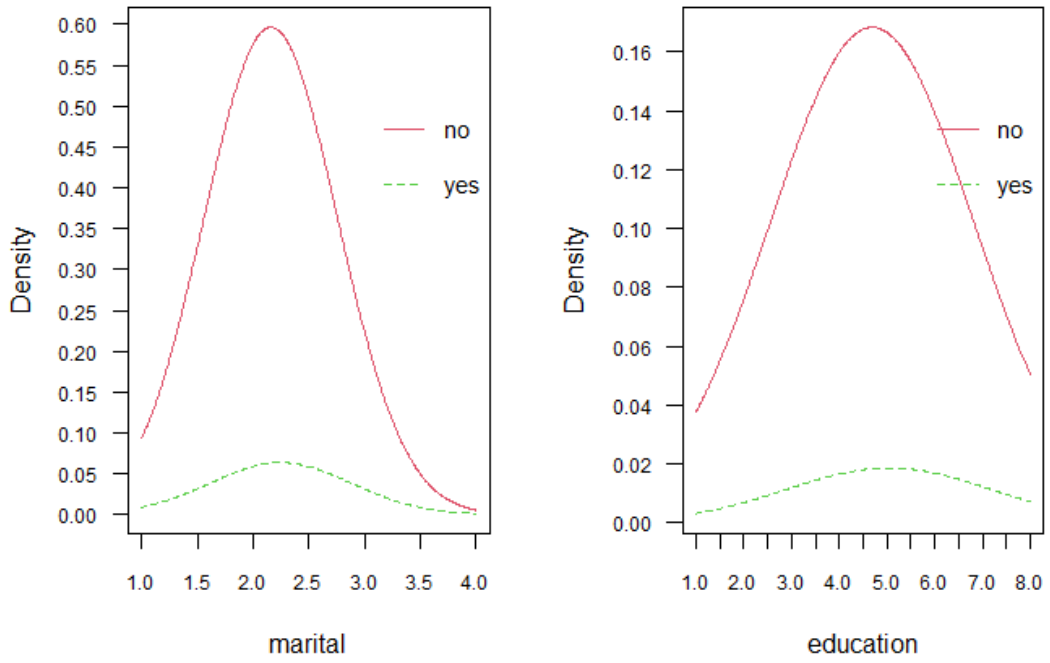

Figure 10: density plot of "marital" variable against each possible outcome of the "SUBSCRIBED" variable, and density plot of the "education" variable for every possible outcome of the "SUBSCRIBED" variable.

At last, we predict with the "predict()" function and use at first the "test" data and then the "valid" data. Then we make a confusion matrix for each set of predictions, and get the in-sample accuracy of 81.84% and the out-of sample accuracy of 82.39%. The "nb_model" preforms worse than pure luck, because as the "a priori" probabilities are 89.96% for "SUBSCRIBED"=no and 10.03% for "SUBSCRIBED"=yes, then if we would just predict that nobody will subscribe, we would probably achieve an accuracy of around 89%. This means that "nb_model" has a lift score less than 1, and should not be used for predictions.

Support Vector Machines Classification

When using a support vector machines algorithm for classification, we construct a hyperplane on the space of predictor variables. The aim of the hyperplane is to separate the observations in two groups (depending on the target variable's value), and the best separation (out of all the possible separations that can be achieved) is the one that maximizes the geometric margin from the support vectors (meaning the observations that are close to the area where the hyperplane is and are the only observations that will be taken into consideration), and minimizes the empirical classification error.

When predicting with a support vector machines model, we assign the new observation on one, from two sides in total, of the hyperplane. If, for example the side in which a new observation is assigned, is the side where the vast majority of observations have their individual "SUBSCRIBED" value equal to "yes", then the algorithm will classify the new observation's "SUBSCRIBED" value as "yes" too.

To train such a model, I used the function “svm()” from the package “e1071”, having “SUBSCRIBED” variable as a target variable, “training” dataset as the data used in the training of the model, and “type” argument equal to “C-classification” to let the algorithm know that I want to classify and not use regression. I also choose the “kernel” argument to be equal to “radial”, after having tried manually the options “linear”, “polynomial”, and “sigmoid” (“radial” type would always yield the highest accuracy in both “test” and “valid” dataset predictions. At last, the data is scaled. It must be noted that when Classifying observations with a Support Vector Machines algorithm, we avoid overfitting and the model is not getting influenced by outliers, since only the support vectors participate in the formation of the hyperplane. This is quite positive in our case, since as it can be seen in the Explanatory Data Analysis section of the assignment, some variables have some outliers.

To get the predictions, we use the “predict()” function. Specifically, we make predictions with both the “test” data and the “valid” data, and after doing so we get the confusion matrix of each set of predictions. At last, we get the in-sample accuracy which is equal to 91.19% and then the out-of-sample accuracy of 91.88%.

Selecting the final model

To select the final model for classification of the “SUBSCRIBED” variable, we will use the accuracy metric, which indicates what percentage of predictions were true, out of all the predictions we did. More specifically, we will take into consideration the in-sample accuracy, since the out-of sample accuracy is calculated by using a quite small (in terms of observations number) data set. The model with the highest in-sample accuracy is the Random Forest (“rf_model”), having an accuracy of 91.98% (in-sample accuracy).

Clustering with Partitioning Around Medoids

An attempt to cluster the customers will be made by using the Partitioning Around Medoids algorithm. This algorithm (PAM) is related to the K-Means algorithm and is a method to partition data points into “k” clusters/groups. The main difference between PAM and K-Means clustering is that PAM is less sensitive to noise and outliers since the centers of the clusters are observations, and not the means of the cluster which can be severely affected by outliers. For this reason PAM will be chosen for our dataset clustering instead of K-Means clustering, because as we already saw in the Exploratory Data Analysis section of the assignment, some of our variables have outliers. PAM algorithm requires the user to specify the exact number of clusters and to do so, I will try to cluster the data with some different numbers of clusters each time (2, 3, 4, 5) and I will choose the value that has the lowest average silhouette width.

Before clustering, it must be noted that not all the variables of the data will be used. The dataset “data_clust” will be used for clustering. In this dataset, the variables will be the following: “age”, “job”, “marital”, “education”, “default”, “housing”, “loan”, “campaign”, “pdays”, “previous”, and “poutcome”. Since the RAM memory of the computer used in clustering can not handle the whole dataset, we will use only a part of it, and infer the results of this part’s clustering to the whole dataset. Only the first 15.000 rows of “data_clust” will be used, and they will be stored in the dataset “data_sample”.

As we can see by using the “str()” function, the variables of “data_sample” are either of “Factor” or “numeric” data type. This means that we can not use the Euclidean distance for clustering, and instead we will make a dissimilarity matrix named “dist” with the “daisy()” function from package “cluster”. An argument of the “daisy()” function is the “metric” which

is equal to “gower” and another one will be the “stand” which is going to be equal to TRUE since we want to standardize the data.

Next, as we mentioned above, we will try to cluster the dataset “data_sample” for some values of “k” (2, 3, 4, 5). When doing so, we can plot them, and the result can be seen in *figure 11*.

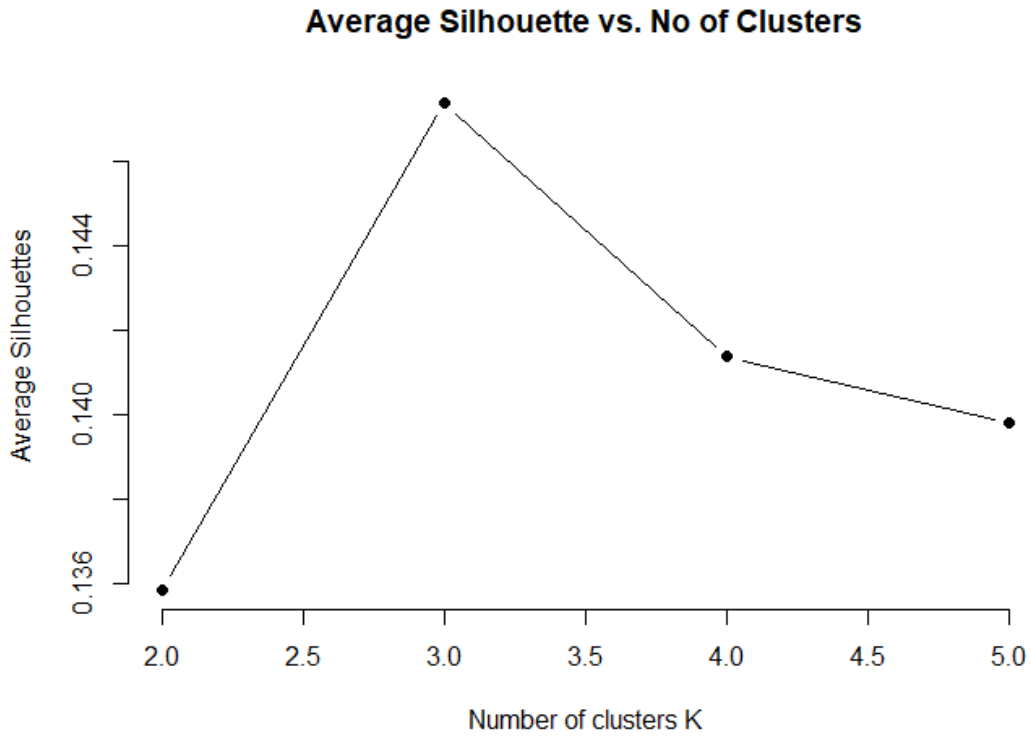

Figure 11: Average Silhouette width of clusters, for 2, 3, 4, and 5 clusters. “K” = 3 seems to be the ideal number of clusters since for 3 clusters, we get the highest average silhouette width.

As it can be seen in *figure 11* the optimal number of clusters is 3, since with this number of clusters we get the highest average silhouette width.

We will now implement the final clustering by using PAM, with 3 clusters. The model is named “pam_final” and we must add the argument “diss”=TRUE in it, so as to let R know that we have a dissimilarity matrix. The three clusters produced have an average silhouette width of 0.1473, and their sizes are 5923, 4152, and 4925. We can also plot the silhouette width of the clusters. In *figure 12* we can see every cluster’s silhouette values.

As we can see in *figure 12* the first cluster consists of 5923 observations and has an average silhouette width of 0.14, which also happens to be the lowest out of all clusters. The second cluster consists of 4152 observations, making it the smallest cluster between them all, and has an average silhouette width of 0.16, making it the cluster with the highest average silhouette width. The third cluster consists of 4925 observations and has an average silhouette width of 0.15. It must be noted that the first cluster has the most observations with negative silhouette values and is the cluster with the lowest average silhouette width. This could mean that this cluster is not so “clear” and many observations belonging to it should not be there. On the other side, the second cluster is the cluster with the least observations having negative

silhouette values, and happens to be the one with the highest average silhouette width. We can conclude that the second cluster is the "clearest" cluster among the 3 clusters.

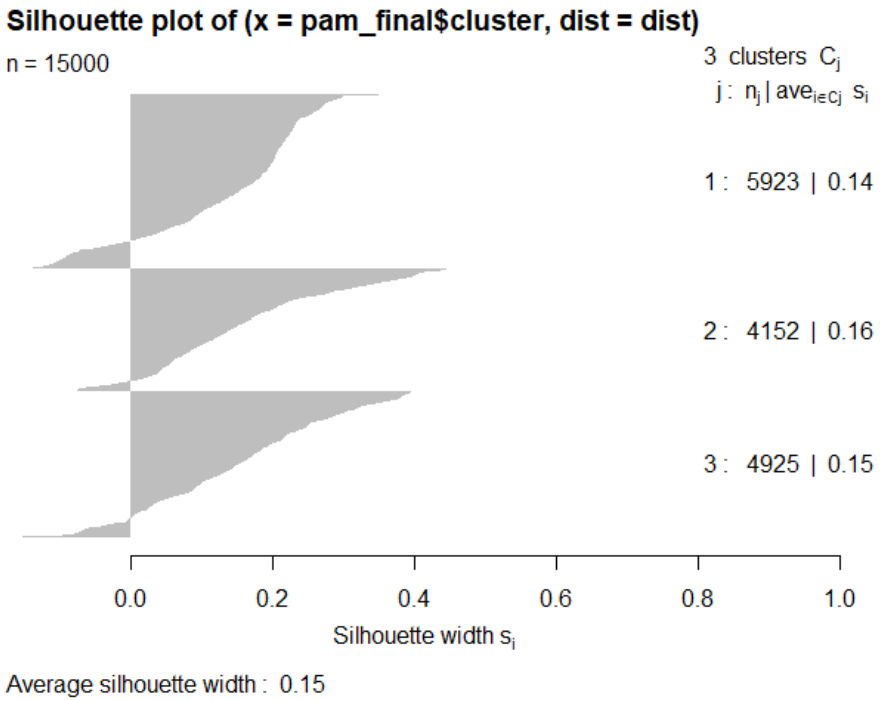

Figure 12: Silhouette widths of the 3 clusters' observations

To interpret the clusters, we must check the medoids. As we mentioned earlier, the medoids are the most "centralized" observations of each cluster, thus we can conclude that they summarize the cluster decently. Since we have 3 clusters, we can see that there 3 different customer profiles.

The first customer profile is a 38 year old (on average) married person with a blue-collar job whose highest education is high school, has no default on his/her name, and does not own the house he/she lives in.

The second customer profile is a 45 year old (on average) married person with a blue-collar job whose highest education is the basic 9 year old education, has unknown default status, and own the house he/she lives in.

The third customer profile is a 35 year old (on average) married person with a job in the administration sector whose highest level of education was the university degree, has no default on his/her name, and own the house he/she lives in.

An effective way to use these 3 clusters/profiles would be to use different pricing policy for each one of them, in order to attempt a more personalized approaching to each group of customers. This could increase the probability that some marginal customers of each group will convert their "SUBSCRIBED" status to "yes" from "no", thus leading to higher earnings.