1. Beneish model was developed using 8 financial indexes to identify whether financial earnings are manipulated. Reasons why it could not be relevant for Indian data are as follows:

* The model is probalisitic method, when data is imbalanced it shows bias towards majority class present in the data. Not ideal for classification when data is highly imbalanced
* Models were used in the financial industry, application on different range of industries might be harder.

1. One of the biggest stumbling blocks is the humongous data and its distribution. Fraudulent transactions are significantly lower than normal healthy transactions i.e. accounting it to around 1-2 % of the total number of observations. The ask is to improve identification of the rare minority class as opposed to achieving higher overall accuracy. Machine Learning algorithms tend to produce unsatisfactory classifiers when faced with imbalanced datasets. For any imbalanced data set, if the event to be predicted belongs to the minority class and the event rate is less than 5%, it is usually referred to as a rare event. Here are different ways to mitigate the problem:

* **Using various evaluation metrics other than just simple accuracy:**

Confusion matrix, Recall, Precision, roc curve ,F score can be used other than just accuracy

* **Re Sampling techniques:**

**Random Over Sampling:** Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

**Random Under Sampling:** Random Under sampling aims to balance class distribution by randomly eliminating majority class examples.  This is done until the majority and minority class instances are balanced out.

**Cluster based over sampling:** In this case, the K-means clustering algorithm is independently applied to minority and majority class instances. This is to identify clusters in the dataset. Subsequently, each cluster is oversampled such that all clusters of the same class have an equal number of instances and all classes have the same size

**Synthetic Minority Over Sampling Technique:** This technique is followed to avoid overfitting which occurs when exact replicas of minority instances are added to the main dataset. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created. These synthetic instances are then added to the original dataset. The new dataset is used as a sample to train the classification models

**Question 3-6)**

library(readxl)  
library(ROSE)

## Loaded ROSE 0.0-3

sample\_data= read\_excel("C:/Users/rakesh/Desktop/Business data mining/Assignment 3/sample\_data.xlsx")  
sample\_data\_2 <- sample\_data[,-c(1,10)]  
sample\_data\_2$`C-MANIPULATOR` <- as.factor(sample\_data\_2$`C-MANIPULATOR`)  
str(sample\_data\_2)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 220 obs. of 9 variables:  
## $ DSRI : num 1.62 1 1 1.49 1 ...  
## $ GMI : num 1.13 1.61 1.02 1 1.37 ...  
## $ AQI : num 7.185 1.005 1.241 0.466 0.637 ...  
## $ SGI : num 0.366 13.081 1.475 0.673 0.861 ...  
## $ DEPI : num 1.38 0.4 1.17 2 1.45 ...  
## $ SGAI : num 1.6241 5.1982 0.6477 0.0929 1.7415 ...  
## $ ACCR : num -0.1668 0.0605 0.0367 0.2734 0.123 ...  
## $ LEVI : num 1.161 0.987 1.264 0.681 0.939 ...  
## $ C-MANIPULATOR: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

prop.table(table(sample\_data\_2$`C-MANIPULATOR`))

##   
## 0 1   
## 0.8227273 0.1772727

colnames(sample\_data\_2)[9] <- c('target')  
colnames(sample\_data\_2)

## [1] "DSRI" "GMI" "AQI" "SGI" "DEPI" "SGAI" "ACCR" "LEVI"   
## [9] "target"

data\_balanced\_both <- ROSE(target ~ ., data = sample\_data\_2, seed = 1)$data  
  
prop.table(table(data\_balanced\_both$target))

##   
## 0 1   
## 0.4863636 0.5136364

table(data\_balanced\_both$target)

##   
## 0 1   
## 107 113

# creating training and test datasets after balancing   
set.seed(1234)  
index = sample(2, nrow(data\_balanced\_both), replace = TRUE, prob = c(0.7,0.3))  
TrainData = data\_balanced\_both[index == 1, ]  
nrow(TrainData)

## [1] 158

TestData = data\_balanced\_both[index == 2,]  
nrow(TestData)

## [1] 62

# Question-3  
## Using logistic regression for the sample data   
# iteration-1  
log\_model <- glm(target ~ ., family = binomial(link = "logit"),   
 TrainData)  
summary(log\_model)

##   
## Call:  
## glm(formula = target ~ ., family = binomial(link = "logit"),   
## data = TrainData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2274 -0.9637 -0.7905 0.9736 2.4752   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.30920 0.53878 -2.430 0.01510 \*   
## DSRI 0.03773 0.04820 0.783 0.43372   
## GMI 0.16235 0.05373 3.021 0.00252 \*\*  
## AQI 0.02976 0.03200 0.930 0.35230   
## SGI 0.29697 0.13988 2.123 0.03375 \*   
## DEPI 0.19328 0.40281 0.480 0.63135   
## SGAI 0.09040 0.03273 2.762 0.00574 \*\*  
## ACCR 2.36269 1.18974 1.986 0.04704 \*   
## LEVI 0.02532 0.16340 0.155 0.87683   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 218.12 on 157 degrees of freedom  
## Residual deviance: 186.67 on 149 degrees of freedom  
## AIC: 204.67  
##   
## Number of Fisher Scoring iterations: 5

# iteration-2  
log\_model <- glm(target ~ DSRI+GMI+AQI+SGI+DEPI+ACCR+SGAI, family = binomial(link = "logit"),   
 TrainData)  
summary(log\_model)

##   
## Call:  
## glm(formula = target ~ DSRI + GMI + AQI + SGI + DEPI + ACCR +   
## SGAI, family = binomial(link = "logit"), data = TrainData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.2274 -0.9628 -0.7924 0.9747 2.4676   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.28431 0.51333 -2.502 0.01235 \*   
## DSRI 0.03714 0.04808 0.772 0.43989   
## GMI 0.16133 0.05322 3.031 0.00243 \*\*  
## AQI 0.03077 0.03122 0.986 0.32429   
## SGI 0.29875 0.13976 2.138 0.03254 \*   
## DEPI 0.19314 0.40250 0.480 0.63133   
## ACCR 2.35768 1.19070 1.980 0.04769 \*   
## SGAI 0.09055 0.03276 2.764 0.00571 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 218.12 on 157 degrees of freedom  
## Residual deviance: 186.70 on 150 degrees of freedom  
## AIC: 202.7  
##   
## Number of Fisher Scoring iterations: 5

# iteration-3  
log\_model <- glm(target ~ DSRI+GMI+SGI, family = binomial(link = "logit"),   
 TrainData)  
summary(log\_model)

##   
## Call:  
## glm(formula = target ~ DSRI + GMI + SGI, family = binomial(link = "logit"),   
## data = TrainData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.1424 -1.0202 -0.9352 1.1823 2.0373   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.86445 0.26848 -3.220 0.00128 \*\*  
## DSRI 0.07174 0.04080 1.758 0.07870 .   
## GMI 0.13595 0.04964 2.739 0.00617 \*\*  
## SGI 0.25315 0.12233 2.069 0.03851 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 218.12 on 157 degrees of freedom  
## Residual deviance: 198.05 on 154 degrees of freedom  
## AIC: 206.05  
##   
## Number of Fisher Scoring iterations: 5

4 and 6 questions) 0.43 is the m-score which would be ideal because it gives

Better precision and recall than any probability scores

######## Measuring accuracies using the above model#############

# training error and estimation  
pred\_train\_model <-predict(log\_model,TrainData,type = 'response')  
pred\_train\_model <- ifelse(pred\_train\_model>0.43,1,0)  
  
confusion\_matrix <- table(pred\_train\_model,TrainData$target,dnn=c("Predicted","Actual"))  
confusion\_matrix

## Actual  
## Predicted 0 1  
## 0 63 23  
## 1 22 50

accuracy=sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.7151899

recall = confusion\_matrix[2,2]/(confusion\_matrix[1,2]+confusion\_matrix[2,2])   
recall

## [1] 0.6849315

precision= confusion\_matrix[2,2]/(confusion\_matrix[2,1]+confusion\_matrix[2,2])  
precision

## [1] 0.6944444

f\_score=2\*precision\*recall/(precision+recall)  
f\_score

## [1] 0.6896552

#######Question-5#######################################

iteration-3  
log\_model <- glm(target ~ DSRI+GMI+SGI, family = binomial(link = "logit"),   
 TrainData)  
summary(log\_model)

##   
## Call:  
## glm(formula = target ~ DSRI + GMI + SGI, family = binomial(link = "logit"),   
## data = TrainData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.1424 -1.0202 -0.9352 1.1823 2.0373   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.86445 0.26848 -3.220 0.00128 \*\*  
## DSRI 0.07174 0.04080 1.758 0.07870 .   
## GMI 0.13595 0.04964 2.739 0.00617 \*\*  
## SGI 0.25315 0.12233 2.069 0.03851 \*

DSRI,GMI,SGI are the significant variables which can affect the likelihood of fraud prediction, hence as company it would be ideal to have a closer look into those indices for fraud prediction

**Question-8)**

**Logistic regression on complete data is more robust than on the sample data with higher accuracies and precision and recall**

library(readxl)  
library(ROSE)

## Loaded ROSE 0.0-3

complete\_data= read\_excel("C:/Users/rakesh/Desktop/Business data mining/Assignment 3/complete\_data.xlsx")  
  
#########complete data set balancing ##################  
  
complete\_data\_2 <- complete\_data[,-c(1,10)]  
complete\_data\_2$`C-MANIPULATOR` <- as.factor(complete\_data\_2$`C-MANIPULATOR`)  
str(complete\_data\_2)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 1239 obs. of 9 variables:  
## $ DSRI : num 1.62 1 1 1.49 1 ...  
## $ GMI : num 1.13 1.61 1.02 1 1.37 ...  
## $ AQI : num 7.185 1.005 1.241 0.466 0.637 ...  
## $ SGI : num 0.366 13.081 1.475 0.673 0.861 ...  
## $ DEPI : num 1.38 0.4 1.17 2 1.45 ...  
## $ SGAI : num 1.6241 5.1982 0.6477 0.0929 1.7415 ...  
## $ ACCR : num -0.1668 0.0605 0.0367 0.2734 0.123 ...  
## $ LEVI : num 1.161 0.987 1.264 0.681 0.939 ...  
## $ C-MANIPULATOR: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

prop.table(table(complete\_data\_2$`C-MANIPULATOR`))

##   
## 0 1   
## 0.968523 0.031477

colnames(complete\_data\_2)[9] <- c('target')  
colnames(complete\_data\_2)

## [1] "DSRI" "GMI" "AQI" "SGI" "DEPI" "SGAI" "ACCR" "LEVI"   
## [9] "target"

complete\_balanced\_data <- ROSE(target ~ ., data = complete\_data\_2, seed = 1)$data  
  
prop.table(table(complete\_balanced\_data$target))

##   
## 0 1   
## 0.5238095 0.4761905

# creating training and test datasets after balancing   
set.seed(1234)  
index = sample(2, nrow(complete\_balanced\_data), replace = TRUE, prob = c(0.7,0.3))  
TrainData = complete\_balanced\_data[index == 1, ]  
nrow(TrainData)

## [1] 874

TestData = complete\_balanced\_data[index == 2,]  
nrow(TestData)

## [1] 365

## Using logistic regression for the complete data   
# iteration-1  
log\_model <- glm(target ~ ., family = binomial(link = "logit"),   
 TrainData)  
summary(log\_model)

##   
## Call:  
## glm(formula = target ~ ., family = binomial(link = "logit"),   
## data = TrainData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6794 -0.9888 -0.7053 0.9664 2.5646   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.96135 0.24553 -3.915 9.03e-05 \*\*\*  
## DSRI 0.11504 0.02188 5.257 1.46e-07 \*\*\*  
## GMI 0.07130 0.01765 4.040 5.35e-05 \*\*\*  
## AQI 0.05565 0.01254 4.438 9.09e-06 \*\*\*  
## SGI 0.20129 0.03651 5.513 3.53e-08 \*\*\*  
## DEPI -0.12526 0.19609 -0.639 0.522949   
## SGAI 0.06907 0.01572 4.393 1.12e-05 \*\*\*  
## ACCR 2.76240 0.41675 6.628 3.39e-11 \*\*\*  
## LEVI 0.14799 0.04472 3.310 0.000935 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1210.7 on 873 degrees of freedom  
## Residual deviance: 1025.1 on 865 degrees of freedom  
## AIC: 1043.1  
##   
## Number of Fisher Scoring iterations: 5

# iteration-2  
log\_model <- glm(target ~ DSRI+GMI+AQI+SGI+ACCR+SGAI, family = binomial(link = "logit"),   
 TrainData)  
summary(log\_model)

##   
## Call:  
## glm(formula = target ~ DSRI + GMI + AQI + SGI + ACCR + SGAI,   
## family = binomial(link = "logit"), data = TrainData)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.6687 -1.0015 -0.7280 0.9784 2.4638   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.91059 0.10985 -8.290 < 2e-16 \*\*\*  
## DSRI 0.12404 0.02169 5.717 1.08e-08 \*\*\*  
## GMI 0.06519 0.01718 3.794 0.000148 \*\*\*  
## AQI 0.05650 0.01202 4.699 2.61e-06 \*\*\*  
## SGI 0.20889 0.03645 5.731 9.96e-09 \*\*\*  
## ACCR 2.72547 0.41522 6.564 5.24e-11 \*\*\*  
## SGAI 0.06195 0.01519 4.078 4.54e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1210.7 on 873 degrees of freedom  
## Residual deviance: 1038.6 on 867 degrees of freedom  
## AIC: 1052.6  
##   
## Number of Fisher Scoring iterations: 5

# training error and estimation  
pred\_train\_model <-predict(log\_model,TrainData,type = 'response')  
pred\_train\_model <- ifelse(pred\_train\_model>0.47,1,0)  
  
confusion\_matrix <- table(pred\_train\_model,TrainData$target,dnn=c("Predicted","Actual"))  
confusion\_matrix

## Actual  
## Predicted 0 1  
## 0 380 144  
## 1 71 279

accuracy=sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.7540046

recall = confusion\_matrix[2,2]/(confusion\_matrix[1,2]+confusion\_matrix[2,2])   
recall

## [1] 0.6595745

precision= confusion\_matrix[2,2]/(confusion\_matrix[2,1]+confusion\_matrix[2,2])  
precision

## [1] 0.7971429

f\_score=2\*precision\*recall/(precision+recall)  
f\_score

## [1] 0.7218629

# testing error and estimation  
pred\_test\_model <-predict(log\_model,TestData,type = 'response')  
pred\_test\_model <- ifelse(pred\_test\_model>0.47,1,0)  
  
confusion\_matrix <- table(pred\_test\_model,TestData$target,dnn=c("Predicted","Actual"))  
confusion\_matrix

## Actual  
## Predicted 0 1  
## 0 159 63  
## 1 39 104

accuracy=sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.7205479

accuracy=sum(diag(confusion\_matrix))/sum(confusion\_matrix)  
accuracy

## [1] 0.7205479

recall = confusion\_matrix[2,2]/(confusion\_matrix[1,2]+confusion\_matrix[2,2])   
recall

## [1] 0.6227545

precision= confusion\_matrix[2,2]/(confusion\_matrix[2,1]+confusion\_matrix[2,2])  
precision

## [1] 0.7272727

f\_score=2\*precision\*recall/(precision+recall)  
f\_score

## [1] 0.6709677

#Questions 7,9,10  
library(ROSE)

## Warning: package 'ROSE' was built under R version 3.4.2

## Loaded ROSE 0.0-3

library(readxl)  
sample\_data <- read\_excel("C:/Users/sruja/Downloads/sample\_data.xlsx")  
  
sample\_data\_2 <- sample\_data[,-c(1,10)]  
sample\_data\_2$`C-MANIPULATOR` <- as.factor(sample\_data\_2$`C-MANIPULATOR`)  
str(sample\_data\_2)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 220 obs. of 9 variables:  
## $ DSRI : num 1.62 1 1 1.49 1 ...  
## $ GMI : num 1.13 1.61 1.02 1 1.37 ...  
## $ AQI : num 7.185 1.005 1.241 0.466 0.637 ...  
## $ SGI : num 0.366 13.081 1.475 0.673 0.861 ...  
## $ DEPI : num 1.38 0.4 1.17 2 1.45 ...  
## $ SGAI : num 1.6241 5.1982 0.6477 0.0929 1.7415 ...  
## $ ACCR : num -0.1668 0.0605 0.0367 0.2734 0.123 ...  
## $ LEVI : num 1.161 0.987 1.264 0.681 0.939 ...  
## $ C-MANIPULATOR: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

prop.table(table(sample\_data\_2$`C-MANIPULATOR`))

##   
## 0 1   
## 0.8227273 0.1772727

colnames(sample\_data\_2)[9] <- c('target')  
colnames(sample\_data\_2)

## [1] "DSRI" "GMI" "AQI" "SGI" "DEPI" "SGAI" "ACCR" "LEVI"   
## [9] "target"

data\_balanced\_both <- ROSE(target ~ ., data = sample\_data\_2, seed = 1)$data  
  
prop.table(table(data\_balanced\_both$target))

##   
## 0 1   
## 0.4863636 0.5136364

table(data\_balanced\_both$target)

##   
## 0 1   
## 107 113

# creating training and test datasets after balancing   
set.seed(1234)  
index = sample(2, nrow(data\_balanced\_both), replace = TRUE, prob = c(0.7,0.3))  
TrainData = data\_balanced\_both[index == 1, ]  
nrow(TrainData)

## [1] 158

TestData = data\_balanced\_both[index == 2,]  
nrow(TestData)

## [1] 62

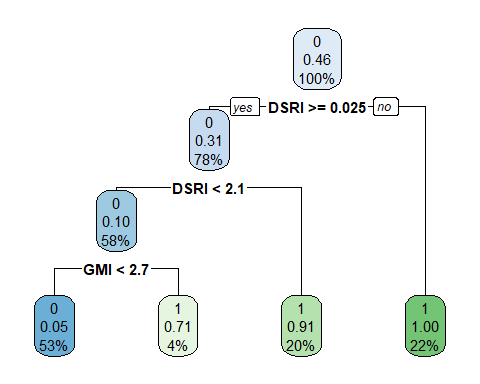
library(rpart)  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.4.2

tree <- rpart(target~., data=TrainData, method = "class")  
bestcp <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]  
  
tree.pruned <- prune(tree, cp = bestcp)  
summary(tree.pruned)

## Call:  
## rpart(formula = target ~ ., data = TrainData, method = "class")  
## n= 158   
##   
## CP nsplit rel error xerror xstd  
## 1 0.47945205 0 1.0000000 1.0000000 0.08584589  
## 2 0.35616438 1 0.5205479 0.5342466 0.07424281  
## 3 0.04109589 2 0.1643836 0.2328767 0.05335592  
##   
## Variable importance  
## DSRI SGAI LEVI SGI GMI ACCR   
## 48 12 11 11 9 9   
##   
## Node number 1: 158 observations, complexity param=0.4794521  
## predicted class=0 expected loss=0.4620253 P(node) =1  
## class counts: 85 73  
## probabilities: 0.538 0.462   
## left son=2 (123 obs) right son=3 (35 obs)  
## Primary splits:  
## DSRI < 0.02488603 to the right, improve=26.02398, (0 missing)  
## SGAI < 2.069646 to the left, improve=26.02398, (0 missing)  
## SGI < 1.604829 to the left, improve=23.22684, (0 missing)  
## GMI < 2.78907 to the left, improve=22.40958, (0 missing)  
## LEVI < 0.4038873 to the right, improve=18.84965, (0 missing)  
## Surrogate splits:  
## GMI < 9.262613 to the left, agree=0.816, adj=0.171, (0 split)  
## SGI < -0.3986336 to the right, agree=0.797, adj=0.086, (0 split)  
## SGAI < -8.950582 to the right, agree=0.797, adj=0.086, (0 split)  
## ACCR < 0.1450163 to the left, agree=0.797, adj=0.086, (0 split)  
## LEVI < 2.397781 to the left, agree=0.797, adj=0.086, (0 split)  
##   
## Node number 2: 123 observations, complexity param=0.3561644  
## predicted class=0 expected loss=0.3089431 P(node) =0.778481  
## class counts: 85 38  
## probabilities: 0.691 0.309   
## left son=4 (91 obs) right son=5 (32 obs)  
## Primary splits:  
## DSRI < 2.132406 to the left, improve=30.86304, (0 missing)  
## SGAI < 2.069646 to the left, improve=21.46263, (0 missing)  
## SGI < 1.604829 to the left, improve=18.84108, (0 missing)  
## GMI < 3.394865 to the left, improve=17.56705, (0 missing)  
## LEVI < 0.3960195 to the right, improve=13.88396, (0 missing)  
## Surrogate splits:  
## SGAI < 7.055184 to the left, agree=0.837, adj=0.375, (0 split)  
## SGI < 0.5691286 to the right, agree=0.829, adj=0.344, (0 split)  
## LEVI < 0.3960195 to the right, agree=0.829, adj=0.344, (0 split)  
## ACCR < -0.2121131 to the right, agree=0.813, adj=0.281, (0 split)  
## GMI < -2.009743 to the right, agree=0.797, adj=0.219, (0 split)  
##   
## Node number 3: 35 observations  
## predicted class=1 expected loss=0 P(node) =0.221519  
## class counts: 0 35  
## probabilities: 0.000 1.000   
##   
## Node number 4: 91 observations  
## predicted class=0 expected loss=0.0989011 P(node) =0.5759494  
## class counts: 82 9  
## probabilities: 0.901 0.099   
##   
## Node number 5: 32 observations  
## predicted class=1 expected loss=0.09375 P(node) =0.2025316  
## class counts: 3 29  
## probabilities: 0.094 0.906

rpart.plot(tree)



summary(tree)

## Call:  
## rpart(formula = target ~ ., data = TrainData, method = "class")  
## n= 158   
##   
## CP nsplit rel error xerror xstd  
## 1 0.47945205 0 1.0000000 1.0000000 0.08584589  
## 2 0.35616438 1 0.5205479 0.5342466 0.07424281  
## 3 0.04109589 2 0.1643836 0.2328767 0.05335592  
## 4 0.01000000 3 0.1232877 0.2328767 0.05335592  
##   
## Variable importance  
## DSRI GMI SGAI SGI LEVI ACCR AQI   
## 44 13 13 11 10 8 1   
##   
## Node number 1: 158 observations, complexity param=0.4794521  
## predicted class=0 expected loss=0.4620253 P(node) =1  
## class counts: 85 73  
## probabilities: 0.538 0.462   
## left son=2 (123 obs) right son=3 (35 obs)  
## Primary splits:  
## DSRI < 0.02488603 to the right, improve=26.02398, (0 missing)  
## SGAI < 2.069646 to the left, improve=26.02398, (0 missing)  
## SGI < 1.604829 to the left, improve=23.22684, (0 missing)  
## GMI < 2.78907 to the left, improve=22.40958, (0 missing)  
## LEVI < 0.4038873 to the right, improve=18.84965, (0 missing)  
## Surrogate splits:  
## GMI < 9.262613 to the left, agree=0.816, adj=0.171, (0 split)  
## SGI < -0.3986336 to the right, agree=0.797, adj=0.086, (0 split)  
## SGAI < -8.950582 to the right, agree=0.797, adj=0.086, (0 split)  
## ACCR < 0.1450163 to the left, agree=0.797, adj=0.086, (0 split)  
## LEVI < 2.397781 to the left, agree=0.797, adj=0.086, (0 split)  
##   
## Node number 2: 123 observations, complexity param=0.3561644  
## predicted class=0 expected loss=0.3089431 P(node) =0.778481  
## class counts: 85 38  
## probabilities: 0.691 0.309   
## left son=4 (91 obs) right son=5 (32 obs)  
## Primary splits:  
## DSRI < 2.132406 to the left, improve=30.86304, (0 missing)  
## SGAI < 2.069646 to the left, improve=21.46263, (0 missing)  
## SGI < 1.604829 to the left, improve=18.84108, (0 missing)  
## GMI < 3.394865 to the left, improve=17.56705, (0 missing)  
## LEVI < 0.3960195 to the right, improve=13.88396, (0 missing)  
## Surrogate splits:  
## SGAI < 7.055184 to the left, agree=0.837, adj=0.375, (0 split)  
## SGI < 0.5691286 to the right, agree=0.829, adj=0.344, (0 split)  
## LEVI < 0.3960195 to the right, agree=0.829, adj=0.344, (0 split)  
## ACCR < -0.2121131 to the right, agree=0.813, adj=0.281, (0 split)  
## GMI < -2.009743 to the right, agree=0.797, adj=0.219, (0 split)  
##   
## Node number 3: 35 observations  
## predicted class=1 expected loss=0 P(node) =0.221519  
## class counts: 0 35  
## probabilities: 0.000 1.000   
##   
## Node number 4: 91 observations, complexity param=0.04109589  
## predicted class=0 expected loss=0.0989011 P(node) =0.5759494  
## class counts: 82 9  
## probabilities: 0.901 0.099   
## left son=8 (84 obs) right son=9 (7 obs)  
## Primary splits:  
## GMI < 2.73977 to the left, improve=5.743590, (0 missing)  
## AQI < -2.424513 to the right, improve=5.743590, (0 missing)  
## SGI < 1.510101 to the left, improve=5.743590, (0 missing)  
## SGAI < 1.725166 to the left, improve=5.743590, (0 missing)  
## LEVI < 1.479553 to the left, improve=3.386447, (0 missing)  
## Surrogate splits:  
## SGAI < 2.942175 to the left, agree=0.956, adj=0.429, (0 split)  
## SGI < 0.1081662 to the right, agree=0.945, adj=0.286, (0 split)  
## AQI < -4.866145 to the right, agree=0.934, adj=0.143, (0 split)  
##   
## Node number 5: 32 observations  
## predicted class=1 expected loss=0.09375 P(node) =0.2025316  
## class counts: 3 29  
## probabilities: 0.094 0.906   
##   
## Node number 8: 84 observations  
## predicted class=0 expected loss=0.04761905 P(node) =0.5316456  
## class counts: 80 4  
## probabilities: 0.952 0.048   
##   
## Node number 9: 7 observations  
## predicted class=1 expected loss=0.2857143 P(node) =0.0443038  
## class counts: 2 5  
## probabilities: 0.286 0.714

summary(tree.pruned)

## Call:  
## rpart(formula = target ~ ., data = TrainData, method = "class")  
## n= 158   
##   
## CP nsplit rel error xerror xstd  
## 1 0.47945205 0 1.0000000 1.0000000 0.08584589  
## 2 0.35616438 1 0.5205479 0.5342466 0.07424281  
## 3 0.04109589 2 0.1643836 0.2328767 0.05335592  
##   
## Variable importance  
## DSRI SGAI LEVI SGI GMI ACCR   
## 48 12 11 11 9 9   
##   
## Node number 1: 158 observations, complexity param=0.4794521  
## predicted class=0 expected loss=0.4620253 P(node) =1  
## class counts: 85 73  
## probabilities: 0.538 0.462   
## left son=2 (123 obs) right son=3 (35 obs)  
## Primary splits:  
## DSRI < 0.02488603 to the right, improve=26.02398, (0 missing)  
## SGAI < 2.069646 to the left, improve=26.02398, (0 missing)  
## SGI < 1.604829 to the left, improve=23.22684, (0 missing)  
## GMI < 2.78907 to the left, improve=22.40958, (0 missing)  
## LEVI < 0.4038873 to the right, improve=18.84965, (0 missing)  
## Surrogate splits:  
## GMI < 9.262613 to the left, agree=0.816, adj=0.171, (0 split)  
## SGI < -0.3986336 to the right, agree=0.797, adj=0.086, (0 split)  
## SGAI < -8.950582 to the right, agree=0.797, adj=0.086, (0 split)  
## ACCR < 0.1450163 to the left, agree=0.797, adj=0.086, (0 split)  
## LEVI < 2.397781 to the left, agree=0.797, adj=0.086, (0 split)  
##   
## Node number 2: 123 observations, complexity param=0.3561644  
## predicted class=0 expected loss=0.3089431 P(node) =0.778481  
## class counts: 85 38  
## probabilities: 0.691 0.309   
## left son=4 (91 obs) right son=5 (32 obs)  
## Primary splits:  
## DSRI < 2.132406 to the left, improve=30.86304, (0 missing)  
## SGAI < 2.069646 to the left, improve=21.46263, (0 missing)  
## SGI < 1.604829 to the left, improve=18.84108, (0 missing)  
## GMI < 3.394865 to the left, improve=17.56705, (0 missing)  
## LEVI < 0.3960195 to the right, improve=13.88396, (0 missing)  
## Surrogate splits:  
## SGAI < 7.055184 to the left, agree=0.837, adj=0.375, (0 split)  
## SGI < 0.5691286 to the right, agree=0.829, adj=0.344, (0 split)  
## LEVI < 0.3960195 to the right, agree=0.829, adj=0.344, (0 split)  
## ACCR < -0.2121131 to the right, agree=0.813, adj=0.281, (0 split)  
## GMI < -2.009743 to the right, agree=0.797, adj=0.219, (0 split)  
##   
## Node number 3: 35 observations  
## predicted class=1 expected loss=0 P(node) =0.221519  
## class counts: 0 35  
## probabilities: 0.000 1.000   
##   
## Node number 4: 91 observations  
## predicted class=0 expected loss=0.0989011 P(node) =0.5759494  
## class counts: 82 9  
## probabilities: 0.901 0.099   
##   
## Node number 5: 32 observations  
## predicted class=1 expected loss=0.09375 P(node) =0.2025316  
## class counts: 3 29  
## probabilities: 0.094 0.906

accuracy\_sample\_data = table(predict(tree, TestData, type="class"), TestData$target)  
sum(diag(accuracy\_sample\_data))/sum(accuracy\_sample\_data)\*100

## [1] 83.87097

prec = 32/40  
rec = 32/34  
fscore = 2.83  
  
# Based on the CART Model we observe the following Precision and Recall.  
# We observe a accuracy of around 83.97 for sample data.  
# To analyse it further we do it on complete dat.  
  
# Complete Data  
  
library(readxl)  
complete\_data <- read\_excel("C:/Users/sruja/Downloads/sample\_data.xlsx")  
  
complete\_data\_2 <- complete\_data[,-c(1,10)]  
complete\_data\_2$`C-MANIPULATOR` <- as.factor(complete\_data\_2$`C-MANIPULATOR`)  
str(complete\_data\_2)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 220 obs. of 9 variables:  
## $ DSRI : num 1.62 1 1 1.49 1 ...  
## $ GMI : num 1.13 1.61 1.02 1 1.37 ...  
## $ AQI : num 7.185 1.005 1.241 0.466 0.637 ...  
## $ SGI : num 0.366 13.081 1.475 0.673 0.861 ...  
## $ DEPI : num 1.38 0.4 1.17 2 1.45 ...  
## $ SGAI : num 1.6241 5.1982 0.6477 0.0929 1.7415 ...  
## $ ACCR : num -0.1668 0.0605 0.0367 0.2734 0.123 ...  
## $ LEVI : num 1.161 0.987 1.264 0.681 0.939 ...  
## $ C-MANIPULATOR: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...

prop.table(table(complete\_data\_2$`C-MANIPULATOR`))

##   
## 0 1   
## 0.8227273 0.1772727

colnames(complete\_data\_2)[9] <- c('target')  
colnames(complete\_data\_2)

## [1] "DSRI" "GMI" "AQI" "SGI" "DEPI" "SGAI" "ACCR" "LEVI"   
## [9] "target"

data\_balanced\_both <- ROSE(target ~ ., data = complete\_data\_2, seed = 1)$data  
  
prop.table(table(data\_balanced\_both$target))

##   
## 0 1   
## 0.4863636 0.5136364

table(data\_balanced\_both$target)

##   
## 0 1   
## 107 113

# creating training and test datasets after balancing   
set.seed(1234)  
index = sample(2, nrow(data\_balanced\_both), replace = TRUE, prob = c(0.7,0.3))  
TrainData = data\_balanced\_both[index == 1, ]  
nrow(TrainData)

## [1] 158

TestData = data\_balanced\_both[index == 2,]  
nrow(TestData)

## [1] 62

#Rpart  
library(rpart)  
library(rpart.plot)  
tree <- rpart(target~., data=TrainData, method = "class")  
  
printcp(tree)

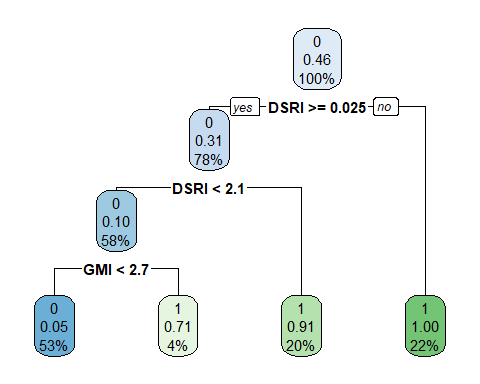
##   
## Classification tree:  
## rpart(formula = target ~ ., data = TrainData, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] DSRI GMI   
##   
## Root node error: 73/158 = 0.46203  
##   
## n= 158   
##   
## CP nsplit rel error xerror xstd  
## 1 0.479452 0 1.00000 1.00000 0.085846  
## 2 0.356164 1 0.52055 0.53425 0.074243  
## 3 0.041096 2 0.16438 0.23288 0.053356  
## 4 0.010000 3 0.12329 0.23288 0.053356

bestcp <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]  
tree.pruned <- prune(tree, cp = bestcp)  
plot(tree.pruned)

summary(tree.pruned)

## Call:  
## rpart(formula = target ~ ., data = TrainData, method = "class")  
## n= 158   
##   
## CP nsplit rel error xerror xstd  
## 1 0.47945205 0 1.0000000 1.0000000 0.08584589  
## 2 0.35616438 1 0.5205479 0.5342466 0.07424281  
## 3 0.04109589 2 0.1643836 0.2328767 0.05335592  
##   
## Variable importance  
## DSRI SGAI LEVI SGI GMI ACCR   
## 48 12 11 11 9 9   
##   
## Node number 1: 158 observations, complexity param=0.4794521  
## predicted class=0 expected loss=0.4620253 P(node) =1  
## class counts: 85 73  
## probabilities: 0.538 0.462   
## left son=2 (123 obs) right son=3 (35 obs)  
## Primary splits:  
## DSRI < 0.02488603 to the right, improve=26.02398, (0 missing)  
## SGAI < 2.069646 to the left, improve=26.02398, (0 missing)  
## SGI < 1.604829 to the left, improve=23.22684, (0 missing)  
## GMI < 2.78907 to the left, improve=22.40958, (0 missing)  
## LEVI < 0.4038873 to the right, improve=18.84965, (0 missing)  
## Surrogate splits:  
## GMI < 9.262613 to the left, agree=0.816, adj=0.171, (0 split)  
## SGI < -0.3986336 to the right, agree=0.797, adj=0.086, (0 split)  
## SGAI < -8.950582 to the right, agree=0.797, adj=0.086, (0 split)  
## ACCR < 0.1450163 to the left, agree=0.797, adj=0.086, (0 split)  
## LEVI < 2.397781 to the left, agree=0.797, adj=0.086, (0 split)  
##   
## Node number 2: 123 observations, complexity param=0.3561644  
## predicted class=0 expected loss=0.3089431 P(node) =0.778481  
## class counts: 85 38  
## probabilities: 0.691 0.309   
## left son=4 (91 obs) right son=5 (32 obs)  
## Primary splits:  
## DSRI < 2.132406 to the left, improve=30.86304, (0 missing)  
## SGAI < 2.069646 to the left, improve=21.46263, (0 missing)  
## SGI < 1.604829 to the left, improve=18.84108, (0 missing)  
## GMI < 3.394865 to the left, improve=17.56705, (0 missing)  
## LEVI < 0.3960195 to the right, improve=13.88396, (0 missing)  
## Surrogate splits:  
## SGAI < 7.055184 to the left, agree=0.837, adj=0.375, (0 split)  
## SGI < 0.5691286 to the right, agree=0.829, adj=0.344, (0 split)  
## LEVI < 0.3960195 to the right, agree=0.829, adj=0.344, (0 split)  
## ACCR < -0.2121131 to the right, agree=0.813, adj=0.281, (0 split)  
## GMI < -2.009743 to the right, agree=0.797, adj=0.219, (0 split)  
##   
## Node number 3: 35 observations  
## predicted class=1 expected loss=0 P(node) =0.221519  
## class counts: 0 35  
## probabilities: 0.000 1.000   
##   
## Node number 4: 91 observations  
## predicted class=0 expected loss=0.0989011 P(node) =0.5759494  
## class counts: 82 9  
## probabilities: 0.901 0.099   
##   
## Node number 5: 32 observations  
## predicted class=1 expected loss=0.09375 P(node) =0.2025316  
## class counts: 3 29  
## probabilities: 0.094 0.906

rpart.plot(tree)



summary(tree)

## Call:  
## rpart(formula = target ~ ., data = TrainData, method = "class")  
## n= 158   
##   
## CP nsplit rel error xerror xstd  
## 1 0.47945205 0 1.0000000 1.0000000 0.08584589  
## 2 0.35616438 1 0.5205479 0.5342466 0.07424281  
## 3 0.04109589 2 0.1643836 0.2328767 0.05335592  
## 4 0.01000000 3 0.1232877 0.2328767 0.05335592  
##   
## Variable importance  
## DSRI GMI SGAI SGI LEVI ACCR AQI   
## 44 13 13 11 10 8 1   
##   
## Node number 1: 158 observations, complexity param=0.4794521  
## predicted class=0 expected loss=0.4620253 P(node) =1  
## class counts: 85 73  
## probabilities: 0.538 0.462   
## left son=2 (123 obs) right son=3 (35 obs)  
## Primary splits:  
## DSRI < 0.02488603 to the right, improve=26.02398, (0 missing)  
## SGAI < 2.069646 to the left, improve=26.02398, (0 missing)  
## SGI < 1.604829 to the left, improve=23.22684, (0 missing)  
## GMI < 2.78907 to the left, improve=22.40958, (0 missing)  
## LEVI < 0.4038873 to the right, improve=18.84965, (0 missing)  
## Surrogate splits:  
## GMI < 9.262613 to the left, agree=0.816, adj=0.171, (0 split)  
## SGI < -0.3986336 to the right, agree=0.797, adj=0.086, (0 split)  
## SGAI < -8.950582 to the right, agree=0.797, adj=0.086, (0 split)  
## ACCR < 0.1450163 to the left, agree=0.797, adj=0.086, (0 split)  
## LEVI < 2.397781 to the left, agree=0.797, adj=0.086, (0 split)  
##   
## Node number 2: 123 observations, complexity param=0.3561644  
## predicted class=0 expected loss=0.3089431 P(node) =0.778481  
## class counts: 85 38  
## probabilities: 0.691 0.309   
## left son=4 (91 obs) right son=5 (32 obs)  
## Primary splits:  
## DSRI < 2.132406 to the left, improve=30.86304, (0 missing)  
## SGAI < 2.069646 to the left, improve=21.46263, (0 missing)  
## SGI < 1.604829 to the left, improve=18.84108, (0 missing)  
## GMI < 3.394865 to the left, improve=17.56705, (0 missing)  
## LEVI < 0.3960195 to the right, improve=13.88396, (0 missing)  
## Surrogate splits:  
## SGAI < 7.055184 to the left, agree=0.837, adj=0.375, (0 split)  
## SGI < 0.5691286 to the right, agree=0.829, adj=0.344, (0 split)  
## LEVI < 0.3960195 to the right, agree=0.829, adj=0.344, (0 split)  
## ACCR < -0.2121131 to the right, agree=0.813, adj=0.281, (0 split)  
## GMI < -2.009743 to the right, agree=0.797, adj=0.219, (0 split)  
##   
## Node number 3: 35 observations  
## predicted class=1 expected loss=0 P(node) =0.221519  
## class counts: 0 35  
## probabilities: 0.000 1.000   
##   
## Node number 4: 91 observations, complexity param=0.04109589  
## predicted class=0 expected loss=0.0989011 P(node) =0.5759494  
## class counts: 82 9  
## probabilities: 0.901 0.099   
## left son=8 (84 obs) right son=9 (7 obs)  
## Primary splits:  
## GMI < 2.73977 to the left, improve=5.743590, (0 missing)  
## AQI < -2.424513 to the right, improve=5.743590, (0 missing)  
## SGI < 1.510101 to the left, improve=5.743590, (0 missing)  
## SGAI < 1.725166 to the left, improve=5.743590, (0 missing)  
## LEVI < 1.479553 to the left, improve=3.386447, (0 missing)  
## Surrogate splits:  
## SGAI < 2.942175 to the left, agree=0.956, adj=0.429, (0 split)  
## SGI < 0.1081662 to the right, agree=0.945, adj=0.286, (0 split)  
## AQI < -4.866145 to the right, agree=0.934, adj=0.143, (0 split)  
##   
## Node number 5: 32 observations  
## predicted class=1 expected loss=0.09375 P(node) =0.2025316  
## class counts: 3 29  
## probabilities: 0.094 0.906   
##   
## Node number 8: 84 observations  
## predicted class=0 expected loss=0.04761905 P(node) =0.5316456  
## class counts: 80 4  
## probabilities: 0.952 0.048   
##   
## Node number 9: 7 observations  
## predicted class=1 expected loss=0.2857143 P(node) =0.0443038  
## class counts: 2 5  
## probabilities: 0.286 0.714

accuracy\_sample\_data = table(predict(tree, TestData, type="class"), TestData$target)  
sum(diag(accuracy\_sample\_data))/sum(accuracy\_sample\_data)\*100

## [1] 83.87097

prec = 162/175  
rec = 162/167  
fscore = 2.91  
  
  
# Here we use random forest and boosting for the same  
# 9)  
#Random Forest  
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.4.2

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

#creating the model  
replace.indicator <- TRUE  
rf = randomForest(target~., data = TrainData, ntree = 100, mtry = 3, proximity = TRUE, replace = replace.indicator,   
 sampsize = if(replace.indicator){nrow(TrainData)}else{ceiling(0.65\*nrow(trainData))}, importance = TRUE )  
print(rf)

##   
## Call:  
## randomForest(formula = target ~ ., data = TrainData, ntree = 100, mtry = 3, proximity = TRUE, replace = replace.indicator, sampsize = if (replace.indicator) { nrow(TrainData) } else { ceiling(0.65 \* nrow(trainData)) }, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 100  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 0.63%  
## Confusion matrix:  
## 0 1 class.error  
## 0 85 0 0.00000000  
## 1 1 72 0.01369863

attributes(rf)

## $names  
## [1] "call" "type" "predicted"   
## [4] "err.rate" "confusion" "votes"   
## [7] "oob.times" "classes" "importance"   
## [10] "importanceSD" "localImportance" "proximity"   
## [13] "ntree" "mtry" "forest"   
## [16] "y" "test" "inbag"   
## [19] "terms"   
##   
## $class  
## [1] "randomForest.formula" "randomForest"

accuracy\_sample\_data = table(predict(rf, TestData, type="class"), TestData$target)  
sum(diag(accuracy\_sample\_data))/sum(accuracy\_sample\_data)\*100

## [1] 95.16129

prec = 166/175  
rec = 166/167  
fscore = 2.98  
  
### To access the error rate   
plot(rf)  
rf$err.rate

## OOB 0 1  
## [1,] 0.129629630 0.06060606 0.23809524  
## [2,] 0.087912088 0.02000000 0.17073171  
## [3,] 0.054054054 0.01639344 0.10000000  
## [4,] 0.064000000 0.04411765 0.08771930  
## [5,] 0.072992701 0.06666667 0.08064516  
## [6,] 0.087837838 0.06250000 0.11764706  
## [7,] 0.079470199 0.06172840 0.10000000  
## [8,] 0.084967320 0.07317073 0.09859155  
## [9,] 0.057692308 0.02380952 0.09722222  
## [10,] 0.038461538 0.02380952 0.05555556  
## [11,] 0.044871795 0.02380952 0.06944444  
## [12,] 0.025641026 0.01190476 0.04166667  
## [13,] 0.031847134 0.02380952 0.04109589  
## [14,] 0.050955414 0.03571429 0.06849315  
## [15,] 0.025477707 0.01190476 0.04109589  
## [16,] 0.025477707 0.01190476 0.04109589  
## [17,] 0.025316456 0.01176471 0.04109589  
## [18,] 0.031645570 0.01176471 0.05479452  
## [19,] 0.031645570 0.02352941 0.04109589  
## [20,] 0.031645570 0.02352941 0.04109589  
## [21,] 0.018987342 0.01176471 0.02739726  
## [22,] 0.018987342 0.01176471 0.02739726  
## [23,] 0.018987342 0.01176471 0.02739726  
## [24,] 0.012658228 0.01176471 0.01369863  
## [25,] 0.018987342 0.01176471 0.02739726  
## [26,] 0.018987342 0.01176471 0.02739726  
## [27,] 0.012658228 0.01176471 0.01369863  
## [28,] 0.012658228 0.01176471 0.01369863  
## [29,] 0.012658228 0.01176471 0.01369863  
## [30,] 0.012658228 0.01176471 0.01369863  
## [31,] 0.012658228 0.01176471 0.01369863  
## [32,] 0.012658228 0.01176471 0.01369863  
## [33,] 0.006329114 0.01176471 0.00000000  
## [34,] 0.012658228 0.01176471 0.01369863  
## [35,] 0.006329114 0.01176471 0.00000000  
## [36,] 0.012658228 0.01176471 0.01369863  
## [37,] 0.006329114 0.01176471 0.00000000  
## [38,] 0.006329114 0.01176471 0.00000000  
## [39,] 0.012658228 0.01176471 0.01369863  
## [40,] 0.012658228 0.01176471 0.01369863  
## [41,] 0.012658228 0.01176471 0.01369863  
## [42,] 0.012658228 0.01176471 0.01369863  
## [43,] 0.012658228 0.01176471 0.01369863  
## [44,] 0.012658228 0.01176471 0.01369863  
## [45,] 0.012658228 0.01176471 0.01369863  
## [46,] 0.012658228 0.01176471 0.01369863  
## [47,] 0.012658228 0.01176471 0.01369863  
## [48,] 0.012658228 0.01176471 0.01369863  
## [49,] 0.012658228 0.01176471 0.01369863  
## [50,] 0.012658228 0.01176471 0.01369863  
## [51,] 0.012658228 0.01176471 0.01369863  
## [52,] 0.012658228 0.01176471 0.01369863  
## [53,] 0.012658228 0.01176471 0.01369863  
## [54,] 0.012658228 0.01176471 0.01369863  
## [55,] 0.012658228 0.01176471 0.01369863  
## [56,] 0.012658228 0.01176471 0.01369863  
## [57,] 0.012658228 0.01176471 0.01369863  
## [58,] 0.012658228 0.00000000 0.02739726  
## [59,] 0.012658228 0.00000000 0.02739726  
## [60,] 0.012658228 0.01176471 0.01369863  
## [61,] 0.012658228 0.01176471 0.01369863  
## [62,] 0.012658228 0.01176471 0.01369863  
## [63,] 0.006329114 0.00000000 0.01369863  
## [64,] 0.006329114 0.00000000 0.01369863  
## [65,] 0.006329114 0.00000000 0.01369863  
## [66,] 0.006329114 0.00000000 0.01369863  
## [67,] 0.006329114 0.00000000 0.01369863  
## [68,] 0.006329114 0.00000000 0.01369863  
## [69,] 0.006329114 0.00000000 0.01369863  
## [70,] 0.006329114 0.00000000 0.01369863  
## [71,] 0.006329114 0.00000000 0.01369863  
## [72,] 0.006329114 0.00000000 0.01369863  
## [73,] 0.006329114 0.00000000 0.01369863  
## [74,] 0.006329114 0.00000000 0.01369863  
## [75,] 0.006329114 0.00000000 0.01369863  
## [76,] 0.006329114 0.00000000 0.01369863  
## [77,] 0.006329114 0.00000000 0.01369863  
## [78,] 0.006329114 0.00000000 0.01369863  
## [79,] 0.006329114 0.00000000 0.01369863  
## [80,] 0.006329114 0.00000000 0.01369863  
## [81,] 0.006329114 0.00000000 0.01369863  
## [82,] 0.006329114 0.00000000 0.01369863  
## [83,] 0.006329114 0.00000000 0.01369863  
## [84,] 0.006329114 0.00000000 0.01369863  
## [85,] 0.006329114 0.00000000 0.01369863  
## [86,] 0.006329114 0.00000000 0.01369863  
## [87,] 0.006329114 0.00000000 0.01369863  
## [88,] 0.000000000 0.00000000 0.00000000  
## [89,] 0.006329114 0.00000000 0.01369863  
## [90,] 0.000000000 0.00000000 0.00000000  
## [91,] 0.000000000 0.00000000 0.00000000  
## [92,] 0.006329114 0.00000000 0.01369863  
## [93,] 0.006329114 0.00000000 0.01369863  
## [94,] 0.000000000 0.00000000 0.00000000  
## [95,] 0.000000000 0.00000000 0.00000000  
## [96,] 0.000000000 0.00000000 0.00000000  
## [97,] 0.000000000 0.00000000 0.00000000  
## [98,] 0.000000000 0.00000000 0.00000000  
## [99,] 0.000000000 0.00000000 0.00000000  
## [100,] 0.006329114 0.00000000 0.01369863

#AdaBoost  
library(adabag)

## Warning: package 'adabag' was built under R version 3.4.2

## Loading required package: mlbench

## Warning: package 'mlbench' was built under R version 3.4.2

## Loading required package: caret

## Warning: package 'caret' was built under R version 3.4.2

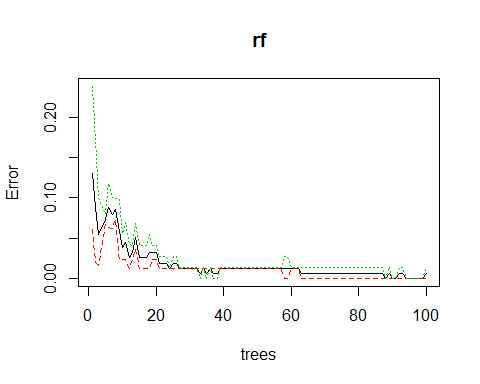
## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.2

##   
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':  
##   
## margin



rf.adaboost = boosting(target ~ ., data = TrainData, boos = TRUE, mfinal = 10)  
library(caret)  
boosting\_v = predict.boosting(rf.adaboost, newdata=TestData)  
boosting\_v$confusion

## Observed Class  
## Predicted Class 0 1  
## 0 20 1  
## 1 2 39

sum(diag(boosting\_v$confusion))/sum(boosting\_v$confusion)\*100

## [1] 95.16129

prec = 165/168  
rec= 165/167  
fscore=2.96

#10) Based on the above statistics using cart, logistic regression and random forest on sample data and complete data we can conclude that random forest has the highest accuracy. Random forest among all the techniques used gives us the highest precision, recall and f score.That accuracy improves further when we do boosting. It is very close to perfect prediction. So random forest along with boosting can be used even though its having certain limitations.