Code

# R Notebook

# CS7DS1 Data Analytics Assignment


## Guidong Xiang, 17301984


# Data Set 1–economic

## 1.Examination of the data

Load in the dataset and analysis, the aim of the assignment is to predict the Net Profit or Loss, our target column with which we can point our model to train on would be the “Net Profit / (Loss)” column.

I used excel to delete some unnecessary symbols such as “€” and made it easy to be processed by R.

```
library(readxl)
df <- read_excel("E:/GitHub/Data-Analytics/economic_1.xlsx")
str(df)
```

```
and 'data.frame':   1432 obs. of  39 variables:
 $ Reference Number                          : num  1 2 3 4 5 6 7 8 9 10 ...
 $ Vessel  ID                                : num  29 190 398 41 310 564 402 363 391 214 ...
 $ Year                                      : num  2008 2008 2008 2008 2008 ...
 $ Segment                                   : chr  "DTS" "TM" "FPO" "DTS" ...
 $ Size Category                             : num  1824 4070 10 1824 1824 ...
 $ Capacity (GT)                             : num  128 774.1 4.5 127.4 121 ...
 $ Length                                    : num  21.4 44.7 9.4 23 17.5 8 5.2 10.2 51.7 17.2 ...
 $ Capacity Index                            : num  4.9 6.7 1.5 4.8 4.8 1.2 0.1 2.5 7 3.7 ...
 $ Size Index                                : num  3.1 3.8 2.2 3.1 2.9 2.1 1.7 2.3 3.9 2.8 ...
 $ Total Number of vessels in Segment        : num  85 20 1034 85 85 ...
 $ Total Active Vessels in Segment           : num  76 20 818 76 76 818 818 818 20 57 ...
 $ Total Engine Power (kW) for Segment-Size: num  31631 39750 24012 31631 31631 ...
 $ Total Capacity (GT)  for Segment-Size     : num  11163 27208 2311 11163 11163 ...
 $ Average Length for Segment                : num  21.3 56.2 6.7 21.3 21.3 6.7 6.7 6.7 56.2 14.8 ...
 $ Total Jobs                                : num  5 14 2 3 5 2 2 2 14 2 ...
 $ FTE                                       : num  5 7.5 1 3 5 1.5 1.5 1 14 2 ...
 $ Fishing Income                            : num  249304 NA 14502 402828 517593 ...
 $ Non Fishing Income                        : num  NA 2925000 NA NA NA ...
 $ Total Income                              : num  249304 2925000 14502 402828 517593 ...
 $ Wages                                     : num  61449 NA NA 35763 74752 ...
 $ Energy Costs (Fuel)                       : num  59487 NA 3850 215922 120089 ...
 $ Repairs & Maintenance                     : num  92954 NA 300 65688 74606 ...
 $ Filters/Lube Oil                          : num  NA NA NA NA NA NA NA NA NA NA ...
 $ Provisions                                : num  6743 NA NA 23394 116730 ...
 $ Ice                                       : num  2800 NA NA 4515 NA ...
 $ Dues & Levies                             : num  NA NA NA NA 1057 ...
 $ Sundry Variable Costs                     : num  1540 47952 2800 2227 124742 ...
 $ Total Variable Costs                      : num  224973 47952 6950 347509 511977 ...
 $ Insurance                                 : num  13000 NA NA 19200 22256 ...
 $ Loan Interest                             : num  47259 699422 NA 36519 36503 ...
 $ Accountancy                               : num  2250 10500 NA 3288 8000 ...
 $ Legal Fees                                : num  3356 17025 NA 8839 NA ...
 $ Sundry fixed costs                        : num  17974 35343 500 11935 37906 ...
 $ Total Fixed Costs                         : num  83839 762290 500 79781 104665 ...
 $ TOTAL COSTS                               : num  308812 810242 7450 427290 616642 ...
 $ GROSS PROFIT                              : num  -59508 2114758 7052 -24462 -99049 ...
 $ Depreciation                              : num  38384 1669320 NA 66049 50547 ...
 $ Sundry receipts                           : num  NA 28375 NA 5000 72 ...
 $ Net Profit / (Loss)                       : num  -97892 473813 7052 -85511 -149523 ...
```

Hide

```
table(sapply(df,class))
```

```

character   numeric
        1        38
```

Hide

```
#convert Segment to factor
df$Segment <- as.factor(df$Segment)
#There will be error when column name has blank, so change the colunm name
names(df)[5]="SizeCategory"
names(df)[39]= "NetProfit"
#str(df)
```

required Libraries:

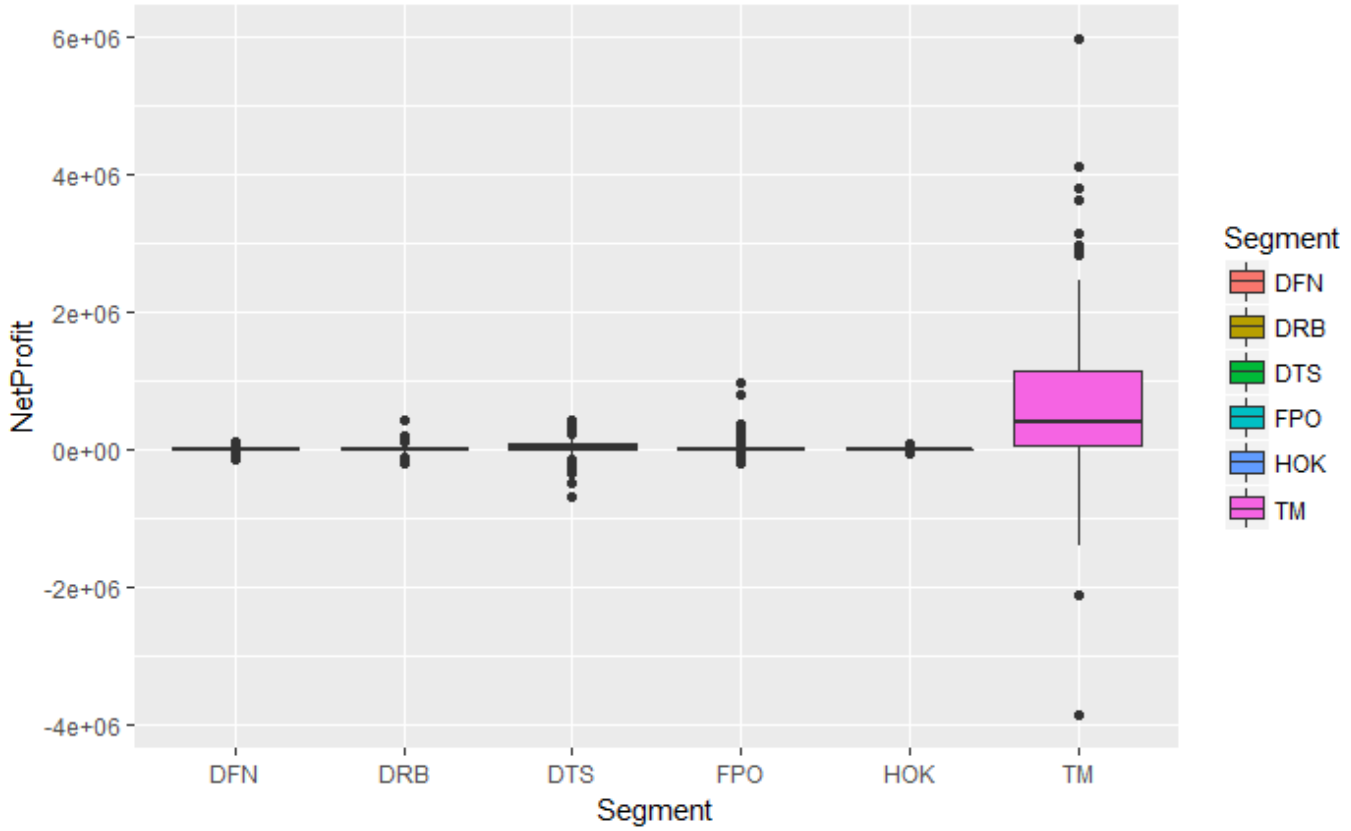
Hide

```
library(mice)
library(VIM)
library(lattice)
library(Hmisc)
library(ggplot2)
library(pls)
library(rpart.plot)
library(rattle)
```

## Detailed examination of the variable “Segment”

Hide

```
ggplot(data = df, aes(x = Segment, y = NetProfit, fill= Segment))+
  geom_boxplot()
```




Segment seems to be an very important variable.But its distribution is very uneven. Most of the Segments are TM, with a wide range of NetProfit, the highest profit and the highest loss both belong to TM. If we look at DTS and FPO, we can find that there seems to be more loss in DTS, while there are more netprofits in FPO.

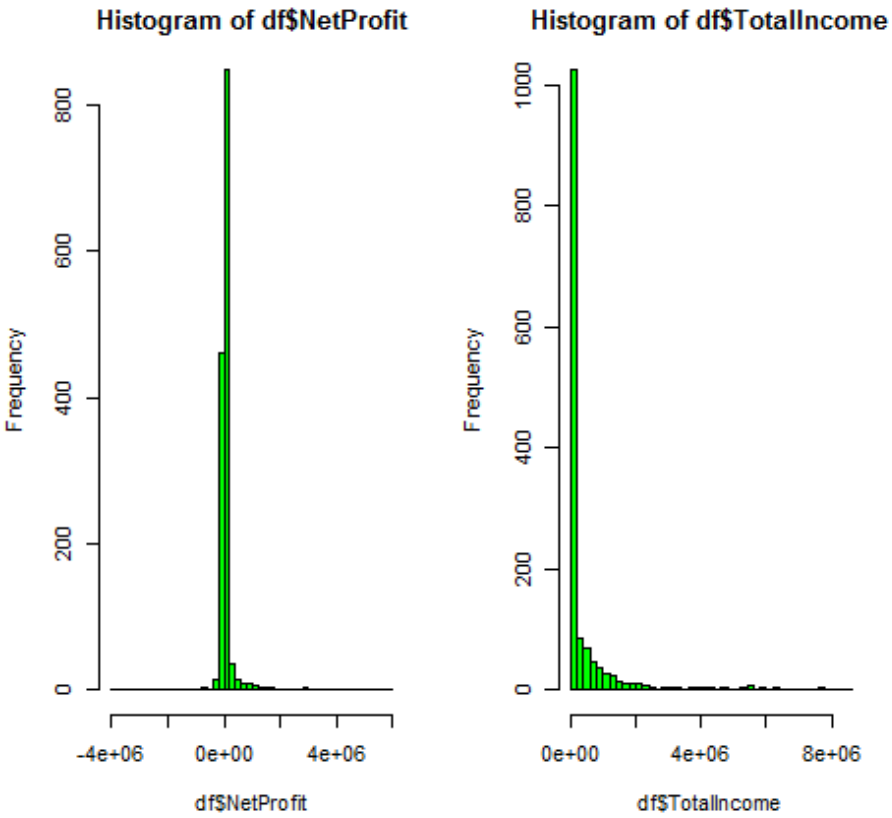
## Detailed examination of some other continuous variable

Hide

```
par(mfrow=c(1,3))
hist(df$NetProfit, breaks = 50,col = "green")
names(df)[19]="TotalIncome"
hist(df$TotalIncome , breaks = 50, col = "green")
```

Hide

```
names(df)[35]="TOTALCOSTS"
hist(df$TOTALCOSTS, breaks = 50,col = "green")
```

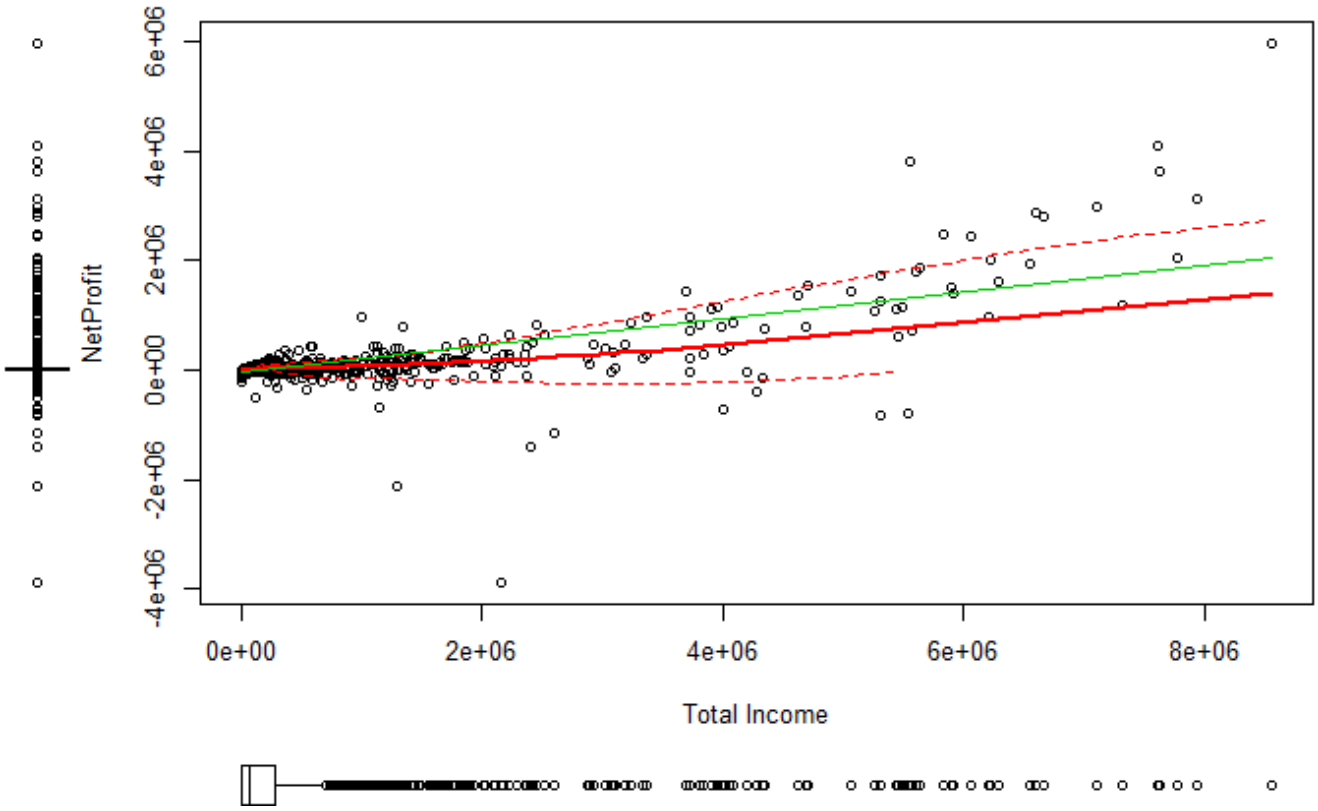



These three plots illustrate the dstribution of NetProfit, TotalIncome and TotalCost. They are concentrated in relatively small regions 。

I'm interested in the relationship between NetProfit and some numeric variables, but these can be tougher to visualize. So I made the following scatterplot for further analysis.

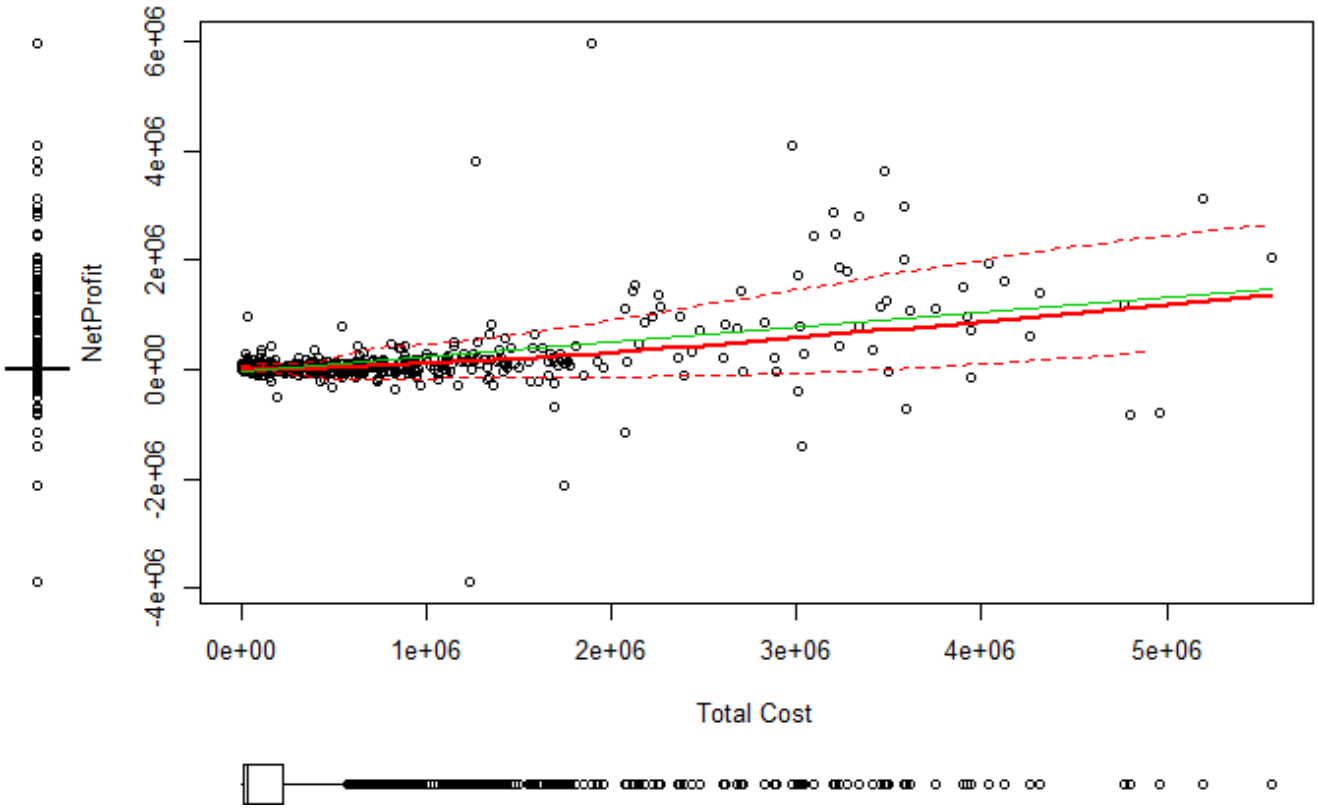
Hide

```
library(car)
scatterplot(NetProfit ~ TotalIncome, data=df,
            xlab="Total Income", ylab="NetProfit", grid=FALSE)
```




Hide

```
library(car)
scatterplot(NetProfit ~ TOTALCOSTS , data=df,
            xlab="Total Cost", ylab="NetProfit", grid=FALSE)
```




From the two plots above, we can find that NetProfit and Total Income are generally proportional, the higher Total Income means higher NetProfit, that makes sense. However, higher Total Cost doesn't always mean higher NetProfit, because too much cost may lead to loss. We could also have some loopy outliers as seen in

the plot, probably bad data but it's not going to have a huge influence.

## Handle missing data

We'll have to investigate the missing data firstly before we train any predictive models. Let's look at the variables individually and asess the number of entries that are missing values.

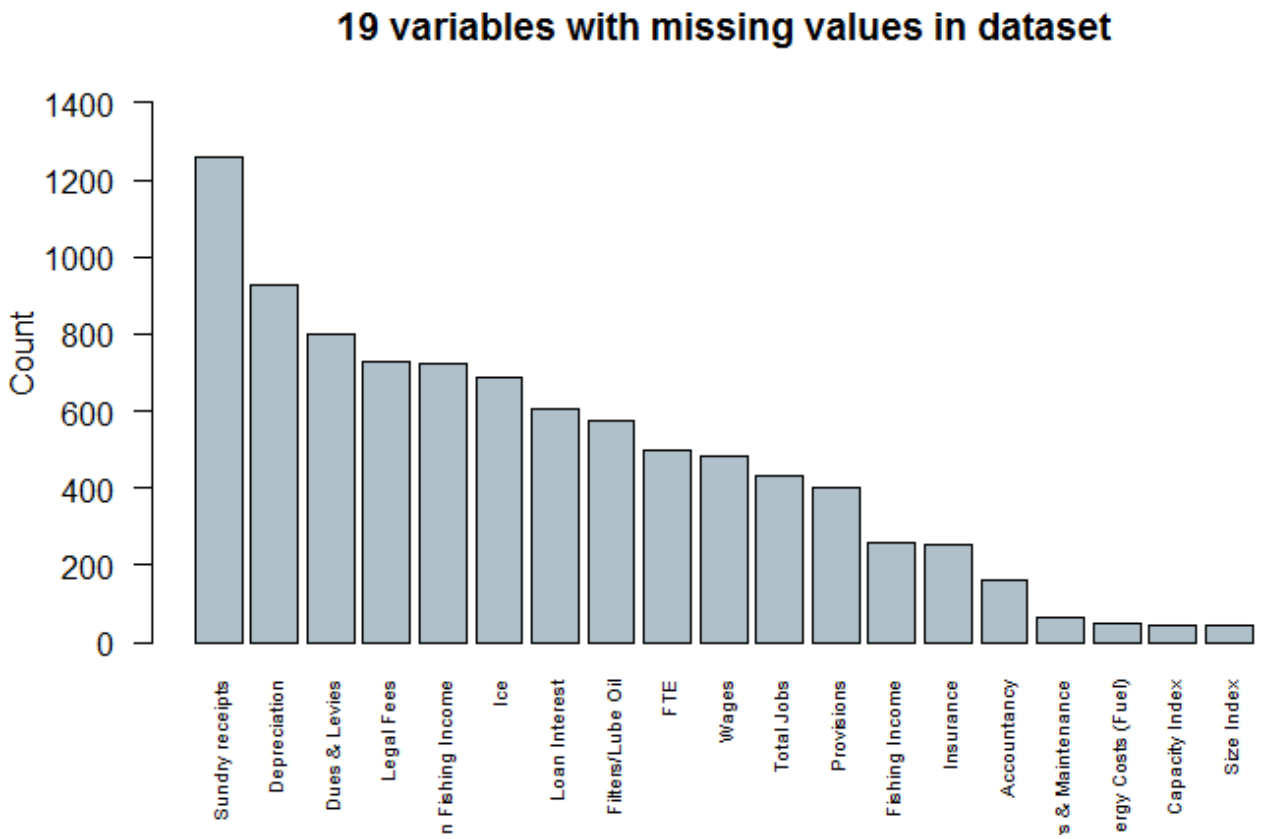
Hide

```
apply(is.na(df), 2, sum)
```

```
                      Reference Number                                Vessel  ID 
                                     0                                         0 
                                  Year                                   Segment 
                                     0                                         0 
                          SizeCategory                            Capacity (GT) 
                                     0                                         0 
                                Length                            Capacity Index 
                                     0                                        43 
                            Size Index              Total Number of vessels in Segment 
                                    43                                         0 
      Total Active Vessels in Segment          Total Engine Power (kW) for Segment-Size 
                                     0                                         0 
  Total Capacity (GT)  for Segment-Size                    Average Length for Segment 
                                     0                                         0 
                            Total Jobs                                       FTE 
                                   432                                       499 
                         Fishing Income                       Non Fishing Income 
                                   257                                       724 
                           TotalIncome                                    Wages 
                                     0                                       481 
                  Energy Costs (Fuel)                     Repairs & Maintenance 
                                    47                                        65 
                     Filters/Lube Oil                               Provisions 
                                   576                                       399 
                                  Ice                              Dues & Levies 
                                   686                                       797 
                  Sundry Variable Costs                       Total Variable Costs 
                                     0                                         0 
                            Insurance                                Loan Interest 
                                   252                                       605 
                          Accountancy                                 Legal Fees 
                                   160                                       729 
                    Sundry fixed costs                       Total Fixed Costs 
                                     0                                         0 
                           TOTALCOSTS                             GROSS PROFIT 
                                     0                                         0 
                         Depreciation                           Sundry receipts 
                                   927                                      1259 
                            NetProfit                                          
                                     0                                          
```

Hide

```
options(repr.plot.width=6, repr.plot.height=5)
cMiss = function(x){sum(is.na(x))}
CM <- sort(apply(df,2,cMiss),decreasing=T);
barplot(CM[CM!=0],
        las=2,
        cex.names=0.6,
        ylab="Count",
        ylim=c(0,1500),
        horiz=F,
        col="#AFC0CB",
        main=paste(toString(sum(CM!=0)),
                   "variables with missing values in dataset"))
```




Let's take a closer look a the definition of these variables. It might give us a clue as to why they are missing. Are they Missing Completely at Random (MCAR) or Missing at Random (MAR) or something else entirely?

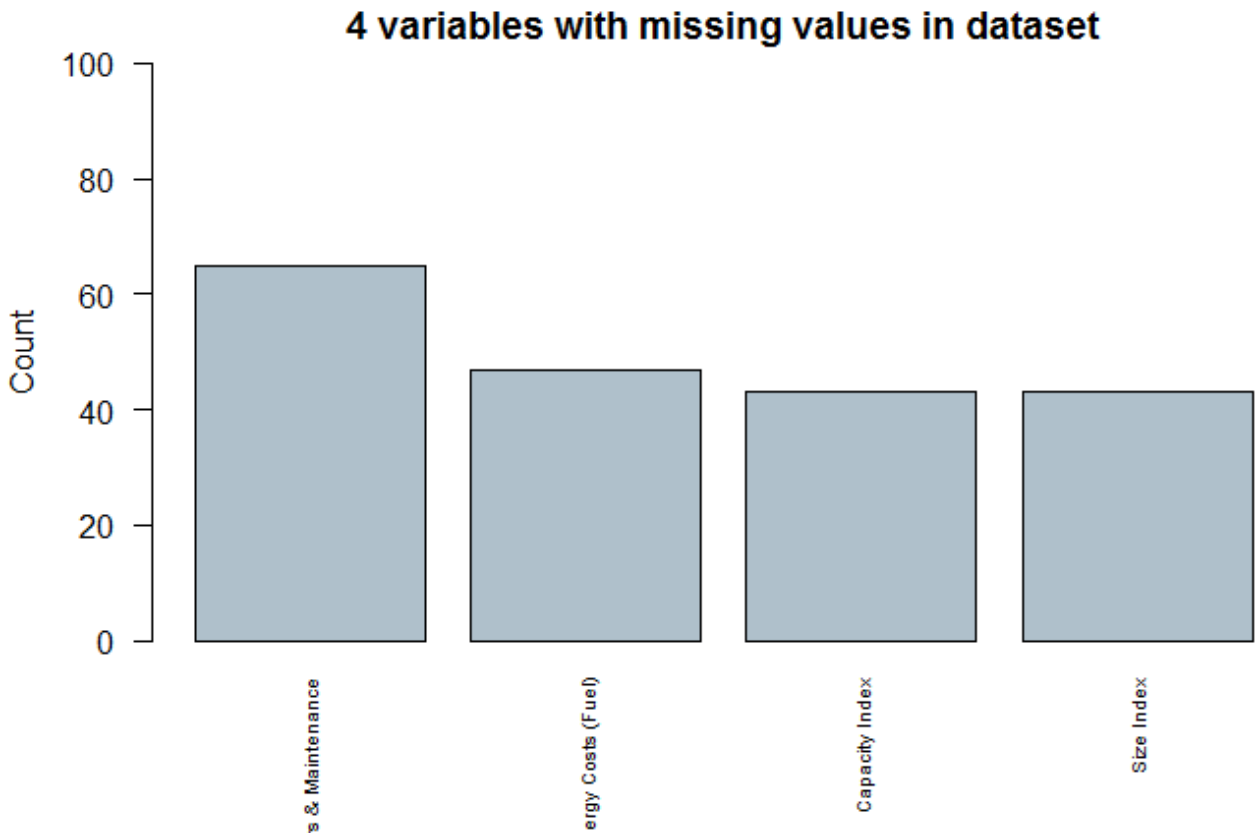
We can find that some of these missing categorical variables aren't missing at all, they simply represent the absence of Depreciation or Sundry receipts. We can impute the missing value with zero.

Hide

```
var_NAtoZero<-c("Depreciation","Sundry receipts",
                  "Total Jobs","FTE","Fishing Income",
                 "Non Fishing Income","Wages",
                   "Filters/Lube Oil","Provisions","Ice","Dues & Levies",
                   "Insurance","Loan Interest","Accountancy","Legal Fees")
zero<-function(data,var){
  levels(data[,var]) <- c(levels(data[,var]), 0)
  data[,var][is.na(data[,var])] <- 0
  return(data[,var])
}
data_clean<-df
for (i in 1:length(var_NAtoZero)){
  data_clean[,var_NAtoZero[i]]<-zero(df,var_NAtoZero[i])
}
```

Hide

```
options(repr.plot.width=6, repr.plot.height=5)
cMiss = function(x){sum(is.na(x))}
CM <- sort(apply(data_clean,2,cMiss),decreasing=T);
barplot(CM[CM!=0],
        las=2,
        cex.names=0.6,
        ylab="Count",
        ylim=c(0,100),
        horiz=F,
        col="#AFC0CB",
        main=paste(toString(sum(CM!=0)),
                   "variables with missing values in dataset"))
```




Hide

```
#drop rows with na
data <- na.omit(data_clean)
```

Hide

```
sum(is.na(data))
```

```
[1] 0
```

Now we can see that there is no missing data at all, by either imputing missing data with zero or delete a small number of rows with missing data.

# 2.Single tree model (Regression Trees)

## Model Prepping

Hide

```
#drop some columns
data <- data[-c(1:2,35:38)]
# Creating train & test sets
set.seed(100)
sub<-sample(1:nrow(data),nrow(data)*0.7)
length(sub)
```

```
[1] 922
```

Hide

```
data_train<-data[sub,]
data_test<-data[-sub,]
dim(data_train)
```

```
[1] 922  33
```

Hide

```
dim(data_test)
```

```
[1] 396  33
```

## R, Regression Trees, function rpart(), method “anova”

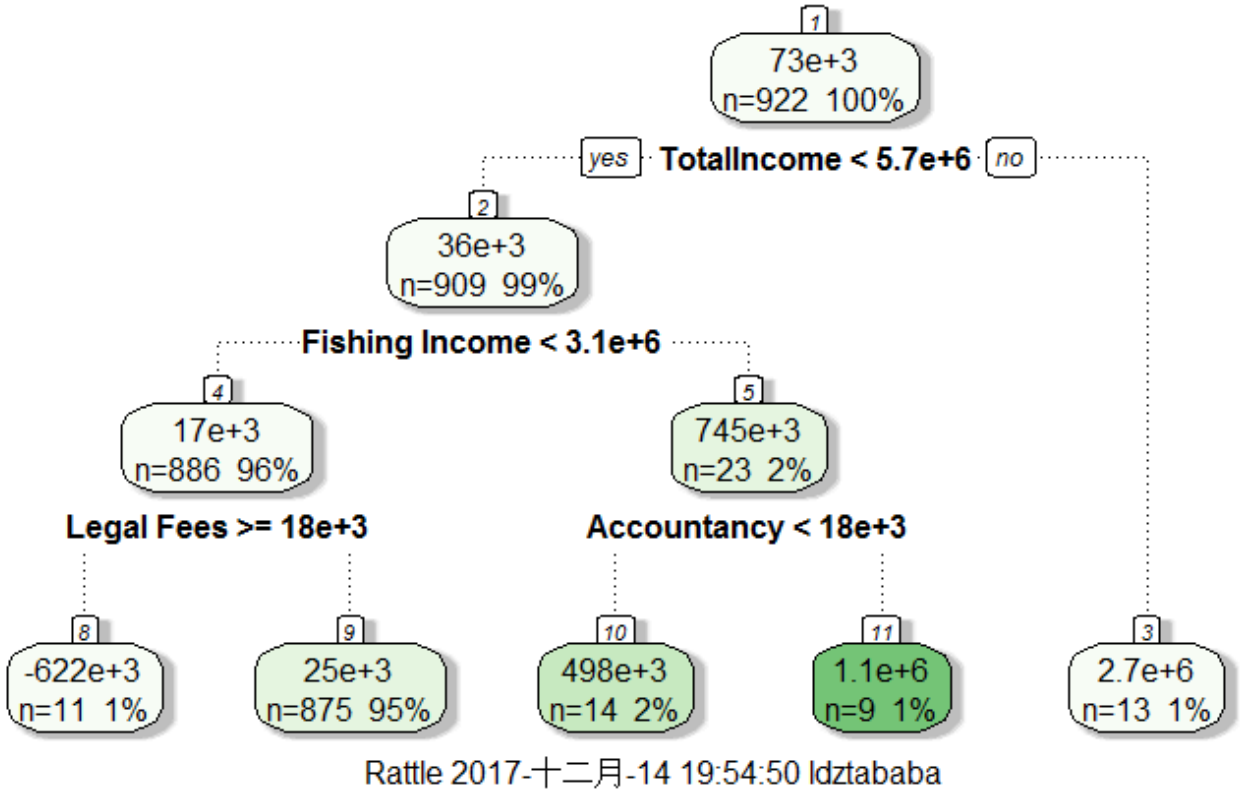
Hide

```
library(rpart)
model <- rpart(NetProfit ~., data = data_train, method = "anova")
predict <- predict(model, data_test)
#Normalization
predict <- scale(predict)
data_test$NetProfit <- scale(data_test$NetProfit)
# RMSE
rmse(predict, data_test$NetProfit)
```

```
[1] 0.8149244
```

Hide

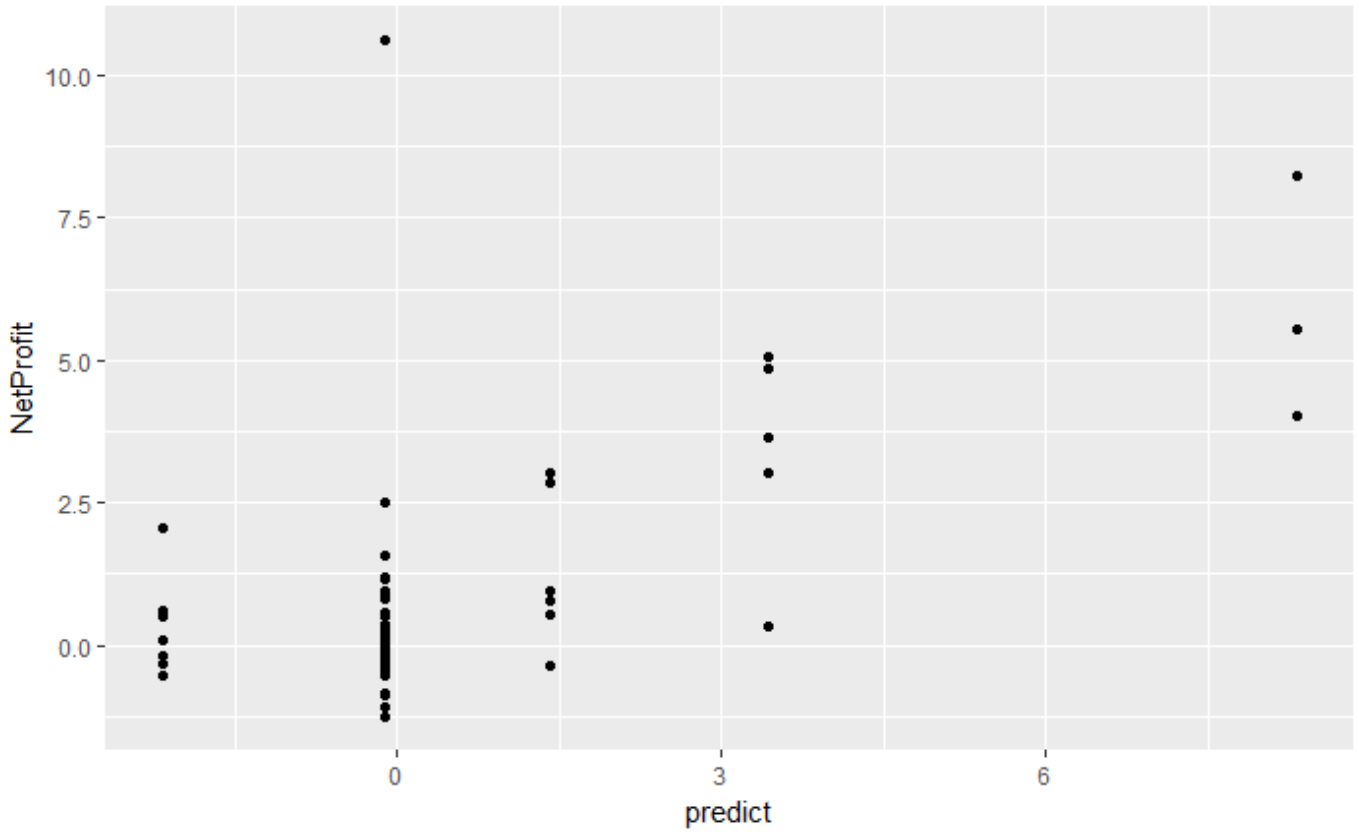
```
library(rpart.plot)
library(rattle)
fancyRpartPlot(model)
```




The tree model seems to perform not very good. Let's plot the result. From the plot below, we can see that the predictions are isolated and the rmse is large.

Hide

```
result <- data.frame(a=predict, b=data_test$NetProfit)
ggplot(result,aes(x=a, y=b))+geom_point()+
  xlab("predict") + ylab("NetProfit")
```




Hide

```
NA
```

# 3.Ensemble Technique 1 – Gradient Boosting Regression Tree

Hide

```
model <- gbm(NetProfit ~., data = data_train, distribution = "gaussian",
             shrinkage = 0.05,
             interaction.depth = 5,
             bag.fraction = 0.66,
             n.minobsinnode = 1,
             cv.folds = 100,
             keep.data = F,
             verbose = F,
             n.trees = 300)
gbmpredict <- predict(model, data_test, n.trees = 300)
gbmpredict <- scale(gbmpredict)
data_test$NetProfit <- scale(data_test$NetProfit)
rmse(gbmpredict, data_test$NetProfit)
```

```
[1] 0.4823278
```

***Some parameters:***

**formula：** NetProfit ~.

**distribution：** “gaussian” (squared error) is used for regression

**shrinkage：** A shrinkage parameter applied to each tree in the expansion, also known as the learning rate, we set it to 0.05

**interaction.depth:** The maximum depth of variable interactions

**bag.fraction:** the fraction of the training set observations randomly selected to propose the next tree in the expansion.

**n.minobsinnode:** Minimum number of observations in the trees terminal nodes.

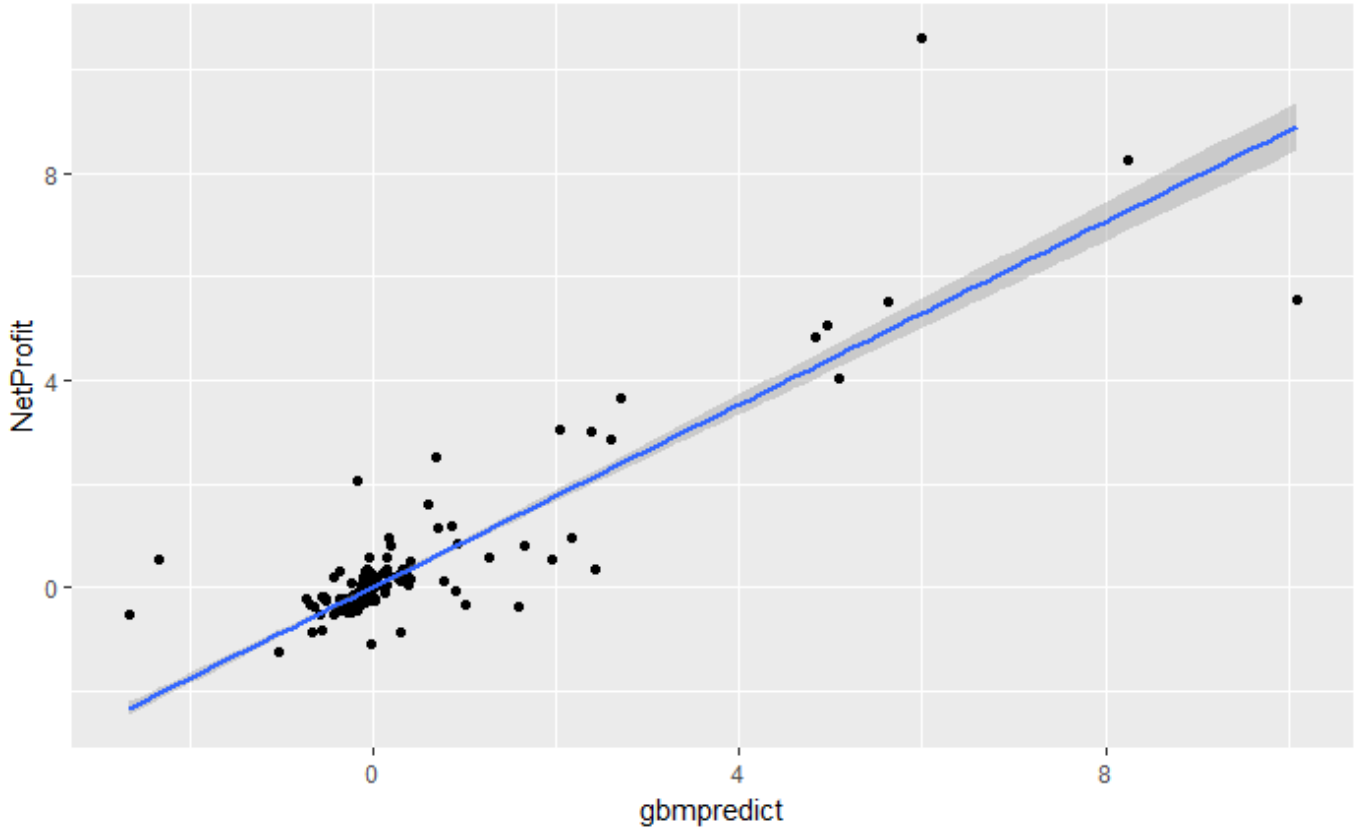
**cv.folds:** Number of cross-validation folds to perform.

**n.trees:** The total number of trees to fit

## Output and evaluation of the model

Hide

```
result <- data.frame(a=gbmpredict, b=data_test$NetProfit)
ggplot(result,aes(x=a, y=b))+geom_point()+
  xlab("gbmpredict") + ylab("NetProfit") +
  geom_smooth(method = lm)
```




The GBM model performed better than the single tree model, the rmse is 0.48, from the linear relationship shown in the plot above, we can say that this gbm model is ok.

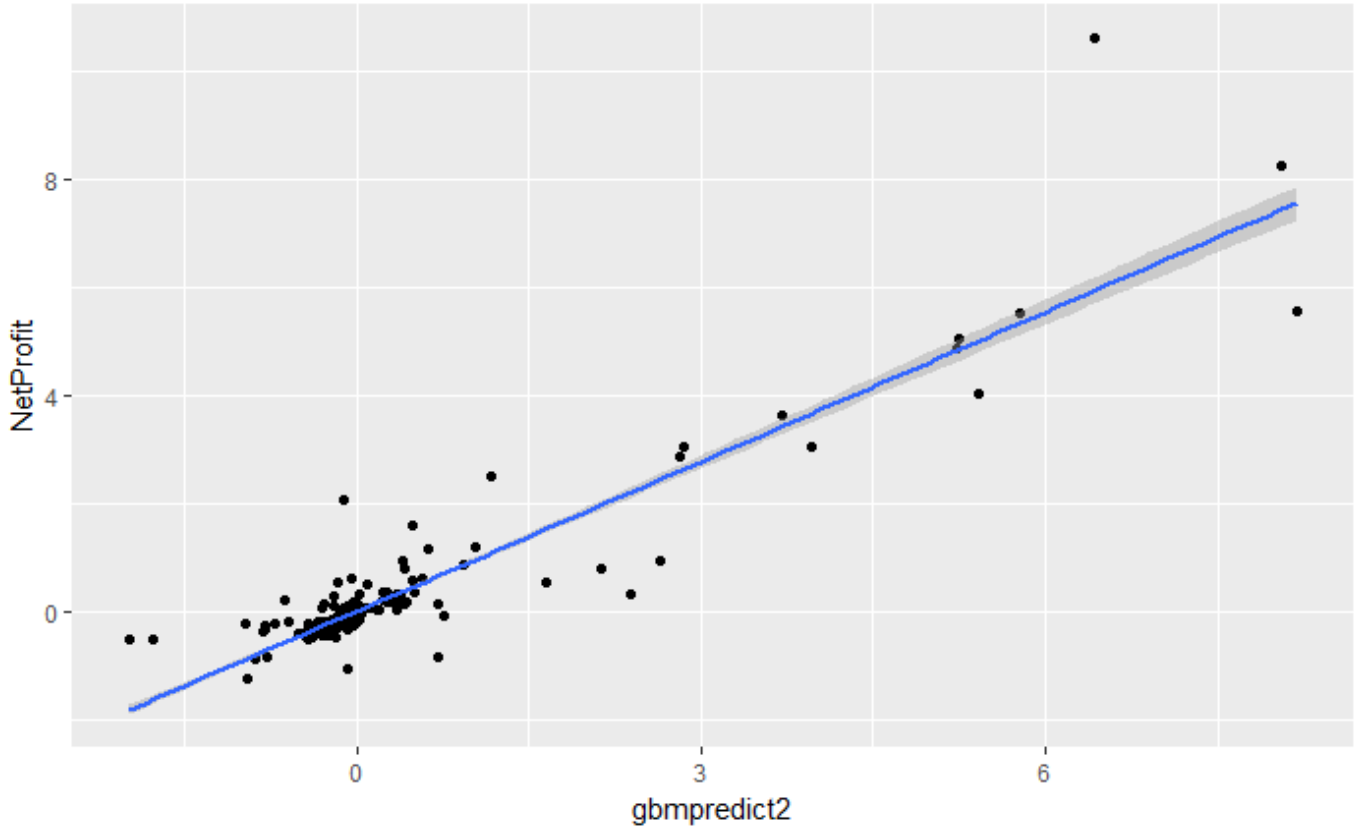
Hide

```
#Change some parameters and look at the output
model <- gbm(NetProfit ~., data = data_train, distribution = "gaussian",
             shrinkage = 0.1,
             interaction.depth = 5,
             bag.fraction = 0.66,
             n.minobsinnode = 1,
             cv.folds = 100,
             keep.data = F,
             verbose = F,
             n.trees = 500)
gbmpredict2 <- predict(model, data_test, n.trees = 500)
gbmpredict2 <- scale(gbmpredict2)
data_test$NetProfit <- scale(data_test$NetProfit)
rmse(gbmpredict2, data_test$NetProfit)
```

```
[1] 0.3897787
```

Hide

```
result <- data.frame(a=gbmpredict2, b=data_test$NetProfit)
ggplot(result,aes(x=a, y=b))+geom_point()+
  xlab("gbmpredict2") + ylab("NetProfit") +
  geom_smooth(method = lm)
```




I changed the shrinkage to 0.1 and the number of trees to 500, the result which I got had smaller rmse, which probably means the higher learning rate and the increase of the number of trees could build a better model.

# 4.Ensemble Technique 2 – Bagging

Hide

```
library(ipred)
model2 <- bagging(NetProfit ~., data = data_train, nbagg = 20,method = "standard")
predict2 <- predict(model2, data_test)
predict2 <- scale(predict2)
data_test$NetProfit <- scale(data_test$NetProfit)
rmse(predict2, data_test$NetProfit)
```

```
[1] 0.602264
```

***Some parameters:***

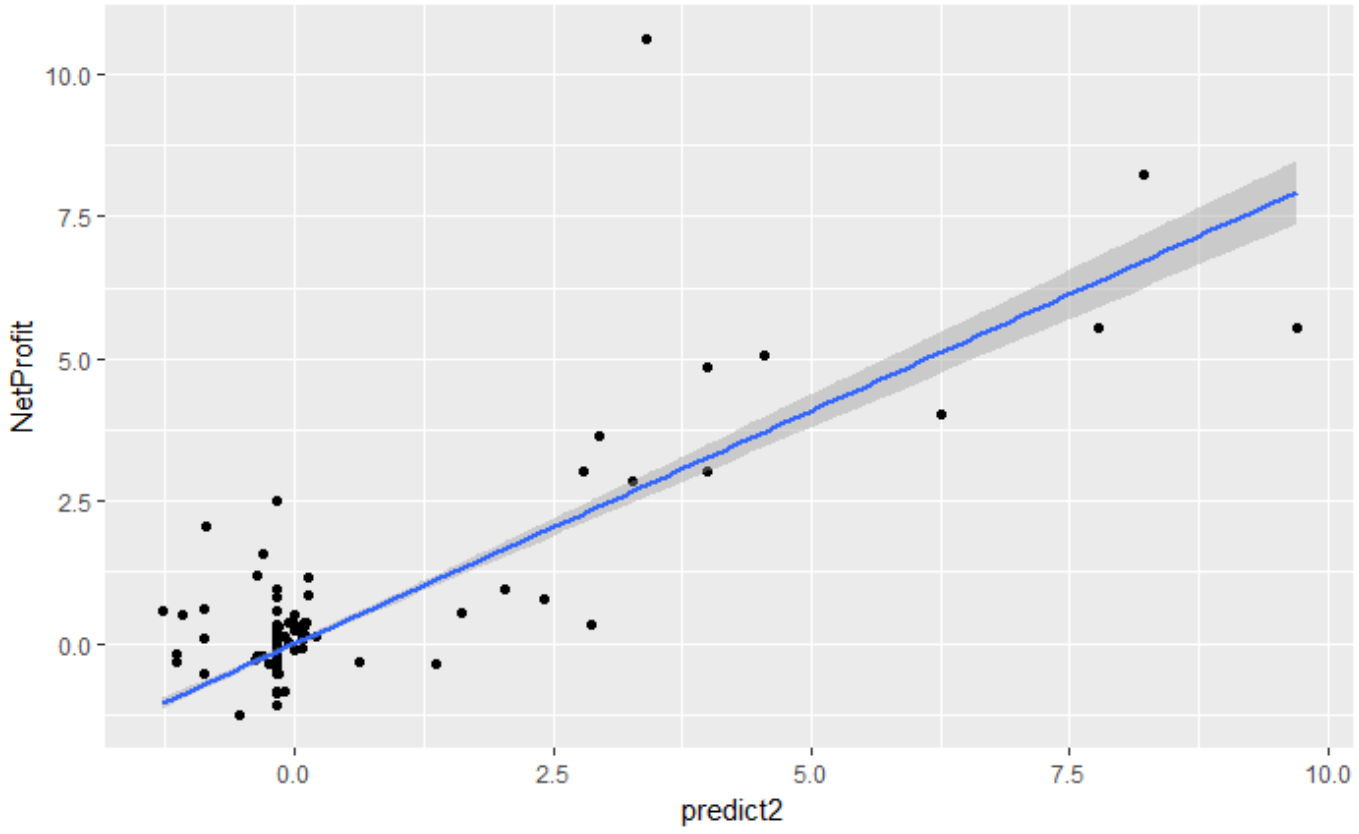
**formula：** NetProfit ~.

**nbagg：** The number of samples and hence the number of trees to include.

**method：** “Standard” for Bagging and “double” for Double-Bagging

Hide

```
result <- data.frame(a=predict2, b=data_test$NetProfit)
ggplot(result,aes(x=a, y=b))+geom_point()+
  xlab(" predict2") + ylab("NetProfit") +
  geom_smooth(method = lm)
```



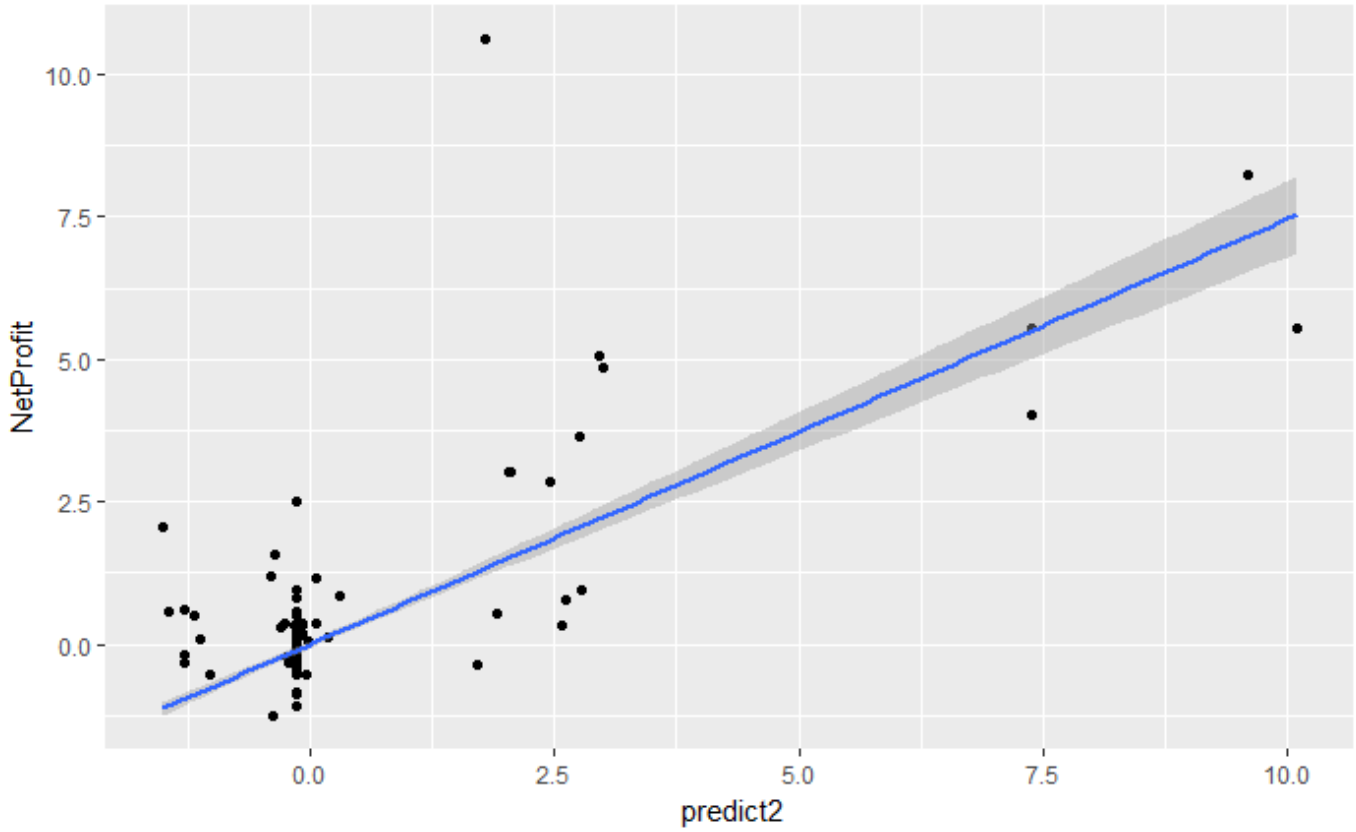

Hide

```
library(ipred)
model2 <- bagging(NetProfit ~., data = data_train, nbagg = 15,method = "standard")
predict2 <- predict(model2, data_test)
predict2 <- scale(predict2)
data_test$NetProfit <- scale(data_test$NetProfit)
rmse(predict2, data_test$NetProfit)
```

```
[1] 0.7115115
```

Hide

```
result <- data.frame(a=predict2, b=data_test$NetProfit)
ggplot(result,aes(x=a, y=b))+geom_point()+
  xlab(" predict2") + ylab("NetProfit") +
  geom_smooth(method = lm)
```




For the given economic data set, the bagging method seems to performe not very good. And there is large error when choosing different parameters. When I set the nbagg to 15, the rmse is 0.71 while it is 0.6 when the nbagg is 20.

Bagging is a simple and very powerful ensemble method. Bagging is a general procedure that can be used to reduce the variance for those algorithm that have high variance. An algorithm that has high variance are decision trees, like classification and regression trees (CART). From the result of this experiment, we could prove that bagging performs better than the single tree model, but not as good as the Gradient Boosting Regression Tree.

We have a dataset of 1318 instances and we are using the CART algorithm. Bagging of the CART algorithm would work as follows.

1.Create many (e.g. 100) random sub-samples of our dataset with replacement.

2.Train a CART model on each sample.

3.Given a new dataset, calculate the average prediction from each model.

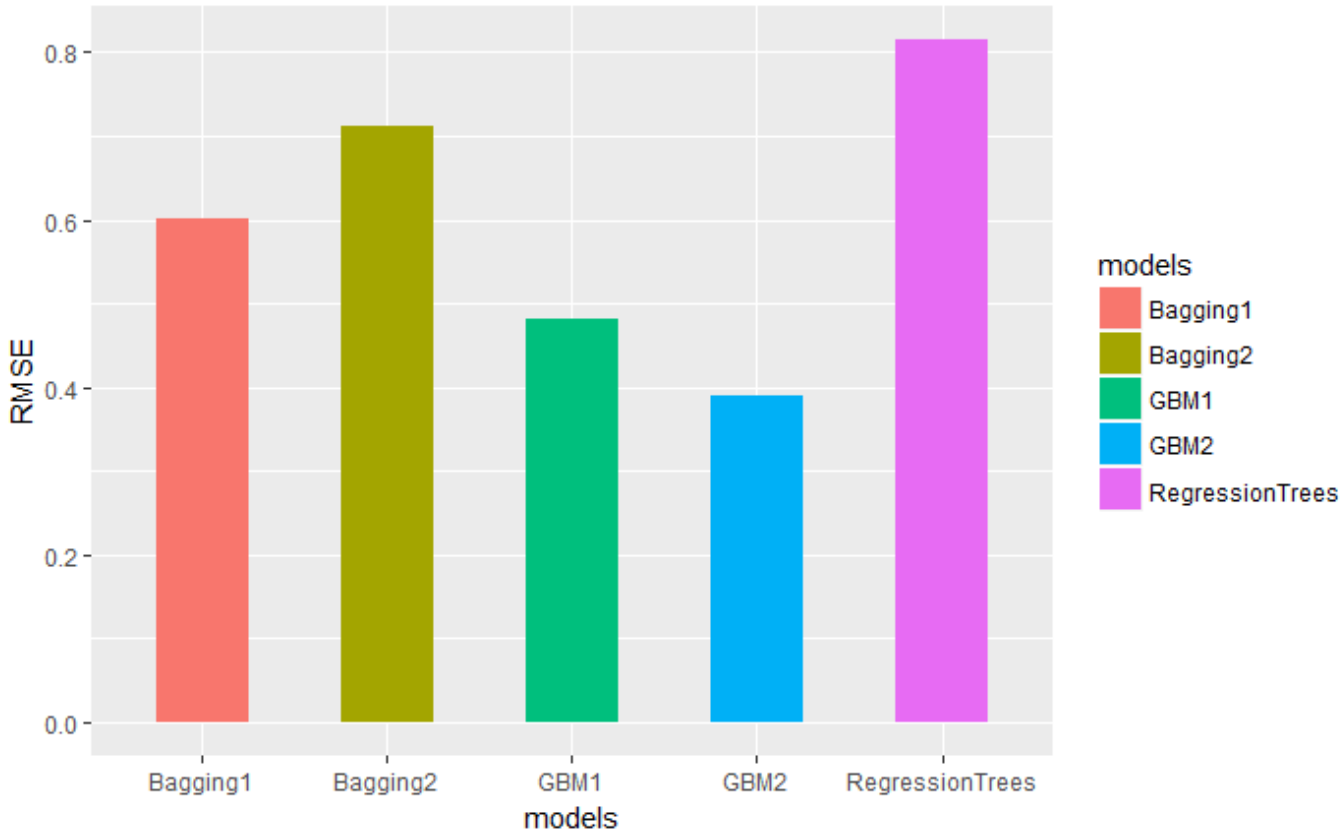
The only parameters when bagging decision trees is the number of samples and hence the number of trees to include. This can be chosen by increasing the number of trees on run after run until the accuracy begins to stop showing improvement. I have tried several nbagg and found that I could get a good result when it is set to 20.

# 5.Comparison of the above models

An ensemble method is a technique that combines the predictions from multiple machine learning algorithms together to make more accurate predictions than any individual model. And what I did for the economic data proved this. The ensemble method bagging and gbm I used performed better than the single tree model. The best rmse I get by gbm was 0.38, while the rmse for the single tree model was 0.81, that's a huge improvement by ensemle method.

Hide

```
Comparison <- data.frame(models=c("RegressionTrees","GBM1","GBM2","Bagging1","Bagging2"),RMSE=c(0.8149
244,0.4823278,0.3897787,0.602264,0.7115115))
ggplot(data = Comparison, aes(x = models, y = RMSE, fill= models))+
  geom_bar(stat = "identity", width = 0.5)
```




we'll notice the deficiencies of the single model When we look at the actual tree. The tree just have five terminal-nodes. That's going to be a major problem when doing any further prediction. I think we can safely say this is not something worth using for new predictions.

The ensemble methods had less error than the regression tree. Bagging, or bootstrap aggregating, reduces the variance found in a single decision tree model by making multiple predictions for each observation and selecting the most commonly occurring response (or class in our case). Theoretically this should reduce the over-fitting found in a basic decision tree model. GBM works by creating an initial classification tree and upon iteration, weighting the mis-classified observations as more significant before creating subsequent trees. The goal of this process is to reduce error on the poorly classified observations.

I expected that the basic tree models probably wouldn't perform that well. I had no idea that some of the ensemble methods like bagging would also perform so poorly. The clear winner from the data above is the GBM model.

# Data Set 2–Employeeattrition

# 1.Examination of the data

Load in the dataset and analysis, the aim of the assignment is to identify the important factors causing the attrition of employees and predict who is going to leave the organisation , our target column with which we can point our model to train on would be the “Attrition” column.

```
library(readxl)
data <- read_excel("E:/GitHub/Data-Analytics/Employeeattrition.xlsx")
```

Show the structure of the data and get the general information

```
str(data)
```

```
and 'data.frame':   1470 obs. of  35 variables:
 $ Age                     : num  41 49 37 33 27 32 59 30 38 36 ...
 $ Attrition               : chr  "Yes" "No" "Yes" "No" ...
 $ BusinessTravel          : chr  "Travel_Rarely" "Travel_Frequently" "Travel_Rarely" "Travel_Frequent
ly" ...
 $ DailyRate               : num  1102 279 1373 1392 591 ...
 $ Department              : chr  "Sales" "Research & Development" "Research & Development" "Research
& Development" ...
 $ DistanceFromHome        : num  1 8 2 3 2 2 3 24 23 27 ...
 $ Education               : num  2 1 2 4 1 2 3 1 3 3 ...
 $ EducationField          : chr  "Life Sciences" "Life Sciences" "Other" "Life Sciences" ...
 $ EmployeeCount           : num  1 1 1 1 1 1 1 1 1 1 ...
 $ EmployeeNumber          : num  1 2 4 5 7 8 10 11 12 13 ...
 $ EnvironmentSatisfaction : num  2 3 4 4 1 4 3 4 4 3 ...
 $ Gender                  : chr  "Female" "Male" "Male" "Female" ...
 $ HourlyRate              : num  94 61 92 56 40 79 81 67 44 94 ...
 $ JobInvolvement          : num  3 2 2 3 3 3 4 3 2 3 ...
 $ JobLevel                : num  2 2 1 1 1 1 1 1 3 2 ...
 $ JobRole                 : chr  "Sales Executive" "Research Scientist" "Laboratory Technician" "Rese
arch Scientist" ...
 $ JobSatisfaction         : num  4 2 3 3 2 4 1 3 3 3 ...
 $ MaritalStatus           : chr  "Single" "Married" "Single" "Married" ...
 $ MonthlyIncome           : num  5993 5130 2090 2909 3468 ...
 $ MonthlyRate             : num  19479 24907 2396 23159 16632 ...
 $ NumCompaniesWorked      : num  8 1 6 1 9 0 4 1 0 6 ...
 $ Over18                  : chr  "Y" "Y" "Y" "Y" ...
 $ OverTime                : chr  "Yes" "No" "Yes" "Yes" ...
 $ PercentSalaryHike       : num  11 23 15 11 12 13 20 22 21 13 ...
 $ PerformanceRating       : num  3 4 3 3 3 3 4 4 4 3 ...
 $ RelationshipSatisfaction: num  1 4 2 3 4 3 1 2 2 2 ...
 $ StandardHours           : num  80 80 80 80 80 80 80 80 80 80 ...
 $ StockOptionLevel        : num  0 1 0 0 1 0 3 1 0 2 ...
 $ TotalWorkingYears       : num  8 10 7 8 6 8 12 1 10 17 ...
 $ TrainingTimesLastYear   : num  0 3 3 3 3 2 3 2 2 3 ...
 $ WorkLifeBalance         : num  1 3 3 3 3 2 2 3 3 2 ...
 $ YearsAtCompany          : num  6 10 0 8 2 7 1 1 9 7 ...
 $ YearsInCurrentRole      : num  4 7 0 7 2 7 0 0 7 7 ...
 $ YearsSinceLastPromotion : num  0 1 0 3 2 3 0 0 1 7 ...
 $ YearsWithCurrManager    : num  5 7 0 0 2 6 0 0 8 7 ...
```

**Data quality checks**

Check if ther is any null values.

Hide

```
sum(is.na(data))
```

```
[1] 0
```

required Libraries:

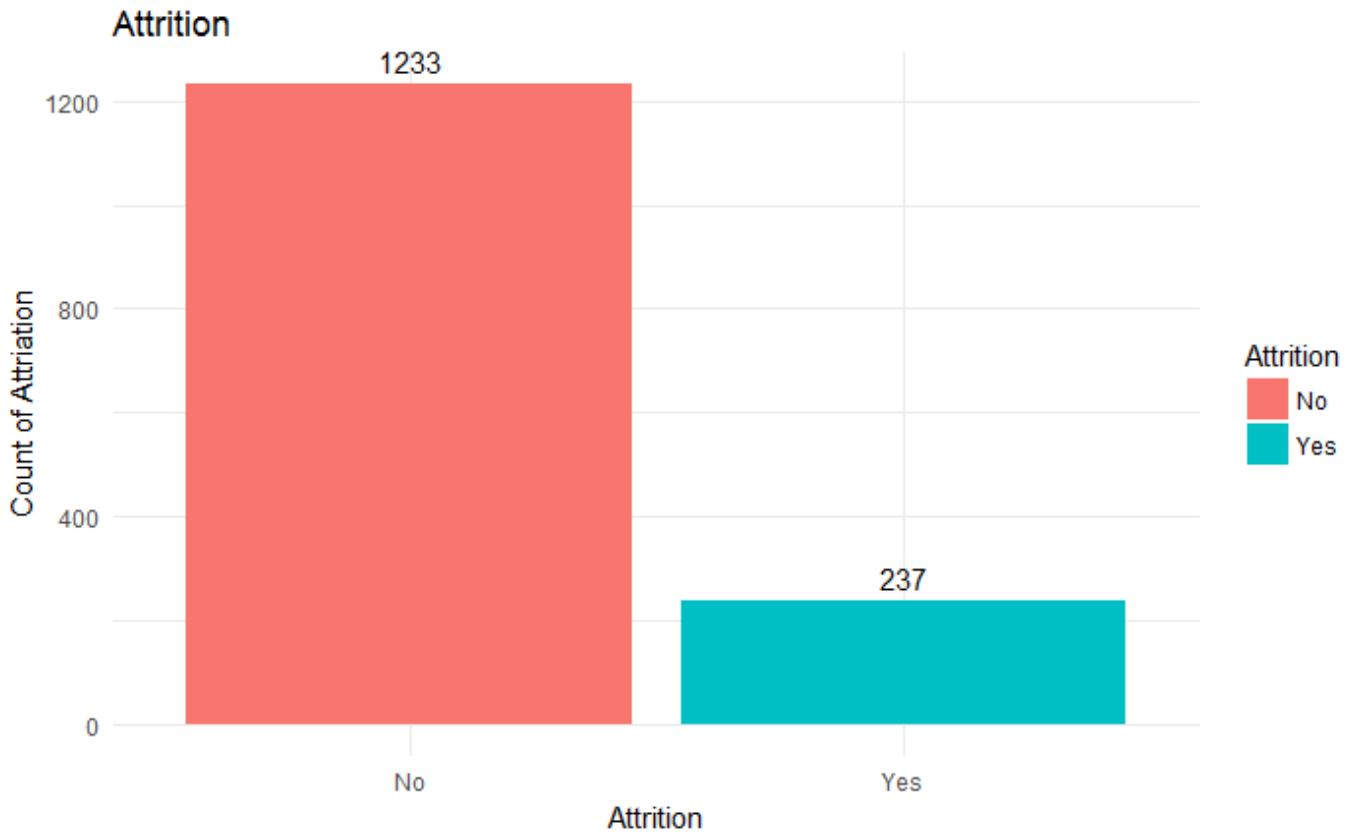
Hide

```
library(ggplot2)
library(magrittr)
library(cluster)
library(dplyr)
library(ggplot2)
library(magrittr)
library(rpart)
library(maptree)
```

Let us firstly start by taking a look at the attrition percentage of the data.

Hide

```
data %>%
        group_by(Attrition) %>%
        tally() %>%
        ggplot(aes(x = Attrition, y = n,fill=Attrition)) +
        geom_bar(stat = "identity") +
        theme_minimal()+
        labs(x="Attrition", y="Count of Attriation")+
        ggtitle("Attrition")+
        geom_text(aes(label = n), vjust = -0.5, position = position_dodge(0.9))
```

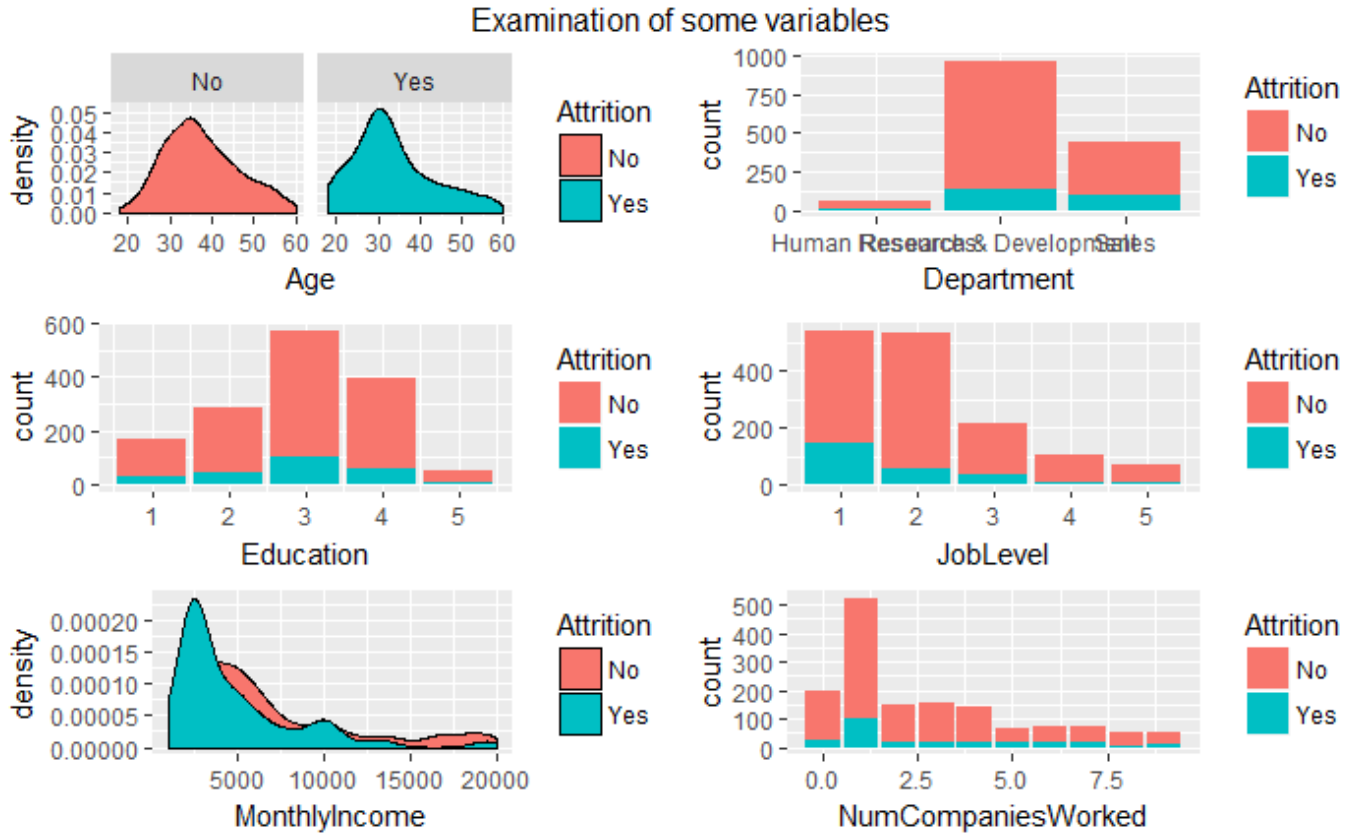


As we can see here, 237/1470 = 16 % of the data label shows the “Yes” in Attrition. this problem should be handeled during the process because unbalanced dataset will bias the prediction model towards the more common class (here is ‘NO’). There are different approaches for dealing with unbalanced data in machine learning like using more data, Resampling , changing the machine performance metric, using various algorithms etc.

Let’s look at some variable and see its influence on the Attrition of the organization.

Hide

```
library(ggplot2)
library(grid)
library(gridExtra)
agePlot <- ggplot(data,aes(Age,fill=Attrition))+geom_density()+facet_grid(~Attrition)
depPlot <- ggplot(data,aes(Department,fill = Attrition))+geom_bar()
eduPlot <- ggplot(data,aes(Education,fill=Attrition))+geom_bar()
jobLevelPlot <- ggplot(data,aes(JobLevel,fill=Attrition))+geom_bar()
monthlyIncPlot <- ggplot(data,aes(MonthlyIncome,fill=Attrition))+geom_density()
numCompPlot <- ggplot(data,aes(NumCompaniesWorked,fill=Attrition))+geom_bar()
grid.arrange(agePlot,depPlot,eduPlot,jobLevelPlot,
             monthlyIncPlot,numCompPlot,ncol=2,top = "Examination of some variables")
```




Age: We see that majority of employees leaving the company are around 30 Years.

Department: Among people attrited employees from Human Resources department are less.It is because of low proportion of HR in the company.

Education: From the data we know that 1 ‘Below College’ 2 ‘College’ 3 ‘Bachelor’ 4 ‘Master’ 5 ‘Doctor’ . Looking at the plot we see that very few Doctors attrite. May be because of less number.

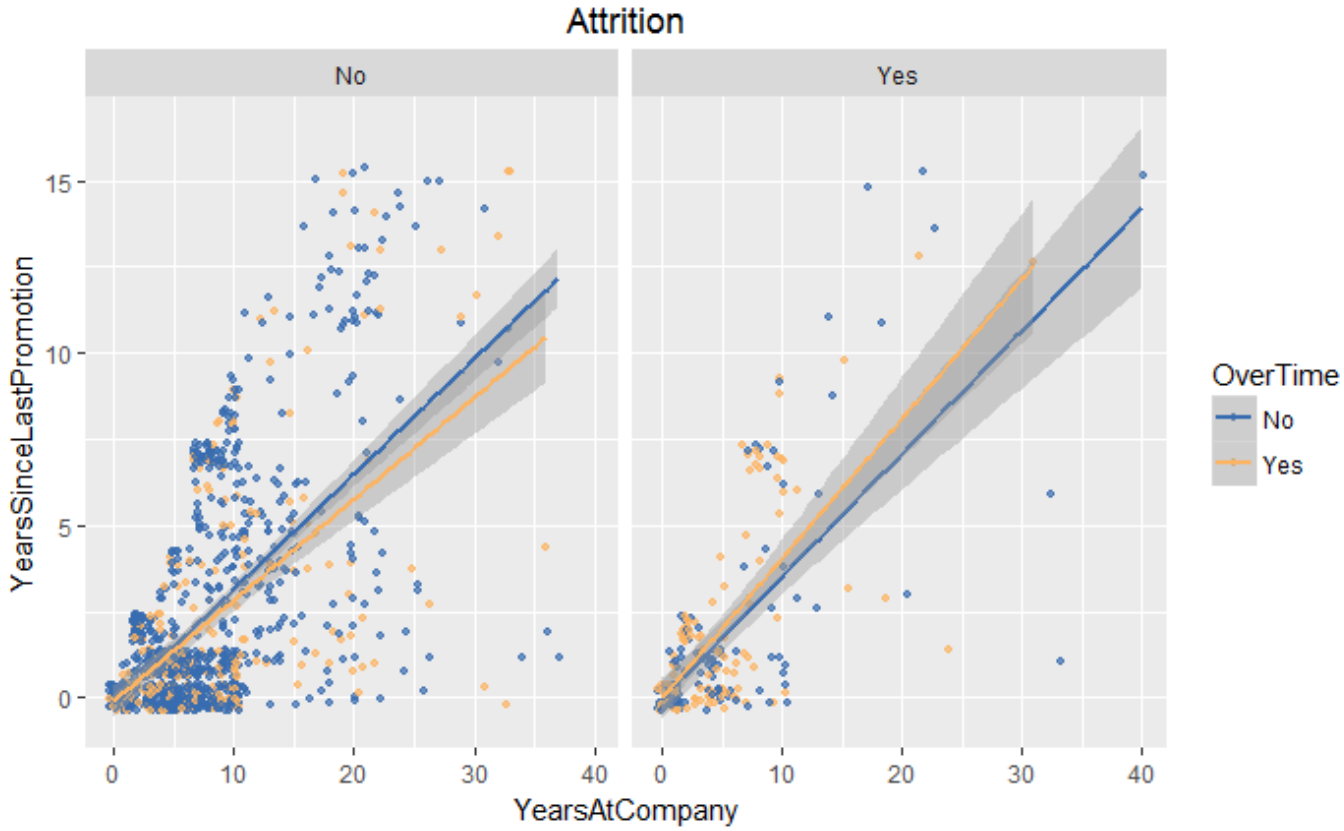
JobLevel: By looking at proportion of people seems like 1 stands for entry level and 5 stands for highest level in our Dataset. By looking at plot we see that as the Job Level increases the number of people quitting decreases.

Monthly Income: We see higher levels of attrition among the lower segment of monthly income. If looked at in isolation, might be due to dissatisfaction of income for the effort out.

Number of Companies Worked: We see a clear indication that many people who have worked only in One company before quit a lot.

Hide

```
ggplot(data,
       aes(y = YearsSinceLastPromotion, x = YearsAtCompany, colour = OverTime)) +
       geom_jitter(size = 1, alpha = 0.7) +
       geom_smooth(method = "gam") +
       facet_wrap(~ Attrition) +
       ggtitle("Attrition") +
       scale_colour_manual(values = c("#386cb0","#fdb462")) +
       theme(plot.title = element_text(hjust = 0.5))
```




This graph shows Years at company in relation to Years since last promotion, grouped by both attrition and overtime. A high correlation between these two variables (longer you are in the company, less chance you have to be promoted, so to speak) may mean that people are not really growing within the company. However, since this is a simulated dataset we cannot compare it with some norms outside it, so we can compare certain groups within our set, e.g. those who are working overtime and those who are not.

# 2.Single tree model

## Creating train & test sets

Hide

```
set.seed(100)
sub<-sample(1:nrow(data),nrow(data)*0.7)
length(sub)
```

```
[1] 1029
```

Hide

```
data_train<-data[sub,]
data_test<-data[-sub,]
dim(data_train)
```

```
[1] 1029   35
```

Hide

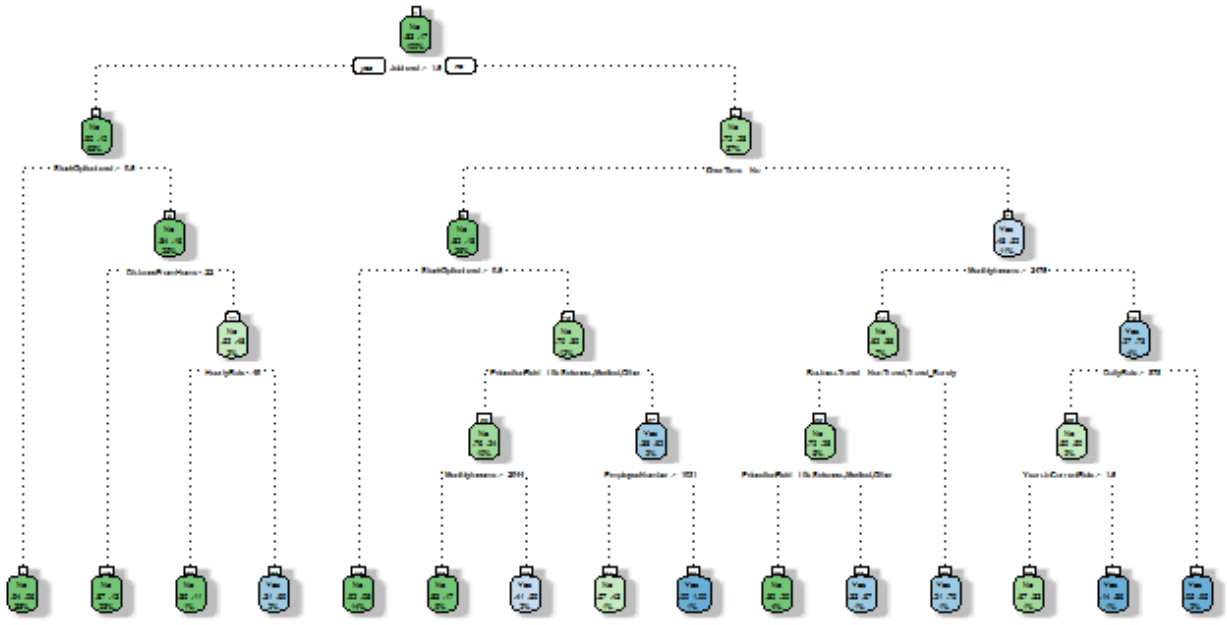
```
dim(data_test)
```

```
[1] 441  35
```

## Fit the single tree model

Hide

```
library(rpart)
library(rpart.plot)
library(rattle)
fit <- rpart(Attrition ~ .,method="class",
             data=data_train)
fancyRpartPlot(fit)
```



Rattle 2017-十二月-14 22:29:10 ldztababa

Hide

```
#Pruning the Decision Tree
fit <- prune(tree = fit , cp = 0.01)
```

## Predicton

Hide

```
pred <- predict(fit,data_test,type="class")
table(pred)
```

```
pred
 No Yes
396  45
```

## Result

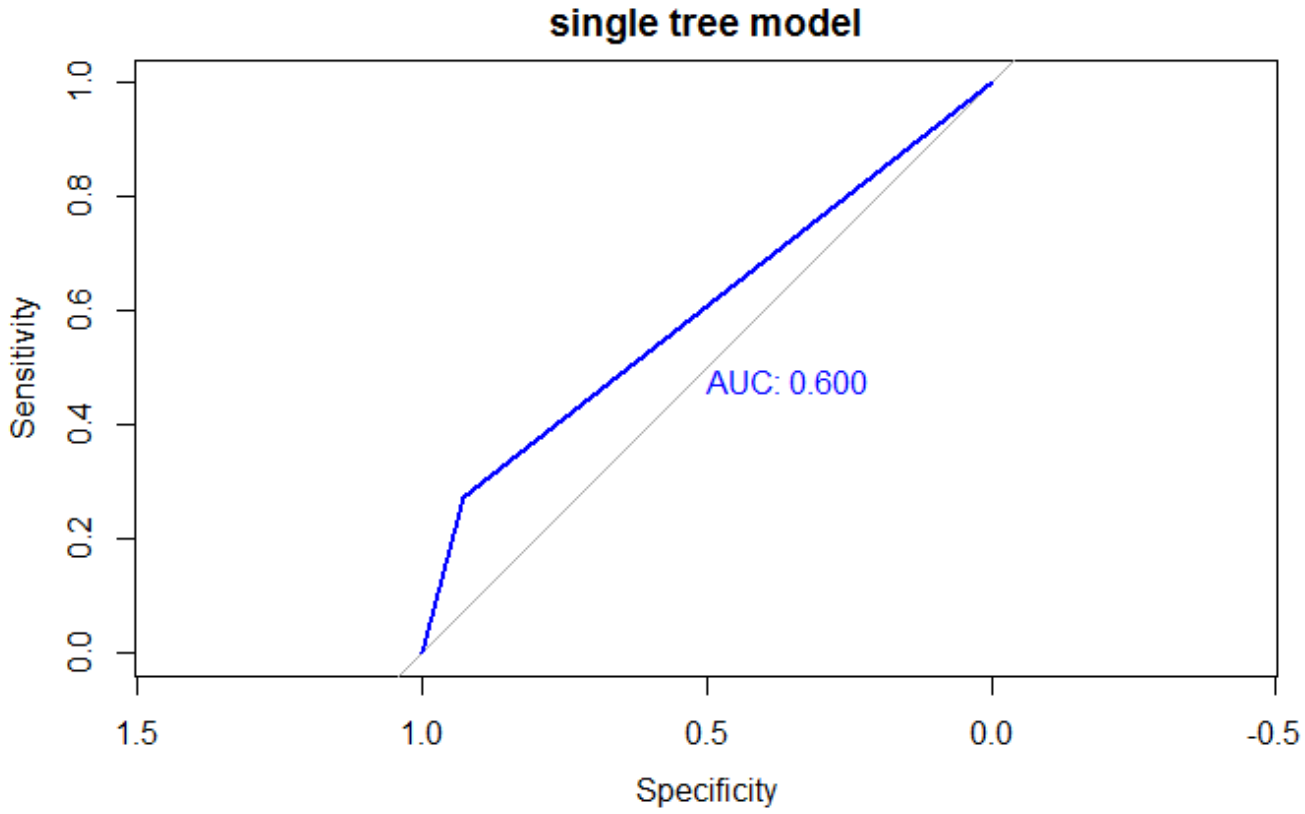
Hide

```
library(caret)
confusionMatrix(data_test$Attrition,pred)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  348  27
       Yes  48  18

               Accuracy : 0.8299
                 95% CI : (0.7915, 0.8638)
    No Information Rate : 0.898
    P-Value [Acc > NIR] : 1.00000

                  Kappa : 0.231
 Mcnemar's Test P-Value : 0.02092

            Sensitivity : 0.8788
            Specificity : 0.4000
         Pos Pred Value : 0.9280
         Neg Pred Value : 0.2727
             Prevalence : 0.8980
         Detection Rate : 0.7891
   Detection Prevalence : 0.8503
      Balanced Accuracy : 0.6394

       'Positive' Class : No
```

Hide

```
data_test$Attrition <- as.factor(data_test$Attrition)
plot.roc(as.numeric(data_test$Attrition), as.numeric(pred),print.auc=TRUE,col ="blue",main="single tre
e model")
```




We got the accuracy of 0.8299 and AUC is 0.600, which we could say that the single tree model is a good model.

# 3.Ensemble Technique 1 – Random Forest

We have to change the character into factor in order to fit the random forest model.

Hide

```
cols <- c(2,3,5,8,12,16,18,22,23)
data[cols] <- lapply(data[cols], factor)
table(sapply(data,class))
```

```
 factor numeric
      9      26
```

Hide

```
set.seed(100)
sub<-sample(1:nrow(data),nrow(data)*0.7)
length(sub)
```

```
[1] 1029
```

Hide

```
train<-data[sub,]
test<-data[-sub,]
dim(train)
```

```
[1] 1029   35
```

Hide

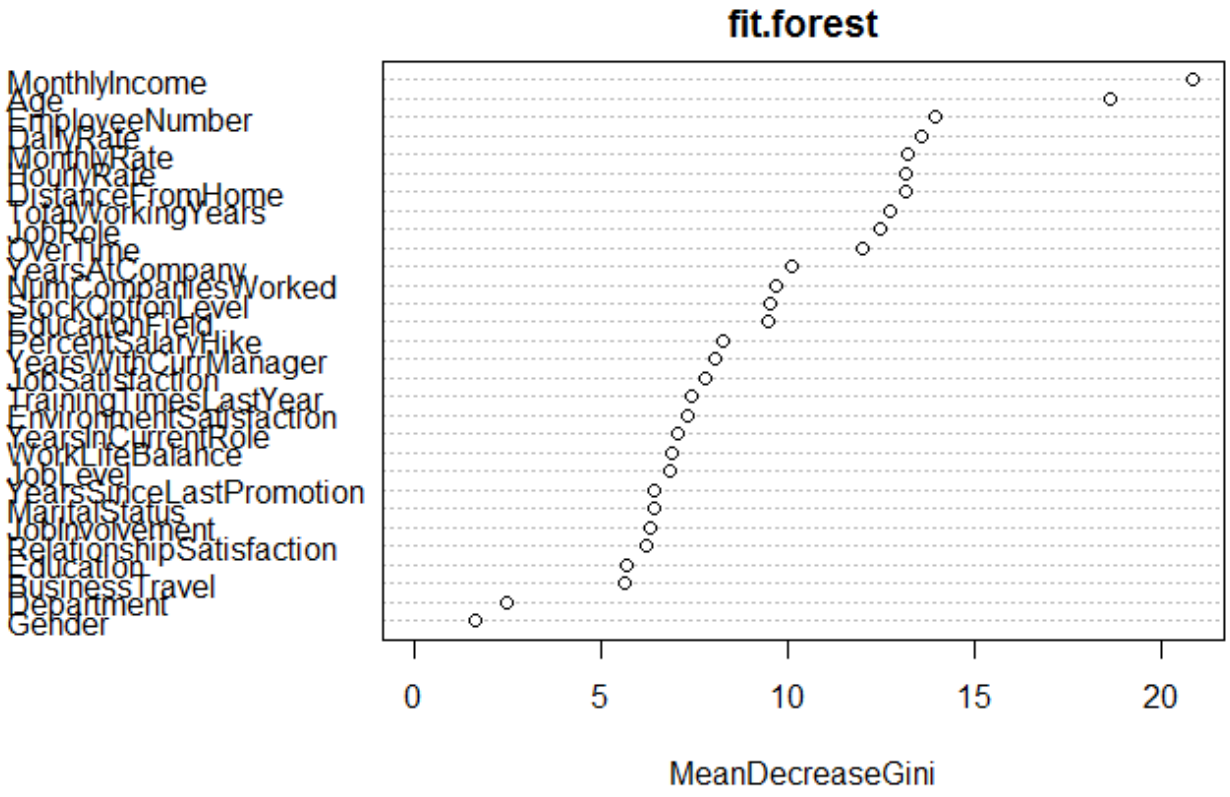
```
dim(test)
```

```
[1] 441  35
```

Fit the random forest model

Hide

```
library(randomForest)
set.seed(2343)
# Random forest
fit.forest <- randomForest(Attrition ~., data = train, ntree = 500)
rfpreds <- predict(fit.forest, test, type = "class")
varImpPlot(fit.forest)
```




Hide

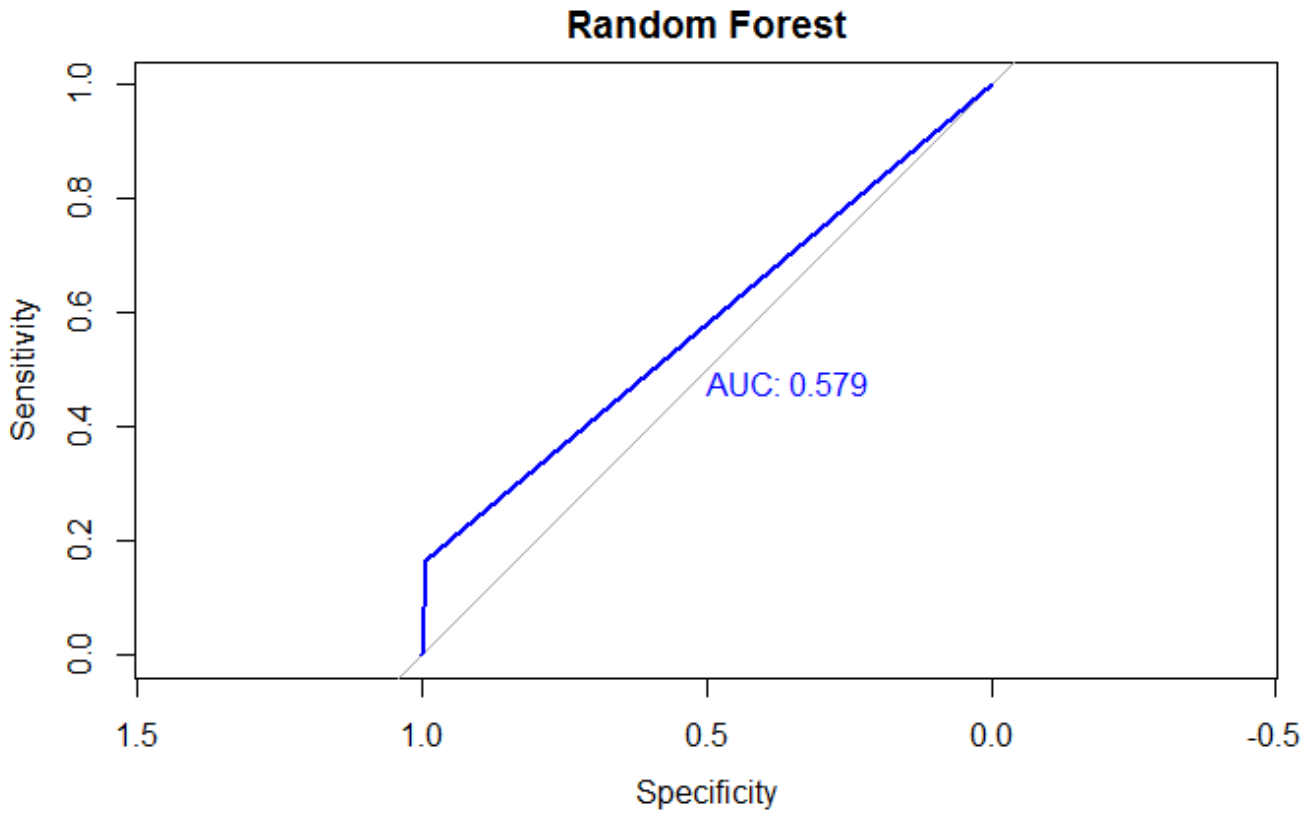
```
library(caret)
confusionMatrix(test$Attrition,rfpreds)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  372   3
       Yes  55  11

               Accuracy : 0.8685
                 95% CI : (0.8333, 0.8986)
    No Information Rate : 0.9683
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2349
 Mcnemar's Test P-Value : 2.133e-11

            Sensitivity : 0.8712
            Specificity : 0.7857
         Pos Pred Value : 0.9920
         Neg Pred Value : 0.1667
             Prevalence : 0.9683
         Detection Rate : 0.8435
   Detection Prevalence : 0.8503
      Balanced Accuracy : 0.8285

       'Positive' Class : No
```

As we see here, MonthlyIncome, Age, EmployeeNumber, DailyRate are the top variables, they are the most important variables for the random forest model.

Hide

```
plot.roc(as.numeric(test$Attrition), as.numeric(rfpreds),print.auc=TRUE,col ="blue",main="Random Forest")
```




***Some parameters:***

**formula：** Attrition ~.

**ntree：** Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

**mtry：** Number of variables randomly sampled as candidates at each split. Here the default is 2.

I changed some parameters here(for example I set the ntree from 10 to 700), but it does not influence the outpute, I got the same accuracy and AUC. The result could probably not be improved though changing the parameter for this data set.

The accuracy of 0.8685 is slightly better than the previous model, but the AUC is lower.We could also say that this ensemble model is a good model for this data set.

# 4.Ensemble Technique 2 – XGBoost

Hide

```
library(xgboost)
set.seed(123)
#We drop column 22 because contrasts can be applied only to factors with 2 or more levels
data <- data[,-22]
xgbData <- data
indexes <- sample(1:nrow(xgbData), size=0.7*nrow(xgbData))
XGBtrain.Data <- xgbData[indexes,]
XGBtest.Data <- xgbData[-indexes,]
```

Hide

```
formula = Attrition~.
fitControl <- trainControl(method="cv", number = 3,classProbs = TRUE )
xgbGrid <- expand.grid(nrounds = 50,
                       max_depth = 12,
                       eta = .03,
                       gamma = 0.01,
                       colsample_bytree = .7,
                       min_child_weight = 1,
                       subsample = 0.9
)
```

Hide

```
XGB.model <- train(formula, data = XGBtrain.Data,
                   method = "xgbTree"
                   ,trControl = fitControl
                   , verbose=0
                   , maximize=FALSE
                   ,tuneGrid = xgbGrid
)
```

***Some parameters:***

**max_depth：** Maximum depth of a tree, increase this value will make the model more complex / likely to be overfitting, we set it to 12.

**colsample_bytree：** Subsample ratio of columns when constructing each tree.

**maximize：** Whether to maximize the evaluation metric.

**gamma：** Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger, the more conservative the algorithm will be.

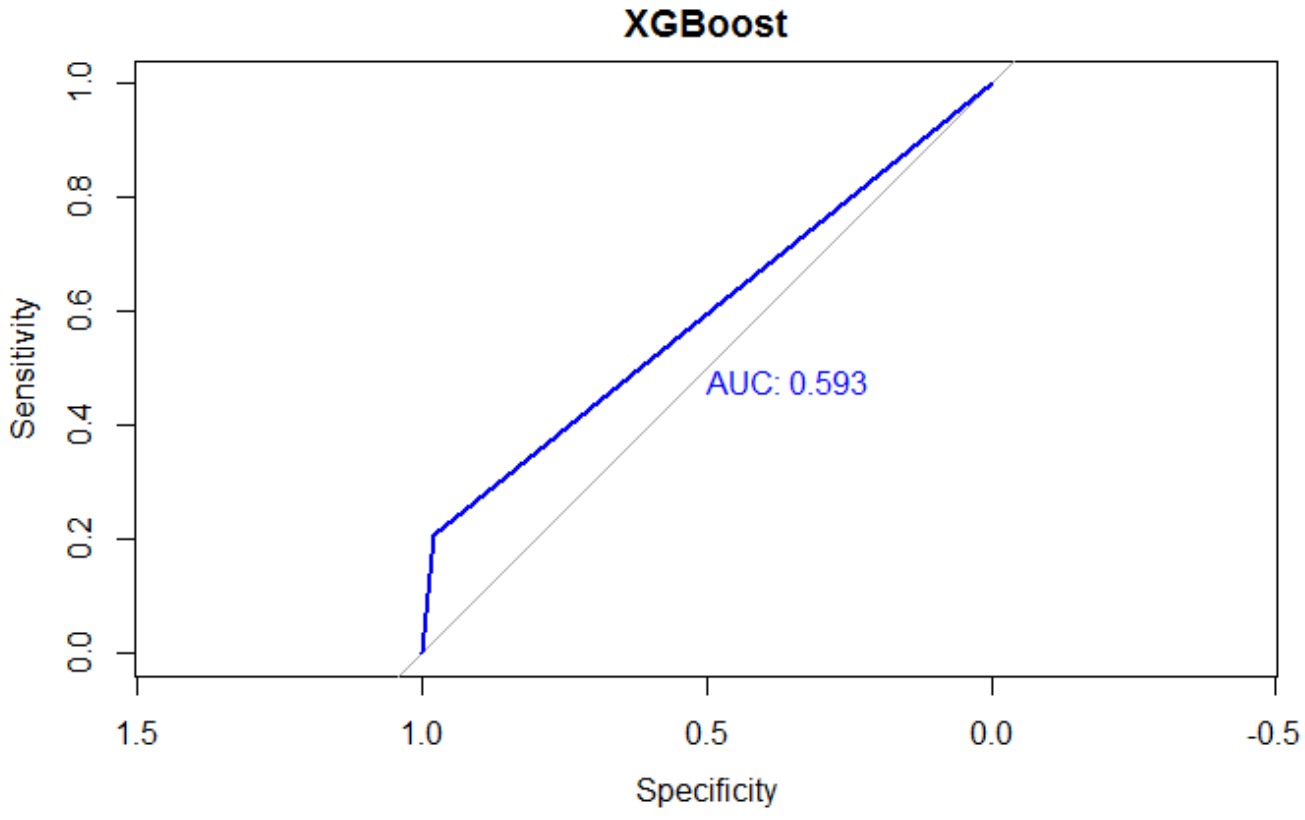
Hide

```
XGB.prd <- predict(XGB.model,XGBtest.Data)
confusionMatrix(XGB.prd, XGBtest.Data$Attrition)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  356  61
       Yes   8  16

               Accuracy : 0.8435
                 95% CI : (0.8062, 0.8762)
    No Information Rate : 0.8254
    P-Value [Acc > NIR] : 0.1738

                  Kappa : 0.255
 Mcnemar's Test P-Value : 3.848e-10

            Sensitivity : 0.9780
            Specificity : 0.2078
         Pos Pred Value : 0.8537
         Neg Pred Value : 0.6667
             Prevalence : 0.8254
         Detection Rate : 0.8073
   Detection Prevalence : 0.9456
      Balanced Accuracy : 0.5929

       'Positive' Class : No
```

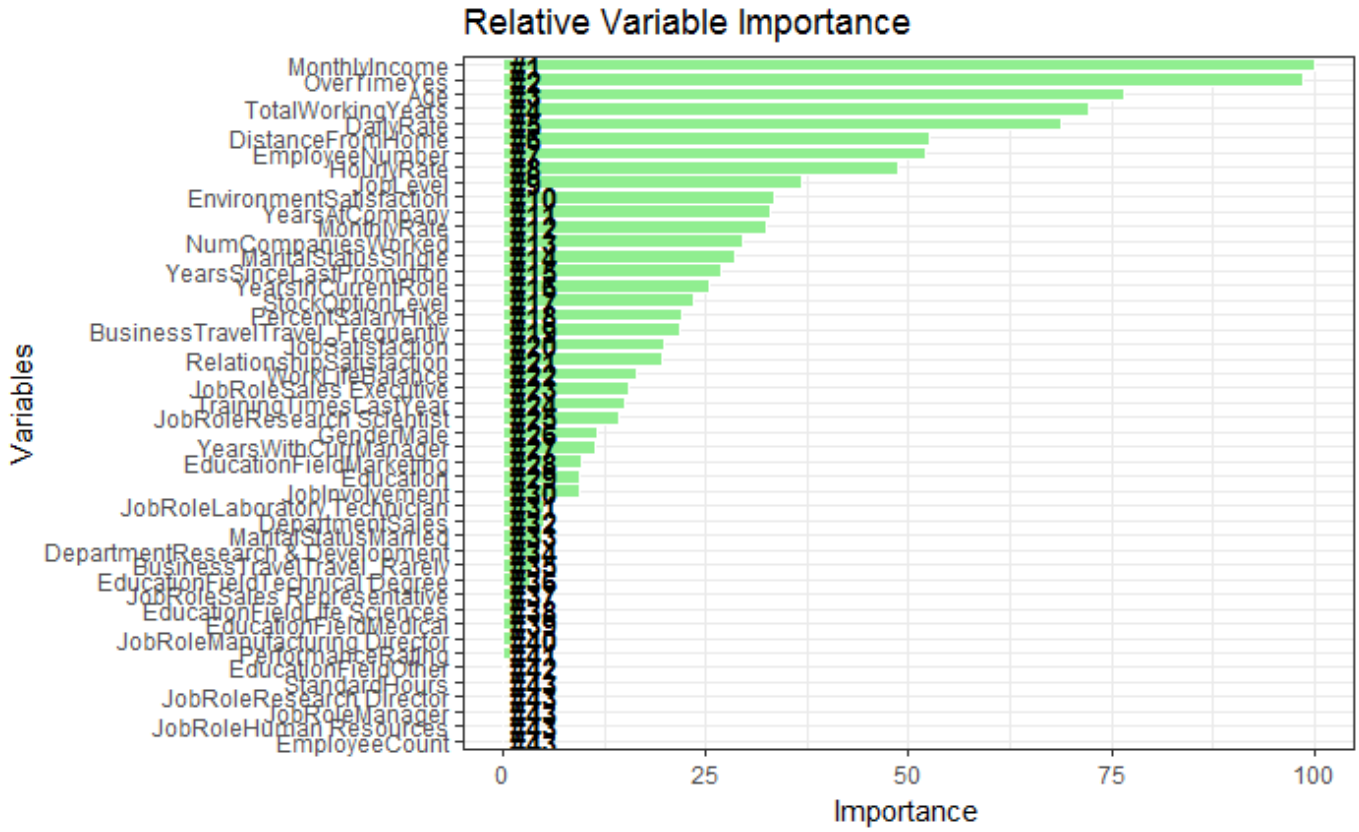
Hide

```
plot.roc(as.numeric(XGBtest.Data$Attrition), as.numeric(XGB.prd),print.auc=TRUE,col ="blue",main="XGBoost")
```




Hide

```
importance <- varImp(XGB.model)
varImportance <- data.frame(Variables = row.names(importance[[1]]),
                            Importance = round(importance[[1]]$Overall,2))
# Create a rank variable based on importance of variables
rankImportance <- varImportance %>%
        mutate(Rank = paste0('#',dense_rank(desc(Importance))))
ggplot(rankImportance, aes(x = reorder(Variables, Importance),
                           y = Importance)) +
        geom_bar(stat='identity',colour="white", fill = "lightgreen") +
        geom_text(aes(x = Variables, y = 1, label = Rank),
                  hjust=0, vjust=.5, size = 4, colour = 'black',
                  fontface = 'bold') +
        labs(x = 'Variables', title = 'Relative Variable Importance') +
        coord_flip() +
        theme_bw()
```




As we see above: MonthlyIncome, OvertimeYes, Age, TotalworkingYears and DailyRate are the most important five attributes.

Hide

```
#Change some parameters and look at the output
xgbGrid <- expand.grid(nrounds = 50,
                       max_depth = 6,
                       eta = .03,
                       gamma = 0.01,
                       colsample_bytree = 1,
                       min_child_weight = 1,
                       subsample = 0.9
)
XGB.model <- train(formula, data = XGBtrain.Data,
                   method = "xgbTree"
                   ,trControl = fitControl
                   , verbose=0
                   , maximize=FALSE
                   ,tuneGrid = xgbGrid
)
XGB.prd <- predict(XGB.model,XGBtest.Data)
confusionMatrix(XGB.prd, XGBtest.Data$Attrition)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  353  55
       Yes  11  22

               Accuracy : 0.8503
                 95% CI : (0.8136, 0.8823)
    No Information Rate : 0.8254
    P-Value [Acc > NIR] : 0.09207

                  Kappa : 0.3298
 Mcnemar's Test P-Value : 1.204e-07

            Sensitivity : 0.9698
            Specificity : 0.2857
         Pos Pred Value : 0.8652
         Neg Pred Value : 0.6667
             Prevalence : 0.8254
         Detection Rate : 0.8005
   Detection Prevalence : 0.9252
      Balanced Accuracy : 0.6277

       'Positive' Class : No
```

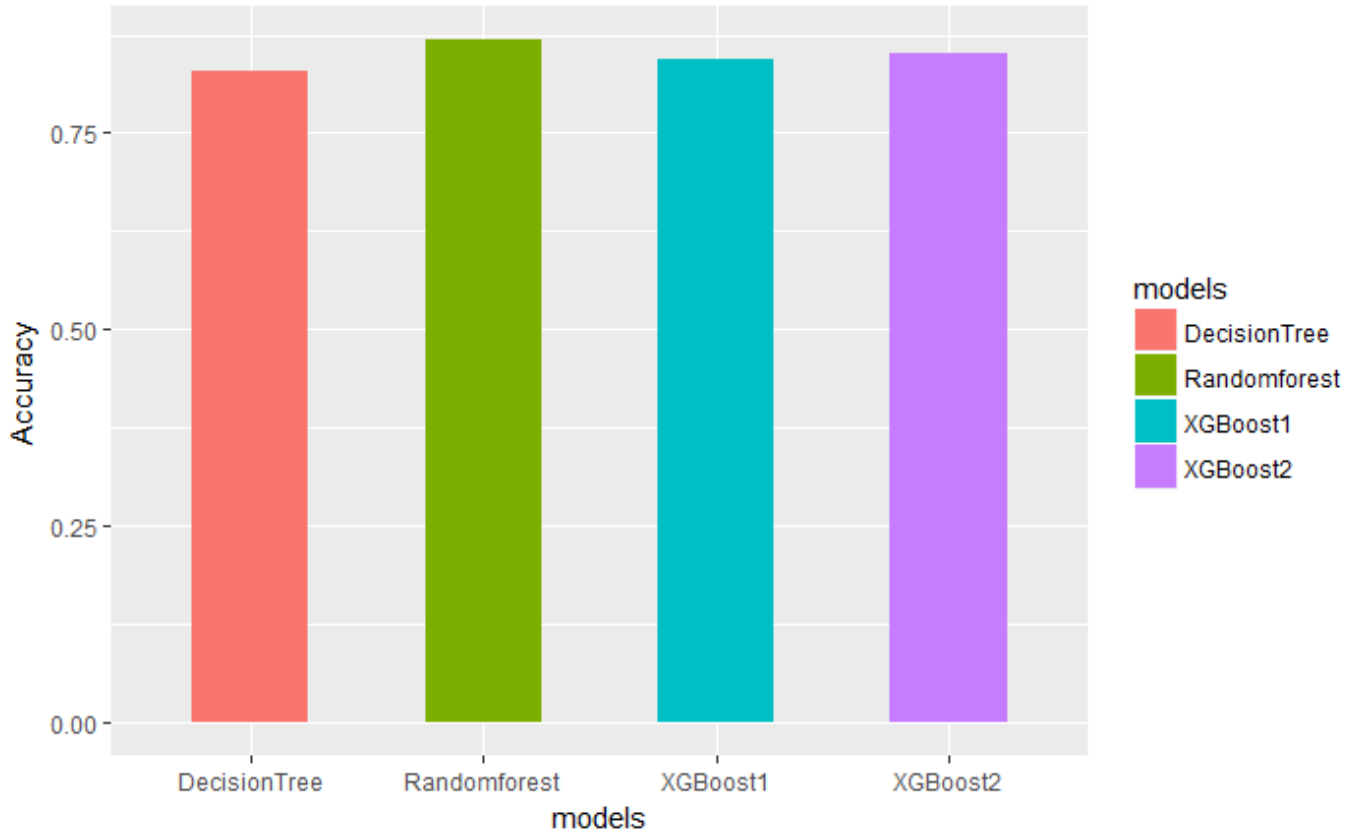
I changed some parameters(max_depth = 6, colsample_bytree = 1), the output didn't have too much difference.

The accuracy could get 0.85 by xgboost and the AUC is 0.593.It is a very good moedl for the classification.

# 5.Comparison of the above models

Hide

```
Comparison <- data.frame(models=c("DecisionTree","Randomforest","XGBoost1","XGBoost2"),Accuracy=c(0.8299,0.8685,0.8435,0.8503))
ggplot(data = Comparison, aes(x = models, y = Accuracy, fill= models))+
  geom_bar(stat = "identity", width = 0.5)
```




From the plot we can see that all of these models have very good performance. We could say that randomfoerest is the best choice for this data set. And different parameters don't have too much influence on these model's performance.

Random Forests are an improvement over bagged decision trees. A problem with decision trees like CART is that they are greedy. They choose which variable to split on using a greedy algorithm that minimizes error. As such, even with Bagging, the decision trees can have a lot of structural similarities and in turn have high correlation in their predictions. Combining predictions from multiple models in ensembles works better if the predictions from the sub-models are uncorrelated or at best weakly correlated. Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all of the subtrees have less correlation. It is a simple tweak. In CART, when selecting a split point, the learning algorithm is allowed to look through all variables and all variable values in order to select the most optimal split-point. The random forest algorithm changes this procedure so that the learning algorithm is limited to a random sample of features of which to search.