HR report on Worker Churn

A Regression Analysis

by Leonard Langat Maina

PART ONE:

ABSTRACT

This is a report of the Human Resource key performance indicators that I will use to measure the performance of an organization. The objective of the report is to build machine learning algorithms to predict with a certain degree of accuracy whilst maintaining the robustness of the model, hence checking for the model parameters; sensitivity and specificity. The original data is obtained from the Predictive Analytics Lab Website.

Importing data into RStudio

```
hr<-read.csv("data/hr_comma_sep.csv",stringsAsFactors = F,na.strings =
c("","NA",""))
```

ANALYSIS OF THE DATA

Data conversion

First I convert data types variables; left, department and salary into factors, view the structure of my data. Left variable will be my dependent variable in my linear equation with which I predict over the other independent variable. The churn problem is of a classification type. (Supervised Learning)

```
hr$department=as.factor(hr$department)
hr$salary=as.factor(hr$salary)
hr$left=as.factor(hr$left)
str(hr)
```

```
## 'data.frame':    14999 obs. of  10 variables:
##  $ satisfaction_level   : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89
0.42 ...
##  $ last_evaluation      : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1
0.53 ...
##  $ number_project       : int  2 5 7 5 2 2 6 5 5 2 ...
##  $ average_montly_hours : int  157 262 272 223 159 153 247 259 224 142 ...
##  $ time_spend_company   : int  3 6 4 5 3 3 4 5 5 3 ...
##  $ Work_accident        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left                 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2
...
##  $ promotion_last_5years: int  0 0 0 0 0 0 0 0 0 0 ...
```

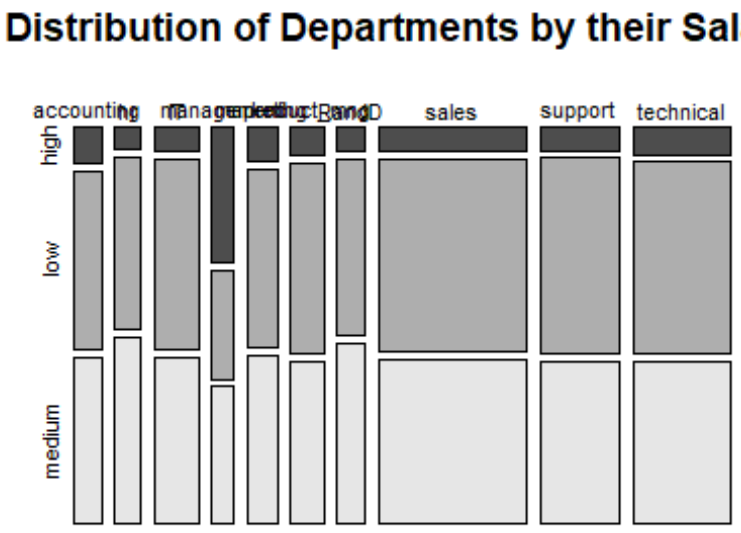
```
##  $ department           : Factor w/ 10 levels "accounting","hr",..: 8 8 8
8 8 8 8 8 8 8 ...
##  $ salary               : Factor w/ 3 levels "high","low","medium": 2 3 3
2 2 2 2 2 2 2 ...
```

Then I generate a frequency table to show the distribution of the data and create a visual by generating a mosaic plot of the distribution of department and their salary.

```
ftable<-table(hr$department,hr$salary)
ftable
```

```
##
##                high  low medium
##   accounting     74  358    335
##   hr             45  335    359
##   IT             83  609    535
##   management    225  180    225
##   marketing      80  402    376
##   product_mng    68  451    383
##   RandD          51  364    372
##   sales         269 2099   1772
##   support       141 1146    942
##   technical     201 1372   1147
```

```
mosaicplot(ftable,main="Distribution of Departments by their
Salary",color=TRUE)
```



Now I perform a chi-square test to understand my hypothesis of whether departments and salary are dependent, in statistical theory; a p-value < 0.5 shows dependance, I surmise that my initial hypothesis stands.

```
chisq.test(ftable)
```

```
##
##  Pearson's Chi-squared test
##
## data:  ftable
## X-squared = 700.92, df = 18, p-value < 2.2e-16
```

To predict churn in the organisation I perform a logistic regression, a linear expression of the left variable modeled on top of the existing observations through an in sample error distribution and I seek to improve my model with an out of sample error distribution, data that my algorithm will be passed through for deployment on real life scenarios and unseen data to merit the robustness of my model.

This is a classification problem because its either of the two outcomes; left denoted by 0 and stayed denoted by 1. The family of distribution I will employ in my analysis is from the binomial distribution.

In-sample distibution

```
## in-sample distribution
logit_model<-glm(left~satisfaction_level+last_evaluation+number_project+
                 Work_accident+promotion_last_5years+average_montly_hours+
                   time_spend_company,data = hr, family = "binomial")
#summary(logit_model)

##predict churn
hr$left_pred<-predict(logit_model,data=hr,type="response")

##changing the probabilities into class (0 or 1)
hr$left_pred<-ifelse(hr$left_pred>0.5,1,0)
hr$left_pred=as.factor(hr$left_pred)
#str(hr)

#confusion matrix
table(hr$left,hr$left_pred)
```

```
##
##         0     1
##   0 10581   847
##   1  2670   901
```

```
misClassError<-mean(hr$left_pred!=hr$left)
print(paste('Accuracy=',1-misClassError))
```

```
## [1] "Accuracy= 0.765517701180079"
```

My prediction from doing an in-sample error is Accuracy= 0.765517701180079. With this accuracy, I will perform an out of sample error estimate to check the accuracy the algorithm whilst maintaining the sensitivity and specificity, the model parameter that show if my model is best accurate and robust. A perfect model is not only accurate(overfitted) in its prediction but is also robust.

Out of sample distribution

This is where I split my data into training set and test set, the idea is to train my data using the training set and making my predicts with the unseen data that my algorithm is unfamiliar with to see how robust my model is for deployment to the real life scenario.

```
#Splitting the data in train and test data
#install.packages('caTools')
library(caTools)
set.seed(123)
split<-sample.split(hr,SplitRatio=0.75)
split
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
```

```
train<-subset(hr,split=="TRUE")
test<-subset(hr,split=="FALSE")
str(train)
```

```
## 'data.frame':    10909 obs. of  11 variables:
##  $ satisfaction_level   : num  0.38 0.8 0.11 0.72 0.41 0.1 0.89 0.42 0.11
0.84 ...
##  $ last_evaluation      : num  0.53 0.86 0.88 0.87 0.5 0.77 1 0.53 0.81
0.92 ...
##  $ number_project       : int  2 5 7 5 2 6 5 2 6 4 ...
##  $ average_montly_hours : int  157 262 272 223 153 247 224 142 305 234 ...
##  $ time_spend_company   : int  3 6 4 5 3 4 5 3 4 5 ...
##  $ Work_accident        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left                 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2
...
##  $ promotion_last_5years: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ department           : Factor w/ 10 levels "accounting","hr",..: 8 8 8
8 8 8 8 8 8 8 ...
##  $ salary               : Factor w/ 3 levels "high","low","medium": 2 3 3
2 2 2 2 2 2 2 ...
##  $ left_pred            : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 1 2 1
...
```

```
str(test)
```

```
## 'data.frame':    4090 obs. of  11 variables:
##  $ satisfaction_level   : num  0.37 0.92 0.45 0.38 0.45 0.38 0.82 0.38 0.4
0.45 ...
##  $ last_evaluation      : num  0.52 0.85 0.54 0.54 0.51 0.55 0.87 0.5 0.51
0.5 ...
```

```
##  $ number_project       : int  2 5 2 2 2 2 4 2 2 2 ...
##  $ average_montly_hours : int  159 259 135 143 160 147 239 132 145 126 ...
##  $ time_spend_company   : int  3 5 3 3 3 3 5 3 3 3 ...
##  $ Work_accident        : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ left                 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2
...
##  $ promotion_last_5years: int  0 0 0 0 1 0 0 0 0 0 ...
##  $ department           : Factor w/ 10 levels "accounting","hr",..: 8 8 8
8 8 8 8 1 2 10 ...
##  $ salary               : Factor w/ 3 levels "high","low","medium": 2 2 2
2 2 2 2 2 2 2 ...
##  $ left_pred            : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
```

I train my data using the training dataset with a generalized linear model(glm) function and thereafter making my predictions on the test data based on the trained dataset.

```
##Train model using glm function

logit_model_tr<-glm(left~satisfaction_level+last_evaluation+number_project+
                  Work_accident+promotion_last_5years+average_montly_hours+
                    time_spend_company,data = train, family = "binomial")
#logit_model_tr

##Predicting test data based on trained model

predicted.result<-predict(logit_model_tr,test,type = "response")
predicted.result<-ifelse(predicted.result>0.5,1,0)

##Evaluating model accuracy using confusion matrix
#table(test$left,predicted.result)
misClassError<-mean(predicted.result!=test$left)
#print(paste('Accuracy=',1-misClassError))
```

MODEL EVALUATION

Evaluating my model using a confusion Matrix.

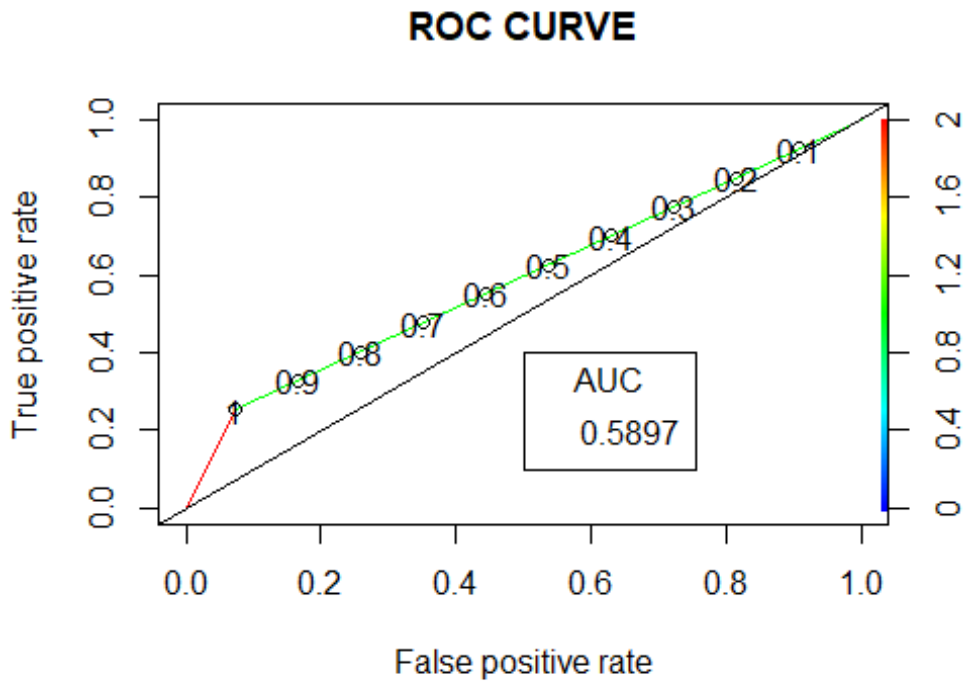
```
library(caret)
confusionMatrix(table(test$left,predicted.result))

## Confusion Matrix and Statistics
##
##    predicted.result
##        0    1
##   0 2885  231
##   1  727  247
##
##                Accuracy : 0.7658
##                  95% CI : (0.7525, 0.7787)
```

```
##     No Information Rate : 0.8831
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2175
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7987
##             Specificity : 0.5167
##          Pos Pred Value : 0.9259
##          Neg Pred Value : 0.2536
##              Prevalence : 0.8831
##          Detection Rate : 0.7054
##    Detection Prevalence : 0.7619
##       Balanced Accuracy : 0.6577
##
##        'Positive' Class : 0
##
```

ROC-AUC CURVE

A visual representation of the model using a Receiver Operating Characteristic (ROC) curve and Area under Curve (AUC) label that displays the actual of the area in the curve.
N/B: Best models have auc values closer to 1.



PART TWO:

DECISION TREE

In my analysis I use this algorithm for illustrative purpose to help us understand Random forest, an algorithm based on decision trees. A random forest is an agglomerative process whereby you generate many decision trees based on various decision nodes to solve a classification problem.

Loading my data into RStudio

Step 1: Splitting the data into train and test

```
library(caTools)
set.seed(1234)
split<-sample.split(hr,SplitRatio = 0.75)
split
```

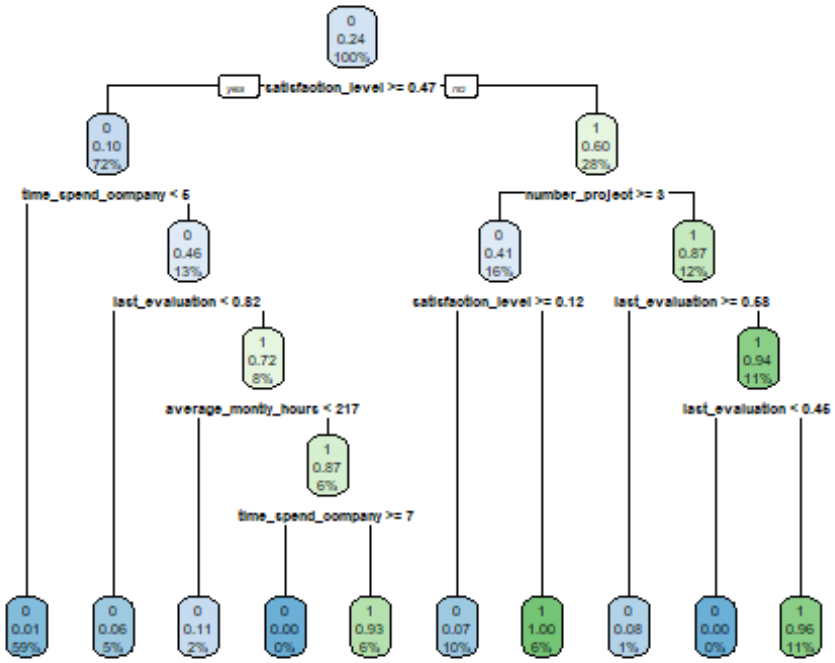
```
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
```

```
train<-subset(hr,split=="TRUE")
test<-subset(hr,split=="FALSE")
#str(train)
#str(test)
```

Step 2: Training model with logistics regression using glm function

A plot of the decision tree

```
library("rpart") ##recursive partioning
library("rpart.plot")
DecTreeModel<-rpart(left~.,data=train,method = "class")
rpart.plot(DecTreeModel)
```



The decision tree is constructed in a top-down manner and involves the following process:
1. Placing all the trained examples at the root
2. Categorizing the attributes
3. Partition the examples recursively based on the selected attributes
4. Select test attributes on the basis of a heuristic or statistical measure

```
#summary(DecTreeModel)
```

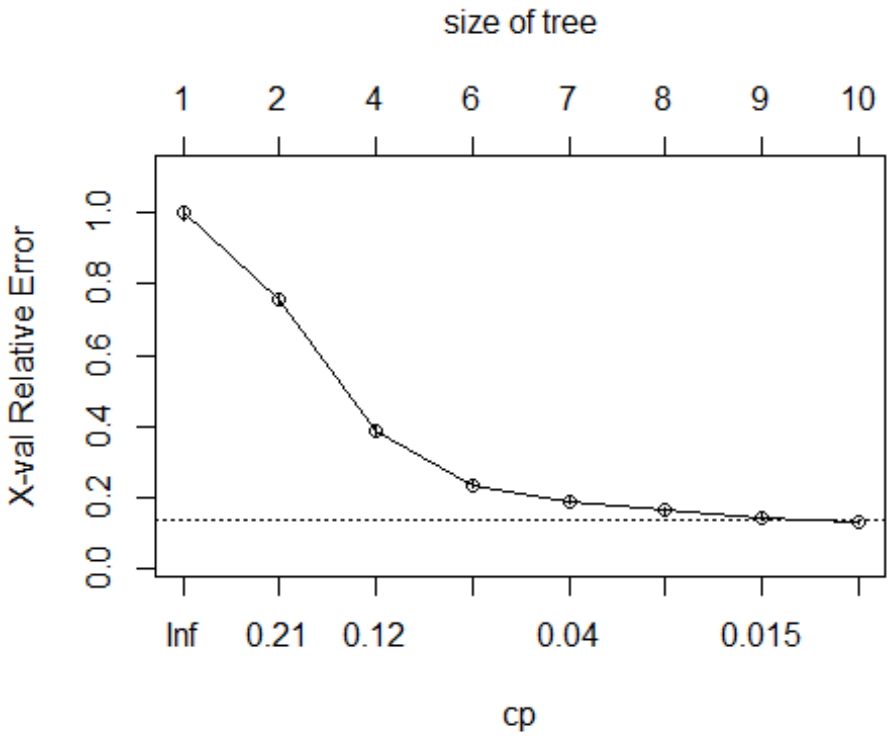
The idea here is to allow the decision tree to grow fully and observe the CP value. The complexity parameter (CP) is used to control the size of the decision tree and to select the optimal tree size. If the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue.

```
printcp(DecTreeModel)

##
## Classification tree:
## rpart(formula = left ~ ., data = train, method = "class")
##
## Variables actually used in tree construction:
## [1] average_montly_hours last_evaluation      number_project
## [4] satisfaction_level   time_spend_company
##
## Root node error: 2500/10499 = 0.23812
##
## n= 10499
##
```

```
##       CP nsplit rel error xerror      xstd
## 1 0.2424      0    1.0000 1.0000 0.0174572
## 2 0.1872      1    0.7576 0.7576 0.0157598
## 3 0.0744      3    0.3832 0.3832 0.0118023
## 4 0.0504      5    0.2344 0.2344 0.0094089
## 5 0.0316      6    0.1840 0.1884 0.0084841
## 6 0.0180      7    0.1524 0.1624 0.0079024
## 7 0.0128      8    0.1344 0.1416 0.0073980
## 8 0.0100      9    0.1216 0.1280 0.0070455
```

```
plotcp(DecTreeModel)
```



The printcp() and plotcp() functions provide the cross-validation error for each nsplit and can be used to prune the tree. The one with the least cross-validation error (xerror) is the optimal value of CP given by the printcp() funtion

Step 3: Predicting test based on trained model

```
test$left_Pred<-predict(DecTreeModel,newdata = test,type = "class")
```

Step 4: Evaluating model accuracy using confusion matrix

```
table(test$left,test$left_Pred)
```

```
##
##        0    1
##   0 3391   38
##   1   93  978
```

```
library(caret)
confusionMatrix(table(test$left,test$left_Pred))
```

```
## Confusion Matrix and Statistics
##
##
##        0    1
##   0 3391   38
##   1   93  978
##
##                Accuracy : 0.9709
##                  95% CI : (0.9655, 0.9756)
##     No Information Rate : 0.7742
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9183
##
##  Mcnemar's Test P-Value : 2.382e-06
##
##             Sensitivity : 0.9733
##             Specificity : 0.9626
##          Pos Pred Value : 0.9889
##          Neg Pred Value : 0.9132
##              Prevalence : 0.7742
##          Detection Rate : 0.7536
##    Detection Prevalence : 0.7620
##       Balanced Accuracy : 0.9680
##
##        'Positive' Class : 0
##
```

Tree pruning

I find the value of CP for which the cross validation error is minimum

```
min(DecTreeModel$cptable[,"xerror"])
```

```
## [1] 0.128
```

```
which.min(DecTreeModel$cptable[,"xerror"])
```

```
## 8
## 8
```

```
cpmin<-DecTreeModel$cptable[8,"CP"]
cpmin
```

```
## [1] 0.01
```

From my analysis I can say the decision tree is optimal and homogeneous and doesn't need further pruning.

PART THREE:

RANDOM FOREST

Random forest is the most prefered and mostly used algorithm in data science projects for best results. It is an ensemble of decision trees that builds and combines multiple decision trees to gives a more accurate prediction. Each decision tree model is weak when epmloyed on its own, but it becomes stable when put together.
It is random because it operates by choosing predictors randomly at the time of training the model.
It is called a forest because it takes the output of multiple decision trees to make a decision.

After loading my data unto Rstudio, my next step will be spliting data.

Step 1: Split data into train and test data

```
library(caTools)
set.seed(123)

split <- sample.split(hr, SplitRatio = 0.75)
#split

training_set = subset(hr, split == TRUE)
test_set = subset(hr, split == FALSE)
```

Step 2: Fitting Random Forest Classifier into the Training set

```
# install.packages('randomForest')
library(randomForest)
classifier <- randomForest(x = training_set[-7],
                           y = training_set$left,
                           ntree = 500)
#bestmtry<-tuneRF(training_set,training_set$left,stepFactor = 1.2,improve =
0.01,trace = T,plot = T)
```

I am running my classifier using a random forest function, the ntree is an arbitrary number of trees to help me determine later if my model still needs optimization or has stabilized.

Step 3: Predicting test set data using training set

```
y_pred = predict(classifier, newdata = test_set[-7])
#y_pred
```

Step 4: Model Evaluation by checking accuracy using a confusion matrix

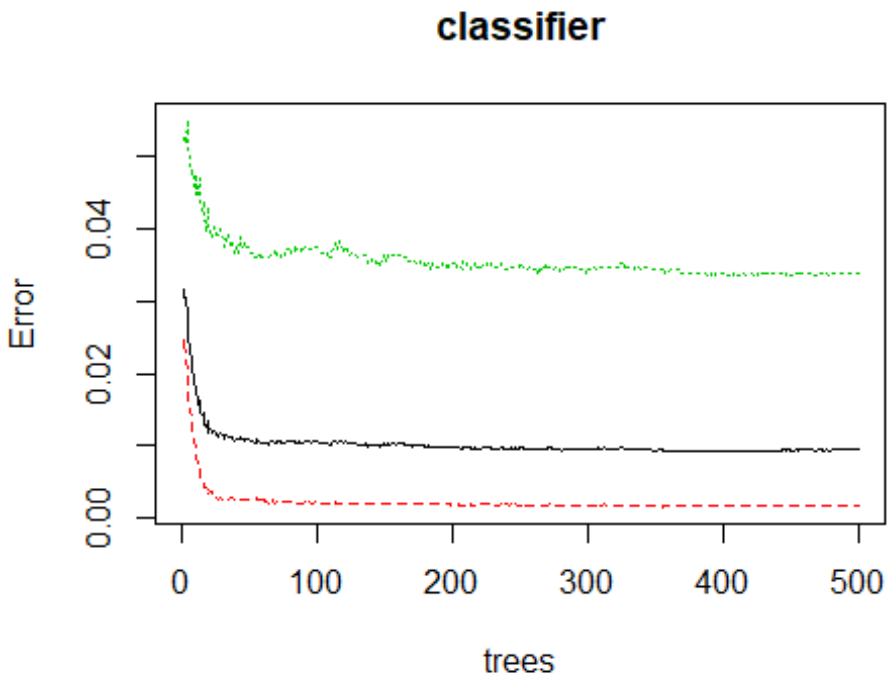
```
cm <- table(test_set$left,y_pred)
misClassError<-mean(test_set$left!=y_pred)
#print(paste('Accuracy=',1-misClassError))
library(caret)
confusionMatrix(cm)

## Confusion Matrix and Statistics
##
```

```
##    y_pred
##        0    1
##   0 3424    5
##   1   28 1043
##
##                Accuracy : 0.9927
##                  95% CI : (0.9897, 0.9949)
##     No Information Rate : 0.7671
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9796
##
##  Mcnemar's Test P-Value : 0.0001283
##
##             Sensitivity : 0.9919
##             Specificity : 0.9952
##          Pos Pred Value : 0.9985
##          Neg Pred Value : 0.9739
##              Prevalence : 0.7671
##          Detection Rate : 0.7609
##    Detection Prevalence : 0.7620
##       Balanced Accuracy : 0.9936
##
##        'Positive' Class : 0
##
```

The information I gather from the confusion matrix is; a high degree of accuracy and a general high score in sensitivity and specificity, which is indeed a good model. Doing a quick comparison with decision tree model you realise that there is an improvement with this Random Forest model.

```
plot(classifier)
```

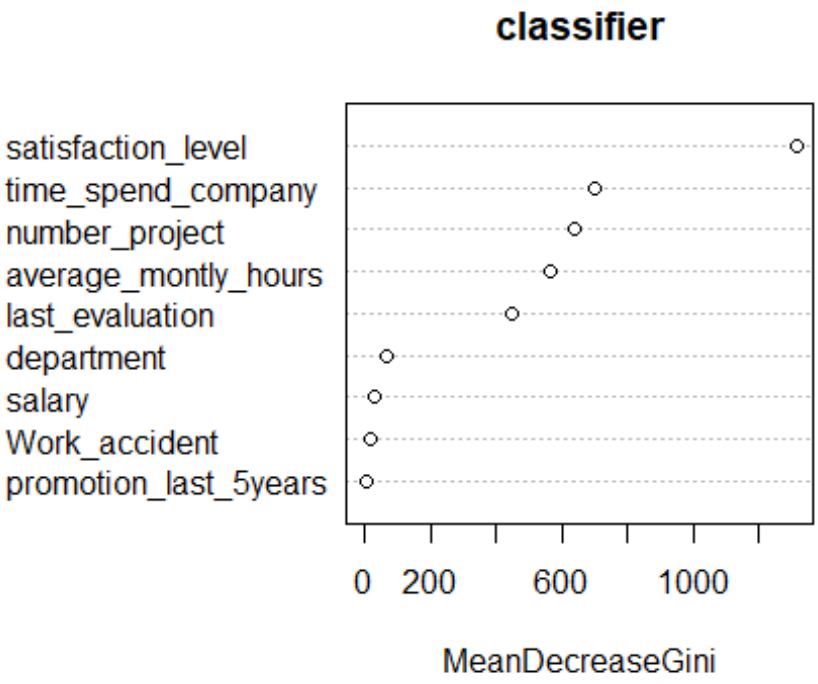


The above plot on the classifier illustrate the number of trees that best optimise the model. By growing the trees more and more you understand from the figure if my error decreases or not. If it were still decreasing I would probably go and change, ntree, from my classifier to accomadate this calculation. It is not decreasing further infact it decreased until at some point in 30 trees and there was no improvements by growing more trees.

```
importance(classifier)
```

```
##                         MeanDecreaseGini
## satisfaction_level           1311.316790
## last_evaluation               450.771887
## number_project                639.670532
## average_montly_hours          562.505808
## time_spend_company            699.796376
## Work_accident                  21.226730
## promotion_last_5years           3.825953
## department                     66.347186
## salary                         30.727622
```

```
varImpPlot(classifier)
```



CONCLUSION

The best thing about Random Forest, it has an inbuilt variable importance function, varImpPlot. In earlier models we used to check on p-value but in Random Forest, it is a diagnostic tool to explain the ***WHY*** in a problem statement. The most important variable in the analysis is displayed in the plot sequentially to explain the significance of each individual attribute.

By looking at the plot you understand why this happened. For instance, why is the organization is experiencing staff turnover; you realise that the satisfaction level plays a major role in this churning process. Time spend in the company by the individual, the number of projects handled by the staff member, and the average monthly hours put in by the employees comes after respectively. Salary comes a distant 7th in the list.