Assignment-3: Predicting Earning Manipulations by Indian Firms using Machine Learning Algorithms

Vineet Srivastava-671461219, Mithilesh Bhutada-677760107, Taksh Ahuja-670941620

```
options(warn=-1)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.1
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readxl)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(dplyr)
library(ROCR)
library(ROSE)
```

```
## Loaded ROSE 0.0-4
```

```
library(DMwR)
```

```
## Loading required package: grid
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(UBL)
```

```
## Loading required package: MBA
## Loading required package: gstat
## Loading required package: automap
## Loading required package: sp
## Loading required package: randomForest
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
library(partykit)
```

```
## Loading required package: libcoin
## Loading required package: mvtnorm
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(tinytex)
```

Data Insights and Exploratory Data Analysis on Complete-Dataset

```
pred_manipulators_dataset <- read_excel('predicting_manipulators_dataset.xlsx',
                                        sheet = 'Complete Data')

head(pred_manipulators_dataset)
```

```
## # A tibble: 6 x 11
##   Company ~1  DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI Manip~2 C-MAN~3
##        <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <chr>     <dbl>
## 1          1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16  Yes           1
## 2          2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987 Yes           1
```

```
## 3           3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26  Yes            1
## 4           4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681 Yes            1
## 5           5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939 Yes            1
## 6           6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54  Yes            1
## # ... with abbreviated variable names 1: ‘Company ID‘, 2: Manipulater,
## #   3: ‘C-MANIPULATOR‘
```

```r
dim(pred_manipulators_dataset) # total 1239 rows and 11 columns
```

```
## [1] 1239   11
```

```r
sum(is.na(pred_manipulators_dataset)) # No NULL Values in entire dataset
```

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

```r
table(pred_manipulators_dataset$Manipulater) #No:1200,Yes:39,dataset is imbalanced
```

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

```r
names(pred_manipulators_dataset)[11] <- 'manipulator_target'
names(pred_manipulators_dataset)[1] <- 'company_ID'

pred_manipulators_dataset <- subset(pred_manipulators_dataset,
                                    select = -c(company_ID, Manipulater))

head(pred_manipulators_dataset) # removed unwanted columns from dataset
```

```
## # A tibble: 6 x 9
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipulator_target
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>              <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16                   1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987                  1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26                   1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681                  1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939                  1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54                   1
```

```r
cleaned_complete_df <- pred_manipulators_dataset
cor(cleaned_complete_df) # computing correlation coefficients
```

```
##                           DSRI           GMI           AQI         SGI
## DSRI                1.00000000 -0.0282895385 -0.0121965389 -0.14516443
## GMI                -0.02828954  1.0000000000  0.0002094946 -0.02436305
## AQI                -0.01219654  0.0002094946  1.0000000000 -0.02613459
## SGI                -0.14516443 -0.0243630469 -0.0261345892  1.00000000
## DEPI                0.02354412 -0.0113275767 -0.0212416146 -0.07044260
## SGAI                0.47076403 -0.0344170374  0.0037123159 -0.05094143
## ACCR                0.01516566 -0.0036744749 -0.0454238270  0.07040804
```

```
## LEVI                  0.15463160 -0.0697562975  0.0702730157  0.06706352
## manipulator_target  0.28512066  0.1333695919  0.1346420085  0.19947358
##                           DEPI          SGAI          ACCR        LEVI
## DSRI                 0.02354412  0.470764025  0.015165656  0.15463160
## GMI                 -0.01132758 -0.034417037 -0.003674475 -0.06975630
## AQI                 -0.02124161  0.003712316 -0.045423827  0.07027302
## SGI                 -0.07044260 -0.050941427  0.070408038  0.06706352
## DEPI                 1.00000000 -0.067247329 -0.016613359 -0.01271157
## SGAI                -0.06724733  1.000000000 -0.090667950  0.02174950
## ACCR                -0.01661336 -0.090667950  1.000000000 -0.01163113
## LEVI                -0.01271157  0.021749500 -0.011631128  1.00000000
## manipulator_target -0.03686290  0.203448761  0.109509864  0.13803432
##                    manipulator_target
## DSRI                        0.2851207
## GMI                         0.1333696
## AQI                         0.1346420
## SGI                         0.1994736
## DEPI                       -0.0368629
## SGAI                        0.2034488
## ACCR                        0.1095099
## LEVI                        0.1380343
## manipulator_target          1.0000000
```

```r
#(DSRI,SGI,SGAI are correlated good with target variable, DEPI,LEVI is weakly correlated,
# GMI,AQI,ACCR have almost same correlation coefficients )

head(cleaned_complete_df)
```

```
## # A tibble: 6 x 9
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipulator_target
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>              <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16                   1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987                  1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26                   1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681                  1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939                  1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54                   1
```

Quest- 1: Do you think the Beneish model developed in 1999 will still be

relevant to Indian data?

About Beneish Model:

1) The Beneish Model was created to detect financial frauds and is estimated using the M-Score.
2) The M-score mathematical formula is given as: 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
3) A M-Score of less than -1.78 means non-manipulator else manipulator

```r
Quest1_data <- cleaned_complete_df

Quest1_data$ben_model_score <- (-4.84+(0.92*Quest1_data$DSRI)+
```

```r
                                    (0.528* Quest1_data$GMI)+
                                    (0.404* Quest1_data$AQI)+
                                    (0.892*Quest1_data$SGI) +
                                    (0.115* Quest1_data$DEPI) -
                                    (0.172*Quest1_data$SGAI)+
                                    (4.679*Quest1_data$ACCR)-
                                    0.327*Quest1_data$LEVI)
head(Quest1_data)
```

```
## # A tibble: 6 x 10
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipulator_target ben_m~1
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>              <dbl>   <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16                   1  -0.800
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987                  1   8.12
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26                   1  -1.79
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681                  1  -0.886
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939                  1  -2.04
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54                   1  -1.56
## # ... with abbreviated variable name 1: ben_model_score
```

```r
sum(Quest1_data$ben_model_score)/length(Quest1_data$ben_model_score)
```

```
## [1] -2.402568
```

```r
# Avg M-Score is -2.45 which clearly Suggests that dataset has high %age of non-manipulators

# add prediction column based on M-Score
Quest1_data$ben_pred <- NA
Quest1_data$ben_pred[Quest1_data$ben_model_score > -1.78] <- 1
Quest1_data$ben_pred[Quest1_data$ben_model_score < -1.78] <- 0

head(Quest1_data)
```

```
## # A tibble: 6 x 11
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipula~1 ben_m~2 ben_p~3
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>      <dbl>   <dbl>   <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16           1  -0.800       1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987          1   8.12        1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26           1  -1.79        0
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681          1  -0.886       1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939          1  -2.04        0
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54           1  -1.56        1
## # ... with abbreviated variable names 1: manipulator_target,
## #   2: ben_model_score, 3: ben_pred
```

```r
# So we included a column as prediction of Beneish model giving 1 or 0
# (1 for manipulators and 0 for non manipulators)

# Lets calculate the accuracy of Beneish model based on our M-Score Prediction

cf_table <- table(Quest1_data$ben_pred, Quest1_data$manipulator_target,
                  dnn = c("Actual", "Prediction"))
confusionMatrix(cf_table, positive = "1" )
```

```
## Confusion Matrix and Statistics
##
##       Prediction
## Actual    0    1
##      0 1029    8
##      1  171   31
##
##                Accuracy : 0.8555
##                  95% CI : (0.8347, 0.8746)
##     No Information Rate : 0.9685
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2159
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.79487
##             Specificity : 0.85750
##          Pos Pred Value : 0.15347
##          Neg Pred Value : 0.99229
##              Prevalence : 0.03148
##          Detection Rate : 0.02502
##    Detection Prevalence : 0.16303
##       Balanced Accuracy : 0.82619
##
##        'Positive' Class : 1
##
```

```
# Accuracy : 0.8555
# Sensitivity : 0.79487
# Specificity : 0.85750

# So the accuracy of the Beneish model is found to be 85 % and sensitivity is
# 79% which is a good performance.
# Hence we can say that as per current sample dataset, the Beneish model for
# Indian data could be still be relevant.
# However we need to look at other ML algorithms to come to a conclusion and
# also our dataset is highly imbalanced.
```

Quest-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 modelling 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?

Answer:

Problems with imbalanced data-set:

With imbalanced dataset, generally ML classification Models becomes biased and as a result model performance (especially Recall/Precision) declines.

Solutions:

1. Under-sampling: Balances data by eliminating majority class data-points. However downside of this is it leads to loss of information.
2. Oversampling: Balances the data by increasing data-points of minority class by replicating them to balance majority class. It can lead to over fitting.
3. Synthetic Minority Over-Sampling Technique (SMOTE): This AI based algorithm randomly selects data points from the k nearest neighbors of minority classes samples and balances the data as in majority class. It reduces Over-fitting problems.

Quest-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.

```
# read 1st 220 rows which has 39 manipulators and rest non-manipulators from
# pred_manipulators_dataset variable
# the pred_manipulators_dataset is cleaned data as per above EDA section

Quest3_data <- pred_manipulators_dataset[1:220,]
dim(Quest3_data)
```

```
## [1] 220   9
```

```
table(Quest3_data$manipulator_target, useNA = "ifany")
```

```
##
##   0   1
## 181  39
```

```r
head(Quest3_data)
```

```
## # A tibble: 6 x 9
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipulator_target
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>              <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16                   1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987                  1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26                   1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681                  1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939                  1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54                   1
```

```r
cor(Quest3_data)
```

```
##                           DSRI          GMI          AQI         SGI
## DSRI                1.00000000 -0.051398585 -0.025035920 -0.11513498
## GMI                -0.05139858  1.000000000  0.003257079  0.05359029
## AQI                -0.02503592  0.003257079  1.000000000 -0.04521795
## SGI                -0.11513498  0.053590290 -0.045217949  1.00000000
## DEPI               -0.04109005  0.080326850 -0.125499496 -0.18222707
## SGAI                0.50677969 -0.040368569 -0.008643188 -0.03710980
## ACCR               -0.08872504  0.002036972 -0.171337773  0.06943921
## LEVI                0.13788958 -0.078350875  0.123316416  0.05506581
## manipulator_target  0.29630908  0.149235600  0.168450195  0.23482900
##                           DEPI         SGAI         ACCR        LEVI
## DSRI               -0.04109005  0.506779695 -0.088725044  0.13788958
## GMI                 0.08032685 -0.040368569  0.002036972 -0.07835088
## AQI                -0.12549950 -0.008643188 -0.171337773  0.12331642
## SGI                -0.18222707 -0.037109803  0.069439214  0.05506581
## DEPI                1.00000000 -0.261457032  0.193143877 -0.10452810
## SGAI               -0.26145703  1.000000000 -0.263648477  0.01883895
## ACCR                0.19314388 -0.263648477  1.000000000 -0.02285318
## LEVI               -0.10452810  0.018838951 -0.022853184  1.00000000
## manipulator_target -0.06845840  0.197847484  0.218016773  0.16653204
##                    manipulator_target
## DSRI                        0.2963091
## GMI                         0.1492356
## AQI                         0.1684502
## SGI                         0.2348290
## DEPI                       -0.0684584
## SGAI                        0.1978475
## ACCR                        0.2180168
## LEVI                        0.1665320
## manipulator_target          1.0000000
```

```r
# from correlation coeff we chose DSRI,SGI,ACCR,SGAI,AQI based on theireffect on target variable

Quest3_data <- Quest3_data %>%
  select(DSRI,SGI,ACCR,SGAI,AQI,manipulator_target)

head(Quest3_data)
```

```
## # A tibble: 6 x 6
##    DSRI    SGI    ACCR   SGAI   AQI manipulator_target
##   <dbl>  <dbl>   <dbl>  <dbl> <dbl>              <dbl>
## 1 1.62   0.366 -0.167  1.62   7.19                   1
## 2 1     13.1    0.0605 5.20   1.00                   1
## 3 1      1.48   0.0367 0.648  1.24                   1
## 4 1.49   0.673  0.273  0.0929 0.466                  1
## 5 1      0.861  0.123  1.74   0.637                  1
## 6 0.906  1.79   0.0546 0.505  0.784                  1
```

```
# Now there are 3 approaches for this problem:
# Step-1: Building model on unbalanced data- Original dataset
# Step-2: Oversampling the data-set to counter imbalance data and build model
# Step-3: Data Balancing Using SMOTE approach and building model for the same

# Note: We will also calculate the Probability threshold cut-off for each of the above steps

##################################################################################################
# Step-1: Building model on unbalanced data- Original dataset
##################################################################################################

# splitting the sample dataset into train and test before creating the logistic regression model

set.seed(1234)
index <- sample(2, nrow(Quest3_data), replace = TRUE, prob = c(0.65,0.35))
train_sample_data <- Quest3_data[index == 1,]
test_sample_data <- Quest3_data[index == 2,]

# now lets create log regression model
# variable selection
null = glm(manipulator_target~1, data = train_sample_data, family = 'binomial')
full = glm(manipulator_target~., data = train_sample_data, family = 'binomial')

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

```
## Start:  AIC=150.58
## manipulator_target ~ 1
##
##        Df Deviance    AIC
## + DSRI  1   125.95 129.95
## + SGAI  1   136.29 140.29
## + SGI   1   139.89 143.89
## + ACCR  1   141.36 145.36
## <none>      148.58 150.58
## + AQI   1   147.65 151.65
##
## Step:  AIC=129.95
## manipulator_target ~ DSRI
##
##        Df Deviance    AIC
## + SGI   1   107.87 113.87
## + ACCR  1   114.37 120.37
## + SGAI  1   118.70 124.70
## <none>      125.95 129.95
```

```
## + AQI    1   124.02 130.01
##
## Step:  AIC=113.87
## manipulator_target ~ DSRI + SGI
##
##        Df Deviance    AIC
## + ACCR  1   95.643 103.64
## + AQI   1  102.481 110.48
## + SGAI  1  102.615 110.61
## <none>     107.869 113.87
##
## Step:  AIC=103.64
## manipulator_target ~ DSRI + SGI + ACCR
##
##        Df Deviance     AIC
## + AQI   1   86.681  96.681
## + SGAI  1   93.376 103.376
## <none>      95.643 103.643
##
## Step:  AIC=96.68
## manipulator_target ~ DSRI + SGI + ACCR + AQI
##
##        Df Deviance    AIC
## + SGAI  1   84.586 96.586
## <none>      86.681 96.681
##
## Step:  AIC=96.59
## manipulator_target ~ DSRI + SGI + ACCR + AQI + SGAI
```

```
##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI +
##     SGAI, family = "binomial", data = train_sample_data)
##
## Coefficients:
## (Intercept)         DSRI          SGI         ACCR          AQI         SGAI
##     -8.9974       2.4607       2.4707       6.4986       0.2732       0.7194
##
## Degrees of Freedom: 152 Total (i.e. Null);  147 Residual
## Null Deviance:       148.6
## Residual Deviance: 84.59     AIC: 96.59
```

```
step(null, scope=list(lower=null, upper=full), direction="backward")
```

```
## Start:  AIC=150.58
## manipulator_target ~ 1
```

```
##
## Call:  glm(formula = manipulator_target ~ 1, family = "binomial", data = train_sample_data)
##
## Coefficients:
## (Intercept)
##      -1.453
##
```

```
## Degrees of Freedom: 152 Total (i.e. Null);  152 Residual
## Null Deviance:       148.6
## Residual Deviance: 148.6     AIC: 150.6
```

```
step(full,scope =list(lower=null,upper=full),direction ="both")
```

```
## Start:  AIC=96.59
## manipulator_target ~ DSRI + SGI + ACCR + SGAI + AQI
##
##        Df Deviance     AIC
## <none>      84.586  96.586
## - SGAI  1   86.681  96.681
## - AQI   1   93.376 103.376
## - ACCR  1   97.454 107.454
## - SGI   1  105.579 115.579
## - DSRI  1  110.854 120.854
```

```
##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + ACCR + SGAI +
##     AQI, family = "binomial", data = train_sample_data)
##
## Coefficients:
## (Intercept)         DSRI          SGI         ACCR         SGAI          AQI
##     -8.9974       2.4607       2.4707       6.4986       0.7194       0.2732
##
## Degrees of Freedom: 152 Total (i.e. Null);  147 Residual
## Null Deviance:       148.6
## Residual Deviance: 84.59     AIC: 96.59
```

```
# after running forward,backward and both variable selection method we found
# DSRI+SG!+ACCR+AQI variables important so we build model on them

log_reg <- glm(manipulator_target~DSRI + SGI + ACCR + AQI, data= train_sample_data, family = "binomial")
summary(log_reg)
```

```
##
## Call:
## glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI, family = "binomial",
##     data = train_sample_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5976  -0.4362  -0.2868  -0.1714   2.2452
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.18659    1.60841  -5.090 3.58e-07 ***
## DSRI         2.43973    0.67236   3.629 0.000285 ***
## SGI          2.40252    0.74622   3.220 0.001284 **
## ACCR         7.10047    2.03335   3.492 0.000479 ***
## AQI          0.29727    0.09621   3.090 0.002004 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 148.581  on 152  degrees of freedom
## Residual deviance:  86.681  on 148  degrees of freedom
## AIC: 96.681
##
## Number of Fisher Scoring iterations: 8
```

```
# Accuracy on Train data (unbalanced)

# Null deviance: 148.581  on 152  degrees of freedom
# Residual deviance:  86.681  on 148  degrees of freedom
# AIC: 96.681

# lets predict the accuracy on test data now

p <- predict(log_reg, test_sample_data, type = 'response')
pred <- ifelse(p>0.5, 1, 0)
tab<- table(pred,test_sample_data$manipulator_target, dnn = c("Actual", "Prediction"))
tab
```

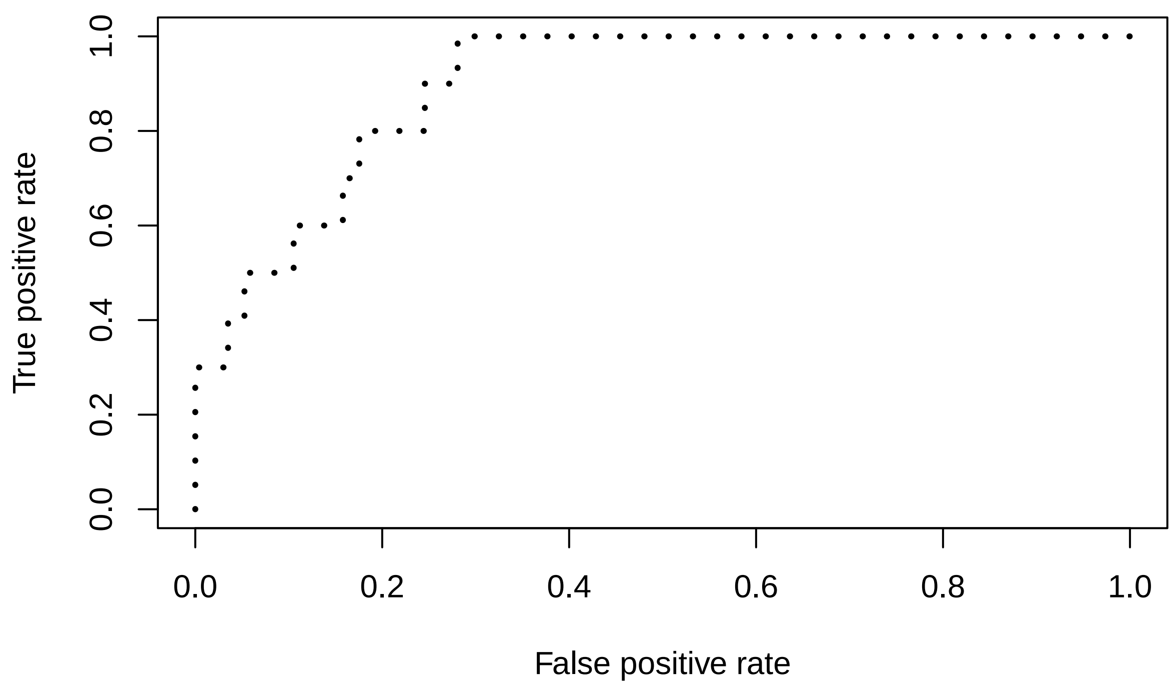
```
##       Prediction
## Actual  0  1
##      0 57  7
##      1  0  3
```

```
cm <- confusionMatrix(as.factor(pred),as.factor(test_sample_data$manipulator_target),positive = "1" )
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 57  7
##          1  0  3
##
##                Accuracy : 0.8955
##                  95% CI : (0.7965, 0.957)
##     No Information Rate : 0.8507
##     P-Value [Acc > NIR] : 0.19855
##
##                   Kappa : 0.4217
##
##  Mcnemar's Test P-Value : 0.02334
##
##             Sensitivity : 0.30000
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.89062
##              Prevalence : 0.14925
##          Detection Rate : 0.04478
```

```
##    Detection Prevalence : 0.04478
##       Balanced Accuracy : 0.65000
##
##        'Positive' Class : 1
##
```

```
# Accuracy : 0.8955
# Sensitivity : 0.30000
# Specificity : 1.00000

# lets get the threshold/cut-off point in ROC curve and then do prediction on test-sample data

pred_roc= prediction(p,test_sample_data$manipulator_target)
perf_roc = performance(pred_roc,"tpr","fpr")

# Plotting the ROC curve
plot(perf_roc, col = "black", lty = 3, lwd = 3)
```




```
# Taking Reference from "Random_Forest_and_Evaluation.r" document in Blackbaord under R documents


mydistance <- function(x,y,p){
  d=(x-0)^2+(y-1)^2 # given the points (x, y), compute the distance to the corner point (0,1)
  ind <- which(d==min(d)) # Find the minimum distance and its index
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
```

```
}

opt.cut <- function(perf){
  cut.ind <- mapply(FUN = mydistance,
                    perf@x.values, perf@y.values,perf@alpha.values)
}

Output <- opt.cut(perf_roc)

Threshold <- Output[,1]["cutoff"] #0.07806067

pred_roc_cut_point <- ifelse(p>Threshold,1,0)
tab<-table(pred_roc_cut_point, test_sample_data$manipulator_target, dnn = c("Predicted","Actual"))
confusionMatrix(tab,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted  0  1
##         0 43  2
##         1 14  8
##
##                Accuracy : 0.7612
##                  95% CI : (0.6414, 0.8569)
##     No Information Rate : 0.8507
##     P-Value [Acc > NIR] : 0.98249
##
##                   Kappa : 0.3709
##
##  Mcnemar's Test P-Value : 0.00596
##
##             Sensitivity : 0.8000
##             Specificity : 0.7544
##          Pos Pred Value : 0.3636
##          Neg Pred Value : 0.9556
##              Prevalence : 0.1493
##          Detection Rate : 0.1194
##    Detection Prevalence : 0.3284
##       Balanced Accuracy : 0.7772
##
##        'Positive' Class : 1
##
```

```
#  Accuracy : 0.7612
#  Sensitivity : 0.8000
#  Specificity : 0.7544

# Observation: Sensitivity improved

# So we see that for unbalanced sample data, we created a logistic regression model
# and tried 2 approaches- one with and without probability threshold using ROC curve
```

```r
####### Final Observation on Unbalanced Log regression model ##################

################################################################################
# Unbalanced Data-set- Log regression model performance evaluation on Train data
################################################################################

# Null deviance: 148.581  on 152  degrees of freedom
# Residual deviance:  86.681  on 148  degrees of freedom
# AIC: 96.681

################################################################################
# Unbalanced Data-set- Log regression model performance evaluation on Test-data
################################################################################

# 1. Without Probability Threshold calculation

# Accuracy : 0.8955
# Sensitivity : 0.30000
# Specificity : 1.0000

# 2. With Probability Threshold calculation

#  Accuracy : 0.7612
#  Sensitivity : 0.8000
#  Specificity : 0.7544




###############################################################################################
# Step-2: Oversampling the data-set to counter imbalance data and build model
###############################################################################################
# We chose oversampling as there is no loss of information here since our data-set
# is already small (sample dataset)
over_sample <- ovun.sample(manipulator_target~., data = Quest3_data, method = "over", N= 500)$data
table(over_sample$manipulator_target)
```

```
##
##   0   1
## 181 319
```

```r
# splitting the dataset into train and test before creating the logistic regression model

set.seed(1234)
index <- sample(2, nrow(over_sample), replace = TRUE, prob = c(0.70,0.30))
train_sample_data <- over_sample[index == 1,]
test_sample_data <- over_sample[index == 2,]

# now let's build the Logistic regression model
null = glm(manipulator_target~1, data = train_sample_data, family = 'binomial')
full = glm(manipulator_target~., data = train_sample_data, family = 'binomial')

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

```
## Start:  AIC=471.63
## manipulator_target ~ 1
##
##        Df Deviance    AIC
## + DSRI  1   425.80 429.80
## + SGI   1   451.33 455.33
## + SGAI  1   452.85 456.85
## + AQI   1   457.21 461.21
## + ACCR  1   462.76 466.76
## <none>      469.63 471.63
##
## Step:  AIC=429.8
## manipulator_target ~ DSRI
##
##        Df Deviance    AIC
## + SGI   1   375.83 381.83
## + ACCR  1   396.24 402.24
## + AQI   1   405.78 411.78
## + SGAI  1   420.90 426.90
## <none>      425.80 429.80
##
## Step:  AIC=381.83
## manipulator_target ~ DSRI + SGI
##
##        Df Deviance    AIC
## + AQI   1   306.31 314.31
## + ACCR  1   352.55 360.55
## + SGAI  1   361.52 369.52
## <none>      375.83 381.83
##
## Step:  AIC=314.31
## manipulator_target ~ DSRI + SGI + AQI
##
##        Df Deviance    AIC
## + ACCR  1   238.59 248.59
## + SGAI  1   296.75 306.75
## <none>      306.31 314.31
##
## Step:  AIC=248.59
## manipulator_target ~ DSRI + SGI + AQI + ACCR
##
##        Df Deviance    AIC
## <none>      238.59 248.59
## + SGAI  1   238.49 250.49
```

```
##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + AQI + ACCR, family = "binomial",
##     data = train_sample_data)
##
## Coefficients:
## (Intercept)         DSRI          SGI          AQI         ACCR
##    -10.8353       3.6775       4.6364       0.6285       9.2733
##
## Degrees of Freedom: 351 Total (i.e. Null);  347 Residual
```

```
## Null Deviance:      469.6
## Residual Deviance: 238.6    AIC: 248.6
```

```r
step(null, scope=list(lower=null, upper=full), direction="backward")
```

```
## Start:  AIC=471.63
## manipulator_target ~ 1

##
## Call:  glm(formula = manipulator_target ~ 1, family = "binomial", data = train_sample_data)
##
## Coefficients:
## (Intercept)
##      0.4626
##
## Degrees of Freedom: 351 Total (i.e. Null);  351 Residual
## Null Deviance:      469.6
## Residual Deviance: 469.6    AIC: 471.6
```

```r
step(full,scope =list(lower=null,upper=full),direction ="both")
```

```
## Start:  AIC=250.49
## manipulator_target ~ DSRI + SGI + ACCR + SGAI + AQI
##
##        Df Deviance    AIC
## - SGAI  1   238.59 248.59
## <none>      238.49 250.49
## - ACCR  1   296.75 306.75
## - AQI   1   346.06 356.06
## - SGI   1   347.98 357.98
## - DSRI  1   379.48 389.48
##
## Step:  AIC=248.59
## manipulator_target ~ DSRI + SGI + ACCR + AQI
##
##        Df Deviance    AIC
## <none>      238.59 248.59
## + SGAI  1   238.49 250.49
## - ACCR  1   306.31 314.31
## - SGI   1   348.18 356.18
## - AQI   1   352.55 360.55
## - DSRI  1   424.84 432.84

##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI, family = "binomial",
##     data = train_sample_data)
##
## Coefficients:
## (Intercept)         DSRI          SGI         ACCR          AQI
##    -10.8353       3.6775       4.6364       9.2733       0.6285
##
## Degrees of Freedom: 351 Total (i.e. Null);  347 Residual
## Null Deviance:      469.6
## Residual Deviance: 238.6    AIC: 248.6
```

```r
# Observation: Again we notice that: DSRI + SGI + ACCR + SGAI + AQI are imp variables
# from variable selection steps as above

log_reg_over_sample <- glm(manipulator_target~DSRI + SGI + ACCR + AQI, data= train_sample_data,
                           family = "binomial")
summary(log_reg_over_sample)
```

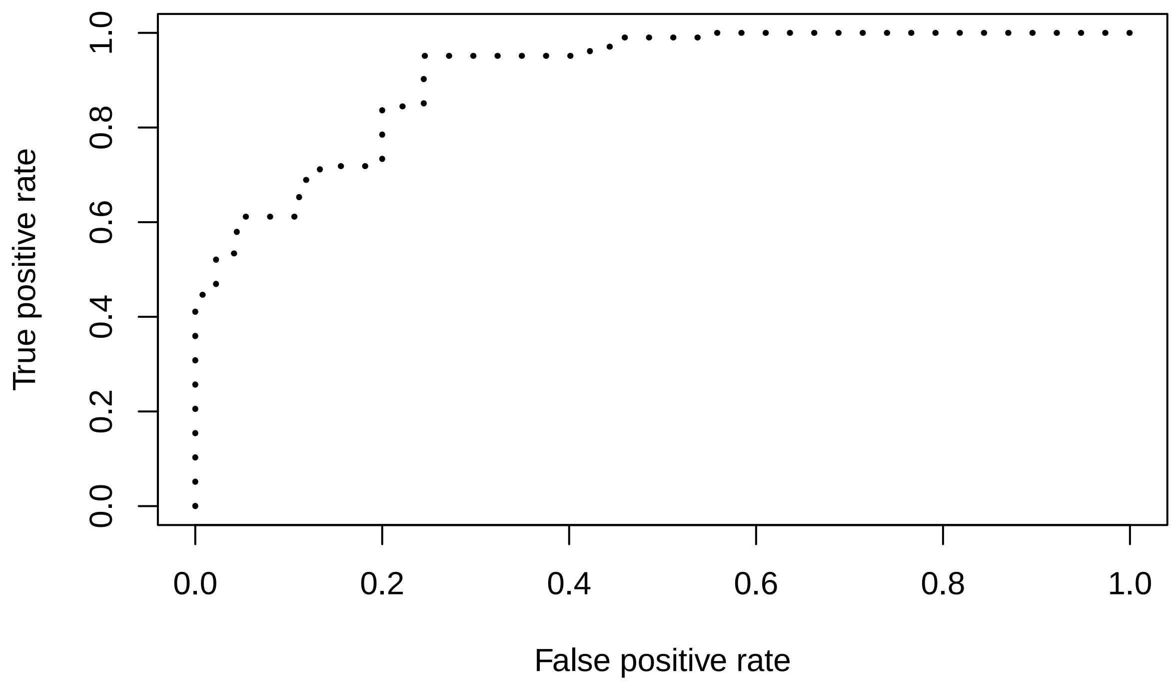
```
##
## Call:
## glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI, family = "binomial",
##     data = train_sample_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.9536  -0.4388   0.0000   0.5723   1.8989
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.8353     1.3381  -8.098 5.61e-16 ***
## DSRI          3.6775     0.4861   7.565 3.87e-14 ***
## SGI           4.6364     0.6699   6.921 4.49e-12 ***
## ACCR          9.2733     1.4067   6.592 4.33e-11 ***
## AQI           0.6285     0.1096   5.733 9.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 469.63  on 351  degrees of freedom
## Residual deviance: 238.59  on 347  degrees of freedom
## AIC: 248.59
##
## Number of Fisher Scoring iterations: 9
```

```r
# Accurancy on Training Data-Oversampling

# Null deviance: 469.63  on 351  degrees of freedom
# Residual deviance: 238.59  on 347  degrees of freedom
# AIC: 248.59



# Lets predict the accuracy on test data for Oversampling condition


p_over <- predict(log_reg_over_sample, test_sample_data, type = 'response')
pred_over_sample <- ifelse(p_over>0.5, 1, 0)

tab_over<- table(pred_over_sample,test_sample_data$manipulator_target,
                 dnn = c("Actual", "Prediction"))
cm_over <- confusionMatrix(as.factor(pred_over_sample),
                           as.factor(test_sample_data$manipulator_target),positive = "1" )
```

```
cm_over
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 34  8
##          1 11 95
##
##                Accuracy : 0.8716
##                  95% CI : (0.8068, 0.9209)
##     No Information Rate : 0.6959
##     P-Value [Acc > NIR] : 4.623e-07
##
##                   Kappa : 0.6909
##
##  Mcnemar's Test P-Value : 0.6464
##
##             Sensitivity : 0.9223
##             Specificity : 0.7556
##          Pos Pred Value : 0.8962
##          Neg Pred Value : 0.8095
##              Prevalence : 0.6959
##          Detection Rate : 0.6419
##    Detection Prevalence : 0.7162
##       Balanced Accuracy : 0.8389
##
##        'Positive' Class : 1
##
```

```
# Performance Oversampling without Probability Threshold calculated:

# Accuracy : 0.8716
# Sensitivity : 0.9223
# Specificity :  0.7556



# lets get the threshold/cut-off point in ROC curve and then do prediction on
# test-sample data for oversampling condition

pred_roc_over= prediction(p_over,test_sample_data$manipulator_target)
perf_roc_over = performance(pred_roc_over,"tpr","fpr")

# Plotting the ROC curve
plot(perf_roc_over, col = "black", lty = 3, lwd = 3)
```




```
# Taking Reference from "Random_Forest_and_Evaluation.r" document in Blackbaord under R documents


mydistance <- function(x,y,p){
  d=(x-0)^2+(y-1)^2 # given the points (x, y), compute the distance to the corner point (0,1)
  ind <- which(d==min(d)) # Find the minimum distance and its index
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
}

opt.cut <- function(perf){
  cut.ind <- mapply(FUN = mydistance,
                    perf@x.values, perf@y.values,perf@alpha.values)
}

Output_over <- opt.cut(perf_roc_over)
Threshold_over <- Output_over[,1]["cutoff"] # 0.4755626


pred_roc_cut_point <- ifelse(p_over>Threshold,1,0)
tab<-table(pred_roc_cut_point, test_sample_data$manipulator_target, dnn = c("Predicted","Actual"))
confusionMatrix(tab,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted   0   1
```

```
##         0   9   0
##         1  36 103
##
##                Accuracy : 0.7568
##                  95% CI : (0.6795, 0.8235)
##     No Information Rate : 0.6959
##     P-Value [Acc > NIR] : 0.06221
##
##                   Kappa : 0.2581
##
##  Mcnemar's Test P-Value : 5.433e-09
##
##             Sensitivity : 1.0000
##             Specificity : 0.2000
##          Pos Pred Value : 0.7410
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6959
##          Detection Rate : 0.6959
##    Detection Prevalence : 0.9392
##       Balanced Accuracy : 0.6000
##
##        'Positive' Class : 1
##
```

```
# Accuracy : 0.7568
# Sensitivity : 1.0000
# Specificity : 0.2000


####### Final Observation on Oversampling Balanced Data-set: Log regression model ###################

################################################################################
# Balanced Data-set Oversampling- Log regression model performance evaluation on Train data
################################################################################

# Null deviance: 469.63  on 351  degrees of freedom
# Residual deviance: 238.59  on 347  degrees of freedom
# AIC: 248.59



################################################################################
# Balanced Data-set Oversampling- Log regression model performance evaluation on
# Test-data
################################################################################

# 1. Without Probability Threshold calculation

# Accuracy : 0.8716
# Sensitivity :0.9223
# Specificity : 0.7556

# 2. With Probability Threshold calculation
```

```r
# Accuracy : 0.7568
# Sensitivity : 1.0000
# Specificity : 0.2000



##############################################################################################
# Step-3: Data Balancing Using SMOTE approach and building Log regression model for the same
##############################################################################################



smote <- SmoteClassif(manipulator_target~.,as.data.frame(Quest3_data), "balance")
prop.table(table(smote$manipulator_target))
```

```
##
##        0        1
## 0.738255 0.261745
```

```r
str(smote)
```

```
## 'data.frame':    219 obs. of  6 variables:
##  $ DSRI              : num  1.145 0.968 0.36 0.928 1.201 ...
##  $ SGI               : num  0.879 1.206 1.014 1.164 0.806 ...
##  $ ACCR              : num  -0.0558 0.01966 -0.0353 0.00569 -0.01249 ...
##  $ SGAI              : num  1.145 0.787 0.39 1.054 0.708 ...
##  $ AQI               : num  1.049 1.094 0.914 1.125 0.96 ...
##  $ manipulator_target: chr  "0" "0" "0" "0" ...
```

```r
smote$manipulator_target = as.factor(smote$manipulator_target)

sum(is.na(smote)) # 70 null values so need to remove them
```

```
## [1] 70
```

```r
smote <- na.omit(smote)

# Lets build the logistic regression model on Smote Data

set.seed(1234)
index <- sample(2, nrow(smote), replace = TRUE, prob = c(0.70,0.30))
train_sample_data <- smote[index == 1,]
test_sample_data <- smote[index == 2,]

# variable selection
null = glm(manipulator_target~1, data = train_sample_data, family = 'binomial')
full = glm(manipulator_target~., data = train_sample_data, family = 'binomial')

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

```
## Start:  AIC=115.27
```

```
## manipulator_target ~ 1
##
##        Df Deviance    AIC
## + DSRI  1    96.62 100.62
## + SGI   1   105.88 109.88
## + SGAI  1   107.95 111.95
## + AQI   1   110.14 114.14
## <none>      113.27 115.27
## + ACCR  1   112.33 116.33
##
## Step:  AIC=100.62
## manipulator_target ~ DSRI
##
##        Df Deviance     AIC
## + SGI   1   83.252  89.252
## + ACCR  1   91.155  97.155
## + AQI   1   91.347  97.347
## + SGAI  1   93.703  99.703
## <none>      96.620 100.620
##
## Step:  AIC=89.25
## manipulator_target ~ DSRI + SGI
##
##        Df Deviance    AIC
## + AQI   1   74.412 82.412
## + ACCR  1   79.203 87.203
## <none>      83.252 89.252
## + SGAI  1   83.052 91.052
##
## Step:  AIC=82.41
## manipulator_target ~ DSRI + SGI + AQI
##
##        Df Deviance    AIC
## + ACCR  1   64.677 74.677
## <none>      74.412 82.412
## + SGAI  1   74.364 84.364
##
## Step:  AIC=74.68
## manipulator_target ~ DSRI + SGI + AQI + ACCR
##
##        Df Deviance    AIC
## <none>      64.677 74.677
## + SGAI  1   64.674 76.674
```

```
##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + AQI + ACCR, family = "binomial",
##     data = train_sample_data)
##
## Coefficients:
## (Intercept)         DSRI          SGI          AQI         ACCR
##     -7.8112       2.6845       2.1096       0.2313       8.0911
##
## Degrees of Freedom: 110 Total (i.e. Null);  106 Residual
## Null Deviance:       113.3
```

```
## Residual Deviance: 64.68     AIC: 74.68
```

```r
step(null, scope=list(lower=null, upper=full), direction="backward")
```

```
## Start:  AIC=115.27
## manipulator_target ~ 1

##
## Call:  glm(formula = manipulator_target ~ 1, family = "binomial", data = train_sample_data)
##
## Coefficients:
## (Intercept)
##      -1.342
##
## Degrees of Freedom: 110 Total (i.e. Null);  110 Residual
## Null Deviance:       113.3
## Residual Deviance: 113.3     AIC: 115.3
```

```r
step(full,scope =list(lower=null,upper=full),direction ="both")
```

```
## Start:  AIC=76.67
## manipulator_target ~ DSRI + SGI + ACCR + SGAI + AQI
##
##        Df Deviance     AIC
## - SGAI  1   64.677  74.677
## <none>      64.674  76.674
## - ACCR  1   74.364  84.364
## - SGI   1   77.969  87.969
## - AQI   1   79.202  89.202
## - DSRI  1   90.946 100.946
##
## Step:  AIC=74.68
## manipulator_target ~ DSRI + SGI + ACCR + AQI
##
##        Df Deviance     AIC
## <none>      64.677  74.677
## + SGAI  1   64.674  76.674
## - ACCR  1   74.412  82.412
## - AQI   1   79.203  87.203
## - SGI   1   80.324  88.324
## - DSRI  1  100.642 108.642

##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI, family = "binomial",
##     data = train_sample_data)
##
## Coefficients:
## (Intercept)         DSRI          SGI         ACCR          AQI
##     -7.8112       2.6845       2.1096       8.0911       0.2313
##
## Degrees of Freedom: 110 Total (i.e. Null);  106 Residual
## Null Deviance:       113.3
## Residual Deviance: 64.68     AIC: 74.68
```

```r
# Observation: Again we notice that: DSRI + SGI + ACCR + SGAI + AQI are imp variables
# from variable selection steps as above

log_reg_smote <- glm(manipulator_target~DSRI + SGI + ACCR + AQI, data= train_sample_data,
                     family = "binomial")
summary(log_reg_smote)
```

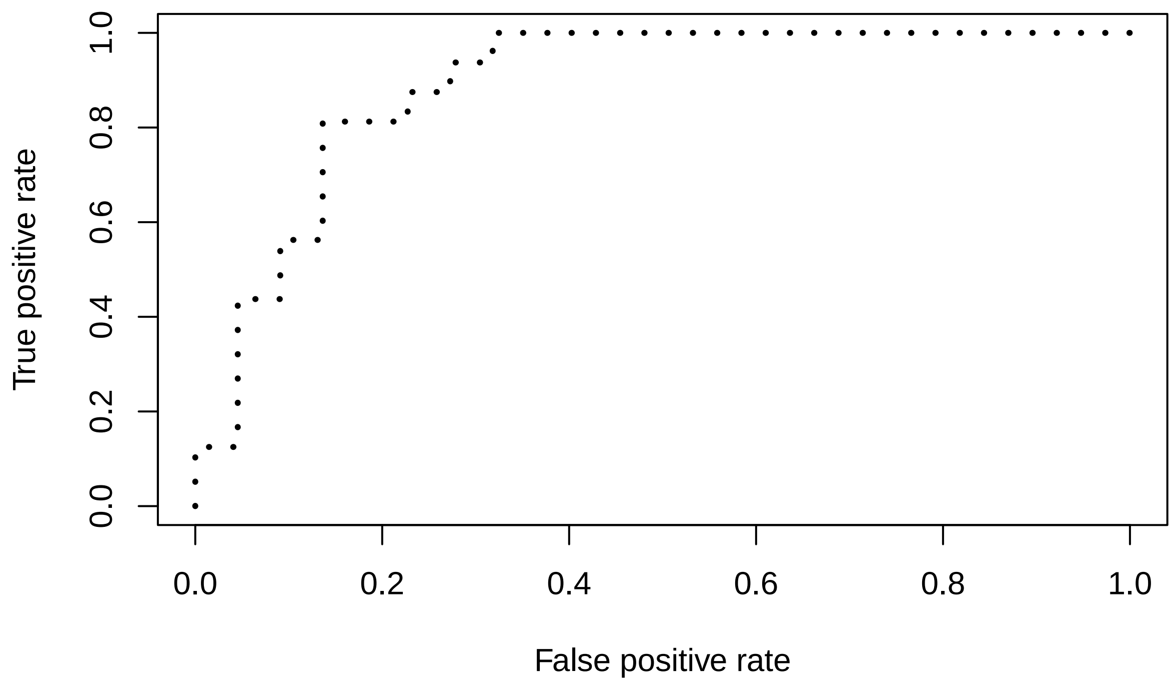
```
##
## Call:
## glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI, family = "binomial",
##     data = train_sample_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9049  -0.4476  -0.3001  -0.1613   2.1760
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.81121    1.96571  -3.974 7.08e-05 ***
## DSRI         2.68446    0.74925   3.583  0.00034 ***
## SGI          2.10964    1.08001   1.953  0.05078 .
## ACCR         8.09107    2.68333   3.015  0.00257 **
## AQI          0.23125    0.09815   2.356  0.01846 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 113.272  on 110  degrees of freedom
## Residual deviance:  64.677  on 106  degrees of freedom
## AIC: 74.677
##
## Number of Fisher Scoring iterations: 7
```

```r
# Observation: Accuracy on Train-data:

# Null deviance: 113.272  on 110  degrees of freedom
# Residual deviance:  64.677   on 106  degrees of freedom
# AIC: 74.677



# Lets predict the accuracy on test data for Smote condition


p_smote <- predict(log_reg_smote, test_sample_data, type = 'response')
pred_smote <- ifelse(p_smote>0.5, 1, 0)

tab_smote<- table(pred_smote,test_sample_data$manipulator_target,
                  dnn = c("Actual", "Prediction"))
cm_smote <- confusionMatrix(as.factor(pred_smote),
                            as.factor(test_sample_data$manipulator_target),positive = "1" )
```

```r
cm_smote
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20  9
##          1  2  7
##
##                Accuracy : 0.7105
##                  95% CI : (0.541, 0.8458)
##     No Information Rate : 0.5789
##     P-Value [Acc > NIR] : 0.06766
##
##                   Kappa : 0.3686
##
##  Mcnemar's Test P-Value : 0.07044
##
##             Sensitivity : 0.4375
##             Specificity : 0.9091
##          Pos Pred Value : 0.7778
##          Neg Pred Value : 0.6897
##              Prevalence : 0.4211
##          Detection Rate : 0.1842
##    Detection Prevalence : 0.2368
##       Balanced Accuracy : 0.6733
##
##        'Positive' Class : 1
##
```

```r
# Performance: Smote Balanced data without Probability Threshold calculated:

# Accuracy : 0.7105
# Sensitivity : 0.4375
# Specificity : 0.9091



# lets get the threshold/cut-off point in ROC curve and then do prediction on
# test-sample data for Smote condition


pred_roc_smote= prediction(p_smote,test_sample_data$manipulator_target)
perf_roc_smote = performance(pred_roc_smote,"tpr","fpr")

# Plotting the ROC curve
plot(perf_roc_smote, col = "black", lty = 3, lwd = 3)
```




```
# Taking Reference from "Random_Forest_and_Evaluation.r" document in Blackboard under R documents


mydistance <- function(x,y,p){
  d=(x-0)^2+(y-1)^2 # given the points (x, y), compute the distance to the corner point (0,1)
  ind <- which(d==min(d)) # Find the minimum distance and its index
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
}

opt.cut <- function(perf){
  cut.ind <- mapply(FUN = mydistance,
                    perf@x.values, perf@y.values,perf@alpha.values)
}

Output_smote <- opt.cut(perf_roc_smote)
Threshold_smote <- Output_smote[,1]["cutoff"] # 0.1892973


pred_roc_cut_point <- ifelse(p_smote>Threshold_smote,1,0)
tab<-table(pred_roc_cut_point, test_sample_data$manipulator_target,
           dnn = c("Predicted","Actual"))
confusionMatrix(tab,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##             Actual
```

```
## Predicted  0  1
##         0 19  4
##         1  3 12
##
##                Accuracy : 0.8158
##                  95% CI : (0.6567, 0.9226)
##     No Information Rate : 0.5789
##     P-Value [Acc > NIR] : 0.001809
##
##                   Kappa : 0.6189
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7500
##             Specificity : 0.8636
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8261
##              Prevalence : 0.4211
##          Detection Rate : 0.3158
##    Detection Prevalence : 0.3947
##       Balanced Accuracy : 0.8068
##
##        'Positive' Class : 1
##
```

```
# Observation: Accuracy on Test-data for Smote condition with Threshold/cut-off point calculated

# Accuracy : 0.8158
# Sensitivity :  0.7500
# Specificity : 0.8636



####### Final Observation on SMOTE Balanced Data-set: Log regression model ##################

################################################################################
# Balanced Data-set SMOTE- Log regression model performance evaluation on Train data
################################################################################

# Null deviance: 113.272  on 110  degrees of freedom
# Residual deviance: 64.677   on 106  degrees of freedom
# AIC: 74.677



################################################################################
# Balanced Data-set SMOTE- Log regression model performance evaluation on
# Test-data
################################################################################

# 1. Without Probability Threshold calculation

# Accuracy : 0.7105
# Sensitivity : 0.4375
```

```
# Specificity : 0.9091

# 2. With Probability Threshold calculation

# Accuracy : 0.8158
# Sensitivity : 0.75
# Specificity : 0.8636
```

Quest-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?

1. Both of these questions has been answered in above code.
2. For detailed Tabular comparison of accuracies of different models developed above, please refer the Presentation pdf shared along with this markdown file.

Quest-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.

1.We calculate the threshold point from the ROC curve using distance(0,1) approach in Quest-3. Please refer the same. 2. Let's calculate the Youden's Index which is given by ### Youden_Index = (sensitivity + Specificity -1)

```
youden_index <- function(sensitivity, Specificity){
  return (sensitivity + Specificity -1)
}

# For Unbalanced Data-set (refer the above code for sensitivity, Specificity values)
sensitivity = 0.8
Specificity = 0.7544

youden_index(sensitivity,Specificity)
```

```
## [1] 0.5544
```

```
# For SMOTE balanced Data-set (refer the above code for sensitivity, Specificity values)
sensitivity = 0.75
Specificity = 0.8636
youden_index(sensitivity,Specificity)
```

```
## [1] 0.6136
```

Quest-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.

Observation:

1. Since the dataset is highly imbalanced, it becomes essential to take into account both Accuracy and Recall/Sensitivity.
2. Hence going by above tables/observations for sample data-set we see that, Balanced Over-sampled Data gives better performance (Please refer Presentation pdf for Tabular comparison of different model's accuracies)
3. Hence alpha (Threshold/cut-off point) = 0.4755626, so M-Score = ln(alpha/(1-alpha)) = -0.098

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

```
tree_smote <- read_excel('predicting_manipulators_dataset.xlsx',
                sheet = 'Complete Data')
names(tree_smote)[1] <- "ID"
tree_smote$`C-MANIPULATOR`<- NULL
tree_smote$ID <- NULL
tree_smote$Manipulater<- factor(tree_smote$Manipulater)
table(tree_smote$Manipulater)
```

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

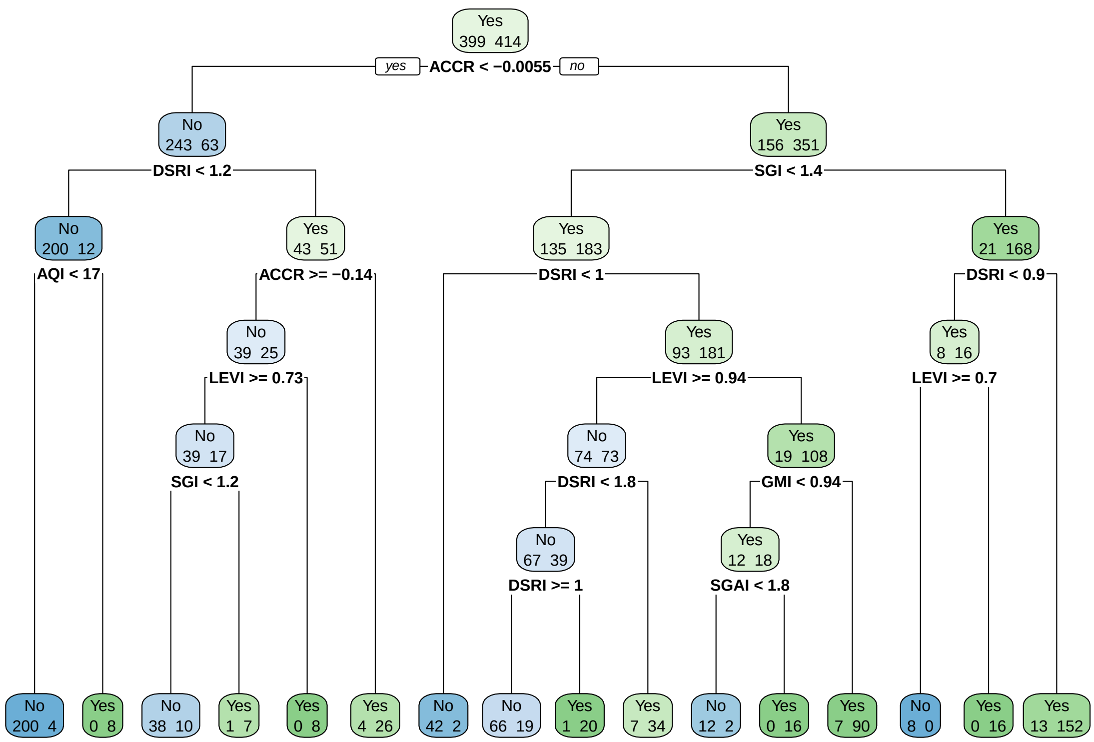
```
library(UBL)
smote <- SmoteClassif(Manipulater~.,as.data.frame(tree_smote), "balance")
#SmoteClassif balances the number of "Yes"and "No" in train$Manipulater

set.seed(1234)
ind <- sample(2, nrow(smote), replace = T, prob = c(0.65, 0.35))
train <- smote[ind==1,]
test <- smote[ind==2,]

#Smote on train data

prop.table(table(smote$Manipulater))
```

```
##
##  No Yes
## 0.5 0.5
```

```
table(smote$Manipulater)
```

```
##
##  No Yes
## 620 620
```

```r
library(rpart)

library(rpart.plot)
tree <- rpart(Manipulater~.,data = train,control=rpart.control(mincriterion=0.95,maxdepth = 6))
rpart.plot(tree,extra = 1)
```




```r
predictcart <- predict(tree,test,type = "class")
confusionMatrix(predictcart,test$Manipulater,positive = "Yes")
```

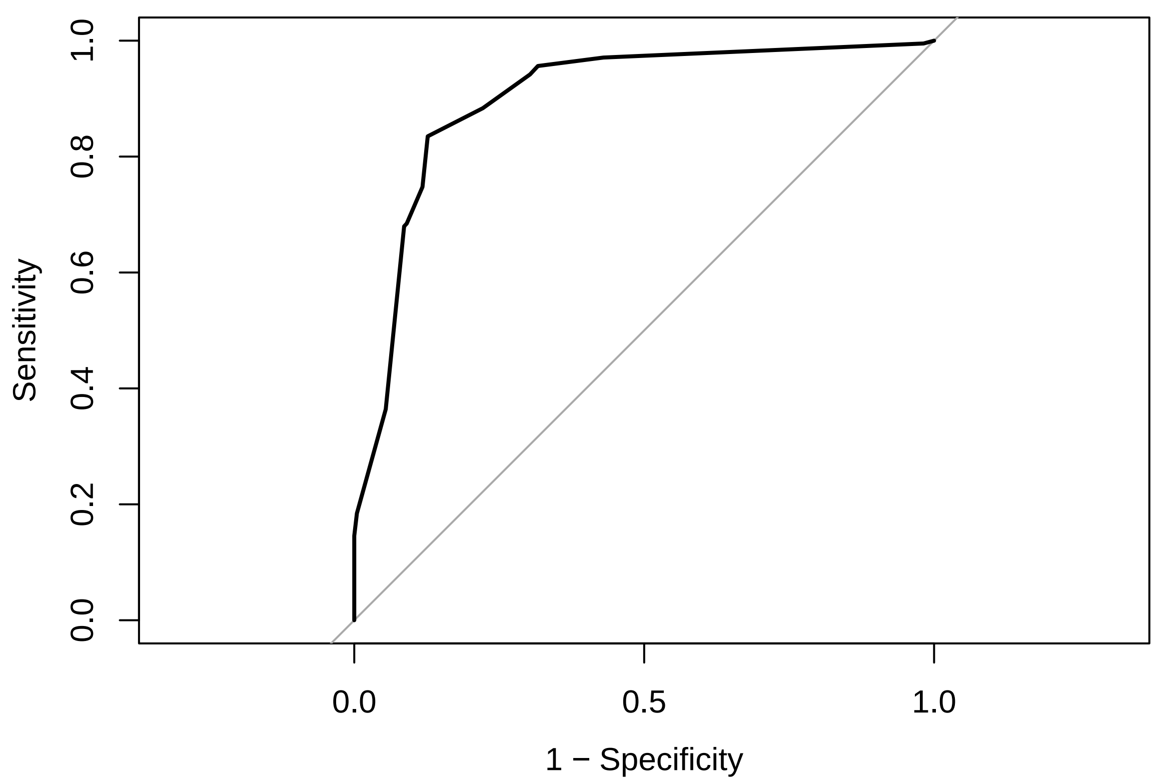
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  193  34
##        Yes  28 172
##
##                Accuracy : 0.8548
##                  95% CI : (0.8178, 0.8868)
##     No Information Rate : 0.5176
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.709
##
##  Mcnemar's Test P-Value : 0.5254
##
```

```
##             Sensitivity : 0.8350
##             Specificity : 0.8733
##          Pos Pred Value : 0.8600
##          Neg Pred Value : 0.8502
##              Prevalence : 0.4824
##          Detection Rate : 0.4028
##    Detection Prevalence : 0.4684
##       Balanced Accuracy : 0.8541
##
##        'Positive' Class : Yes
##
```

```
predictroc <- predict(tree,test,type = "prob")
roc(test$Manipulater,predictroc[,2],plot=TRUE,legacy.axes=TRUE)
```

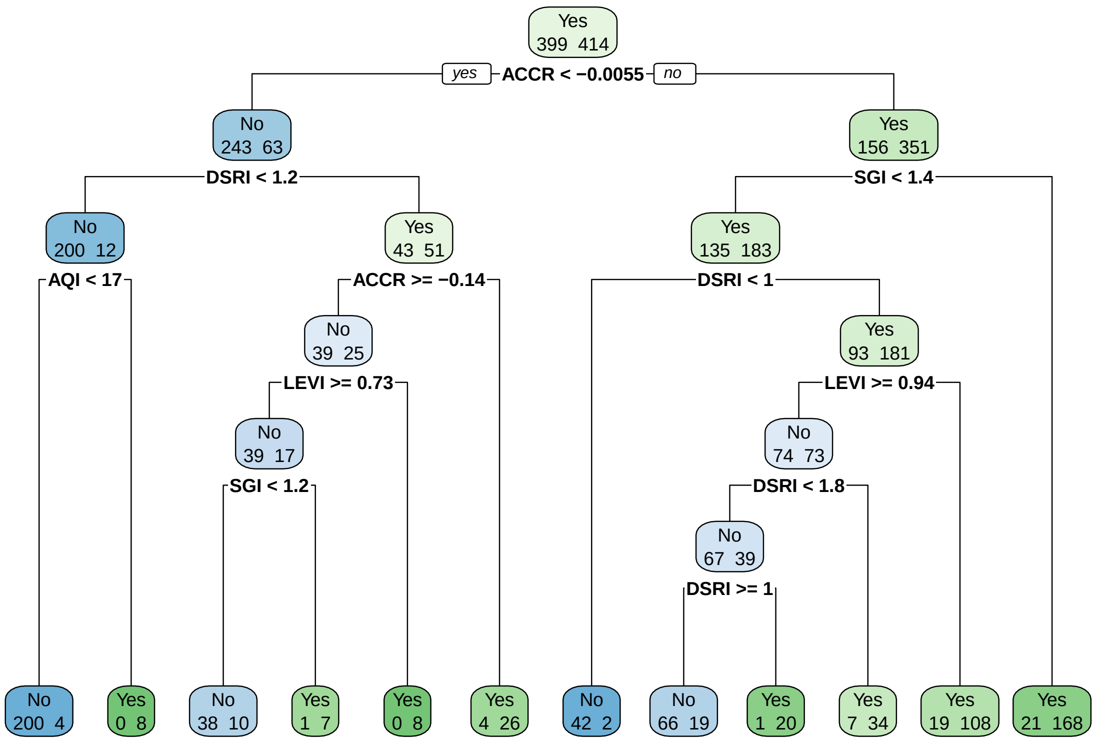
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.default(response = test$Manipulater, predictor = predictroc[,     2], plot = TRUE, legacy.axes =
##
## Data: predictroc[, 2] in 221 controls (test$Manipulater No) < 206 cases (test$Manipulater Yes).
## Area under the curve: 0.8992
```

```r
#Pruning with cp

opt <- which.min(tree$cptable[,"xerror"])
cp <- tree$cptable[opt, "CP"]
tree_prune <- prune(tree, cp = cp)
rpart.plot(tree_prune,extra = 1)
```




```r
predictcart <- predict(tree_prune,test,type = "class")
confusionMatrix(predictcart,test$Manipulater,positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  186  30
##        Yes  35 176
##
##                Accuracy : 0.8478
##                  95% CI : (0.8101, 0.8805)
##     No Information Rate : 0.5176
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6954
##
##  Mcnemar's Test P-Value : 0.6198
```

```
##
##             Sensitivity : 0.8544
##             Specificity : 0.8416
##          Pos Pred Value : 0.8341
##          Neg Pred Value : 0.8611
##              Prevalence : 0.4824
##          Detection Rate : 0.4122
##    Detection Prevalence : 0.4941
##       Balanced Accuracy : 0.8480
##
##        'Positive' Class : Yes
##
```

Quest-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.

```r
pred_manipulators_dataset <- read_excel('predicting_manipulators_dataset.xlsx',
                                        sheet = 'Complete Data')

head(pred_manipulators_dataset)
```

```
## # A tibble: 6 x 11
##   Company ~1  DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI Manip~2 C-MAN~3
##        <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <chr>     <dbl>
## 1          1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16  Yes           1
## 2          2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987 Yes           1
## 3          3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26  Yes           1
## 4          4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681 Yes           1
## 5          5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939 Yes           1
## 6          6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54  Yes           1
## # ... with abbreviated variable names 1: 'Company ID', 2: Manipulater,
## #   3: 'C-MANIPULATOR'
```

```r
dim(pred_manipulators_dataset) # total 1239 rows and 11 columns
```

```
## [1] 1239   11
```

```r
sum(is.na(pred_manipulators_dataset)) # No NULL Values in entire dataset
```

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

```r
table(pred_manipulators_dataset$Manipulater) #No:1200,Yes:39,dataset is imbalanced
```

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

```r
names(pred_manipulators_dataset)[11] <- 'manipulator_target'
names(pred_manipulators_dataset)[1] <- 'company_ID'

pred_manipulators_dataset <- subset(pred_manipulators_dataset,
                                    select = -c(company_ID, Manipulater))

head(pred_manipulators_dataset) # removed unwanted columns from dataset
```

```
## # A tibble: 6 x 9
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipulator_target
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>              <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16                   1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987                  1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26                   1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681                  1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939                  1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54                   1
```

```r
Complete_final <- pred_manipulators_dataset
head(Complete_final)
```

```
## # A tibble: 6 x 9
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI manipulator_target
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl>              <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16                   1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987                  1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26                   1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681                  1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939                  1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54                   1
```

```r
set.seed(1234)
index = sample(2, nrow(Complete_final), replace = TRUE, prob = c(0.65,0.35))
TrainData = Complete_final[index == 1, ]
TestData = Complete_final[index == 2,]
dim(TrainData) # 812   9
```

```
## [1] 812   9
```

```r
#### Over-sampling #############################
Complete_Data_over <- ovun.sample(manipulator_target~.,
                                  data = TrainData,method = "over",
                                  N=1624)$data

table(Complete_Data_over$manipulator_target)
```

```
##
##   0   1
## 783 841
```

```r
# Lets create the logistic regression model

null = glm(manipulator_target~1, data= Complete_Data_over, family = "binomial")
full = glm(manipulator_target~., data = Complete_Data_over, family = 'binomial')

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

```
## Start:  AIC=2251.27
## manipulator_target ~ 1
##
##        Df Deviance    AIC
## + DSRI  1   1956.7 1960.7
## + SGAI  1   2119.9 2123.9
## + SGI   1   2152.1 2156.1
## + ACCR  1   2156.3 2160.3
## + DEPI  1   2219.3 2223.3
## + AQI   1   2229.9 2233.9
## + LEVI  1   2241.6 2245.6
## <none>      2249.3 2251.3
## + GMI   1   2248.9 2252.9
##
## Step:  AIC=1960.67
## manipulator_target ~ DSRI
##
##        Df Deviance    AIC
## + SGI   1   1658.8 1664.8
## + ACCR  1   1689.4 1695.4
## + SGAI  1   1894.5 1900.5
## + DEPI  1   1926.8 1932.8
## + AQI   1   1928.8 1934.8
## + LEVI  1   1945.5 1951.5
## <none>      1956.7 1960.7
## + GMI   1   1956.3 1962.3
##
## Step:  AIC=1664.78
## manipulator_target ~ DSRI + SGI
##
##        Df Deviance    AIC
## + ACCR  1   1401.4 1409.4
## + AQI   1   1535.5 1543.5
## + SGAI  1   1607.3 1615.3
## + LEVI  1   1632.6 1640.6
## + GMI   1   1651.0 1659.0
## <none>      1658.8 1664.8
## + DEPI  1   1658.6 1666.6
##
## Step:  AIC=1409.38
## manipulator_target ~ DSRI + SGI + ACCR
##
##        Df Deviance    AIC
## + AQI   1   1159.1 1169.1
## + SGAI  1   1379.6 1389.6
## + LEVI  1   1393.6 1403.6
```

```
## + DEPI  1   1399.0 1409.0
## <none>      1401.4 1409.4
## + GMI   1   1401.1 1411.1
##
## Step:  AIC=1169.07
## manipulator_target ~ DSRI + SGI + ACCR + AQI
##
##        Df Deviance    AIC
## + GMI   1   1130.8 1142.8
## + LEVI  1   1137.7 1149.7
## + DEPI  1   1154.4 1166.4
## <none>      1159.1 1169.1
## + SGAI  1   1159.1 1171.1
##
## Step:  AIC=1142.76
## manipulator_target ~ DSRI + SGI + ACCR + AQI + GMI
##
##        Df Deviance    AIC
## + LEVI  1   1109.9 1123.9
## + DEPI  1   1126.5 1140.5
## <none>      1130.8 1142.8
## + SGAI  1   1130.4 1144.4
##
## Step:  AIC=1123.89
## manipulator_target ~ DSRI + SGI + ACCR + AQI + GMI + LEVI
##
##        Df Deviance    AIC
## + DEPI  1   1102.8 1118.8
## <none>      1109.9 1123.9
## + SGAI  1   1109.1 1125.1
##
## Step:  AIC=1118.79
## manipulator_target ~ DSRI + SGI + ACCR + AQI + GMI + LEVI + DEPI
##
##        Df Deviance    AIC
## <none>      1102.8 1118.8
## + SGAI  1   1102.6 1120.6

##
## Call:  glm(formula = manipulator_target ~ DSRI + SGI + ACCR + AQI +
##     GMI + LEVI + DEPI, family = "binomial", data = Complete_Data_over)
##
## Coefficients:
## (Intercept)         DSRI          SGI         ACCR          AQI          GMI
##     -9.3758       2.5401       3.4022       8.7983       0.6607       0.9313
##        LEVI         DEPI
##     -0.7897       0.5274
##
## Degrees of Freedom: 1623 Total (i.e. Null);  1616 Residual
## Null Deviance:       2249
## Residual Deviance: 1103  AIC: 1119
```

```r
step(null, scope=list(lower=null, upper=full), direction="backward")
```

```
## Start:  AIC=2251.27
## manipulator_target ~ 1
```

```
##
## Call:  glm(formula = manipulator_target ~ 1, family = "binomial", data = Complete_Data_over)
##
## Coefficients:
## (Intercept)
##     0.07146
##
## Degrees of Freedom: 1623 Total (i.e. Null);  1623 Residual
## Null Deviance:       2249
## Residual Deviance: 2249  AIC: 2251
```

```r
step(full,scope =list(lower=null,upper=full),direction ="both")
```

```
## Start:  AIC=1120.58
## manipulator_target ~ DSRI + GMI + AQI + SGI + DEPI + SGAI + ACCR +
##     LEVI
##
##        Df Deviance    AIC
## - SGAI  1   1102.8 1118.8
## <none>      1102.6 1120.6
## - DEPI  1   1109.1 1125.1
## - LEVI  1   1125.5 1141.5
## - GMI   1   1129.5 1145.5
## - AQI   1   1375.4 1391.4
## - ACCR  1   1463.8 1479.8
## - SGI   1   1521.8 1537.8
## - DSRI  1   1656.9 1672.9
##
## Step:  AIC=1118.79
## manipulator_target ~ DSRI + GMI + AQI + SGI + DEPI + ACCR + LEVI
##
##        Df Deviance    AIC
## <none>      1102.8 1118.8
## + SGAI  1   1102.6 1120.6
## - DEPI  1   1109.9 1123.9
## - LEVI  1   1126.5 1140.5
## - GMI   1   1130.8 1144.8
## - AQI   1   1391.7 1405.7
## - ACCR  1   1476.5 1490.5
## - SGI   1   1521.9 1535.9
## - DSRI  1   2003.8 2017.8
```

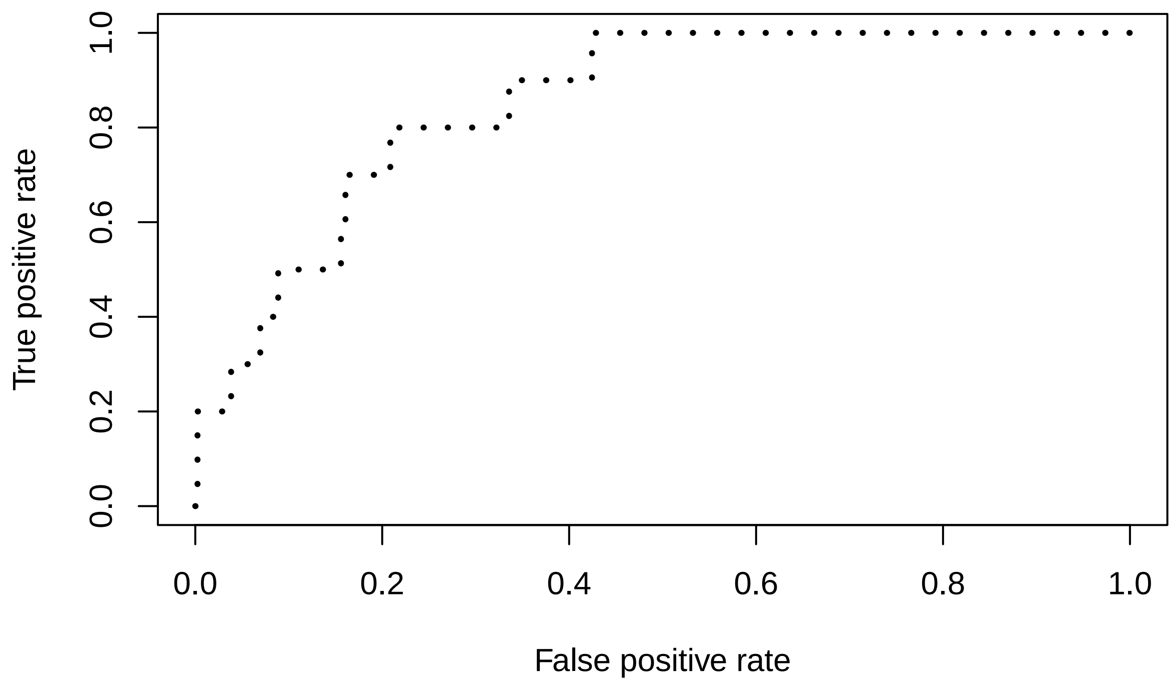
```
##
## Call:  glm(formula = manipulator_target ~ DSRI + GMI + AQI + SGI + DEPI +
##     ACCR + LEVI, family = "binomial", data = Complete_Data_over)
##
## Coefficients:
```

```
## (Intercept)        DSRI         GMI         AQI         SGI        DEPI
##     -9.3758      2.5401      0.9313      0.6607      3.4022      0.5274
##        ACCR        LEVI
##      8.7983     -0.7897
##
## Degrees of Freedom: 1623 Total (i.e. Null);  1616 Residual
## Null Deviance:       2249
## Residual Deviance: 1103  AIC: 1119
```

```
log_reg_complete_over<- glm(manipulator_target ~ DSRI + ACCR + SGI + AQI, data= Complete_Data_over,
                       family = "binomial")

summary(log_reg_complete_over)
```

```
##
## Call:
## glm(formula = manipulator_target ~ DSRI + ACCR + SGI + AQI, family = "binomial",
##     data = Complete_Data_over)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5227  -0.5184   0.0000   0.5118   1.7568
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.70874    0.49435  -17.62   <2e-16 ***
## DSRI         2.89939    0.17475   16.59   <2e-16 ***
## ACCR         8.26723    0.55895   14.79   <2e-16 ***
## SGI          3.25342    0.23656   13.75   <2e-16 ***
## AQI          0.56360    0.05208   10.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2249.3  on 1623  degrees of freedom
## Residual deviance: 1159.1  on 1619  degrees of freedom
## AIC: 1169.1
##
## Number of Fisher Scoring iterations: 9
```

```
# Accuracy of Log regression on Train-data with oversampling condition

# Null deviance: 2249.3  on 1623  degrees of freedom
# Residual deviance: 1159.1  on 1619  degrees of freedom
# AIC: 1169.1

pred1 = predict(log_reg_complete_over, newdata = TestData, type="response")

# Calculating the ROC values
pred_ROC1 = prediction(pred1,TestData$manipulator_target)
perf1 = performance(pred_ROC1,"tpr","fpr")
plot(perf1, col = "black", lty = 3, lwd = 3)
```




```r
mydistance <- function(x,y,p){
  d=(x-0)^2+(y-1)^2 # given the points (x, y), compute the distance to the corner point (0,1)
  ind <- which(d==min(d)) # Find the minimum distance and its index
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
}

opt.cut <- function(perf){
  cut.ind <- mapply(FUN = mydistance,
                    perf@x.values, perf@y.values,perf@alpha.values)
}

Output<- opt.cut(perf1)
Threshold <- Output[,1]["cutoff"] # 0.3231143


pred_roc_cut_point <- ifelse(pred1>Threshold,1,0)
tab<-table(pred_roc_cut_point, TestData$manipulator_target,
           dnn = c("Predicted","Actual"))
confusionMatrix(tab,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##            Actual
## Predicted    0   1
##          0 330   3
```

```
##         1  87   7
##
##                Accuracy : 0.7892
##                  95% CI : (0.7474, 0.827)
##     No Information Rate : 0.9766
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0964
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.70000
##             Specificity : 0.79137
##          Pos Pred Value : 0.07447
##          Neg Pred Value : 0.99099
##              Prevalence : 0.02342
##          Detection Rate : 0.01639
##    Detection Prevalence : 0.22014
##       Balanced Accuracy : 0.74568
##
##        'Positive' Class : 1
##
```

```
# Accuracy on Test-data with Oversampling condition applied

# Accuracy : 0.7892
# Sensitivity : 0.70000
# Specificity : 0.79137



########## SMOTE CONDITION #####################################################

pred_manipulators_dataset <- read_excel('predicting_manipulators_dataset.xlsx',
                                        sheet = 'Complete Data')

head(pred_manipulators_dataset)
```

```
## # A tibble: 6 x 11
##   Company ~1  DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI Manip~2 C-MAN~3
##        <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <chr>     <dbl>
## 1          1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16  Yes           1
## 2          2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987 Yes           1
## 3          3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26  Yes           1
## 4          4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681 Yes           1
## 5          5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939 Yes           1
## 6          6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54  Yes           1
## # ... with abbreviated variable names 1: `Company ID`, 2: Manipulater,
## #   3: `C-MANIPULATOR`
```

```
dim(pred_manipulators_dataset) # total 1239 rows and 11 columns
```

```
## [1] 1239   11
```

```r
sum(is.na(pred_manipulators_dataset)) # No NULL Values in entire dataset
```

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

```r
table(pred_manipulators_dataset$Manipulater) #No:1200,Yes:39,dataset is imbalanced
```

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

```r
names(pred_manipulators_dataset)[11] <- 'manipulator_target'
names(pred_manipulators_dataset)[1] <- 'company_ID'

pred_manipulators_dataset <- subset(pred_manipulators_dataset,
                                    select = -c(company_ID))
table(pred_manipulators_dataset$Manipulater)
```

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

```r
head(pred_manipulators_dataset) # removed unwanted columns from dataset
```

```
## # A tibble: 6 x 10
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI Manipulater manipulator_~1
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <chr>                <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16  Yes                      1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987 Yes                      1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26  Yes                      1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681 Yes                      1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939 Yes                      1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54  Yes                      1
## # ... with abbreviated variable name 1: manipulator_target
```

```r
Complete_final <- pred_manipulators_dataset
head(Complete_final)
```

```
## # A tibble: 6 x 10
##    DSRI   GMI   AQI    SGI  DEPI   SGAI    ACCR  LEVI Manipulater manipulator_~1
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <chr>                <dbl>
## 1 1.62   1.13 7.19   0.366  1.38 1.62   -0.167  1.16  Yes                      1
## 2 1      1.61 1.00  13.1    0.4  5.20    0.0605 0.987 Yes                      1
## 3 1      1.02 1.24   1.48   1.17 0.648   0.0367 1.26  Yes                      1
## 4 1.49   1    0.466  0.673  2    0.0929  0.273  0.681 Yes                      1
## 5 1      1.37 0.637  0.861  1.45 1.74    0.123  0.939 Yes                      1
## 6 0.906  1.36 0.784  1.79   1.28 0.505   0.0546 1.54  Yes                      1
## # ... with abbreviated variable name 1: manipulator_target
```

```r
set.seed(1234)
index = sample(2, nrow(Complete_final), replace = TRUE, prob = c(0.65,0.35))
TrainData = Complete_final[index == 1, ]
TestData = Complete_final[index == 2,]
dim(TrainData) # 812   9
```

```
## [1] 812  10
```

```r
smote_complete<-SmoteClassif(Manipulater~.,as.data.frame(train), "balance")
sum(is.na(smote_complete))
```

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

```r
null = glm(Manipulater~1, data= smote_complete, family = "binomial")
full = glm(Manipulater~., data= smote_complete, family = "binomial")

step(null, scope=list(lower=null, upper=full), direction="both")
```

```
## Start:  AIC=1127.67
## Manipulater ~ 1
##
##        Df Deviance    AIC
## + ACCR  1   1027.5 1031.5
## + DSRI  1   1051.9 1055.9
## + SGI   1   1067.5 1071.5
## + SGAI  1   1084.9 1088.9
## + AQI   1   1101.2 1105.2
## + GMI   1   1113.7 1117.7
## + DEPI  1   1116.8 1120.8
## + LEVI  1   1120.7 1124.7
## <none>      1125.7 1127.7
##
## Step:  AIC=1031.49
## Manipulater ~ ACCR
##
##        Df Deviance     AIC
## + DSRI  1   917.31  923.31
## + SGAI  1   956.82  962.82
## + AQI   1   974.42  980.42
## + SGI   1   977.56  983.56
## + DEPI  1  1009.19 1015.19
## + GMI   1  1010.65 1016.65
## + LEVI  1  1016.49 1022.49
## <none>     1027.49 1031.49
## - ACCR  1  1125.67 1127.67
##
## Step:  AIC=923.31
## Manipulater ~ ACCR + DSRI
##
##        Df Deviance     AIC
## + AQI   1   838.91  846.91
## + SGI   1   842.30  850.30
```

```
## + GMI   1   886.80  894.80
## + DEPI  1   908.87  916.87
## + SGAI  1   909.71  917.71
## + LEVI  1   914.41  922.41
## <none>      917.31  923.31
## - DSRI  1  1027.49 1031.49
## - ACCR  1  1051.88 1055.88
##
## Step:  AIC=846.91
## Manipulater ~ ACCR + DSRI + AQI
##
##        Df Deviance     AIC
## + SGI   1   702.61  712.61
## + GMI   1   781.17  791.17
## + DEPI  1   835.21  845.21
## <none>      838.91  846.91
## + SGAI  1   838.14  848.14
## + LEVI  1   838.88  848.88
## - AQI   1   917.31  923.31
## - DSRI  1   974.42  980.42
## - ACCR  1  1019.63 1025.63
##
## Step:  AIC=712.61
## Manipulater ~ ACCR + DSRI + AQI + SGI
##
##        Df Deviance    AIC
## + GMI   1   660.50 672.50
## + LEVI  1   690.19 702.19
## + DEPI  1   695.70 707.70
## <none>      702.61 712.61
## + SGAI  1   701.89 713.89
## - SGI   1   838.91 846.91
## - AQI   1   842.30 850.30
## - ACCR  1   864.15 872.15
## - DSRI  1   899.07 907.07
##
## Step:  AIC=672.5
## Manipulater ~ ACCR + DSRI + AQI + SGI + GMI
##
##        Df Deviance    AIC
## + LEVI  1   651.10 665.10
## + SGAI  1   655.47 669.47
## + DEPI  1   657.93 671.93
## <none>      660.50 672.50
## - GMI   1   702.61 712.61
## - SGI   1   781.17 791.17
## - AQI   1   825.07 835.07
## - ACCR  1   851.79 861.79
## - DSRI  1   878.83 888.83
##
## Step:  AIC=665.1
## Manipulater ~ ACCR + DSRI + AQI + SGI + GMI + LEVI
##
##        Df Deviance    AIC
```

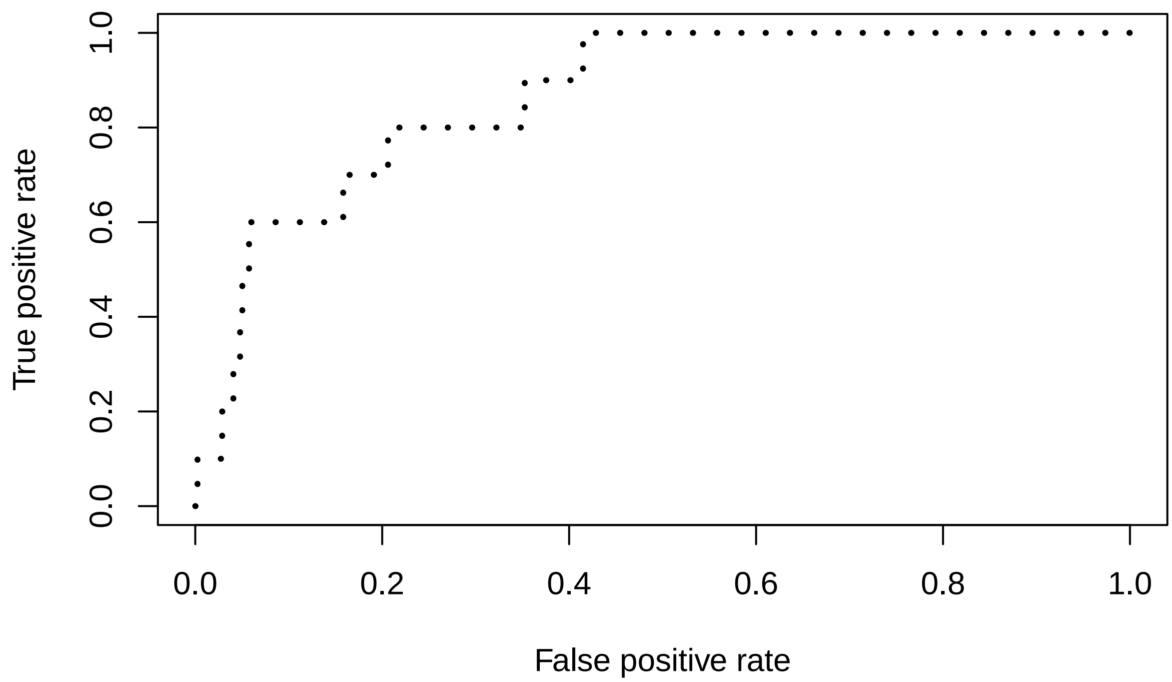
```
## + DEPI  1   647.25 663.25
## + SGAI  1   648.27 664.27
## <none>      651.10 665.10
## - LEVI  1   660.50 672.50
## - GMI   1   690.19 702.19
## - SGI   1   781.16 793.16
## - ACCR  1   819.31 831.31
## - AQI   1   824.36 836.36
## - DSRI  1   871.96 883.96
##
## Step:  AIC=663.25
## Manipulater ~ ACCR + DSRI + AQI + SGI + GMI + LEVI + DEPI
##
##        Df Deviance    AIC
## + SGAI  1   644.26 662.26
## <none>      647.25 663.25
## - DEPI  1   651.10 665.10
## - LEVI  1   657.93 671.93
## - GMI   1   681.25 695.25
## - SGI   1   774.42 788.42
## - ACCR  1   811.87 825.87
## - AQI   1   822.39 836.39
## - DSRI  1   870.48 884.48
##
## Step:  AIC=662.26
## Manipulater ~ ACCR + DSRI + AQI + SGI + GMI + LEVI + DEPI + SGAI
##
##        Df Deviance    AIC
## <none>      644.26 662.26
## - SGAI  1   647.25 663.25
## - DEPI  1   648.27 664.27
## - LEVI  1   652.62 668.62
## - GMI   1   681.03 697.03
## - DSRI  1   749.41 765.41
## - SGI   1   772.86 788.86
## - AQI   1   796.47 812.47
## - ACCR  1   802.10 818.10
```

```
##
## Call:  glm(formula = Manipulater ~ ACCR + DSRI + AQI + SGI + GMI + LEVI +
##     DEPI + SGAI, family = "binomial", data = smote_complete)
##
## Coefficients:
## (Intercept)         ACCR         DSRI          AQI          SGI          GMI
##     -9.6328       9.8243       1.6296       0.5969       3.0597       1.6506
##        LEVI         DEPI         SGAI
##     -0.5757       1.0488       0.4298
##
## Degrees of Freedom: 811 Total (i.e. Null);  803 Residual
## Null Deviance:       1126
## Residual Deviance: 644.3     AIC: 662.3
```

```r
log_reg<- glm(Manipulater ~ ACCR + DSRI + SGI + AQI + LEVI,
                   data= smote_complete, family = "binomial")
summary(log_reg)
```

```
##
## Call:
## glm(formula = Manipulater ~ ACCR + DSRI + SGI + AQI + LEVI, family = "binomial",
##     data = smote_complete)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8561  -0.6517  -0.0009   0.6614   1.8870
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.22116    0.57820 -10.759  < 2e-16 ***
## ACCR         8.23140    0.86780   9.485  < 2e-16 ***
## DSRI         1.91666    0.22125   8.663  < 2e-16 ***
## SGI          2.94881    0.33998   8.673  < 2e-16 ***
## AQI          0.51812    0.07151   7.245 4.31e-13 ***
## LEVI        -0.74473    0.16642  -4.475 7.64e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1125.67  on 811  degrees of freedom
## Residual deviance:  690.19  on 806  degrees of freedom
## AIC: 702.19
##
## Number of Fisher Scoring iterations: 8
```

```r
# Accuracy on train-data with SMOTE condition applied

# Null deviance: 1125.7  on 811  degrees of freedom
# Residual deviance:  599.9  on 806  degrees of freedom
# AIC: 611.9





# Calculating the values for ROC curve
pred1 = predict(log_reg, newdata = TestData, type="response")

# Calculating the ROC values
pred_ROC1 = prediction(pred1,TestData$manipulator_target)
perf1 = performance(pred_ROC1,"tpr","fpr")
plot(perf1, col = "black", lty = 3, lwd = 3)
```




```
mydistance <- function(x,y,p){
  d=(x-0)^2+(y-1)^2 # given the points (x, y), compute the distance to the corner point (0,1)
  ind <- which(d==min(d)) # Find the minimum distance and its index
  c(recall = y[[ind]], specificity = 1-x[[ind]],cutoff = p[[ind]])
}

opt.cut <- function(perf){
  cut.ind <- mapply(FUN = mydistance,
                    perf@x.values, perf@y.values,perf@alpha.values)
}

Output<- opt.cut(perf1)
Threshold <- Output[,1]["cutoff"] # 0.3289778


pred_roc_cut_point <- ifelse(pred1>Threshold,1,0)
tab<-table(pred_roc_cut_point, TestData$manipulator_target,
           dnn = c("Predicted","Actual"))
confusionMatrix(tab,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##          Actual
## Predicted   0   1
##         0 331   3
```

```
##         1  86   7
##
##                Accuracy : 0.7916
##                  95% CI : (0.7499, 0.8291)
##     No Information Rate : 0.9766
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0978
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.70000
##             Specificity : 0.79376
##          Pos Pred Value : 0.07527
##          Neg Pred Value : 0.99102
##              Prevalence : 0.02342
##          Detection Rate : 0.01639
##    Detection Prevalence : 0.21780
##       Balanced Accuracy : 0.74688
##
##        'Positive' Class : 1
##
```

```
# Accuracy on Test-data with SMOTE condition applied

# Accuracy : 0.7799
# Sensitivity : 0.70000
# Specificity :  0.78177


####### Final Observation on Log regression model on complete dataset ###################

################################################################################
# Balanced Data-set Oversampling- Log regression model performance evaluation on Train data
################################################################################

# Null deviance: 2249.3  on 1623  degrees of freedom
# Residual deviance: 1159.1  on 1619  degrees of freedom
# AIC: 1169.1


################################################################################
# Balanced Data-set Oversampling- Log regression model performance evaluation on
# Test-data
################################################################################

# Accuracy : 0.7892
# Sensitivity : 0.70000
# Specificity : 0.79137

################################################################################
# Balanced Data-set SMOTE- Log regression model performance evaluation on Train data
################################################################################
```

```
# Null deviance: 1125.7  on 811  degrees of freedom
# Residual deviance:  599.9  on 806  degrees of freedom
# AIC: 611.9


################################################################################
# Balanced Data-set SMOTE- Log regression model performance evaluation on
# Test-data
################################################################################

# Accuracy : 0.7799
# Sensitivity : 0.70000
# Specificity :  0.78177
```