(a) Do you think the Beneish model developed in 1999 will still be relevant to Indian data?

As per Beneish Model , the M-Score should be less than -2.22 for a company unlikely to be a manipulator. If M-Score is greater than -2.22 the company is likely to be a manipulator. Here, in our model which we have created using the optimum cut-off point of 0.32; we came across with certain False positives and false negatives. However, considering the M-score we are getting of -1.98, we can say the model is performing good. As the Beneish model indicated this to be near -2.2. Thus we can say, the Beneish model developed in 1999 will be relevant to Indian data with certain limitations.

(b) The number of manipulators is usually much less than non-manipulators (in the accompanying spreadsheet, the percentage of manipulators is less than 4% in the complete data). What kind of modeling problems can one expect when cases in one class are much lower than the other class in a binary classification problem? In other words, which models are robust to unbalanced data? How can one handle unbalanced problems?

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally. For example, you may have a 2-class (binary) classification problem with 100 instances (rows). A total of 80 instances are labeled with Class-1 and the remaining 20 instances are labeled with Class-2. This is an imbalanced dataset and the ratio of Class-1 to Class-2 instances is 80:20 or more concisely 4:1. As you might have guessed, the reason we get 80% accuracy on an imbalanced data (with 80% of the instances in Class-1) is because our models look at the data and cleverly decide that the best thing to do is to always predict “Class-1” and achieve high accuracy.
This type of modeling problem is incorporated with unbalanced data.

All models are sensitive towards the dataset. If the data is unbalanced, none of the models will perform well irrespective of having high accuracy.
To handle unbalanced data we have 3 methods:

1. Under Sampling: Making the dataset balanced by reducing the majority class instances corresponding to the minority class, but here we may miss out on certain important data.
2. Over Sampling: Making the dataset balanced by increasing the minority class instances by random duplication corresponding to the majority class. This may overfit at times.
3. SMOTE: It’s a type of oversampling, but here more samples are taken with respect to nearest neighbours. This is the most optimum method.

c) Develop a stepwise logistic regression model that can be used by MCA Technologies Private Limited for predicting probability of earnings manipulation. Write down the probability formulas for both classes using your logistic regression results.

```
library(ROCR)
library(readr)
SampleData <- read_csv("SampleData.csv")
#Removing First ID Column, C- Manipulator
SampleData <-SampleData[,-c(1,ncol(SampleData))]
```

```
SampleData$Manipulator <- as.factor(SampleData$Manipulator)
```

#Subsetting Manipulators and Non- Manipulators

```
Manipulators <- which(SampleData$Manipulator == "Yes")
Non_Manipulators <- which(SampleData$Manipulator == "No")
```

#Under Sampling the data

```
nsamp <- min(length(Manipulators), length(Non_Manipulators))
Manipulators_Data <- sample(Manipulators, nsamp)
Non_Manipulators_Data <- sample(Non_Manipulators, nsamp)
new_data <- SampleData[c(Manipulators_Data, Non_Manipulators_Data), ]
```

#Test and Train

```
index <- sample(2, nrow(new_data), replace = T, prob = c(0.75, 0.25))
TrainData <- new_data[index == 1, ]
TestData <- new_data[index == 2, ]
```

#Stepwise Logistic Regression

```
null <- glm(Manipulator ~ 1, data= TrainData,family="binomial")
full <- glm(Manipulator ~ ., data= TrainData,family="binomial")
logitModel <- step(null, scope = list(lower = null, upper = full),
direction = "both")
```

Model 1 :

```
> summary(logitModel)

Call:
glm(formula = Manipulator ~ SGI + DSRI + AQI + GMI + ACCR, family = "binomial",
    data = TrainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2367  -0.4433  -0.1209   0.3385   2.2174

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -14.3854     4.7746  -3.013 0.002587 **
SGI           3.8127     1.4123   2.700 0.006941 **
DSRI          2.1938     1.1979   1.831 0.067049 .
AQI           1.1687     0.3493   3.346 0.000819 ***
GMI           4.9481     1.6881   2.931 0.003377 **
ACCR         19.3003     6.6975   2.882 0.003955 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 75.791  on 54  degrees of freedom
Residual deviance: 33.713  on 49  degrees of freedom
AIC: 45.713

Number of Fisher Scoring iterations: 10
```

Model 2 :

```
> summary(logitModel)

Call:
glm(formula = Manipulator ~ DSRI + ACCR + AQI + SGI + GMI, family = "binomial",
    data = TrainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3746  -0.5200   0.0000   0.6104   1.8098

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -9.0488     2.6435  -3.423 0.000619 ***
DSRI          2.5715     0.9220   2.789 0.005286 **
ACCR         13.4386     4.2157   3.188 0.001434 **
AQI           0.5562     0.1974   2.818 0.004827 **
SGI           2.2420     0.9093   2.466 0.013675 *
GMI           1.8173     0.8166   2.225 0.026058 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 87.321  on 62  degrees of freedom
Residual deviance: 46.186  on 57  degrees of freedom
AIC: 58.186

Number of Fisher Scoring iterations: 8
```

Model 3 :

```
> summary(logitModel)

Call:
glm(formula = Manipulator ~ DSRI + SGI + AQI + ACCR + GMI + LEVI +
    DEPI, family = "binomial", data = TrainData)

Deviance Residuals:
    Min       1Q    Median       3Q      Max
-2.7696  -0.2299   0.0000   0.2797   1.7528

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -21.7913     7.3146  -2.979  0.00289 **
DSRI          5.4743     2.1325   2.567  0.01026 *
SGI           7.5022     2.6523   2.829  0.00467 **
AQI           1.3204     0.4401   3.000  0.00270 **
ACCR         10.3924     3.7264   2.789  0.00529 **
GMI           2.6044     1.4244   1.828  0.06749 .
LEVI         -2.4078     1.1048  -2.179  0.02930 *
DEPI          3.2240     2.0646   1.562  0.11839
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 84.416  on 60  degrees of freedom
Residual deviance: 29.544  on 53  degrees of freedom
AIC: 45.544

Number of Fisher Scoring iterations: 9
```

	AIC
Model 1	45.713
Model 2	58.186
Model 3	45.544

Significant Factors at 95% Confidence

	DSRI	GMI	AQI	SGI	DEPI	SGAI	ACCR	LEVI
Model 1		✓	✓	✓			✓	
Model 2	✓	✓	✓	✓			✓	
Model 3	✓		✓	✓			✓	✓

Model 3 has lowest AIC. The Significant factors are DSRI,GMI,AQI,SGI,ACCR and these factors have appeared at least in 2 of the 3 models created.

Probability Equation for Manipulator Class

$$P(Y == 1) = \frac{e^{-21.7913+5.7443(DSRI)+7.5022(SGI)+1.3204(AQI)+10.3924(ACCR)-2.4075(LEVI)}}{1 + e^{-21.7913+5.7443(DSRI)+7.5022(SGI)+1.3204(AQI)+10.3924(ACCR)-2.4075(LEVI)}}$$

Probability Equation for Non-Manipulator Class

$$P(Y == 0) = \frac{1}{1 + e^{-21.7913+5.7443(DSRI)+7.5022(SGI)+1.3204(AQI)+10.3924(ACCR)-2.4075(LEVI)}}$$

d)Comment on the model developed; how do you evaluate your model? Do you think the cut-off probability of 0.5 results in a good model? Try different cut-off points and see how the performance of your model change.

The Logistic Regression model developed has a lower AIC. Accuracy, Sensitivity and Specificity can be used to evaluate the model at different cut off points.

```
pred_prob <- predict(logitModel,newdata = TestData, type = "response")
```

#0.5 Cut-Off

```
pred_0.5 <- as.factor(ifelse(pred_prob > 0.5,"Yes","No"))

confusionMatrix(pred_0.5,TestData$Manipulator)

#Accuracy : 0.5294
#Sensitivity : 0.6000
#Specificity : 0.4286
```

#0.7 Cut-Off

```
pred_0.7 <- as.factor(ifelse(pred_prob > 0.7,"Yes","No"))

confusionMatrix(pred_0.7,TestData$Manipulator)

#Accuracy : 0.5882
#Sensitivity : 0.7000
#Specificity : 0.4286
```

#0.3 Cut-Off

```
pred_0.3 <- as.factor(ifelse(pred_prob > 0.3,"Yes","No"))

confusionMatrix(pred_0.3,TestData$Manipulator)

#Accuracy : 0.4706
#Sensitivity : 0.5000
#Specificity : 0.4286
```

Differing the cut off point changes the accuracy and sensitivity, specificity remains the same.

#Finding the optimum cut-off point

```
pred <- prediction( predictions = pred_prob,
TestData$Manipulator,label.ordering = c("Yes","No"))

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


opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x-0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)}
```

```
print(opt.cut(perf, pred))

#[,1]
#sensitivity 0.5000000
#specificity 0.5714286
#cutoff      0.3204989
```

e) What should be the strategy adopted by MCA Technology Solutions to deploy the logistic regression model developed? To answer this question you two different strategies to find the best cut-off point.

(1) Youden's index

```
y.index = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d =  y +(1-x)-1 #max {sensitivity(p) + specificity(p) -1}
    ind = which(d == max(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)}


print(y.index(perf, pred))

            [,1]
sensitivity 0.5000000
specificity 0.5714286
cutoff      0.3204989
```

(2) Cost-based method

```
#Giving a higher penalty for classifying  a  manipulators  (Y= 1)  as  non-manipulator(Y= 0) and

#lower penalty for classifying  a  non-manipulators  (Y= 0)  as manipulator (Y= 1)

cost.index = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p,tp,tn,fn,fp){
    p = 1*(fn/(tp+fn))+0.5*(fp/(tn+fp)) #min_p { p1 × P10 + p2 × P01 }
    ind = which(p == min(p))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values,
pred@cutoffs,pred@tp,pred@tn,pred@fn,pred@fp)}

print(cost.index(perf, pred))

#              [,1]
#sensitivity  1.0
#specificity  0.0
#cutoff       0.5
```

#Giving a equal penalty for classifying a manipulators (Y= 1) as non-manipulator(Y= 0) and

for classifying a non-manipulators (Y= 0) as manipulator (Y= 1)

```
cost.index = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p,tp,tn,fn,fp){
    p = 0.5*(fn/(tp+fn))+0.5*(fp/(tn+fp))
    ind = which(p == min(p))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values,
pred@cutoffs,pred@tp,pred@tn,pred@fn,pred@fp)}
print(cost.index(perf, pred))
                 [,1]
sensitivity 0.5000000
specificity 0.5714286
cutoff      0.4642857
```

f) Based on the models developed in questions 4 and 5, suggest a M-score (Manipulator score)that can be used by regulators to identify potential manipulators.

M-Score = −4.84 + 0.92 × DSRI + 0.528 × GMI + 0.404 × AQI + 0.892 × SGI + 0.115 × DEPI −0.172 × SGAI + 4.679 × ACCR − 0.327 × LEVI

```
pred_num <- lapply(pred_prob, round, 7)
which(pred_num ==0.3204989 )
evalq((-4.84 + (0.92 * DSRI) + (0.528 * GMI) + (0.404 * AQI) + (0.892 *
SGI)
       + (0.115 * DEPI) -(0.172 * SGAI) + (4.679 * ACCR) - (0.327 *
LEVI)),TestData[12,])
  # -1.982544
```

If M-Score is less than -1.982544, the company is unlikely to be a manipulator.
If M-Score is greater than -1.982544 , the company is likely to be a manipulator.

(g) Develop classication and regression tree (CART) model. What insights do you obtain from the CART model? Discuss the best decision rules that can be used. Explain your choices.

```
library(readr)
```

```r
library(ROCR)

library(DMwR)

library('smotefamily')
```

Smote : Synthetic Minority Oversampling Technique To Handle Class Imbalancy In Binary Classification

```r
IMB579_XLS_ENG <-IMB579_XLS_ENG[,c(2:10)]

IMB579_XLS_ENG<- as.data.frame(IMB579_XLS_ENG)


SampleData <- SMOTE(IMB579_XLS_ENG[,-9],IMB579_XLS_ENG[,9])

nrow(SampleData$data) #2370 rows

SampleData <- SampleData$data

colnames(SampleData)[9] <- "Manipulater"

SampleData$Manipulater <- as.factor(SampleData$Manipulater)

str(SampleData)
```

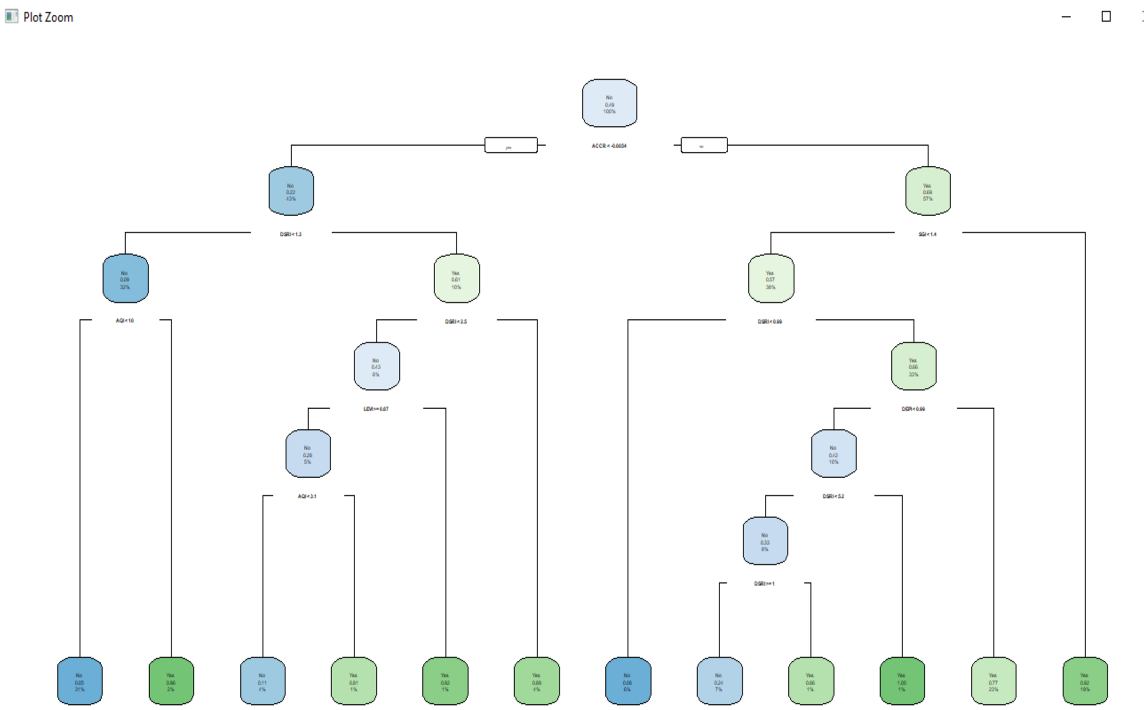
#CART Model

```r
set.seed(1234)

index <- sample(2, nrow(SampleData), replace = T, prob = c(0.75,0.25))

TrainData <- SampleData[index == 1, ]

TestData <- SampleData[index == 2, ]


library(rpart)

man_rpart <- rpart(Manipulater ~ ., data = TrainData, parms =
list(split = "gini"))

printcp(man_rpart)

opt <- which.min(man_rpart$cptable[,"xerror"])

opt

cp1 <- man_rpart$cptable[opt, "CP"]

cp1
```

#0.01

We can use the rpart.plot to plot the decision tree

```r
library(rpart.plot)

rpart.plot(man_rpart)
```



Print the decision tree and take a look at the summary of rpart

```
> print(man_rpart)
n= 1799

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 1799 878 No (0.51195108 0.48804892)
   2) ACCR< -0.005415092 766 168 No (0.78067885 0.21932115)
     4) DSRI< 1.341391 582  55 No (0.90549828 0.09450172)
       8) AQI< 10.22395 554  28 No (0.94945848 0.05054152) *
       9) AQI>=10.22395 28   1 Yes (0.03571429 0.96428571) *
     5) DSRI>=1.341391 184  71 Yes (0.38586957 0.61413043)
      10) DSRI< 2.513104 111  48 No (0.56756757 0.43243243)
        20) LEVI>=0.8668999 85  24 No (0.71764706 0.28235294)
          40) AQI< 3.07305 64   7 No (0.89062500 0.10937500) *
          41) AQI>=3.07305 21   4 Yes (0.19047619 0.80952381) *
        21) LEVI< 0.8668999 26   2 Yes (0.07692308 0.92307692) *
      11) DSRI>=2.513104 73   8 Yes (0.10958904 0.89041096) *
   3) ACCR>=-0.005415092 1033 323 Yes (0.31268151 0.68731849)
     6) SGI< 1.424486 692 295 Yes (0.42630058 0.57369942)
      12) DSRI< 0.9940763 106   8 No (0.92452830 0.07547170) *
      13) DSRI>=0.9940763 586 197 Yes (0.33617747 0.66382253)
        26) DEPI< 0.9943299 174  73 No (0.58045977 0.41954023)
          52) DSRI< 5.189696 150  49 No (0.67333333 0.32666667)
           104) DSRI>=1.009087 129  31 No (0.75968992 0.24031008) *
           105) DSRI< 1.009087 21   3 Yes (0.14285714 0.85714286) *
          53) DSRI>=5.189696 24   0 Yes (0.00000000 1.00000000) *
        27) DEPI>=0.9943299 412  96 Yes (0.23300971 0.76699029) *
     7) SGI>=1.424486 341  28 Yes (0.08211144 0.91788856) *
> |
```

```
> summary(man_rpart)
Call:
rpart(formula = Manipulater ~ ., data = TrainData, parms = list(split = "gini"))
  n= 1799

           CP nsplit rel error    xerror       xstd
1  0.44077449      0 1.0000000 1.0000000 0.02414721
2  0.05125285      1 0.5592255 0.5717540 0.02166765
3  0.04783599      3 0.4567198 0.5239180 0.02107459
4  0.03189066      4 0.4088838 0.4271071 0.01962278
5  0.02961276      5 0.3769932 0.4134396 0.01938744
6  0.02733485      6 0.3473804 0.3952164 0.01906066
7  0.02107062      7 0.3200456 0.3633257 0.01845084
8  0.01708428      9 0.2779043 0.3394077 0.01795926
9  0.01480638     10 0.2608200 0.3200456 0.01753795
10 0.01000000     11 0.2460137 0.3132118 0.01738396

Variable importance
DSRI ACCR  SGI  AQI LEVI DEPI SGAI  GMI 
  26   25   16    9    7    7    6    5 
```

```
pred_Test_class<- predict(man_rpart, newdata = TestData, type =
"class")
(mean(pred_Test_class == TestData$Manipulater))*100
```

#87.39054%

```
confusionMatrix(pred_Test_class, TestData$Manipulater, positive =
"Yes")
```

#pred_Test_class **No** **Yes**

No 238 31

Yes 41 261

###Accuracy : 87.39054%
###Sensitivity : 0.8938
###Specificity : 0.8530

Two Best Decision Rules:

1) Class=”Yes” >>>{ACCR >= (-0.0054), SGI < 1.4, DSRI >= 0.99, DEPI < 0.99, DSRI >= 5.2} with 100% confidence and 1% support

```
Node number 53: 24 observations
  predicted class=Yes  expected loss=0  P(node) =0.01334074
    class counts:     0    24
   probabilities: 0.000 1.000
```

2) Class=”Yes” >>>{ACCR < (-0.0054), DSRI < 1.3, AQI >= 10} with 96.4% confidence and 2% support.

```
Node number 9: 28 observations
  predicted class=Yes  expected loss=0.03571429  P(node) =0.0155642
    class counts:     1    27
   probabilities: 0.036 0.964
```

(h) Develop a logistic regression model using the complete data set (1200 non-manipulators and 39 manipulators), compare the results with the previous logistic regression model.

```
set.seed(1234)

null <- glm(Manipulater ~ 1, data= TrainData,family="binomial") # only includes one variable

full <- glm(Manipulater ~ ., data= TrainData,family="binomial") # includes all the variables

logitModel <- step(null, scope = list(lower = null, upper = full), direction = "both")

summary(logitModel)


mylogit = glm( Manipulater ~ ACCR + DSRI + SGI + AQI + GMI + LEVI +
                 DEPI, family = "binomial", data = TrainData)
```

```
> summary(mylogit)

Call:
glm(formula = Manipulater ~ ACCR + DSRI + SGI + AQI + GMI + LEVI + 
    DEPI, family = "binomial", data = TrainData)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-4.6486  -0.5658  -0.0765   0.6251   1.7624  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -8.35129    0.55113 -15.153  < 2e-16 ***
ACCR         9.76364    0.66391  14.706  < 2e-16 ***
DSRI         1.88113    0.15056  12.495  < 2e-16 ***
SGI          3.31377    0.25893  12.798  < 2e-16 ***
AQI          0.61310    0.04767  12.863  < 2e-16 ***
GMI          1.13698    0.18539   6.133 8.62e-10 ***
LEVI        -0.61250    0.13519  -4.531 5.88e-06 ***
DEPI         0.19353    0.19992   0.968    0.333    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2492.9  on 1798  degrees of freedom
Residual deviance: 1401.2  on 1791  degrees of freedom
AIC: 1417.2

Number of Fisher Scoring iterations: 8
```

```
pred_prob <- predict(mylogit,newdata = TestData, type = "response")
table(pred_prob > 0.5 ,TestData$Manipulater )
```

#Getting Probability cut off point using ROC curve

```
pred <- prediction( predictions = pred_prob, TestData$Manipulater)
perf <- performance(pred,"tpr","fpr")

opt.cut = function(perf, pred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x-0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf@x.values, perf@y.values, pred@cutoffs)}

print(opt.cut(perf, pred))
table(pred_prob >  0.4324689,TestData$Manipulater)
pred_Class <- as.factor(ifelse(pred_prob > 0.4324689,"Yes","No"))
with(mylogit, null.deviance - deviance) #1091.685
(1-mean(pred_Class != TestData$Manipulater))*100
```

#	No	Yes
#FALSE	239	35
#TRUE	40	257

###Accuracy : 86.86515%
###Sensitivity : 0.8709677
###Specificity : 0.8664384

The Logistic regression model created using complete data set has a higher Accuracy , Sensitivity and Specificity than the model created with 220 sample data.

(i) Develop models using ensemble machine learning algorithms such as random forest and Ada-boosting. compare the outputs from these methods with logistic regression and classification tree.

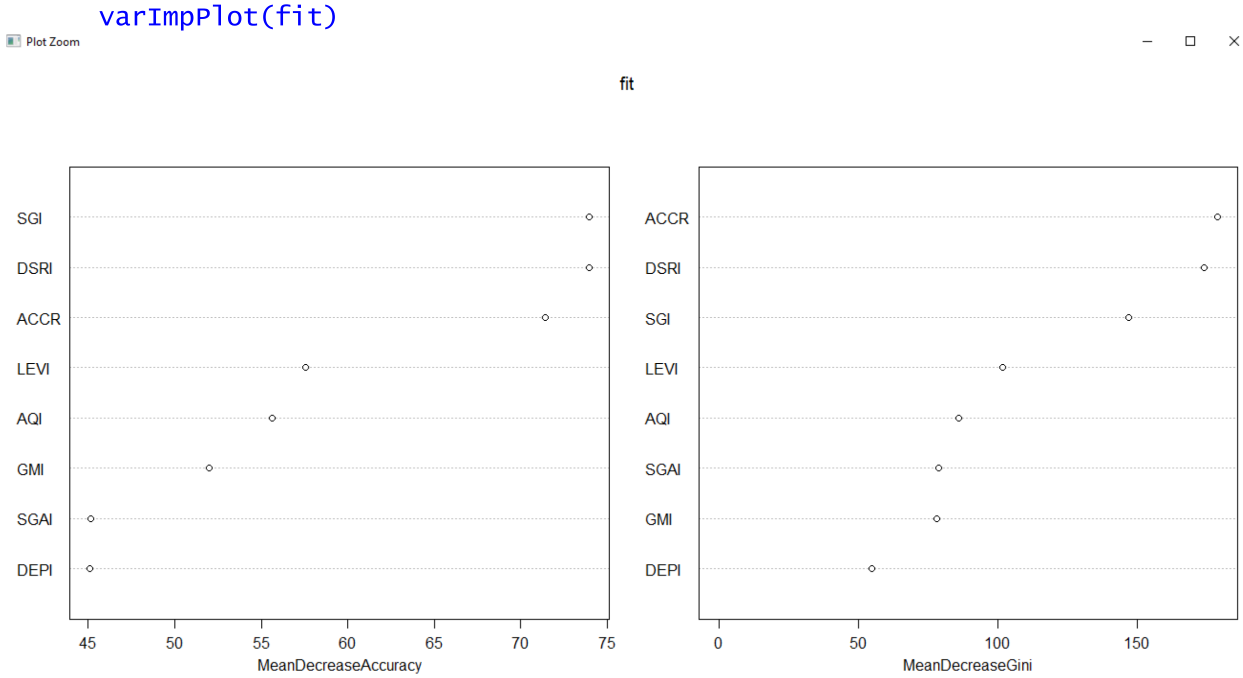
#RANDOM FOREST

```
set.seed(1234)
library(randomForest)
library(caret)
fit = randomForest(Manipulater ~ ., data=TrainData,
```

```
                    importance=TRUE, proximity=TRUE)
fit
predict.rf <- predict(fit,newdata = TestData)
confusionMatrix(predict.rf, TestData$Manipulater, positive = "Yes")
```

```
> importance(fit)
            No      Yes MeanDecreaseAccuracy MeanDecreaseGini
DSRI 38.417540 76.70083             73.93459        173.67322
GMI  15.825303 51.88635             52.01556         78.11774
AQI  12.144346 56.42823             55.60899         85.90023
SGI  34.935005 71.60039             73.94785        146.77115
DEPI  5.553763 46.45350             45.10540         54.78787
SGAI 18.550453 46.60173             45.17286         78.69418
ACCR 24.627742 71.06120             71.41700        178.75662
LEVI 33.041839 55.57578             57.55714        101.75472
```

```
varImpPlot(fit)
```



#		Actual	
#Prediction		**No**	**Yes**
#	**No**	267	0
#	**Yes**	12	292

###Accuracy : 97.9%
###Sensitivity : 1.0000
###Specificity : 0.9570

#ADABOOST

```
set.seed(1234)
```

```
man.adaboost <- boosting(Manipulater ~ ., data = TrainData, mfinal =
10, control = rpart.control(maxdepth = 1))

man.adaboost
```

trees show the weaklearners used at each iteration

```
man.adaboost$trees

man.adaboost$trees[[1]]
```

weights returns the voting power

```
man.adaboost$weights
```

prob returns the confidence of predictions

```
man.adaboost$prob
```

class returns the predicted class

```
man.adaboost$class
```

votes indicates the weighted predicted class

```
man.adaboost$votes
```

#importance returns important variables

```
man.adaboost$importance

table(man.adaboost$class, TrainData$Manipulater, dnn = c("Predicted
Class", "Observed Class"))
```

#		Observed Class	
#Predicted Class		**No**	**Yes**
#	**No**	791	265
#	**Yes**	130	613

```
errorrate <- 1 - sum(man.adaboost$class == TrainData$Manipulater)
/length(TrainData$Manipulater)

errorrate
```

To get predicted class on test data we can use predict function

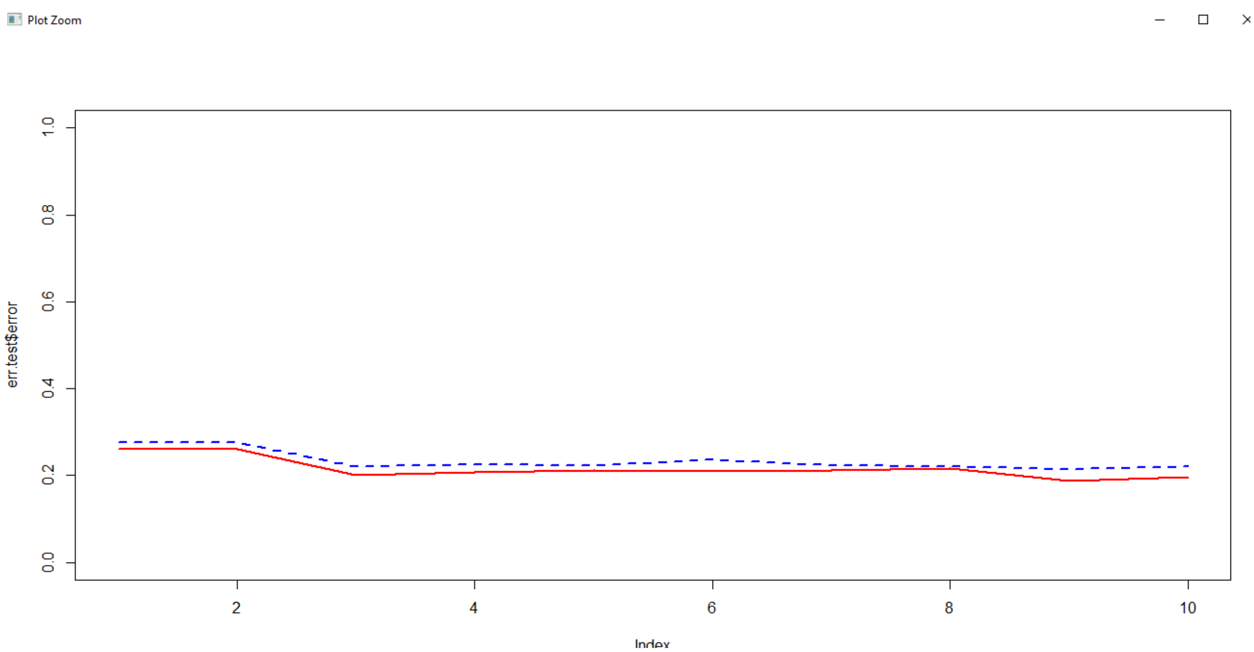
```
pred <- predict(man.adaboost,newdata = TestData)
```

#		Observed Class	
#Predicted Class		**No**	**Yes**
#	**No**	242	75
#	**Yes**	37	217

Accuracy : 80.3853%
###Sensitivity : 0.7432
###Specificity : 0.8674

However if you use predict.boosting, you can change mfinal
```
man.predboosting <- predict.boosting(man.adaboost, newdata = TestData)
```

errorevol calculates errors at each iteration of adaboost
```
err.train <- errorevol(man.adaboost,TrainData)

err.test <- errorevol(man.adaboost,TestData)

plot(err.test$error, type = "l", ylim = c(0,1), col = "red", lwd = 2)

lines(err.train$error, cex = 0.5, col = "blue", lty = 2, lwd = 2)
```



Method	Accuracy	Sensitivity	Specificity
CART	87.3905%	0.8938	0.8530
Logistic Regression	86.8652%	0.871	0.8664
Random Forest	97.9%	1.0000	0.9570
Adaboost	80.3853%	0.7432	0.8674

(j) What will be your final recommendation for predicting earnings manipulators? What variables should be considered important?

Based on the analysis done using different models, we believe Random Forest model to be the optimum model. Our decision is influenced by the near to perfect values for Accuracy, Sensitivity and Specificity as mentioned and highlighted in the above table. From the Variable Importance plot, the mean GINI decrease is maximum for **ACCR** followed by **DSRI, SGI, LEVI, AQI, GMI.** Thus we can say these variables should be considered important while making a decision.

Problem 2

(a) Compute the output of the hidden-layer and the output-layer neurons for the given input (0:5; 1).

Outputs of the hidden-layer neurons:

0.5*(-1) + 1*0 + (-0.5) = -1

0.5*2 + 1*1 + 2 = 4

0.5*(-2) + 1*1 + 0 = 0

Outputs of the output-layer neurons:

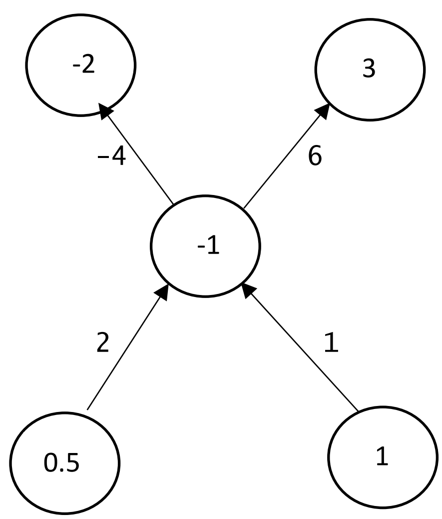
(-1)*(2) + 4*(-0.5) + 0*1 +(-2) = -6

(-1)*(-2) + 4*(1) + 0*0.5 +(3) = 9

(-6,9)

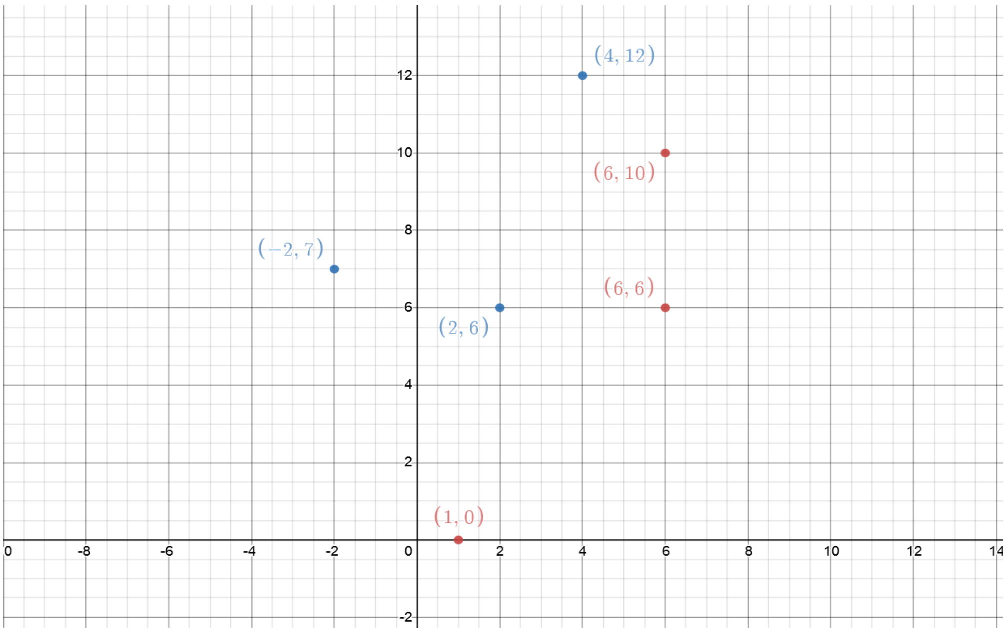
(b) Can you replace the above network with only one neuron that produces the same output? If yes, what are the weights and activation function in this neuron.

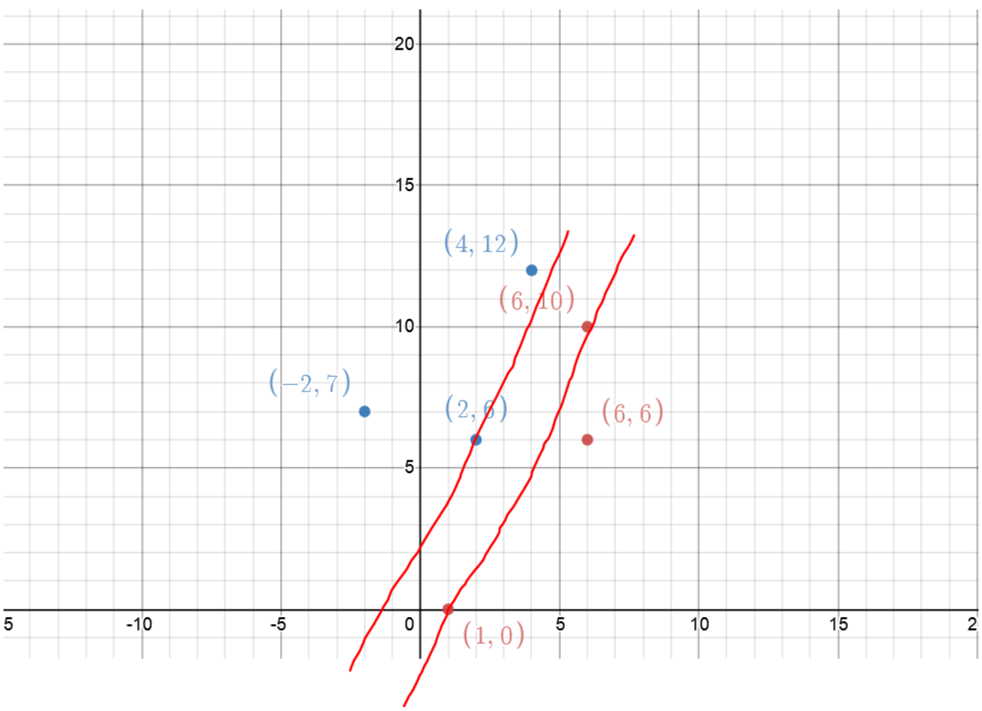
Yes, we can replace the above network with only one neuron that produces the same output. The activation function used here is Identity. Refer the figure below for weights.



Problem 3

Construct a SVM model that separates this data. Indicate the classification (+1 or−1) that the SVM gives to each of these points: (4,3), (0,4), (3,7). Removing which points change y our SVM decision boundary?





The points selected that are support vectors are (2,6) (6,10) (1,0):

$2w_1 + 6w_2 + b \ = \ -1 \ldots\ldots\ldots\ldots (1)$

$6w_1 + 10w_2 + b \ = \ 1 \ldots\ldots\ldots (2)$

$$w_1 + 0w_2 + b = 1 \ldots\ldots\ldots\ldots (3)$$

Solving (1),(2),(3) we get

$$w_1 = 1, w_2 = -0.5, b = 0$$

$$H(x) = 1x_1 - 0.5x_2 + 0$$

Identifying the class for point (4,3)

$H(x) =$ 1(4) - 0.5(3) = 4-1.5 = 2.5>1 ~ +1

Identifying the class for point (0,4) in H(x)

$H(x) =$ 1(0) - 0.5(4) = 0-2 = -2<1 ~ -1

Identifying the class for point (3,7) in H(x)

$H(x) =$ 1(3) - 0.5(7) = 3-3.5 =-0.5<1 ~ -1

It is obvious from the graph that if we change any of the points that is (2,6) (6,10) (1,0) then the value of the margin would increase. For Example , removing the point (6,10) will change the SVM as shown below.