**Machine Learning Models**

Since the target variable in this data set is binary, this is a classification problem. The classification models used in this project are Logistic Regression, Decision Tree, Random Forest and Neural Network. The objective of the prediction is to increase the true positive and decrease the false negative. In addition to that, the challenge with this data set is the target variable is imbalanced. To overcome imbalance, Synthetic Minority Oversampling Technique (SMOTE) is used and evaluate the result on Neural Network.

**DATA PARTITION**

#Data partition

idata <- complete(idata, 5)

#Droping subscribed column  
idata <- subset(idata, select=-c(subscribed))

#75% of the sample size  
size <- floor(0.75 \* nrow(idata))  
  
#set the seed to make partition reproducible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(idata)), size = size)  
  
train <- idata[train\_ind, ]  
val <- idata[-train\_ind, ]

* 1. **LOGISTIC REGRESSION**

Logistic regression models the probabilities for classification problems with two possible outcomes. It’s an extension of the linear regression model for classification problems [8].

options(scipen=999)  
  
#Generalized linear model with family = "binomial"  
logit <- glm(y ~ ., data = train, family = "binomial")   
summary(logit)

##   
## Call:  
## glm(formula = y ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -5.7159 -0.3026 -0.1869 -0.1353 3.3677   
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) -227.57587319 43.77213316 -5.199  
## age -0.00157519 0.00275333 -0.572  
## jobblue-collar -0.24940877 0.09172337 -2.719  
## jobentrepreneur -0.04068733 0.13918063 -0.292  
## jobhousemaid -0.04320416 0.17118264 -0.252  
## jobmanagement -0.00941436 0.09690334 -0.097  
## jobretired 0.38285247 0.12131084 3.156  
## jobself-employed -0.18782217 0.13742678 -1.367  
## jobservices -0.16696249 0.09859138 -1.693  
## jobstudent 0.23466276 0.12440887 1.886  
## jobtechnician -0.02166036 0.08158783 -0.265  
## jobunemployed 0.15265233 0.14304999 1.067  
## maritalmarried -0.09384480 0.07716200 -1.216  
## maritalsingle 0.00281030 0.08807412 0.032  
## educationbasic.6y 0.07858596 0.13608353 0.577  
## educationbasic.9y 0.01992458 0.10706993 0.186  
## educationhigh.school 0.11046131 0.10284663 1.074  
## educationilliterate 1.00244817 0.87649186 1.144  
## educationprofessional.course 0.15887409 0.11341050 1.401  
## educationuniversity.degree 0.24909117 0.10377820 2.400  
## defaultyes -7.25475213 139.07767037 -0.052  
## housingyes 0.00248670 0.04695137 0.053  
## loanyes -0.01920225 0.06441407 -0.298  
## contacttelephone -0.60242185 0.08754319 -6.881  
## monthApr -2.01686229 0.16677459 -12.093  
## monthMay -2.50232868 0.14142763 -17.693  
## monthJun -2.49185648 0.24064856 -10.355  
## monthJul -1.74486572 0.17519956 -9.959  
## monthAug -1.23315830 0.14845404 -8.307  
## monthSep -1.67354866 0.17974286 -9.311  
## monthOct -1.90314518 0.17523029 -10.861  
## monthNov -2.41065630 0.16704835 -14.431  
## monthDec -1.70168197 0.24782431 -6.866  
## day\_of\_weekTue 0.18380563 0.07476134 2.459  
## day\_of\_weekWed 0.28942757 0.07432647 3.894  
## day\_of\_weekThu 0.17756961 0.07257277 2.447  
## day\_of\_weekFri 0.09919779 0.07600262 1.305  
## duration 0.00468969 0.00008552 54.837  
## campaign -0.04800140 0.01346887 -3.564  
## pdays -0.00104891 0.00025792 -4.067  
## previous 0.04331118 0.06958201 0.622  
## poutcomenonexistent 0.58026220 0.10937912 5.305  
## poutcomesuccess 0.86862142 0.25084791 3.463  
## emp.var.rate -1.67390305 0.16341065 -10.244  
## cons.price.idx 2.09323995 0.28894239 7.244  
## cons.conf.idx 0.02307599 0.00893014 2.584  
## euribor3m 0.23918459 0.14934027 1.602  
## nr.employed 0.00587935 0.00356780 1.648  
## Pr(>|z|)   
## (Intercept) 0.000000200251451 \*\*\*  
## age 0.567252   
## jobblue-collar 0.006545 \*\*   
## jobentrepreneur 0.770031   
## jobhousemaid 0.800742   
## jobmanagement 0.922606   
## jobretired 0.001600 \*\*   
## jobself-employed 0.171717   
## jobservices 0.090364 .   
## jobstudent 0.059265 .   
## jobtechnician 0.790636   
## jobunemployed 0.285915   
## maritalmarried 0.223907   
## maritalsingle 0.974545   
## educationbasic.6y 0.563613   
## educationbasic.9y 0.852375   
## educationhigh.school 0.282805   
## educationilliterate 0.252746   
## educationprofessional.course 0.161251   
## educationuniversity.degree 0.016385 \*   
## defaultyes 0.958399   
## housingyes 0.957761   
## loanyes 0.765622   
## contacttelephone 0.000000000005926 \*\*\*  
## monthApr < 0.0000000000000002 \*\*\*  
## monthMay < 0.0000000000000002 \*\*\*  
## monthJun < 0.0000000000000002 \*\*\*  
## monthJul < 0.0000000000000002 \*\*\*  
## monthAug < 0.0000000000000002 \*\*\*  
## monthSep < 0.0000000000000002 \*\*\*  
## monthOct < 0.0000000000000002 \*\*\*  
## monthNov < 0.0000000000000002 \*\*\*  
## monthDec 0.000000000006580 \*\*\*  
## day\_of\_weekTue 0.013949 \*   
## day\_of\_weekWed 0.000098603056011 \*\*\*  
## day\_of\_weekThu 0.014414 \*   
## day\_of\_weekFri 0.191828   
## duration < 0.0000000000000002 \*\*\*  
## campaign 0.000365 \*\*\*  
## pdays 0.000047672599046 \*\*\*  
## previous 0.533647   
## poutcomenonexistent 0.000000112639324 \*\*\*  
## poutcomesuccess 0.000535 \*\*\*  
## emp.var.rate < 0.0000000000000002 \*\*\*  
## cons.price.idx 0.000000000000434 \*\*\*  
## cons.conf.idx 0.009765 \*\*   
## euribor3m 0.109242   
## nr.employed 0.099375 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21946 on 30890 degrees of freedom  
## Residual deviance: 12951 on 30843 degrees of freedom  
## AIC: 13047  
##   
## Number of Fisher Scoring iterations: 10

From the above coefficients it is evident that there are several features that are significant to predict the target variable. One of the features is duration. It can be interpreted that for every unit of duration (last contact duration, in seconds) increase, the log of odds of being subscribed increases by 0.00468969. However, for every unit of campaign (number of contacts performed during this campaign) increase, the log of odds of being subscribed decreases by 0.04800140.

#Prediction

pred <- predict(logit, val[, -ncol(val)], type = "response")  
pred <- as.factor(ifelse(pred > 0.5,'yes','no'))

#Evaluation

library(caret)

##   
## Attaching package: 'caret'

## The following object is masked from 'package:survival':  
##   
## cluster

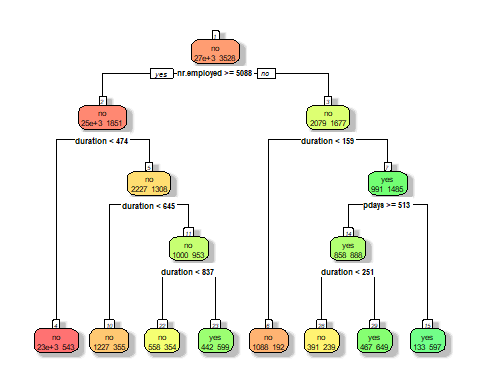
confusionMatrix(pred, val[, ncol(val)], positive='yes')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 8926 626  
## yes 259 486  
##   
## Accuracy : 0.9141   
## 95% CI : (0.9085, 0.9194)   
## No Information Rate : 0.892   
## P-Value [Acc > NIR] : 0.00000000000005707   
##   
## Kappa : 0.4782   
##   
## Mcnemar's Test P-Value : < 0.00000000000000022  
##   
## Sensitivity : 0.43705   
## Specificity : 0.97180   
## Pos Pred Value : 0.65235   
## Neg Pred Value : 0.93446   
## Prevalence : 0.10799   
## Detection Rate : 0.04720   
## Detection Prevalence : 0.07235   
## Balanced Accuracy : 0.70443   
##   
## 'Positive' Class : yes

From the above confusion matrix, the true positives are 486 and the false negatives are 626. The objective is to increase the true positives and decrease the false negatives. Thus, the accuracy metric that is suitable for this problem is Sensitivity (or) Recall. The sensitivity for logistic regression is 0.43705. Sensitivity = TP / ( TP + FN )

* 1. **DECISION TREE**

In statistics, Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves) [9].

library(rpart)  
library(rpart.plot)  
  
#Decision Tree  
dt <- rpart(y ~ ., data = train ,method = "class", parms = list(split = 'information'))  
  
#Visualize the decision tree with rpart.plot  
rpart.plot(dt, box.palette="RdYlGn", shadow.col="gray", nn=TRUE, extra= 1)

From the above decision tree plot it can be inferred that the algorithm decides *nr.employed* feature to be the best feature to first split the entire data set. There are 27363 negative classes and 3528 positive classes in the training data set before the split up. The *nr.employed* variable is split up in to two classes, one with approximately 25k negative classes & 1851 positive classes when *nr.employed* is greater than or equal to 5088 and the other with 2079 negative classes and 1677 positive classes when *nr.employed* is lesser than 5088. The next level of split up happens at the *duration* variable.

#Prediction

pred <- predict(dt, val[, -ncol(val)], type = 'class')

#Evaluation

confusionMatrix(pred, val[, ncol(val)], positive='yes')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 8853 519  
## yes 332 593  
##   
## Accuracy : 0.9174   
## 95% CI : (0.9119, 0.9226)   
## No Information Rate : 0.892   
## P-Value [Acc > NIR] : < 0.00000000000000022  
##   
## Kappa : 0.5368   
##   
## Mcnemar's Test P-Value : 0.0000000001818   
##   
## Sensitivity : 0.53327   
## Specificity : 0.96385   
## Pos Pred Value : 0.64108   
## Neg Pred Value : 0.94462   
## Prevalence : 0.10799   
## Detection Rate : 0.05759   
## Detection Prevalence : 0.08983   
## Balanced Accuracy : 0.74856   
##   
## 'Positive' Class : yes   
##

From the above confusion matrix, it can be inferred that Decision Tree algorithm has provided a true positive of 593 and false negatives of 519. The sensitivity score is 0.53327 which is an improvement from logistic regression where the sensitivity score is 0.43705. Decision tree has performed well compared to Logistic Regression. However, Decision tree tends to overfit the model. One of the ways to reduce overfitting is pruning the tree. In addition to that tuning the hyperparameters can lead to better results in decision tree.

* 1. **RANDOM FOREST**

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result [10].

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

rfm <- randomForest(y ~ ., data = train, ntree = 500, mtry = 6, importance = TRUE)

#Prediction

pred <- predict(rfm, val[, -ncol(val)], type = 'class')

#Evaluation

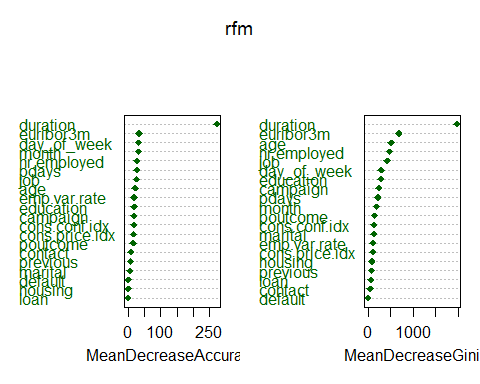
confusionMatrix(pred, val[, ncol(val)], positive='yes')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 8846 488  
## yes 339 624  
##   
## Accuracy : 0.9197   
## 95% CI : (0.9143, 0.9249)   
## No Information Rate : 0.892   
## P-Value [Acc > NIR] : < 0.00000000000000022  
##   
## Kappa : 0.557   
##   
## Mcnemar's Test P-Value : 0.0000002654   
##   
## Sensitivity : 0.56115   
## Specificity : 0.96309   
## Pos Pred Value : 0.64798   
## Neg Pred Value : 0.94772   
## Prevalence : 0.10799   
## Detection Rate : 0.06060   
## Detection Prevalence : 0.09352   
## Balanced Accuracy : 0.76212   
##   
## 'Positive' Class : yes   
##

From the above confusion matrix, it can be inferred that Random Forest algorithm has provided a true positive of 624 and false negatives of 488. The sensitivity score is 0.56115 which is an improvement from Decision Tree where the sensitivity score is 0.53327. Random Forest has performed well compared to Decision Tree. In addition to this tuning the hyperparameters can lead to better results in Random forest.

#Variable Importance Plot

varImpPlot(rfm,pch=18,col='darkgreen')



The above plot shows the variable importance in respective to mean decrease in accuracy and mean decrease in gini. *Duration, euribor3m, day of week, month and nr.employed* are the top 5 important variables. There is a huge difference in mean decrease accuracy between the variable *duration* and the rest of the variables. Thus, duration might be a very good variable for predicting whether an user will subscribe or not.

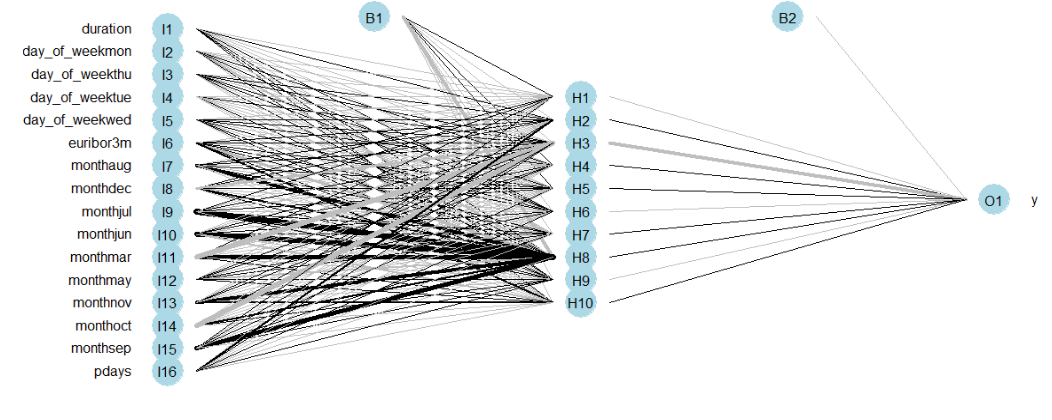
* 1. **NEURAL NETWORK**

A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria [11]

library(nnet)  
nn <- nnet(y ~ duration + day\_of\_week + euribor3m + month + pdays, data = train, size = 10)

## # weights: 201  
## initial value 29940.194132   
## iter 10 value 10258.462736  
## iter 20 value 8457.804082  
## iter 30 value 6931.968796  
## iter 40 value 6817.233794  
## iter 50 value 6770.535291  
## iter 60 value 6764.007423  
## iter 70 value 6716.495396  
## iter 80 value 6565.707717  
## iter 90 value 6551.823078  
## iter 100 value 6543.232680  
## final value 6543.232680   
## stopped after 100 iterations

library(NeuralNetTools)  
plotnet(nn)



The top 5 important variables from the random forest model are used as the predictor variables in the neural network model to see if there is any improvement in the recall compared to Random Forest. The above diagram shows a neural network model with one hidden layer with 10 units.

#Prediction

pred <- predict(nn, val[, -ncol(val)], type = 'class')

#Evaluation

confusionMatrix(factor(pred), val[, ncol(val)], positive='yes')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 9114 909  
## yes 71 203  
##   
## Accuracy : 0.9048   
## 95% CI : (0.899, 0.9104)   
## No Information Rate : 0.892   
## P-Value [Acc > NIR] : 0.00001081   
##   
## Kappa : 0.2614   
##   
## Mcnemar's Test P-Value : < 0.00000000000000022  
##   
## Sensitivity : 0.18255   
## Specificity : 0.99227   
## Pos Pred Value : 0.74088   
## Neg Pred Value : 0.90931   
## Prevalence : 0.10799   
## Detection Rate : 0.01971   
## Detection Prevalence : 0.02661   
## Balanced Accuracy : 0.58741   
##   
## 'Positive' Class : yes   
##

From the above confusion matrix, it can be inferred that Neural Network algorithm has provided a true positive of 203 and false negatives of 909. The sensitivity score is 0.18255 which is not an improvement from Decision Tree or Random Forest. So far, Random Forest has performed well compared all other algorithm. The main reason why all the algorithms give a low recall value is because of the imbalance in the data set. Therefore, a sampling technique is required to get a decent recall value. One of the famous sampling techniques is Synthetic Minority Oversampling Technique, which will be used in the next session followed by another Neural Network model with the sampled train data set.

* 1. **NEURAL NETWORK AFTER SAMPLING TECHNIQUE**

**SMOTE**

SMOTE synthesizes new minority instances between existing (real) minority instances [12]. SMOTE is a technique based on nearest neighbours judged by Euclidean Distance between data points in feature space. There is a percentage of Over-Sampling which indicates the number of synthetic samples to be created and this percentage parameter of Over-sampling is always a multiple of 100. If the percentage of Over-sampling is 100, then for each instance, a new sample will be created. Hence, the number of minority class instances will get doubled. Similarly, if the percentage of Over-sampling is 200, then the total number of minority class samples will get tripled [13].

library(DMwR)

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

##   
## Attaching package: 'DMwR'

## The following object is masked from 'package:VIM':  
##   
## kNN

strain <- SMOTE(y ~ ., train,K = 5, perc.over = 100, perc.under=600)  
table(train$y)

##   
## no yes   
## 27363 3528

table(strain$y)

##   
## no yes   
## 21168 7056

The above SMOTE algorithm oversamples the minority and under samples the majority. Before applying SMOTE to the training data set, the number of negative classes are 27363 and the number of positive classes are 3528. After applying SMOTE, the number of negative classes become 21168 and the number of negative classes become 7058. The amount of oversampling and under-sampling can be changed using *perc.over and perc.under* parameter in the SMOTE function.

*K*: A number indicating the number of nearest neighbours that are used to generate the new examples of the minority class [14]

*perc.over*: A number that drives the decision of how many extra cases from the minority class are generated (known as over-sampling) [14]

*perc.under*: A number that drives the decision of how many extra cases from the majority classes are selected for each case generated from the minority class (known as under-sampling) [14]

* 1. **NEURAL NETWORK AFTER SMOTE**

nn <- nnet(y ~ duration + day\_of\_week + euribor3m + month + pdays, data = strain, size = 10)

## # weights: 201  
## initial value 15750.833700   
## iter 10 value 11172.864731  
## iter 20 value 9674.581555  
## iter 30 value 9034.112204  
## iter 40 value 8880.252904  
## iter 50 value 8774.977823  
## iter 60 value 7998.844758  
## iter 70 value 7658.853484  
## iter 80 value 7553.015977  
## iter 90 value 7493.129325  
## iter 100 value 7412.061343  
## final value 7412.061343   
## stopped after 100 iterations

#Prediction

pred <- predict(nn, val[, -ncol(val)], type = 'class')

#Evaluation

confusionMatrix(factor(pred), val[, ncol(val)], positive='yes')

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 8578 352  
## yes 607 760  
##   
## Accuracy : 0.9069   
## 95% CI : (0.9011, 0.9124)   
## No Information Rate : 0.892   
## P-Value [Acc > NIR] : 0.0000003878633836030  
##   
## Kappa : 0.5608   
##   
## Mcnemar's Test P-Value : 0.0000000000000002362  
##   
## Sensitivity : 0.68345   
## Specificity : 0.93391   
## Pos Pred Value : 0.55596   
## Neg Pred Value : 0.96058   
## Prevalence : 0.10799   
## Detection Rate : 0.07381   
## Detection Prevalence : 0.13276   
## Balanced Accuracy : 0.80868   
##   
## 'Positive' Class : yes   
##

From the above confusion matrix, it can be inferred that Neural Network algorithm after SMOTE has provided a true positive of 760 and false negatives of 352. The sensitivity score is 0.68345 which is an improvement from all other algorithms. Also, Neural Network after SMOTE has performed very well compared to Neural Network without SMOTE. In addition to this, tuning the hyperparameters can lead to better results.

1. **COMPARISION OF THE MODELS**

|  |  |  |  |
| --- | --- | --- | --- |
| Models | True Positive | False Negative | Recall |
| Logistic Regression | 486 | 626 | 0.43705 |
| Decision Tree | 593 | 519 | 0.53327 |
| Random Forest | 624 | 488 | 0.56115 |
| Neural Network | 203 | 909 | 0.18255 |
| Neural Network after SMOTE | 760 | 352 | 0.68345 |

The above tables compare the models that were applied to the dataset. Neural Network after sampling gives a better result compared to all other algorithms. This is because of the imbalance in the dataset. The recall achieved using Neural network after SMOTE indicates the ratio of the model truly predicts an event to the total number of true events is 68.3%. Even though the recall value is too low, this can still be improved by hyperparameter tuning in Random Forest or Neural Network.

Logistic Regression performs poor with the data set. Regularization can be applied to generalize the model by adding penalty. In the case of ridge regression, the effect of the penalty term is to shrink the coefficients that contribute most to the error. Put another way, it reduces the magnitude of the coefficients that contribute to increasing L. In the case of lasso regression, the effect of the penalty term is to set these coefficients exactly to zero. It means that lasso regression works like a feature selector that picks out the most important coefficients, i.e. those that are most predictive (and have the lowest p-values) [15].

There are multiple boosting algorithms like Ada Boost, Gradient Boosting algorithm, XGBoost etc. which gives good results since they sequentially learn from bad learners. In addition to this, changing the amount of oversampling using SMOTE in the training data set and test with cross validation can result in good recall value.