IDS-572: DATA MINING

CASE STUDY #1

Predicting Earnings Manipulation by Indian firms using Machine Learning Algorithms

University of Illinois @ Chicago

(FALL 2017)

Project By :-

- Abhinav Gupta (665078643)
- Prashansa Nande (655111131)

PROBLEM 1

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

Solution

- After the primary analysis of the given dataset, we found that–

Number of observations for Manipulatros	*39*
Number of observations for Non-Manipulatros	*1200*

Here, number of observations belonging to 'Manipulators' (i.e., 39) is significantly less than those belonging to 'Non-Manipulators' (i.e., 1200). Number of 'Manipulators' are about 3.14 % of the total data collected, thus making the resultant dataset highly unbalanced.

- Beneish Model, also known as 'M-Score Model' is a very important model used in Financial Analytics to find out the scope of 'Earning Manipulation'.

  According to the Beneish model, M-score is the parameter that is used to categorise Manipulators / Non-Manipulators. Originally, the M-score was calculated using the following formula -

  M = -4.84 + 0.92 DSRI + 0.528 GMI + 0.404 AQI + 0.892 SGI + 0.115 DEPI – 0.172 SGAI + 4.679 TATA – 0.327 LVGI

  Beneish model categories (based on M-score) the scope of Earning Manipulation as follows –

M-Score > -2.2	High probability of earning manipulation(Manipulators)
Else	Non - Manipulators

We calculated M-score for the complete data-

```
>library(readxl)
>Complete_Data <- read_excel("~/UIC/Courses/DataMining/Assignments/Case Study 1/Dataset/Complete_Data.xlsx")
>View(Complete_Data)

>my_data <- Complete_Data
>my_data$Mscore<-(-4.84+(0.92*my_data$DSRI)+(0.528* y_data$GMI) +(0.404* my_data$AQI)+(0.892*my_data$SGI) + (0.115* my_data$DEPI)
-(0.172*my_data$SGAI)+(4.679*my_data$ACCR)-0.327*my_data$LEVI))
>my_data$Beneish_prediction <- NA
>my_data$Beneish_prediction[my_data$Mscore > -2.2] <- 1
>my_data$Beneish_prediction[my_data$Mscore <= -2.2] <- 0
>sum(is.na(my_data$Beneish_prediction))
```

```
>tab <- table(my_data$Beneish_prediction, my_data$`C-MANIPULATOR`,
        dnn = c("Predicted", "Actual"))
>confusionMatrix(tab, positive = "1" )
```

```
> confusionMatrix(tab, positive = "1" )
Confusion Matrix and Statistics

         Actual
Predicted    0    1
        0 1032   17
        1  168   22

               Accuracy : 0.8507
                 95% CI : (0.8296, 0.8701)
    No Information Rate : 0.9685
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1476
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.56410
            Specificity : 0.86000
         Pos Pred Value : 0.11579
         Neg Pred Value : 0.98379
             Prevalence : 0.03148
         Detection Rate : 0.01776
   Detection Prevalence : 0.15335
      Balanced Accuracy : 0.71205

       'Positive' Class : 1
```

From the above output of Confusion Matrix, we can see that the accuracy of the model is approximately 85%, which is a good performance. However, in this case the data is unbalanced, and the class of interest (i.e., class = "1") is a minority class, we have to consider the performance of the model to correctly predict the class of interest. Considering the class of interest as positive class, we need to focus on the sensitivity value of the model. And the sensitivity for the Beneish model is around 56%. Thus, we can say that Beneish model is not relevant to the indian data under consideration (as it does not take the the issue of unbalanced data into consideration).

In such scenario, where we come across **class-imbalance problem,** we can use machine learning algorithms like – Classification Trees, Logistic Regression. This is because, such machine learning algorithms give better result(accuracy) than the Beneish Model for such unbalanced data. This is also proven by the chief data scientist of MCA Technology Solutions, Saurabh Rishi – as mentioned in the given document.

PROBLEM 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?

<u>Solution</u>

Such scenario where the number of observations belonging to one class is significantly lower than those belonging to the other classes is known as Class-imbalance problem.

In such scenario, one struggles to get a well-performing model. We may face below mentioned issues-

- In such cases, models developed using traditional statistical algorithms (like Logistics Regression and Decision Tree) does not perform well, as they could be biased, because they don't consider the proportion of classes in the data.
- Also, the conventional model evaluation methods do not accurately measure model performance when faced with imbalanced datasets.
- The standard classifier algorithms only concentrate on reducing errors and not on the structure of the data.
- They focus and tend to predict only the majority class data and ignore the minority class data by treating them as noise.
- If the event to be predicted belongs to the minority class and the event rate is less than 5%, it is usually referred to as a rare event. Thus, there is a high probability of misclassification of the minority class as compared to the majority class. Hence, it reduces the prediction accuracy of the model.

We can adopt various techniques to deal with such issue of class-imbalance problem-

1. *<u>Data Level approach: Resampling Techniques</u>*

1.1. Random Under-Sampling
- It balances the data by randomly eliminating majority class examples.
- It can improve the run time when dataset is huge.
- It can remove data points which may be useful information.

1.2. Random Over-Sampling
- This increases the number of instances of minority class by replicating them randomly.
- Better than under sampling. There is no information loss.

1.3. Cluster-Based Over Sampling

- Here, K-means clustering algorithm is applied to both the class of the target variable.
- Each cluster is oversampled such that all clusters of the same class have an equal number of instances and all classes have the same size

1.4. Informed Over Sampling: Synthetic Minority Over-Sampling Technique

- A subset of data is taken from the minority class as an example and then new synthetic similar instances are created.
- Reduces over-fitting problem.

1.5. Modified synthetic minority oversampling technique (MSMOTE)

- Modified version of SMOTE.
- This algorithm classifies the samples of minority classes into 3 distinct groups – Security/Safe samples, Border samples, and latent nose samples.
- The algorithm randomly selects a data point from the k nearest neighbors for the security sample, selects the nearest neighbor from the border samples and does nothing for latent noise.

2. <u>Algorithmic Ensemble Techniques</u>

2.1. Bagging Based

- Generates different training samples (with replacement), trains each sample using the bootstrapped algorithm and aggregates the result at the end.
- Reduces over-fitting.
- Reduces variance

2.2. Boosting-Based

2.2.1. Adaptive Boosting- Ada Boost

- Adaboost either requires the users to specify a set of weak learners or randomly generates the weak learners before the actual learning process.
- The weight of each learner is adjusted at every step depending on whether it predicts a sample correctly.

2.2.2. Gradient Tree Boosting

- Here, each models are trained sequentially.
- Each model minimizes the loss function.

2.2.3. XG Boost

- XGBoost (Extreme Gradient Boosting) is an advanced implementation of Gradient Boosting.
- 10 times faster than the normal Gradient Boosting.
- XG Boost splits up to the maximum depth specified and prunes the tree backward
-

PROBLEM 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.

Solution

#Read Data
```
>library(readxl)
>Sample_Data <- read_excel("~/UIC/Courses/DataMining/Assignments/Case Study 1/Dataset/Sample_Data.xlsx")
```

#View and primary analysis of the imported dataset
```
>View(Sample_Data)
>dim(Sample_Data) #220 11
>str(Sample_Data)
>summary(Sample_Data)
```

#Converting target variable to factor type
```
>sample_final <- Sample_Data
>sample_final$`C-MANIPULATOR`<-as.factor(sample_final$`C-MANIPULATOR`)
>class(sample_final$`C-MANIPULATOR`)
```

#Removing unwanted variables
```
>sample_final$`Company ID` <- NULL
>sample_final$Manipulator <- NULL
```

Changing the target variable name to a proper format
```
>colnames(sample_final)[9] <- "C_MANIPULATOR"
```

#Checking the count of classes of target variable
```
>tab <- table(sample_final$C_MANIPULATOR)
>tab
# 0 1
# 181 39
```
As we can see from the above result(count) that the number of observation for the event class is very less as compared to the other class of the target variable. This indicates that the dataset is unbalanced.

#<u>Classification before Data - balancing</u>

Sampling the sample dataset - Partition the sample data into training & Test data
```
>set.seed(1234)
>index <- sample(2, nrow(sample_final), replace = TRUE, prob = c(0.65,0.35))
>sample_train <- sample_final[index == 1,]
>sample_test <- sample_final[index == 2,]

>tab <- table(sample_train$C_MANIPULATOR)
>tab
```

Model : Logistic Regression

Variable Selection

```
>null = glm(C_MANIPULATOR~1, data = sample_train, family = binomial)
>full = glm(C_MANIPULATOR~., data = sample_train, family = binomial)
```

#Forward Selection

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

After running either or both (forward & backward) variable selection method, we can see from the output that the important variables are 'DSRI + SGI + ACCR + AQI + GMI'. Hence, we will run our model using only these important input variables

#Runnig Logistic Regression model

```
>lg_model_impVar <- glm(C_MANIPULATOR~DSRI + SGI + ACCR + AQI, data= sample_train, family =
"binomial")
>summary(lg_model_impVar)
```

```
> summary(lg_model_impVar)

Call:
glm(formula = C_MANIPULATOR ~ DSRI + SGI + ACCR + AQI, family = "binomial",
    data = sample_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8718  -0.4418  -0.3138  -0.1932   3.2342

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -6.6137     1.2266  -5.392 6.96e-08 ***
DSRI          0.8385     0.2447   3.426 0.000612 ***
SGI           2.5762     0.7357   3.502 0.000462 ***
ACCR          7.5036     1.9625   3.824 0.000132 ***
AQI           0.4309     0.1469   2.932 0.003364 **
---
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:  89.691  on 148  degrees of freedom
AIC: 99.691

Number of Fisher Scoring iterations: 7
```

PROBLEM 4

Comment on the model developed; how do you measure the accuracy of the model?

Solution

Output of the above model-

```
# Null deviance: 148.581 on 152 degrees of freedom
# Residual deviance:  89.691 on 148 degrees of freedom
# AIC: 99.691
```

```
> lg_model_impVar$null.deviance-lg_model_impVar$deviance
[1] 58.89034
```

COMMENTS

- AIC of the model is – 99.691
- Important variables for predicting the target variable are - DSRI + SGI + ACCR + AQI
- Null deviance: 148.581 on 152 degrees of freedom
- Residual deviance: 89.691 on 148 degrees of freedom
- Difference between Null deviance and Residual Deviance – 58.89034

To measure the accuracy and performance of the model, we will perform following steps-

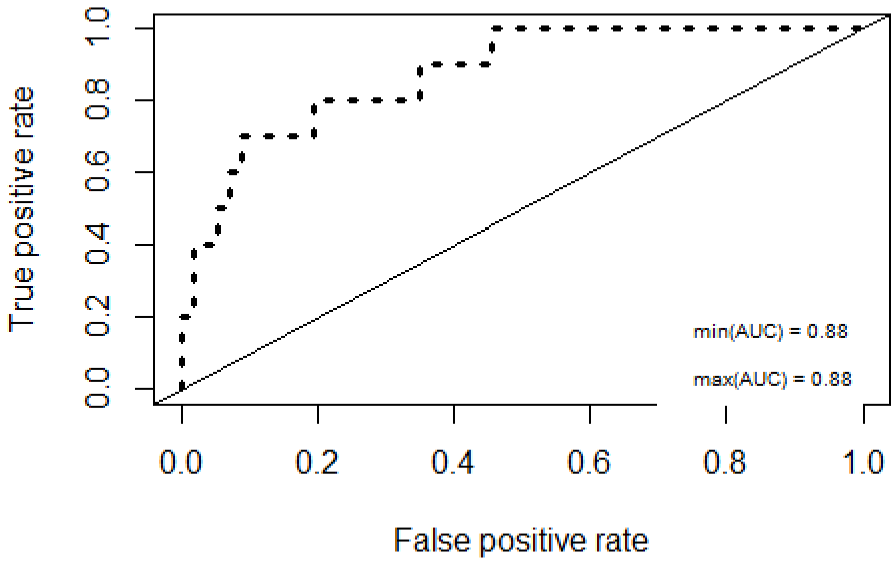
1. Predict the target variable using predict.glm() function.

   ```
   >predict_test_lr1 <- predict.glm(lg_model_impVar, sample_test, type = "response")
   ```

2. Plot ROC curve to get the cut-off point (Point on the ROC curve which has the least distant from the “(0,1)” point (i.e., fpr=0 and tpr=1) of the plot.

   ```
   # ROC curve
   >pred_roc= prediction(predict_test_lr1, sample_test$C_MANIPULATOR)
   perf_roc = performance(pred_roc,"tpr","fpr")

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

   # Calculating AUC
   >auc = performance(pred_roc, "auc")

   # Now converting S4 class to a vector
   >auc = unlist(slot(auc, "y.values"))

   # Adding min and max ROC AUC to the center of the plot
   >minauc = min(round(auc, digits = 2))
   >maxauc = max(round(auc, digits = 2))
   >minauct = paste(c("min(AUC) = "), minauc, sep = "")
   ```

```
>maxauct = paste(c("max(AUC) = "), maxauc, sep = "")
>legend(0.7, 0.3, c(minauct, maxauct, "\n"),
    border = "pink", cex = 0.6, box.col = "white")
>abline(a= 0, b=1)
```




```
#Getting an optimal cut point
>opt.cut = function(perf_roc, pred_roc){
 cut.ind = mapply(FUN=function(x, y, p){
   d = (x - 0)^2 + (y-1)^2
   ind = which(d == min(d))
   c(sensitivity = y[[ind]], specificity = 1-x[[ind]], cutoff = p[[ind]])
 }, perf_roc@x.values, perf_roc@y.values, pred_roc@cutoffs)}
>print(opt.cut(perf_roc, pred_roc))
#sensitivity 0.8000000
#specificity 0.8070175
#cutoff      0.1442311
```

3. Plot Confusion Matrix, using the cut-off point that we got in the previous step, to see the accuracy(performance) of the model

```
>predict_test_lr1 <- ifelse(predict_test_lr1>0.1442311,1,0)
>ptab<-table(predict_test_lr1, sample_test$C_MANIPULATOR, dnn = c("Predicted","Actual"))
>confusionMatrix(ptab,positive = "1")
```

```
> confusionMatrix(ptab,positive = "1")
Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 46  2
        1 11  8

               Accuracy : 0.806
                 95% CI : (0.6911, 0.8924)
    No Information Rate : 0.8507
    P-Value [Acc > NIR] : 0.8824

                  Kappa : 0.4427
 Mcnemar's Test P-Value : 0.0265

            Sensitivity : 0.8000
            Specificity : 0.8070
         Pos Pred Value : 0.4211
         Neg Pred Value : 0.9583
             Prevalence : 0.1493
         Detection Rate : 0.1194
   Detection Prevalence : 0.2836
      Balanced Accuracy : 0.8035

       'Positive' Class : 1
```

OBSERVATIONS

- Accuracy : 0.806
- Sensitivity : 0.8000

PROBLEM 5

What should be the strategy adopted by MCA Technology Solutions to deploy the logistic regression model developed?

Solution

To deploy the model, MCA technology should first be confirmed if they should move ahead with their current consideration of unbalanced data or should they balance the data and then create a model to be deployed.

To help for this decision, we decided to compare our above obtained result with a model developed over a balanced data. Hence, we adopted few approaches to balance the data.

To balance the given data sample (220 cases), we adopted following 2 approaches –

1. Over-sampling method
2. SMOTE

Classification after Data - balancing

Balancing the data - using 'Oversample' technique

```
>install.packages("ROSE")
>library(ROSE)

>over_sample <- ovun.sample(C_MANIPULATOR~., data = sample_train,
                  method = "over", N= 248)$data
>table(over_sample$C_MANIPULATOR)
0   1
124 124
```

Logistic Regression

Variable Selection

```
>null = glm(C_MANIPULATOR~1, data= over_sample, family = "binomial") # Includes only the intercept
>full = glm(C_MANIPULATOR~., data= over_sample, family = "binomial")
#Forward Selection
>step(null, scope=list(lower=null, upper=full), direction="forward")

>lr_model_bal<- glm(C_MANIPULATOR ~ DSRI + SGI + AQI + ACCR + LEVI + GMI, data= over_sample,
family = "binomial")
>summary(lr_model_bal)
```

```
> summary(lr_model_bal)

Call:
glm(formula = C_MANIPULATOR ~ DSRI + SGI + AQI + ACCR + LEVI +
    GMI, family = "binomial", data = over_sample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3339  -0.5981  -0.0310   0.6480   1.6544

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -7.8273     1.1318  -6.916 4.65e-12 ***
DSRI          1.7765     0.3511   5.060 4.18e-07 ***
SGI           3.5773     0.6173   5.795 6.83e-09 ***
AQI           0.7189     0.1346   5.341 9.22e-08 ***
ACCR          6.7137     1.2943   5.187 2.13e-07 ***
LEVI         -1.0741     0.3516  -3.055  0.00225 **
GMI           0.9967     0.3827   2.604  0.00920 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 343.80  on 247  degrees of freedom
Residual deviance: 200.85  on 241  degrees of freedom
AIC: 214.85

Number of Fisher Scoring iterations: 8
```

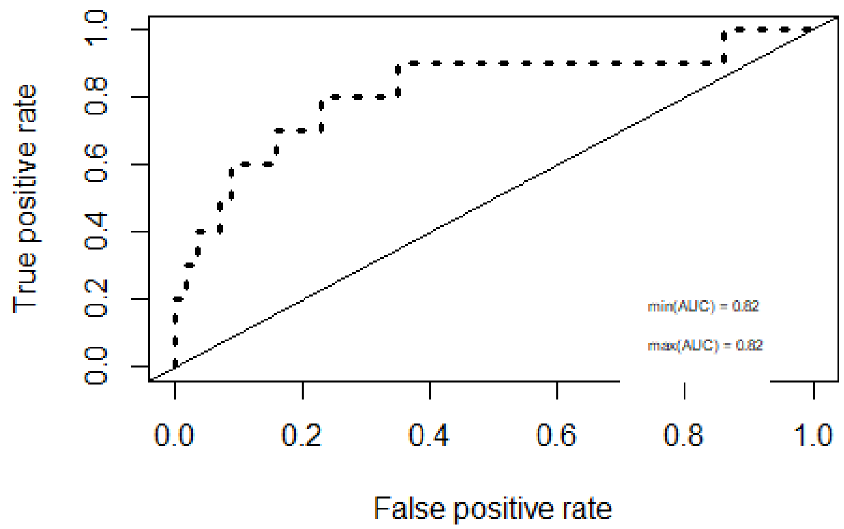
```
>lr_model_bal$null.deviance-lr_model_bal$deviance
[1] 142.9477


>pred_test_bal = predict.glm(lr_model_bal, newdata = sample_test, type="response")
```

Calculating the values for ROC curve

```
>pred_ROC_bal = prediction(pred_test_bal,sample_test$C_MANIPULATOR)
```

```
>perf_bal = performance(pred_ROC_bal,"tpr","fpr")
# Plotting the ROC curve
>plot(perf_bal, col = "black", lty = 3, lwd = 3)

# Calculating AUC
>auc = performance(pred_ROC_bal,"auc")
# Now converting S4 class to a vector
>auc = unlist(slot(auc, "y.values"))
# Adding min and max ROC AUC to the center of the plot
>minauc = min(round(auc, digits = 2))
>maxauc = max(round(auc, digits = 2))
>minauct = paste(c("min(AUC) = "), minauc, sep = "")
>maxauct = paste(c("max(AUC) = "), maxauc, sep = "")
>legend(0.7, 0.3, c(minauct, maxauct, "\n"), border = "white", cex = 0.5, box.col = "white")
>abline(a= 0, b=1)
>opt.cut = function(perf_bal, pred_ROC_bal){
 cut.ind = mapply(FUN=function(x, y, p){
  d = (x - 0)^2 + (y-1)^2
  ind = which(d == min(d))
  c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
   cutoff = p[[ind]])
 }, perf_bal@x.values, perf_bal@y.values, pred_ROC_bal@cutoffs)}

>print(opt.cut(perf_bal, pred_ROC_bal))

#sensitivity 0.8000000
#specificity 0.7719298
#cutoff      0.3575694

>pred_test_bal = ifelse(pred_test_bal>0.3575694,1,0)

>ptab<-table(pred_test_bal, sample_test$C_MANIPULATOR, dnn = c("Predicted","Actual"))

>library(robustbase)
>library(caret)

>confusionMatrix(ptab,positive = "1")
# Accuracy : 0.7761
# Sensitivity : 0.8000
# Specificity : 0.7719
```

```
> confusionMatrix(ptab,positive = "1")
Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 44  2
        1 13  8

               Accuracy : 0.7761
                 95% CI : (0.6578, 0.8689)
    No Information Rate : 0.8507
    P-Value [Acc > NIR] : 0.964568

                  Kappa : 0.3935
 Mcnemar's Test P-Value : 0.009823

            Sensitivity : 0.8000
            Specificity : 0.7719
         Pos Pred Value : 0.3810
         Neg Pred Value : 0.9565
             Prevalence : 0.1493
         Detection Rate : 0.1194
   Detection Prevalence : 0.3134
      Balanced Accuracy : 0.7860

       'Positive' Class : 1
```



Balancing the data - using 'SMOTE' technique

```
>install.packages("DMwR")
>library(DMwR)
>smote_sample<-SMOTE(C_MANIPULATOR~.,data = as.data.frame(sample_train),perc.over = 330,perc.under = 140)
>table(smote_sample$C_MANIPULATOR)
```

Logistic Regression

Variable Selection
>null = glm(C_MANIPULATOR~1, data= smote_sample, family = "binomial") # Includes only the intercept
>full = glm(C_MANIPULATOR~., data= smote_sample, family = "binomial")
#Forward Selection
>step(null, scope=list(lower=null, upper=full), direction="forward")

>lr_model_bal<- glm(C_MANIPULATOR ~ ACCR + DSRI + SGI + AQI + LEVI + DEPI + GMI,
data= smote_sample, family = "binomial")
summary(lr_model_bal)
>

```
> summary(lr_model_bal)

Call:
glm(formula = C_MANIPULATOR ~ ACCR + DSRI + SGI + AQI + LEVI +
    DEPI + GMI, family = "binomial", data = smote_sample)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5461  -0.5467  -0.0569   0.5494   1.8298

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.9067     2.1825  -5.456 4.88e-08 ***
ACCR          8.5209     1.4712   5.792 6.97e-09 ***
DSRI          1.6652     0.3868   4.305 1.67e-05 ***
SGI           5.1791     0.8822   5.870 4.35e-09 ***
AQI           0.9198     0.1661   5.539 3.05e-08 ***
LEVI         -1.4407     0.5208  -2.767  0.00567 **
DEPI          2.3330     1.0860   2.148  0.03169 *
GMI           0.9656     0.3786   2.551  0.01076 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 328.45  on 236  degrees of freedom
Residual deviance: 178.08  on 229  degrees of freedom
AIC: 194.08

Number of Fisher Scoring iterations: 8
```

> lr_model_bal$null.deviance-lr_model_bal$deviance
[1] 150.3641

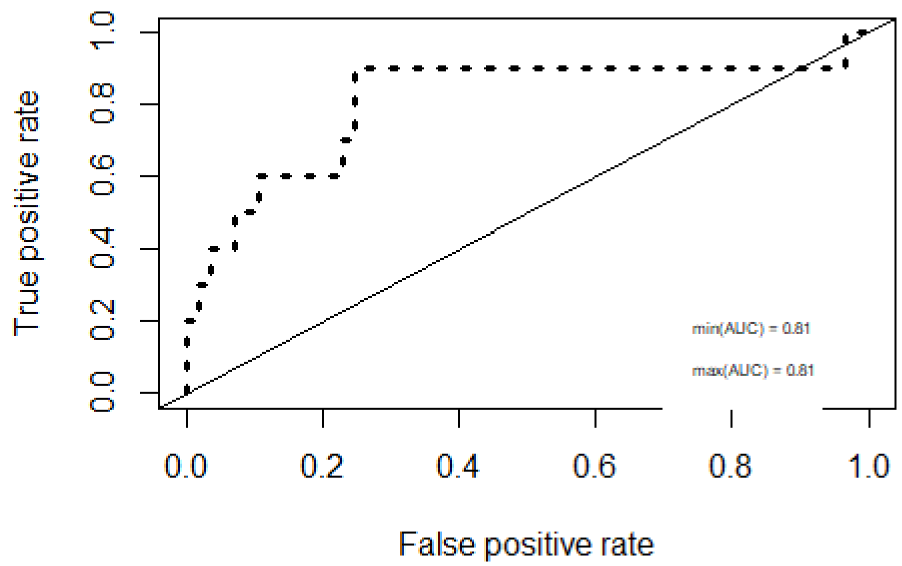
>pred_test_bal = predict.glm(lr_model_bal, newdata = sample_test, type="response")

Calculating the values for ROC curve
>pred_ROC_bal = prediction(pred_test_bal,sample_test$C_MANIPULATOR)

>perf_bal = performance(pred_ROC_bal,"tpr","fpr")
Plotting the ROC curve
>plot(perf_bal, col = "black", lty = 3, lwd = 3)

Calculating AUC
>auc = performance(pred_ROC_bal,"auc")
Now converting S4 class to a vector

```
>auc = unlist(slot(auc, "y.values"))
# Adding min and max ROC AUC to the center of the plot
>minauc = min(round(auc, digits = 2))
>maxauc = max(round(auc, digits = 2))
>minauct = paste(c("min(AUC) = "), minauc, sep = "")
>maxauct = paste(c("max(AUC) = "), maxauc, sep = "")
>legend(0.7, 0.3, c(minauct, maxauct, "\n"), border = "white", cex = 0.5, box.col = "white")
>abline(a= 0, b=1)
```




```
>opt.cut = function(perf_bal, pred_ROC_bal){
 cut.ind = mapply(FUN=function(x, y, p){
  d = (x - 0)^2 + (y-1)^2
  ind = which(d == min(d))
  c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
   cutoff = p[[ind]])
 }, perf_bal@x.values, perf_bal@y.values, pred_ROC_bal@cutoffs)}

>print(opt.cut(perf_bal, pred_ROC_bal))

#sensitivity 0.9000000
#specificity 0.7543860
#cutoff      0.2835403

>pred_test_bal = ifelse(pred_test_bal>0.2835403,1,0)

>ptab<-table(pred_test_bal, sample_test$C_MANIPULATOR, dnn = c("Predicted","Actual"))

>library(robustbase)
>library(caret)
```

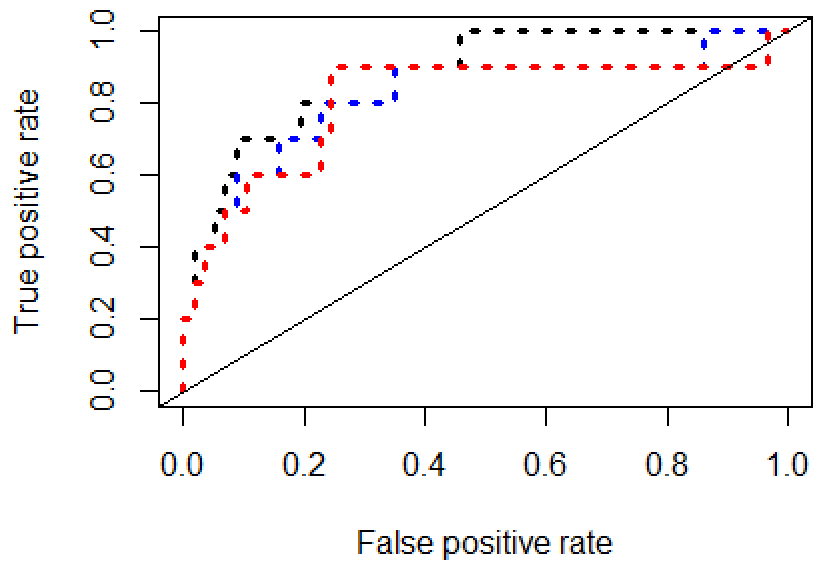
>confusionMatrix(ptab,positive = "1")
Accuracy : 0.7761
Sensitivity : 0.9000
Specificity : 0.7544

```
> confusionMatrix(ptab,positive = "1")
Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 43  1
        1 14  9

               Accuracy : 0.7761
                 95% CI : (0.6578, 0.8689)
    No Information Rate : 0.8507
    P-Value [Acc > NIR] : 0.964568

                  Kappa : 0.426
 Mcnemar's Test P-Value : 0.001946

            Sensitivity : 0.9000
            Specificity : 0.7544
         Pos Pred Value : 0.3913
         Neg Pred Value : 0.9773
             Prevalence : 0.1493
         Detection Rate : 0.1343
   Detection Prevalence : 0.3433
      Balanced Accuracy : 0.8272

       'Positive' Class : 1
```

OBSERVATIONS

	Unbalanced data	Over - Sampling	SMOTE
FORMULA (MODEL)	C_MANIPULATOR~DSRI + SGI + ACCR + AQI	C_MANIPULATOR ~ DSRI + SGI + AQI + ACCR + LEVI + GMI	C_MANIPULATOR ~ ACCR + DSRI + SGI + AQI + LEVI + DEPI + GMI
AIC	99.691	214.85	194.08
RESIDUAL DEVIANCE	148.581	343.80	328.45
NULL DEVIANCE	89.691	200.85	178.08
DEVIANCE DIFFERENCE	58.89034	142.95	150.36
CUT-OFF	0.1442311	0.3575694	0.2835403
SENSITIVITY	80 %	80%	90 %
SPECIFICITY	80.7 %	77.2 %	75.4 %
ACCURACY	80.6 %	77.6 %	77.6 %

Plotting ROC curve for all model constructed above to compare AUC for determining the best model.



Black →Unbalanced Data (Data before sampling)
Red →Data sampled using over-sampling technique
Green → Data sampled using SMOTE technique

CONCLUSION

- On the basis of the above observation, we can say that the SMOTE method is the most effective technique to balance the sample data.
- As we can see from the output table above, the logistic Regression algorithm gives the best result(highest sensitivity) in case of SMOTE sampled data

Hence, we recommend MCA Technology Solutions to adopt SMOTE method as the balancing technique to construct and deploy the logistic regression model.

PROBLEM 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.

Solution

As we selected the logistic Regression model which was constructed on the balanced data (data balanced using the SMOTE technique), the cut-off point obtained in that model will be considered as the M-score.

M-score → 0.2835403

This signifies that –

- M-score of greater than 0.2835403 would be classified as potential Manipulators.
- M-score less than or equal to 0.3402 would be classified as potential Non-manipulators

PROBLEM 7

Develop classification and regression tree (CART) model. What insights do you obtain from the CART model?

Solution

CART model on sample data (220 cases)

```
>install.packages("partykit")
>install.packages("rpart")
>install.packages("rpart.plot")

>library(rpart)
>library(rpart.plot)
>library(partykit)


>dt_bal_full = rpart(C_MANIPULATOR~., data = smote_sample,
              control= rpart.control(cp= -1, minsplit = 0,
                        minbucket = 0 ),
              parms = list(split="gini"))



>printcp(dt_bal_full)
>opt <-which.min(dt_bal_full$cptable[ ,"xerror"])
>opt

>cp<-dt_bal_full$cptable[opt, "CP"]
>cp
```

```
>dt_bal_pruned <- prune(dt_bal_full, cp = cp)

>summary(dt_bal_pruned)
>dt_bal_pruned$variable.importance
>dt_bal_pruned$splits


>tab_train_dt<-table(predict(dt_bal_pruned,type="class"), >smote_sample$C_MANIPULATOR, dnn =
c("Predicted","Actual"))

>confusionMatrix(tab_train_dt, positive = "1")

>tab_test_dt<-table(predict(dt_bal_pruned, type="class", newdata = >sample_test), sample_test$C_
MANIPULATOR, dnn = c("Predicted","Actual"))
>confusionMatrix(tab_test_dt, positive = "1")

# Accuracy : 0.8806
# Sensitivity : 0.60000
# Specificity : 0.92982
```

CART model on complete data

```
>dt_bal_full = rpart(C_MANIPULATOR~., data = Complete_Data_bal1,
              control= rpart.control(cp= -1, minsplit = 0,
                               minbucket = 0 ),
              parms = list(split="gini"))



>printcp(dt_bal_full)
>opt <-which.min(dt_bal_full$cptable[ ,"xerror"])
>opt

>cp<-dt_bal_full$cptable[opt, "CP"]
>cp

>dt_bal_pruned <- prune(dt_bal_full, cp = cp)

>summary(dt_bal_pruned)
>dt_bal_pruned$variable.importance
>dt_bal_pruned$splits


>tab_train_dt<-table(predict(dt_bal_pruned, type="class"), >Complete_Data_bal1$C_MANIPULATO
R, dnn = c("Predicted","Actual"))

>confusionMatrix(tab_train_dt, positive = "1")
```

```
>tab_test_dt<-table(predict(dt_bal_pruned, type="class", newdata = TestData), sample_test$C_MA
NIPULATOR, dnn = c("Predicted","Actual"))
>confusionMatrix(tab_test_dt, positive = "1")

# Accuracy : 0.8507
# Sensitivity : 0.30000
    # Specificity : 0.94737
```

OBSERVATIONS

	Sample data(SMOTE)	Complete data
SENSITIVITY	60 %	30 %
SPECIFICITY	92.98 %	94.7 %
ACCURACY	88.06 %	85.07%

PROBLEM 8

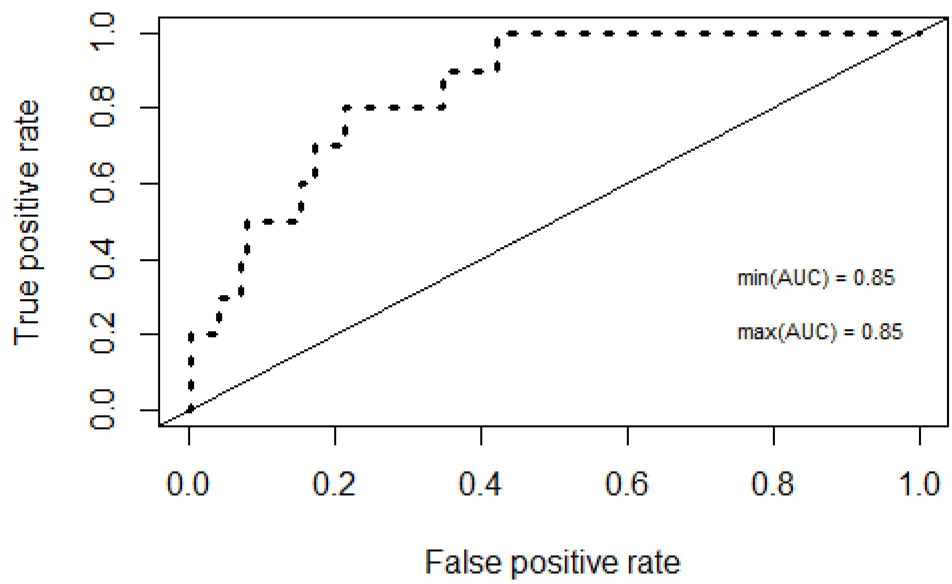
Develop a logistic regression model using the complete data set (1200 non-manipulators and 39 manipulators), compare the results with the previous logistic regression model.

Solution

Balancing the data - using "over-sampling" technique

```
>library(readxl)
>Complete_Data <- read_excel("~/UIC/Courses/DataMining/Assignments/Case >Study 1/Dataset/Co
mplete_Data.xlsx")
>View(Complete_Data)

>Complete_Data$`Company ID`<- NULL
>Complete_Data$Manipulater<- NULL

>colnames(Complete_Data)[9]<-"C_MANIPULATOR"

>Complete_final <- Complete_Data
>Complete_final$C_MANIPULATOR <- as.factor(Complete_final$C_MANIPULATOR)
>class(Complete_final$C_MANIPULATOR)

>table(Complete_final$C_MANIPULATOR)
# 0     1
# 1200  39

>set.seed(1234)
>index = sample(2, nrow(Complete_final), replace = TRUE, prob = c(0.65,0.35))
>TrainData = Complete_final[index == 1, ]
>table(TrainData$C_MANIPULATOR)
>TestData = Complete_final[index == 2,]
```

```
>table(TestData$C_MANIPULATOR)

############## oversampling ########
>Complete_Data_bal1 <- ovun.sample(C_MANIPULATOR~.,
                        data = TrainData,method = "over",
                        N=1566)$data
>table(Complete_Data_bal1$C_MANIPULATOR)

# Variable Selection
>null = glm(C_MANIPULATOR~1, data= Complete_Data_bal1, family = "binomial")
>data= Complete_Data_bal1, family = "binomial")
#Forward Selection
>step(null, scope=list(lower=null, upper=full), direction="forward")

>lr_complete_bal1<- glm(C_MANIPULATOR ~ DSRI + ACCR + SGI + AQI, data= Complete_Data_bal1,
                family = "binomial")
>summary(lr_complete_bal1)
#Null deviance: 2170.9  on 1565  degrees of freedom
#Residual deviance: 1097.3  on 1558  degrees of freedom
#AIC: 1113.3

>lr_complete_bal1$null.deviance-lr_complete_bal1$deviance
#1073.635

>pred1 = predict.glm(lr_complete_bal1, newdata = TestData, type="response")

# Calculating the values for ROC curve
>pred_ROC1 = prediction(pred1,TestData$C_MANIPULATOR)
>perf1 = performance(pred_ROC1,"tpr","fpr")
# Plotting the ROC curve
>plot(perf1, col = "black", lty = 3, lwd = 3)

# Calculating AUC
>auc1 = performance(pred_ROC1,"auc")
# Now converting S4 class to a vector
>auc1 = unlist(slot(auc1, "y.values"))
# Adding min and max ROC AUC to the center of the plot
>minauc1 = min(round(auc1, digits = 2))
>maxauc1 = max(round(auc1, digits = 2))
>minauct1 = paste(c("min(AUC) = "), minauc1, sep = "")
>maxauct1 = paste(c("max(AUC) = "), maxauc1, sep = "")
>legend(0.7, 0.5, c(minauct1, maxauct1, "\n"), border = "white",
     cex = 0.7, box.col = "white")
>abline(a= 0, b=1)
```



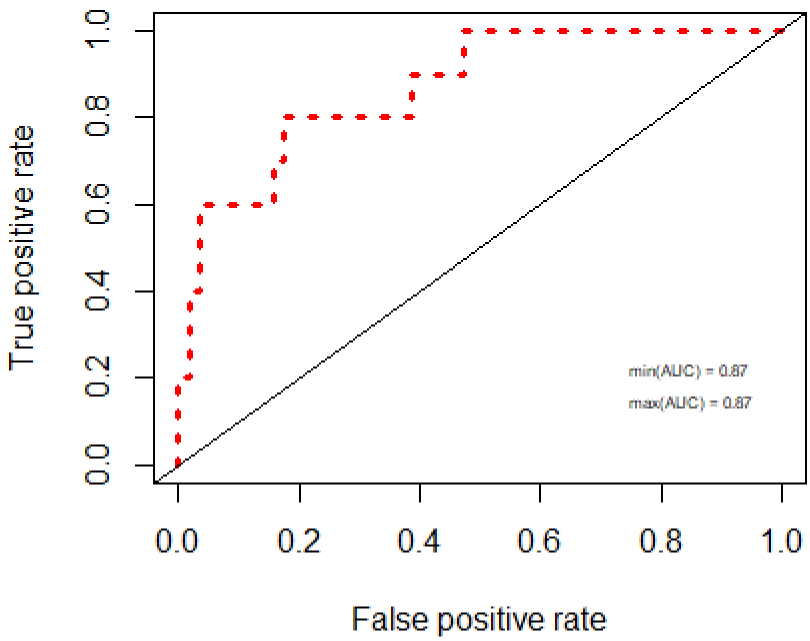

```
>opt.cut1 = function(perf, pred_ROC1){
  cut.ind = mapply(FUN=function(x, y, p){
    d = (x - 0)^2 + (y-1)^2
    ind = which(d == min(d))
    c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
      cutoff = p[[ind]])
  }, perf1@x.values, perf1@y.values, pred_ROC1@cutoffs)}

>opt.cut1

>print(opt.cut1(perf1, pred_ROC1))

#sensitivity 0.8000000
#specificity 0.7697800
#cutoff      0.3251948

>pred1$C_MANIPULATOR = ifelse(pred1>0.3251948,1,0)

>tab1<-table(pred1$C_MANIPULATOR, TestData$C_MANIPULATOR, dnn = c("Predicted","Actual"))
>tab1

>library(robustbase)
>library(caret)

>confusionMatrix(tab1,positive = "1")
#Accuracy : 0.7705
#Sensitivity : 0.80000
#Specificity : 0.76978
```

```
> confusionMatrix(tab1,positive = "1")
Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 321   2
        1  96   8

               Accuracy : 0.7705
                 95% CI : (0.7276, 0.8096)
    No Information Rate : 0.9766
    P-Value [Acc > NIR] : 1

                  Kappa : 0.102
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.80000
            Specificity : 0.76978
         Pos Pred Value : 0.07692
         Neg Pred Value : 0.99381
             Prevalence : 0.02342
         Detection Rate : 0.01874
   Detection Prevalence : 0.24356
      Balanced Accuracy : 0.78489

       'Positive' Class : 1
```

Balancing the data - using “SMOTE” technique

```
>smote_complete<-SMOTE(C_MANIPULATOR~.,data = as.data.frame(TrainData),perc.over = 2600,pe
rc.under = 105)
>table(smote_sample$C_MANIPULATOR)

# Logistic Regression

# Variable Selection
>null = glm(C_MANIPULATOR~1, data= smote_complete, family = "binomial") # Includes only the int
ercept
>full = glm(C_MANIPULATOR~., data= smote_complete, family = "binomial")
#Forward Selection
>step(null, scope=list(lower=null, upper=full), direction="forward")

>lr_model_bal3<- glm(C_MANIPULATOR ~ ACCR + DSRI + SGI + AQI + LEVI,
          data= smote_sample, family = "binomial")
>summary(lr_model_bal3)

>lr_model_bal3$null.deviance-lr_model_bal3$deviance


>pred_test_bal3 = predict.glm(lr_model_bal3, newdata = TestData, type="response")

# Calculating the values for ROC curve
>pred_ROC_bal3 = prediction(pred_test_bal3, TestData$C_MANIPULATOR)

>perf_bal3 = performance(pred_ROC_bal3,"tpr","fpr")
# Plotting the ROC curve
```

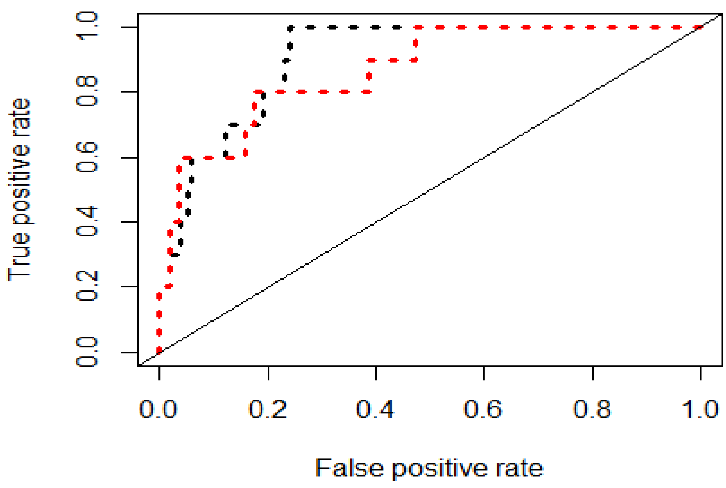
```
>plot(perf_bal3, col = "red", lty = 3, lwd = 3)

# Calculating AUC
>auc3 = performance(pred_ROC_bal3,"auc")
# Now converting S4 class to a vector
>auc3 = unlist(slot(auc3, "y.values"))
# Adding min and max ROC AUC to the center of the plot
>minauc3 = min(round(auc3, digits = 2))
>maxauc3 = max(round(auc3, digits = 2))
>minauct3 = paste(c("min(AUC) = "), minauc3, sep = "")
>maxauct3 = paste(c("max(AUC) = "), maxauc3, sep = "")
>legend(0.7, 0.3, c(minauct3, maxauct3, "\n"), border = "white", cex = 0.5,
    box.col = "white")
>abline(a= 0, b=1)

>opt.cut3 = function(perf_bal3, pred_ROC_bal3){
  cut.ind = mapply(FUN=function(x, y, p){
   d = (x - 0)^2 + (y-1)^2
   ind = which(d == min(d))
   c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
     cutoff = p[[ind]])
  }, perf_bal3@x.values, perf_bal3@y.values, pred_ROC_bal3@cutoffs)}

>print(opt.cut(perf_bal3, pred_ROC_bal3))

#sensitivity 0.8000000
#specificity 0.8245614
#cutoff      0.3706433

>pred_test_bal3 = ifelse(pred_test_bal3>0.3706433,1,0)

>ptab<-table(pred_test_bal3, TestData$C_MANIPULATOR, dnn = c("Predicted","Actual"))

>library(robustbase)
>library(caret)

>confusionMatrix(ptab,positive = "1")
# Accuracy : 0.7761
# Sensitivity : 0.9000
# Specificity : 0.7544
```




```
> confusionMatrix(ptab,positive = "1")
Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 47  2
        1 10  8

               Accuracy : 0.8209
                 95% CI : (0.708, 0.9039)
    No Information Rate : 0.8507
    P-Value [Acc > NIR] : 0.80753

                  Kappa : 0.4697
 Mcnemar's Test P-Value : 0.04331

            Sensitivity : 0.8000
            Specificity : 0.8246
         Pos Pred Value : 0.4444
         Neg Pred Value : 0.9592
             Prevalence : 0.1493
         Detection Rate : 0.1194
   Detection Prevalence : 0.2687
      Balanced Accuracy : 0.8123

       'Positive' Class : 1
```

OBSERVATIONS

	Over - Sampling	SMOTE
FORMULA (MODEL)	C_MANIPULATOR ~ DSRI + SGI + AQI + ACCR + LEVI + GMI	C_MANIPULATOR ~ ACCR + DSRI + SGI + AQI + LEVI
AIC	1113.3	1142.1
RESIDUAL DEVIANCE	2170.9	2182.0
NULL DEVIANCE	1097.3	1130.1
DEVIANCE DIFFERENCE	1073.635	1051.926
CUT-OFF	0.2711244	0.3706433
SENSITIVITY	100%	80%
SPECIFICITY	76.02 %	82.46 %
ACCURACY	76.58 %	82.09 %



Black → Data sampled using over-sampling technique

Red → Data sampled using SMOTE technique

CONCLUSION:

Looking into the above observation, we can say that for the complete dataset, “over-sampling” technique is better than “SMOTE” because sensitivity is higher in case of over-sampling method.

PROBLEM 9

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

Solution

We will construct Random Forest and Boosting model for both the sample data and complete data.

1. **For 220 samples:**

Random Forest:

```
>install.packages("randomForest")
>library(randomForest)
>rf1 = randomForest(C_MANIPULATOR~SGI+LEVI+ACCR,
            data = sample_train, ntree = 100,
            proximity = TRUE, replace= TRUE,
            importance = TRUE, mtry = sqrt(ncol(sample_train)))

>pred_test_rf1=predict(rf1,newdata = sample_test)
>confusionMatrix(pred_test_rf1,sample_test$C_MANIPULATOR, dnn=c("Predicted","Actual"),positiv
e = "1")
```

```
Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 53  5
        1  4  5

               Accuracy : 0.8657
                 95% CI : (0.7603, 0.9367)
    No Information Rate : 0.8507
    P-Value [Acc > NIR] : 0.4477

                  Kappa : 0.4483
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.50000
            Specificity : 0.92982
         Pos Pred Value : 0.55556
         Neg Pred Value : 0.91379
             Prevalence : 0.14925
         Detection Rate : 0.07463
   Detection Prevalence : 0.13433
      Balanced Accuracy : 0.71491

       'Positive' Class : 1
```

Boosting:

```
>install.packages("mboost")
>library(mboost)
```

```
>data.adaboost1 <- mboost(C_MANIPULATOR ~., data =sample_train,
                  family = Binomial(type=c("adaboost")),
                  control = boost_control(mstop=500))

>data_pred2 <- predict(data.adaboost1, newdata = sample_test, type="class")
>ptab2<-table(data_pred2, sample_test$C_MANIPULATOR, dnn = c("Predicted","Actual"))
>confusionMatrix(ptab2, positive = "1")
Confusion Matrix and Statistics

         Actual
Predicted  0  1
        0 55  6
        1  2  4

               Accuracy : 0.8806
                 95% CI : (0.7782, 0.947)
    No Information Rate : 0.8507
    P-Value [Acc > NIR] : 0.3144

                  Kappa : 0.437
 Mcnemar's Test P-Value : 0.2888

            Sensitivity : 0.40000
            Specificity : 0.96491
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.90164
             Prevalence : 0.14925
         Detection Rate : 0.05970
   Detection Prevalence : 0.08955
      Balanced Accuracy : 0.68246

       'Positive' Class : 1
```

COMPARISON AMONG MODELS BASED ON PERFORMANCE ON THE SAMPLE DATA

Method	Sensitivity on Test Data in %	Accuracy on Test Data in %
Logistic Regression	90	77.6
Classification tree	60	88.06
Random Forest	50	86.57
Boosting	40	88.06

CONCLUSION:

As we can see from the above observation (showing performance of each model), Logistic Regression would be the best model to predict the manipulator class in the sample data. We can see that the sensitivity is the highest for Logistic Regression model among all the cases. Even though the overall model accuracy is not the highest in this case, we would recommend using Logistic Regression, on a balanced sampled data using SMOTE method, because the class of interest is predicted better by this model.

2. **For complete data set:**

Random Forest:

```
>rf3 = randomForest(C_MANIPULATOR~SGI+LEVI+DSRI+ACCR,
            data = TrainData, ntree = 100,
            proximity = TRUE, replace= TRUE,
            importance = TRUE, mtry = sqrt(ncol(sample_train)))

>pred_test_rf3=predict(rf3,newdata = TestData)
>confusionMatrix(pred_test_rf3,TestData$C_MANIPULATOR,          dnn=c("Predicted","Actual"),p
ositive = "1” )
```

```
Confusion Matrix and Statistics

         Actual
Predicted   0   1
        0 414   7
        1   3   3

               Accuracy : 0.9766
                 95% CI : (0.9574, 0.9887)
    No Information Rate : 0.9766
    P-Value [Acc > NIR] : 0.5830

                  Kappa : 0.3638
 Mcnemar's Test P-Value : 0.3428

            Sensitivity : 0.300000
            Specificity : 0.992806
         Pos Pred Value : 0.500000
         Neg Pred Value : 0.983373
             Prevalence : 0.023419
         Detection Rate : 0.007026
   Detection Prevalence : 0.014052
      Balanced Accuracy : 0.646403

       'Positive' Class : 1
```

Boosting:

```
>data.adaboost4 <- mboost(C_MANIPULATOR ~., data =TrainData,
             family = Binomial(type=c("adaboost")),
             control = boost_control(mstop=500))

>data_pred4 <- predict(data.adaboost4, newdata = TestData, type="class")
>ptab4<-table(data_pred4, TestData$C_MANIPULATOR, dnn = c("Predicted","Actual"))
>confusionMatrix(ptab4, positive = "1")
```

```
Confusion Matrix and Statistics

          Actual
Predicted   0   1
        0 415   7
        1   2   3

               Accuracy : 0.9789
                 95% CI : (0.9604, 0.9903)
    No Information Rate : 0.9766
    P-Value [Acc > NIR] : 0.4564

                  Kappa : 0.3905
 Mcnemar's Test P-Value : 0.1824

            Sensitivity : 0.300000
            Specificity : 0.995204
         Pos Pred Value : 0.600000
         Neg Pred Value : 0.983412
             Prevalence : 0.023419
         Detection Rate : 0.007026
   Detection Prevalence : 0.011710
      Balanced Accuracy : 0.647602

       'Positive' Class : 1
```

COMPARISON AMONG MODELS BASED ON PERFORMANCE ON COMPLETE DATA

Method	Accuracy on Test Data in %	Sensitivity on Test Data in %
Logistic Regression	77.05	80
Classification tree	85.07	30
Random Forest	97.66	30
Boosting	97.89	30

CONCLUSION:

As we can see from the above observation (showing performance of each model), Logistic Regression would be the best model to predict the manipulator class in the sample data. We can see that the sensitivity is the highest for Logistic Regression model among all the cases. Even though the overall model accuracy is not the highest in this case, we would recommend using Logistic Regression, on a balanced sampled data using “over-Sampling” method, because the class of interest is predicted better by this model.

PROBLEM 10

What will be your final recommendation for predicting earnings manipulators?

Solution

After our exploration and analysis of the given sample and complete data, we would provide below recommendations for predicting earning manipulators-

1. As we saw that the given dataset is an unbalanced data, in which case the models does not perform well (as they may become biased towards the majority class), we should take actions to balance the data using appropriate technique. As per our analysis and comparison between over-sampling and SMOTE method, we found that SMOTE method provided better sensitivity rate.
   Hence, we recommend to balance the data using SMOTE sampling technique and then perform desired modelling on the balanced data.
2. As clear from all the above observations and comparison, Logistic regression model provided best performance(highest sensitivity rate) for predicting earning manipulators when compared to other machine learning techniques like – Classification & Regression Tree, Boosting and Random Forest.
   Hence, we recommend to apply Logistic Regression model on a balanced data to predict earning manipulators most effectively.