# Final_Project_64060

## Contents



### Data Preparation

```
getwd()
```

```
## [1] "C:/Users/Mukht/OneDrive/Desktop/Kent State University/College of Business Admin-Bus. Analytics 
```

```
setwd("C:\\Users\\Mukht\\OneDrive\\Desktop\\Kent State University\\College of Business Admin-Bus. Analy
```

```
FinalProject<-read.csv("MukhtarMLProject.csv")
str(FinalProject)
```

```
## 'data.frame':    250 obs. of  5 variables:
##  $ ï..FDIInflow      : int  7 12 12 2 1 13 12 2 4 5 ...
##  $ Security          : num  0.0825 1.4484 1.1884 -0.6964 0.3228 ...
##  $ EaseOfDoingBus    : num  -1.068 -0.747 0.495 -0.87 0.122 ...
##  $ InvestFacilitation: num  2.8 4.2 4.8 5 4.2 5 5 4.8 3.4 3 ...
##  $ Corruption        : num  3.14 3.86 3.71 2.57 3.86 ...
```

```r
head(FinalProject)
```

```
##   ï..FDIInflow    Security EaseOfDoingBus InvestFacilitation Corruption
## 1            7  0.08251019     -1.0681966                2.8   3.142857
## 2           12  1.44839773     -0.7473753                4.2   3.857143
## 3           12  1.18841274      0.4951372                4.8   3.714286
## 4            2 -0.69644101     -0.8701858                5.0   2.571429
## 5            1  0.32278869      0.1220579                4.2   3.857143
## 6           13  0.54041487      0.2062462                5.0   3.571429
```

```r
library(class)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(lattice)
library(ggplot2)
library(ISLR)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.4     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(e1071)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.1.2
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(esquisse)
```

```
## Warning: package 'esquisse' was built under R version 4.1.2
```

#Plot correlation headmap

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.1.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
ggcorr(FinalProject, label = TRUE, palette = "RdBu", name = "Correlation", hjust = 0.75, label_size =3,
```


## Problem Statement

## Understand3, the complexity of FDI decision factors by analysing their effects and using them to make FDI prediction

```
FinalProject_normalized<-preProcess (FinalProject, method = "range")
FinalProject_normalized = predict(FinalProject_normalized, FinalProject)
summary(FinalProject_normalized)
```

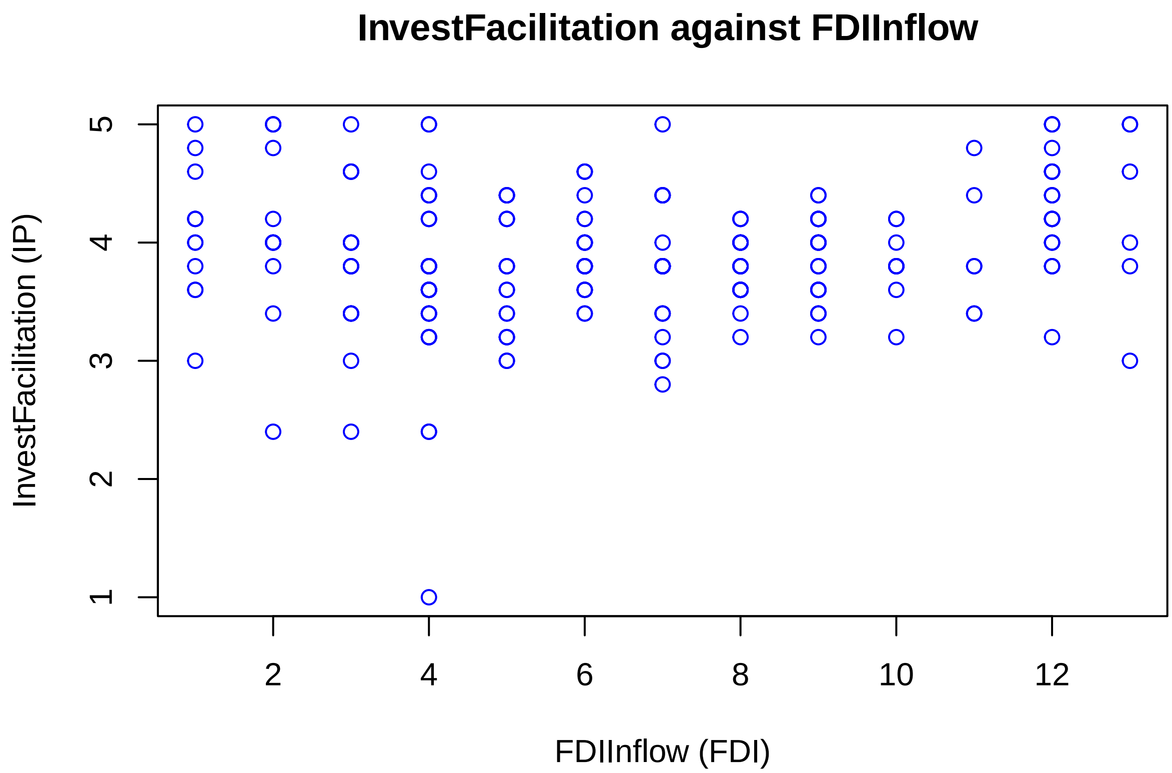
```
##   ï..FDIInflow      Security      EaseOfDoingBus    InvestFacilitation
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.2500   1st Qu.:0.4761   1st Qu.:0.5139   1st Qu.:0.650
##  Median :0.5000   Median :0.6806   Median :0.6980   Median :0.700
##  Mean   :0.4703   Mean   :0.6442   Mean   :0.6674   Mean   :0.723
##  3rd Qu.:0.6667   3rd Qu.:0.8369   3rd Qu.:0.7588   3rd Qu.:0.800
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000
##    Corruption
##  Min.   :0.0000
##  1st Qu.:0.4286
##  Median :0.5714
##  Mean   :0.5460
```

```
##   3rd Qu.:0.6429
##   Max.   :1.0000
```

```r
#Linear Regression
# Creates a linear model for all the variables vs FDI Inflow and displays a plot of the points
Modela = lm(ï..FDIInflow ~ EaseOfDoingBus + InvestFacilitation + FinalProject$Corruption + Security, da
summary(Modela)
```

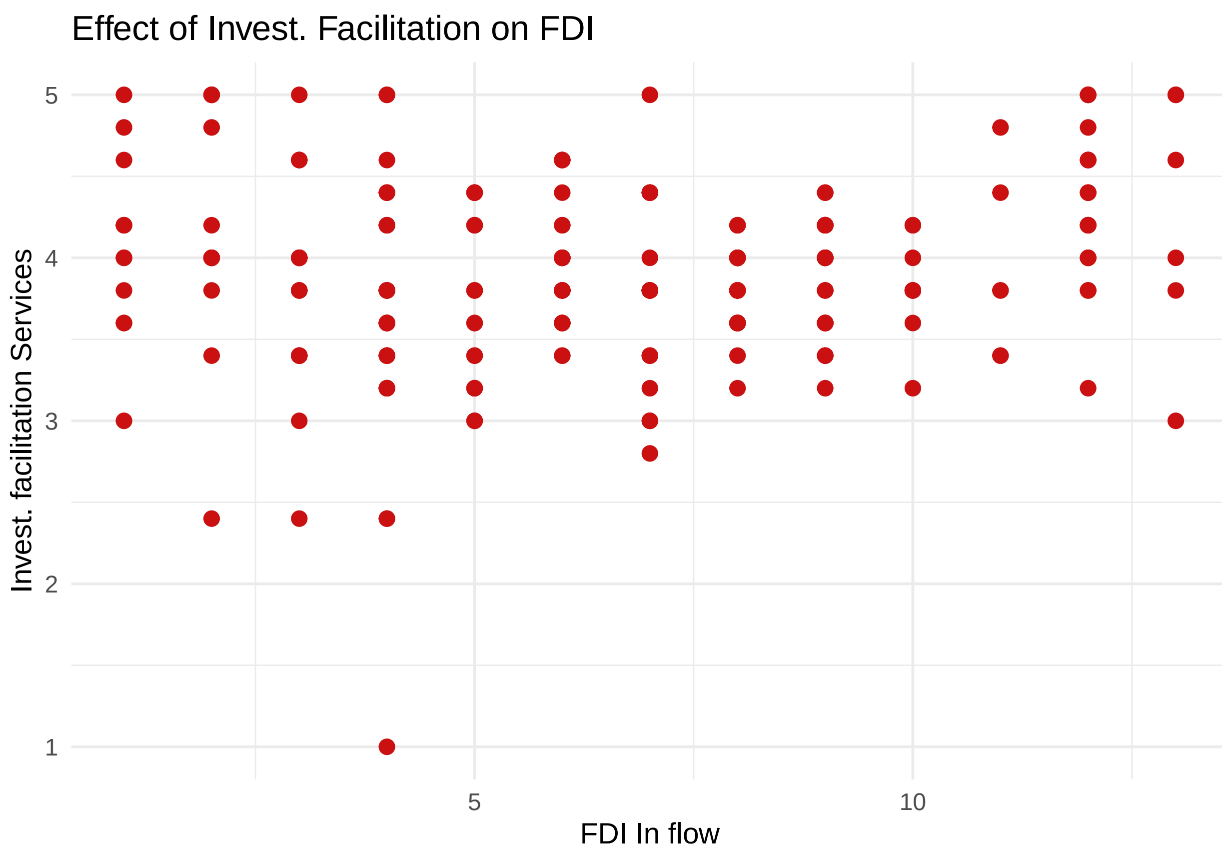
```
##
## Call:
## lm(formula = ï..FDIInflow ~ EaseOfDoingBus + InvestFacilitation +
##     FinalProject$Corruption + Security, data = FinalProject)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4034 -2.3982  0.0278  2.1127  7.0927
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.4920     2.7276   0.180    0.857
## EaseOfDoingBus            0.2119     0.2161   0.981    0.328
## InvestFacilitation        0.8768     0.3774   2.323    0.021 *
## FinalProject$Corruption   0.7494     0.5775   1.298    0.196
## Security                 -0.1548     0.1997  -0.775    0.439
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.182 on 245 degrees of freedom
## Multiple R-squared:  0.03244,    Adjusted R-squared:  0.01664
## F-statistic: 2.053 on 4 and 245 DF,  p-value: 0.08759
```

```r
plot(FinalProject$ï..FDIInflow, FinalProject$InvestFacilitation, xlab = "FDIInflow (FDI)", ylab = "Inves
```




```
esquisser(FinalProject)
```

```
ggplot(FinalProject) +
  aes(x = ï..FDIInflow, y = InvestFacilitation) + geom_point(shape = "circle", size = 2.25, colour = "#(
```




***

---

## We now use Naive Bayes on select variables to predict Foreign Direct Investment Inflows.

We will use the e1070 package.

```
library(caret)
library(ISLR)
library(e1071)
```

## we divide data set into 80% training and 20% testing

```
#Divide data into test and train
FinalProject_Index_Train<-createDataPartition(FinalProject$ï..FDIInflow, p=0.8, list=FALSE)
Train <-FinalProject[FinalProject_Index_Train,]
Test <-FinalProject[-FinalProject_Index_Train,]
summary(Train)
```

```
##   ï..FDIInflow      Security        EaseOfDoingBus     InvestFacilitation
##  Min.   : 1.000   Min.   :-3.04528   Min.   :-3.13918   Min.   :1.00
##  1st Qu.: 4.000   1st Qu.:-0.76467   1st Qu.:-0.72391   1st Qu.:3.60
##  Median : 7.000   Median : 0.15293   Median : 0.11871   Median :3.80
##  Mean   : 6.663   Mean   :-0.03277   Mean   :-0.01463   Mean   :3.87
##  3rd Qu.: 9.000   3rd Qu.: 0.85501   3rd Qu.: 0.41935   3rd Qu.:4.20
##  Max.   :13.000   Max.   : 1.66035   Max.   : 1.50753   Max.   :5.00
##    Corruption
##  Min.   :2.571
##  1st Qu.:3.464
##  Median :3.714
##  Mean   :3.673
##  3rd Qu.:3.857
##  Max.   :4.571
```

```
summary(Test)
```

```
##   ï..FDIInflow      Security       EaseOfDoingBus     InvestFacilitation
##  Min.   : 1.000   Min.   :-2.2252   Min.   :-2.96839   Min.   :3.000
##  1st Qu.: 4.000   1st Qu.:-0.8293   1st Qu.:-0.80094   1st Qu.:3.600
##  Median : 7.000   Median : 0.1625   Median :-0.02989   Median :4.000
##  Mean   : 6.562   Mean   : 0.0648   Mean   :-0.13573   Mean   :3.983
##  3rd Qu.: 8.250   3rd Qu.: 1.0043   3rd Qu.: 0.31326   3rd Qu.:4.200
##  Max.   :12.000   Max.   : 1.5852   Max.   : 1.27770   Max.   :5.000
##    Corruption
##  Min.   :2.571
##  1st Qu.:3.429
##  Median :3.714
##  Mean   :3.625
##  3rd Qu.:3.857
##  Max.   :4.143
```

#Now, run the Naive Bayes classifier model, and predict FDI status on the test set

```
# Build a naïve Bayes classifier
FinalProject_nb_model <-naiveBayes(ï..FDIInflow ~ EaseOfDoingBus + InvestFacilitation + Corruption + Se
FinalProject_nb_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##          1          2          3          4          5          6          7
## 0.04950495 0.03960396 0.07920792 0.14851485 0.06930693 0.10891089 0.09405941
##          8          9         10         11         12         13
## 0.10891089 0.12376238 0.03960396 0.02970297 0.07920792 0.02970297
##
## Conditional probabilities:
##     EaseOfDoingBus
```

```
## Y            [,1]      [,2]
##   1  -0.30027024 0.7135320
##   2  -0.20558517 1.4438026
##   3  -0.16639186 1.0450017
##   4  -0.07188945 1.0075543
##   5   0.40838988 0.9238494
##   6   0.26496896 0.6553708
##   7  -0.29008117 0.9012087
##   8  -0.28828079 1.1568248
##   9   0.19200974 0.8718009
##   10  0.22611897 0.6518544
##   11  0.43386962 0.8237663
##   12 -0.05021067 1.0790939
##   13 -0.26504840 0.5961266
##
##     InvestFacilitation
## Y         [,1]      [,2]
##   1  4.000000 0.5249339
##   2  4.000000 0.7782765
##   3  3.837500 0.6031860
##   4  3.653333 0.7912161
##   5  3.771429 0.4889999
##   6  3.945455 0.3608552
##   7  3.694737 0.4636494
##   8  3.781818 0.2538023
##   9  3.888000 0.3320643
##   10 3.800000 0.3207135
##   11 3.933333 0.5609516
##   12 4.325000 0.5208967
##   13 4.233333 0.7840068
##
##     Corruption
## Y         [,1]      [,2]
##   1  3.800000 0.2446711
##   2  3.410714 0.6363045
##   3  3.696429 0.4201555
##   4  3.614286 0.3542245
##   5  3.775510 0.2949831
##   6  3.720779 0.3648216
##   7  3.503759 0.3242589
##   8  3.746753 0.4180844
##   9  3.731429 0.2745435
##   10 3.607143 0.3030458
##   11 3.690476 0.2102800
##   12 3.696429 0.3720690
##   13 3.642857 0.2347382
##
##     Security
## Y            [,1]      [,2]
##   1   0.23072050 0.7155632
##   2   0.21890802 1.0884330
##   3  -0.42054470 1.3128653
##   4   0.19391240 1.0111003
##   5   0.18806906 1.1063097
```

```
##   6  -0.08549680 0.7974383
##   7  -0.22843763 1.1563954
##   8   0.06919476 0.9855971
##   9  -0.23728942 1.0662863
##   10  0.10035228 0.6081962
##   11 -0.70653062 0.8054158
##   12  0.02937670 1.2257045
##   13  0.19957601 0.8441008
```

```r
summary(FinalProject_nb_model)
```

```
##           Length Class  Mode
## apriori   13     table  numeric
## tables     4     -none- list
## levels    13     -none- character
## isnumeric  4     -none- logical
## call       4     -none- call
```

#The first part of the output above shows the ratios of default (yes) and default (no) in the training set (called a priori probabilities), followed by a table giving for each target class, mean and standard deviation of the (sub-)variable. Also, note that the Naive Bayes algorithm assumes a Normal distribution for the independent variables. In accordance with the rule of the use of categorical predictors (the independent variables have been converted to categorical), we now have the conditional probabilities p(X|Y) for each attribute level given the default status.

Now, use the model on the test set

```r
set.seed(123)
# Predict the default status of test data set
FinalProject_Predicted_Test_labels <-predict(FinalProject_nb_model, Test)
library(gmodels)
```

```
##
## Attaching package: 'gmodels'

## The following object is masked from 'package:pROC':
##
##     ci
```

```r
# Show the confusion matrix of the classifier
CrossTable(x=Test$ï..FDIInflow, y=FinalProject_Predicted_Test_labels, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
```

```
##
## Total Observations in Table:  48
##
##
##                    | FinalProject_Predicted_Test_labels
## Test$ï..FDIInflow |          2 |          4 |          5 |          6 |          7 |          8 |          9
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  1 |          1 |          0 |          0 |          0 |          0 |          0 |          1
##                    |      0.500 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.50
##                    |      0.333 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.07
##                    |      0.021 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.02
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  2 |          1 |          1 |          1 |          1 |          0 |          0 |          0
##                    |      0.200 |      0.200 |      0.200 |      0.200 |      0.000 |      0.000 |      0.00
##                    |      0.333 |      0.167 |      1.000 |      0.125 |      0.000 |      0.000 |      0.00
##                    |      0.021 |      0.021 |      0.021 |      0.021 |      0.000 |      0.000 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  3 |          0 |          1 |          0 |          0 |          0 |          0 |          1
##                    |      0.000 |      0.333 |      0.000 |      0.000 |      0.000 |      0.000 |      0.33
##                    |      0.000 |      0.167 |      0.000 |      0.000 |      0.000 |      0.000 |      0.07
##                    |      0.000 |      0.021 |      0.000 |      0.000 |      0.000 |      0.000 |      0.02
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  4 |          0 |          0 |          0 |          1 |          1 |          1 |          2
##                    |      0.000 |      0.000 |      0.000 |      0.200 |      0.200 |      0.200 |      0.40
##                    |      0.000 |      0.000 |      0.000 |      0.125 |      0.250 |      0.167 |      0.14
##                    |      0.000 |      0.000 |      0.000 |      0.021 |      0.021 |      0.021 |      0.04
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  5 |          1 |          2 |          0 |          0 |          1 |          1 |          0
##                    |      0.200 |      0.400 |      0.000 |      0.000 |      0.200 |      0.200 |      0.00
##                    |      0.333 |      0.333 |      0.000 |      0.000 |      0.250 |      0.167 |      0.00
##                    |      0.021 |      0.042 |      0.000 |      0.000 |      0.021 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  6 |          0 |          0 |          0 |          1 |          1 |          0 |          0
##                    |      0.000 |      0.000 |      0.000 |      0.500 |      0.500 |      0.000 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.125 |      0.250 |      0.000 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.021 |      0.021 |      0.000 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  7 |          0 |          1 |          0 |          0 |          1 |          0 |          2
##                    |      0.000 |      0.167 |      0.000 |      0.000 |      0.167 |      0.000 |      0.33
##                    |      0.000 |      0.167 |      0.000 |      0.000 |      0.250 |      0.000 |      0.14
##                    |      0.000 |      0.021 |      0.000 |      0.000 |      0.021 |      0.000 |      0.04
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  8 |          0 |          0 |          0 |          1 |          0 |          3 |          4
##                    |      0.000 |      0.000 |      0.000 |      0.125 |      0.000 |      0.375 |      0.50
##                    |      0.000 |      0.000 |      0.000 |      0.125 |      0.000 |      0.500 |      0.28
##                    |      0.000 |      0.000 |      0.000 |      0.021 |      0.000 |      0.062 |      0.08
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  9 |          0 |          0 |          0 |          2 |          0 |          0 |          1
##                    |      0.000 |      0.000 |      0.000 |      0.667 |      0.000 |      0.000 |      0.33
##                    |      0.000 |      0.000 |      0.000 |      0.250 |      0.000 |      0.000 |      0.07
##                    |      0.000 |      0.000 |      0.000 |      0.042 |      0.000 |      0.000 |      0.02
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                 10 |          0 |          0 |          0 |          1 |          0 |          0 |          1
##                    |      0.000 |      0.000 |      0.000 |      0.500 |      0.000 |      0.000 |      0.50
```

```
##                   |     0.000 |     0.000 |     0.000 |     0.125 |     0.000 |     0.000 |     0.07
##                   |     0.000 |     0.000 |     0.000 |     0.021 |     0.000 |     0.000 |     0.02
## ------------------|-----------|-----------|-----------|-----------|-----------|-----------|---------
##                12 |         0 |         1 |         0 |         1 |         0 |         1 |         
##                   |     0.000 |     0.143 |     0.000 |     0.143 |     0.000 |     0.143 |     0.28
##                   |     0.000 |     0.167 |     0.000 |     0.125 |     0.000 |     0.167 |     0.14
##                   |     0.000 |     0.021 |     0.000 |     0.021 |     0.000 |     0.021 |     0.04
## ------------------|-----------|-----------|-----------|-----------|-----------|-----------|---------
##      Column Total |         3 |         6 |         1 |         8 |         4 |         6 |        1
##                   |     0.062 |     0.125 |     0.021 |     0.167 |     0.083 |     0.125 |     0.29
## ------------------|-----------|-----------|-----------|-----------|-----------|-----------|---------
##
##
```

#Our results indicate that we mis-classified a total of X cases. X as False Positives, and X as False Negatives.

---

#It is sometimes useful to output the raw prediction probabilities rather than the predicted class. To do that, we use the raw option in the model.

```r
FinalProject_nb_model <- naiveBayes(ï..FDIInflow ~ EaseOfDoingBus + InvestFacilitation + Corruption + S
```

```r
#Make predictions and return probability of each class
FinalProject_Predicted_Test_labels <-predict(FinalProject_nb_model,Test, type = "raw")
#show the first few values
head(FinalProject_Predicted_Test_labels)
```

```
##                 1          2          3          4            5           6
## [1,] 1.521914e-05 0.69414227 0.09052915 0.09353182 0.0001582876 0.003030543
## [2,] 3.035470e-02 0.04461605 0.07335099 0.13092896 0.1978481926 0.123562523
## [3,] 3.509910e-02 0.01504716 0.03584012 0.10058550 0.0909301362 0.219344713
## [4,] 3.563917e-02 0.02250682 0.04785579 0.10258168 0.1809680922 0.159132145
## [5,] 4.498815e-03 0.06978865 0.06364104 0.18657268 0.0556801454 0.181261108
## [6,] 2.295755e-04 0.42008221 0.10374540 0.17150416 0.0025488703 0.028874318
##               7            8            9           10           11        12
## [1,] 0.01612071 1.319611e-05 3.637884e-05 3.305501e-05 1.481988e-06 0.1021497
## [2,] 0.01324486 2.867580e-02 1.240216e-01 6.911072e-03 3.376322e-03 0.2205693
## [3,] 0.04387863 4.826621e-02 1.928601e-01 8.919988e-02 4.079269e-02 0.0756084
## [4,] 0.01950271 7.458167e-02 2.048791e-01 2.551990e-02 9.784032e-03 0.1136775
## [5,] 0.09115523 7.600574e-02 8.332303e-02 5.802816e-02 2.593149e-03 0.1231007
## [6,] 0.10593413 2.769379e-02 2.649594e-03 2.869177e-03 2.428194e-06 0.1327659
##                13
## [1,] 0.0002381556
## [2,] 0.0025396872
## [3,] 0.0125473759
## [4,] 0.0033713954
## [5,] 0.0043515765
## [6,] 0.0011004738
```

---

```
set.seed(123)
data <- data.frame(FinalProject = sample(c("True","False"), 250, replace = TRUE),
                   FinalProject_Predicted_Test_labels = sample(c("True","False"), 250, replace = TRUE)
                   )
table(data$FinalProject_Predicted_Test_labels, data$FinalProject)
```

```
##
##         False True
##   False    58   54
##   True     69   69
```

#The confusionMatrix function is very helpful as not only does it display a confusion matrix, it calculates many relevant statistics alongside:

```
set.seed(123)
data <- data.frame(FinalProject = sample(c("True","False"), 250, replace = TRUE),
FinalProject_Predicted_Test_labels = sample(c("True","False"), 250, replace = TRUE)
)
library(caret)
confusionMatrix(as.factor(data$FinalProject_Predicted_Test_labels), as.factor(data$FinalProject), posit
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False    58   54
##      True     69   69
##
##                Accuracy : 0.508
##                  95% CI : (0.4443, 0.5716)
##     No Information Rate : 0.508
##     P-Value [Acc > NIR] : 0.5253
##
##                   Kappa : 0.0176
##
##  Mcnemar's Test P-Value : 0.2068
##
##             Sensitivity : 0.5610
##             Specificity : 0.4567
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5179
##              Prevalence : 0.4920
##          Detection Rate : 0.2760
##    Detection Prevalence : 0.5520
##       Balanced Accuracy : 0.5088
##
##        'Positive' Class : True
##
```

## ROC Curves

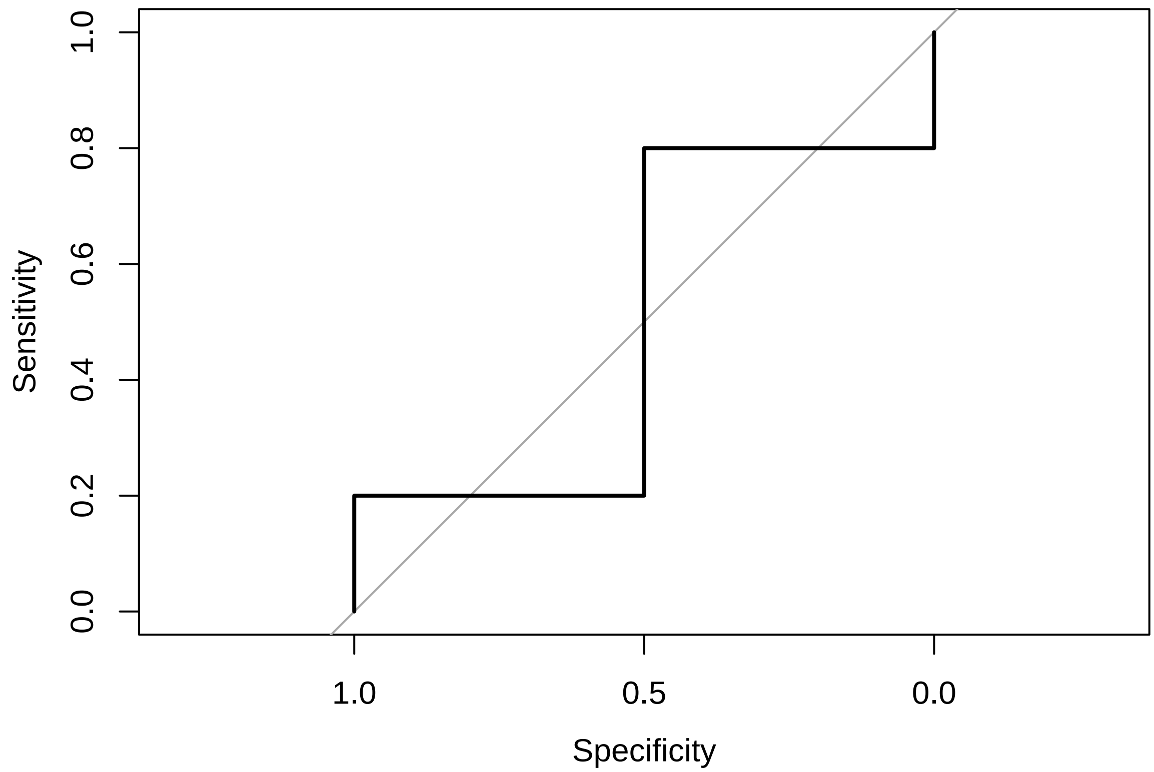
We can now output the ROC curves. we should emember that ROC curves plot sensitivity (true positive rate) versus (1 - specificity), which is (1 - TNR) or false positive rate. See here for more details

```r
# install.packages("pROC") # install if necessary
library(pROC)
#Passing the column of the predicted probabilities
#That column contains the probability associate to 'yes'
roc(Test$ï..FDIInflow, FinalProject_Predicted_Test_labels[, 2])
```

```
## Warning in roc.default(Test$ï..FDIInflow, FinalProject_Predicted_Test_labels[, :
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead

## Setting levels: control = 1, case = 2

## Setting direction: controls > cases

##
## Call:
## roc.default(response = Test$ï..FDIInflow, predictor = FinalProject_Predicted_Test_labels[,     2])
##
## Data: FinalProject_Predicted_Test_labels[, 2] in 2 controls (Test$ï..FDIInflow 1) > 5 cases (Test$ï.
## Area under the curve: 0.5
```

```r
plot.roc(Test$ï..FDIInflow, FinalProject_Predicted_Test_labels[, 2])
```

```
## Warning in roc.default(x, predictor, plot = TRUE, ...): 'response' has more
## than two levels. Consider setting 'levels' explicitly or using 'multiclass.roc'
## instead

## Setting levels: control = 1, case = 2
## Setting direction: controls > cases
```




The AUC is 1. The ROC curve is also plotted, though note that the X-Axis is Specificity (True Negative Rate), rather than 1-Specificity (False Positive Rate). This function can also be thought of as a plot of the FDI as a function of the Type I Error of the decision rule.

---

## Box-Cox Transformation

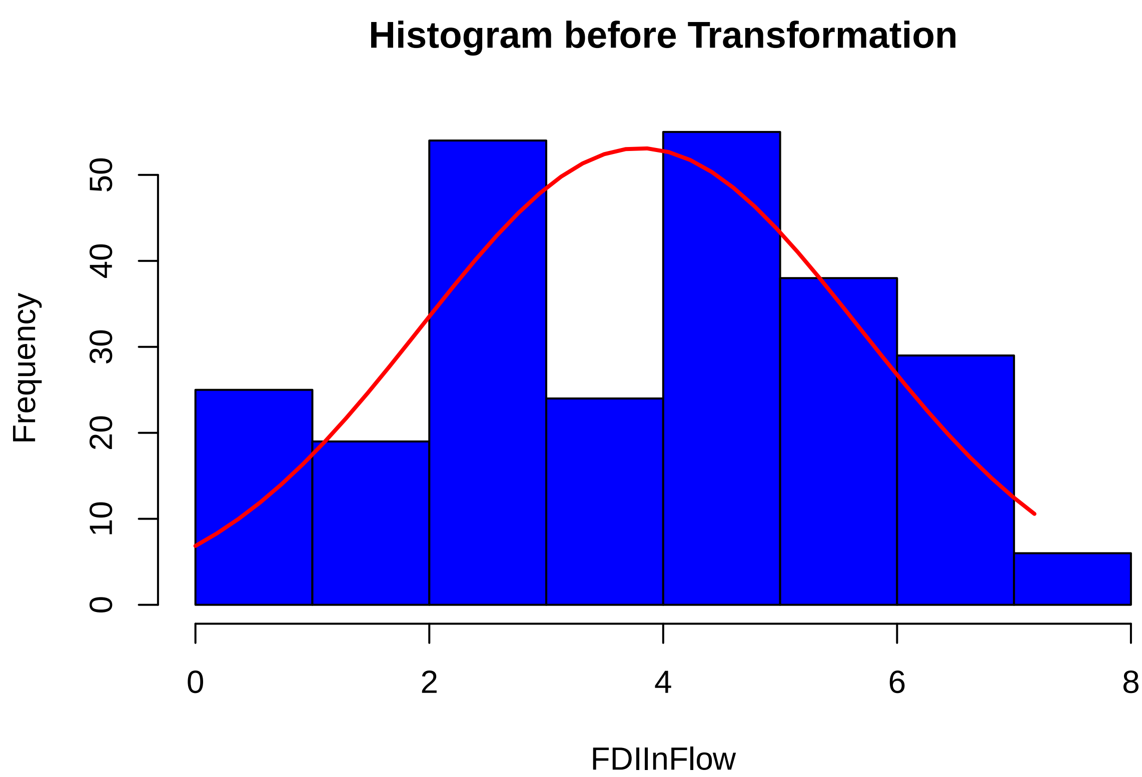
We first illustrate the transformation of data using the Box-Cox transformation approach

```
library(ISLR)
library(caret)
#Create a Box-Cox Transformation Model
FinalProject_Box_Cox_Transform<-preProcess(FinalProject,method = "BoxCox")
FinalProject_Box_Cox_Transform
```

```
## Created from 250 samples and 3 variables
##
## Pre-processing:
##   - Box-Cox transformation (3)
##   - ignored (0)
##
## Lambda estimates for Box-Cox transformation:
## 0.7, 1.8, 2
```

Now, we apply the transformation

```
FinalProject_Transformed=predict(FinalProject_Box_Cox_Transform, FinalProject)
y <-FinalProject_Transformed$ï..FDIInflow
h<-hist(y, breaks=10, col="blue", xlab="FDIInFlow",
   main="Histogram before Transformation")
xfit<-seq(min(y),max(y),length=40)
yfit<-dnorm(xfit,mean=mean(y),sd=sd(y))
yfit <- yfit*diff(h$mids[1:2])*length(y)
lines(xfit, yfit, col="red", lwd=2)
```




***

### Hypertuning

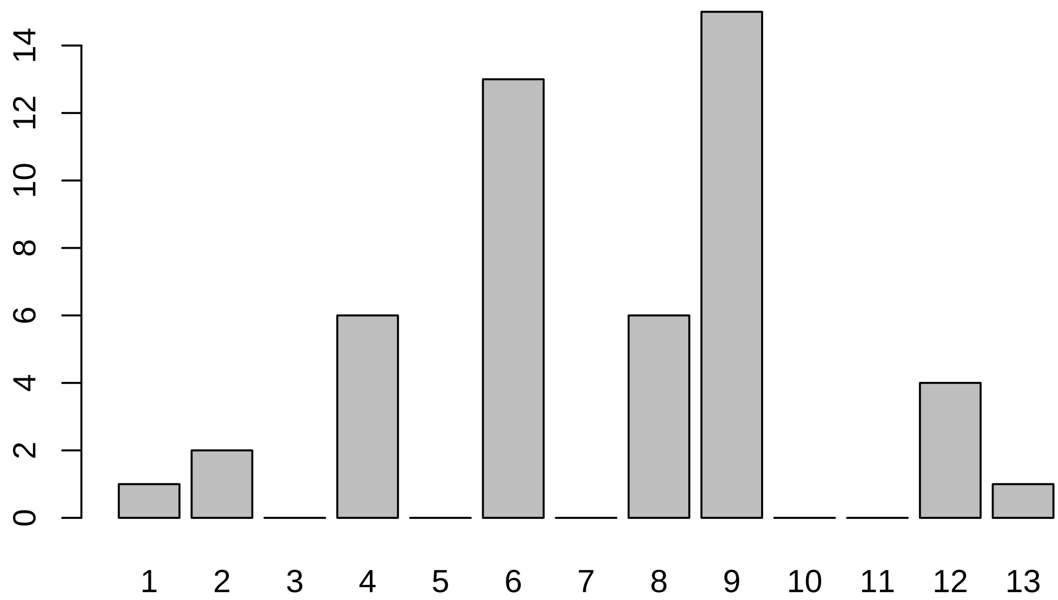
```
library(caret)
library(ISLR)
```

```
set.seed(123)
#Divide data into test and train
Index_Train<-createDataPartition(FinalProject$ï..FDIInflow, p=0.8, list= FALSE)
Train <-FinalProject[Index_Train,]
Test  <-FinalProject[-Index_Train,]
```

```r
nb_model <-train(ï..FDIInflow ~ EaseOfDoingBus + InvestFacilitation + Corruption + Security, data = Trai
# Predict the default status of test dataset
Predicted_Test_labels <-predict(FinalProject_nb_model,Test)
summary(Predicted_Test_labels)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13
##  1  2  0  6  0 13  0  6 15  0  0  4  1
```

```r
plot(Predicted_Test_labels)
```




```r
library(gmodels)
# Show the confusion matrix of the classifier
CrossTable(x=Test$ï..FDIInflow,y=Predicted_Test_labels, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
```

```
##
## Total Observations in Table:  48
##
##
##                    | Predicted_Test_labels
## Test$ï..FDIInflow |          1 |          2 |          4 |          6 |          8 |          9 |         1
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  1 |          0 |          1 |          0 |          2 |          0 |          0 |
##                    |      0.000 |      0.333 |      0.000 |      0.667 |      0.000 |      0.000 |      0.00
##                    |      0.000 |      0.500 |      0.000 |      0.154 |      0.000 |      0.000 |      0.00
##                    |      0.000 |      0.021 |      0.000 |      0.042 |      0.000 |      0.000 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  2 |          0 |          0 |          0 |          2 |          0 |          0 |
##                    |      0.000 |      0.000 |      0.000 |      0.667 |      0.000 |      0.000 |      0.33
##                    |      0.000 |      0.000 |      0.000 |      0.154 |      0.000 |      0.000 |      0.25
##                    |      0.000 |      0.000 |      0.000 |      0.042 |      0.000 |      0.000 |      0.02
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  3 |          0 |          0 |          1 |          1 |          0 |          1 |
##                    |      0.000 |      0.000 |      0.333 |      0.333 |      0.000 |      0.333 |      0.00
##                    |      0.000 |      0.000 |      0.167 |      0.077 |      0.000 |      0.067 |      0.00
##                    |      0.000 |      0.000 |      0.021 |      0.021 |      0.000 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  4 |          0 |          0 |          1 |          2 |          2 |          1 |
##                    |      0.000 |      0.000 |      0.167 |      0.333 |      0.333 |      0.167 |      0.00
##                    |      0.000 |      0.000 |      0.167 |      0.154 |      0.333 |      0.067 |      0.00
##                    |      0.000 |      0.000 |      0.021 |      0.042 |      0.042 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  5 |          1 |          1 |          1 |          0 |          0 |          1 |
##                    |      0.250 |      0.250 |      0.250 |      0.000 |      0.000 |      0.250 |      0.00
##                    |      1.000 |      0.500 |      0.167 |      0.000 |      0.000 |      0.067 |      0.00
##                    |      0.021 |      0.021 |      0.021 |      0.000 |      0.000 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  6 |          0 |          0 |          0 |          1 |          0 |          3 |
##                    |      0.000 |      0.000 |      0.000 |      0.250 |      0.000 |      0.750 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.077 |      0.000 |      0.200 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.021 |      0.000 |      0.062 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  7 |          0 |          0 |          2 |          1 |          0 |          1 |
##                    |      0.000 |      0.000 |      0.400 |      0.200 |      0.000 |      0.200 |      0.20
##                    |      0.000 |      0.000 |      0.333 |      0.077 |      0.000 |      0.067 |      0.25
##                    |      0.000 |      0.000 |      0.042 |      0.021 |      0.000 |      0.021 |      0.02
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  8 |          0 |          0 |          1 |          1 |          2 |          1 |
##                    |      0.000 |      0.000 |      0.200 |      0.200 |      0.400 |      0.200 |      0.00
##                    |      0.000 |      0.000 |      0.167 |      0.077 |      0.333 |      0.067 |      0.00
##                    |      0.000 |      0.000 |      0.021 |      0.021 |      0.042 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                  9 |          0 |          0 |          0 |          2 |          1 |          3 |
##                    |      0.000 |      0.000 |      0.000 |      0.333 |      0.167 |      0.500 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.154 |      0.167 |      0.200 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.042 |      0.021 |      0.062 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                 10 |          0 |          0 |          0 |          1 |          0 |          1 |
##                    |      0.000 |      0.000 |      0.000 |      0.500 |      0.000 |      0.500 |      0.00
```

```
##                    |      0.000 |      0.000 |      0.000 |      0.077 |      0.000 |      0.067 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.021 |      0.000 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                 11 |          0 |          0 |          0 |          0 |          0 |          1 |
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      1.000 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.067 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.021 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                 12 |          0 |          0 |          0 |          0 |          1 |          2 |
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.200 |      0.400 |      0.40
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.167 |      0.133 |      0.50
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.021 |      0.042 |      0.04
## -------------------|------------|------------|------------|------------|------------|------------|----------
##                 13 |          0 |          0 |          0 |          0 |          0 |          0 |
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.00
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.000 |      0.00
## -------------------|------------|------------|------------|------------|------------|------------|----------
##       Column Total |          1 |          2 |          6 |         13 |          6 |         15 |
##                    |      0.021 |      0.042 |      0.125 |      0.271 |      0.125 |      0.312 |      0.08
## -------------------|------------|------------|------------|------------|------------|------------|----------
##
##
```

```r
set.seed(123)
data <- data.frame(FinalProject = sample(c("True","False"), 250, replace = TRUE),
Predicted_Test_labels = sample(c("True","False"), 250, replace = TRUE)
)
library(caret)
confusionMatrix(as.factor(data$Predicted_Test_labels), as.factor(data$FinalProject), positive = "True")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction False True
##      False    58   54
##      True     69   69
##
##                Accuracy : 0.508
##                  95% CI : (0.4443, 0.5716)
##     No Information Rate : 0.508
##     P-Value [Acc > NIR] : 0.5253
##
##                   Kappa : 0.0176
##
##  Mcnemar's Test P-Value : 0.2068
##
##             Sensitivity : 0.5610
##             Specificity : 0.4567
##          Pos Pred Value : 0.5000
##          Neg Pred Value : 0.5179
##              Prevalence : 0.4920
##          Detection Rate : 0.2760
##    Detection Prevalence : 0.5520
```

```
##       Balanced Accuracy : 0.5088
##
##        'Positive' Class : True
##
```

```r
#FDI Decision Tree
library(rpart)
library(ISLR)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.1.2
```

```r
library(datasets)
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.1.2
```

```r
library(party)
```

```
## Warning: package 'party' was built under R version 4.1.2

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 4.1.2

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 4.1.2

##
## Attaching package: 'strucchange'

## The following object is masked from 'package:stringr':
##
##     boundary
```

```r
library(dplyr)
library(magrittr)
```

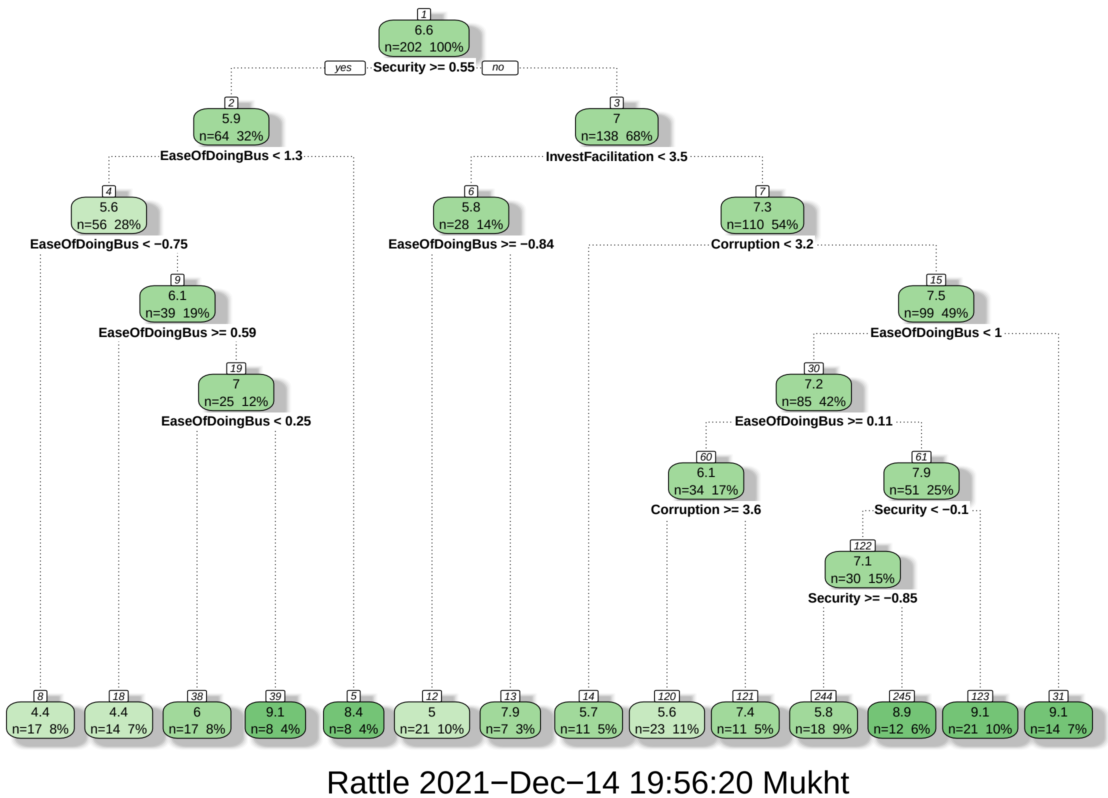
```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
# Decision Tree
library(rpart)
library(tidyverse)
library(rpart.plot)
```

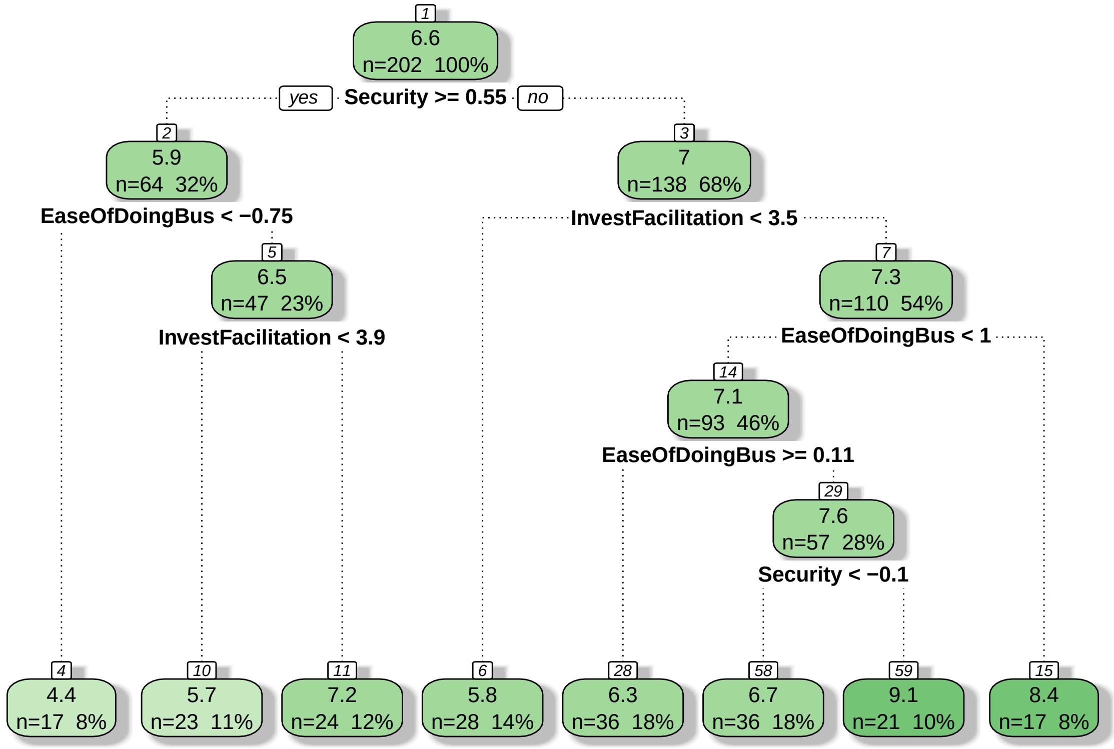
```r
# Tree plot
FDI<- rpart(ï..FDIInflow ~ InvestFacilitation, data = Train) # only one independent variable
FDI<- rpart(ï..FDIInflow ~ ., data = Train) # Except one dependent variable, others are all independent
```

```r
# Predict Tree
PredictFDI<-predict(FDI, newdata = Test)
```

```r
fancyRpartPlot(FDI)
```




```
# Tree with Minimum Observation
PredictFDI<-rpart(ï..FDIInflow ~ EaseOfDoingBus + InvestFacilitation + Corruption + Security, data = Tr
fancyRpartPlot(PredictFDI)
```




Rattle 2021−Dec−14 19:56:21 Mukht

## Make a prediction; Predict FDI In flow

```
library(rpart)
library(rpart.plot)
PredictFDIUnknown<-predict(PredictFDI, Test)
table_FDI <-table(Test$ï..FDIInflow, PredictFDIUnknown)
summary(table_FDI)
```

```
## Number of cases in table: 48
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 82.96, df = 84, p-value = 0.5115
##  Chi-squared approximation may be incorrect
```

```
table_FDI
```

```
##     PredictFDIUnknown
##      4.41176470588235 5.65217391304348 5.75 6.30555555555556 6.72222222222222
##    1                1                0    0                2                0
##    2                0                0    0                0                1
##    3                0                0    1                0                1
##    4                0                0    1                2                3
##    5                1                0    1                1                0
##    6                0                0    0                2                1
##    7                0                1    1                0                1
##    8                0                0    1                0                0
```

```
##    9                  0                 0    0                 1                 4
##    10                 0                 1    0                 1                 0
##    11                 0                 0    1                 0                 0
##    12                 0                 1    0                 0                 2
##    13                 0                 0    0                 1                 0
##      PredictFDIUnknown
##       7.25 8.35294117647059 9.09523809523809
##    1     0                0                0
##    2     0                1                1
##    3     0                0                1
##    4     0                0                0
##    5     0                0                1
##    6     0                1                0
##    7     0                1                1
##    8     0                1                3
##    9     1                0                0
##    10    0                0                0
##    11    0                0                0
##    12    1                0                1
##    13    0                0                0
```

```
#accuracy test
accuracy_Test <- sum(diag(table_FDI)) / sum(table_FDI)
print(paste('Accuracy for test', accuracy_Test))
```

```
## [1] "Accuracy for test 0.166666666666667"
```

#performance measurement- confusion matrix

```
accuracy_tune <- function(fit) {
    PredictFDIUnknown <- predict(FDI, Test, type = 'class')
    table_FDI <- table(FinalProject$ï..FDIInflow, PredictFDIUnknown)
    accuracy_Test <- sum(diag(table_FDI)) / sum(table_FDI)
    accuracy_Test
}
```

#We tried to tune the parameters and see if we could improve the model over the default value. As a reminder, you need to get an accuracy higher than 0.78

```
control <- rpart.control(minsplit = 4,
    minbucket = round(5 / 3),
    maxdepth = 3,
    cp = 0)
tune_fit<-rpart(ï..FDIInflow ~., data = Train, method = "class", control = control)
summary(tune_fit)
```

```
## Call:
## rpart(formula = ï..FDIInflow ~ ., data = Train, method = "class",
##     control = control)
##   n= 202
##
##             CP nsplit rel error    xerror       xstd
```

```
## 1 0.02890173      0 1.0000000 1.000000 0.02880715
## 2 0.01445087      2 0.9421965 1.046243 0.02507420
## 3 0.01156069      4 0.9132948 1.046243 0.02507420
## 4 0.00000000      7 0.8786127 1.034682 0.02609568
##
## Variable importance
##          Security     EaseOfDoingBus         Corruption InvestFacilitation
##                30                 25                 24                 21
##
## Node number 1: 202 observations,    complexity param=0.02890173
##   predicted class=4   expected loss=0.8564356  P(node) =1
##     class counts:     9    10    16    29    15    20    20    25    22     8     5    18     5
##    probabilities: 0.045 0.050 0.079 0.144 0.074 0.099 0.099 0.124 0.109 0.040 0.025 0.089 0.025
##   left son=2 (176 obs) right son=3 (26 obs)
##   Primary splits:
##       InvestFacilitation < 4.5         to the left,  improve=2.545489, (0 missing)
##       Security           < 0.5538103   to the right, improve=2.141005, (0 missing)
##       Corruption         < 2.785714    to the left,  improve=2.069528, (0 missing)
##       EaseOfDoingBus     < 0.2855935   to the left,  improve=1.768244, (0 missing)
##   Surrogate splits:
##       Corruption < 2.642857    to the right, agree=0.881, adj=0.077, (0 split)
##
## Node number 2: 176 observations,    complexity param=0.02890173
##   predicted class=4   expected loss=0.8522727  P(node) =0.8712871
##     class counts:     7     7    12    26    15    17    19    25    22     8     4    11     3
##    probabilities: 0.040 0.040 0.068 0.148 0.085 0.097 0.108 0.142 0.125 0.045 0.023 0.062 0.017
##   left son=4 (20 obs) right son=5 (156 obs)
##   Primary splits:
##       InvestFacilitation < 3.3         to the left,  improve=2.066900, (0 missing)
##       EaseOfDoingBus     < -0.01131459 to the right, improve=1.963858, (0 missing)
##       Security           < 0.5538103   to the right, improve=1.743861, (0 missing)
##       Corruption         < 3.642857    to the left,  improve=1.188560, (0 missing)
##
## Node number 3: 26 observations,    complexity param=0.01445087
##   predicted class=12  expected loss=0.7307692  P(node) =0.1287129
##     class counts:     2     3     4     3     0     3     1     0     0     0     1     7     2
##    probabilities: 0.077 0.115 0.154 0.115 0.000 0.115 0.038 0.000 0.000 0.000 0.038 0.269 0.077
##   left son=6 (9 obs) right son=7 (17 obs)
##   Primary splits:
##       EaseOfDoingBus     < -0.7376851  to the left,  improve=2.233786, (0 missing)
##       Corruption         < 2.857143    to the left,  improve=1.993590, (0 missing)
##       Security           < 1.170145    to the left,  improve=1.776923, (0 missing)
##       InvestFacilitation < 4.9         to the right, improve=1.082984, (0 missing)
##   Surrogate splits:
##       Corruption < 3.214286    to the left,  agree=0.769, adj=0.333, (0 split)
##       Security   < -1.164044   to the left,  agree=0.692, adj=0.111, (0 split)
##
## Node number 4: 20 observations,    complexity param=0.01156069
##   predicted class=4   expected loss=0.65  P(node) =0.0990099
##     class counts:     1     1     1     7     4     0     2     0     1     1     0     1     1
##    probabilities: 0.050 0.050 0.050 0.350 0.200 0.000 0.100 0.000 0.050 0.050 0.000 0.050 0.050
##   left son=8 (13 obs) right son=9 (7 obs)
##   Primary splits:
##       Corruption         < 3.785714    to the left,  improve=1.870330, (0 missing)
```

```
##      Security           < -0.9054582  to the left,  improve=1.755556, (0 missing)
##      EaseOfDoingBus     < 0.08610139  to the right, improve=1.331313, (0 missing)
##      InvestFacilitation < 2.7         to the left,  improve=1.133333, (0 missing)
##  Surrogate splits:
##      Security       < 0.9845555   to the left,  agree=0.75, adj=0.286, (0 split)
##      EaseOfDoingBus < -0.8286571  to the right, agree=0.70, adj=0.143, (0 split)
##
## Node number 5: 156 observations,    complexity param=0.01156069
##  predicted class=8   expected loss=0.8397436  P(node) =0.7722772
##    class counts:     6     6    11    19    11    17    17    25    21     7     4    10     2
##   probabilities: 0.038 0.038 0.071 0.122 0.071 0.109 0.109 0.160 0.135 0.045 0.026 0.064 0.013
##  left son=10 (45 obs) right son=11 (111 obs)
##  Primary splits:
##      Security           < 0.5538103   to the right, improve=2.125757, (0 missing)
##      EaseOfDoingBus     < 0.2840279   to the left,  improve=1.844490, (0 missing)
##      InvestFacilitation < 4.1         to the left,  improve=1.722112, (0 missing)
##      Corruption         < 3.642857    to the left,  improve=1.336114, (0 missing)
##  Surrogate splits:
##      EaseOfDoingBus < 1.235788    to the right, agree=0.731, adj=0.067, (0 split)
##      Corruption     < 3.071429    to the left,  agree=0.724, adj=0.044, (0 split)
##
## Node number 6: 9 observations,    complexity param=0.01445087
##  predicted class=2   expected loss=0.6666667  P(node) =0.04455446
##    class counts:     1     3     0     3     0     0     0     0     0     0     0     1     1
##   probabilities: 0.111 0.333 0.000 0.333 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.111 0.111
##  left son=12 (3 obs) right son=13 (6 obs)
##  Primary splits:
##      EaseOfDoingBus     < -0.8043081  to the right, improve=2.666667, (0 missing)
##      Corruption         < 2.857143    to the left,  improve=1.523810, (0 missing)
##      Security           < 0.3176438   to the right, improve=1.366667, (0 missing)
##      InvestFacilitation < 4.9         to the right, improve=1.266667, (0 missing)
##  Surrogate splits:
##      Security   < 0.3176438   to the right, agree=0.778, adj=0.333, (0 split)
##      Corruption < 3.642857    to the right, agree=0.778, adj=0.333, (0 split)
##
## Node number 7: 17 observations,    complexity param=0.01156069
##  predicted class=12  expected loss=0.6470588  P(node) =0.08415842
##    class counts:     1     0     4     0     0     3     1     0     0     0     1     6     1
##   probabilities: 0.059 0.000 0.235 0.000 0.000 0.176 0.059 0.000 0.000 0.000 0.059 0.353 0.059
##  left son=14 (13 obs) right son=15 (4 obs)
##  Primary splits:
##      Security           < 1.112083    to the left,  improve=2.7149320, (0 missing)
##      InvestFacilitation < 4.7         to the left,  improve=1.2320260, (0 missing)
##      Corruption         < 4.071429    to the left,  improve=1.1764710, (0 missing)
##      EaseOfDoingBus     < 0.1551669   to the right, improve=0.7193277, (0 missing)
##  Surrogate splits:
##      Corruption < 4.071429    to the left,  agree=0.882, adj=0.5, (0 split)
##
## Node number 8: 13 observations
##  predicted class=4   expected loss=0.4615385  P(node) =0.06435644
##    class counts:     1     0     1     7     2     0     1     0     0     0     0     1     0
##   probabilities: 0.077 0.000 0.077 0.538 0.154 0.000 0.077 0.000 0.000 0.000 0.000 0.077 0.000
##
## Node number 9: 7 observations
```

```
##   predicted class=5   expected loss=0.7142857  P(node) =0.03465347
##     class counts:     0     1     0     0     2     0     1     0     1     1     0     0     1
##    probabilities: 0.000 0.143 0.000 0.000 0.286 0.000 0.143 0.000 0.143 0.143 0.000 0.000 0.143
##
## Node number 10: 45 observations
##   predicted class=4   expected loss=0.7777778  P(node) =0.2227723
##     class counts:     1     4     3    10     7     4     1     8     4     0     0     2     1
##    probabilities: 0.022 0.089 0.067 0.222 0.156 0.089 0.022 0.178 0.089 0.000 0.000 0.044 0.022
##
## Node number 11: 111 observations
##   predicted class=8   expected loss=0.8468468  P(node) =0.549505
##     class counts:     5     2     8     9     4    13    16    17    17     7     4     8     1
##    probabilities: 0.045 0.018 0.072 0.081 0.036 0.117 0.144 0.153 0.153 0.063 0.036 0.072 0.009
##
## Node number 12: 3 observations
##   predicted class=4   expected loss=0  P(node) =0.01485149
##     class counts:     0     0     0     3     0     0     0     0     0     0     0     0     0
##    probabilities: 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
##
## Node number 13: 6 observations
##   predicted class=2   expected loss=0.5  P(node) =0.02970297
##     class counts:     1     3     0     0     0     0     0     0     0     0     0     1     1
##    probabilities: 0.167 0.500 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.167 0.167
##
## Node number 14: 13 observations
##   predicted class=3   expected loss=0.6923077  P(node) =0.06435644
##     class counts:     1     0     4     0     0     3     1     0     0     0     1     2     1
##    probabilities: 0.077 0.000 0.308 0.000 0.000 0.231 0.077 0.000 0.000 0.000 0.077 0.154 0.077
##
## Node number 15: 4 observations
##   predicted class=12  expected loss=0  P(node) =0.01980198
##     class counts:     0     0     0     0     0     0     0     0     0     0     0     4     0
##    probabilities: 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000
```