Machine Learning

Bach, Helena * Fleming, María† Karampelas, Petro‡ Romero, Pablo José§

19/01/2022

*Università di Bologna, 975492
†Università di Bologna, xxx
‡Università di Bologna, 943760
§Università di Bologna, xxx

Abstract

This project uses a dataset about a Portuguese Banking Marketing Campaign, collecting demographic and social-economic information of the customer as well as records of direct marketing campaigngs phone calls. The goal of the project is to compare the performance of different models when predicting whether a customer will subscribe or not to a term deposit and identify which variables are more relevant. We consider a Lasso Logistic, Neural Networks, Classification Trees, Random Forests and Support Vector Mechanism. Some of the hyperparameters of the models are determined through cross-validation. To assess the execution of each model we use the ROC curve and the AUC. Random forest is the model that performs better with an AUC of 0.928. | Random Forests and Decision Trees allow us to perform feature selection and to have an idea of what variables are more important when determining the success of the marketing campaign. It has been concluded that the two main variables are *duration* and *poutcomesuccess*.

Introduction

Can machines be trained to predict the success of marketing campaigns? In this paper, we attempt to address this question using the Portuguese Bank Marketing dataset available in the UCI Machine Learning Repository. The dataset relates to direct phone call marketing campaigns, which aimed to promote term deposits among existing customers, by a Portuguese banking institution from May 2008 to November 2010. The dataset contains data specific to each customer as well as information related with the response of the marketing campaign. We will use these variables to train models to be able to predict and classify which clients are more likely to subscribe to the term deposit marketed in the campaign based on their response in these phone calls and their characteristics. For this purpose, Helena used a regularized logistic model on the dataset as a reference point, and then she supplemented this linear model with the more sophisticated non linear algorithm Neural Networks. Petros and Maria focused on Binomial Trees and Random Forest, while Pablo concentrated on Support Vector Machines.

Descriptive Analysis

The dimentionality of the dataset is such that it contains 45.211 observations across 17 variables, 11 of which are categorical and the remaining 6 are numerical. As the variables in the dataset are scaled differently, to enhance comparability, we re-scale the data and encode categorical variables to be treated as dummy variables. Moreover we duplicate the original dataset so we normalize the variables between 0 and 1 and use this dataset in further analysis

```
bank <- read.csv(file="bank-full.csv", header=TRUE, sep=";")
#variables that should be treated as categorical
bank$marital <- as.factor(bank$marital)
bank$education <- as.factor(bank$education)
bank$default <- as.factor(bank$default)
bank$housing <- as.factor(bank$housing)
bank$job <- as.factor(bank$job)
bank$month <- as.factor(bank$month)
bank$contact <- as.factor(bank$contact)
bank$loan <- as.factor(bank$loan)
bank$poutcome <- as.factor(bank$poutcome)
bank$y <- as.factor(bank$y)
#scale numeric variables
bank %>%
mutate_if(is.numeric, scale)
```

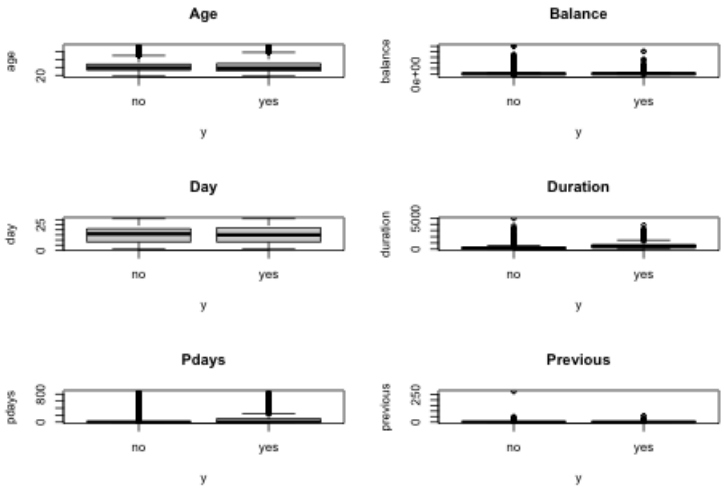
```
#normalize
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
bank_n <- mutate_if(bank, is.numeric, normalize)
#Dummy encoding
dummy_vars <- dummyVars(" ~ .", data=bank, fullRank=TRUE)
bank_m <- data.frame(predict(dummy_vars, newdata=bank))
bank_nn<-data.frame(predict(dummy_vars, newdata=bank_n))
```

We define the train and test sample in order to build and evaluate the model using different data.

```
#create test and training sample
set.seed(101)
sample<- sample(1:nrow(bank), nrow(bank)*0.75)
train <- bank_m[sample, ]
test <- bank_m[-sample, ]
```

Figure 1 shows the box plot of the numerical variables. It can be noticed that *age* follows a very similar distribution for clients who subscribed to a term deposit and for those who did not, indicating that the age of the customer won't play a significant role when predicting the behavior of clients. On the other hand, from the box-plot of *duration* it can be inferred that high duration (long contact duration in second) might have a positive impact on the subscription rate.

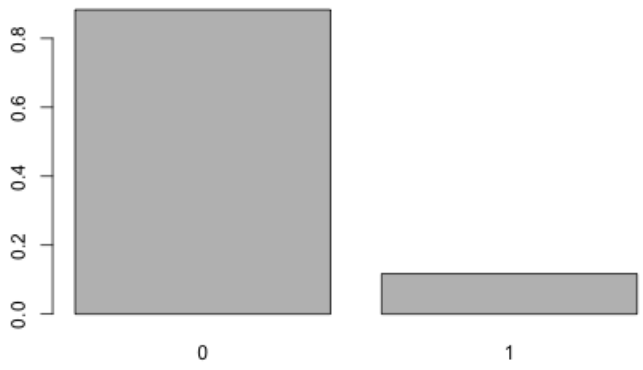
Figure 1: Boxplot



It is interesting to also have a graphical idea of how our variable of interest (whether the client subscribes or not) is distributed.

```
barplot(prop.table(table(bank_m$y.yes)))
```

Figure 2: Proportion



The plot shown above in Figure 2 evidences that we have imbalanced data. The average of our dependent dummy variable is 0.12, meaning that we only have 12% of observations that belong to class 1 (yes).

Logit

The logistic regression models the probability of belonging to one class, under the assumption that the dependent variable follows a binomial distribution. It is a classification technique that allows to predict a dichotomous variable.

Given X (explanatory variables) we can represent the probability that the client has subscribed to a term deposit (Y) as $p(X) = P(y = yes|X)$, using the logistic function to ensure that the output lies between 0 and 1.

$$p(X) = \frac{e^{\beta_0+\beta_1 \cdot X}}{1+e^{\beta_0+\beta_1 \cdot X}}$$

Logistic models are fit through maximum likelihood: $l(\beta) = \sum_{i=1}^{N} [y_i \beta^T x_i - \log(1 + e^{\beta^T x_i})]$ As it has been mentioned before, the dataset used in this project contains 17 variables. It is desirable to work with a reduced set of variables. For this purpose we will perform regularization through the LASSO approach. The logistic LASSO regression regularizes by maximizing a penalized version of the previous written equation for the likelihood function:

$$l(\beta) = \sum_{i=1}^{N} [y_i(\beta_0 \beta^T x_i - \log(1 + e^{\beta_0+\beta^T x_i}) - \lambda \sum_{j=1}^{p} |\beta_k|]$$

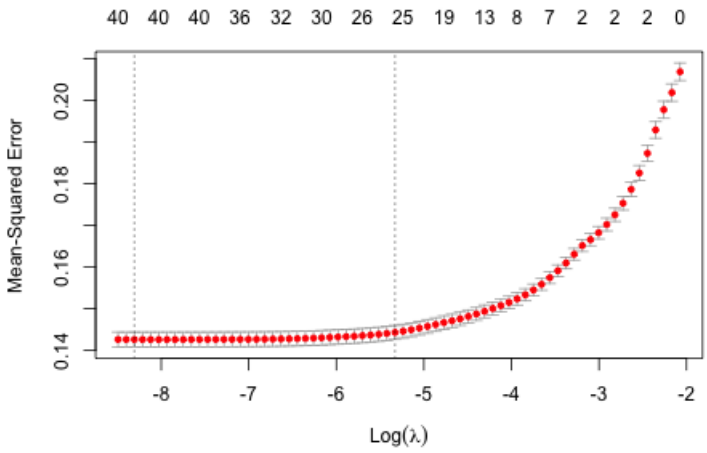
The coefficients contributing less to the model are forced to 0, allowing for only the most significant variables to be analyzed.

The tuning parameter λ is data dependent and therefore crucial to choose its value correctly. To assist us in making the correct choice, we perform cross-validation. As the value of λ increases, the regularization effect is strengthened . The function *cv.glmnet* carries out 10-fold cross validation by default, such that it divides our sample into 10 folds to determine the value of λ yielding the most predictive model.

```
#find the optimal lambda using cross-validation
set.seed(1)
cv.lambda <- cv.glmnet(x.train,y.train, alpha=1, family = "binomial",
                       type.measure="mse", expand.grid=c(10^-4, 1))
```

```
plot(cv.lambda)
```

Figure 3: Lambda



The plot in Figure 3 demonstrates how the Mean-Squared Error changes for different values of lambda. As λ increases, variables are excluded from the model and the MSE increases. *lambda.min* is the λ that minimizes the Mean Cross-Validation Error. However, it is recommended to choose the λ that is withing one standard error of the minimum, so to choose the simplest model (most regularized model) whose accuracy is equivalent to the best model.

```
#we choose the regularization factor lambda which minimizes the mean squared error
print(cv.lambda$lambda.min)
```

```
## [1] 0.0002467566
```

```
print(cv.lambda$lambda.1se)
```

```
## [1] 0.004843932
```

```
#we fit the logistic model with a binomial dependent variable
mylogit <-  glmnet(x.train,y.train,family = "binomial", alpha =1, lambda=cv.lambda$lambda.1se)
#we can get the coefficients of the regularized model
coef(mylogit, cv.lambda$lambda.1se)
```

```
## 43 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)         -2.736117e+00
```

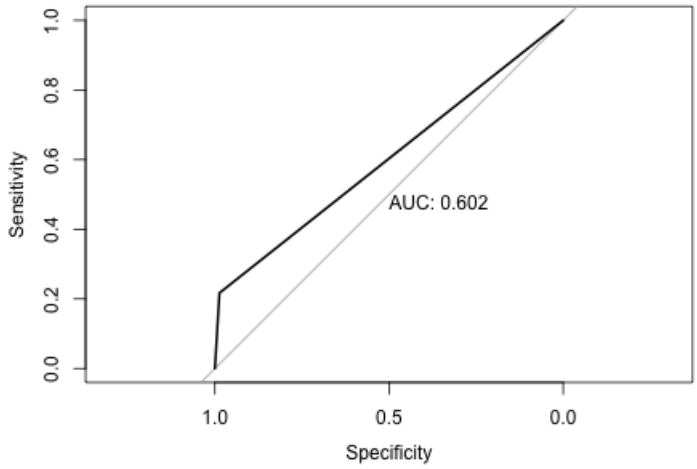
```
## age                   .
## job.blue.collar      -1.179750e-01
## job.entrepreneur      .
## job.housemaid         .
## job.management        .
## job.retired           2.118127e-01
## job.self.employed     .
## job.services          .
## job.student           4.515183e-01
## job.technician        .
## job.unemployed        .
## job.unknown           .
## marital.married      -1.217711e-01
## marital.single        8.920254e-02
## education.secondary   .
## education.tertiary    1.265837e-01
## education.unknown     .
## default.yes           .
## balance               4.039558e-06
## housing.yes          -5.754621e-01
## loan.yes             -2.927816e-01
## contact.telephone     .
## contact.unknown      -1.083279e+00
## day                   .
## month.aug            -1.290140e-01
## month.dec             7.570018e-01
## month.feb             1.555136e-02
## month.jan            -2.801866e-01
## month.jul            -2.496538e-01
## month.jun             3.016061e-01
## month.mar             1.792391e+00
## month.may             .
## month.nov            -1.823844e-01
## month.oct             1.126862e+00
## month.sep             1.023873e+00
## duration              3.720946e-03
## campaign             -4.379854e-02
## pdays                 .
## previous              8.442542e-03
## poutcome.other        .
## poutcome.success      2.209822e+00
## poutcome.unknown     -1.364436e-01
```

The output above shows how the LASSO shrinks some coefficients to 0. Interestingly, *age* shrinks 0, in line with the comment done in the descriptive analysis, suggesting that the age of the customer has no significant effect when determining whether he/she will subscribe to the term deposit advertised. | To evaluate the performance of the LASSO logistic model defined previously, we could create a confusion matrix and compare the predicted values obtained to the actual values. However, it is important to account for the fact that we have an imbalanced data. Because of the unequal distribution of classes, the accuracy measure is not reliable. For this reason, we use the ROC curve. The ROC curve summarizes the performance of a binary classification by plotting the False Positive Rate against the True Positive Rate. The Area Under the curve (AUC) is used as a diagnostic tool to compare different classification models.

```r
#prediction
probabilities <- mylogit %>% predict(x.test)
test$predicted <- as.vector(ifelse(probabilities > 0.5, 1, 0))
```

```r
# #AUC
g <- roc(test$y.yes, test$predicted, plot=TRUE, print.auc=TRUE)
```

Figure 4: ROC plot




```r
confusionMatrix(table(test$y.yes, test$predicted))
```

```
## Confusion Matrix and Statistics
##
##
##        0    1
##   0 9859  130
##   1 1029  285
##
##                Accuracy : 0.8975
##                  95% CI : (0.8917, 0.903)
##     No Information Rate : 0.9633
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2901
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9055
##             Specificity : 0.6867
##          Pos Pred Value : 0.9870
##          Neg Pred Value : 0.2169
##              Prevalence : 0.9633
##          Detection Rate : 0.8722
##    Detection Prevalence : 0.8837
```

```
##       Balanced Accuracy : 0.7961
##
##        'Positive' Class : 0
##
```

Figure 4 shows the ROC plot for the LASSO logistic regression with the corresponding AUC. An AUC of 0.5 indicates that the model is not able to discriminate between "yes" and "no". Our AUC is equal to 0.60 which means that the model can correctly distinguish between the two classes with a 60% chance. If we compute the confusion matrix, we obtain an accuracy of 0.897 and high measures of Sensitivity but very low performance of Negative Predicted Values (0.216). This exemplifies the use of AUC metric to account for imbalanced data.

Neural Networks

Neural Networks are biological inspired algorithms that recognize non-linear relationships between variables. In this project we will focus on Supervised learning (as we have labeled data) and we will use a Multi-Layer Neural Network. The algorithm works from left to right, starting in the input layer. In the output layer we use the Sigmoid activation function to keep values between 0 and 1, as we want to predict 2 different classes indicated as 0 and 1. Before proceeding to the model we ensured we normalised the numeric inputs to a standard scale so to improve the NN'accuracy. Next, We redefined the train and test sample on the scaled and normalized dataset.

The capacity of the neural network to learn is rooted in its design, that is, the number of layers and nodes. We use the *keras* package to model our NN. For simplicity, we have decided to model a Neural Network with one only hidden layer and we have used cross-validation to choose the number of neurons and the dropout rate. Adding a dropout rate avoids overfitting by randomly selecting neurons and set them equal to 0 with a given probability. The dropout rate that is commonly used is 50%. However, there is some evidence that low rates perform better. For this reason, we decided to try two different dropout rates: 10% and 50%.

```
library(tfruns)
runs <- tuning_run("Runs.R",
                   flags = list(dense_units2=c(16, 32),
                                dropout1=c(0.1, 0.5)))
```

In the first part of the code we define our model: we specify the layers (input, hidden and output) and how they are going to behave (by specifying the activation function). Notice that in the input and hidden layer we use the Rectified Linear (ReLu) function, which is the default activation function. However, in the output layer we specify the Sigmoid activation function as our output variables ranges from 0 to 1. In the second part of the code, we compile our model, defining the loss function and the metrics. The *binary cross Entropy* loss function is used in binary classification problems and we specify AUC as the metric. In the last part, the model is trained through the *fit* function. The process is iterated 10 different times and, to validate we are not overfitting, we hold out a 10% of the dataset.

```
for (i in 1:4){
model <- keras_model_sequential() %>%
  # Input layer
  layer_dense(units = runs$flag_dense_units2[i], activation = "relu", input_shape = 42) %>%
  # Hidden layer
  layer_dense(units = runs$flag_dense_units2[i], activation = "relu") %>%
  #Dropout layer (to avoid overfitting)
  layer_dropout(rate = runs$flag_dropout1[i])  %>%
  # Output layer
```

```
  layer_dense(units = 2, activation = "sigmoid")
model %>% compile(
  optimizer = optimizer_rmsprop(),
  loss = "binary_crossentropy",
  metrics = tf$keras$metrics$AUC())
#we inspect the training history
history <- fit(
  object           = model,
  x                = x.train,
  y                = y.train,
  batch_size       = 35,
  epochs           = 10,
  validation_split = 0.10
)
#we predict
model_pred<- as.integer(predict(model, x.test) > 0.5)
assign(paste("model_pred",i), model_pred)
}
```

We repeat the process 4 times in a loop in order to get the ROC curve for each model considered and be able to choose the optimal hyperparameter.

Figure 5: ROC plot

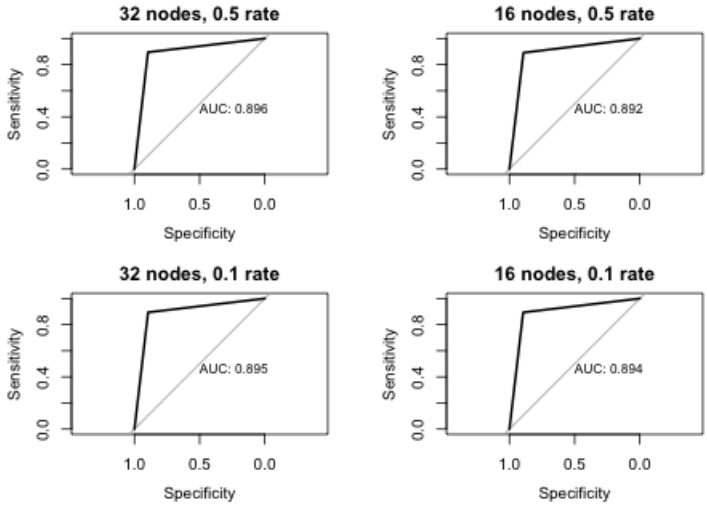
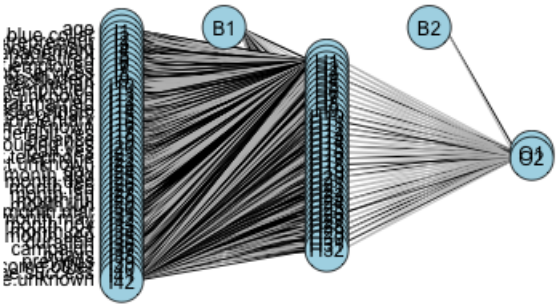


Figure 5 displays the ROC plot with its respective AUC for each of the 4 models. It can be seen that the one that performs better is the model containing 32 nodes with a dropout rate of 0.1 Figure 6 shows the structure of our Neural Network. We have 42 inputs (42 independent variables) and one hidden layer with 32 nodes.

```
plot.nnet(nn)
```

Figure 6: Neural Network



It can be seen that the Neural Network we designed performs better than the logistic regression as its AUC is higher, indicating that it is able to better distinguish between the two classes. However, Neural Networks have a main drawback: they are computationally expensive to train. In this project we have simplified the problem by designing a model with one single hidden layer and defining two possible number of neurons and dropout rates. Another drawback of Neural Networks that they can be difficult to interpret. In the logistic regression, the coefficient's size carried information regarding the importance of the variable in the classification result. Neural Networks seem to have a higher discrimination power and classification performance but the model cannot be interpreted.

Trees

Decision trees are a supervised learning method that can either be used for regression or classification problems. In this project we make use of the classification version of tree methods, as we want to predict a qualitative response variable.Their biggest advantage is the ease of interpreting the results, which can also be visualized and explained to a person who is not an expert in machine learning. In their simplest form, decision trees don't perform as well compared to other models such as linear regression. However, there are ways to vastly improve their predictability. For this project, 3 tree variants are applied and compared: | 1. A simple tree | 2. Pruning the above tree, using the results of cross validation with the misclassification error as a criterion | 3. Random Forest

Before assessing the results, the two latter methods are ensemble methods, meaning that many decision trees are being created and each tree's result is then aggregated to make a prediction for the output.

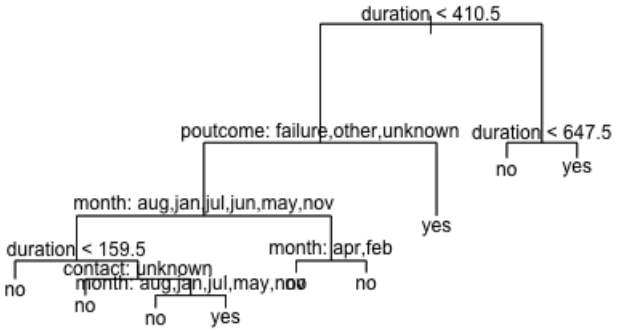
```
library(tree)
onetree <- tree(train$y ~.,train[,c(1:16)],split='deviance')
summary(onetree)
```

```
##
## Classification tree:
## tree(formula = train$y ~ ., data = train[, c(1:16)], split = "deviance")
## Variables actually used in tree construction:
## [1] "duration" "poutcome" "month"    "contact"
## Number of terminal nodes:  9
```

```
## Residual mean deviance:  0.4913 = 16660 / 33900
## Misclassification error rate: 0.1106 = 3752 / 33910
```

```
plot(onetree)
text(onetree , pretty = 0)
```

Figure 7: One simple tree




```
predict_onetree <- predict(onetree,test, type = "class")
```

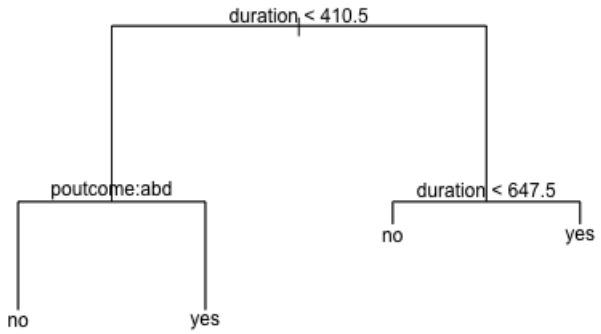
The first tree shown in Figure 7 is obtained through an algorithm that performs recursive partition of the predictor space with minimizing the deviance as a criterion in each step. This is a greedy, top-down approach as it performs this minimization in each partition. According to this model , the variable that could be used as good determinant of the outcome is duration. Also, the outcome of the previous marketing campaign seems to be important, as we infer that if a customer already has a certificate it is less likely for him to purchase another one.
Trees like this usually overfit the data and as a result they have high variance. This can also be corroborated by the depth of the tree, because trees that are this deep somewhat overfit. This is why cost-complexity pruning is performed on this tree, with the hopes of seeing a difference in performance and possibly obtaining a different tree structure and enriching our understanding of the data.

```
## $size
## [1] 9 8 4 3 1
##
## $dev
## [1] 3738 3738 3739 3760 4009
##
## $k
## [1] -Inf  0.0  0.5 65.0 95.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

In Figure 8, we perform cost-complexity pruning using the number of misclassifications as the criterion, as proposed in the recommended textbook. After obtaining the results, we prune the tree so that it has 4 terminal nodes. We find out that duration is considered the most important variable, followed by *poutcome*

```
plot(prunetree)
text(prunetree)
```

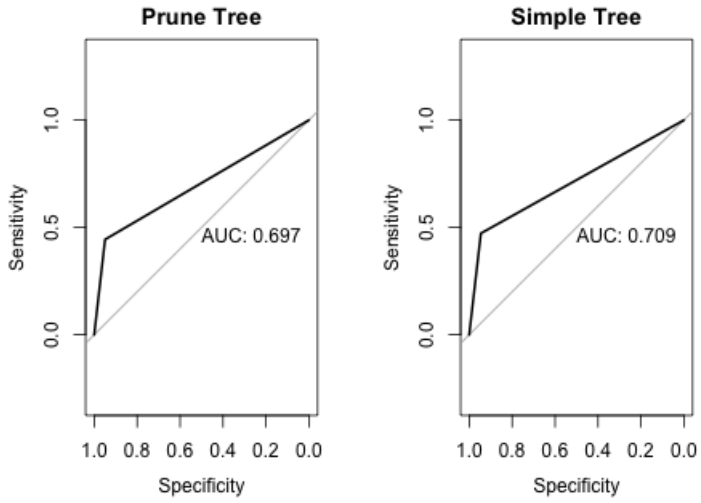
Figure 8: Prune Tree




```
predict_prunetree <- predict(prunetree,test, type = "class")
```

```
par(mfrow=c(1,2))
s1 <- roc(test$y, as.numeric(predict_prunetree), plot=TRUE, print.auc=TRUE, main="Prune Tree")
s2 <-roc(test$y, as.numeric(predict_onetree), plot=TRUE, print.auc=TRUE, main= "Simple Tree")
```

Figure 9: ROC curve



From Figure 9 it can be seen that both trees have a close metric, however the first tree scores better, as it captures more information fro the data. Hypothetically, if the pruning parameters (terminal nodes) changed then this result could be improved. Nonetheless, a reason for doing pruning apart from performance is interpretability, as it helps us map the most important variables.

Random Forest

Random forest is an ensemble machine learning algorithm that operates to build multiple decision trees that work both for regression and classification. Random forests are similar to bagged trees such that the random forest algorithm builds a number of decision trees on bootstrapped training samples. Random forests improve upon bagged trees such that the they have an added improvement that they de-correlates the trees which allow for greater accuracy. Moreover, when building these decision trees, each time a split in a tree occurs, a new random sample of mtry predictors is picked, these randomly chosen mtry predictors then become potential candidates for splitting from the full set of p predictors.

We make use of the *tuneRF* function to find the optimal number of variables randomly sampled as candidates at each split (mtry). The default value for classification is $\sqrt{p}$. In our case, the hyperparameter mtry can take any value from 1 to 17 (the number of predictors) so we expect the best value to be near

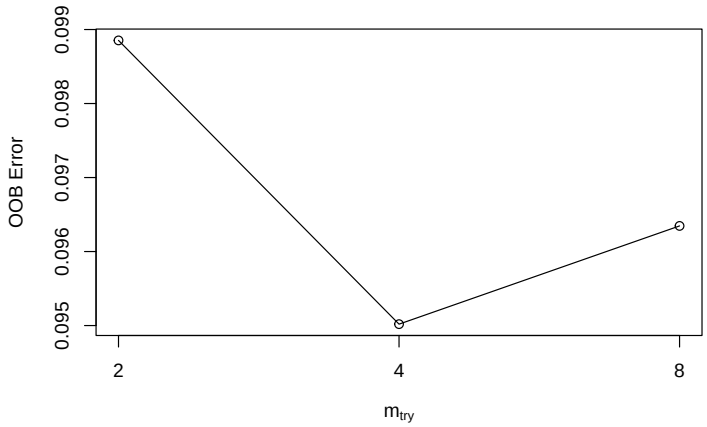
$$\sqrt{17} \sim 4$$

Figure 10 shows that the optimal value of predictors to be used is 4 (minimize the Out-Of-Bag error).

```
library(randomForest)
#Cross-validation for optimal mtry value(size of slitting variable subset)
cvforest <- tuneRF(bank_train[,c(1:16)], bank_train[,17],mtryStart=4,ntreeTry=100, stepFactor=2, improv
trace=TRUE, plot=TRUE, doBest=FALSE)
```

```
## mtry = 4  OOB error = 9.5%
## Searching left ...
## mtry = 2     OOB error = 9.89%
## -0.04034761 0.01
## Searching right ...
## mtry = 8     OOB error = 9.63%
## -0.01396648 0.01
```

Figure 10: Cross Validation Random Forest



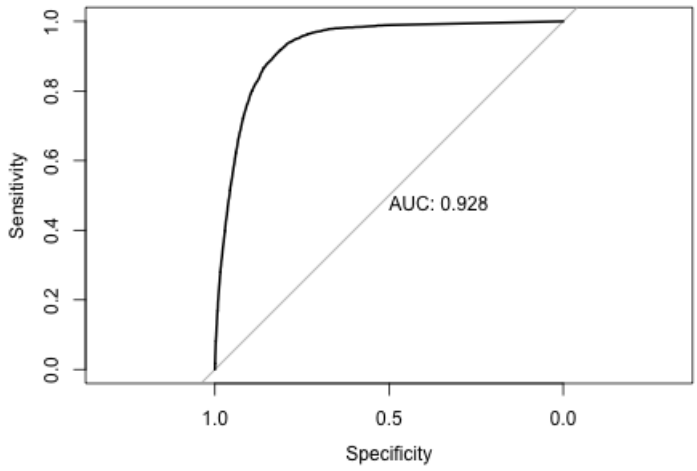
Having defined the optimal number of variables to be considered, we can proceed by creating a randomForest object:

```
forest <- randomForest(bank_train[,c(1:16)], bank_train[,17], ntree=150,mtry=4,importance=TRUE)
pred <- predict(forest,bank_test[,c(1:16)], type="vote",
norm.votes=FALSE, predict.all=FALSE, nodes=TRUE)
```

The *randomForest* function will create an object that contains votes. These votes are based on the Out of Bag (OOB) sample tree classification votes for each data point. These votes roughly represent a probability, and therefore can be used to create a ROC and AUC measure.

```
require(pROC)
rock<-roc(bank_train$y,forest$votes[,2],print.auc=TRUE, plot=TRUE)
```

Figure 11: ROC curve



From Figure 11 we can see the ROC curve and the AUC metric obtained for the Random Forest. It evidences that the model is highly able to correctly classify (92.8% of chance) and distinguish observations of the two different classes.
| One important advantage of Random Forest and Trees is that, differently from Neural Networks, these models offer a simple way to visualize the relative importance of the variables. Estimating the feature importance can provide us with a better understanding of the problem and simplify the model.

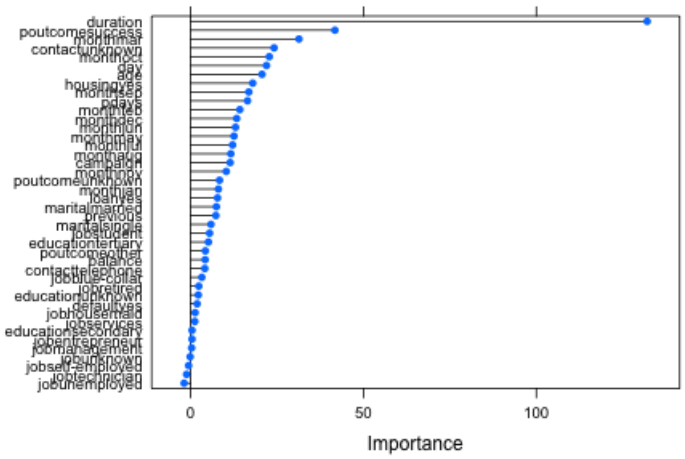
```
ctrl <- trainControl(method = "cv", number = 10, classProbs = TRUE)
mtryGrid <- data.frame(mtry = floor(seq(10, ncol(bank_train), length = 10)))
set.seed(22)
rf3 <- caret::train(y ~., data = bank_train, method = "rf", tuneGrid = mtryGrid, ntree = 200, importanc
rf_imp <- varImp(rf3, scale = FALSE, competes = FALSE)
rf_imp
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 42)
##
```

```
##                  Importance
## duration             132.067
## poutcomesuccess       41.737
## monthmar              31.345
## contactunknown        24.144
## monthoct              22.758
## day                   21.949
## age                   20.654
## housingyes            17.966
## monthsep              16.836
## pdays                 16.460
## monthfeb              14.237
## monthdec              13.326
## monthjun              12.984
## monthmay              12.546
## monthjul              12.136
## monthaug              11.628
## campaign              11.439
## monthnov              10.313
## poutcomeunknown        8.357
## monthjan               8.035
```

```
plot(rf_imp)
```

Figure 12: Most Important Variables: Random Forest



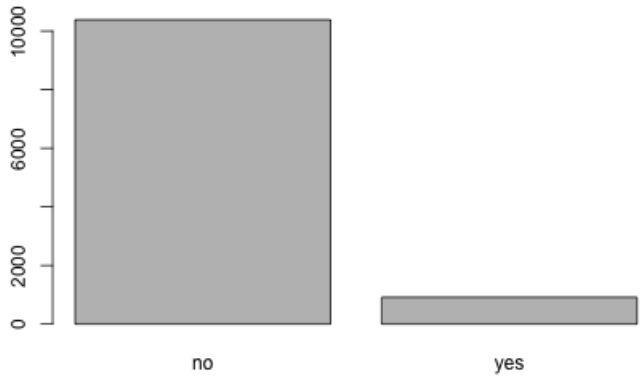
From Figure 12 it can be seen that the most important variable is *duration* (last contact duration), followed by *poutcomesuccess*. Thus, longer calls can be an indicator of the customer's interest in the product and be a sign of success of the marketing campaign. Also, taking into account the outcome of previous campaigns can also give us an insight of the result of the current marketing campaign. Next, we can use the Random forest to predict between the two classes.

```
rf_pred <- predict(rf3, newdata = bank_test)
confusionMatrix(rf_pred, bank_test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  9673  713
##        yes  307  609
##
##                Accuracy : 0.9098
##                  95% CI : (0.9043, 0.915)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.496
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9692
##             Specificity : 0.4607
##          Pos Pred Value : 0.9313
##          Neg Pred Value : 0.6648
##              Prevalence : 0.8830
##          Detection Rate : 0.8559
##    Detection Prevalence : 0.9190
##       Balanced Accuracy : 0.7150
##
##        'Positive' Class : no
##
```

```
plot(rf_pred)
```

Figure 13: Random Forest: Predictions



The data above displays some performance measures of the classification model. Before we've seen that the AUC is the highest we have obtained so far. We can also observe that the other indicators are also very high.

Support Vector Machine

The main idea of a support vector machine (SVM) is that it uses an hyperplane to classify observation in two classes. It uses non-linear kernels to enlarge a feature space to address non-linearity in the original space. It is an extension of the support vector classifier that uses a hyperplane in the presence of a linear boundary for classifying an observation in two classes. In doing so, it takes into account a cost parameter that represents the "size" of the violations of the margin that is formed between the hyperplane and the nearest observations of the support vector classifiers. This can be written as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

where $\langle x, x_i \rangle$ represents the inner product between the observation. Hence, *a priori* there are $\binom{n}{2}$ inner products to be computed. Nevertheless, the α_i is nonzero only for the support vectors, so we could rewrite $f(x)$ as follows:

$$f(x) = \beta_0 + \sum_{i \epsilon S} \alpha_i \langle x, x_i \rangle$$

where S is the subset of indices of the support vectors. This $f(x)$ determines the predicted class label for a test observation based on the sign of the function. The kernel function, of the form $K(x_i, x_{i'})$, quantifies the similarity between two observations. Common examples of kernels in SVM are linear, polynomial, radial and sigmoidal kernels. Combining support vector classifier with non-linear kernels results in the support vector machine classifier.

First, we will fit a Support Vector Classifier using a linear kernel in order to compare the radial and polynomial kernels that

give us the Support Vector Machine. Then we will compare the misclasification erros and AUC-ROC curves to decide which of these kernels is more appropriate.

```
svmfit.linear <- svm(y ~ ., data = train, kernel = "linear", cost = 1, decision.values = TRUE)
svmfit.radial <- svm(y ~ ., data = train, kernel = "radial",
                     gamma = 1, cost = 1, decision.values = TRUE)
svmfit.poly <- svm(y ~ ., data = train, kernel = "radial",
                     degree = 3, cost = 1, decision.values = TRUE)
```

The numbers of support vectors in the linear, radial and polynomial kernel are 7840 , 20779 and 7505, respectively,

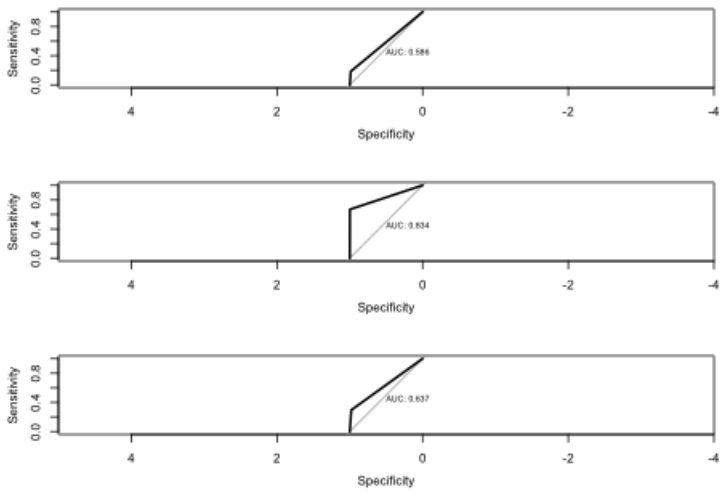
```
pred.linear <- predict(svmfit.linear, newdata = test)
pred.radial <- predict(svmfit.radial, newdata = test)
pred.poly <- predict(svmfit.poly, newdata = test)
merror.linear <- table(pred.linear,test$y)
merror.radial <- table(pred.radial,test$y)
merror.poly <- table(pred.poly,test$y)
merrors.svm <- c(sum(merror.linear[1,2],merror.linear[2,1])/ sum(merror.linear),
                     sum(merror.radial[1,2],merror.radial[2,1])/ sum(merror.radial),
                     sum(merror.poly[1,2],merror.poly[2,1])/ sum(merror.poly))

names(merrors.svm) <- c("Linear (SVC)","Radial (SVM)","Polynomial(SVM)")

minerror <- min(merrors.svm)
```

Observing only the misclassification error we would choose the polynomial kernel ($d = 3$) with a misclassification error of 0.0402.

Figure 14: ROC plot SVM



The first plot of 14 is the ROC curve generated from the linear kernel, the second one is from the radial kernel and the third is from the polynomial. The AUC are, in order, 0.5857804 , 0.8336286 and 0.6371349. Following the AUC-ROC curve criterion we should choose the support vector classifier and not the polynomial support vector machine. | One of them main disadvantages of the SVM method is that, as the support vector classifier works using the relative distance of observations, there is no probabilistic explanation to our models. However, this method is highly effective in high dimensional spaces, such as the one generated from our data set.

Conclusion:

In this paper we have used machine learning to predict and classify clients between two classes, that is we presented and attempted to train 5 models, logistical Regression , Neural Networks, Decision Trees, Random Forest and Structural Vector Machines to be able to predict which clients are more likely to respond well to the marketing campaign posed by the bank and subscribe to a term deposit.

	Logistic Regression	Neural Networks	Trees	Random Forest	SVM
AUC	0.602	0.895	0.697	0.928	0.609

The table above summarizes the AUC obtained in each model. As it has been previously mentioned, the AUC measures the ability of the model to distinguish between the two different classes. High values of AUC indicate that the classifier is capable of correctly discriminate the categories. In the table above, it can be seen that Random Forest is the model that performs better, followed by Neural Networks. Random Forest also gave us an insight about the variables that have a more significant role when determining the effect of the marketing campaign. The *duration* and the success of previous campaigns (*poutcomesuccess*) have a strong effect when determining the behavior of the customer. This information can be useful when planning future campaigns to reduce costs and have a better idea of what customers to target. Further analysis, in further studies, we would try to ensemble logistical regression, Neural Networks and Decision Trees as due to time constraints , it was outside the scope of this paper.

References

Chipman, H., George, E., & McCulloch, R. (2010, 03). Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, *4*. doi: 10.1214/09-AOAS285

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction* (2nd ed.). Springer. Retrieved from `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in r*. Springer. Retrieved from `https://faculty.marshall.usc.edu/gareth-james/ISL/`

Ramasubramanian, K., & Singh, A. (2017). *Machine learning using r*. doi: 10.1007/978-1-4842-2334-5

Tan, Y., & Roy, J. (2019, 08). Bayesian additive regression trees and the general bart model. *Statistics in Medicine*, *38*. doi: 10.1002/sim.8347