HW 3

IDS 572

Ajay Pawar (676955899) Darshan Radadiya (656230601) Priyanshi Patel (650927804)

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

Beneish model was found in 1999 commonly known as "M-Score Model" which uses eight financial ratios weighted by coefficients to identify whether a company has manipulated its profits and to categorize Manipulators and Non-manipulators, using the formula,

M-score = −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.

This model does not perform well when used with the Indian data as the data is very unbalanced. It does predict all the manipulators but also predicts More non-manipulators as manipulators. Moreover, the model was built keeping US companies as its base, it would be difficult to match parameters on how the companies perform on Indian companies.

Also, there is uncertainty on how the data is organized.

The correlation between variable for Indian data may lead different outcomes when the model is applied.

So No, Beneish Model isn't the best model when used with the Indian data.

(2) The number of manipulators is usually much less than non-manipulators (in the accompanying spreadsheet, the percentage of manipulators is less than 4% in the complete data). What kind of modeling problems can one expect when cases in one class are much lower than the other class in a binary classification problem? How can one handle these problems? – To answer this question: Watch "balancing data in R" video uploaded under "R documents".

Unbalanced Data: this is a scenario when number of observations in one class outnumbers another class by a large proportion, which creates "data-imbalance". If, faced with this scenario, the model performance is not accurate, and the algorithm only concentrates on predicting the majority class and ignore the minority class data by treating them as noise.

The problem is most common in binary-level classification than multi-class classification.

Common Techniques used to handle class-imbalance:

- Under-sampling: Randomly selects a subset of sample from the class with more instances. E.g.: 458 negative samples & 240 positive samples. Randomly select 240 from negative samples and ignore the rest. Disadvantage: We lose potentially relevant info from the left-out sample.

- Over-sampling: Randomly duplicate samples from the class with fewer instances. Disadvantage: Might over-estimated the model.
- Synthetic Minority Sampling Technique (SMOTE)

(3) Use 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.

First, we imported the necessary libraries required for the model to further built our Logistic Regression Model.

```
library(readxl)
library(caret)
library(pROC)
library(ROSE)
library(robustbase)
library(smotefamily)
library(ROCR)
library(rpart)
library(rpart.plot)

#Read the DataSet from Excel
dataSet <- read_excel("C:/Users/DELL/Desktop/UIC/Sem 3/IDS 572/Case Study/IMB579-XLS-ENG.xlsx",
                      sheet = "Sample for Model Development")

#Replace '-' in the column name 'C-MANIPULATOR' to BLANK
names(dataSet)[names(dataSet) == "C-MANIPULATOR"] <- "CMani"
str(dataSet)

colSums(is.na(dataSet)) #Check for NA values

#Change the Manipulator column values to numeric along with its type
dataSet$Manipulator <- ifelse(dataSet$Manipulator == "Yes", 1, 0)
df <- data.frame(dataSet)
```

```
#df$CMani <- as.factor(df$CMani)
str(df)
table(df$CMani)
prop.table(table(df$CMani))

####### Q3- Developing Logistic Regression Model #######

#Divide the data into Training and Testing
set.seed(1234)
splitIndex <- createDataPartition(df$CMani, p = .50,
                                  list = FALSE, times = 1)
trainingSet <- df[splitIndex, ]
testingSet <- df[-splitIndex, ]
str(trainingSet)
table(trainingSet$CMani)
prop.table(table(trainingSet$CMani))

#Using Treebag Model by Cross Validation, a simple bagging model is created.
#The Treebag Model will provide the Accuracy for training data
ctrl <- trainControl(method = "cv", number = 5)
tbModel <- train(CMani ~ ., data = trainingSet,
                 method = "treebag", trControl = ctrl)
predictors <- names(trainingSet)[names(trainingSet) != 'CMani']
pred <- predict(tbModel$finalModel, testingSet[, predictors])
auc <- roc(testingSet$CMani, pred)
print(auc)

#Using Treebag Model, the following determines the accuracy on Test Data
str(testingSet)
table(testingSet$CMani)
```

```
prop.table(table(testingSet$CMani))


ctrlTest <- trainControl(method = "cv", number = 5)
tbTestModel <- train(CMani ~ ., data = testingSet,
                     method = "treebag", trControl = ctrl)
predictorsTest <- names(testingSet)[names(testingSet) != 'CMani']
predTest <- predict(tbTestModel$finalModel, testingSet[, predictorsTest])
aucTest <- roc(testingSet$CMani, predTest)
print(aucTest)
```

Using Tree bag Model by Cross Validation, we first predicted the accuracy on training data and then found 100% accuracy on test data.

(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?

We found imbalance in the previous dataset, so we implemented **Smote Technique to create new data frame which is more balanced.**

Using this more balanced dataset we built a Logistic Regression model with 100% Accuracy.

```
#Using SMOTE - will design a new dataset for the modeling
smoteTrain <- trainingSet
smoteTrain$CMani <- as.factor(smoteTrain$CMani)
table(smoteTrain$CMani)
prop.table(table(smoteTrain$CMani))
str(smoteTrain)


smoteTrain <- SMOTE(smoteTrain[-11], smoteTrain$CMani)
smoteNew <- smoteTrain$data
table(smoteNew$class)
prop.table(table(smoteNew$class))


######## Q4-Performance on Test&Train #########
```

```
# Logistic Regression for the data Balanced by Smote

# We use Accuracy measure to evaluate performance.

# Along AIC.

smoteLogistic <- glm(as.factor(class) ~ ., data = smoteNew, family = "binomial")

summary(smoteLogistic)

smotePredict <- predict(smoteLogistic, type = "response")

smotePredict <- ifelse(smotePredict > 0.50, 1, 0)

confusionMatrix(as.factor(smoteNew$class), as.factor(smotePredict), positive = "1")

## The model gives 100% accuracy of Training Data
```

```
          Reference
Prediction  0  1
         0 94  0
         1  0 80

               Accuracy : 1
                 95% CI : (0.979, 1)
    No Information Rate : 0.5402
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4598
         Detection Rate : 0.4598
   Detection Prevalence : 0.4598
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

```
#Logistic Regression on Test Data

#Using SMOTE - will design a new Test dataset for the modeling

smoteTest <- testingSet

smoteTest$CMani <- as.factor(smoteTest$CMani)

table(smoteTest$CMani)

prop.table(table(smoteTest$CMani))

str(smoteTest)
```

```
smoteTest <- SMOTE(smoteTest[-11], smoteTest$CMani)
smoteTestNew <- smoteTest$data
table(smoteTestNew$class)
prop.table(table(smoteTestNew$class))

#Logistic Regression for the data Balanced by Smote
#Running Logistic Regression Model
smoteTestLogistic <- glm(as.factor(class) ~ ., data = smoteTestNew, family = "binomial")
summary(smoteTestLogistic)

smoteTestPredict <- predict(smoteTestLogistic, type = "response")
smoteTestPredict <- ifelse(smoteTestPredict > 0.50, 1, 0)
confusionMatrix(as.factor(smoteTestNew$class), as.factor(smoteTestPredict), positive = "1")
## The model gives 100% accuracy for Test Data
```

```
          Reference
Prediction  0  1
         0 87  0
         1  0 69

               Accuracy : 1
                 95% CI : (0.9766, 1)
    No Information Rate : 0.5577
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4423
         Detection Rate : 0.4423
   Detection Prevalence : 0.4423
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

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

```
library(readxl)
earn <- read_excel("IMB579-XLS-ENG.xlsx",sheet = "Complete Data")
sam <-  read_excel("IMB579-XLS-ENG.xlsx",sheet = "Sample for Model Development")
# Backup of the data
e<- earn
s<- sam

# removing variables which are not required for analysis
earn$`Company ID`<- NULL
earn$`C-MANIPULATOR`<- NULL

sam$`Company ID`<- NULL
sam$`C-MANIPULATOR`<- NULL

# changing name of the target variable
names(earn)[9] <- "Target"
names(sam)[9] <- "Target"

str(earn)

# Changing target variable to Factor
earn$Target <- factor(earn$Target)
sam$Target <- factor(sam$Target)

# Checking Missing values in the data

colSums(is.na(earn))
```

```
colSums(is.na(sam))
#install.packages('UBL')

library(UBL)
# Complete Data
# let us split the data into Training and Testing data
set.seed(1234)
index <- sample(1:2, nrow(earn), replace = TRUE, prob = c(0.7,0.3))
train_com <- earn[index == 1, ]
test_com <- earn[index == 2, ]

# sampling the data from 3 fronts
table(train_com$Target)

# Sample Data
# let us split the data into Training and Testing data
set.seed(9999)
index1 <- sample(1:2, nrow(sam), replace = TRUE, prob = c(0.7,0.3))
train_sam <- sam[index1 == 1, ]
test_sam <- sam[index1 == 2, ]

# sampling the data from three fronts
table(train_sam$Target)

over_com <- RandOverClassif(Target~.,as.data.frame(train_com), "balance")
table(over_com$Target)

smote_com <- SmoteClassif(Target~.,as.data.frame(train_com), "balance")
table(smote_com$Target)
```

```
under_com <- RandUnderClassif(Target~.,as.data.frame(train_com), "balance")
table(under_com$Target)

smote_sam <- SmoteClassif(Target~.,as.data.frame(train_sam), "balance")
table(smote_sam$Target)

under_sam <- RandUnderClassif(Target~.,as.data.frame(train_sam), "balance")
table(under_sam$Target)

over_sam <- RandOverClassif(Target~.,as.data.frame(train_sam), "balance")
table(over_sam$Target)
# original Complete Data
table(earn$Target)
prop.table(table(earn$Target)) # High Imbalance
table(train_com$Target)
prop.table(table(train_sam$Target)) # further Imbalance
# Sampled Data
table(under_com$Target)
table(over_com$Target)
table(smote_com$Target)
# Model 2 - only on training data with under Sampling Technique
model_und_random <- randomForest(Target ~ ., data = under_com, ntree = 500, mtry = 6, importance =
TRUE, proximity= TRUE)

# Checking which model performs best on Testing data
# Model 2 - Under Sampling
Pred2 <- predict(model_und_random, test_com)
confusionMatrix(Pred2,test_com$Target, positive = "Yes")

# Looks like undersampling  technique, we are getting the best result
```

```
# where RECALL is 87.5%, FPR - 27.8% and Acc - 72.6%

# To check important variables

importance(model_und_random)

varImpPlot(model_und_random)


# Top 4 Variable Importance

varImpPlot(model_und_random, sort = T, n.var = 4, main = "Top 4 Important Variables by Random
Forest")


library("adabag")


adaboost <- boosting(Target ~ ., data = under_com, mfinal = 100, control = rpart.control(maxdepth = 1))

summary(adaboost)


pred_boost = predict.boosting(adaboost, newdata = as.data.frame(test_com))

summary(pred_boost)


print("Confusion Matrix using Adaboost")


print(pred_boost$confusion)


test_acc = 1-(pred_boost$error)

test_acc # Accuracy of 80.82%


library(ROCR)

pred <- prediction(model_und_random$votes[, 2], under_com$Target)


# Gain Chart

perf <- performance(pred, "tpr", "rpp")

plot(perf)
```

```
# Response Chart
perf <- performance(pred, "ppv", "rpp")
plot(perf)

# Lift Chart
perf <- performance(pred, "lift", "rpp")
plot(perf)

# ROC Curve
perf <- performance(pred, "tpr", "fpr")
plot(perf)

# auc
auc <- performance(pred, "auc")
auc
auc <- unlist(slot(auc, "y.values"))
auc

# Identifying the optimal cut-off point from ROC curve using the distance to (0,1) approach
library(ROCR)
pred<-prediction(rf$votes[,2], under_comunder_com$Target)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
# The perf contain all information we require
# The tpr, fpr values and their corresponding cut-off points are considered in perf@x.values, perf@y.values,perf@alpha.values respectively.
# Notice that perf@x.values, perf@y.values,perf@alpha.values are lists. To access the ith element of these list you need to use perf@x.values[[i]], perf@y.values[[i]], perf@alpha.values[[i]]
# We want to write a function that receives "perf". It then computes the distances of all the points in (perf@x.values, perf@y.values) from the point (0, 1) and finds the minimum
```

```
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]]) # return the corresponding tpr, fpr and the cutoff point
}

opt.cut <- function(perf){
  cut.ind <- mapply(FUN = mydistance,
                    perf@x.values, perf@y.values,perf@alpha.values)
}
Output <- opt.cut(perf)
print(Output[,1])

Threshold <- Output[,1]["cutoff"]
predictedClass <- as.factor(ifelse(model_und_random$votes[,2] >= Threshold, "Low", "High"))
CM1 <- table(predictedClass, under_com$Target, dnn = c("Predicted","Actual"))
```

---

(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.

Since we used the unbalanced dataset and constructed a more balanced dataset using SMOTE Technique, the cut-off point will be considered as the M-Score.

This means that an M-Score greater than 0.296 would be classified as potential manipulators

While the M-score less than 0.296 would be potential non-manipulators.

Thus,

M-Score > 0.29 → Manipulators

M-Score <=0.29 → Non-Manipulators

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

```
dtree <- rpart(CMani~ ., data = trainingSet,
               control = rpart.control(cp = -1, minsplit = 0, minbucket = 0),
               parms = list(split = "gini"))
print(dtree)
rpart.plot(dtree)

# Decision Tree with testing data
dtree_test <- rpart(CMani~ ., data = testingSet,
               control = rpart.control(cp = -1, minsplit = 0, minbucket = 0),
               parms = list(split = "gini"))
print(dtree_test)
rpart.plot(dtree_test)

# Optimal CP
optCP <- which.min(dtree_test$cptable[,"xerror"] )
optCP
# the best CP value
CP <- dtree_test$cptable[optCP, "CP"]
CP

#Now we can prune the tree based on this best CP value
dtree_pruned  <- prune(dtree_test, cp = CP)
summary(dtree_pruned)

#Confusion matrix for the DecisionTree
train_dt<-table(predict(dtree_pruned, type="class", trainingSet$CMani))

confusionMatrix(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")
```

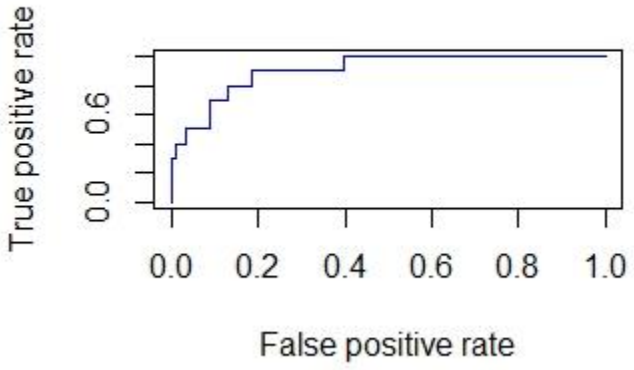
Model	Sensitivity	Specificity	Cut-off
Logistic Regression	1.0	1.0	0.50
Logistic Regression (complete data)	0.90	0.814	0.296

(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.

```
##### Q8-Linear Regression model for Complete data ######


dataset_complete <- read_excel("C:/Users/DELL/Desktop/UIC/Sem 3/IDS 572/Case Study/IMB579-XLS-
ENG.xlsx",
                   sheet = "Complete Data")

dataset_complete$`Company ID`<- NULL


# Replace '-' in the column name 'C-MANIPULATOR' to BLANK

names(dataset_complete)[names(dataset_complete) == "C-MANIPULATOR"] <- "CMani"

str(dataset_complete)


table(dataset_complete$CMani)

colSums(is.na(dataset_complete))


# Change the Manipulator column values to numeric along with its type

dataset_complete$Mani_New <- ifelse(dataset_complete$Manipulater == "Yes", 1, 0)

str(dataset_complete)


# Splitting the complete data into Training and Testing

set.seed(123)
```

```
index = sample(2, nrow(dataset_complete), replace = TRUE, prob = c(0.8,0.2))
Train_complete = dataset_complete[index == 1, ]
nrow(Train_complete)
table(Train_complete$CMani)
Test_complete = dataset_complete[index == 2,]
nrow(Test_complete)
table(Test_complete$CMani)

# Building Logistic Regression model
LTrain_complete <- glm(CMani ~ .,
                       data = Train_complete,
                       family = "binomial")
summary(LTrain_complete)

#Using oversampling for predicting
Ltrain_over <- ovun.sample(CMani~ .,
                           data = Train_complete,
                           method = "over",
                           N=1566)$data
table(Ltrain_over$CMani)

# Variable selection for Logistic Model
var = glm(CMani~1, data = Ltrain_over, family = "binomial")
var2 = glm(CMani~., data = Ltrain_over, family = "binomial")
#using forward Method to select  variable
step(var, scope = list(lower=var, upper=var2), direction = "forward")
LTrain_complete_variable <- glm(CMani ~  DSRI + ACCR + SGI + AQI,
                                data = Ltrain_over,
                                family = "binomial")
summary(LTrain_complete_variable)
```

```
# Deviance for the model
Lt <- summary(LTrain_complete_variable)$deviance

# Predict test data based on model
Tpred_variable = predict.glm(LTrain_complete_variable, newdata = Test_complete, type="response")
Tpred_variable

#Plotting ROC Curve
roc_pred = prediction(Tpred_variable, Test_complete$CMani)
roc_perf = performance(roc_pred, "tpr", "fpr")
plot(roc_perf, col = "blue")
```




```
#calculating Optimal Cutoff
opt.cut = function(roc_perf, roc_pred){
  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]])
  }, roc_perf@x.values, roc_perf@y.values, roc_pred@cutoffs)}
print(opt.cut(roc_perf, roc_pred))
#sensitivity 0.9000000
```

```
#specificity 0.8146552

#cutoff      0.2960608


#Using the cutoff Point to Plot Confusion Matrix

Tpred_variable$CMani = ifelse(Tpred_variable> 0.2960608,1,0)


ptab<-table(Tpred_variable$CMani, Test_complete$CMani, dnn = c("Predicted","Actual"))

ptab

confusionMatrix(ptab,positive = "1")

# We Get 95% Accuracy
```

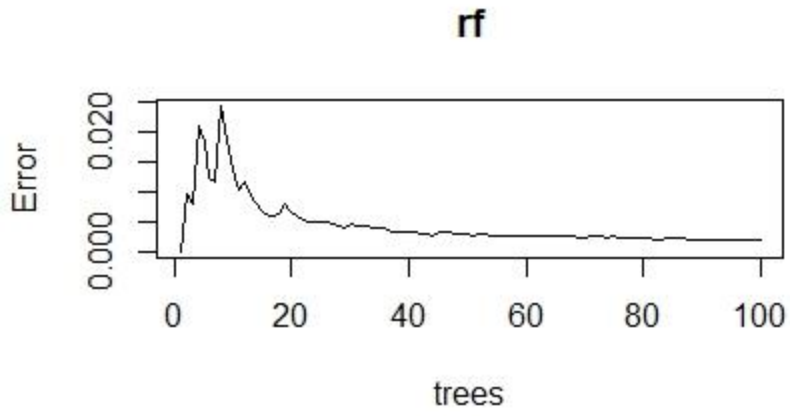
```
          Actual
Predicted   0   1
        0 189   2
        1  43   8

               Accuracy : 0.814
                 95% CI : (0.7592, 0.861)
    No Information Rate : 0.9587
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2075

 Mcnemar's Test P-Value : 2.479e-09

            Sensitivity : 0.80000
            Specificity : 0.81466
         Pos Pred Value : 0.15686
         Neg Pred Value : 0.98953
             Prevalence : 0.04132
         Detection Rate : 0.03306
   Detection Prevalence : 0.21074
      Balanced Accuracy : 0.80733

       'Positive' Class : 1
```

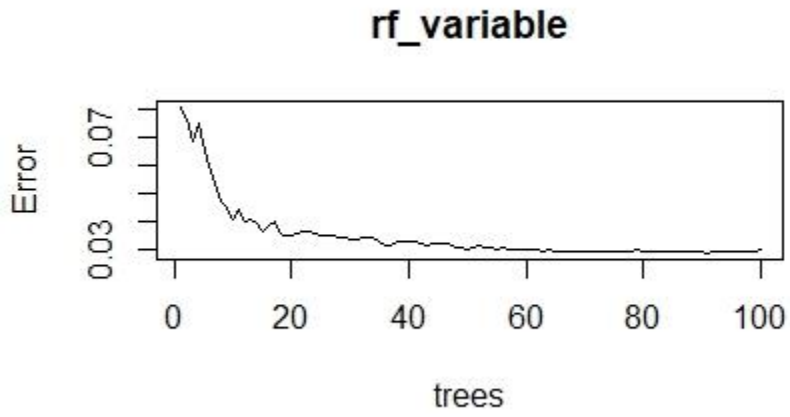
Random Forest Model

We also calculated the Random Forest Model ROC curve with and without variable specification meanwhile getting 95% accuracy.

```
library(randomForest)

install.packages("ROSE")

library(ROSE)


# Sampling Test and Train data

dataSet$`Company ID`<- NULL
```

```
set.seed(1234)
index <- sample(2, nrow(dataSet), replace = TRUE, prob = c(0.65,0.35))
rf_train <- dataSet[index == 1,]
rf_test <- dataSet[index == 2,]

# Balancing the data by Oversampling
over_sample_rf <- ovun.sample(CMani~., data = rf_train, method = "over", N= 248)$data
over_sample_rf
table(over_sample_rf$CMani)
rf = randomForest(CMani~.,
                  data = over_sample_rf, ntree = 100,
                  proximity = TRUE, replace= TRUE,
                  importance = TRUE,
                  mtry = sqrt(ncol(over_sample_rf)))
rf_variable = randomForest(CMani~ DSRI + SGI + ACCR,
            data = over_sample_rf, ntree = 100,
            proximity = TRUE, replace= TRUE,
            importance = TRUE,
            mtry = sqrt(ncol(over_sample_rf)))

print(rf)
plot(rf)
plot(rf_variable)

rf_test_pred <- predict(rf, newdata = rf_test)
rf_test_pred
rf_table <- table(rf_test_pred, newdata = rf_test$CMani)

# Predicting Model Accuracy
confusionMatrix(rf_table, positive = "1")
```





Conclusion:

In conclusion we suggest MCA Technologies Private Limited to choose Random Forest model to predict their earning manipulation because when we are dealing with imbalanced data**, random forest is the suitable choice over Logistic regression.**