Statistical and Machine Learning Models

Individual Project: Machine Learning Models Benchmarking

Mario Serrano

1. OBJECTIVE

In this report, I am going to explain and evaluate five different machine learning models. For accomplishing this, I first explain each one of the selected models and the apply a benchmark experiment using a banks marketing data set. The goal of the modeling experiment is to predict whether a client is going to subscribe or not, hence, a classification prediction.

2. MACHINE LEARNING MODELS

As mentioned before, the goal for each one of the models is to predict whether a client is going to subscribe or not using all the available information from past campaigns, demographics, etc., which corresponds to a classification problem. Since we are trying to predict a specific outcome (1 or 0), we need to apply supervised learning methods. For this type there is a wide set of models to be selected, in this particular case, the selected models for this experiment are the following:

- Logistic Regression
- Decision Trees
- Random Forest
- Linear Discriminant Analysis (LDA)
- K-Nearest Neighbors (KNN)

As a main basis for all the mentioned models, it is required to have a set of one or more training observations in order to predict the main outcome. It is important that the prediction also performs in the best way possible in the test set to then proceed with a final prediction.

2.1. LOGISTIC REGRESSION

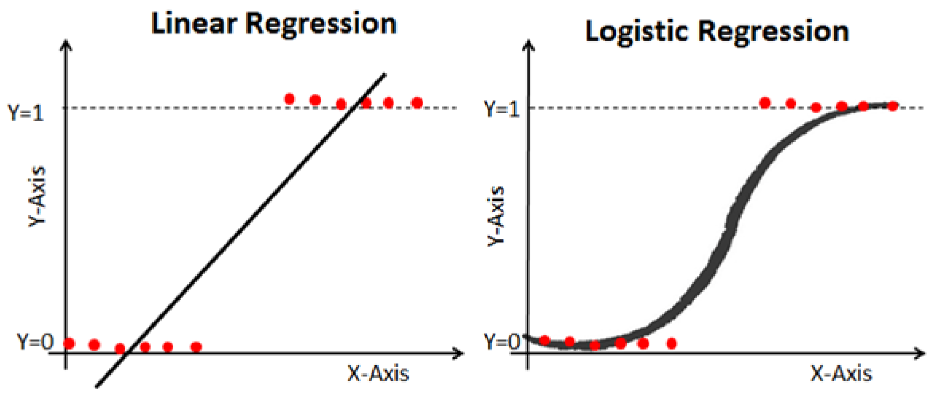
As a fast and easy interpretation for this classifying model, we can say that it uses a binomial distribution to predict the probability of the client subscribing or not (binary outcome).

Now, getting into more specific details of the model, what we get from this model is the following:

$$\Pr(Subscribe = Yes|\, X_i)$$

Which translates to the probability of a customer getting a subscription given an independent variable X_i.

[1]



As an initial comparison, in the previous plot we can see a regression made with a simple linear model and a second one made with a logistic regression. Clearly, there is something wrong with the linear regression in this case, since it is predicting negative or greater than 1 probabilities, whereas the logistic regression prediction always lies within the range between 0 and 1.

In order to obtain this "S" shaped predicted probabilities, the model uses a logistic function (or sigmoid function) from the exponential family and fits each predicted value by maximizing the likelihood. This function is represented as:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

Since, this is a parametric model, the betas used in this function correspond to the estimated coefficients for each one of the independent variables used to predict the probability of the target variable. The process of obtaining the best coefficients follow a maximum likelihood method which uses the likelihood function:

$$\ell(\beta_0,\beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1-p(x_{i'})).$$

Once we have each of the optimum parameters we can proceed with the last step, which is making our prediction. For this, we use the logistic function and replace with the estimated parameters in the way:

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1{,}000}}{1+e^{-10.6513+0.0055\times 1{,}000}} = 0.00576,$$

In this example, the estimated parameters lead to a probability of 0.00576 to correspond to the 1 category. Since we are dealing with a classification problem, we need to use the estimated probability to classify whether the predicted outcome would be 1 or 0. To get a precise estimation of the classification we can define a threshold for the probabilities. As a rule of thumb, the threshold is defined to 0.5 and it would work as follows:

$$\hat{X} = \begin{cases} \hat{p}(X) \geq 0.5 & 1 \\ \hat{p}(X) < 0.5 & 0 \end{cases}$$

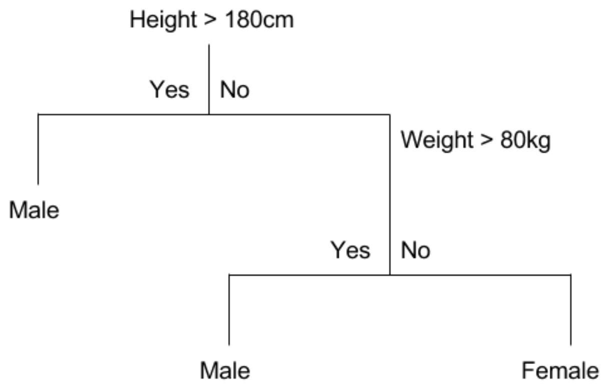
Using our defined threshold, we can now determine where our outcome is 1 or 0, get our final classification and test it.

[1] Logistic Regression in Machine Learning | by Krantiwadmare | Analytics Vidhya | Medium

2.2. DECISION TREES (CLASSIFICATION)

Before talking about how trees work, lets state some of the main advantages and disadvantages of these type of models. First, they are really easy to explain since they have a simple graphical representation. As they have a shape of a tree (branches, leaves, etc.) it is easy to interpret by a non-expert. Something that is useful is that they can handle categorical variables without the need of creating dummy variables.

By the other side, most of other regression and classification methods usually have a higher accuracy than decision trees. They are also sensible to changes in the data, meaning that a small change can drastically result in a different output.



[2]

Decision trees, in general, create a stratification of variables using a series of splitting rules for regression or classification and create predictions. As an example, the tree shown above, it is classifying whether if gender is female or male based on height and weight variables. In this example, if height is greater than 180 cm then gender is male, if height is less than 180cm and weight is greater than 80kg then gender is male and female otherwise.

Classification trees predict each observation class based on the most commonly occurring class within the region, meaning that it groups observations with similar behavior to create a classification. To grow a decision tree, binary splits are made which are decided by a classification error rate. The error rate explains the fraction of observations that do not belong the most common class.

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Besides the error rate, another important measure can be used, the Gini Index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

This basically shows the total variance across the classes, where if it has a small value, it means that the classes are closer together.

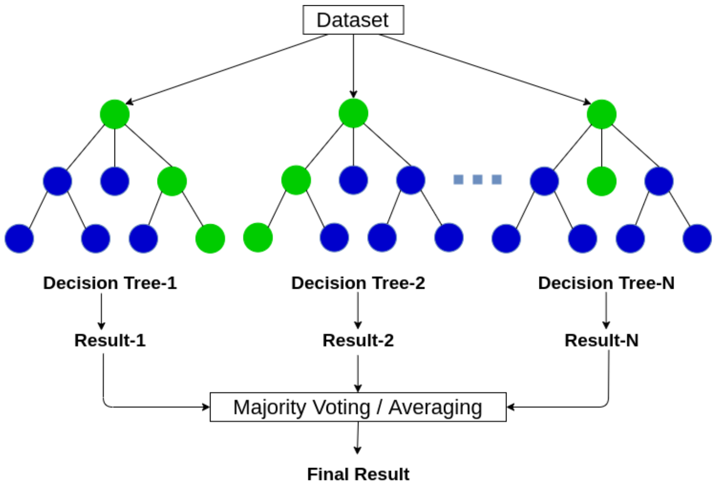
An important aspect from decision trees is that it starts to create partitions from the most important variable to the least important. In this case, a higher hierarchy level (first nodes) would be the most relevant for the classification. In this sense, it is relevant not to select too many levels of partition as it will overfit the classification (it becomes too specific). That is when pruning comes into effect, where the process helps to reduce the lower part of the tree so we can keep only the most relevant partitions. The level of prune can

[2] Classification And Regression Trees for Machine Learning (machinelearningmastery.com)

be determined using the Gini Index or the classification error rate. This way we can ensure the best outcome possible from the tree and proceed with testing.

2.3. RANDOM FOREST

As the name suggests, this method is based on decision trees, where it builds decision trees over bootstrapped training samples. As stated before, since decision trees are sensible to data, they have a high variance. By taking bootstrapped samples from our data and creating many trees, we can reduce this high variance. In this sense, the random forest model uses the built trees and estimates the average of the results to get a more robust classification, normally with a higher accuracy and more importantly with low variance:



$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

Th formula above represents the average taken from the results of each tree for B training sets, which is the basis of the bagging trees methodology. This methodology is also applied with random forests, excepting that in the latter case it eliminates the correlation between the decision trees that were created. This is done in the way that for every split of the tree, a random sample of predictors is chosen from the total number of predictors.

Taking this to an example, let m be the number of random selected predictors for the split. If we define a $m = 5$ from a full set (p) of 20 predictors, then for every tree split, 5 independent variables will be chosen to create the classification. Then for the next partition, another random set of 5 predictors will be selected and so on. As a common practice, the number of selected predictors is defined as $m \approx \sqrt{p}$, but it can also be selected by cross validation methods or out of bag (OOB) errors, where we would select m by minimizing the error.

This process is the main characteristic from the random forest model since it allows different variables to be used as classification splits and not only variables with the higher importance, thus decorrelating the trees built.

2.4. LINEAR DISCRIMINANT ANALYSIS

In contrast to logistic regression, which works only for a binary classification, a discriminant analysis can be useful for a multiple class classification problem. Another great advantage is that is usually more stable than the logistic regression.

One of the quick ways of defining the process of this model is that it takes each of the predictor variables, models the distribution for each of the classification classes and then it applies Bayes to estimate the probability.

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Applying the Bayes theorem, the model gets the posterior probability that each one of the observations belongs to a class. This means, the probability of an observation belonging to a class given the independent variable value. With the estimated probabilities, the model classifies each of the observation the class where its probability is the greatest.

Another important factor for this model, is that is takes the distribution of each one of the independent variables to estimate the probability. In LDA, the assumption is that the predictors follow a normal (or Gaussian) distribution. Continuing with the use of a normal distribution, we take the probability density function and plug it into the Bayes formula (presented above):

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

So, to get the prior probability of an observation belonging to an x class:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_l)^2\right)}.$$

With this defined, the model estimates the given probabilities and assigns each to a class according to the greatest probability. Then, we can continue testing our predictions to determine the accuracy or AUC for the obtained prediction.

2.5. K-NEAREST NEIGHBORS

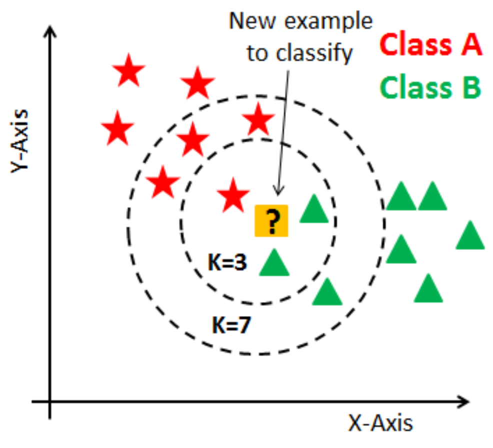
Is a non-parametric method, which means that it does not estimate any parameters in order to create a classification. Another characteristic of these methods is that they do not make hard assumptions about the form of our target, as for example a linear regression assumes a linear relation. Basically, non-parametric methods try to get the closest data points as possible to a specific observation to make the classification using averages, this is the case of KNN.

One main advantage about this model against other parametric models is that it may be able to fit better to different shaped dependent variable as it does not assume any form. On the other hand, since non-parametric models are based on averages, they need a large number of observations to obtain more accurate estimate.

Now, specifically for KNN, the model starts with a defined K value and identifies the K nearest observations closest to a prediction point. Then it sums the number of data points that belongs to a class i within the nearest selected observations and divides it by the defined K, in the way:

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i.$$

Once it has this average for the prediction point it uses the Bayes rule to classify the point to a certain class by taking into account the greatest probability.



[3]

Using the previous plot, we can explain it in a graphical way. For example, if we define a K=3 (smaller dotted circle), we have 2 green triangles and 1 red star, which will lead to a 2/3 probability for triangles and a 1/3 probability for stars. In this case, the new point to classify will be assigned to the green triangles since they have a greater probability within the K=3 defined region. On the other hand, if we define a K=7, we have a wider region and now we have a 3/7 probability for triangles and a 4/7 probability for stars, hence the new data point will be classified as a red star.

As we can see, the K is of high importance to determine the classification of a new data point. In general, defining a higher K, results in a smoother fit, but it will be less flexible than a low value for K. Taking into consideration the flexibility of the model, a higher K will result in less variance but a higher bias, whereas a smaller K has less bias but a higher variance.

3. BENCHMARK EXPERIMENT

The goal in this section, is to present a full classification problem, ranging from data processing to model results. The same models that were explained in the previous section were used for this new section, where we are going to compare each one of the obtained results by different measures.

3.1. PIPELINE

In order to have a structured working process, then setting up a pipeline is highly recommended to get the best results out of our data problem. By keeping this process clean, we can ensure easy interpretability, understanding and replication. The pipeline shows the course of action starting from data processing, machine learning and evaluation. The next sections, present in detail the followed procedure to build the benchmark experiment.

3.2. DATA PROCESSING

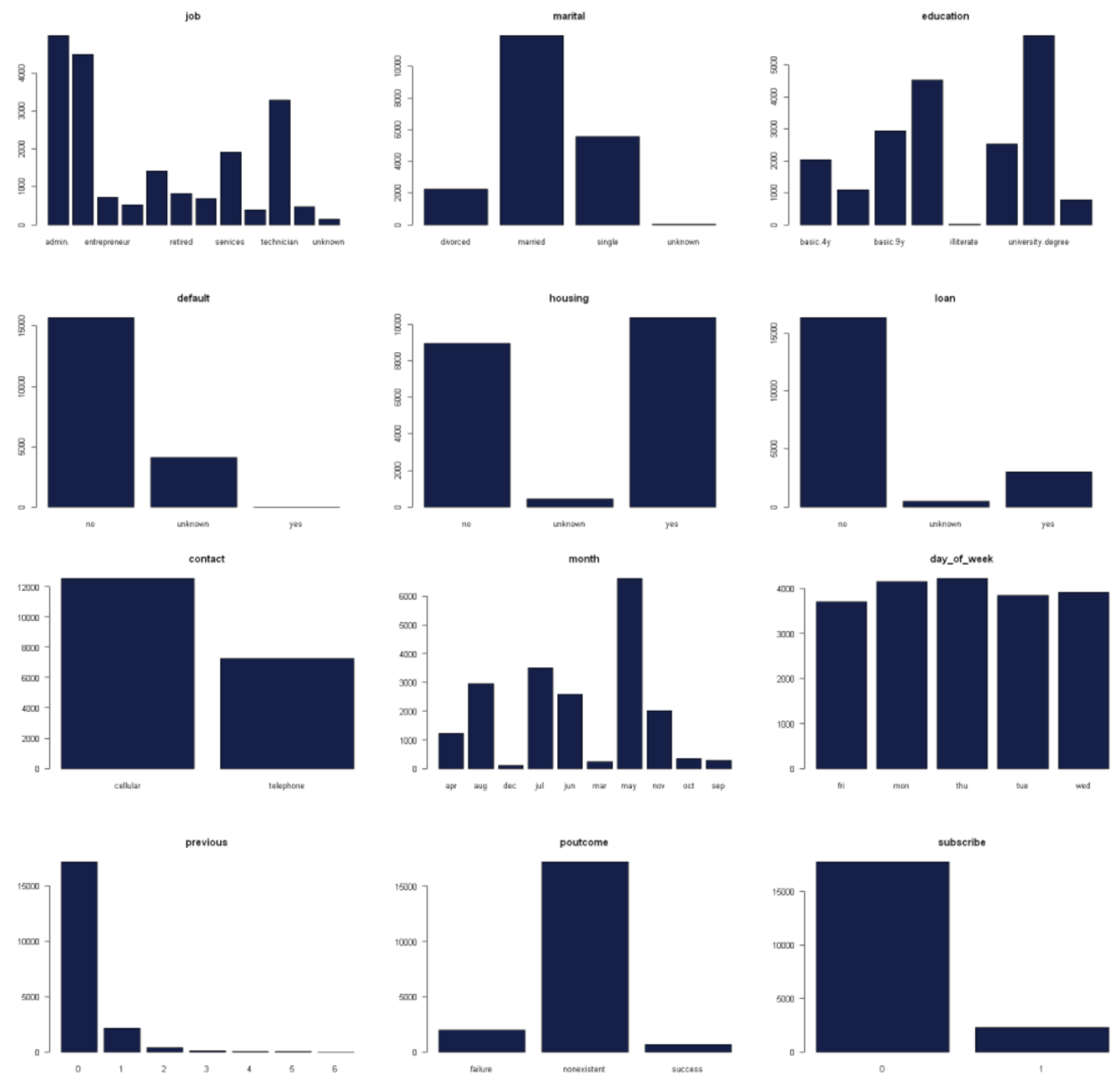
The data used for this experiment, corresponds to marketing information from a bank, where we can find the details on clients (demographics and financial situation), current campaigns and different attributes, social and economic context values, and the target variable. In this case, the target variable shows whether a client subscribed or not.

3.2.1. EXPLORATION

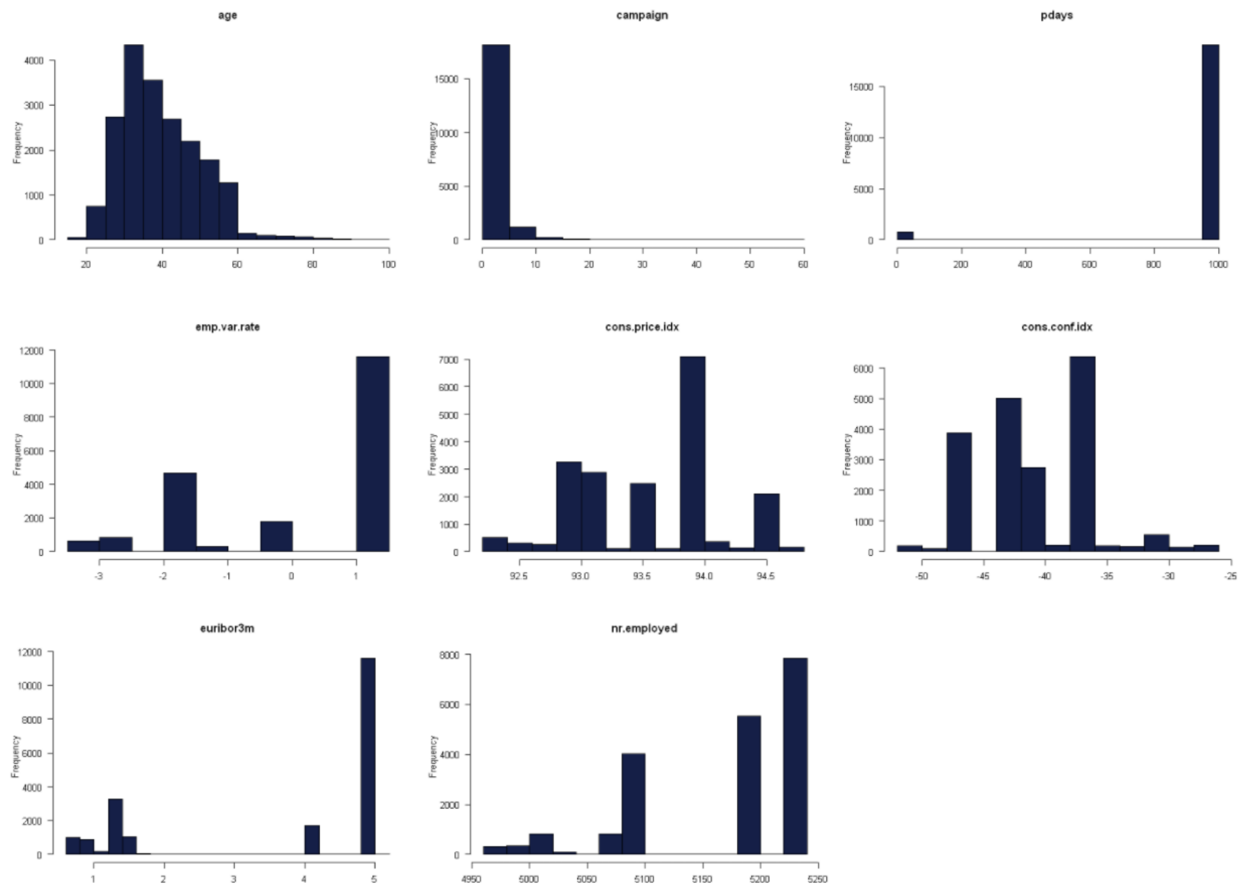
The mentioned information has a total of 21 variables including the target variable and 20,000 observations. There is a total of 10 numeric variables, and 11 categorical variables, that are going to be treated independently checking for missing values, outliers, grouping, etc.

[3] Most Popular Distance Metrics Used in KNN and When to Use Them - KDnuggets

The following bar plots show the categorical variables:



We can see that there are several variables with small groups that do not represent more than 5% of the observations. The last plot (subscribe) shows the distribution of the target variable, where there is a much greater number of non-subscribed clients than subscribed clients.



The previous histograms show the value distribution for each of the numeric variables in the data, which are going to be analyzed later.

3.2.2. MISSING VALUES

Continuing with the data processing, I check on missing values and decide on what treatment to give them. In this case, there is a total of 3,664 observations with missing values.

Column	Missing Values
client_id	0
age	202
job	161
marital	199
education	170
default	214
housing	195
loan	219
contact	217
month	199
day_of_week	185

Column	Missing Values
campaign	203
pdays	185
previous	209
poutcome	175
emp.var.rate	165
cons.price.idx	181
cons.conf.idx	197
euribor3m	204
nr.employed	184
subscribe	0

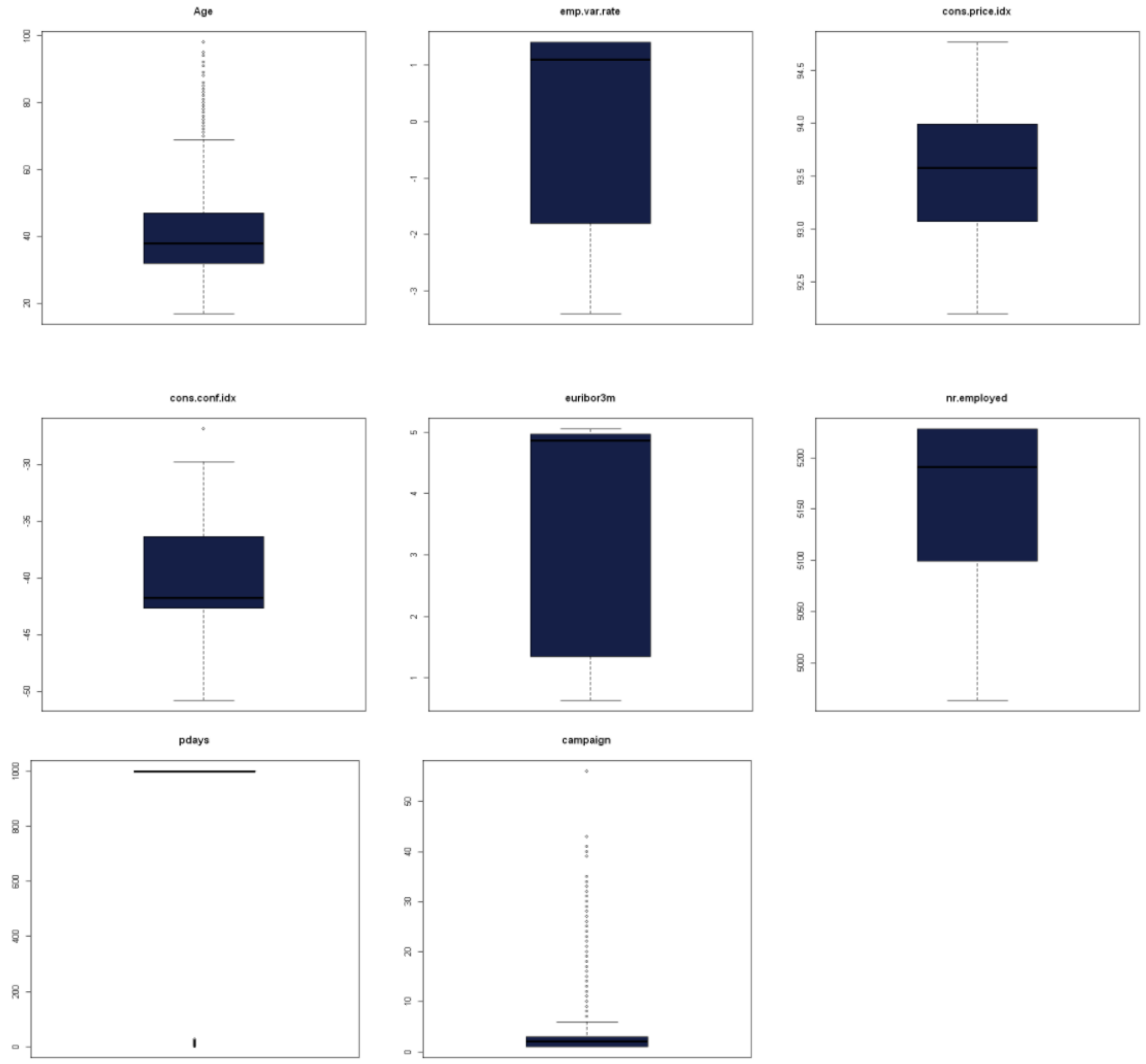
On average, 1% of the observations for each column has missing values. A simple approach was taken to fill up these values depending on if they correspond to categorical variables or numeric ones. In the case of character class columns, a random assignation was made using as probabilities the percentage distribution of the categories. For example, if a variable has two classes (yes, no) with a 70/30 distribution, then the missing values will be filled up using a 70% probability for “yes” and 30% for “no”.

By other hand, for numeric variables a simpler approach was taken and filled up the missing with the average values for each column. An additional column was created for each variable to identify the missing values. In total, 7 new variables were added for this identification.

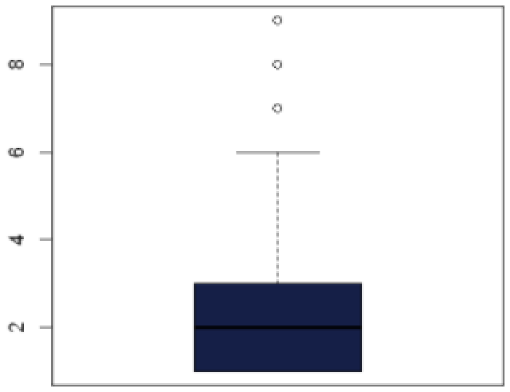
In the case of the “pdays” variable, the missing values were imputed with 999, since it shows that the clients where not contacted.

3.2.3. OUTLIERS

Now, to check for outliers within the numeric variables we first observed the distribution on boxplots.



We can see that there might be some outliers for age, pdays, and campaign. In the case of age, no outliers were taken out or windsorized since there are not really unprobeable ages. For campaign, since it tells us the number of contacts made, and there a few what have more than 11 contacts a windosization was used. To determine 11 as the max number of contacts, I used the mean plus 3 times the standard deviation:



At last, for pdays, a new dummy variable was created, to identify if the clients were contacted or not. If pdays=999, then 1, else 0.

3.2.4. GROUPING AND DUMMIES

For categorical variables, there are some categories that have less than 5% of the observations. Within each column, the classes with less than 5% were grouped into "others". This is done for the job and education variables.

education

illiterate	unknown	basic.6y	basic.4y
0.00045	0.0389	0.055	0.10195
professional.course	basic.9y	high.school	university.degree
0.126	0.1471	0.2263	0.3043

job

unknown	student	unemployed	housemaid	self-employed
0.0072	0.0196	0.02335	0.0262	0.0347
entrepreneur	retired	management	services	technician
0.0358	0.04145	0.071	0.09565	0.1639
blue-collar	admin.			
0.2243	0.25685			

After grouping these new categories for these variables, we took all the categorical variables and created dummy variables for each one of the groups, having now a base table with 53 columns.

3.2.5. HOLDOUT

After the processes mentioned above, we proceed to generate our train and test tables using a random selection method and 80% for the train set and 20% for the test. This means, 16,000 observations for train and 4,000 for test.

Since the selected models have different qualities and process different types of data, I created 3 sets of train and test tables. In this matter, a first set of train and test was created using the base table with all the dummy variables created and with no categorical variables. This tables have 53 columns and are used for the Logistic Regression and Linear Discriminant Analysis.

A second set of train and test was made with the base table without dummy variables. This means, that all the categorical variables are left as factors for the model application. These tables have a total of 27 variables and are used for Decision Tree and Random Forest.

At last, a set of train and test was created from the base table with dummy variables. The only difference from this set and the first one is that it contains scaled and centered numeric variables. Hence, this tables have 53 columns and were only used for the K-Nearest Neighbors model.

3.3. MODEL APPLICATION

Once the base table, and our train and test tables were defined, we can proceed with the model application.

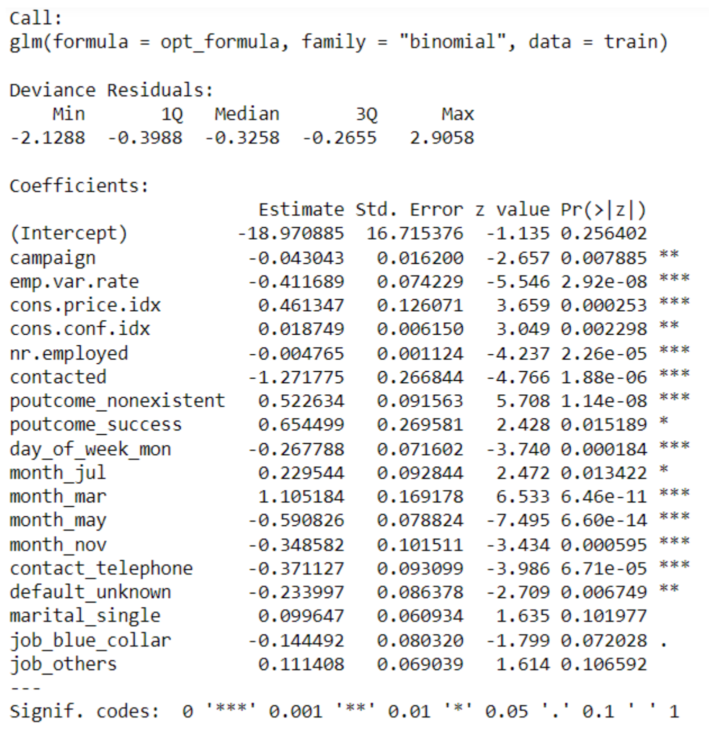
3.3.1. LOGISTIC REGRESSION

Initially, a logistic regression was applied using the complete train table in order to fit a stepwise variable selection. I used a “stepAIC” function in R software with a backward process to select the variables that will result in the best AIC outcome.

Logistic regression

Variables	AIC
52	9052.0
18	8998.7

The best model is created with18 variables and has the following coefficients:

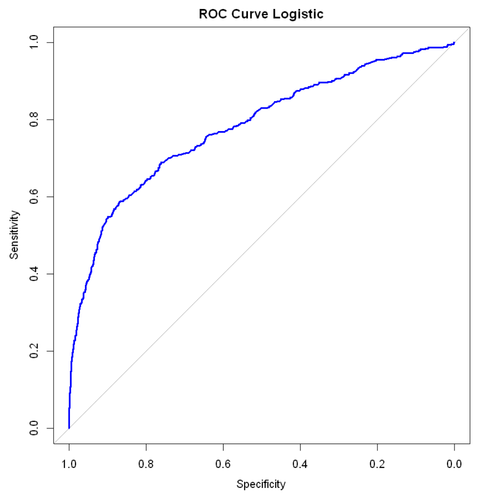


```
Call:
glm(formula = opt_formula, family = "binomial", data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.1288  -0.3988  -0.3258  -0.2655   2.9058  

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)    
(Intercept)         -18.970885  16.715376  -1.135 0.256402    
campaign             -0.043043   0.016200  -2.657 0.007885 ** 
emp.var.rate         -0.411689   0.074229  -5.546 2.92e-08 ***
cons.price.idx        0.461347   0.126071   3.659 0.000253 ***
cons.conf.idx         0.018749   0.006150   3.049 0.002298 ** 
nr.employed          -0.004765   0.001124  -4.237 2.26e-05 ***
contacted            -1.271775   0.266844  -4.766 1.88e-06 ***
poutcome_nonexistent  0.522634   0.091563   5.708 1.14e-08 ***
poutcome_success      0.654499   0.269581   2.428 0.015189 *  
day_of_week_mon      -0.267788   0.071602  -3.740 0.000184 ***
month_jul             0.229544   0.092844   2.472 0.013422 *  
month_mar             1.105184   0.169178   6.533 6.46e-11 ***
month_may            -0.590826   0.078824  -7.495 6.60e-14 ***
month_nov            -0.348582   0.101511  -3.434 0.000595 ***
contact_telephone    -0.371127   0.093099  -3.986 6.71e-05 ***
default_unknown      -0.233997   0.086378  -2.709 0.006749 ** 
marital_single        0.099647   0.060934   1.635 0.101977    
job_blue_collar      -0.144492   0.080320  -1.799 0.072028 .  
job_others            0.111408   0.069039   1.614 0.106592    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This model is used to predict the class, whether a client is going to subscribe or not on the test set. Since from a logistic regression we get the predicted probabilities, a threshold of 0.5 was defined to classify to 1 or 0. From the test set we get the following measures:

log_pred	**0**	**1**
0	3,461	303
1	99	137

Accuracy	0.8995
Error	0.1005
AUC	0.7811



The confusion matrix is saying that 137 (1s) were predicted correctly. Taking into account the zeros predicted and since they are the most, the accuracy of the model is 0.9. By another side, using a measure that captures only the predicted outcome for 1, we get an AUC of 0.78.

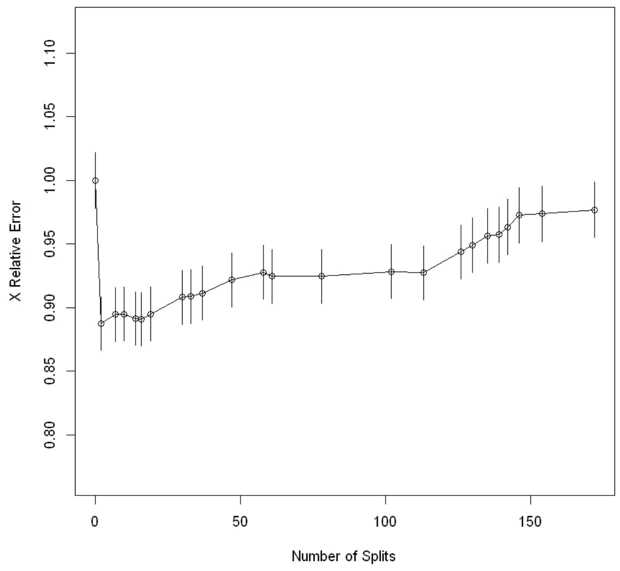
At last, as an additional validation method, we used a 10 fold cross-validation using only our selected features and the complete base table. From this process we get the following results:

auc.test.mean	0.7852
mmce.test.mean	0.1004
acc.test.mean	0.8996

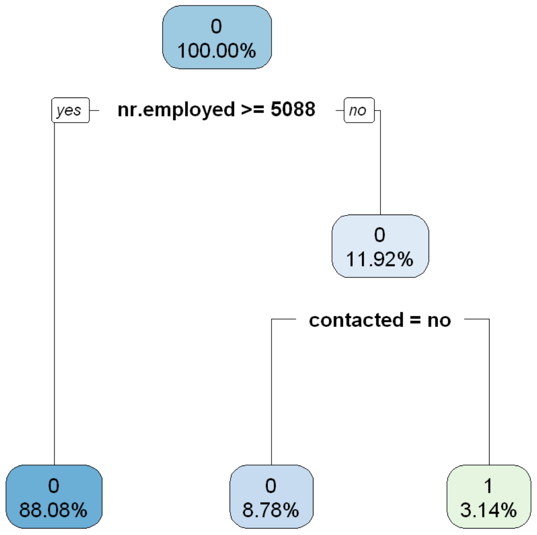
Results are similar to the obtained with the train and test method.

3.3.2. DECISION TREE

With this model, we used the train and test sets without dummies. For this, the rpart algorithm for decision trees was used in R software and it was applied for all the variables from the train set. The first classification tree shows up to 172 splits, so it is necessary to go through a pruning process for the tree. To find the optimal number of splits to be selected we check the relative error for each one of the splits:



According to the relative error, the optimal number of splits is 2, since from that point forward the error just keeps incrementing. Pruning the tree, we obtain the only two variables were selected to create the classification, which in this case resulted in contacted and nr.employed.

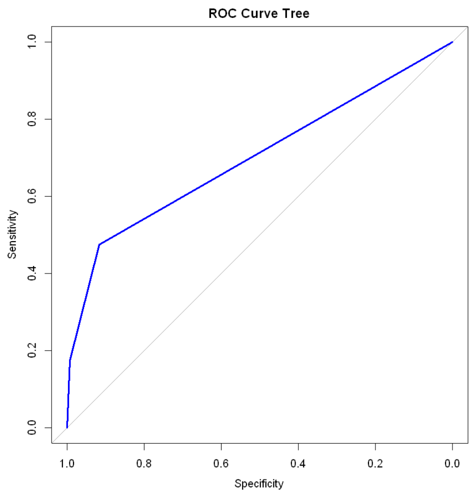


From the obtained tree, we can say that if the number of employees is less than 5,088 and they have been contacted then the classification is 1 (client will subscribe), any other combination will be classified as a non-subscription.

Using this fitted model, we predict over the test set and obtain the following metrics:

tree_pred	0	1
0	3,532	363
1	28	77

Accuracy	0.9022
Error	0.0978
AUC	0.7013



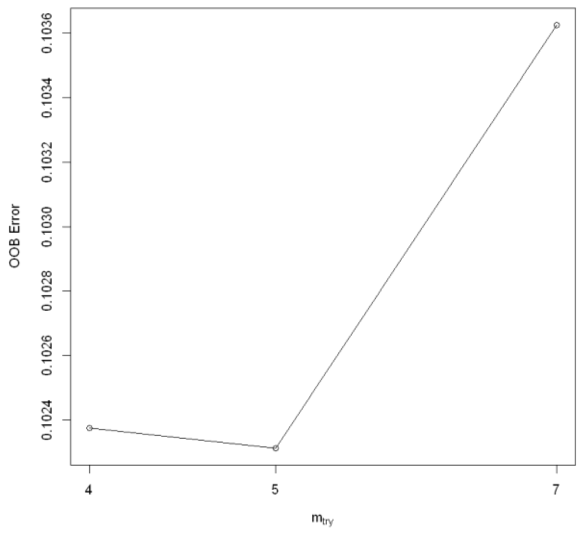
As stated before, the accuracy can be misleading in these cases since most of the predictions are zeros. Using the AUC, we get 0.7 for this model. Applying a 10-fold cross-validation over the complete base table, we get similar results:

auc.test.mean	0.7033
mmce.test.mean	0.1007
acc.test.mean	0.8992

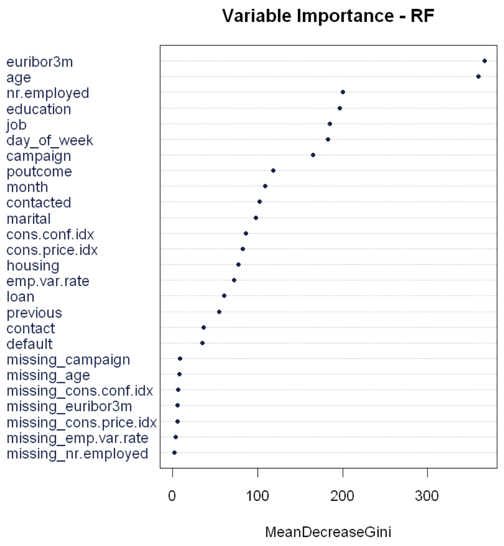
3.3.3. RANDOM FOREST

After using a simple decision tree, now we use a random forest, which is basically a combination of many decision trees.

For this model, we need to define the m number of predictors to be randomly selected. We use a tune function for random forest models, and obtain the optimal number of predictors by minimizing the OOB Error:



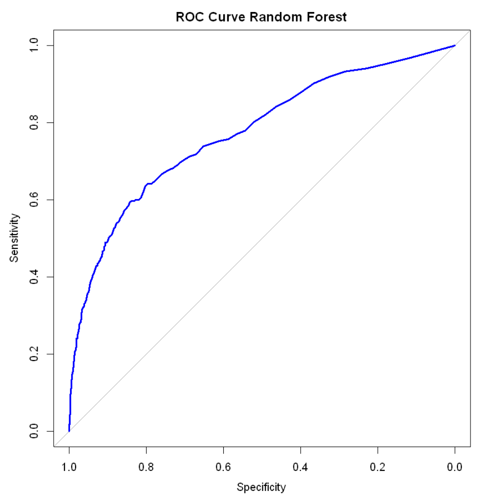
Using a $m = 5$, the random forest model is fitted on the train data without dummies with all the variables. To select only the most important variables, the Gini Index was used and selected the best 17 features:



Once having selected the best variables for the model, we fit the model again to the training data. As with the other models, the model was tested on the test set and it shows the following measures:

rf_pred	**0**	**1**
0	3,477	327
1	83	113

Accuracy	0.8975
Error	0.1025
AUC	0.7731



For this model, in the test data, the AUC turned to be 0.77 and similar values were obtained with a 10-fold cross-validation method:

auc.test.mean	0.7749
mmce.test.mean	0.1032
acc.test.mean	0.8968

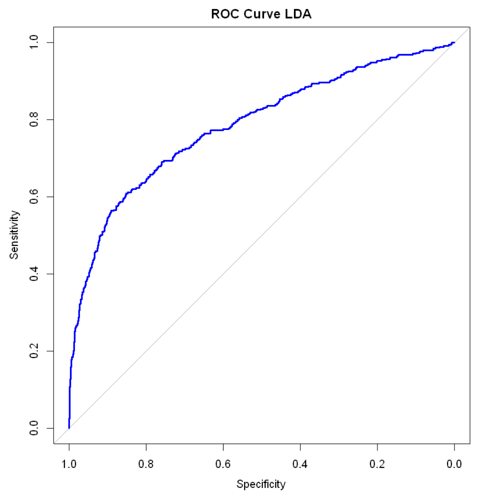
3.3.4. LINEAR DISCRIMINANT ANALYSIS

To use the LDA model we first apply it to the train table with all the variables. Then, for feature selection a backward stepwise process is used, which in this case only one variable was dropped (nr.employed). This means that 51 variables are going to be used to train the model.

So, with the selected variables we run a new LDA model, and use it to predict on the test set, from where we get the following:

lda_pred	**0**	**1**
0	3,411	279
1	149	161

Accuracy	0.8930
Error	0.1070
AUC	0.7839



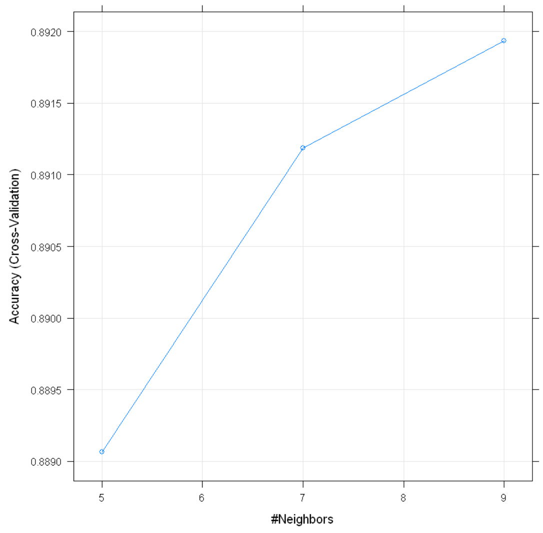
For this model, we obtained an AUC of 0.7839, whereas for the average AUC obtained with the cross-validation is 0.7783, slightly lower.

auc.test.mean	0.7783
mmce.test.mean	0.1093
acc.test.mean	0.8907

3.3.5. K-NEAREST NEIGHBORS

As explained before, the KNN is a non-parametric method, meaning that it directly predicts the class of new data point. In this case, the new data points are the ones from the test set with scaled numeric variables and the train points are from the train table with scaled variables.

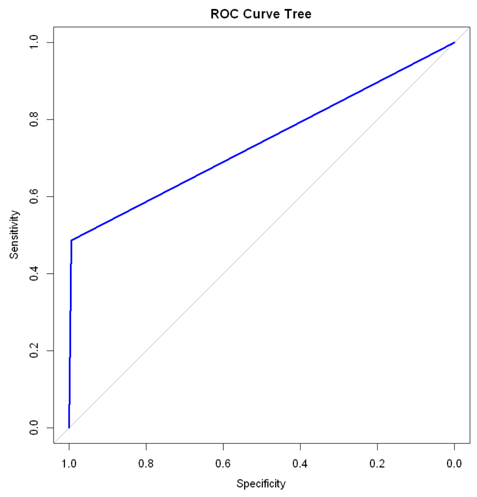
Now, to define the optimal K number of neighbors a 10 fold cross-validation was used and based on the accuracy $K = 9$ was selected.



Once defined K, we fit the model to get the predicted classes on the test set and get the following metrics:

knn_pred	0	1
0	3,538	226
1	22	214

Accuracy	0.9380
Error	0.0620
AUC	0.7401



With the KNN classification model, we get a high AUC, 0.7401. Nevertheless, when using a 10 fold cross-validation method we get a slightly lower AUC (0.7263).

auc.test.mean	0.7263
mmce.test.mean	0.1135
acc.test.mean	0.8864

3.4. EVALUATION METRICS COMPARISON

Now we are going to make a comparison between the metrics for all the selected models to find, in this case, which model would yield the best prediction of the bank’s customer subscribing or not.

Train and Test

Model	Accuracy	Error	AUC
Logistic Regression	0.8995	0.1005	0.7811
Decision Tree	0.9022	0.0978	0.7013
Random Forest	0.8975	0.1025	0.7731
LDA	0.8930	0.1070	0.7839
KNN	0.9380	0.0620	0.7401

From the presented results, and with the AUC as the main metric to take into consideration we can say that the model that best predicted the outcome is the LDA.

Cross-Validation

Model	Accurracy	Error	AUC
Logistic Regression	0.8996	0.1004	0.7852
Decision Tree	0.8992	0.1007	0.7033
Random Forest	0.8968	0.1032	0.7749
LDA	0.8907	0.1093	0.7783
KNN	0.8864	0.1135	0.7263

On another hand, while using a 10 fold cross-validation for all the models, based on the AUC the best model to predict the class is the logistic regression.

4. CONCLUSION

Once we have processed the data, applied different models, analyzing, and testing over the marketing information for the bank we can conclude that the models that performed the best were Linear Discriminant Analysis and the Logistic Regression. These models resulted in the best predictions to determine whether a client is going to subscribe or not.

Considering this experiment was made on a simple approach for all the models, it could be improved by feature engineering or hyper parameter tuning. By creating additional variables or modifying the parameters for each one of the models it is possible to get a higher AUC while being careful of not overfitting.

5. REFERENCES

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R. Springer.