# Predicting Earning Manipulation

Solution 1:

**a)** Though Beneish model is developed 20 years back, it is based on the M score(using 8 variables) where the companies are classified into likely manipulator and non-likely manipulator which meets our need to identify the earnings manipulators among Indian companies.

**b)** Model might just pick up the patterns in the most popular classes and ignore the least popular ones and create a false sense of performance with high accuracy of the model and with low recall for minority class. So, there might a problem that the model is more biased and overfit majority class of the data. Decision trees can perform well on imbalanced data as the splitting rules can make sure both classes are addressed. So, we can use popular decision tree algorithms like C4.5, C5.0, CART, and Random Forest. We can handle the data by collecting more data of the minority class if possible or resampling the data like under-sampling(randomly deleting data) majority class or over sampling minority class( randomly duplicating data) can help us balancing the data. Also, precision and recall can be considered instead of accuracy.

**c)** *install.packages("dplyr")*

*install.packages("ROSE")*

*library(car)*

*library(dplyr) # data aggregates*

*library(gplots)*

*library(ggplot2)*

*library("readxl")*

*library(ROSE)*

*sample_data <- read_excel(file.choose(), sheet = 5)*

*str(sample_data)*

*sample_data$`C-MANIPULATOR` <- as.factor(sample_data$`C-MANIPULATOR`)*

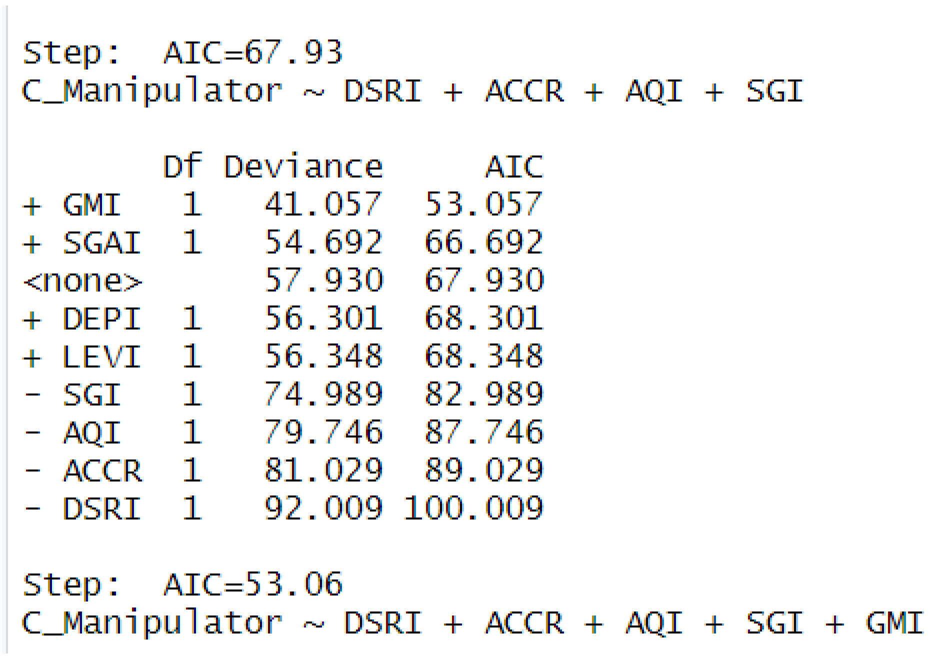
*sample_data$Manipulator <- as.factor(sample_data$Manipulator)*

*colnames(sample_data)[1] <- "Company_ID"*

*colnames(sample_data)[11] <- "C_Manipulator"*

```
  #running for 10 times to take samples to see the significant variables
for (i in 1:10) {

  #sampling into training and testing
  indx[i] <- sample(2, nrow(sample_data), replace = T, prob = c(0.8, 0.2)) #divide into train and test data
  train <- sample_data[indx[i] == 1, ]
  test <- sample_data[indx[i] == 2, ]

  #under-sampling the data from the sample data
  data_balanced_under <- ovun.sample(C_Manipulator~ ., data = train, method = "under", N =
78,seed=1)$data

  full <- glm(C_Manipulator ~ . -Manipulator -Company_ID, data = data_balanced_under, family =
"binomial")
  null <- glm(C_Manipulator ~ 1, data = data_balanced_under,family = "binomial")
  #step wise logistic regression
  step(null,scope = list(lower = null, upper = full), direction = "both")
}
```

When the results from the step function are analysed most of the times it ends up with the following variables:



```
Step:  AIC=67.93
C_Manipulator ~ DSRI + ACCR + AQI + SGI

       Df Deviance     AIC
+ GMI   1   41.057  53.057
+ SGAI  1   54.692  66.692
<none>      57.930  67.930
+ DEPI  1   56.301  68.301
+ LEVI  1   56.348  68.348
- SGI   1   74.989  82.989
- AQI   1   79.746  87.746
- ACCR  1   81.029  89.029
- DSRI  1   92.009 100.009

Step:  AIC=53.06
C_Manipulator ~ DSRI + ACCR + AQI + SGI + GMI
```

#Dividing into train and test data

*indx <- sample(2, nrow(sample_data), replace = T, prob = c(0.8, 0.2))*

*train2 <- sample_data[indx == 1, ]*

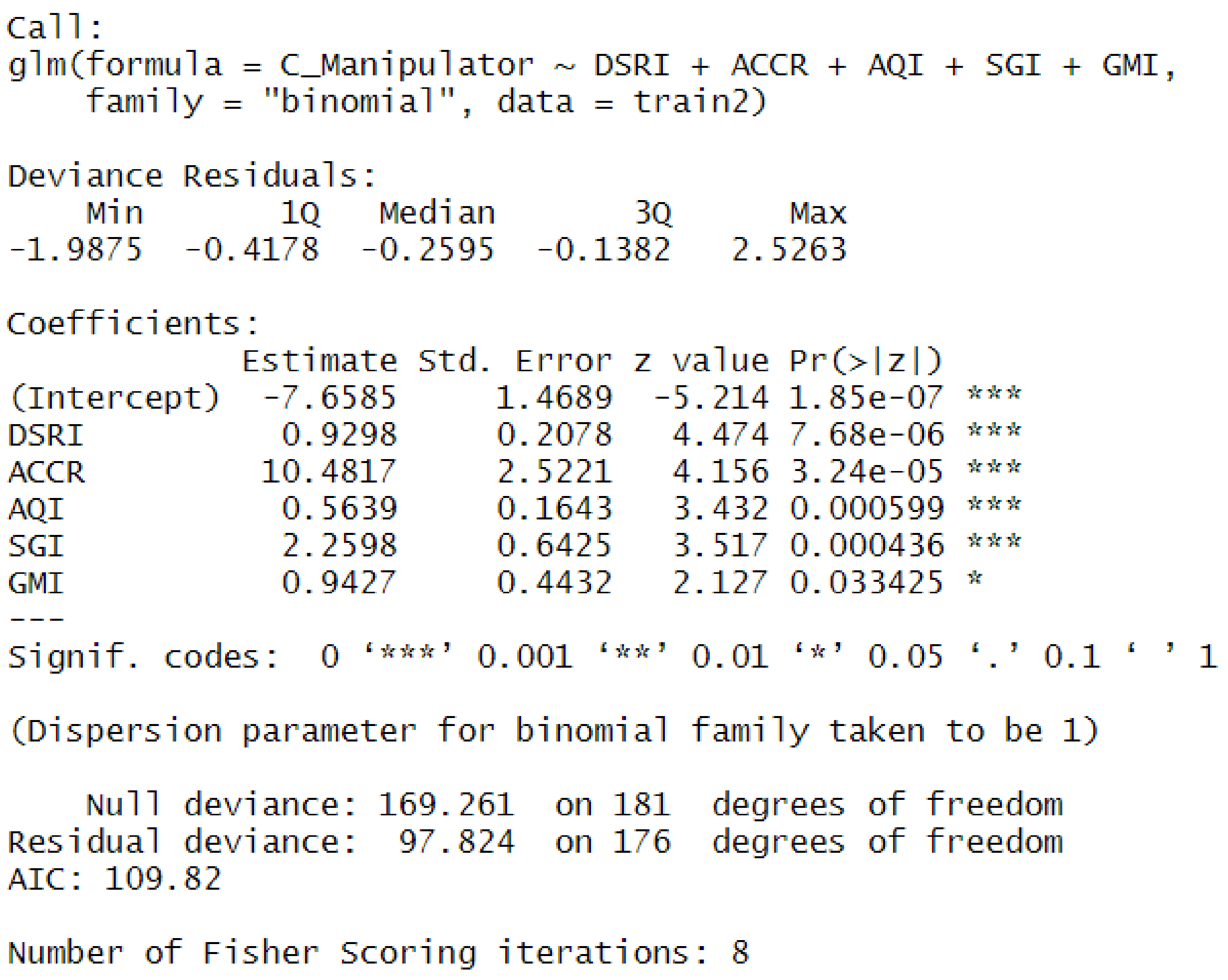
*test2 <- sample_data[indx == 2, ]*

#Building logistic regression using the significant variables

*logistic_model <- glm(C_Manipulator ~ DSRI + ACCR + AQI + SGI + GMI, data = train2, family = "binomial")*

*summary(logistic_model)*

**Summary of logistic model is as follows:**



```
Call:
glm(formula = C_Manipulator ~ DSRI + ACCR + AQI + SGI + GMI,
    family = "binomial", data = train2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9875  -0.4178  -0.2595  -0.1382   2.5263

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -7.6585     1.4689  -5.214 1.85e-07 ***
DSRI          0.9298     0.2078   4.474 7.68e-06 ***
ACCR         10.4817     2.5221   4.156 3.24e-05 ***
AQI           0.5639     0.1643   3.432 0.000599 ***
SGI           2.2598     0.6425   3.517 0.000436 ***
GMI           0.9427     0.4432   2.127 0.033425 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 169.261  on 181  degrees of freedom
Residual deviance:  97.824  on 176  degrees of freedom
AIC: 109.82

Number of Fisher Scoring iterations: 8
```

**Probability formulas for both majority and minority classes are as follows:**

*minority_class_pred<-predict(logistic_model, test2[test2$C_Manipulator==1,], type="response")*

*majority_class_pred<-predict(logistic_model, test2[test2$C_Manipulator==0,], type="response")*

**d) As the data is imbalanced accuracy gives false impression and precision and recall can be taken into consideration**

#predicting from the logistic model

*Pred <- predict(logistic_model,type="response")*

*Pred*

*library(ROCR)*

*ROC_pred <- prediction(Pred,train2$C_Manipulator)*

*perf <- performance(ROC_pred, "tpr", "fpr")* *plot(perf)*

*auc <- performance(ROC_pred, "auc")*

*auc*

*auc <- unlist(slot(auc, "y.values"))*

*auc #91.6*

# to find the best cut-off point in ROC curve

*opt.cut <- function(perf){*

*# mapply function applies the function FUN to all perf@x.values, perf@y.values,perf@alpha.values*

*cut.ind <- mapply(FUN = function(x,y,p){d=(x-0)^2+(y-1)^2 # We compute the distance of all the points from the corner point [1,0]*

*ind<- which(d==min(d)) # We find the index of the point that is closest to the corner*

*c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])},perf@x.values, perf@y.values,perf@alpha.values)*

*}*

**From ROC Curve we got best cut-off point as 0.13**

*print(opt.cut(perf)) #0.13*

**<u>Predicting confusion matrix for cut off points 0.13, 0.25, 0.5, 0.6,0.65,0.7,0.75</u>**

#predicting the probability to find precision and recall

*prob<-predict(logistic_model,test2,type="response")*

*pred_0.13<-ifelse(prob>=0.13,1,0)*

*consufusion<-table(test2$C_Manipulator,pred_0.13)*

*consufusion*

*pred_0.25<-ifelse(prob>=0.25,1,0)*

*consufusion_0.25<-table(test2$C_Manipulator,pred_0.25)*

*consufusion_0.25*

*pred_0.5<-ifelse(prob>=0.5,1,0)*

*consufusion_0.5<-table(test2$C_Manipulator,pred_0.5)*

*consufusion_0.5*

*pred_0.6<-ifelse(prob>=0.6,1,0)*

*consufusion_0.6<-table(test2$C_Manipulator,pred_0.6)*

*consufusion_0.6*

*pred_0.65<-ifelse(prob>=0.65,1,0)*

*consufusion_0.65<-table(test2$C_Manipulator,pred_0.65)*

*consufusion_0.65*

*pred_0.7<-ifelse(prob>=0.7,1,0)*

*consufusion_0.7<-table(test2$C_Manipulator,pred_0.7)*

*consufusion_0.7*

*pred_0.75<-ifelse(prob>=0.75,1,0)*

*consufusion_0.75<-table(test2$C_Manipulator,pred_0.75)*

*consufusion_0.75*

**<u>Finding precision, recall and Youden's index of all the cut-off points mentioned above</u>**

tnr_0.13 <- consufusion[4]/(consufusion[4] + consufusion[3])

tnr_0.13 = 0.5333

tpr_0.13 <- consufusion[1]/(consufusion[1] + consufusion[2])

tpr_0.13 = 0.9697

precision_0.13 <- consufusion[1]/(consufusion[1] + consufusion[3])

precision_0.13 #0.8205

#youdens_index_0.13 = 0.503 (senisitivity + specificity -1)

tnr_0.25 <- consufusion_0.25[4]/(consufusion_0.25[4] + consufusion_0.25[3])

```
tnr_0.25 = 0.6

tpr_0.25 <- consufusion_0.25[1]/(consufusion_0.25[1] + consufusion_0.25[2])

tpr_0.25 = 0.921

precision_0.25=consufusion_0.25[1]/(consufusion_0.25[1] + consufusion_0.25[3])

precision_0.25 #0.897

#youdens_index_0.25 = 0.521 (senisitivity + specificity -1)


tnr_0.5 <- consufusion_0.5[4]/(consufusion_0.5[4] + consufusion_0.5[3])

tnr_0.5 = 0.8

tpr_0.5 <- consufusion_0.5[1]/(consufusion_0.5[1] + consufusion_0.5[2])

tpr_0.5 = 0.8837

precision_0.5 <- consufusion_0.5[1]/(consufusion_0.5[1] + consufusion_0.5[3])

precision_0.5 #0.974

#youdens_index_0.5 = 0.683 (senisitivity + specificity -1)


tnr_0.6 <- consufusion_0.6[4]/(consufusion_0.6[4] + consufusion_0.6[3])

tnr_0.6 = 0.75

tpr_0.6 <- consufusion_0.6[1]/(consufusion_0.6[1] + consufusion_0.6[2])

tpr_0.6 = 0.8636

precision_0.6 <- consufusion_0.6[1]/(consufusion_0.6[1] + consufusion_0.6[3])

precision_0.6 #0.974

#youdens_index_0.6 = 0.613 (senisitivity + specificity -1)


tnr_0.65 <- consufusion_0.65[4]/(consufusion_0.65[4] + consufusion_0.65[3])

tnr_0.65 = 0.75

tpr_0.65 <- consufusion_0.65[1]/(consufusion_0.65[1] + consufusion_0.65[2])

tpr_0.65 = 0.8636

precision_0.65=consufusion_0.65[1]/(consufusion_0.65[1] + consufusion_0.65[3])

precision_0.65 #0.974
```

```
#yonex_index_0.65 = 0.613 (senisitivity + specificity -1)


tnr_0.75 <- consufusion_0.75[4]/(consufusion_0.75[4] + consufusion_0.75[3])

tnr_0.75 = 0.75

tpr_0.75 <- consufusion_0.75[1]/(consufusion_0.75[1] + consufusion_0.75[2])

tpr_0.75 = 0.8636

precision_0.75=consufusion_0.75[1]/(consufusion_0.75[1] + consufusion_0.75[3])

precision_0.75 #0.974

#yonex_index_0.25 = 0.613 (senisitivity + specificity -1)

#From 0.6 the confusion matrix remains same as


#consufusion_0.6

#pred_0.6

    0   1

0   38  1

1    6  3
```

**When compared to the cut-off point given from the ROC Curve the cut-off point 0.5 has higher precision and we can say that it performs better**

**We are looking for more precision here as we do not want our model to classify manipulators as non-manipulators.**

**After trying different cut-off points the precision keeps on increasing till 0.6, where the performance stabilizes afterwards, and we get the same confusion matrix for the 0.6-0.75. Hence, we considered our cut off point as 0.6**

**e)** The following are the output for Youden's index which are calculated from above confusion matrices of different cut-off points:

*#yonex_index_0.13 = 0.503 (senisitivity + specificity -1)*

*yonex_index_0.25 = 0.521 (senisitivity + specificity -1)*

*youdens_index_0.5 = 0.683 (senisitivity + specificity -1)*

*youdens_index_0.6 = 0.613 (senisitivity + specificity -1)*

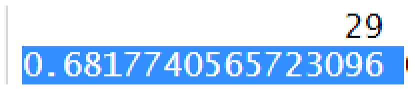
**#Cost based method**

```
cbm_0.13 <- consufusion[2]*(consufusion[2]/consufusion[1]+consufusion[2]) +
consufusion[3]*(consufusion[3]/consufusion[3]+consufusion[4])

cbm_0.13  #64.03

cbm_0.25 <- consufusion_0.25[2]*(consufusion_0.25[2]/consufusion_0.25[1]+consufusion_0.25[2]) +
consufusion_0.25[3]*(consufusion_0.25[3]/consufusion_0.25[3]+consufusion_0.25[4])

cbm_0.25  #64.03

cbm_0.5 <- consufusion_0.5[2]*(consufusion_0.5[2]/consufusion_0.5[1]+consufusion_0.5[2]) +
consufusion_0.5[3]*(consufusion_0.5[3]/consufusion_0.5[3]+consufusion_0.5[4])

cbm_0.5 #30.65

cbm_0.6 <- consufusion_0.6[2]*(consufusion_0.6[2]/consufusion_0.6[1]+consufusion_0.6[2]) +
consufusion_0.6[3]*(consufusion_0.6[3]/consufusion_0.6[3]+consufusion_0.6[4])

cbm_0.6 #40.94
```

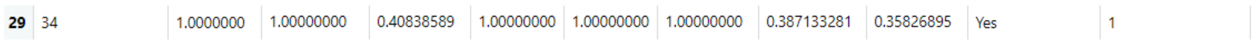
**From the above calculations we got cut-off point from youden's index as 0.683(maximum of all values) and cut-off point from cost-based method as 0.3065(minimum of all values). As we have seen earlier 0.6 and above has the best precision and we can consider Youden's index strategy.**

**f)** The best cut off point we got from models 4 and 5 are 0.68 and the significant variables are DSRI, GMI, AQI, SGI, TATA(ACCR)

So, finding the values of best cut off point, gives the following values:



```
                29
0.6817740565723096
```



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 29 | 34 | 1.0000000 | 1.00000000 | 0.40838589 | 1.00000000 | 1.00000000 | 1.00000000 | 0.387133281 | 0.35826895 | Yes | 1 |

BM-Score = −4.84 + 0.92*DSRI + 0.528*GMI + 0.404*AQI + 0.892*SGI + 4.679*TATA(ACCR)

= -4.84 + 0.92*1 + 0.528*1 + 0.404*0.4083 + 0.892*1 + 4.679*0.387

= -4.84 + 0.92 + 0.528  + 0.164 + 0.892 + 1.81

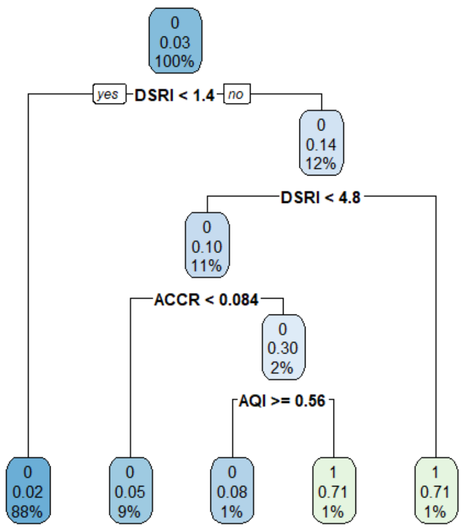
= -0.526

[M-score greater than -2.22 gives the company is likely to be a manipulator]

## g) <u>CART Model :</u>

```
library(rpart)

library(rpart.plot)

index <- sample(2, nrow(India), replace = T, prob = c(0.7,0.3))

TrainData <- India[index == 1, ]

TestData <- India[index == 2, ]

India_rpart <- rpart(C_Manipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = TrainData,
parms = list(split = "information"))

rpart.plot(India_rpart)

print(India_rpart)

summary(iris_rpart)

pred_Train <- predict(India_rpart, TrainData)

pred_Test_class <- predict(India_rpart, newdata = TestData, type = "class")

pred_Test_prob <- predict(India_rpart, newdata = TestData, type = "prob")

table_mat <- table(TestData$C_Manipulator, pred_Test_class)

table_mat

accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)

print(paste('Accuracy for test', accuracy_Test))
```

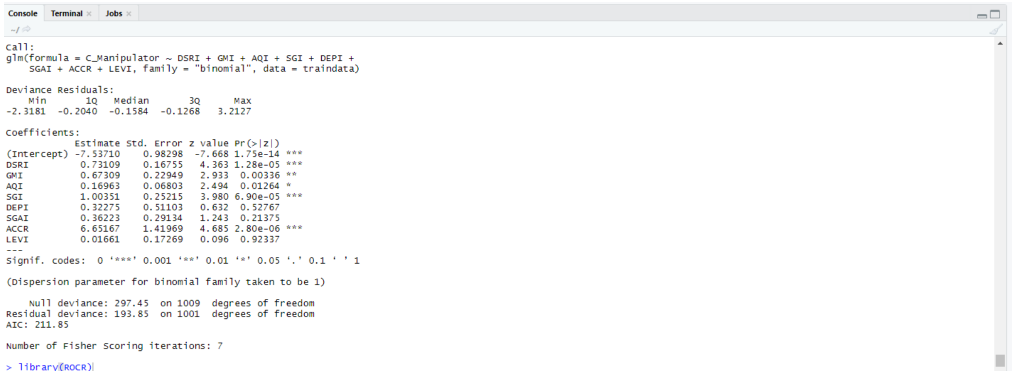
The Decision tree obtained for above model:



The Accuracy we obtained for the model is 96%, the decision rule which we consider is when DSRI is less than 1.4 they are non- manipulators forms pure subset, when No the DSRI is less than 4.8 they are manipulators with 0.71 probability, when ACCR is less than 0.084 they are considered non-manipulators.

**h)**

```
 library(ISLR)

set.seed(256)

str(India)

colnames(Complete_data)[11]<- "C_Manipulator"

Complete_data$C_Manipulator<- as.factor(Complete_data $C_Manipulator)

index <- sample(2, nrow(Complete_data), replace = T, prob = c(0.7, 0.3))

traindata <- Complete_data [indx == 1, ]

testdata <- Complete_data [indx == 2, ]

logitModel <- glm(C_Manipulator ~DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = traindata, family
= "binomial")

summary(logitModel)

Pred <- predict(logitModel, type = "response")

Pred

#To find the Accuracy, Precision, Recall

library(ROCR)

ROC_pred = prediction( Pred, traindata$C_Manipulator)

perf <- performance(ROC_pred, "tpr", "fpr")

plot(perf)

auc <- performance(ROC_pred, "auc")

auc <- unlist(slot(auc, "y.values"))

auc    #0.899

opt.cut <- function(perf){

  # mapply function applies the function FUN to all perf@x.values, perf@y.values,perf@alpha.values

  cut.ind <- mapply(FUN = function(x,y,p){d=(x-0)^2+(y-1)^2 # We compute the distance of all the points
from the corner point [1,0]

  ind<- which(d==min(d)) # We find the index of the point that is closest to the corner

  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])},perf@x.values,
perf@y.values,perf@alpha.values)
}

print(opt.cut(perf))


prob<- predict(logitModel, testdata, type="response")

pred_0.899<-ifelse (prob>=0.027,1,0)

confusion_data<-table(testdata$C_Manipulator, pred_0.899)

accuracy<- (confusion_data[1]+ confusion_data[2])/(confusion_data[1]+confusion_data[2]+
confusion_data[3]+ confusion_data[4])    #82%

precison<- confusion_data[1]/(confusion_data[1]+confusion_data[3]) #82%
```



```
Call:
glm(formula = C_Manipulator ~ DSRI + GMI + AQI + SGI + DEPI +
    SGAI + ACCR + LEVI, family = "binomial", data = traindata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3181  -0.2040  -0.1584  -0.1268   3.2127

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.53710    0.98298  -7.668 1.75e-14 ***
DSRI         0.73109    0.16755   4.363 1.28e-05 ***
GMI          0.67309    0.22949   2.933  0.00336 **
AQI          0.16963    0.06803   2.494  0.01264 *
SGI          1.00351    0.25215   3.980 6.90e-05 ***
DEPI         0.32275    0.51103   0.632  0.52767
SGAI         0.36223    0.29134   1.243  0.21375
ACCR         6.65167    1.41969   4.685 2.80e-06 ***
LEVI         0.01661    0.17269   0.096  0.92337
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 297.45  on 1009  degrees of freedom
Residual deviance: 193.85  on 1001  degrees of freedom
AIC: 211.85

Number of Fisher Scoring iterations: 7

> library(ROCR)
```

The logistic Regression model was constructed for the whole dataset, from the above observation we find out that the p value<0.05 are the significant variables which are DSRI, GMI, AQI, SGI, ACCR . Comparing this model to the previous logistic model discussed, we can conclude the significant variables are the same and the accuracy and precision is almost same. So both are a good model.

The cutoff point for this variable is

| | |
|---|---|
| recall | 0.85294118 |
| specificity | 0.85553279 |
| Cutoff | 0.02780606 |

Accuracy is 0.82

Precision is 0.82

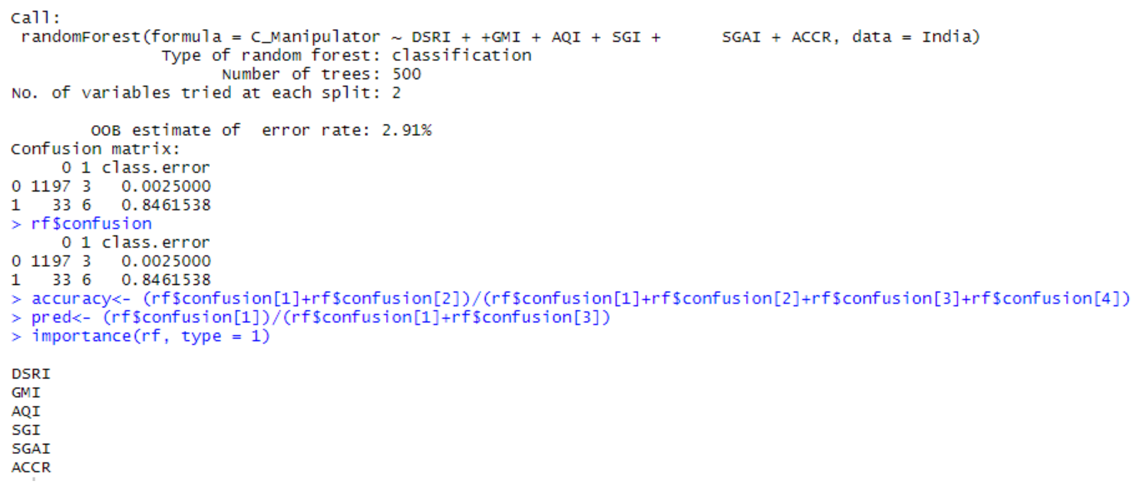
## *i) #Random Forest*

```
 library(randomForest)
rf <- randomForest(C_Manipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI , data =
Complete_data)
plot(rf)
print(rf)
rf$confusion
accuracy<-
(rf$confusion[1]+rf$confusion[2])/(rf$confusion[1]+rf$confusion[2]+rf$confusion[3]+rf$confusion[4])
```

```
pred<- (rf$confusion[1])/(rf$confusion[1]+rf$confusion[3])
importance(rf, type = 1)
rf
rf$proximity
rf$predicted
rf$votes
options(max.print=10000)

library(ROCR)
score <- rf$votes[, 2]
pred <- prediction(score, Complete_data $C_Manipulator)
# Gain Chart
perf <- performance(pred, "tpr", "rpp")
plot(perf)

# ROC Curve
perf <- performance(pred, "tpr", "fpr")
plot(perf)
auc <- performance(pred, "auc")
auc <- unlist(slot(auc, "y.values"))
auc
opt.cut <- function(perf){
  # mapply function applies the function FUN to all perf@x.values, perf@y.values,perf@alpha.values
  cut.ind <- mapply(FUN = function(x,y,p){d=(x-0)^2+(y-1)^2 # We compute the distance of all the points
from the corner point [1,0]
  ind<- which(d==min(d)) # We find the index of the point that is closest to the corner
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])},perf@x.values,
perf@y.values,perf@alpha.values)
}
print(opt.cut(perf))
```

```
Accuracy=99%
Precision= 99%
OOB error rate= 98%
recall      0.89743590
specificity 0.78416667
cutoff      0.02072539
```

Random forests are built on decision trees, and decision trees are sensitive to class imbalance. Each tree is built on a bag, and each bag is a uniform random sample from the data (with replacement) considering there are 1200 non manipulators and 39 manipulators .
Therefore, each tree will be biased in the same direction and magnitude (on average) by class Imbalance, as the accuracy and precision being 99% suggests that there is high bias, hence not a good model in comparison to Logistic model and CART.



```
Call:
 randomForest(formula = C_Manipulator ~ DSRI + +GMI + AQI + SGI +      SGAI + ACCR, data = India)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 2.91%
Confusion matrix:
     0 1 class.error
0 1197 3   0.0025000
1   33 6   0.8461538
> rf$confusion
     0 1 class.error
0 1197 3   0.0025000
1   33 6   0.8461538
> accuracy<- (rf$confusion[1]+rf$confusion[2])/(rf$confusion[1]+rf$confusion[2]+rf$confusion[3]+rf$confusion[4])
> pred<- (rf$confusion[1])/(rf$confusion[1]+rf$confusion[3])
> importance(rf, type = 1)

DSRI
GMI
AQI
SGI
SGAI
ACCR
```

***#ada boost***

*install.packages("adabag")*

*library("adabag")*

*names(complete_data)[1] <- "Company_ID"*

*names(complete_data)[11] <- "C_Manipulator"*

*complete_data$C_Manipulator <- as.factor(complete_data$C_Manipulator)*

*#train and test*

*indx <- sample(2, nrow(complete_data), replace = T, prob = c(0.7, 0.3)) #divide into training and test data*

*train <- complete_data[indx == 1, ]*

*test <- complete_data[indx == 2, ]*

*mani_0 <- train[which(train$C_Manipulator==0),]*

*mani_1 <- train[which(train$C_Manipulator==1),]*

*count(mani_0)*

*count(mani_1) #undersample train data with N=58*

*complete_UB <- ovun.sample(C_Manipulator~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = train, method = "under",N = 58,seed=1)*

*undersampled_data_complete <- complete_UB$data*

*table(undersampled_data_complete$C_Manipulator)*

*# We can use boosting() function to construct a boosting model*

*mod.adaboost <- boosting(C_Manipulator ~ ., data = train, mfinal = 10)*

*# mfinal is an integer that indicates the number of weak learner includes in the boosted model*

*# coeflearn determines the formula for computing voting powers*

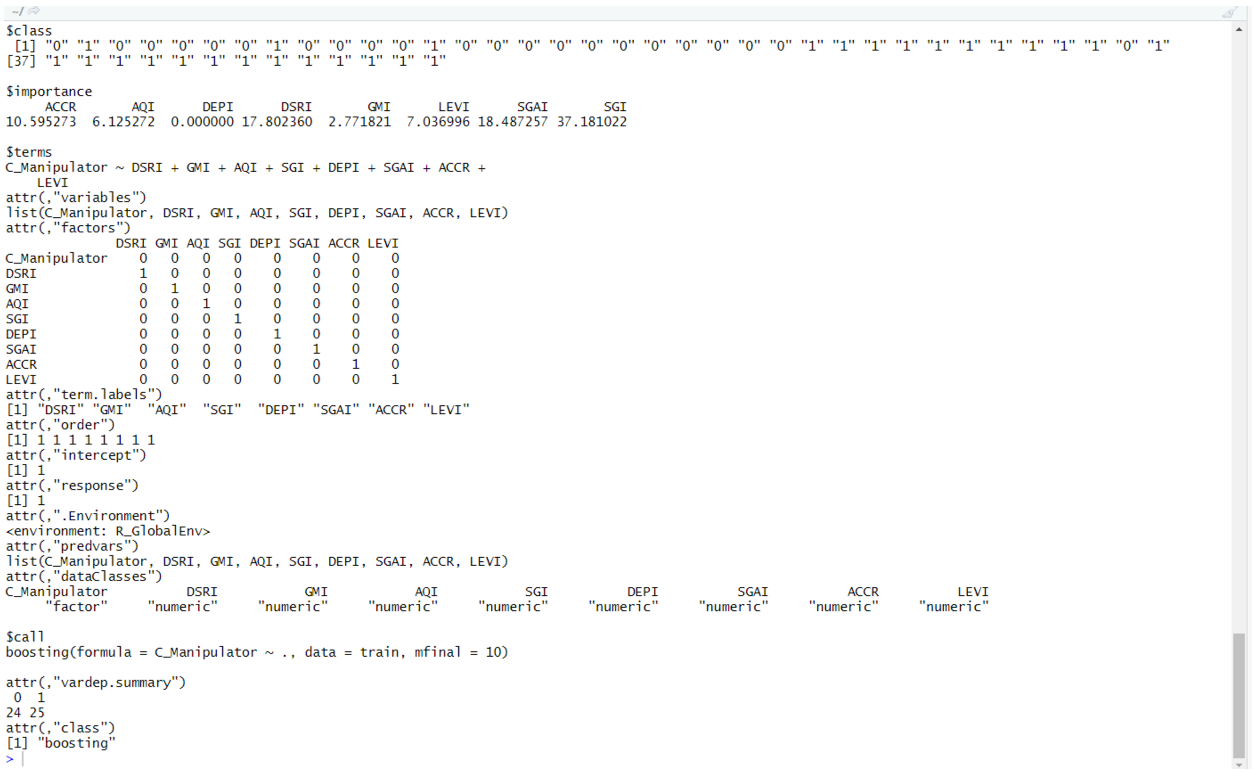
*# You can use coeflearn  = 'Breiman' or 'Freund' or 'Zhu'*

*# control is the same as rpart.control for controling the weak learners.*
*# You need to be careful with overfitting if you use more complicated tree as weak-learners*

*mod.adaboost*
*#importance returns important variables*
*mod.adaboost$importance*

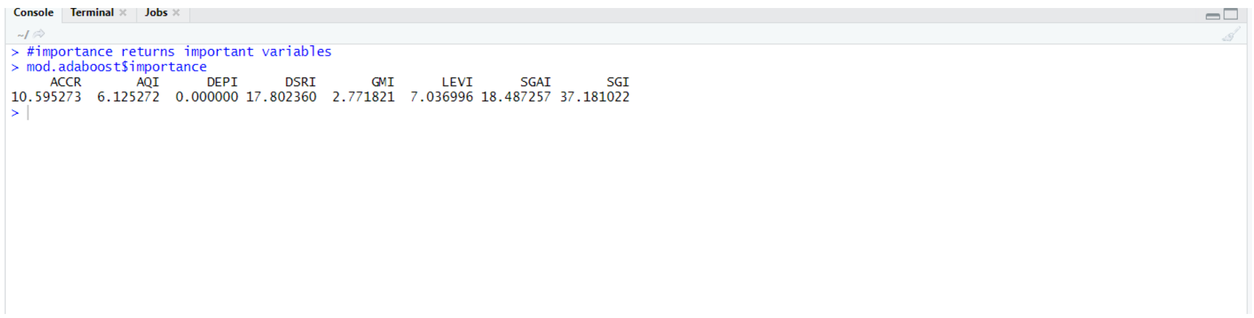
*# To get predicted class on test data we can use predict function*
*pred <- predict(mod.adaboost,newdata = test)*
*table(mod.adaboost$class, train$C_Manipulator, dnn = c("Predicted Class", "Observed Class"))*

**The results of adaboost are as follows:**



```
$class
 [1] "0" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "0" "1"
[37] "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1"

$importance
     ACCR       AQI      DEPI      DSRI       GMI      LEVI      SGAI       SGI 
10.595273  6.125272  0.000000 17.802360  2.771821  7.036996 18.487257 37.181022 

$terms
C_Manipulator ~ DSRI + GMI + AQI + SGI + DEPI + SGAI + ACCR + 
    LEVI
attr(,"variables")
list(C_Manipulator, DSRI, GMI, AQI, SGI, DEPI, SGAI, ACCR, LEVI)
attr(,"factors")
              DSRI GMI AQI SGI DEPI SGAI ACCR LEVI
C_Manipulator    0   0   0   0    0    0    0    0
DSRI             1   0   0   0    0    0    0    0
GMI              0   1   0   0    0    0    0    0
AQI              0   0   1   0    0    0    0    0
SGI              0   0   0   1    0    0    0    0
DEPI             0   0   0   0    1    0    0    0
SGAI             0   0   0   0    0    1    0    0
ACCR             0   0   0   0    0    0    1    0
LEVI             0   0   0   0    0    0    0    1
attr(,"term.labels")
[1] "DSRI" "GMI"  "AQI"  "SGI"  "DEPI" "SGAI" "ACCR" "LEVI"
attr(,"order")
[1] 1 1 1 1 1 1 1 1
attr(,"intercept")
[1] 1
attr(,"response")
[1] 1
attr(,".Environment")
<environment: R_GlobalEnv>
attr(,"predvars")
list(C_Manipulator, DSRI, GMI, AQI, SGI, DEPI, SGAI, ACCR, LEVI)
attr(,"dataClasses")
C_Manipulator          DSRI           GMI           AQI           SGI          DEPI          SGAI          ACCR          LEVI 
     "factor"     "numeric"     "numeric"     "numeric"     "numeric"     "numeric"     "numeric"     "numeric"     "numeric" 

$call
boosting(formula = C_Manipulator ~ ., data = train, mfinal = 10)

attr(,"vardep.summary")
 0  1 
24 25 
attr(,"class")
[1] "boosting"
> 
```

**The important variables from ada boost are as follows:**



```
> #importance returns important variables
> mod.adaboost$importance
     ACCR       AQI      DEPI      DSRI       GMI      LEVI      SGAI       SGI 
10.595273  6.125272  0.000000 17.802360  2.771821  7.036996 18.487257 37.181022 
> 
```

**Accuracy= 24+25/ 24+25+3+2  = 91%**

**j)** Comparing the different model from C to I, based on accuracy and precision. We can conclude that the logistic regression model for the sampled data and complete data gives us the best Accuracy – Precision. Hence, the important variables to be considered to give the best result is DSRI, GMI, AQI, SGI, ACCR.