Predicting Earnings Manipulation

Krishangi, Dhruv

6/22/2020

Analysis for Predicting Earnings Manipulation by Indian Firms.

```
#summary(Sample.data)
str(Sample.data)

## tibble [220 x 11] (S3: tbl_df/tbl/data.frame)
##  $ Company ID   : num [1:220] 1 2 3 4 5 6 7 8 9 10 ...
##  $ DSRI         : num [1:220] 1.62 1 1 1.49 1 ...
##  $ GMI          : num [1:220] 1.13 1.61 1.02 1 1.37 ...
##  $ AQI          : num [1:220] 7.185 1.005 1.241 0.466 0.637 ...
##  $ SGI          : num [1:220] 0.366 13.081 1.475 0.673 0.861 ...
##  $ DEPI         : num [1:220] 1.38 0.4 1.17 2 1.45 ...
##  $ SGAI         : num [1:220] 1.6241 5.1982 0.6477 0.0929 1.7415 ...
##  $ ACCR         : num [1:220] -0.1668 0.0605 0.0367 0.2734 0.123 ...
##  $ LEVI         : num [1:220] 1.161 0.987 1.264 0.681 0.939 ...
##  $ Manipulator  : chr [1:220] "Yes" "Yes" "Yes" "Yes" ...
##  $ C-MANIPULATOR: num [1:220] 1 1 1 1 1 1 1 1 1 1 ...
```

Converting our target variable into factor. Making data ready for analysis.

```
table(Sample.data$Mani)

## 
##   0   1 
## 181  39
```

The data is unbalanced since there is an unequal distribution of data amongst the two classes.

Sampling the Sample.data dataset into train and test data.

```
set.seed(123)
index<-sample(2, nrow(Sample.data), replace=TRUE,prob=c(0.7,0.3))
train<-Sample.data[index==1,]
test<-Sample.data[index==2,]

table(train$Mani)

## 
##   0   1 
## 135  25
```

Clearly the data is unbalanced. Since the number of observations for no manipulation(0) is more than that of manipulated observations(1).

Undersampling the data for better analysis.

```
under<-ovun.sample(Mani~.,data=train , method="under", N=50)$data
table(under$Mani)

##
##  0  1
## 25 25
```

LOGISTIC REGRESSION MODEL using stepwise variable selection for *Sample Data*.

Variables selected using stepwise method: DSRI, SGI, AQI, ACCR, GMI

```
logit.model <- glm(Mani~DSRI+SGI+AQI+ACCR+GMI, data=under, family = "binomial
")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(logit.model)

##
## Call:
## glm(formula = Mani ~ DSRI + SGI + AQI + ACCR + GMI, family = "binomial",
##     data = under)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.40030  -0.32878  -0.00348   0.34686   2.00616
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.9907     4.6283  -3.239  0.00120 **
## DSRI          4.2540     1.4458   2.942  0.00326 **
## SGI           5.7864     2.3208   2.493  0.01266 *
## AQI           0.9798     0.3352   2.923  0.00347 **
## ACCR         14.0498     6.7634   2.077  0.03777 *
## GMI           1.2567     0.5838   2.153  0.03135 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 69.315  on 49  degrees of freedom
## Residual deviance: 26.100  on 44  degrees of freedom
## AIC: 38.1
```

```
##
## Number of Fisher Scoring iterations: 9
```

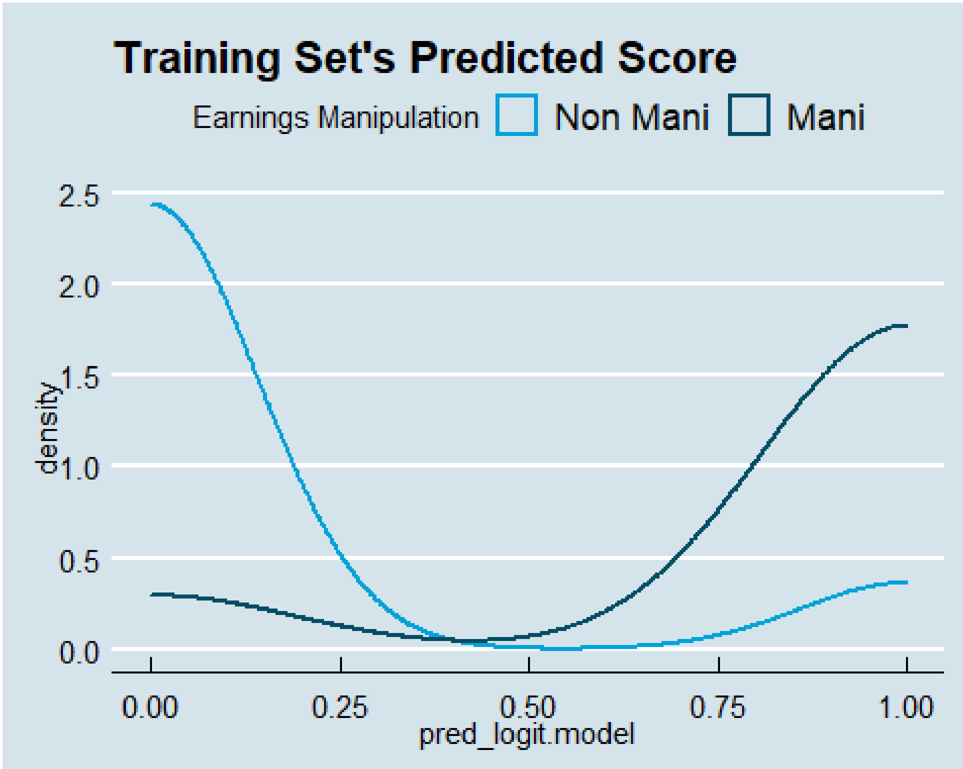
Equation: y = -14.9907 +4.2540DSRI + 5.7864SGI + 0.9798AQI + 14.0498ACCR + 1.2567GMI

Predicting our model performance on test data.

```
pred_logit.model <- predict(logit.model, test, type = "response")
pred_logit.model <- round(pred_logit.model)
pred_logit.model

##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26
##  1  1  0  1  0  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  1  0  0
0
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52
##  0  0  0  0  0  1  0  0  0  1  1  0  0  0  0  0  1  0  0  0  0  0  0  0  1
0
## 53 54 55 56 57 58 59 60
##  0  0  0  0  0  0  0  0

#summary(pred_logit.model)
```

Training set's predicted score.

```
ggplot( test, aes( pred_logit.model, color = Mani ) ) +
geom_density( size = 1 ) +
ggtitle( "Training Set's Predicted Score" ) +
scale_color_economist( name = "Earnings Manipulation", labels = c( "Non Mani"
, "Mani" ) ) +
theme_economist()
```



Confusion Matrix

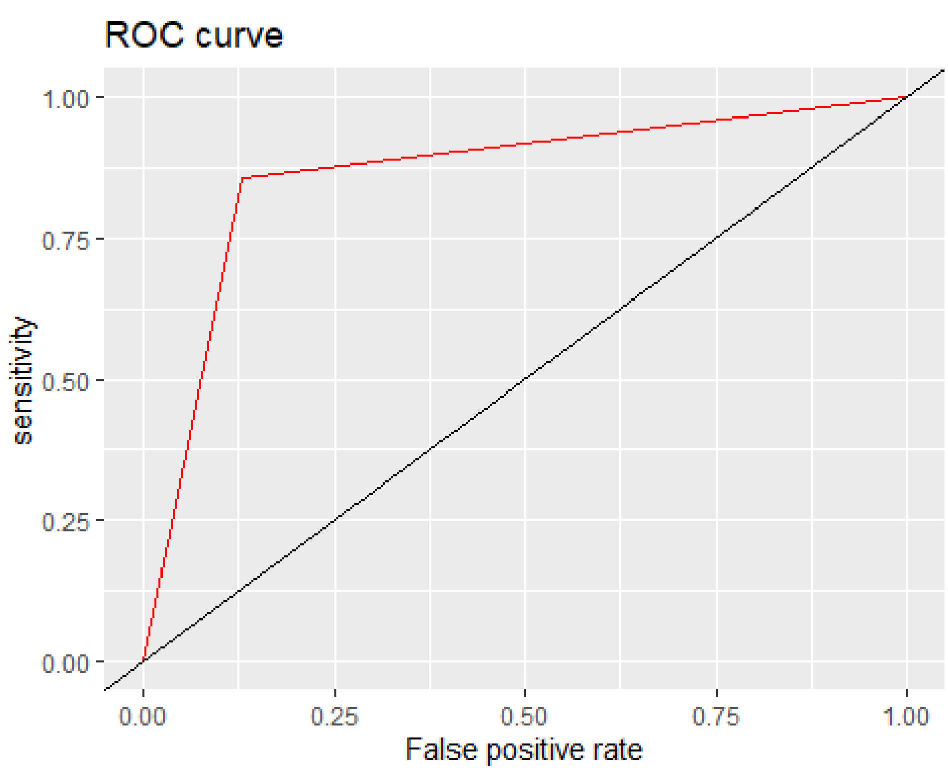
```
actual<-test$Mani
ConfMat <- table(pred_logit.model,actual,dnn=c("Prediction","Actual"))
ConfMat

##             Actual
## Prediction  0  1
##           0 40  2
##           1  6 12

result <- confusionMatrix(ConfMat)
result

## Confusion Matrix and Statistics
##
##             Actual
## Prediction  0  1
##           0 40  2
##           1  6 12
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.7667
##     P-Value [Acc > NIR] : 0.04068
##
##                   Kappa : 0.661
##
```

```
##  Mcnemar's Test P-Value : 0.28884
## 
##             Sensitivity : 0.8696
##             Specificity : 0.8571
##          Pos Pred Value : 0.9524
##          Neg Pred Value : 0.6667
##              Prevalence : 0.7667
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8634
## 
##        'Positive' Class : 0
## 
```

Accuracy of the logistic model: *86.67%*

Precision of the logistic model: *95.24%*

```
pred <- ROCR::prediction(pred_logit.model,actual)
perf <- ROCR::performance(pred, 'tpr', 'fpr')
pf <- data.frame(perf@x.values, perf@y.values)
names(pf) <- c("fpr", "tpr")
ggplot(data=pf,aes(x=fpr,y=tpr))+geom_line(colour='red')+geom_abline(intercep
t=0,slope=1)+labs(x='False positive rate',y='sensitivity',title='ROC curve')
```




```
#plot(perf)
```

Calculating area under curve for the ROC plot.

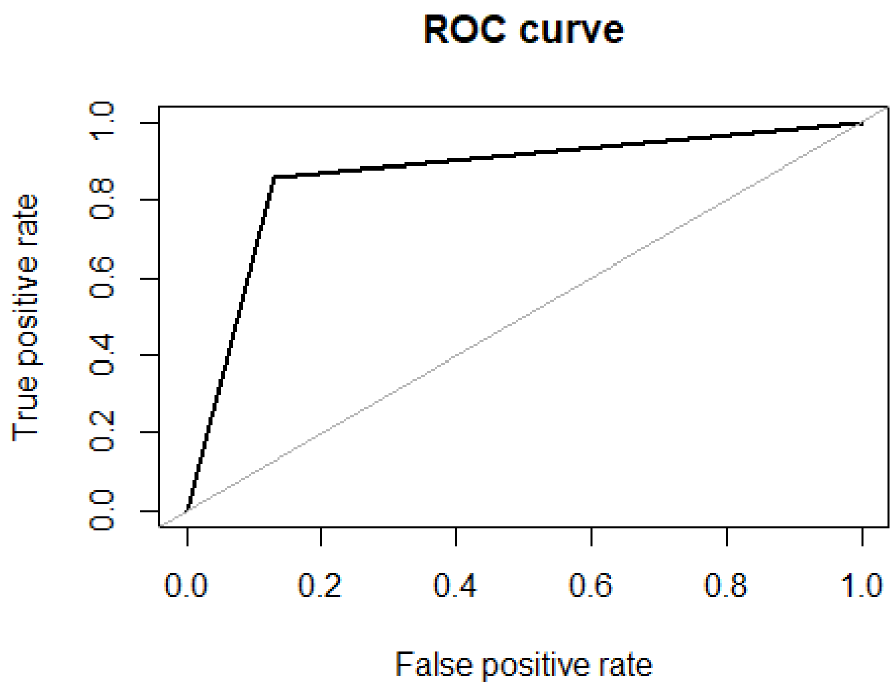
```
auc <- performance(pred,"auc")
auc <- unlist(slot(auc,"y.values"))
paste("Area under curve: ", auc)

## [1] "Area under curve:  0.863354037267081"
```

Default cut off is at 0.5

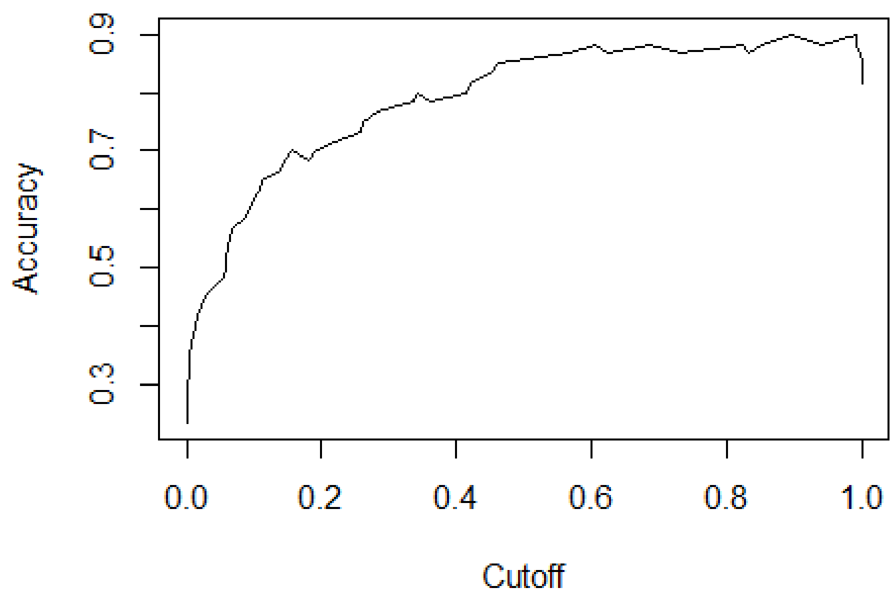
```
metrics<-function(model,data,cutoff){
  data$model_prob<-predict(model,newdata=data, type = "response")
  data <- data  %>% mutate(model_pred = 1*(model_prob > cutoff) + 0)
  data <- data %>% mutate(accurate = 1*(model_pred == Mani))
  confusion<-table(data$Mani,data$model_pred)
  sens<-confusion[[1]]/sum(confusion[,1])
  spec<-confusion[[1,2]]/sum(confusion[,2])
  accuaracy<-sum(data$accurate)/nrow(data)
  fnr<-confusion[2,1]/sum(confusion[,1])
  fOr<-confusion[2,1]/sum(confusion[2,])
  return(list(Dataset = data,
              Confusin_matrix = confusion,
              Accuracy = accuaracy,
              Predicted_Class = data$model_pred,
              Sensitivity = sens,
              Specificity = spec,
              P10 = fnr,
              P01 = fOr,
              p1 = confusion[1,2],
              p2 = confusion[2,1]))
}

test_data <- metrics(logit.model,test,0.5)$Dataset
```

Calculating Youden's Index to find best cut-off point.

```
roc.plot<-roc.curve(test_data$Mani,test_data$model_pred)
```




```
pred<-prediction(test_data$model_prob,test_data$Mani)
plot(ROCR::performance(pred,"acc")) #accuracy by cutoff
```

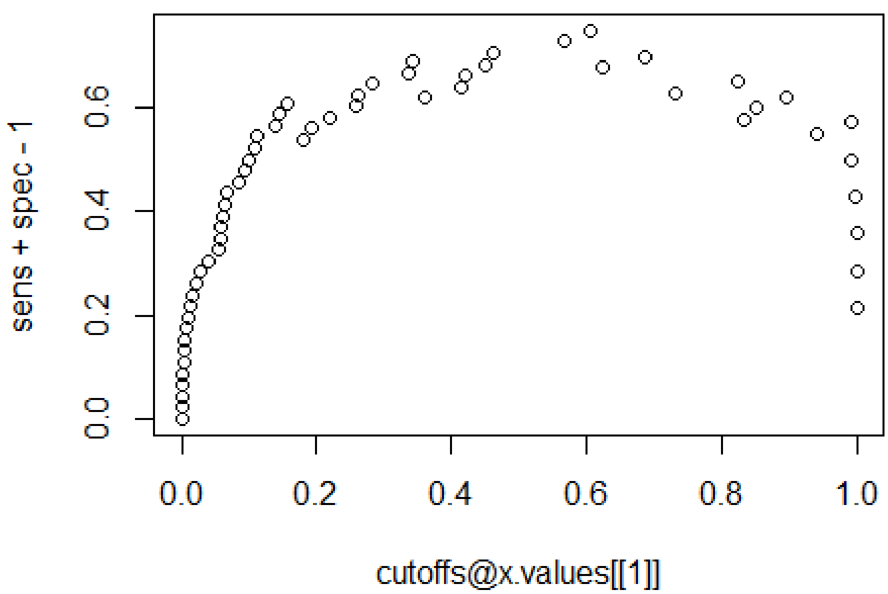


```r
cutoffs<-ROCR::performance(pred,"acc")
sens<-ROCR::performance(pred,"sens")@y.values[[1]]
spec<-ROCR::performance(pred,"spec")@y.values[[1]]
max(sens+spec-1)

## [1] 0.7484472
```

Youden's Index = 0.7484472

Now, to find the classification cutoff probability let's plot it and see.

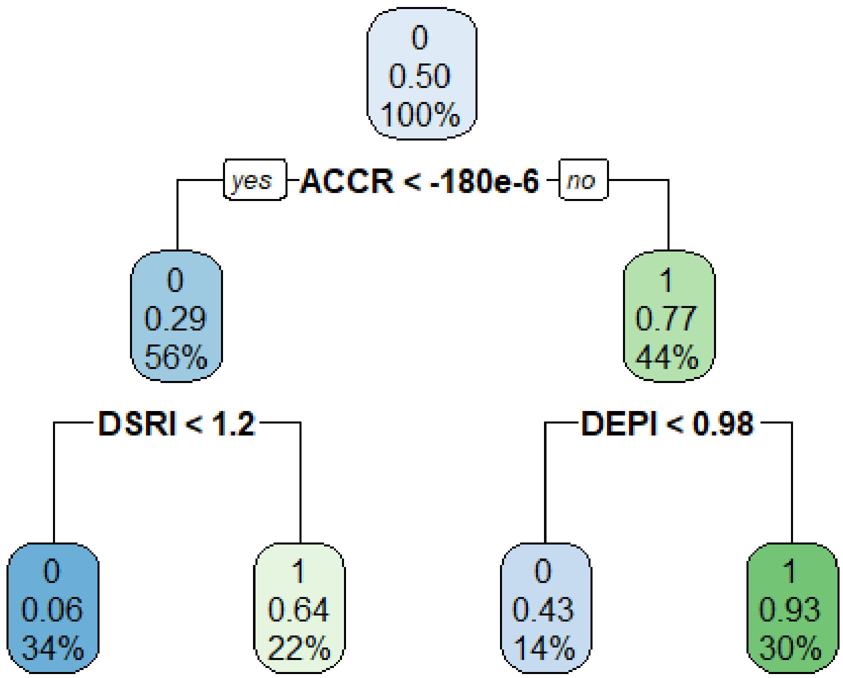
```r
plot(cutoffs@x.values[[1]],sens+spec-1)
```



The best cut off point is approximately *0.6*

CLASSIFICATION AND REGRESSION TREE MODEL (CART)

```r
set.seed(1234)
tree <- rpart(Mani ~ DSRI + GMI + AQI + SGI + DEPI + SGAI
              + ACCR + LEVI, data = under,control = rpart.control(c=-1),
              parms = list(split = "gini"),cp = 0.001)
rpart.plot(tree)
```




```
print(tree)

## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 25 0 (0.50000000 0.50000000)
##   2) ACCR< -0.000179724 28  8 0 (0.71428571 0.28571429)
##     4) DSRI< 1.247628 17  1 0 (0.94117647 0.05882353) *
##     5) DSRI>=1.247628 11  4 1 (0.36363636 0.63636364) *
##   3) ACCR>=-0.000179724 22  5 1 (0.22727273 0.77272727)
##     6) DEPI< 0.9802424 7  3 0 (0.57142857 0.42857143) *
##     7) DEPI>=0.9802424 15  1 1 (0.06666667 0.93333333) *

summary(tree)

## Call:
## rpart(formula = Mani ~ DSRI + GMI + AQI + SGI + DEPI + SGAI +
##     ACCR + LEVI, data = under, parms = list(split = "gini"),
##     control = rpart.control(c = -1), cp = 0.001)
##   n= 50
##
##       CP nsplit rel error xerror      xstd
## 1  0.48      0      1.00   1.48 0.1240645
## 2  0.12      1      0.52   0.92 0.1409681
## 3  0.04      2      0.40   1.00 0.1414214
```

```
## 4 -1.00      3      0.36   1.00 0.1414214
##
## Variable importance
## ACCR  SGI DSRI LEVI SGAI DEPI  GMI
##   24   19   19   13   11    8    7
##
## Node number 1: 50 observations,    complexity param=0.48
##   predicted class=0  expected loss=0.5  P(node) =1
##     class counts:    25    25
##    probabilities: 0.500 0.500
##   left son=2 (28 obs) right son=3 (22 obs)
##   Primary splits:
##       ACCR < -0.000179724 to the left,  improve=5.844156, (0 missing)
##       SGI  < 1.167219     to the left,  improve=5.392157, (0 missing)
##       SGAI < 0.9052362    to the right, improve=4.960317, (0 missing)
##       GMI  < 1.081076     to the left,  improve=3.319783, (0 missing)
##       DSRI < 1.724051     to the left,  improve=2.678571, (0 missing)
##   Surrogate splits:
##       SGI  < 1.167219     to the left,  agree=0.78, adj=0.500, (0 split)
##       LEVI < 0.7543345    to the right, agree=0.70, adj=0.318, (0 split)
##       SGAI < 0.892645     to the right, agree=0.68, adj=0.273, (0 split)
##       GMI  < 0.9992037    to the left,  agree=0.62, adj=0.136, (0 split)
##       DSRI < 1.004014     to the right, agree=0.60, adj=0.091, (0 split)
##
## Node number 2: 28 observations,    complexity param=0.12
##   predicted class=0  expected loss=0.2857143  P(node) =0.56
##     class counts:    20     8
##    probabilities: 0.714 0.286
##   left son=4 (17 obs) right son=5 (11 obs)
##   Primary splits:
##       DSRI < 1.247628     to the left,  improve=4.4553090, (0 missing)
##       GMI  < 1.00324      to the left,  improve=2.5785710, (0 missing)
##       SGI  < 1.008475     to the right, improve=2.5714290, (0 missing)
##       LEVI < 1.019826     to the left,  improve=2.1157510, (0 missing)
##       AQI  < 0.9206898    to the left,  improve=0.8086884, (0 missing)
##   Surrogate splits:
##       SGI  < 0.7367221    to the right, agree=0.821, adj=0.545, (0 split)
##       SGAI < 0.9022978    to the right, agree=0.750, adj=0.364, (0 split)
##       GMI  < 0.3469312    to the right, agree=0.714, adj=0.273, (0 split)
##       ACCR < -0.3739558   to the right, agree=0.714, adj=0.273, (0 split)
##       LEVI < 1.019826     to the left,  agree=0.714, adj=0.273, (0 split)
##
## Node number 3: 22 observations,    complexity param=0.04
##   predicted class=1  expected loss=0.2272727  P(node) =0.44
##     class counts:     5    17
##    probabilities: 0.227 0.773
##   left son=6 (7 obs) right son=7 (15 obs)
##   Primary splits:
##       DEPI < 0.9802424    to the left,  improve=2.432035, (0 missing)
##       LEVI < 0.9704765    to the left,  improve=1.893939, (0 missing)
```

```
##       SGI  < 1.137832     to the left,  improve=1.870130, (0 missing)
##       GMI  < 1.002935     to the right, improve=1.436674, (0 missing)
##       SGAI < 1.065197     to the right, improve=1.093939, (0 missing)
##   Surrogate splits:
##       DSRI < 0.7672722    to the left,  agree=0.773, adj=0.286, (0 split)
##       LEVI < 0.1925579    to the left,  agree=0.773, adj=0.286, (0 split)
##       SGI  < 2.26495      to the right, agree=0.727, adj=0.143, (0 split)
##
## Node number 4: 17 observations
##   predicted class=0  expected loss=0.05882353  P(node) =0.34
##     class counts:    16     1
##    probabilities: 0.941 0.059
##
## Node number 5: 11 observations
##   predicted class=1  expected loss=0.3636364  P(node) =0.22
##     class counts:     4     7
##    probabilities: 0.364 0.636
##
## Node number 6: 7 observations
##   predicted class=0  expected loss=0.4285714  P(node) =0.14
##     class counts:     4     3
##    probabilities: 0.571 0.429
##
## Node number 7: 15 observations
##   predicted class=1  expected loss=0.06666667  P(node) =0.3
##     class counts:     1    14
##    probabilities: 0.067 0.933
```

Decision Tree Inference ## If ACCR is greater than -18e-6 and DEPI > 0.98 then the company has 30% chance of being a Manipulator. ## If ACCR is less than -18e-6 and DSRI > 1.2 the firm has 22% chance of being a Manipulator.

```
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = Mani ~ DSRI + GMI + AQI + SGI + DEPI + SGAI +
##     ACCR + LEVI, data = under, parms = list(split = "gini"),
##     control = rpart.control(c = -1), cp = 0.001)
##
## Variables actually used in tree construction:
## [1] ACCR DEPI DSRI
##
## Root node error: 25/50 = 0.5
##
## n= 50
##
##      CP nsplit rel error xerror    xstd
## 1  0.48      0      1.00   1.48 0.12406
## 2  0.12      1      0.52   0.92 0.14097
```

```
## 3   0.04       2       0.40    1.00 0.14142
## 4  -1.00       3       0.36    1.00 0.14142
```

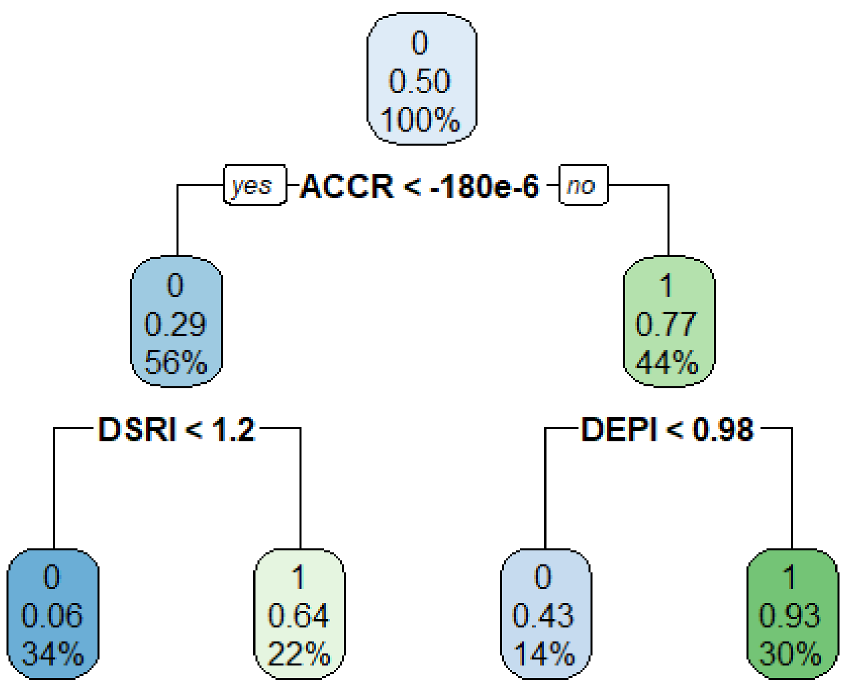
```
opt <-which.min(tree$cptable[ ,"xerror"])
cp<-tree$cptable[opt, "CP"]
cp
```

```
## [1] 0.12
```

Complexity parameter is 0.12

Decision tree after changing the cp value.

```
tree.opt <- rpart(Mani~., data = under, control = rpart.control(c=-1),parms =
list(split = "gini"), cp = 0.12)
rpart.plot(tree.opt)
```




```
print(tree.opt)
```

```
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 25 0 (0.50000000 0.50000000)
##   2) ACCR< -0.000179724 28  8 0 (0.71428571 0.28571429)
##     4) DSRI< 1.247628 17  1 0 (0.94117647 0.05882353) *
##     5) DSRI>=1.247628 11  4 1 (0.36363636 0.63636364) *
```

```
##   3) ACCR>=-0.000179724 22  5 1 (0.22727273 0.77272727)
##     6) DEPI< 0.9802424 7  3 0 (0.57142857 0.42857143) *
##     7) DEPI>=0.9802424 15  1 1 (0.06666667 0.93333333) *
```

```r
result.tree <- confusionMatrix(under$Mani, predict(tree.opt,type="class"))
result.tree
```

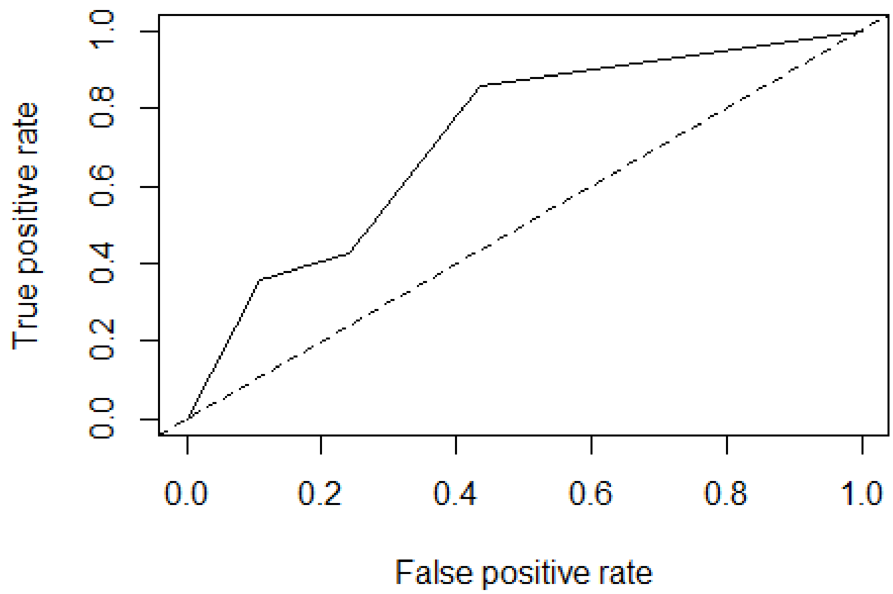
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20  5
##          1  4 21
##
##                Accuracy : 0.82
##                  95% CI : (0.6856, 0.9142)
##     No Information Rate : 0.52
##     P-Value [Acc > NIR] : 9.913e-06
##
##                   Kappa : 0.64
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8333
##             Specificity : 0.8077
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8400
##              Prevalence : 0.4800
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.8205
##
##        'Positive' Class : 0
##
```

The accuracy of the model is *82%*

The precision of the model is *80%*

Predicting performance of CART model on Test Data.

```r
Pred.cart = predict(tree.opt, newdata = test, type = "prob")[,2]
Pred2 = prediction(Pred.cart, test$Mani)
plot(performance(Pred2, "tpr", "fpr"))
abline(0, 1, lty = 2)
```



Calculating area under curve for the ROC plot.

```
auc <- performance(Pred2,"auc")
auc <- unlist(slot(auc,"y.values"))
paste("Area under curve: ", auc)

## [1] "Area under curve:  0.721273291925466"
```

LOGISTIC REGRESSION using stepwise variable selection for *Complete Data*.

```
set.seed(1234)
index<-sample(2, nrow(Complete.data), replace=TRUE,prob=c(0.7,0.3))
train.c<-Complete.data[index==1,]
test.c<-Complete.data[index==2,]
table(train.c$Manipulator)

##
##   0   1
## 843  31
```

Clearly the data is unbalanced again. Since the number of observations for no manipulation(0) is more than that of manipulated observations(1).

Undersampling the data for better analysis.

```
under.c<-ovun.sample(Manipulator~.,data=train.c , method="under", N=62)$data
table(under.c$Manipulator)

##
##  0  1
## 31 31
```

DSRI, SGI, ACCR, AQI are the significant variables. So we'll use these variables in our logistic regression model.

```
mylogit<-glm(Manipulator~DSRI+SGI+ACCR+AQI,data=under.c, family = "binomial")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(mylogit)

##
## Call:
## glm(formula = Manipulator ~ DSRI + SGI + ACCR + AQI, family = "binomial",
##     data = under.c)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.32410  -0.54957  -0.01272   0.35506   1.63113
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.3769     3.8195  -2.979  0.00290 **
## DSRI          4.9715     1.8565   2.678  0.00741 **
## SGI           3.5622     1.5135   2.354  0.01859 *
## ACCR         11.0802     4.3089   2.571  0.01013 *
## AQI           0.5271     0.2430   2.169  0.03007 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 85.950  on 61  degrees of freedom
## Residual deviance: 41.514  on 57  degrees of freedom
## AIC: 51.514
##
## Number of Fisher Scoring iterations: 9
```

Comparison of the previous logistic model and the new logistic model: EQUATIONS OF THE PREVIOUS(y1) AND THE NEW(y2) LOGISTIC MODELS RESPECTIVELY ARE: *y1 = -14.9907 +4.2540DSRI + 5.7864SGI + 0.9798AQI + 14.0498ACCR + 1.2567GMI y2 = -11.3769 + 4.9715DSRI + 3.5622SGI + 11.0802ACCR + 0.5271AQI*

The old model has more AIC (38.1) compared to the new model(51.514). The new model has only four significant variables whereas the previous model had five significant variables.

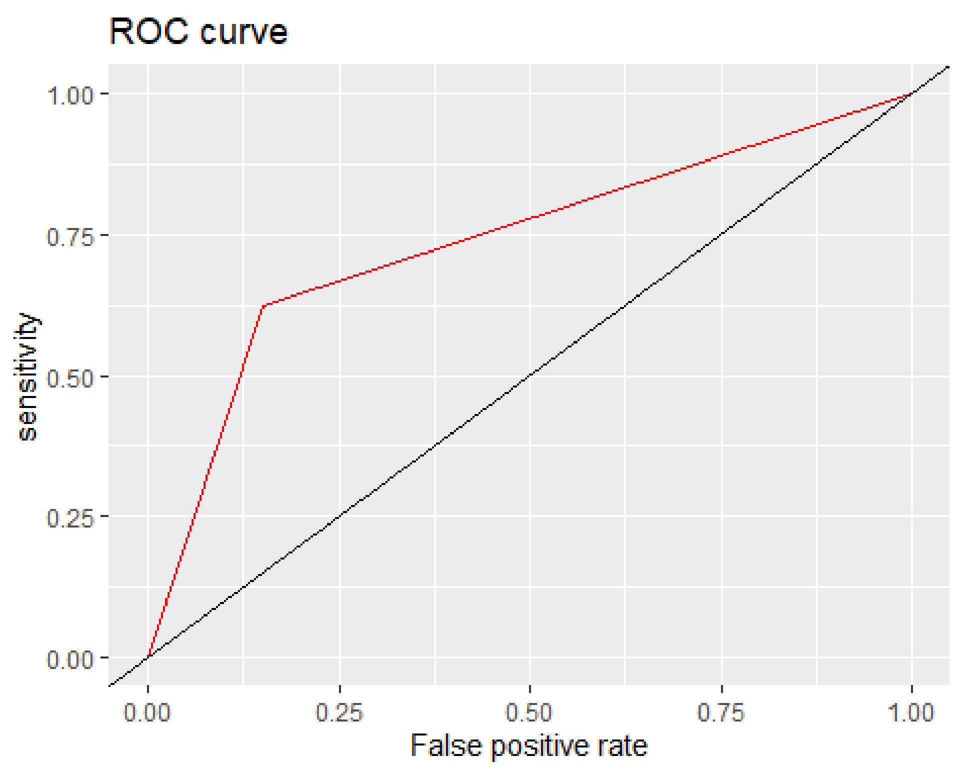
Predicting performance of new logistic model on Test Data.

```
pred_mylogit <- predict(mylogit, test.c, type = "response")
pred_mylogit <- round(pred_mylogit)
pred_mylogit

##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  1
9  20
##   0   1   1   0   1   0   1   1   0   0   0   0   0   0   0   0   0   1
0   0
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  3
9  40
##   1   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   1   0
0   0
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  5
9  60
##   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   1   0   0
0   0
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  7
9  80
##   0   0   0   0   0   1   0   0   0   1   1   0   0   0   0   0   0   0
0   0
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  9
9 100
##   0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   1   0   0
0   0
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 11
9 120
##   0   0   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0
0   0
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 13
9 140
##   0   0   0   0   0   0   0   0   0   1   0   0   1   0   0   0   0   0
0   0
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 15
9 160
##   0   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   0   0
1   0
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 17
9 180
```

```
##   0   0   0   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0 
1   0 
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 19
9 200 
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0 
0   0 
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 21
9 220 
##   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0   1   0   0 
0   0 
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 23
9 240 
##   0   0   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0 
0   0 
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 25
9 260 
##   0   0   0   1   0   0   0   0   0   0   1   0   1   0   0   0   0   0 
1   0 
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 27
9 280 
##   0   0   0   0   1   1   0   1   1   1   0   1   0   0   0   0   0   0 
0   0 
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 29
9 300 
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1 
0   0 
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 31
9 320 
##   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0 
0   0 
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 33
9 340 
##   0   0   0   0   0   1   0   1   1   0   1   1   1   0   1   0   0   0 
0   0 
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 35
9 360 
##   0   0   0   0   0   1   0   0   1   1   0   0   0   0   0   0   0   1 
0   0 
## 361 362 363 364 365 
##   1   0   0   0   1
```

```r
#summary(pred_mylogit)

actual.c <- test.c$Manipulator
pred.c <- ROCR::prediction(pred_mylogit,actual.c)
perf.c <- ROCR::performance(pred.c, 'tpr', 'fpr')
pf.c <- data.frame(perf.c@x.values, perf.c@y.values)
names(pf.c) <- c("fpr", "tpr")
ggplot(data=pf.c,aes(x=fpr,y=tpr))+geom_line(colour='red')+geom_abline(interc
```

```
ept=0,slope=1)+labs(x='False positive rate',y='sensitivity',title='ROC curve'
)
```




```
#plot(perf.c)
```

Calculating area under curve for the ROC plot.

```
auc.c <- performance(pred.c,"auc")
auc.c <- unlist(slot(auc.c,"y.values"))
paste("Area under curve: ", auc.c)

## [1] "Area under curve:  0.73686974789916"
```

The *previous model* had an area under curve: 0.863

The *new model* has an area under curve: 0.737

Thus the previous model is better.

```
ConfMat.c <- table(pred_mylogit,actual.c,dnn=c("Prediction","Actual"))
ConfMat.c

##             Actual
## Prediction    0   1
##            0 303   3
##            1  54   5
```

```
result.c <- confusionMatrix(ConfMat.c)
result.c
```

```
## Confusion Matrix and Statistics
## 
##           Actual
## Prediction   0   1
##          0 303   3
##          1  54   5
## 
##                Accuracy : 0.8438
##                  95% CI : (0.8025, 0.8795)
##     No Information Rate : 0.9781
##     P-Value [Acc > NIR] : 1
## 
##                   Kappa : 0.1151
## 
##  Mcnemar's Test P-Value : 3.528e-11
## 
##             Sensitivity : 0.84874
##             Specificity : 0.62500
##          Pos Pred Value : 0.99020
##          Neg Pred Value : 0.08475
##              Prevalence : 0.97808
##          Detection Rate : 0.83014
##    Detection Prevalence : 0.83836
##       Balanced Accuracy : 0.73687
## 
##        'Positive' Class : 0
## 
```

Previous model Accuracy = 88.33%, AUC = 0.86, AIC = 38.1 *New model* Accuracy = 84.38%, AUC = 0.74, AIC = 51.514 *Clearly the previous logistic regression model is better than the new model*

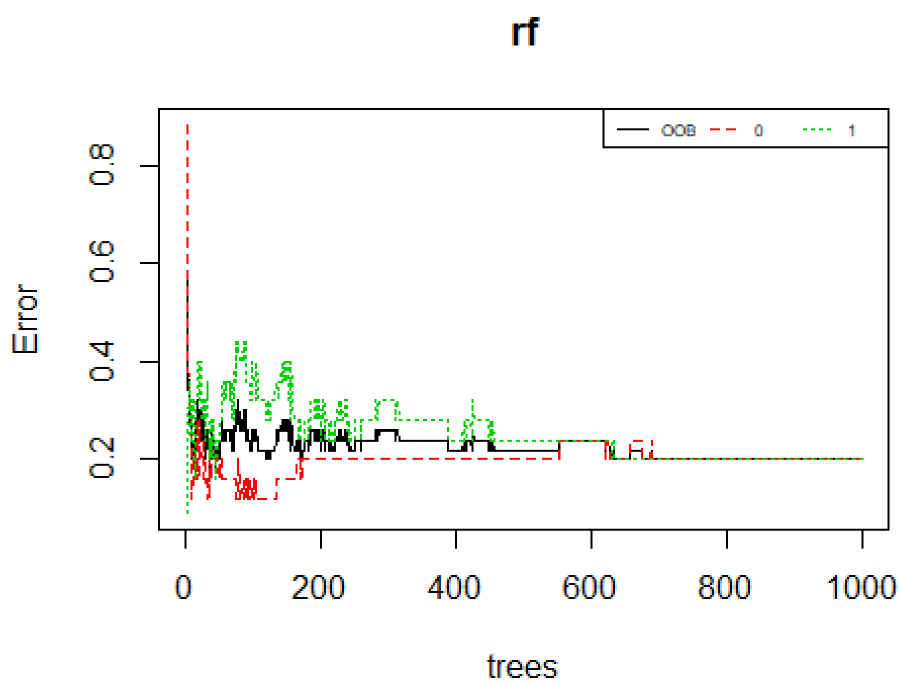
RANDOM FOREST

```
set.seed(123)
rf = randomForest(Mani~., data = under, ntree=1000,proximity=TRUE, replace=TR
UE,sampsize=ceiling(0.65*nrow(under)),importance=TRUE, mtry=sqrt(ncol(under))
)
print(rf)
```

```
## 
## Call:
##  randomForest(formula = Mani ~ ., data = under, ntree = 1000,      proximi
ty = TRUE, replace = TRUE, sampsize = ceiling(0.65 *          nrow(under)), i
mportance = TRUE, mtry = sqrt(ncol(under)))
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 3
```

```
##
##         OOB estimate of  error rate: 20%
## Confusion matrix:
##    0  1 class.error
## 0 20  5         0.2
## 1  5 20         0.2
```

```r
plot(rf)
legend("topright", legend = colnames(rf$err.rate), cex = 0.5,lty = c(1,2,3),
col = c(1,2,3), horiz = T)
```



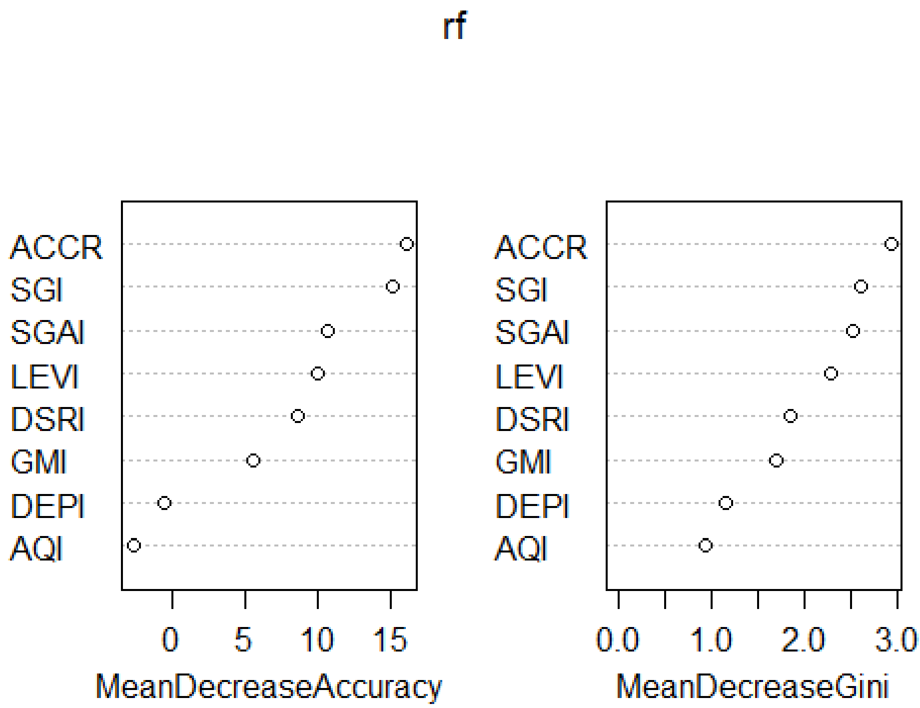

```r
attributes(rf)
```

```
## $names
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"
```

```r
importance(rf)
```

```
##             0        1 MeanDecreaseAccuracy MeanDecreaseGini
## DSRI 5.950208 7.975085            8.5607446        1.8470780
## GMI  3.216007 5.816212            5.5608033        1.6949807
```

```
## AQI   -3.113684 -1.400374           -2.7225561        0.9318873
## SGI   12.394962 12.851584           15.2466650        2.6160268
## DEPI   1.407830 -2.104763           -0.6410312        1.1683092
## SGAI   9.304523  8.780438           10.7691824        2.5244637
## ACCR  12.196358 14.239433           16.1430543        2.9413077
## LEVI  10.925662  6.164514           10.0576577        2.2881889

varImpPlot(rf)
```



Order of importance = ACCR > SGI > SGAI > LEVI> DSRI > GMI > DEPI >AQI

Confusion Matrix

```
maniPred = predict(rf, newdata = test)
actual.rf <- test$Mani
ConfMat.rf <- table(maniPred,actual.rf,dnn=c("Prediction","Actual"))
ConfMat.rf

##             Actual
## Prediction   0  1
##            0 35  4
##            1 11 10

result.rf <- confusionMatrix(ConfMat.rf)
result.rf

## Confusion Matrix and Statistics
##
```

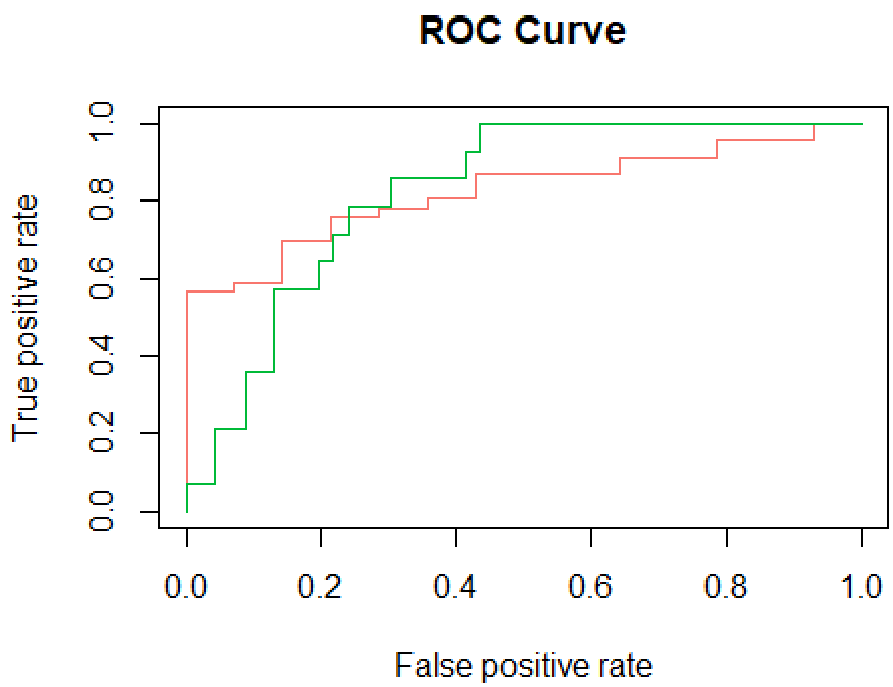
```
##           Actual
## Prediction  0  1
##          0 35  4
##          1 11 10
##
##                Accuracy : 0.75
##                  95% CI : (0.6214, 0.8528)
##     No Information Rate : 0.7667
##     P-Value [Acc > NIR] : 0.6840
##
##                   Kappa : 0.4048
##
##  Mcnemar's Test P-Value : 0.1213
##
##             Sensitivity : 0.7609
##             Specificity : 0.7143
##          Pos Pred Value : 0.8974
##          Neg Pred Value : 0.4762
##              Prevalence : 0.7667
##          Detection Rate : 0.5833
##    Detection Prevalence : 0.6500
##       Balanced Accuracy : 0.7376
##
##        'Positive' Class : 0
##
```

Accuracy: *75%*

Precision: *89.74%*

```
prediction_for_roc_curve <- predict(rf,test[,-9],type="prob")
pretty_colours <- c("#F8766D","#00BA38")
# Specify the different classes
classes <- levels(test$Mani)
# For each class
for (i in 1:2)
{
 # Define which observations belong to class[i]
 true_values <- ifelse(test[,9]==classes[i],1,0)
 # Assess the performance of classifier for class[i]
 pred <- prediction(prediction_for_roc_curve[,i],true_values)
 perf <- performance(pred, "tpr", "fpr")
 if (i==1)
 {
     plot(perf,main="ROC Curve",col=pretty_colours[i])
 }
 else
 {
     plot(perf,main="ROC Curve",col=pretty_colours[i],add=TRUE)
 }
```

```
# Calculate the AUC and print to screen
 auc.perf <- performance(pred, measure = "auc")
 print(auc.perf@y.values)
}
```




```
## [[1]]
## [1] 0.8245342
##
## [[1]]
## [1] 0.8245342
```

AUC: *0.825*

Accuracy: *75%*

Precision: *89.74%*

ADABOOST

```
ada.mod= adaboost(Mani ~ ., data = under, nIter=10)

ada.pred<-matrix(predict( ada.mod,newdata=test))
actual.ada<-test$Mani

ConfMat.ada <- table(ada.pred[[3]],actual.ada,dnn=c("Prediction","Actual"))
ConfMat.ada
```

```
##            Actual
## Prediction  0  1
##          0 30  6
##          1 16  8
```

```
result.ada <- confusionMatrix(ConfMat.ada)
result.ada
```

```
## Confusion Matrix and Statistics
##
##            Actual
## Prediction  0  1
##          0 30  6
##          1 16  8
##
##                Accuracy : 0.6333
##                  95% CI : (0.499, 0.7541)
##     No Information Rate : 0.7667
##     P-Value [Acc > NIR] : 0.99339
##
##                   Kappa : 0.1791
##
##  Mcnemar's Test P-Value : 0.05501
##
##             Sensitivity : 0.6522
##             Specificity : 0.5714
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.3333
##              Prevalence : 0.7667
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.6118
##
##        'Positive' Class : 0
##
```

Accuracy: *63.33%*

Precision: *83.33%*

```
ada.mod$trees
```

```
## $`0`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 0.50 0 (0.50000000 0.50000000)
##   2) ACCR< -0.000179724 28 0.16 0 (0.71428571 0.28571429)
##     4) DSRI< 1.247628 17 0.02 0 (0.94117647 0.05882353) *
```

```
##     5) DSRI>=1.247628 11 0.08 1 (0.36363636 0.63636364) *
##   3) ACCR>=-0.000179724 22 0.10 1 (0.22727273 0.77272727)
##     6) DEPI< 0.9802424 7 0.06 0 (0.57142857 0.42857143) *
##     7) DEPI>=0.9802424 15 0.02 1 (0.06666667 0.93333333) *
##
## $`1`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 0.49057970 0 (0.5188338 0.4811662)
##   2) SGI< 1.167219 33 0.20153630 0 (0.7030908 0.2969092)
##     4) GMI< 1.051857 26 0.09965226 0 (0.8162766 0.1837234) *
##     5) GMI>=1.051857 7 0.03321742 1 (0.2529244 0.7470756) *
##   3) SGI>=1.167219 17 0.04982613 1 (0.1518258 0.8481742) *
##
## $`2`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 0.49986690 0 (0.5002662 0.4997338)
##   2) ACCR< -0.000179724 28 0.17219150 0 (0.6992347 0.3007653)
##     4) DSRI< 1.329535 19 0.04324749 0 (0.8782347 0.1217653) *
##     5) DSRI>=1.329535 9 0.08834906 1 (0.4067180 0.5932820) *
##   3) ACCR>=-0.000179724 22 0.10002660 1 (0.2339652 0.7660348)
##     6) GMI>=1.002935 9 0.06898903 0 (0.5058218 0.4941782) *
##     7) GMI< 1.002935 13 0.02944969 1 (0.1022564 0.8977436) *
##
## $`3`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 0.49795090 0 (0.5040981 0.4959019)
##   2) ACCR< -0.000179724 28 0.17472530 0 (0.6964116 0.3035884)
##     4) GMI< 1.00324 20 0.07352123 0 (0.8220814 0.1779186) *
##     5) GMI>=1.00324 8 0.06060097 1 (0.3764529 0.6235471) *
##   3) ACCR>=-0.000179724 22 0.10451270 1 (0.2458545 0.7541455) *
##
## $`4`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 50 0.46795760 0 (0.56382262 0.43617738)
```

```
##    2) SGI< 1.41559 42 0.34051820 0 (0.63982393 0.36017607)
##      4) AQI< 1.56872 35 0.23632960 0 (0.71145816 0.28854184)
##        8) DSRI< 1.854931 28 0.11270650 0 (0.82716635 0.17283365)
##         16) SGAI>=0.9439092 21 0.03848508 0 (0.92359034 0.07640966) *
##         17) SGAI< 0.9439092 7 0.06528078 1 (0.49999742 0.50000258) *
##        9) DSRI>=1.854931 7 0.03809759 1 (0.25946701 0.74053299) *
##      5) AQI>=1.56872 7 0.01951275 1 (0.17555054 0.82444946) *
##    3) SGI>=1.41559 8 0.00000000 1 (0.00000000 1.00000000) *
##
## $`5`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 50 0.428681000 0 (0.63979385 0.36020615)
##    2) LEVI>=0.6066266 42 0.284636800 0 (0.72506237 0.27493763)
##      4) LEVI< 0.9981138 16 0.008086503 0 (0.98392678 0.01607322) *
##      5) LEVI>=0.9981138 26 0.191803100 1 (0.48033801 0.51966199)
##       10) ACCR< -0.02052736 11 0.078300380 0 (0.74933380 0.25066620) *
##       11) ACCR>=-0.02052736 15 0.016173010 1 (0.09806163 0.90193837) *
##    3) LEVI< 0.6066266 8 0.008086503 1 (0.06961042 0.93038958) *
##
## $`6`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 0.488101900 0 (0.52378278 0.47621722)
##   2) SGAI< 1.768119 43 0.323017900 0 (0.62123685 0.37876315)
##     4) GMI< 1.081076 34 0.154578900 0 (0.76498902 0.23501098)
##       8) SGI< 1.167219 22 0.039611150 0 (0.91855116 0.08144884) *
##       9) SGI>=1.167219 12 0.053828070 1 (0.32932163 0.67067837) *
##     5) GMI>=1.081076 9 0.025394230 1 (0.13652578 0.86347422) *
##   3) SGAI>=1.768119 7 0.006721683 1 (0.04095304 0.95904696) *
##
## $`7`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 50 0.45864190 0 (0.5821540 0.4178460)
##    2) SGAI< 1.768119 43 0.32173280 0 (0.6589502 0.3410498)
##      4) GMI< 0.9979643 20 0.04442581 0 (0.8839410 0.1160590) *
##      5) GMI>=0.9979643 23 0.23998540 1 (0.5053158 0.4946842)
##       10) LEVI< 0.9981138 12 0.08325034 0 (0.7563348 0.2436652) *
##       11) LEVI>=0.9981138 11 0.02106020 1 (0.1135528 0.8864472) *
##    3) SGAI>=1.768119 7 0.01471142 1 (0.1125572 0.8874428) *
```

```
##
## $`8`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 50 0.48619850 1 (0.47241800 0.52758200)
##    2) LEVI>=0.6066266 42 0.30330600 0 (0.59612351 0.40387649)
##      4) LEVI< 0.9981138 16 0.02963920 0 (0.90589668 0.09410332) *
##      5) LEVI>=0.9981138 26 0.17157320 1 (0.37235634 0.62764366)
##       10) DEPI>=1.000954 17 0.10038650 0 (0.57577446 0.42422554) *
##       11) DEPI< 1.000954 9 0.02758969 1 (0.13093826 0.86906174) *
##    3) LEVI< 0.6066266 8 0.01310115 1 (0.05562010 0.94437990) *
##
## $`9`
## n= 50
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 50 0.44394100 1 (0.3892740 0.6107260)
##   2) SGI< 1.41559 42 0.40281310 0 (0.4680527 0.5319473)
##     4) DSRI< 1.247628 23 0.08077084 0 (0.7591974 0.2408026)
##       8) AQI< 1.390057 16 0.00000000 0 (1.0000000 0.0000000) *
##       9) AQI>=1.390057 7 0.05569400 1 (0.3550472 0.6449528) *
##     5) DSRI>=1.247628 19 0.12497560 1 (0.2365394 0.7634606) *
##   3) SGI>=1.41559 8 0.00000000 1 (0.0000000 1.0000000) *
```

```
ada.mod$weights
```

```
##  [1] 0.7581737 0.7490944 0.6040560 0.5800890 0.8240120 1.0420778 0.9704221
##  [8] 0.8163977 0.7902786 0.7559087
```

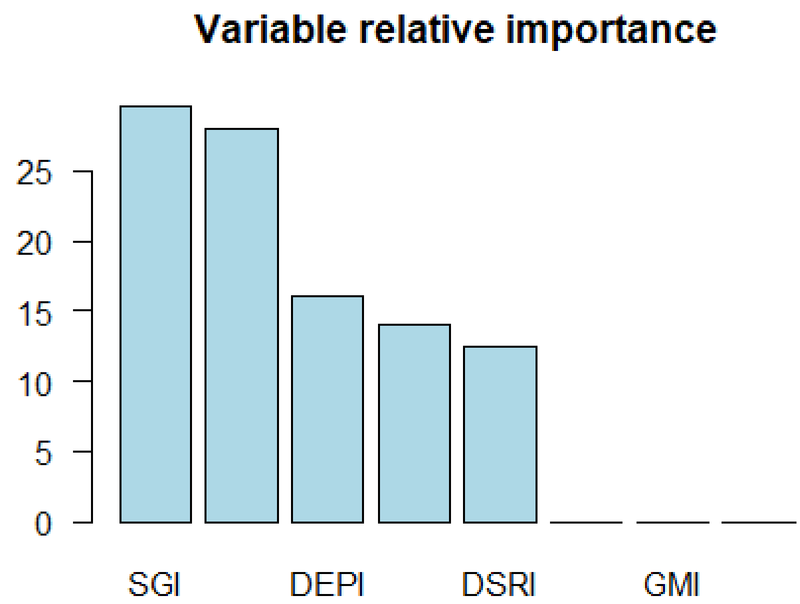
```
ada.mod$prob
```

```
## NULL
```

```
ada.mod$class
```

```
##   A   B
## "0" "1"
```

```
importanceplot(Mani.adaboost)
```



Important Variables in the Adaboost model: *SGAI, ACCR, DEPI, SGI*

FINAL ANALYSIS:

Logistic Regression Model :
Accuracy : 86.67%
Precision : 95.24%
Equation: y = -14.9907 + 4.2540DSRI + 5.7864SGI + 0.9798AQI + 14.0498ACCR + 1.2567GMI
Area under curve: 0.863
Key predictors: ACCR > SGI > DSRI > GMI > AQI

Decision Tree Model (CART) :
Accuracy : 82%
Precision : 80%
Area under curve: 0.721
Key predictors: ACCR > DSRI > DEPI

Random Forest :
Accuracy : 75%
Precision : 89.74%
Area under curve : 0.825
Key predictors: SGI > ACCR > LEVI > SGAI > DSRI > GMI > DEPI > AQI

ADA boost :
Accuracy : 63.33%
Precision : 83.33%
Key predictors: SGAI > ACCR > DEPI > SGI

CONCLUSION :

Out of all the models, Logistic regression model is the best model with highest accuracy, precision and AUC(ROC Curve) For predicting earning manipulators, the following varaibles can be used as predictors : ACCR, SGI, DSRI, DEPI