IDS 572 Data Mining for Business

HW3

Student Name	UIN
Siddartha Padala	677043564
Shiwaani Gossulaeh	670276889

Assignment is based on the Case Study - “Predicting Earnings Manipulation By Indian Firms Using Machine Learning Algorithms.”

Summary of the case study:
MCA Technology Solutions Private Limited was founded in Bangalore in 2015 with the goal of fusing business with technology and analytics. Customer intelligence, forecasting, optimization, risk assessment, web analytics, text mining, and cloud solutions are the areas where MCA Technology Solutions assisted its clients. MCA technology solutions’ risk assessment sector focused on issues including credit scoring and fraud detection. A commercial bank asked Sachin Kumar, Director at MCA Technology Solutions to help them identify earnings manipulators among the bank’s customers. The bank provided business loans to small and medium enterprises and the value of loan ranged from INR 10 million to 500 million. The bank had a suspicion that some of its clients were engaging in earnings manipulation to improve their chances of getting a loan. Saurabh Rishi, the chief data scientist at MCA Technologies was assigned the task of developing a use case for predicting earnings manipulations. He was aware of models for anticipating earnings manipulations, such as Benford’s law and the Beneish model, but he was unsure of how well they performed, particularly in the context of India. Using the data downloaded from the Prowess database maintained by the Centre of Monitoring Indian Economy(CMIE), Saurabh made the decision to create his own model for predicting earnings manipulations. Daniel obtained data on income manipulators from the Lexis Nexis database and the Securities Exchange Board of India (SEBI). To create the model, data on more than 1200 companies was gathered. In comparison to other traditional models like the Beneish model, which is used to anticipate earnings manipulation, MCA Technology believed that machine learning algorithms may provide greater accuracy.

Loading the data:

```
complete_og <- read_excel("IMB579-XLS-ENG.xlsx", sheet = "Complete Data")
sample_og <- read_excel("IMB579-XLS-ENG.xlsx", sheet = "Sample for Model Development")
mani <- read_excel("IMB579-XLS-ENG.xlsx", sheet = "Manipulator")
nonmani <- read_excel("IMB579-XLS-ENG.xlsx", sheet = "Non-Manipulator")

target_s <- as.factor(sample_og$'C-MANIPULATOR')
tab <- table(complete_og$Manipulater)
tab
```

```
##
##   No  Yes
## 1200   39
```

We see that there are 1200 non-manipulators and 39 manipulators in the complete dataset from the tab output above. The data is imbalanced as the number of non-manipulators outnumber the manipulators. This imbalance will be a problem for model creation as there might not be sufficient data to predict the manipulators.

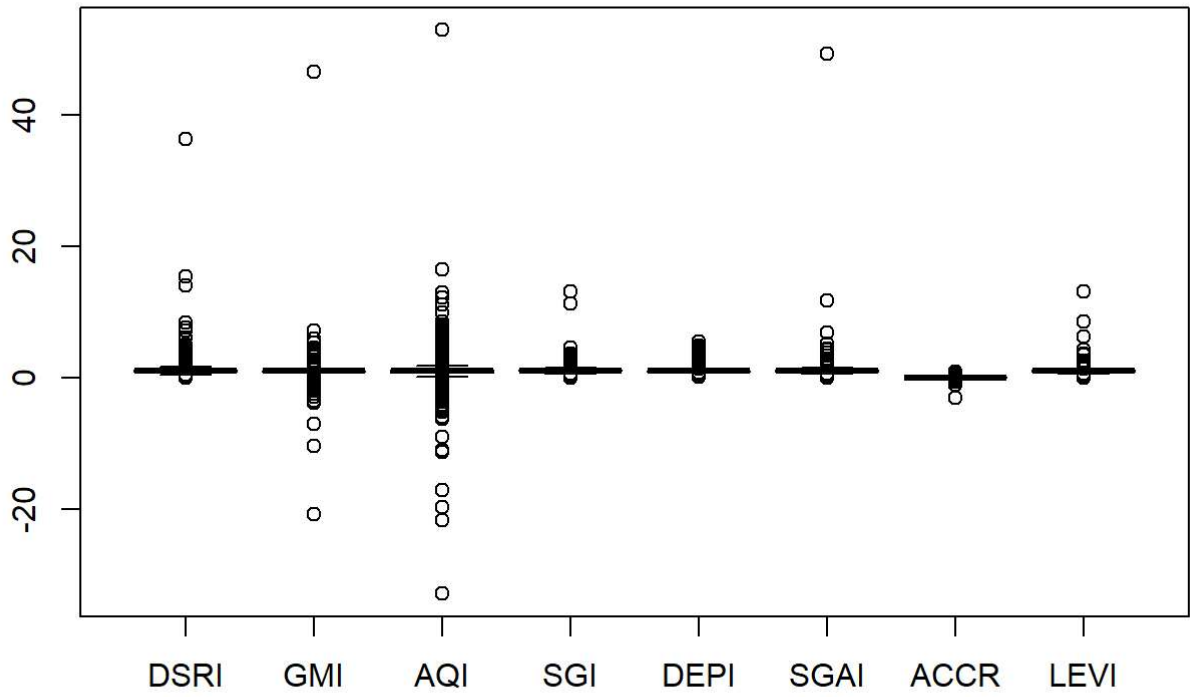
Let's check if there are any null values in the data.

Number of null values in the complete_og dataset = 0

As there are no null values, let's check for outliers.

We can check for outliers in each variable using the boxplots.

```
boxplot(complete_og[,c(-1,1-ncol(complete_og),-ncol(complete_og))])
```



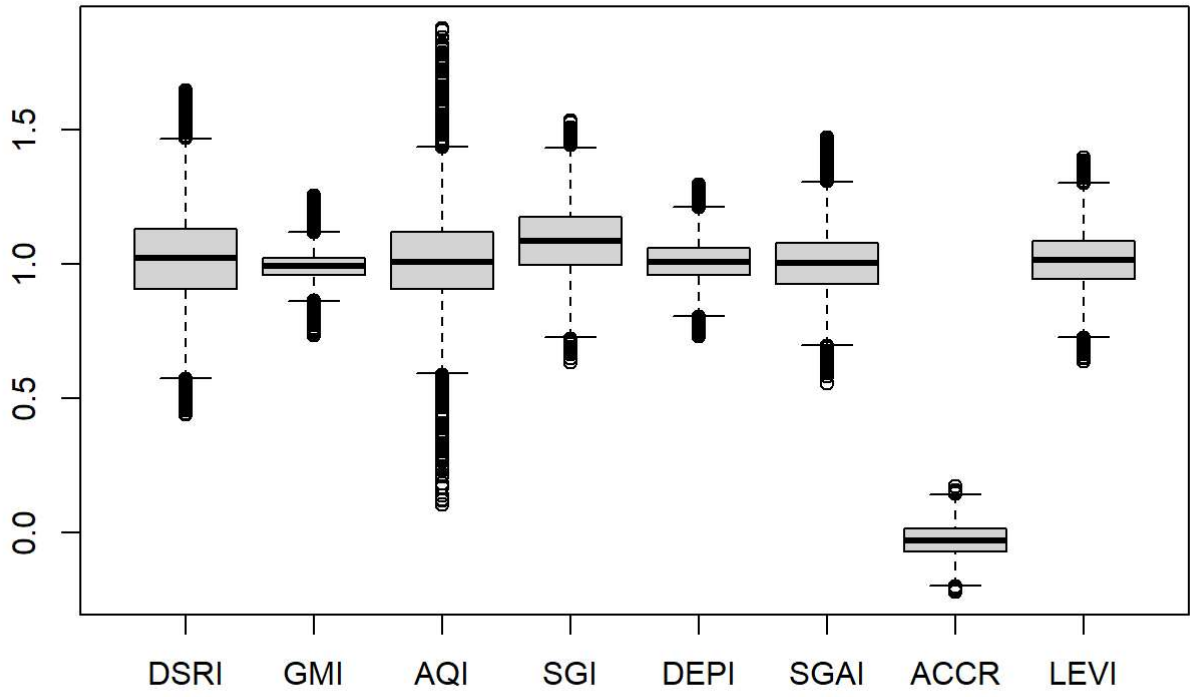
We see that most of the variables have outliers which might affect our model. We can handle outliers in many different ways. But, since we have limited number of manipulators, we can't remove the outliers from our dataset completely as we would run into the risk of not having enough data to build the model. Therefore, we normalize the outlier values by replacing them with the mean of the variable to clean the data.

```
#Saving the original database in a new dataframe
complete <- complete_og
# moving manipulator to target variable
target <- as.factor(complete$'C-MANIPULATOR')
complete <- complete[,c(-1,1-ncol(complete),-ncol(complete))]
str(complete)
```

```
## tibble [1,239 × 8] (S3: tbl_df/tbl/data.frame)
##  $ DSRI: num [1:1239] 1.62 1 1 1.49 1 ...
##  $ GMI : num [1:1239] 1.13 1.61 1.02 1 1.37 ...
##  $ AQI : num [1:1239] 7.185 1.005 1.241 0.466 0.637 ...
##  $ SGI : num [1:1239] 0.366 13.081 1.475 0.673 0.861 ...
##  $ DEPI: num [1:1239] 1.38 0.4 1.17 2 1.45 ...
##  $ SGAI: num [1:1239] 1.6241 5.1982 0.6477 0.0929 1.7415 ...
##  $ ACCR: num [1:1239] -0.1668 0.0605 0.0367 0.2734 0.123 ...
##  $ LEVI: num [1:1239] 1.161 0.987 1.264 0.681 0.939 ...
```

```
# First replacing the outliers with NA and then relacing NA with mean
value = complete$DSRI[complete$DSRI %in% boxplot.stats(complete$DSRI)$out]
complete$DSRI[complete$DSRI %in% value] <- NA
complete$DSRI[is.na(complete$DSRI)] <- mean(complete$DSRI, na.rm = TRUE)

value = complete$GMI[complete$GMI %in% boxplot.stats(complete$GMI)$out]
complete$GMI[complete$GMI %in% value] <- NA
complete$GMI[is.na(complete$GMI)] <- mean(complete$GMI, na.rm = TRUE)

value = complete$AQI[complete$AQI %in% boxplot.stats(complete$AQI)$out]
complete$AQI[complete$AQI %in% value] <- NA
complete$AQI[is.na(complete$AQI)] <- mean(complete$AQI, na.rm = TRUE)

value = complete$SGI[complete$SGI %in% boxplot.stats(complete$SGI)$out]
complete$SGI[complete$SGI %in% value] <- NA
complete$SGI[is.na(complete$SGI)] <- mean(complete$SGI, na.rm = TRUE)

value = complete$DEPI[complete$DEPI %in% boxplot.stats(complete$DEPI)$out]
complete$DEPI[complete$DEPI %in% value] <- NA
complete$DEPI[is.na(complete$DEPI)] <- mean(complete$DEPI, na.rm = TRUE)

value = complete$SGAI[complete$SGAI %in% boxplot.stats(complete$SGAI)$out]
complete$SGAI[complete$SGAI %in% value] <- NA
complete$SGAI[is.na(complete$SGAI)] <- mean(complete$SGAI, na.rm = TRUE)

value = complete$ACCR[complete$ACCR %in% boxplot.stats(complete$ACCR)$out]
complete$ACCR[complete$ACCR %in% value] <- NA
complete$ACCR[is.na(complete$ACCR)] <- mean(complete$ACCR, na.rm = TRUE)

value = complete$LEVI[complete$LEVI %in% boxplot.stats(complete$LEVI)$out]
complete$LEVI[complete$LEVI %in% value] <- NA
complete$LEVI[is.na(complete$LEVI)] <- mean(complete$LEVI, na.rm = TRUE)
```

```
boxplot(complete)
```



After cleaning up the data, we can see from the updated boxplots that the spread of the variables has decreased considerably. Most of the data is centered at the means of the vairables.

1. Do you think the Beneish model developed in 1999 will still be relevant to Indian data?

A. To answer this question, we have to know what Beneish model is
Beneish model - A mathematical model that uses financial ratios and eight variables to identify whether a company has manipulated its earnings. It is used as a tool to uncover financial fraud.

The basic theory that Beneish bases the ratio upon is that companies may be more likely to manipulate their profits if they show deteriorating gross margins, operating expenses, and leverage both rising, along with significant sales growth. These factors may cause profit manipulation through various means.

Once the eight variables are calculated, they are then combined to achieve an M-Score for the company. An M-Score of less than -1.78 suggests that the company will not be a manipulator. An M-Score of greater than -1.78 signals that the company is likely to be a manipulator.

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.

Let’s calculate M-score for the complete data.

```
beneish <- complete_og
beneish$mscore <- (-4.84+(beneish$DSRI*0.92)+(0.528*beneish$GMI)+(0.404**beneish$AQI)+(0.892*ben
eish$SGI)+(0.115*beneish$DEPI)-(0.172*beneish$SGAI)+(4.679*beneish$ACCR)-(0.327*beneish$LEVI))

beneish$pred[beneish$mscore > -1.78] <- 1
beneish$pred[beneish$mscore <= -1.78] <- 0
```

Evaluating the performance of beneish using confusion matrix.

```
t <- table(beneish$pred, beneish$'C-MANIPULATOR', dnn = c("Predicted", "Actual"))
confusionMatrix(t, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted   0   1
##         0 972  11
##         1 228  28
##
##                Accuracy : 0.8071
##                  95% CI : (0.784, 0.8287)
##     No Information Rate : 0.9685
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.143
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.71795
##             Specificity : 0.81000
##          Pos Pred Value : 0.10938
##          Neg Pred Value : 0.98881
##              Prevalence : 0.03148
##          Detection Rate : 0.02260
##    Detection Prevalence : 0.20662
##       Balanced Accuracy : 0.76397
##
##        'Positive' Class : 1
##
```

The accuracy of the model is 80.7% and the recall(Sensitivity) is 72%.

Although the accuracy of the Beneish model is high, the recall value is only 72%. Since, the class of interest is a minority class with only 39 observations we should measure the performance of the model using recall or sensitivity which is nothing but the percentage of positive cases caught. So, the beneish model predicts whether a company is manipulating it's financial performance 72% of the time which is not very useful in the Indian context.

2. 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? How can one handle these problems?

A. There is class imbalance in the data which causes the statistical algorithms to perform badly due to bias. In addition, the traditional model evaluations won't be able to accurately measure the performance of the model.

It is typically referred to as a rare event if the event that needs to be predicted is in the minority class and the event rate is less than 5%. As a result, the minority class is more likely to be incorrectly classified than the dominant class. As a result, the model's prediction accuracy is weakened.

Most of the algorithms tend to predict the majority class and ignore the minority class as insignificant or as outliers.

We can handle these problems using the below methods:
For resolving class-imbalance in data, we can use
1. Random Under Sampling - Balancing the data by randomly eliminating majority class examples.
2. Random Over Sampling - Increasing the number of instances in minority class by replicating them randomly.
3. Synthetic Minority Over Sampling Technique (SMOTE) - Creating new synthetic instances in the minority class based on the existing minority class instances.

For evaluating the performance of the models, we can use various other metrics like recall, precision, F-score etc depending on the problem that we are trying to solve.

3. Use a sample data (220 cases including 39 manipulators) and develop a logistic regression model that can be used by MCA Technologies Private Limited for predicting probability of earnings manipulation.

A. First, we have to clean the sample data by repeating the process we did on complete data.

First, let's split the data into train and test data. Then, To handle the class-imbalance, we will do both over and under sampling to the sample train data.

```
c_sample <- cbind(sample, target_s)
str(c_sample)
```

```
## 'data.frame':    220 obs. of  9 variables:
##  $ DSRI    : num  1.62 1 1 1.49 1 ...
##  $ GMI     : num  1.129 0.996 1.016 1 0.996 ...
##  $ AQI     : num  0.992 1.005 1.241 0.466 0.637 ...
##  $ SGI     : num  1.098 1.098 1.475 0.673 0.861 ...
##  $ DEPI    : num  1.01 1.01 1.17 1.01 1.01 ...
##  $ SGAI    : num  1.624 1.022 0.648 1.022 1.022 ...
##  $ ACCR    : num  -0.1668 0.0605 0.0367 -0.0223 0.123 ...
##  $ LEVI    : num  1.161 0.987 1.264 0.681 0.939 ...
##  $ target_s: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
set.seed(999)
index <- sample(2, nrow(c_sample), replace = TRUE, prob = c(0.75,0.25))
s_train <- c_sample[index ==1,]
s_test <- c_sample[index==2,]

os_sample <- ovun.sample(target_s ~., data = s_train, method = "both")$data
summary(os_sample$target_s)
```

```
##  0  1
## 83 80
```

Running forward selection for variable selection as this method gives better results consuming less amount of time.

```
# Variable selection
full <- glm(target_s ~., data = os_sample, family = binomial)
null <- glm(target_s ~1, data = os_sample, family = binomial)

# Forward Selection
sample_forward <- step(null, scope = list(lower = null, upper = full), direction = "forward")
```

We will use the call suggested by the forward selection to create the logistic regression model.

```
logit_sample <- glm(formula(sample_forward),  data = os_sample, family = binomial)
summary(logit_sample)
```

```
##
## Call:
## glm(formula = formula(sample_forward), family = binomial, data = os_sample)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9955  -0.7435  -0.1477   0.8489   1.7735
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -18.553      3.859  -4.808 1.53e-06 ***
## ACCR          19.192      3.897   4.925 8.45e-07 ***
## DSRI           4.073      1.075   3.790  0.00015 ***
## DEPI           6.943      2.099   3.308  0.00094 ***
## SGI            3.484      1.249   2.789  0.00528 **
## LEVI           3.362      2.002   1.680  0.09305 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.91  on 162  degrees of freedom
## Residual deviance: 158.31  on 157  degrees of freedom
## AIC: 170.31
##
## Number of Fisher Scoring iterations: 5
```

logit_sample is the logistic regression model developed on the sample data that can be used by MCA Technologies to predict probability of earnings manipulation.

4. What measure do you use to evaluate the performance of your logistic regression model? How does your model perform on the training and test datasets?

A. We can calculate the pvalue from the Chi-Square test for our model to see if the p-value is less than the alpha value(statistical threshold). If p-value is less than the alpha value, our model is performing well.

```
rd_sample <- summary(logit_sample)$deviance
pvalue <- 1-pchisq(rd_sample, 3)
pvalue
```

```
## [1] 0
```

From above, p-value for the model 'logit_sample' is 0 which is very small therefore we can conclude that the model is performing well.

Testing the model on Train and test data:

```
pred_s_train <- predict(logit_sample, newdata = os_sample, type = "response")
pred_s_test <- predict(logit_sample, newdata = s_test, type = "response")

s_train_op <- as.factor(ifelse(pred_s_train>0.36,1,0))
t_train <- table(os_sample$target_s,s_train_op)
sensitivity(t_train)
```

```
## [1] 0.877193
```

```
confusionMatrix(t_train)
```

```
## Confusion Matrix and Statistics
##
##    s_train_op
##      0  1
##   0 50 33
##   1  7 73
##
##                Accuracy : 0.7546
##                  95% CI : (0.6812, 0.8185)
##     No Information Rate : 0.6503
##     P-Value [Acc > NIR] : 0.00275
##
##                   Kappa : 0.5119
##
##  Mcnemar's Test P-Value : 7.723e-05
##
##             Sensitivity : 0.8772
##             Specificity : 0.6887
##          Pos Pred Value : 0.6024
##          Neg Pred Value : 0.9125
##              Prevalence : 0.3497
##          Detection Rate : 0.3067
##    Detection Prevalence : 0.5092
##       Balanced Accuracy : 0.7829
##
##        'Positive' Class : 0
##
```

```
s_test_op <- as.factor(ifelse(pred_s_test>0.2,1,0))
t_test <- table(s_test$target_s,s_test_op)
sensitivity(t_test)
```

```
## [1] 0.9411765
```

```
confusionMatrix(t_test)
```

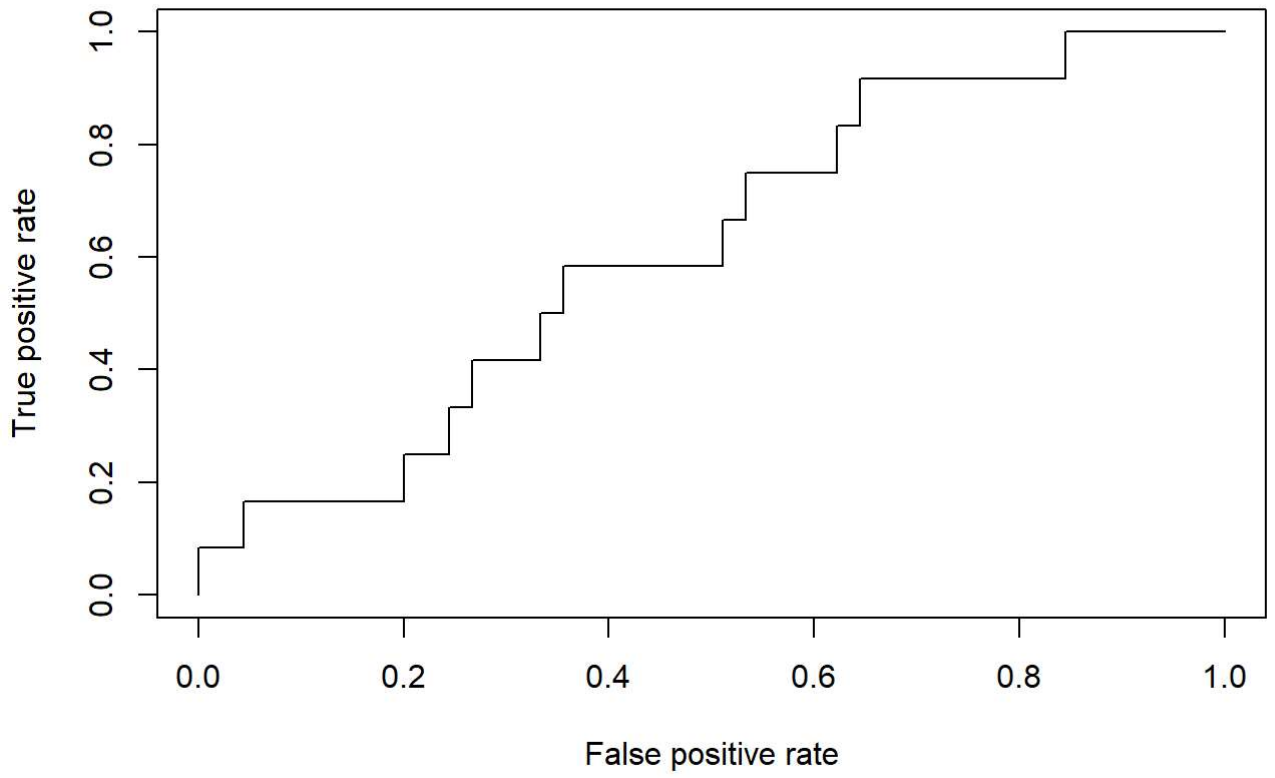
```
## Confusion Matrix and Statistics
##
##    s_test_op
##      0  1
##   0 16 29
##   1  1 11
##
##                Accuracy : 0.4737
##                  95% CI : (0.3398, 0.6103)
##     No Information Rate : 0.7018
##     P-Value [Acc > NIR] : 0.9999
##
##                   Kappa : 0.1467
##
##  Mcnemar's Test P-Value : 8.244e-07
##
##             Sensitivity : 0.9412
##             Specificity : 0.2750
##          Pos Pred Value : 0.3556
##          Neg Pred Value : 0.9167
##              Prevalence : 0.2982
##          Detection Rate : 0.2807
##    Detection Prevalence : 0.7895
##       Balanced Accuracy : 0.6081
##
##        'Positive' Class : 0
##
```

The sensitivity of logit_sample model on training data is 0.877193 and the sensitivity on test data is 0.9411765 which is much better than the beneish model as we saw in question 1.

5. What is the best probability threshold that can be used to assign instances to different classes? Write two functions that receive the output of the ROC performance function and return the best probability thresholds using the distance to (0,1) and Youden's approach respectively.

A. To obtain the best probability threshold we should use the ROC curve.

```
ROCR_spred <- prediction(pred_s_test, s_test$target_s)
ROCR_sperf <- performance(ROCR_spred, "tpr", "fpr")
plot(ROCR_sperf)
```




```
auc_sample <- performance(ROCR_spred, "auc")
unlist(slot(auc_sample, "y.values"))
```

```
## [1] 0.6166667
```

```
# Youden's index
opt_youden = function(ROCR_sperf,ROCR_spred){
  cut_ind = mapply(FUN=function(x,y,p){
    l=y-x
    ind1 = which (l == max(l))
    c(sensitivity = y[[ind1]], specificity = 1-x[[ind1]], cutoff = p[[ind1]])
  }, ROCR_sperf@x.values, ROCR_sperf@y.values, ROCR_spred@cutoffs)
  return(cut_ind)
}

prob_youden <- opt_youden(ROCR_sperf,ROCR_spred)[3,1]
prob_youden <- as.numeric(prob_youden)
prob_youden
```

```
## [1] 0.2173994
```

```
# Distance to (0,1)
opt.cut1 = function(ROCR_sperf, ROCR_spred){
  cut.ind = mapply(FUN=function(x, y, p){
    d = ((x - 0)^2 + (y-1)^2)^(1/2)
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]], cutoff = p[[ind]])
  }, ROCR_sperf@x.values, ROCR_sperf@y.values, ROCR_spred@cutoffs)
  return(cut.ind)
}

prob_01 <- opt.cut1(ROCR_sperf,ROCR_spred)[3,1]
prob_01 <- as.numeric(prob_01)
prob_01
```

```
## [1] 0.5423225
```

6. 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.

A. To identify the best M-score, we have to look at the performance after applying the youden's threshold and distance threshold.

```
s_test_op <- as.factor(ifelse(pred_s_test>prob_01,1,0))
t_test <- table(s_test$target_s,s_test_op)
recall_01 <- sensitivity(t_test)

s_test_op <- as.factor(ifelse(pred_s_test>prob_youden,1,0))
t_test <- table(s_test$target_s,s_test_op)
recall_youden <- sensitivity(t_test)

ifelse(recall_youden>=recall_01,"Youden's approach is providing better recall on the test data,
therefore this can be used as the M-score","Distance to (0,1) is providing better recall on the
test data, therefore this can be used as the M-score")
```

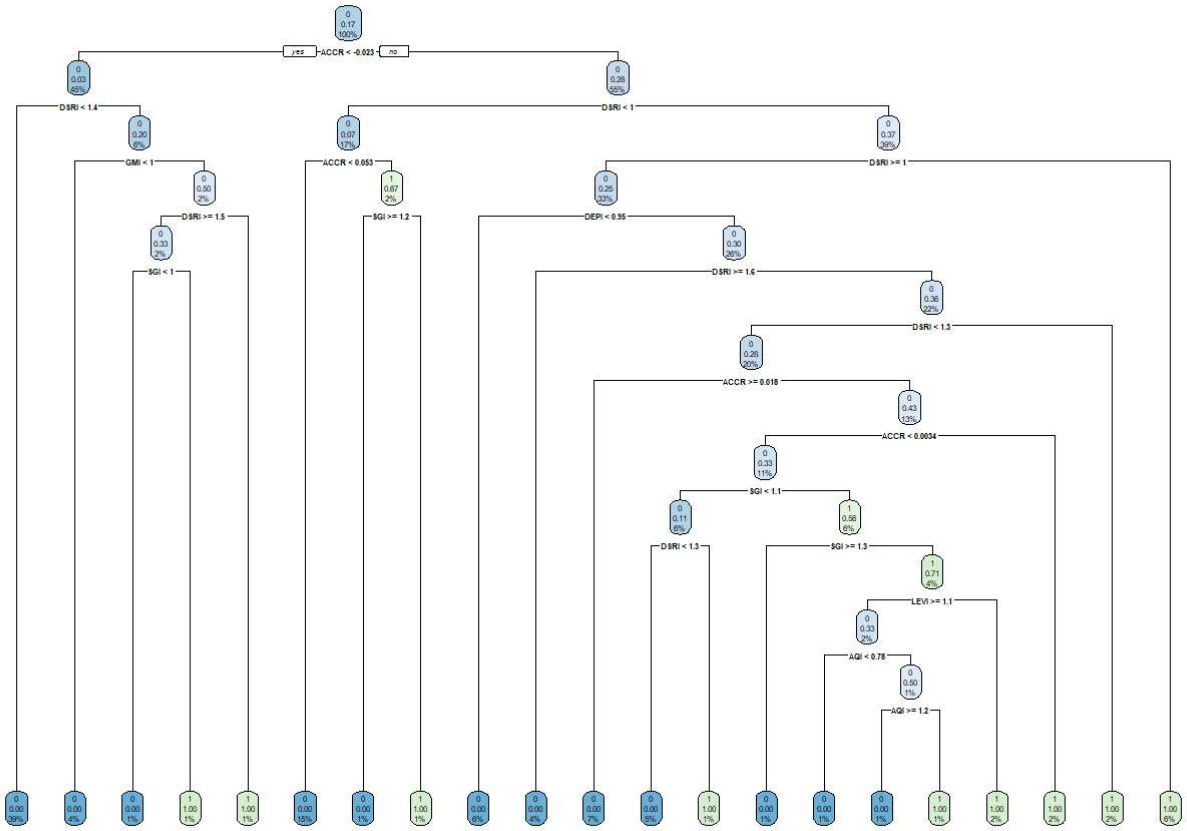
```
## [1] "Youden's approach is providing better recall on the test data, therefore this can be use
d as the M-score"
```

The best M-score is 0.2173994 which can be used by the regulators to identify potential manipulators.

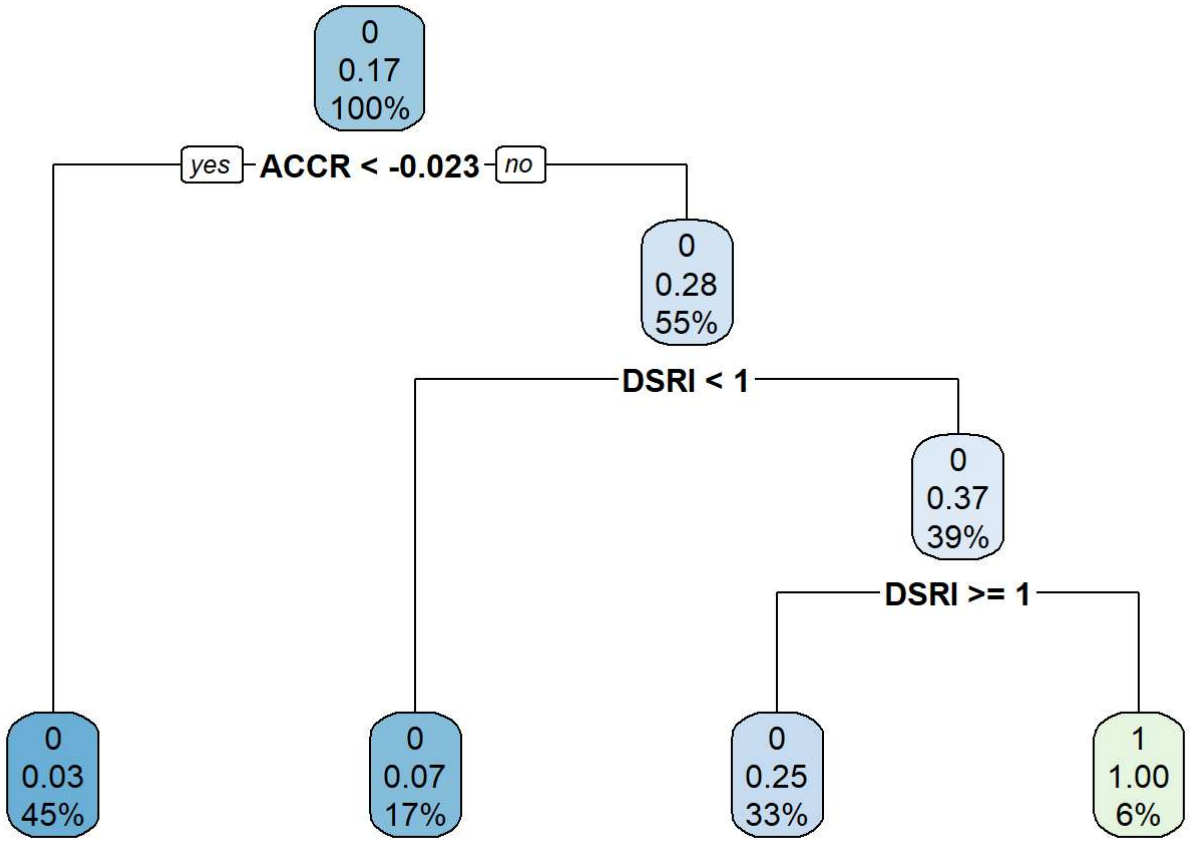
7. Develop a decision tree model. What insights do you obtain from the tree model?

A. Decision tree model for the sample data

```
# Constructing full tree
man_tree <- rpart(target_s ~., data = s_train,
                  parms = list(split = "information"),
                  control = rpart.control(minbucket = 0, minsplit = 0, cp = -1))

rpart.plot(man_tree)
```




```
mincp_i <- which.min(man_tree$cptable[,"xerror"])
optCP <- man_tree$cptable[mincp_i,"CP"]
mantree_pruned <- prune(man_tree, cp = optCP)

rpart.plot(mantree_pruned)
```




```
tree_pred_train <- predict(mantree_pruned, newdata = s_train, type = "class")
tree_pred_test <- predict(mantree_pruned, newdata = s_test, type = "class")

sensitivity(tree_pred_train,os_sample$target_s)
```

```
## [1] 0.8795181
```

```
sensitivity(tree_pred_test,s_test$target_s)
```

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

```
confusionMatrix(tree_pred_train, os_sample$target_s)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 73 80
##          1 10  0
##
##                Accuracy : 0.4479
##                  95% CI : (0.37, 0.5276)
##     No Information Rate : 0.5092
##     P-Value [Acc > NIR] : 0.9501
##
##                   Kappa : -0.1224
##
##  Mcnemar's Test P-Value : 3.51e-13
##
##             Sensitivity : 0.8795
##             Specificity : 0.0000
##          Pos Pred Value : 0.4771
##          Neg Pred Value : 0.0000
##              Prevalence : 0.5092
##          Detection Rate : 0.4479
##    Detection Prevalence : 0.9387
##       Balanced Accuracy : 0.4398
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(tree_pred_test, s_test$target_s)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 45  8
##          1  0  4
##
##                Accuracy : 0.8596
##                  95% CI : (0.7421, 0.9374)
##     No Information Rate : 0.7895
##     P-Value [Acc > NIR] : 0.12519
##
##                   Kappa : 0.4412
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.3333
##          Pos Pred Value : 0.8491
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7895
##          Detection Rate : 0.7895
##    Detection Prevalence : 0.9298
##       Balanced Accuracy : 0.6667
##
##        'Positive' Class : 0
##
```

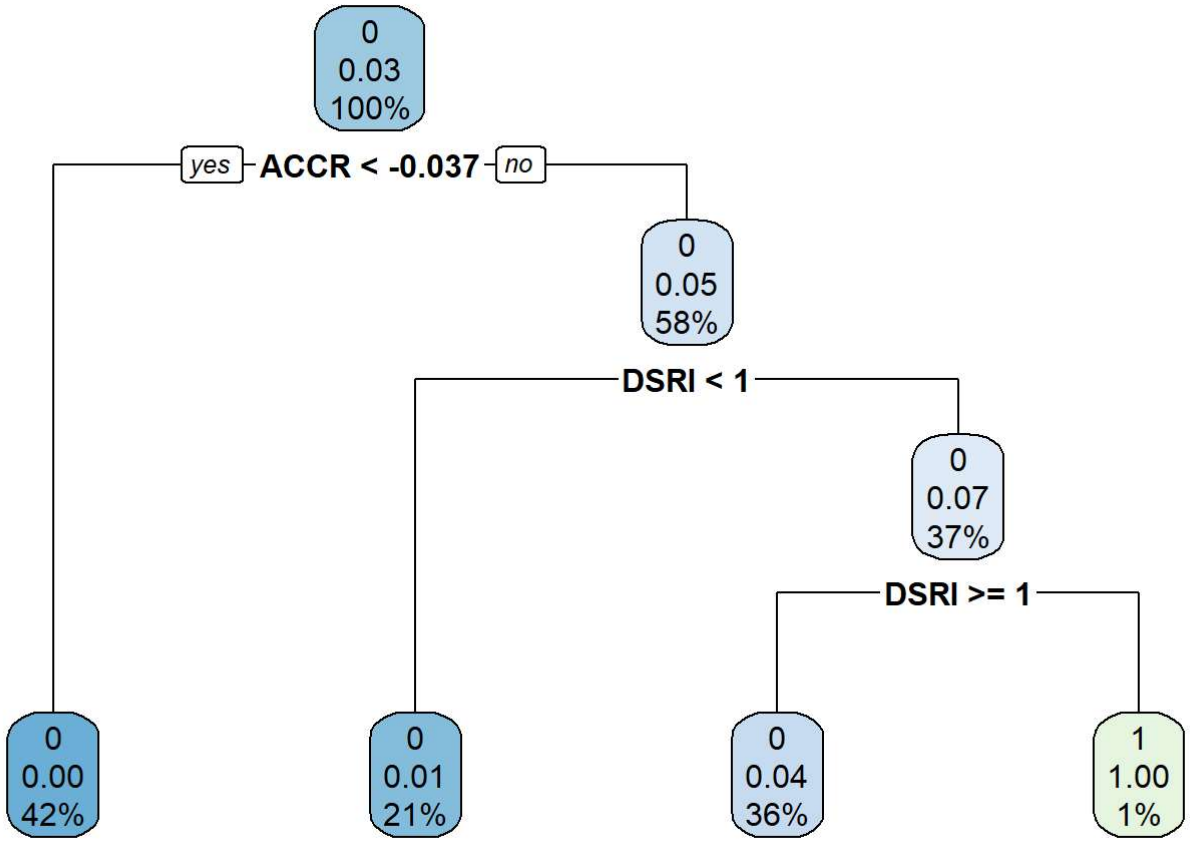
As we can see, the recall values are very high compared to the logistic regression model even though the sampling is not done on the training data. Decision tree model is giving a better prediction algorithm compared to logistic regression at least on the sample data.

Let's check for complete database:

```
# Splitting Complete data into train and test subsets

n_complete <- cbind(complete, target)
str(n_complete)
```

```
## 'data.frame':    1239 obs. of  9 variables:
##  $ DSRI  : num  1.62 1 1 1.49 1 ...
##  $ GMI   : num  1.129 0.993 1.016 1 0.993 ...
##  $ AQI   : num  1.006 1.005 1.241 0.466 0.637 ...
##  $ SGI   : num  1.085 1.085 1.475 0.673 0.861 ...
##  $ DEPI  : num  1.01 1.01 1.17 1.01 1.01 ...
##  $ SGAI  : num  1.003 1.003 0.648 1.003 1.003 ...
##  $ ACCR  : num  -0.1668 0.0605 0.0367 -0.0283 0.123 ...
##  $ LEVI  : num  1.161 0.987 1.264 0.681 0.939 ...
##  $ target: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 ...
```

```
#Splitting into train and test data
set.seed(999)
index <- sample(2, nrow(n_complete), replace = TRUE, prob = c(0.75,0.25))
c_train <- n_complete[index ==1,]
c_test <- n_complete[index==2,]

# Decision tree on complete data
cman_tree <- rpart(target ~., data = c_train,
                   parms = list(split = "information"),
                   control = rpart.control(minbucket = 0, minsplit = 0, cp = -1))

mincp_i <- which.min(cman_tree$cptable[,"xerror"])
optCP <- cman_tree$cptable[mincp_i,"CP"]
cmantree_pruned <- prune(cman_tree, cp = optCP)

rpart.plot(cmantree_pruned)
```




```
ctree_pred_train <- predict(cmantree_pruned, newdata = c_train, type = "class")
ctree_pred_test <- predict(cmantree_pruned, newdata = c_test, type = "class")

sensitivity(ctree_pred_train,c_train$target)
```

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

```
tc_recall <- sensitivity(ctree_pred_test,c_test$target)
tc_recall
```

```
## [1] 0.9932432
```

From above, we again see that the recall values accuracy of the decision tree model are extremely superior compared to the logistic regression model. But the output of the decision tree might be unstable and since the data seems to be linearly related to the target logistic should be a stable acceptable solution for this problem.

8. 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 and comment on differences.

A. Developing logistic regression model for the complete dataset.

```
#applying over and under sampling on the complete data to resolve class-imbalance
os_complete <- ovun.sample(target ~., data = c_train, method = "both")$data
summary(os_complete$target)
```

```
##   0   1
## 490 441
```

```
# Variable selection using forward selection
full <- glm(target ~., data = os_complete, family = binomial)
null <- glm(target ~1, data = os_complete, family = binomial)

complete_forward <- step(null, scope = list(lower = null, upper = full), direction = "forward")
```

```
## Start:  AIC=1290.06
## target ~ 1
##
##        Df Deviance    AIC
## + ACCR  1   1179.2 1183.2
## + DSRI  1   1252.5 1256.5
## + SGAI  1   1257.4 1261.4
## + DEPI  1   1265.1 1269.1
## + SGI   1   1270.0 1274.0
## + AQI   1   1272.2 1276.2
## + LEVI  1   1281.0 1285.0
## + GMI   1   1284.3 1288.3
## <none>      1288.1 1290.1
##
## Step:  AIC=1183.19
## target ~ ACCR
##
##        Df Deviance    AIC
## + DSRI  1   1115.6 1121.6
## + SGAI  1   1147.1 1153.1
## + DEPI  1   1160.0 1166.0
## + LEVI  1   1160.3 1166.3
## + AQI   1   1165.8 1171.8
## + SGI   1   1173.4 1179.4
## <none>      1179.2 1183.2
## + GMI   1   1177.9 1183.9
##
## Step:  AIC=1121.56
## target ~ ACCR + DSRI
##
##        Df Deviance    AIC
## + SGAI  1   1081.3 1089.3
## + LEVI  1   1099.5 1107.5
## + DEPI  1   1099.5 1107.5
## + SGI   1   1101.1 1109.1
## + AQI   1   1104.0 1112.0
## <none>      1115.6 1121.6
## + GMI   1   1115.1 1123.1
##
## Step:  AIC=1089.31
## target ~ ACCR + DSRI + SGAI
##
##        Df Deviance    AIC
## + LEVI  1   1067.4 1077.4
## + DEPI  1   1067.5 1077.5
## + AQI   1   1069.7 1079.7
## + SGI   1   1070.5 1080.5
## <none>      1081.3 1089.3
## + GMI   1   1081.1 1091.1
##
## Step:  AIC=1077.44
## target ~ ACCR + DSRI + SGAI + LEVI
```

```
##
##        Df Deviance    AIC
## + DEPI  1   1056.4 1068.4
## + AQI   1   1057.0 1069.0
## + SGI   1   1060.2 1072.2
## <none>      1067.4 1077.4
## + GMI   1   1067.2 1079.2
##
## Step:  AIC=1068.38
## target ~ ACCR + DSRI + SGAI + LEVI + DEPI
##
##        Df Deviance    AIC
## + SGI   1   1045.5 1059.5
## + AQI   1   1048.0 1062.0
## <none>      1056.4 1068.4
## + GMI   1   1056.2 1070.2
##
## Step:  AIC=1059.46
## target ~ ACCR + DSRI + SGAI + LEVI + DEPI + SGI
##
##        Df Deviance    AIC
## + AQI   1   1036.0 1052.0
## <none>      1045.5 1059.5
## + GMI   1   1045.3 1061.3
##
## Step:  AIC=1051.96
## target ~ ACCR + DSRI + SGAI + LEVI + DEPI + SGI + AQI
##
##        Df Deviance    AIC
## <none>      1036.0 1052.0
## + GMI   1   1035.6 1053.6
```

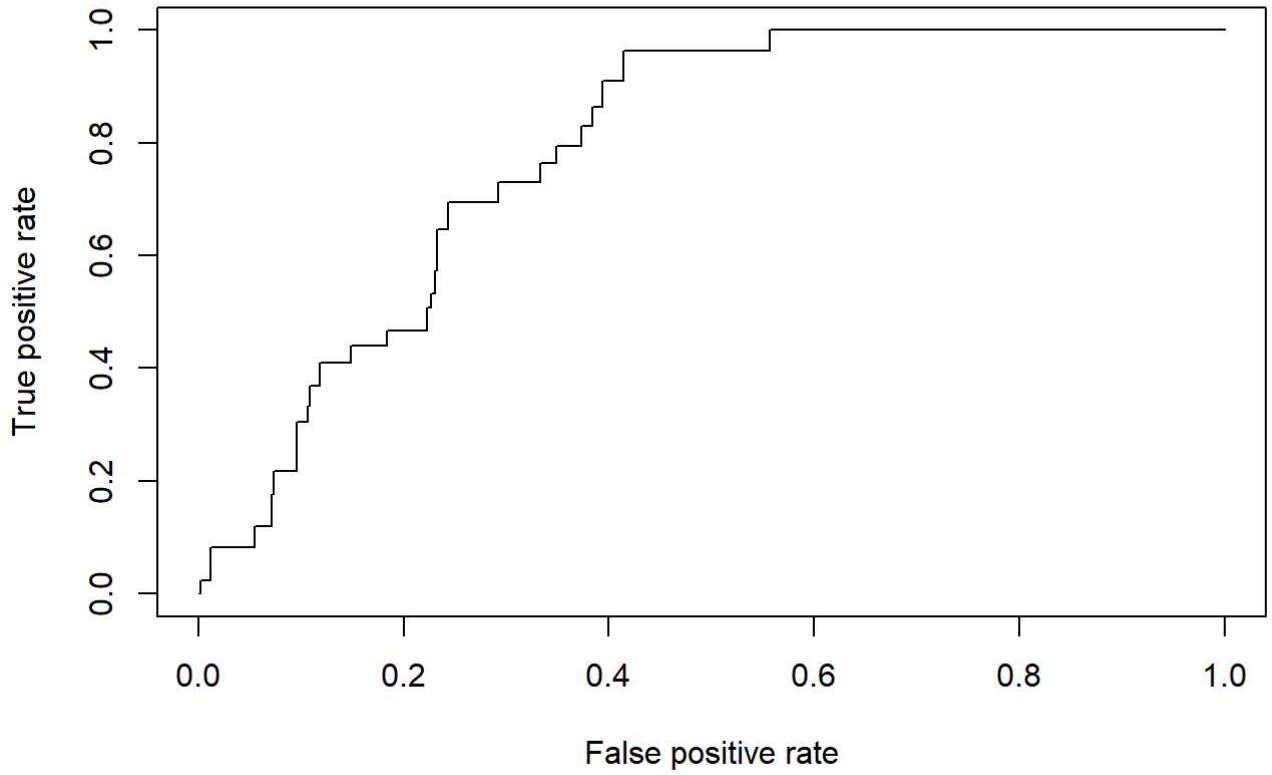
```
# logistic model
logit_complete <- glm(formula(complete_forward), data = os_complete, family = binomial)

rd_complete <- summary(logit_complete)$deviance
rd_complete
```

```
## [1] 1035.958
```

```
pvalue_c <- 1-pchisq(rd_complete, 3)
pvalue_c
```

```
## [1] 0
```

```
# predicting on train and test data
pred_c_train <- predict(logit_complete, newdata = os_complete, type = "response")
pred_c_test <- predict(logit_complete, newdata = c_test, type = "response")

#ROC curve
ROCR_cpred <- prediction(pred_c_train, os_complete$target)
ROCR_cperf <- performance(ROCR_cpred, "tpr", "fpr")
plot(ROCR_cperf)
```




```
auc_complete <- performance(ROCR_cpred, "auc")
unlist(slot(auc_complete, "y.values"))
```

```
## [1] 0.7885187
```

```
prob_01_c <- opt.cut1(ROCR_cperf,ROCR_cpred)[3,1]
prob_01_c <- as.numeric(prob_01_c)

prob_youden_c <- opt_youden(ROCR_cperf,ROCR_cpred)[3,1]
prob_youden_c <- as.numeric(prob_youden_c)

c_test_op <- as.factor(ifelse(pred_c_test>prob_01_c,1,0))
t_test_c <- table(c_test$target,c_test_op)
recall_c01 <- sensitivity(t_test_c)

c_test_op <- as.factor(ifelse(pred_c_test>prob_youden_c,1,0))
t_test_c <- table(c_test$target,c_test_op)
recall_cyouden <- sensitivity(t_test_c)

ifelse(recall_cyouden >= recall_c01, "Youden's approach is giving better performance threshold o
n the complete dataset", "Distance to (0,1) approach is giving better performance threshold on t
he complete dataset")
```

```
## [1] "Youden's approach is giving better performance threshold on the complete dataset"
```

the performance of the model has improved after training on the complete dataset

The M-score after constructing model using complete dataset is 0.3712968

Comparing the logistic model built using complete data with the decision tree built with complete data, Logistic model built on complete data is worse at predicting manipulators than the decision tree