# IDS 572 – Predicting Earnings Manipulation by Indian Firms

---

## QUESTION 1a - Do you think the Beneish model developed in 1999 will still be relevant to Indian data?

**Answers 1 A)**

Beneish model was developed in late 90s and was based on the US data to predict accounting fraud, the case study suggests that the chief Data scientist at MCA Saurabh Rishi was able to predict manipulators on Indian companies' data using other machine learning models with better accuracy than Beneish.

Also, the M-score we are getting using Machine learning model is different from Benish model M-score. Hence, we think the Beneish model developed in 1999 is not relevant to Indian data.

## QUESTION 1b - The number of manipulators is usually much less than non-manipulators. 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?

**Answers 1 B)**

- In the given companies' dataset, we see that number of Manipulators constitute only 4% of all the observations resulting in a highly unbalanced dataset. If this class imbalance is not handled, any classifier trained can result in a high bias over the majority class.
- In situations where there exists a huge class imbalance in a binary classification problem, the classifier if trained on such dataset would be extremely bias towards the majority class.
- Some of the models which are robust to class imbalance are radial Support Vector Machines and Ada-boost ensemble classifiers.
- We can handle the class imbalance by either doing Oversampling or Under sampling techniques. These techniques ensure the classes are balanced in the data for training a classifier.

1. Resampling Techniques:
    - Random Under-Sampling:
        - It involves randomly selecting examples from the majority class and deleting them from the training dataset.
        - It can result in losing information invaluable to a model.
    - Random Over-Sampling:

        - It involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset.
        - Random oversampling duplicates examples from the minority class in the training dataset and can result in overfitting for some models
    - Cluster-based Over-Sampling:
        - This approach uses K-mean clustering.
        - The clustering algorithm is applied to both the majority class and the minority class in which each class is oversampled, such that each class has the same number of data elements.
        - Though this is an efficient method, it suffers from the issue of overfitting.
    - SMOTE: Synthetic Minority Over-sampling Technique:
        - It is used to synthetically oversample the minority class.
        - A random example from the minority class is first chosen. Then k of the nearest neighbors for that example are found (typically k=5). A randomly selected neighbor is chosen, and a synthetic example is created at a randomly selected point between the two examples in feature space.
2. Ensemble Techniques:
    - Boosting based techniques:
    - AdaBoost:
        - AdaBoost helps you combine multiple weak classifiers into a single strong classifier
        - It requires the users to specify the weak learners or randomly generates the weak learners
        - AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well
    - Gradient Tree Boosting:
        - Each model is trained sequentially
        - It involves 3 elements: A loss function to be optimized, a weak learner to make predictions, an additive model to add weak learners to minimize the loss function.
    - XG Boost:
        - Extreme Gradient Boosting is an advanced implementation of Gradient boosting
        - It splits up to the maximum depth specified and prunes the tree backward
    - Bagging based techniques:
        - Different training samples are generated with replacement, and then each sample is trained using bootstrapped algorithm and the results are aggregated
        - Reduces overfitting and variance

**QUESTION 1c - Use a sample data (220 cases including 39 manipulators) and apply an undersampling technique to balance your data. Then 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.**

## Answer 1 C)

Step 1: Choosing a sample dataset

```
###################### Choosing the sample dataset #################
df_sample <- read_excel("Predict_Earning_Manipulation.xlsx", sheet = "Sample for Model Development")
dim(df_sample)
#220 rows and 11 columns
colnames(df_sample)[11] <- "C_MANIPULATOR"
#Removing ID and C-MANIPULATOR columns
df_sample$`Company ID` <- NULL
df_sample$Manipulator <- NULL
#Convert Manipulator into Factor
df_sample$C_MANIPULATOR <- as.factor(df_sample$C_MANIPULATOR)
str(df_sample)
dim(df_sample)
#220 rows and 9 columns
```

Step 2: Under sample the sample dataset

```
#################### UNDERSAMPLING THE DATASET #####################

#We use randomised undersampling technique
install.packages("ROSE")
library(ROSE)
dfs <- ovun.sample(C_MANIPULATOR ~ ., data = df_sample, method = "under", N = 100, seed = 1)$data
```

Step 3: Split into train & test split

```
#creating test and train splits
split <- sample(nrow(dfs), 0.7*nrow(dfs))

dfs_train <- dfs[split,]
dfs_test <- dfs[-split,]

table(dfs_train$C_MANIPULATOR) #46 0s and 24 1s
table(dfs_test$C_MANIPULATOR) # 15 0s and 15 1s
```

The sampled train data has 46 non-manipulators and 24 Manipulators.

Step 4: Train a stepwise Logistic Regression model

```r
################ STEPWISE LOGISTIC REGRESIION ####################
step_mod <- stepAIC(glm(C_MANIPULATOR~., data=dfs_train, family="binomial"))
summary(step_mod)

#Coefficients:
#               Estimate Std. Error z value Pr(>|z|)
#(Intercept)    -7.0554     2.1684  -3.254  0.00114 **
#   DSRI         0.8285     0.2880   2.877  0.00402 **
#   GMI          1.5997     0.9153   1.748  0.08051 .
#   AQI          0.4246     0.2938   1.445  0.14837
#   SGI          2.2892     1.0672   2.145  0.03195 *
#   ACCR         6.2375     2.1698   2.875  0.00404 **
```

Output Model summary

```
#Coefficients:
#               Estimate Std. Error z value Pr(>|z|)
#(Intercept)    -7.0554     2.1684  -3.254  0.00114 **
#   DSRI         0.8285     0.2880   2.877  0.00402 **
#   GMI          1.5997     0.9153   1.748  0.08051 .
#   AQI          0.4246     0.2938   1.445  0.14837
#   SGI          2.2892     1.0672   2.145  0.03195 *
#   ACCR         6.2375     2.1698   2.875  0.00404 **
```

Step 5: We try the model on various samples and compare the results to choose the statistically significant variables

```r
#We build the model on different samples of the dataset for identifying the significant variables

#Sample 2
dfs2 <- ovun.sample(C_MANIPULATOR ~ ., data = df_sample, method = "under", N = 78, seed = 123)$data
step_mod2 <- stepAIC(glm(C_MANIPULATOR~., data=dfs2, family="binomial"))
summary(step_mod2)

#Sample 3
dfs3 <- ovun.sample(C_MANIPULATOR ~ ., data = df_sample, method = "under", N = 78, seed = 44)$data
step_mod3 <- stepAIC(glm(C_MANIPULATOR~., data=dfs3, family="binomial"))
summary(step_mod3)
#41.597

dfs4 <- ovun.sample(C_MANIPULATOR ~ ., data = df_sample, method = "under", N = 100, seed = 24)$data
step_mod4 <- stepAIC(glm(C_MANIPULATOR~., data=dfs4, family="binomial"))
summary(step_mod4)
#47.461

#From the models built using four different samples, following features are found to be significant are
#ACCR SGI AQI DSRI GMI
```

**ANSWERS: The statistically significant variables are:**

## ACCR, SGI, AQI, DSRI, and GMI

**Probability Formulas using the Logistic Regression output**

Using C_MANIPULATOR = 0 as the reference class

Log Odds(1) = ln( $\frac{P(C_{MANIPULATOR}==1)}{P(C_{MANIPULATOR}==0)}$)

= -7.0554 + 0.8285 (DSRI) + 1.5997(GMI) + 0.4246 (AQI) + 2.2892 (SGI) + 6.2375 (ACCR)

P(C_MANIPULATOR=1) = $\frac{\exp(L(1))}{1+\exp(L(0))}$

P(C_MANIPULATOR=0) = $\frac{1}{1+\exp(L(0))}$

## QUESTION 1d - Comment on the model developed; how do you evaluate your model? Do you think the cutoff probability of 0.5 results in a good model? Try different cut-off points and see how the performance of your model change.

## Answer 1d)

Step 1: Evaluating the model performance

```
#Predictions on test-data
pred <- predict(step_mod, newdata = dfs_test, type = "response")
pred <- ifelse(pred > 0.5, 1,0)
confusionMatrix(table(pred, dfs_test$C_MANIPULATOR))

#Accuracy : 0.8333
#Sensitivity : 1.000
#Specificity : 0.6667
```

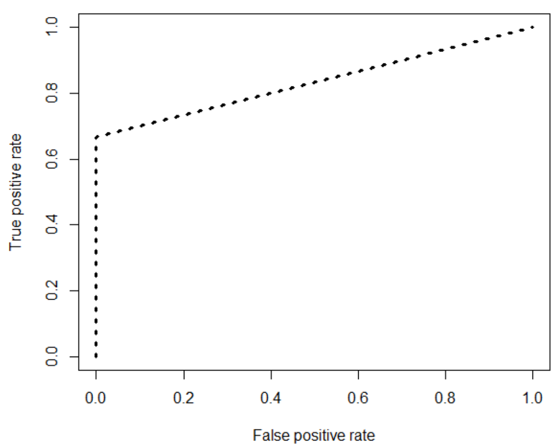
**Accuracy: 0.8333**
**Sensitivity: 1.000**
**Specificity: 0.6667**

**ANSWERS: Comments on the model built**

- The stepwise logistic model developed on a balanced dataset gives an accuracy of 83.3% on the test data
- It also has a Sensitivity value of 0.6667 for the chosen cut-off value of 0.5

Step 2: We plot a ROC curve to understand and evaluate the performance of the model

```
#We plot a ROC curve to understand and evaluate the performance of the model and
#also to identify the optimal cut-off point

install.packages("ROCR")
library(ROCR)

pr <- predict.glm(step_mod, dfs_test, type = "response")
pr <- round(pr)

pred_roc = prediction(pr, dfs_test$C_MANIPULATOR)
perf_roc = performance(pred_roc,"tpr","fpr")

#PLOTTING ROC Curve
plot(perf_roc, col = "black", lty = 3, lwd = 3)
```




Step 3: Choosing a cut-off point

```r
#Trying cut-off points
#We choose different cut-off points and try to evaluate the performance of the model

#cut off 0.4
pred2 <- predict(step_mod, newdata = dfs_test, type = "response")
pred2 <- ifelse(pred2 > 0.4, 1,0)
confusionMatrix(table(pred2, dfs_test$C_MANIPULATOR))
#Accuracy 86.67%
#Sensitivity 0.9333
#Specificity 0.8

#cut-off 0.6
pred3 <- predict(step_mod, newdata = dfs_test, type = "response")
pred3 <- ifelse(pred3 > 0.6, 1,0)
confusionMatrix(table(pred3, dfs_test$C_MANIPULATOR))
#Accuracy 73.33%
#Sensitivity 1.000
#Specificity 0.4667

#cut-off 0.3
pred4 <- predict(step_mod, newdata = dfs_test, type = "response")
pred4 <- ifelse(pred4 > 0.3, 1,0)
confusionMatrix(table(pred4, dfs_test$C_MANIPULATOR))

#Accuracy 86.67%
#Sensitivity 0.8667
#Specificity 0.8667
```

**ANSWERS:**

We see that increasing the cut-off from 0.5 to 0.6 decreased the accuracy by 10%
We see that decreasing the cut-off from 0.5 to 0.4 increased the accuracy by 3%.
However, the cut off value of 0.3 has a balanced value for Sensitivity and Specificity compared to 0.4 cut-off value. Therefore, the optimal cut-off point would be 0.3 for the model on the sample data.

## QUESTION 1e - 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 . (2) Cost-based method

## Answer 1e)

Step 1: We compute the Youden's Index for various cut-off points using our model

```
######################### QUESTION 1E ############################

#We choose different cut off values and find the max Youden's Index

yIndex <- c()
j=1
for(i in c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)){
  predvals <- ifelse(predict(step_mod, newdata=dfs_test, type="response")>i,1,0)
  conf_mat <- table(predvals, dfs_test$C_MANIPULATOR)
  yIndex[j] <- (specificity(conf_mat) + sensitivity(conf_mat) -1)
  j=j+1
}
```

Step 2: We find the maximum value of Youden's Index and choose the corresponding cut-off probability

```
> max(yIndex)
[1] 0.7333333
> #0.7333 is the max value for Youden's index
> yIndex
[1] 0.4000000 0.6666667 0.7333333 0.7333333 0.6666667 0.4666667 0.3333333 0.2666667 0.2000000
```

**ANSWERS:**

Maximum value for **Youden's Index is 0.7333** and the corresponding cut-off probability value is **0.3 & 0.4.** However, we choose **0.3** as the optimal one as we are getting equally good sensitivity & specificity on **0.3**.

We compute cost-probability similarly using Cost-based method

Step 1: We choose the penalties for the misclassifications

- The case study suggests that accuracy is important measure and misclassification of manipulators as non-manipulators is equally alarming as misclassifying a non-manipulator as a manipulator
- Therefore, we penalize both the misclassifications with equal weights of 0.5

Step 2: Finding the cut-off probability for which the cost has the minimum value

```
#Cost-based method
#The case study suggests that accuracy is important measure and misclassification of manipulators as
#non-manipulators is equally alarming as misclassifying a non-manipulator as a manipulator

#Threfore, we penalize both the misclassifications with equal weights of 0.5
#Finding the cut-off probability for which the cost has the minimum value

cost <- c()
j=1
for(i in c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9)){
  predvals <- ifelse(predict(step_mod, newdata=dfs_test, type="response")>i,1,0)
  cnf <- table(predvals, dfs_test$C_MANIPULATOR)
  cost[j] <- (0.5*(cnf[2,1]/(cnf[2,1] + cnf[2,2])) + 0.5*(cnf[1,2]/(cnf[1,2] + cnf[1,1])))
  j=j+1
}
```

Step 3: Find the minimum cost value

```
min(cost)
cost
#The minimum cost is 0.125 for cut off values of 0.2 and 0.5.
#Since, cut-off 0.5 gives better model performance, we choose cut-off value to be 0.5
```

**ANSWERS:**

The minimum cost is 0.125 for cut off values of 0.2 and 0.5.

Since, cut-off 0.5 gives better model performance, we choose cut-off value to be **0.5.**

## QUESTION 1f - 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.

## Answer 1f)

Step 1: Find the linear combination of significant variables to compute M-score

```
#In order to determine the M-score we go back to our model and the co-efficient values
#of the significant variables
summary(step_mod)

#Coefficients:
#              Estimate
#(Intercept)    -7.0554
#  DSRI          0.8285
#  GMI           1.5997
#  AQI           0.4246
#  SGI           2.2892
#  ACCR          6.2375


#We determine the M-score by substituting the key predictors in a linear equation and predict the
#resulting values.

#For cut-off=0.5, we predict the C_MANIPULATOR values on the test data
dt <- dfs_test

#Predict on test_data
dt$C_MANIPULATOR <- predict(step_mod, newdata = dt, type = "response")
dt$C_MANIPULATOR <- ifelse(dt$C_MANIPULATOR > 0.5, 1,0)

#We compute the m-score by creating a new feature on the test dataset by linearly combining
#the key predictors found from the model using their co-efficient and intercept values
dt$m_score <- -7.0554 + dt$DSRI*0.8285 + dt$GMI*1.5997 + dt$AQI*0.4246 + dt$SGI*2.2892 +
              dt$ACCR*6.2375
```

Step 2: We observe the range of M-scores for both the classes

```
#Now, we compare the values of m_score and the predicted C_MANIPULATOR values
summary(dt[dt[,9]==1,'m_score'])
#Min.   1st Qu.  Median    Mean   3rd Qu.    Max.
#0.2196  0.3889  0.9037  3.7002   2.5629  22.5568
summary(dt[dt[,9]==0,'m_score'])
#    Min.  1st Qu.   Median     Mean   3rd Qu.     Max.
#-6.89833 -2.31394 -1.55922 -1.71584  -0.91776 -0.03211

#We see the range of m_scores for both Manipulators and Non-manipulators
range(dt[dt[,9]==0,'m_score'])
# -6.89833100 -0.03210547
range(dt[dt[,9]==1,'m_score'])
# 0.2196122 22.5567557

#From the above range, we can conclude that
#Any company having a m_score > 0.22 can be considered as Manipulator.
```

Output:

```
#We see the range of m_scores for both Manipulators and Non-manipulators
range(dt[dt[,9]==0,'m_score'])
# -6.89833100 -0.03210547
range(dt[dt[,9]==1,'m_score'])
# 0.2196122 22.5567557
```

**ANSWERS:**

From the above range, we can conclude that any company having a **M_score > 0.22** can be considered as Manipulator.

## QUESTION 1g - Develop classification 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.

## Answer 1g)

Performing CART on complete dataset

Step 1 – Loading the dataset, converting variable data type and checking how imbalanced the data is –

```
str(E_Manipulator)
#Changing variable type of target to factor
E_Manipulator$Cmanipulator<- as.factor(E_Manipulator$Cmanipulator)

#Checking how imbalanced the data is
table(E_Manipulator$Cmanipulator)
#0     | 1
#1200   39
#Just 39 instances of manipulator against 1200 instances of non manipulators.
```

Step 2 – Splitting dataset into Train & Test

```
#Forming Train & Test model
set.seed(123)
ind <- sample(2, nrow(E_Manipulator), replace=TRUE, prob=c(0.7, 0.3))
EM.train <- E_Manipulator[ind==1,]
EM.test <- E_Manipulator[ind==2,]
```

Step 3 - Since dataset is highly imbalanced, applying SMOTE technique to balance out the train data (Fitting imbalanced dataset to model will give us spurious results

```
#Performing Smote on Train data since the data is quite imbalanced
install.packages("ROSE")
library(ROSE)
data.rose <- ROSE(Cmanipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data=EM.train ,seed = 123)$data
```

Step 4 – Fitting Decision tree using rpart

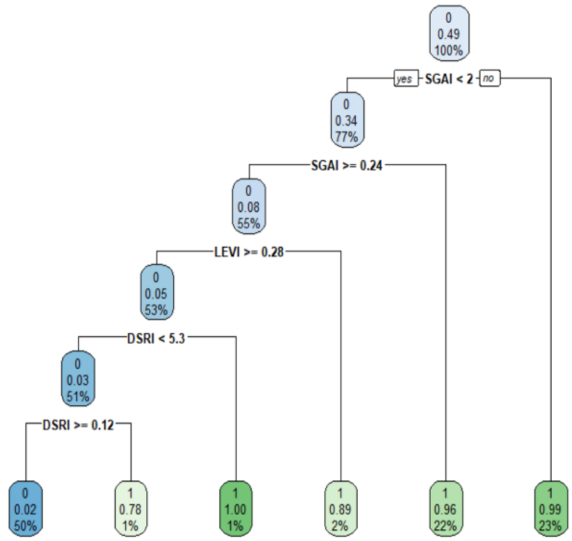
```
#Performing CART on balanced train data
tree.rose <- rpart(Cmanipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = data.rose, parms = list(split = "gini"))
printcp(tree.rose)
#Important variables in tree
varImp(tree.rose)
#Rpart plot
rpart.plot(tree.rose)
summary(tree.rose)
tree.rose
#Finding cp
opt <- which.min(tree.rose$cptable[ ,"xerror"])
cp <- tree.rose$cptable[opt, "CP"]
cp_tree <- prune(tree.rose, cp = cp)
#Best cp value is 0.01
```

Step 5 – Predicting results on my test data & building confusion matrix.

```
#Predicting on train
predicto_test <- predict(cp_tree, newdata = EM.test, type="class")
Answer <-table(predicto_test,EM.test$Cmanipulator)
confusionMatrix(Answer, positive = "1")
#Accuracy is approx 95%, specificity is 96% but Sensitivity is just 57%.
#Misclassification Rate
mean(predicto_test != EM.test$Cmanipulator)
#missclassification rate is 0.05205479
```

## Output – Confusion matrix, plot & summary of decision tree & Important variables

```
predicto_test   0   1
            0 338   6
            1  13   8

               Accuracy : 0.9479
                 95% CI : (0.9199, 0.9684)
    No Information Rate : 0.9616
    P-Value [Acc > NIR] : 0.9275

                  Kappa : 0.431

 Mcnemar's Test P-Value : 0.1687

            Sensitivity : 0.57143
            Specificity : 0.96296
         Pos Pred Value : 0.38095
         Neg Pred Value : 0.98256
             Prevalence : 0.03836
         Detection Rate : 0.02192
   Detection Prevalence : 0.05753
      Balanced Accuracy : 0.76720

       'Positive' Class : 1
```




```
> varImp(tree.rose)
       Overall
ACCR  42.75669
AQI  208.49930
DSRI 261.19576
LEVI 202.22790
SGAI 343.79336
SGI  186.84655
GMI    0.00000
DEPI   0.00000
```

```
n= 874

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

 1) root 874 425 0 (0.51372998 0.48627002)
   2) SGAI< 2.01679 672 226 0 (0.66369048 0.33630952)
     4) SGAI>=0.2369282 477  39 0 (0.91823899 0.08176101)
       8) LEVI>=0.2814986 459  23 0 (0.94989107 0.05010893)
        16) DSRI< 5.264031 450  14 0 (0.96888889 0.03111111)
          32) DSRI>=0.1233647 441   7 0 (0.98412698 0.01587302) *
          33) DSRI< 0.1233647 9   2 1 (0.22222222 0.77777778) *
        17) DSRI>=5.264031 9   0 1 (0.00000000 1.00000000) *
       9) LEVI< 0.2814986 18   2 1 (0.11111111 0.88888889) *
     5) SGAI< 0.2369282 195   8 1 (0.04102564 0.95897436) *
   3) SGAI>=2.01679 202   3 1 (0.01485149 0.98514851) *
```

**Answer g) –** Top 3 best decision rules obtained from CART model are

- If SGAI < 2.02, then it predicts that earnings are not manipulated with confidence of 66% and support of 77%.
- If SGAI>= 0.27, then it predicts that earnings are not manipulated with confidence of 92% and support of 55%.
- If LEVI >= 0.28, then it predicts that earnings are not manipulated with confidence of 95% and support of 53%

## QUESTION 1h - 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.

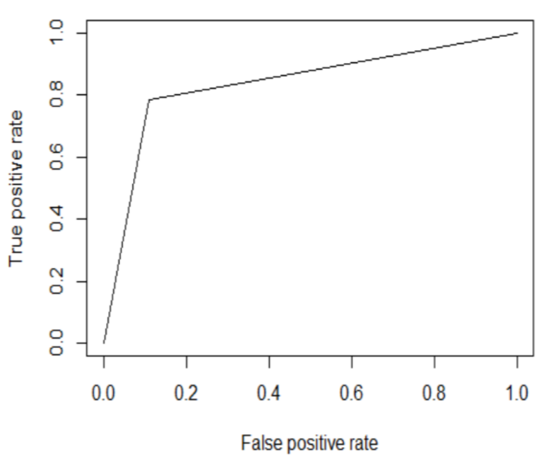
## Answer 1h)

**Logistic Regression with Cross Validation**

Step 1 – Logistic regression with Cross Validation

```
#CV with Logistic regresision #Let's see how Logistic regression with CV performs

ctrl <- trainControl(method = "repeatedcv", number = 10, savePredictions = TRUE, repeats = 5)

mod_fit <- train(Cmanipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = data.rose, method="glm", family="binomial",
                 trControl = ctrl, tuneLength = 5)
summary(mod_fit)
#Looking at results and finalmodel of my model
mod_fit$results
mod_fit$finalModel

#Predicting on Test Data
predict <- predict(mod_fit,newdata = EM.test,type = "raw")
confusionMatrix(data=predict, EM.test$Cmanipulator, positive = "1")
#Accuracy is 88.5%, sensitivity is 89% and specificity is 79%
mean(predict != EM.test$Cmanipulator)
#Missclassification error is 0.11

#Plotting ROC and calculating AUC
pr2 <- prediction(as.numeric(predict),as.numeric(EM.test$Cmanipulator))
perf2 <- performance(pr2,measure = "tpr",x.measure = "fpr")
plot(perf2)
auc(EM.test$Cmanipulator,predict)
#auc is 0.84
```

**Output – ROC curve, Confusion matrix and summary of Logistic regression model**



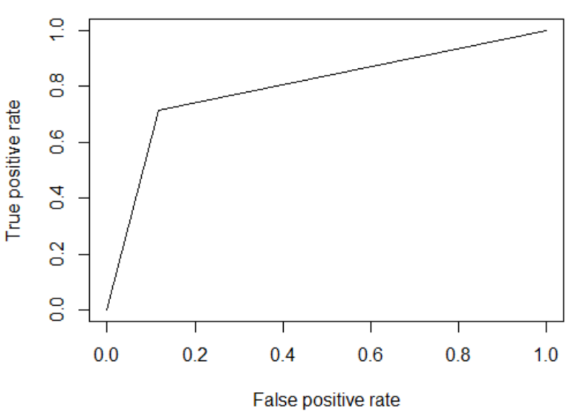

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 312   3
         1  39  11

               Accuracy : 0.8849
                 95% CI : (0.8477, 0.9158)
    No Information Rate : 0.9616
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3019

 Mcnemar's Test P-Value : 6.641e-08

            Sensitivity : 0.78571
            Specificity : 0.88889
         Pos Pred Value : 0.22000
         Neg Pred Value : 0.99048
             Prevalence : 0.03836
         Detection Rate : 0.03014
   Detection Prevalence : 0.13699
      Balanced Accuracy : 0.83730

       'Positive' Class : 1
```

```
> summary(mod_fit)

Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2079  -1.0132  -0.6278   1.0091   2.6819

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.08492    0.29375  -3.693 0.000221 ***
DSRI         0.06797    0.01577   4.309 1.64e-05 ***
GMI          0.41149    0.09827   4.187 2.82e-05 ***
AQI          0.05779    0.01088   5.310 1.09e-07 ***
SGI          0.21264    0.04218   5.042 4.62e-07 ***
DEPI        -0.11120    0.22203  -0.501 0.616477
SGAI         0.05226    0.01173   4.455 8.38e-06 ***
ACCR         3.12715    0.39935   7.831 4.86e-15 ***
LEVI         0.06135    0.05864   1.046 0.295480
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1211.0  on 873  degrees of freedom
Residual deviance: 1046.2  on 865  degrees of freedom
AIC: 1064.2

Number of Fisher Scoring iterations: 4
```

**Part h – Alternative method) – Lasso Regression with Cross Validation (For variable selection & to reduce overfitting)**

```r
#Lasso Regression------------------------------------------
#Preparing variables for Lasso regression
x_vars <- model.matrix(Cmanipulator~. , data.rose)[,-1]
y <- data.rose %>% select(Cmanipulator) %>% as.matrix()

#Defining lamba
lambda_seq <- 10^seq(2, -2, by = -.1)
#Lasso regression
lasso <- cv.glmnet(x_vars, y,
                       alpha = 1, lambda = lambda_seq, standardize = TRUE, nfolds = 10, family="binomial")
#Plotting lasso regression
plot(lasso)
#Obtainingbestvalue of lambda
lambda_cv <- lasso$lambda.min
#best lambda value is 0.01
#Plotting best lasso regression
lasso_best <- glmnet(x_vars, y,
                     alpha = 1, lambda = 0.01, standardize = TRUE)
#having a look at variables shrinked to 0 in lasso
coef(lasso_best)
#Answer - As you can see, variable DEPI comes out to be insignificant and hence shrinked to 0 to

x.test <- model.matrix(Cmanipulator~.-`Company ID`, EM.test)[,-1]
pred_lasso <- predict(lasso_best, s = lambda_cv, newx = x.test)
pred_lasso <- ifelse(pred_lasso > 0.48,1,0)

Answer3<-table(pred_lasso,EM.test$Cmanipulator) #94% accuracy for test data
confusionMatrix(Answer3, positive = "1")
#87.7% accuracy on  test data
#cutoff off of 0.48 is ideal where there is a balance between Sensitvity of 71% & Specificity of 88%
#Plotting ROC and calculating AUC
pr3 <- prediction(as.numeric(pred_lasso),as.numeric(EM.test$Cmanipulator))
perf3 <- performance(pr3,measure = "tpr",x.measure = "fpr")
plot(perf3)
auc(EM.test$Cmanipulator,pred_lasso)
#auc is 0.7987383
```

**Output – ROC curve, Confusion matrix and summary of Lasso model**




```
Confusion Matrix and Statistics

pred_lasso   0   1
         0 310   4
         1  41  10

               Accuracy : 0.8767
                 95% CI : (0.8385, 0.9086)
    No Information Rate : 0.9616
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2634

 Mcnemar's Test P-Value : 8.025e-08

            Sensitivity : 0.71429
            Specificity : 0.88319
         Pos Pred Value : 0.19608
         Neg Pred Value : 0.98726
             Prevalence : 0.03836
         Detection Rate : 0.02740
   Detection Prevalence : 0.13973
      Balanced Accuracy : 0.79874

       'Positive' Class : 1
```

**Question-** compare the results with the previous logistic regression model?

**Solution –** Considering 0 i.e. Non-Manipulator as the positive class

| Model | Sensitivity | Specificity |
|---|---|---|
| Logistic Regression (Complete Dataset) | 89% | 79% |
| Stepwise Logistic Regression (Sample data of 220) | 87% | 87% |

Logistic Regression using complete dataset where train data is balanced using SMOTE techniques gives better results on Sensitivity as compared to Stepwise Logistic Regression built on the sample data using undersampling technique and is vice versa for Specificity.

Develop models using ensemble machine learning algorithms such as random forest and Adaboosting. compare the outputs from these methods with logistic regression and classification tree

## QUESTION 1i - Develop models using ensemble machine learning algorithms such as random forest and Adaboosting. compare the outputs from these methods with logistic regression and classification tree.

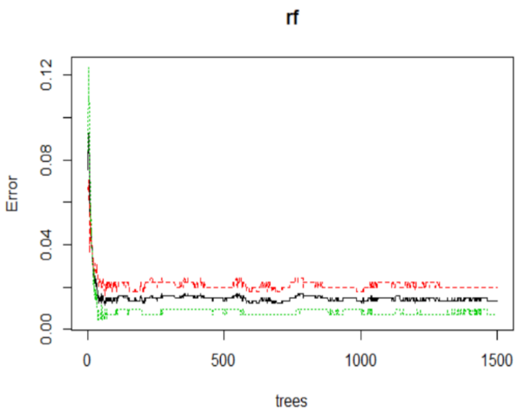
**Random Forest with Cross Validation & Parameter Tuning**

Step 1 – Fit a Random Forest model as below, parameter tuning has been done on parameters like mtry and ntree

```
#Random Forest with CV and Parameter Tuning---------------------------------
#Cross Validation with Parameter Tuning
control1 <- trainControl(method="repeatedcv", number=10, repeats=3)
tunegrid1 <- expand.grid(.mtry=c(1:8))
set.seed(123)
metric <-  c("Sensitivity","Specificity")
rf <- train(Cmanipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = data.rose,
            method="rf", metric=metric, tuneGrid=tunegrid1, trControl=control1)
print(rf)
plot(rf) #best mtry is 1
#Using mtry=1 and finding best value for ntree below
control2 <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid2 <- expand.grid(.mtry=1)
metric <- c("Sensitivity","Specificity")
modellist <- list()
for (ntree in c(100, 500, 1000, 1500)) {
  set.seed(123)
  rf1 <- train(Cmanipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = data.rose,
               method="rf", metric=metric, tuneGrid=tunegrid2, trControl=control2, ntree=ntree)
  key <- toString(ntree)
  modellist[[key]] <- rf1
}
# compare results
results <- resamples(modellist)
summary(results)
dotplot(results)
```

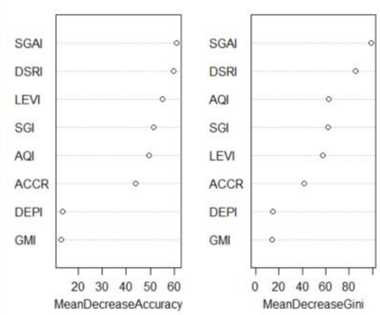
```
#For mtry=1 and ntree = 1500, we get the max accuracy
#Let's train Random Forest model using above parameters
rf <- randomForest(Cmanipulator ~ DSRI+GMI+AQI+SGI+DEPI+SGAI+ACCR+LEVI, data = data.rose,
                   mtry = 1, ntree = 1500, proximity = T, importance = T)
rf_tree_pred <- predict(rf, EM.test, type = "class")
confusionMatrix(data=rf_tree_pred, EM.test$Cmanipulator, positive = "1")
#96% accuracy with 57% sensitivity and 99% specificity

plot(rf)
print(rf)
#OOB estimate of  error rate: 1.37%
importance(rf, type = 1)
rf
varImpPlot(rf)
```

**Output – RF plot with oob error, Confusion matrix, summary of Random Forest model & Important variables plot**




```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 347   6
         1   4   8

               Accuracy : 0.9726
                 95% CI : (0.9502, 0.9868)
    No Information Rate : 0.9616
    P-Value [Acc > NIR] : 0.1705

                  Kappa : 0.6013

 Mcnemar's Test P-Value : 0.7518

            Sensitivity : 0.57143
            Specificity : 0.98860
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.98300
             Prevalence : 0.03836
         Detection Rate : 0.02192
   Detection Prevalence : 0.03288
      Balanced Accuracy : 0.78002

       'Positive' Class : 1
```

```
 randomForest(formula = Cmanipulator ~ DSRI + GMI + AQI + SGI +      DEPI + SGAI
 + ACCR + LEVI, data = data.rose, mtry = 1, ntree = 1500,      proximity = T, imp
ortance = T)
               Type of random forest: classification
                     Number of trees: 1500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 1.37%
Confusion matrix:
    0   1 class.error
0 440   9 0.020044543
1   3 422 0.007058824
```




## Part I ) Adaboosting

```
#Adaboosting-----------------------------------------------
install.packages("JOUSBoost")
library(JOUSBoost)
data <- data.rose
data$Cmanipulator <- as.factor(data$Cmanipulator)
#Preparing data to fit adaboost model
y<- data[,9]
y <- ifelse(y==0,-1,1)
y <- c(y)
#Adaboost
ada <- adaboost(as.matrix(data[,-9]), y, tree_depth = 5, n_rounds = 100, verbose = FALSE,control = NULL)

#Predicting on Test data
t<- EM.test[,-1]
yhat_ada = predict(ada, as.matrix(t[,-9]))
i<- EM.test[,10]
i <- ifelse(i==0,-1,1)
Answer4<-table(yhat_ada,i)
confusionMatrix(Answer4, positive = "1")
#Accuracy is 97%, sensitivity is 43% & specificity is 99%
mean(i != yhat_ada)
#missclassification rate is 0.03
```

**Output – Confusion Matrix**

```
Confusion Matrix and Statistics

          i
yhat_ada  -1   1
      -1 346   8
      1    5   6

               Accuracy : 0.9644
                 95% CI : (0.9399, 0.9809)
    No Information Rate : 0.9616
    P-Value [Acc > NIR] : 0.4624

                  Kappa : 0.4618

 Mcnemar's Test P-Value : 0.5791

            Sensitivity : 0.42857
            Specificity : 0.98575
         Pos Pred Value : 0.54545
         Neg Pred Value : 0.97740
             Prevalence : 0.03836
         Detection Rate : 0.01644
   Detection Prevalence : 0.03014
      Balanced Accuracy : 0.70716

       'Positive' Class : 1
```

## QUESTION 1j - What will be your final recommendation for predicting earnings manipulators? What variables should be considered important?

Answer **1j)**

**Final Recommendations & Comparison between various models for Complete Dataset where Training data was balanced using SMOTE technique.**
**Sensitivity & Specificity are calculated considering 1 i.e. Manipulator as the positive class**

| Model | Sensitivity | Specificity | Accuracy |
|---|---|---|---|
| Decision Tree (CART) | 57% | 96.3% | 95% |
| Logistic Regression with Cross Validation | 78.5% | 89% | 88.5% |
| Lasso Regression | 71% | 88% | 88% |
| Random Forest with Parameter Tuning | 57% | 99% | 97% |
| Adaboosting | 43% | 98.6% | 96% |

**Answer - Conclusion**

As stated in the case, classifying a manipulator as non-manipulator may encourage companies to repeat the practice, whereas classifying a non-manipulator as manipulator may involve wasting the resources of regulators, **Sensitivity** and **Specificity** are the important measures that we are taking into considerations for recommending the best model. We are not taking accuracy into account for now, since train data has more number of non-manipulators as compared to manipulators.

Since **Logistic Regression with Cross Validation** gives us a good score on both Sensitivity which is 78.5% and Specificity which is 89%, we would recommend going ahead with Logistic regression using Cross validation model.

Variables that should be considered important are SGAI, DSRI, GMI, ACCR, SGI & AQI whereas DEPI & LEVI comes out to be insignificant.