Predicting Earnings Manipulation - Logistic Regression, Decision Trees and Random Forest

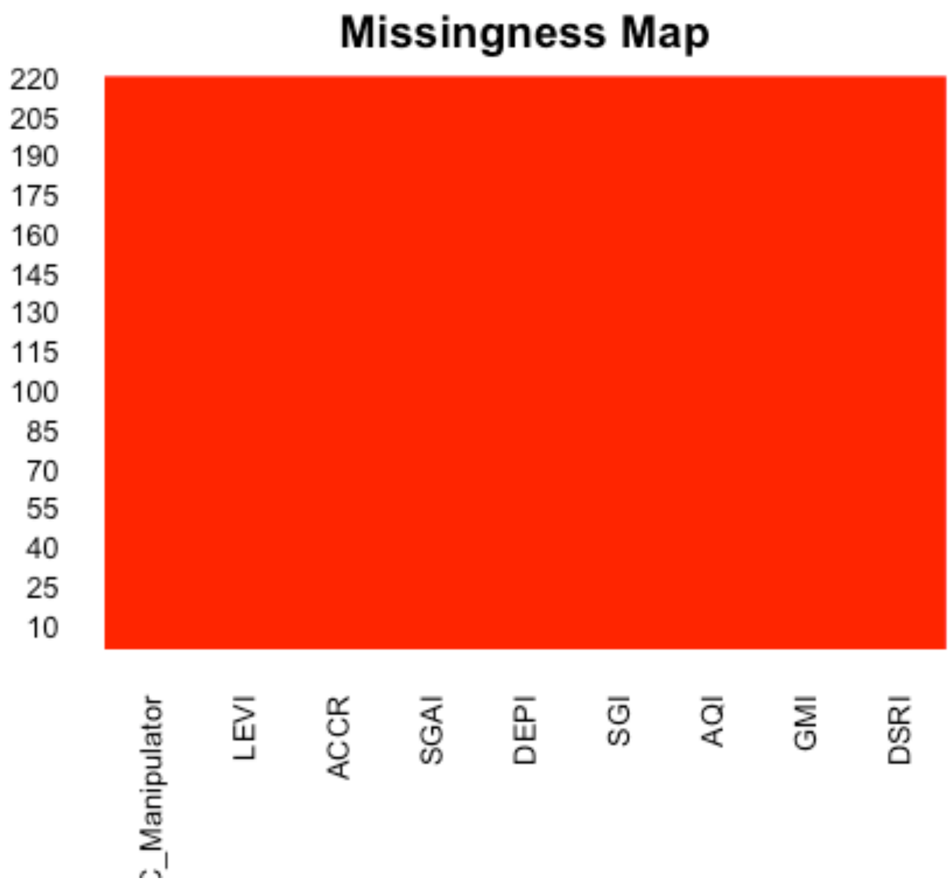
We will try to predict earnings manipulations by Indian firms using machine learning algorithms. The data comes from a case study by the Indian Institute of Management Bangalore. Entire dataset contains 1200 non-manipulators and 39 manipulators and sample dataset contains 220 cases including 39 manipulators.

```
#LOAD and fix STR of data

rm(list=ls())
library(readxl)

sample_data <- read_excel("IMB579-XLS-ENG.xlsx", sheet = "Sample for Model
Development")

#rename and convert to categorical variable (1 YES manipulator/0 NO not
manipulator)
sample_data<- sample_data[, -10] #remove manipulator yes/no column
names(sample_data)[10] <- "c_Manipulator"
C_Manipulator <- as.factor(sample_data$c_Manipulator)
sample_data <- data.frame(sample_data, C_Manipulator)
sample_data<- sample_data[, -10] #remove c_maniputalor column that is not
categorical
sample_data<- sample_data[, -1] #remove company.ID column
str(sample_data)

## '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 ...
```

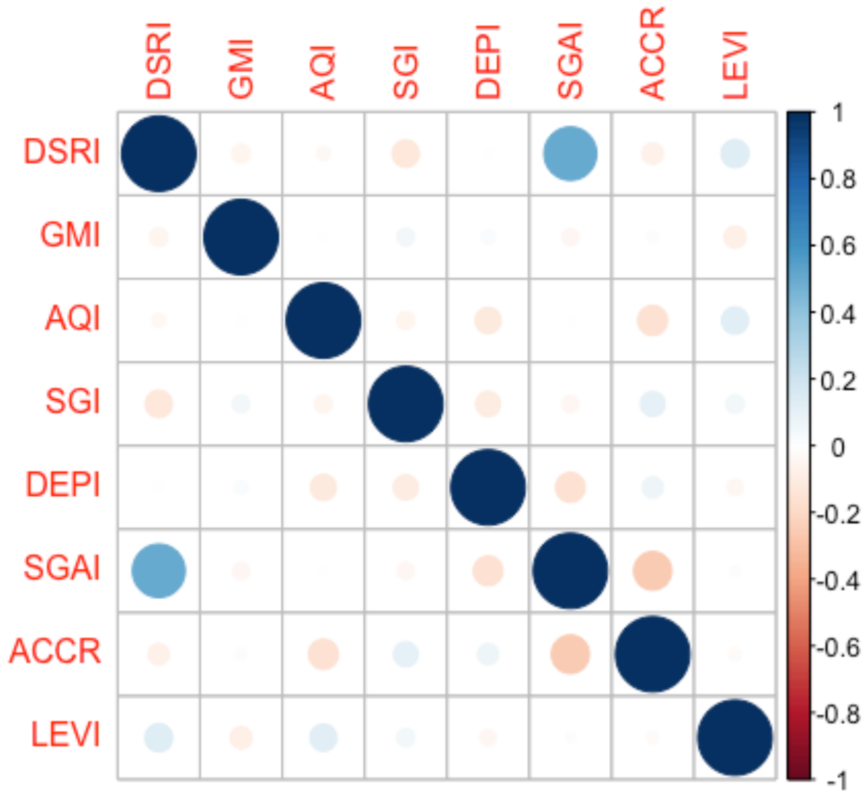
EDA

```
library(dplyr)

count(sample_data, vars= C_Manipulator)

##   vars   n
## 1    0 181
## 2    1  39
```

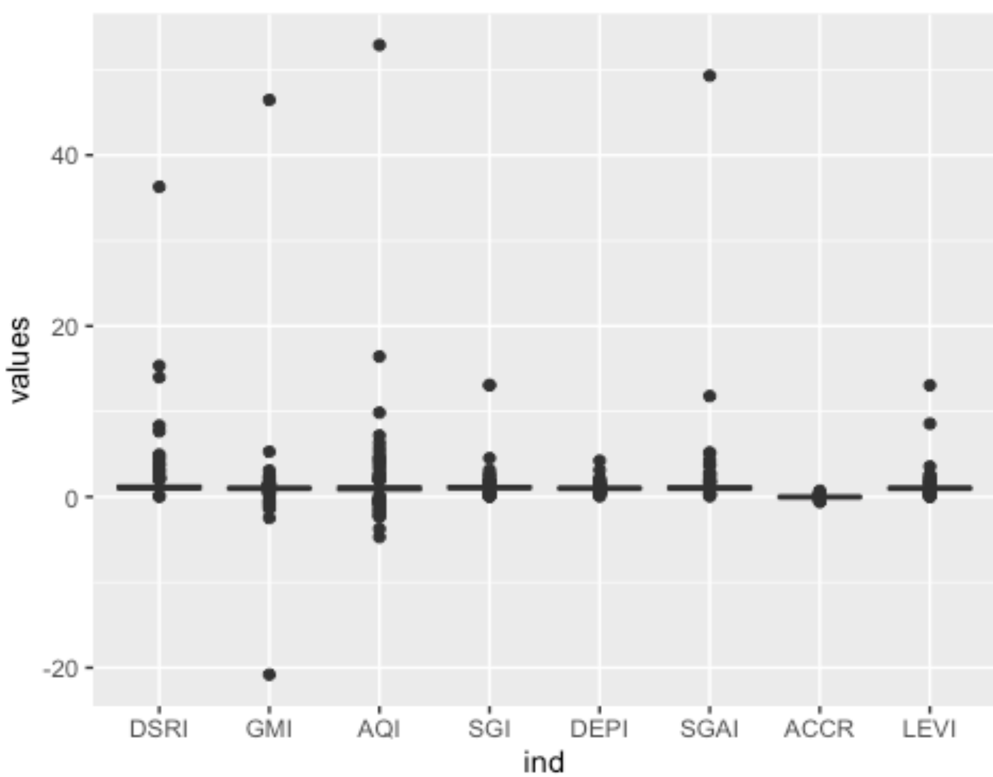
```r
library(Amelia)

library(mlbench)

missmap(sample_data, col=c("blue", "red"), legend=FALSE)
```




```r
#no missing data in dataset

library(corrplot)

correlations <- cor(sample_data[,1:8])
corrplot(correlations, method="circle")

library(ggplot2)
```




```
ggplot(stack(sample_data), aes(x = ind, y = values)) +
  geom_boxplot()
```



FUNCTIONS TO BE USED (Accuracy, Recall, Precision, F-1 score)

```
#function to calculate accuracy
my.accuracy <- function(actual, predictions)
{
  y<- as.vector(table(predictions, actual))
  names(y) <- c("TN", "FP", "FN", "TP")
  accuracy<- (y["TN"]+ y["TP"])/ sum(y)
  return(as.numeric(accuracy))
}
#function to calculate recall
my.recall <- function(actual, predictions)
{
  y<- as.vector(table(predictions, actual))
  names(y) <- c("TN", "FP", "FN", "TP")
  recall <-  y["TP"]/ (y["TP"] + y["FN"])
  return(as.numeric(recall))
}
#function to calculate precision TP/TP+FP
my.precision <- function(actual, predictions)
{
  y<- as.vector(table(predictions, actual))
  names(y) <- c("TN", "FP", "FN", "TP")
  precision <-  y["TP"]/ (y["TP"] + y["FP"])
  return(as.numeric(precision))
}
#function to calculate F-1 score 2*(Recall * Precision) / (Recall +
Precision)
my.f1 <- function(actual, predictions)
{
  y<- as.vector(table(predictions, actual))
  names(y) <- c("TN", "FP", "FN", "TP")
  f1 <-  (2*((y["TP"]/ (y["TP"] + y["FN"]))* y["TP"]/ (y["TP"] + y["FP"]))) /
((y["TP"]/ (y["TP"] + y["FN"]) + (y["TP"]/ (y["TP"] + y["FP"]))))
  return(as.numeric(f1))
}
```

MODEL 1: LOGISTIC REGRESSION using sample data- NO BALANCING

```
#SPLIT SAMPLE DATA
set.seed(123)
indx<- sample(2, size=nrow(sample_data),replace= T, prob = c(0.7, 0.3))
train <- sample_data[indx==1,]
test <- sample_data[indx==2, ]
actual<- test$C_Manipulator

#TRAIN MODEL
logitModel <- glm(C_Manipulator ~ . , data = train, family = "binomial")
summary(logitModel)
```

```
##
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6588  -0.4022  -0.2812  -0.2009   2.4824
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -8.3466     2.0376  -4.096  4.2e-05 ***
## DSRI          0.9096     0.3706   2.455 0.014108 *
## GMI           1.3304     0.4578   2.906 0.003659 **
## AQI           0.4764     0.1641   2.903 0.003693 **
## SGI           2.2498     0.6788   3.315 0.000917 ***
## DEPI          0.2925     0.6000   0.487 0.625909
## SGAI          0.1892     0.4570   0.414 0.678823
## ACCR          7.1227     2.1026   3.387 0.000705 ***
## LEVI         -0.1301     0.7333  -0.177 0.859182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.688  on 159  degrees of freedom
## Residual deviance:  87.272  on 151  degrees of freedom
## AIC: 105.27
##
## Number of Fisher Scoring iterations: 8

#TEST MODEL
predictions <- predict(logitModel, newdata = test, type = "response")
actual <- test$C_Manipulator

#confusion matrix
predict_logitmodel <- rep(0, length(predictions))  #all zeros for predictions
predict_logitmodel[predictions >= 0.5 ] <- 1   #replace with 1 of prob >= 0.5
table(predict_logitmodel,actual , dnn= c("Predictions", "Actual"))
#confusion matrix

##            Actual
## Predictions  0  1
##           0 46  9
##           1  0  5

#Evaluation of model
my.accuracy(actual, predict_logitmodel)
## [1] 0.85
my.recall(actual, predict_logitmodel)
## [1] 0.3571429
```

```
my.precision(actual, predict_logitmodel)
## [1] 1
my.f1(actual, predict_logitmodel)
## [1] 0.5263158


#NOT a really good f-1 score, almost the same as a random model (.50)

#LOGISTIC REGRESSION MODEL
# y = -8.3466 +  0.9096*DSRI + 1.3304*GMI +  0.4764*AQI + 2.2498*SGI +
7.1227*ACCR

      #using a p-value of 0.5 to select significant variables
```

Balancing TRAINING data with 4 different techniques (over, under, both, rose)

```
library(ROSE)

set.seed(123)
#UNDERSAMPLED - uses all instances from minority training class(uses 25 out
of 25 observations in minority class) and ONLY the same num of instances for
majority class
sample_data_balanced_under <- ovun.sample(C_Manipulator ~ ., data = train,
method = "under")$data
table(sample_data_balanced_under$C_Manipulator)
##  0  1
## 25 25

#OVERSAMPLED - duplicates examples from the minority class
sample_data_balanced_over <- ovun.sample(C_Manipulator ~ ., data = train,
method = "over", N=500)$data
table(sample_data_balanced_over$C_Manipulator)
##   0   1
## 135 365

#BOTH-the minority class is oversampled with replacement and majority class
is undersampled without replacement
sample_data_balanced_both <- ovun.sample(C_Manipulator ~ ., data = train,
method = "both", N=500)$data
table(sample_data_balanced_both$C_Manipulator)
##   0   1
## 244 256

#ROSE-generate data synthetically as well.
sample_data_balanced_rose <- ROSE(C_Manipulator ~ ., data  = train)$data
table(sample_data_balanced_rose$C_Manipulator)
##  0  1
## 86 74
```

MODEL 2,3,4,5: LOGISTIC REGRESSION using different sample techniques

```
set.seed(123)
#***UNDERSAMPLIG****
#train
sample_under_fit <- glm(C_Manipulator ~ . , data =
sample_data_balanced_under, family = "binomial")

summary(sample_under_fit)

##
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data =
sample_data_balanced_under)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.89210  -0.46380  -0.01759   0.40329   1.74083
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.7099     5.6874  -2.586   0.0097 **
## DSRI          4.7427     2.0445   2.320   0.0204 *
## GMI           2.3973     1.5939   1.504   0.1326
## AQI           1.8086     0.7859   2.301   0.0214 *
## SGI           6.8424     2.6860   2.547   0.0109 *
## DEPI         -1.7355     1.2769  -1.359   0.1741
## SGAI         -0.6365     0.9269  -0.687   0.4923
## ACCR         10.2933     4.4202   2.329   0.0199 *
## LEVI         -2.0795     1.7785  -1.169   0.2423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 69.315  on 49  degrees of freedom
## Residual deviance: 30.185  on 41  degrees of freedom
## AIC: 48.185
##
## Number of Fisher Scoring iterations: 9

#test
sample_predictions_under <- predict(sample_under_fit, newdata = test,  type =
"response")
#confusion matrix
predict_under_sample<- rep(0, length(sample_predictions_under))   #all zeros
```

```
for predictions
predict_under_sample[sample_predictions_under >= 0.5 ] <- 1        #replace
with 1 of prob >= 0.5
table(predict_under_sample,actual , dnn= c("Predictions", "Actual"))

##            Actual
## Predictions  0  1
##           0 37  3
##           1  9 11

#***OVERSAMPLIG****
#train
sample_over_fit <- glm(C_Manipulator ~ . , data = sample_data_balanced_over,
family = "binomial")

summary(sample_over_fit)

##
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data =
sample_data_balanced_over)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7250  -0.1922   0.2110   0.4881   1.6669
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.14674    1.39169  -7.291 3.08e-13 ***
## DSRI          2.02334    0.35142   5.758 8.53e-09 ***
## GMI           1.96464    0.42087   4.668 3.04e-06 ***
## AQI           0.85664    0.12354   6.934 4.08e-12 ***
## SGI           4.21524    0.59038   7.140 9.34e-13 ***
## DEPI         -0.05482    0.52391  -0.105    0.917
## SGAI          0.01251    0.30412   0.041    0.967
## ACCR          9.68868    1.37848   7.029 2.09e-12 ***
## LEVI         -0.23833    0.43402  -0.549    0.583
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 583.26  on 499  degrees of freedom
## Residual deviance: 310.63  on 491  degrees of freedom
## AIC: 328.63
##
## Number of Fisher Scoring iterations: 9

#test
sample_predictions_over <- predict(sample_over_fit, newdata = test,  type =
"response")
```

```
#confusion matrix
predict_over_sample<- rep(0, length(sample_predictions_over))   #all zeros
for predictions
predict_over_sample[sample_predictions_over >= 0.5 ] <- 1        #replace
with 1 of prob >= 0.5
table(predict_over_sample,actual , dnn= c("Predictions", "Actual"))

##             Actual
## Predictions  0  1
##           0 35  2
##           1 11 12

#***BOTH****
#train
sample_both_fit <- glm(C_Manipulator ~ . , data = sample_data_balanced_both,
family = "binomial")

summary(sample_both_fit)

##
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data =
sample_data_balanced_both)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6787  -0.5121   0.0000   0.5631   1.9443
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -14.67630    1.57498  -9.318  < 2e-16 ***
## DSRI          2.12304    0.29525   7.191 6.45e-13 ***
## GMI           1.81684    0.29933   6.070 1.28e-09 ***
## AQI           0.65673    0.09758   6.730 1.70e-11 ***
## SGI           4.64189    0.55758   8.325  < 2e-16 ***
## DEPI          3.52297    0.75546   4.663 3.11e-06 ***
## SGAI         -0.04422    0.26861  -0.165    0.869
## ACCR          8.40806    1.06452   7.898 2.82e-15 ***
## LEVI         -0.38377    0.41262  -0.930    0.352
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 692.86  on 499  degrees of freedom
## Residual deviance: 366.42  on 491  degrees of freedom
## AIC: 384.42
##
## Number of Fisher Scoring iterations: 9
```

```
#test
sample_predictions_both <- predict(sample_both_fit, newdata = test,  type =
"response")
#confusion matrix
predict_both_sample<- rep(0, length(sample_predictions_both))   #all zeros
for predictions
predict_both_sample[sample_predictions_both >= 0.5 ] <- 1         #replace
with 1 of prob >= 0.5
table(predict_both_sample,actual , dnn= c("Predictions", "Actual"))

##              Actual
## Predictions  0  1
##            0 43  3
##            1  3 11

#***ROSE****
#train
sample_rose_fit <- glm(C_Manipulator ~ . , data = sample_data_balanced_rose,
family = "binomial")
summary(sample_rose_fit)

##
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data =
sample_data_balanced_rose)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9413  -0.9968  -0.7160   0.9469   2.2675
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.48760    0.71124  -2.092  0.03648 *
## DSRI         0.04479    0.03214   1.394  0.16346
## GMI          0.68807    0.28385   2.424  0.01535 *
## AQI          0.05178    0.02462   2.103  0.03543 *
## SGI          0.05354    0.11357   0.471  0.63735
## DEPI         0.12462    0.50263   0.248  0.80418
## SGAI         0.07291    0.03518   2.072  0.03822 *
## ACCR         3.07205    0.95629   3.212  0.00132 **
## LEVI         0.10692    0.15213   0.703  0.48217
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 220.91  on 159  degrees of freedom
## Residual deviance: 194.05  on 151  degrees of freedom
## AIC: 212.05
```

```
##
## Number of Fisher Scoring iterations: 4
```

```
#test
sample_predictions_rose <- predict(sample_rose_fit, newdata = test,  type =
"response")
#confusion matrix
predict_rose_sample<- rep(0, length(sample_predictions_rose))   #all zeros
for predictions
predict_rose_sample[sample_predictions_rose >= 0.5 ] <- 1        #replace
with 1 of prob >= 0.5
table(predict_rose_sample,actual , dnn= c("Predictions", "Actual"))
```

```
##            Actual
## Predictions  0  1
##           0 42  6
##           1  4  8
```

EVALUATION OF LOGISTIC REGRESSION MODELS using balanced data

```
print("Accuracy/Recall/Precision/F-1 score using UNDER SAMPLING")
```

```
my.accuracy(actual, predict_under_sample)
```

```
## [1] 0.8
```

```
my.recall(actual, predict_under_sample)
```

```
## [1] 0.7857143
```

```
my.precision(actual, predict_under_sample)
```

```
## [1] 0.55
```

```
my.f1(actual, predict_under_sample)
```

```
## [1] 0.6470588
```

```
print("Accuracy/Recall/Precision/F-1 score using OVER SAMPLING")
```

```
my.accuracy(actual, predict_over_sample)
```

```
## [1] 0.7833333
```

```
my.recall(actual, predict_over_sample)
```

```
## [1] 0.8571429
```

```
my.precision(actual, predict_over_sample)
```

```
## [1] 0.5217391
```

```
my.f1(actual, predict_over_sample)

## [1] 0.6486486

print("Accuracy/Recall/Precision/F-1 score using BOTH UNDER AND OVER
SAMPLING")

my.accuracy(actual, predict_both_sample)

## [1] 0.9

my.recall(actual, predict_both_sample)

## [1] 0.7857143

my.precision(actual, predict_both_sample)

## [1] 0.7857143

my.f1(actual, predict_both_sample)

## [1] 0.7857143

print("Accuracy/Recall/Precision/F-1 score using ROSE")

my.accuracy(actual, predict_rose_sample)

## [1] 0.8333333

my.recall(actual, predict_rose_sample)

## [1] 0.5714286

my.precision(actual, predict_rose_sample)

## [1] 0.6666667

my.f1(actual, predict_rose_sample)

## [1] 0.6153846
```

	UNDERSAMPLING	OVERSAMPLING	BOTH	ROSE
Accuracy	0.8	0.7833333	0.9	0.8333333
Recall	0.7857143	0.8571429	0.7857143	0.5714286
Precision	0.55	0.5217391	0.7857143	0.6666667
F-1 Score	0.6470588	0.6486486	0.7857143	0.6153846

```
#BEST F-1 SCORE WAS OBTAINED BY BALANCING DATA WITH BOTH UNDER AND OVER
SAMPLING.
#F-1 SCORE INCREASED WITH ALL BALANCING TECHNIQUES COMPARED TO THE ORIGINAL
SAMPLE DATA. (0.5263158)
```

LOGISTIC REGRESSION MODELS using entire dataset

```
set.seed(123)
data <- read_excel("IMB579-XLS-ENG.xlsx", sheet = "Complete Data")
data<- data[, -10] #remove manipulator yes/no column
names(data)[10] <- "c_Manipulator"
C_Manipulator <- as.factor(data$c_Manipulator)
data <- data.frame(data, C_Manipulator)
data<- data[, -10] #remove c_maniputalor column that is not categorical
data<- data[, -1] #remove company.ID column
```

Create balanced datasets(4)

```
#UNDERSAMPLED
down_data <- ovun.sample(C_Manipulator ~ ., data = data, method =
"under")$data
table(down_data$C_Manipulator)
##  0  1
## 38 39

#OVERSAMPLED
up_data <- ovun.sample(C_Manipulator ~ ., data = data, method = "over",
N=2400)$data
table(up_data$C_Manipulator)
##    0    1
## 1200 1200

#BOTH
both_data <- ovun.sample(C_Manipulator ~ ., data = data, method = "both",
p=0.5, N=1250)$data
table(both_data$C_Manipulator)
##   0   1
## 622 628

#ROSE-generate data synthetically as well.
rose_data <- ROSE(C_Manipulator ~ ., data  = data)$data
table(rose_data$C_Manipulator)
##   0   1
## 634 605
```

#LOGISTIC REGRESSION MODEL USING NO BALANCING AND 4 BALANCING TECHNIQUES(under, over, both, rose)

MODELS using entire dataset (Unbalanced data, under, over, both, and rose sampled)

```
#***NOT BALANCED****
indx<- sample(2, size=nrow(data),replace= T, prob = c(0.7, 0.3))
train1 <- data[indx==1,]
```

```
test1 <- data[indx==2, ]
actual2<- test1$C_Manipulator
#train
logitModel.fit <- glm(C_Manipulator ~ . , data = train1, family = "binomial")

summary(logitModel.fit)
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data = train1)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.10782  -0.17717  -0.12640  -0.09576   3.05648
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.19075    1.51533  -4.085 4.40e-05 ***
## DSRI         0.64549    0.16662   3.874 0.000107 ***
## GMI          0.78188    0.30658   2.550 0.010762 *
## AQI          0.27769    0.09708   2.861 0.004229 **
## SGI          1.18102    0.30660   3.852 0.000117 ***
## DEPI        -1.31165    1.15049  -1.140 0.254254
## SGAI         0.36634    0.40228   0.911 0.362475
## ACCR         9.12279    1.85525   4.917 8.78e-07 ***
## LEVI        -0.28205    0.30942  -0.912 0.362008
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 235.72  on 902  degrees of freedom
## Residual deviance: 144.56  on 894  degrees of freedom
## AIC: 162.56
##
## Number of Fisher Scoring iterations: 8

#test
logitModel.fit.predictions <- predict(logitModel.fit, newdata = test1,  type
= "response")
#confusion matrix
predict_logitModel.fit<- rep(0, length(logitModel.fit.predictions))   #all
zeros for predictions
predict_logitModel.fit[logitModel.fit.predictions >= 0.5 ] <- 1
#replace with 1 of prob >= 0.5
table(predict_logitModel.fit,actual2 , dnn= c("Predictions", "Actual"))

##            Actual
## Predictions   0   1
##           0 321  11
##           1   2   2
```

```
#***UNDERSAMPLING***
#split
indx<- sample(2, size=nrow(down_data),replace= T, prob = c(0.7, 0.3))
train_down <- down_data[indx==1,]
test_down<- down_data[indx==2, ]
actual_down<- test_down$C_Manipulator
#train
logitModel.fit.UNDER <- glm(C_Manipulator ~ . , data = train_down, family =
"binomial")

summary(logitModel.fit.UNDER)
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data = train_down)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1817  -0.7679   0.0000   0.9191   1.5822
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -5.1216     2.9362  -1.744   0.0811 .
## DSRI          1.8083     0.7949   2.275   0.0229 *
## GMI           0.6987     0.5384   1.298   0.1944
## AQI           0.7630     0.3431   2.224   0.0261 *
## SGI           2.1790     1.2278   1.775   0.0760 .
## DEPI         -0.4671     1.2498  -0.374   0.7086
## SGAI         -0.8592     0.7173  -1.198   0.2310
## ACCR          7.1009     3.8698   1.835   0.0665 .
## LEVI         -0.6268     0.5173  -1.212   0.2257
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 74.786  on 53  degrees of freedom
## Residual deviance: 51.917  on 45  degrees of freedom
## AIC: 69.917
##
## Number of Fisher Scoring iterations: 8

#test
logitModel.fit.UNDER.predictions <- predict(logitModel.fit.UNDER, newdata =
test_down,  type = "response")
#confusion matrix
predict_logitModel.UNDER.fit<- rep(0,
length(logitModel.fit.UNDER.predictions))   #all zeros for predictions
predict_logitModel.UNDER.fit[logitModel.fit.UNDER.predictions >= 0.5 ] <- 1
#replace with 1 of prob >= 0.5
table(predict_logitModel.UNDER.fit,actual_down , dnn= c("Predictions",
"Actual"))
```

```
##            Actual
## Predictions  0  1
##           0 12  4
##           1  0  7
```

```
#***OVERSAMPLING***
#split
indx<- sample(2, size=nrow(up_data),replace= T, prob = c(0.7, 0.3))
train_up <- up_data[indx==1,]
test_up<- up_data[indx==2, ]
actual_up<- test_up$C_Manipulator
#train
logitModel.fit.OVER <- glm(C_Manipulator ~ . , data = train_up, family =
"binomial")

summary(logitModel.fit.OVER)
```

```
##
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data = train_up)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3761  -0.5375   0.0000   0.6736   1.7691
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.97978    0.59189 -15.171  < 2e-16 ***
## DSRI         1.77400    0.15314  11.584  < 2e-16 ***
## GMI          2.04306    0.19315  10.578  < 2e-16 ***
## AQI          0.64578    0.04269  15.126  < 2e-16 ***
## SGI          2.58751    0.21936  11.796  < 2e-16 ***
## DEPI         0.49009    0.21517   2.278   0.0227 *
## SGAI         0.17636    0.14058   1.255   0.2096
## ACCR        10.52655    0.73138  14.393  < 2e-16 ***
## LEVI        -0.47889    0.11400  -4.201 2.66e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2398.2  on 1730  degrees of freedom
## Residual deviance: 1270.4  on 1722  degrees of freedom
## AIC: 1288.4
##
## Number of Fisher Scoring iterations: 9
```

```
#test
logitModel.fit.OVER.predictions <- predict(logitModel.fit.OVER, newdata =
test_up,  type = "response")
#confusion matrix
```

```r
predict_logitModel.OVER.fit<- rep(0, length(logitModel.fit.OVER.predictions))
#all zeros for predictions
predict_logitModel.OVER.fit[logitModel.fit.OVER.predictions >= 0.5 ] <- 1
#replace with 1 of prob >= 0.5
table(predict_logitModel.OVER.fit,actual_up , dnn= c("Predictions",
"Actual"))
```

```
##              Actual
## Predictions   0   1
##           0 311  56
##           1  49 253
```

```r
#***BOTH***
#split
indx<- sample(2, size=nrow(both_data),replace= T, prob = c(0.7, 0.3))
train_both <- both_data[indx==1,]
test_both<- both_data[indx==2, ]
actual_both<- test_both$C_Manipulator
#the minority class is oversampled with replacement and majority class is
undersampled without replacement
#train
logitModel.fit.BOTH <- glm(C_Manipulator ~ . , data = train_both, family =
"binomial")
```

```r
summary(logitModel.fit.BOTH)
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data = train_both)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.8537  -0.5095   0.0000   0.4688   1.7349
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.92276    0.79701 -11.195  < 2e-16 ***
## DSRI         2.03032    0.21681   9.364  < 2e-16 ***
## GMI          1.55101    0.25035   6.195 5.82e-10 ***
## AQI          0.68715    0.06205  11.075  < 2e-16 ***
## SGI          2.60875    0.29080   8.971  < 2e-16 ***
## DEPI         0.67790    0.31784   2.133   0.0329 *
## SGAI         0.10742    0.19080   0.563   0.5734
## ACCR        10.21293    0.95445  10.700  < 2e-16 ***
## LEVI        -0.61845    0.14105  -4.385 1.16e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1233.73  on 889  degrees of freedom
## Residual deviance:  632.34  on 881  degrees of freedom
```

```
## AIC: 650.34
##
## Number of Fisher Scoring iterations: 8
```

```
#test
logitModel.fit.BOTH.predictions <- predict(logitModel.fit.BOTH, newdata =
test_both,  type = "response")
#confusion matrix
predict_logitModel.BOTH.fit<- rep(0, length(logitModel.fit.BOTH.predictions))
#all zeros for predictions
predict_logitModel.BOTH.fit[logitModel.fit.BOTH.predictions >= 0.5 ] <- 1
#replace with 1 of prob >= 0.5
table(predict_logitModel.BOTH.fit,actual_both , dnn= c("Predictions",
"Actual"))
```

```
##             Actual
## Predictions   0   1
##           0 161  31
##           1  20 148
```

```
#***ROSE***
#split
indx<- sample(2, size=nrow(rose_data),replace= T, prob = c(0.7, 0.3))
train_rose <- rose_data[indx==1,]
test_rose<- rose_data[indx==2, ]
actual_rose<- test_rose$C_Manipulator
#train
logitModel.fit.ROSE <- glm(C_Manipulator ~ . , data = train_rose, family =
"binomial")
summary(logitModel.fit.ROSE)
## Call:
## glm(formula = C_Manipulator ~ ., family = "binomial", data = train_rose)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0275  -1.0041  -0.5014   0.9723   2.8542
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.736406   0.226330  -3.254  0.00114 **
## DSRI         0.090829   0.019241   4.721 2.35e-06 ***
## GMI          0.034552   0.016716   2.067  0.03874 *
## AQI          0.067753   0.012514   5.414 6.16e-08 ***
## SGI          0.197038   0.040583   4.855 1.20e-06 ***
## DEPI        -0.004098   0.182147  -0.022  0.98205
## SGAI         0.035921   0.013894   2.585  0.00973 **
## ACCR         3.547205   0.437171   8.114 4.90e-16 ***
## LEVI         0.005422   0.040957   0.132  0.89469
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1230.7  on 887  degrees of freedom
## Residual deviance: 1050.8  on 879  degrees of freedom
## AIC: 1068.8
##
## Number of Fisher Scoring iterations: 4
```

```
#test
logitModel.fit.ROSE.predictions <- predict(logitModel.fit.ROSE, newdata =
test_rose,  type = "response")
#confusion matrix
predict_logitModel.ROSE.fit<- rep(0, length(logitModel.fit.ROSE.predictions))
#all zeros for predictions
predict_logitModel.ROSE.fit[logitModel.fit.ROSE.predictions >= 0.5 ] <- 1
#replace with 1 of prob >= 0.5
table(predict_logitModel.ROSE.fit,actual_rose , dnn= c("Predictions",
"Actual"))
```

```
##            Actual
## Predictions   0   1
##           0 142  74
##           1  39  96
```

```
#EVALUATION
print("Accuracy/Recall/Precision/F-1 score NOT using balancing technique")
```

```
my.accuracy(actual2, predict_logitModel.fit)
```

```
## [1] 0.9613095
```

```
my.recall(actual2, predict_logitModel.fit)
```

```
## [1] 0.1538462
```

```
my.precision(actual2, predict_logitModel.fit)
```

```
## [1] 0.5
```

```
my.f1(actual2, predict_logitModel.fit)
```

```
## [1] 0.2352941
```

```
print("Accuracy/Recall/Precision/F-1 score using UNDER SAMPLING")
```

```
my.accuracy(actual_down, predict_logitModel.UNDER.fit)
```

```
## [1] 0.826087
```

```
my.recall(actual_down, predict_logitModel.UNDER.fit)
```

```
## [1] 0.6363636
```

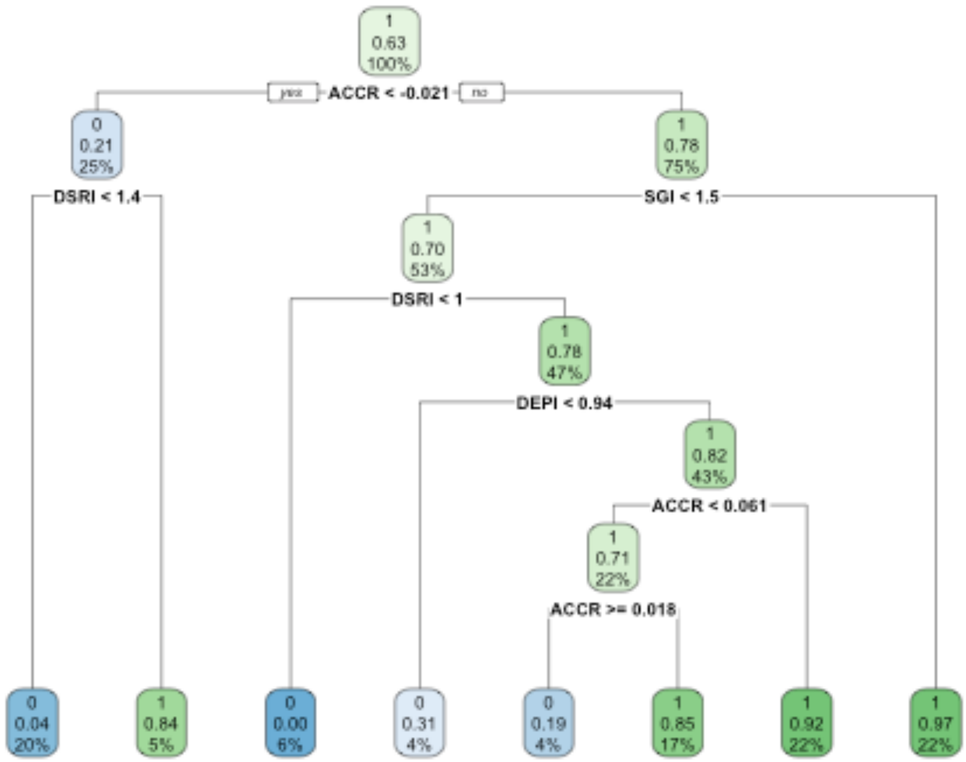
```
my.precision(actual_down, predict_logitModel.UNDER.fit)

## [1] 1

my.f1(actual_down, predict_logitModel.UNDER.fit)

## [1] 0.7777778

print("Accuracy/Recall/Precision/F-1 score using OVER SAMPLING")

my.accuracy(actual_up, predict_logitModel.OVER.fit)

## [1] 0.8430493

my.recall(actual_up, predict_logitModel.OVER.fit)

## [1] 0.8187702

my.precision(actual_up, predict_logitModel.OVER.fit)

## [1] 0.8377483

my.f1(actual_up, predict_logitModel.OVER.fit)

## [1] 0.8281506

print("Accuracy/Recall/Precision/F-1 score using BOTH UNDER AND OVER
SAMPLING")

my.accuracy(actual_both, predict_logitModel.BOTH.fit)

## [1] 0.8583333

my.recall(actual_both, predict_logitModel.BOTH.fit)

## [1] 0.8268156

my.precision(actual_both, predict_logitModel.BOTH.fit)

## [1] 0.8809524

my.f1(actual_both, predict_logitModel.BOTH.fit)

## [1] 0.8530259

print("Accuracy/Recall/Precision/F-1 score using ROSE")

my.accuracy(actual_rose, predict_logitModel.ROSE.fit)

## [1] 0.6780627

my.recall(actual_rose, predict_logitModel.ROSE.fit)

## [1] 0.5647059

my.precision(actual_rose, predict_logitModel.ROSE.fit)
```

```
## [1] 0.7111111

my.f1(actual_rose, predict_logitModel.ROSE.fit)

## [1] 0.6295082

#Model with highest F-1 score is obtained while using BOTH under and over
sampling. It’s important to notice that all models with the different
balancing techniques have a much greater F-1 score compared to the model
using the unbalanced data.
```

	UNBALANCED	UNDERSAMPLE	OVERSAMPLE	BOTH	ROSE
Accuracy	0.9613095	0.826087	0.8430493	0.8583333	0.6780627
Recall	0.1538462	0.6363636	0.8187702	0.8268156	0.5647059
Precision	0.5	1	0.8377483	0.8809524	0.7111111
F-1 Score	0.2352941	0.77777781	0.8281506	0.8530259	0.6295082

CLASSIFICATION & REGRESSION TREE(CART model)-using sample data

```
Balanced_Data <- ovun.sample(C_Manipulator ~., data = sample_data, method =
"over", N = 500)$data

set.seed(123)
indx <- sample(2, nrow(Balanced_Data), replace = T, prob = c(0.7, 0.3))
train <- Balanced_Data [indx == 1, ]
test <- Balanced_Data [indx == 2, ]

library (rpart)

decision_tree <- rpart(C_Manipulator ~., data = train)
library(rpart.plot)

rpart.plot(decision_tree)
```




```
decision_tree_pred <- predict(decision_tree, test, type = "prob")
decision_tree_class <- predict(decision_tree, test, type = "class")
actual <- test$C_Manipulator
table(decision_tree_class, actual, dnn= c("predictions", "actual"))

##              actual
## predictions  0  1
##            0 36  7
##            1 14 86

my.accuracy(actual, decision_tree_class)

## [1] 0.8531469

my.recall(actual, decision_tree_class)

## [1] 0.9247312

my.precision(actual, decision_tree_class)

## [1] 0.86

my.f1(actual, decision_tree_class)

## [1] 0.8911917
```

	OVERSAMPLED
Accuracy	0.8531469
Recall	0.9247312
Precision	0.86
F-1 Score	0.8911917

RANDOM FOREST-using sample data

```
set.seed(123)
library(randomForest)

Balanced_Data <- ovun.sample(C_Manipulator ~., data = sample_data, method =
"over", N = 500)$data

rf <- randomForest(C_Manipulator ~ ., data = Balanced_Data, mtry =
sqrt(ncol(Balanced_Data)-1), ntree = 300, proximity = T, importance = T)
rf

##
## Call:
##  randomForest(formula = C_Manipulator ~ ., data = Balanced_Data,      mtry
= sqrt(ncol(Balanced_Data) - 1), ntree = 300, proximity = T,      importance
= T)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 1.8%
## Confusion matrix:
##     0   1 class.error
## 0 172   9  0.04972376
## 1   0 319  0.00000000

#recall
319/(319+9) # = 0.972561

## [1] 0.972561

#precision
319/(319+0)  # = 1

## [1] 1

 #f-score
2 * 1 * 0.972561 / (1 + 0.972561) # = 0.9860897

 ## [1] 0.9860897
```

	OVERSAMPLED
Accuracy	0.8531469
Recall	0.972561
Precision	1
F-1 Score	0.9860897

Conclusion: The final recommendation for predicting earnings manipulation would be to use balanced data and then deploy a random forest model. This model yields the highest F-1 Score of all models tried, which best measures accuracy of results for this case.

	RF-Oversampled	Decision Tree-Oversampled	Logistic Reg-Unbalanced	Logistic Reg-Undersampled	Logistic Reg-Oversampled	Logistic Reg-Both	Logistic Reg-Rose
F-1 Score	0.9860897	0.8911917	0.2352941	0.77777781	0.8281506	0.8530259	0.6295082