DATA MINING - CS7202
PROJECT

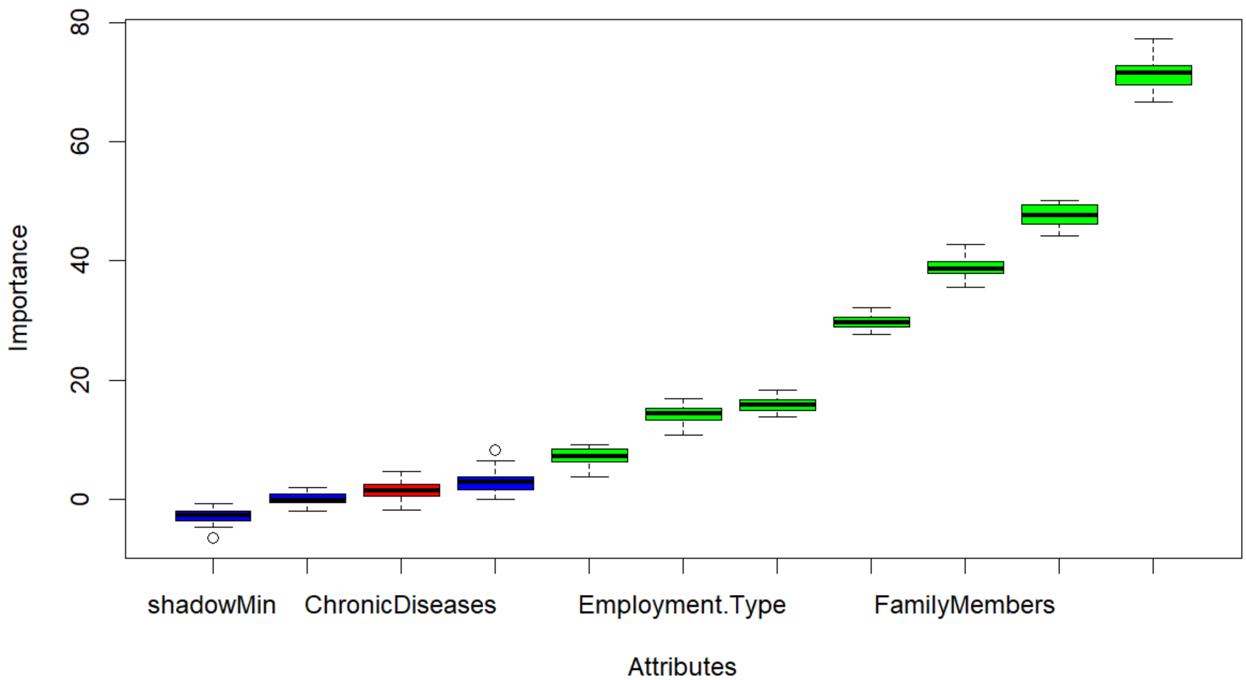
INTERPRETATIONS AND PLOTS/GRAPHS

1. The dataset chosen is from Kaggle named “TravelInsurancePrediction.csv “ containing 9 columns and 1987 observations.

```
> summary(ds)
       X               Age        Employment.Type    GraduateOrNot
 Min.   :   0.0   Min.   :25.00   Length:1987        Length:1987
 1st Qu.: 496.5   1st Qu.:28.00   Class :character   Class :character
 Median : 993.0   Median :29.00   Mode  :character   Mode  :character
 Mean   : 993.0   Mean   :29.65
 3rd Qu.:1489.5   3rd Qu.:32.00
 Max.   :1986.0   Max.   :35.00
  AnnualIncome      FamilyMembers   ChronicDiseases  FrequentFlyer
 Min.   : 300000   Min.   :2.000   Min.   :0.0000   Length:1987
 1st Qu.: 600000   1st Qu.:4.000   1st Qu.:0.0000   Class :character
 Median : 900000   Median :5.000   Median :0.0000   Mode  :character
 Mean   : 932763   Mean   :4.753   Mean   :0.2778
 3rd Qu.:1250000   3rd Qu.:6.000   3rd Qu.:1.0000
 Max.   :1800000   Max.   :9.000   Max.   :1.0000
 EverTravelledAbroad TravelInsurance
 Length:1987         Min.   :0.0000
 Class :character    1st Qu.:0.0000
 Mode  :character    Median :0.0000
                     Mean   :0.3573
                     3rd Qu.:1.0000
                     Max.   :1.0000
```

```
> ds
     X Age              Employment.Type GraduateOrNot AnnualIncome FamilyMembers ChronicDiseases FrequentFlyer EverTravelledAbroad TravelInsurance
1    0  31            Government Sector           Yes       400000             6               1            No                  No               0
2    1  31 Private Sector/Self Employed           Yes      1250000             7               0            No                  No               0
3    2  34 Private Sector/Self Employed           Yes       500000             4               1            No                  No               1
4    3  28 Private Sector/Self Employed           Yes       700000             3               1            No                  No               0
5    4  28 Private Sector/Self Employed           Yes       700000             8               1           Yes                  No               0
6    5  25 Private Sector/Self Employed            No      1150000             4               0            No                  No               0
7    6  31            Government Sector           Yes      1300000             4               0            No                  No               0
8    7  31 Private Sector/Self Employed           Yes      1350000             3               0           Yes                 Yes               1
9    8  28 Private Sector/Self Employed           Yes      1450000             6               1           Yes                 Yes               1
10   9  33            Government Sector           Yes       800000             3               0           Yes                  No               0
11  10  31            Government Sector           Yes       400000             9               1            No                  No               0
12  11  26 Private Sector/Self Employed           Yes      1400000             5               0           Yes                 Yes               1
13  12  32            Government Sector           Yes       850000             6               0            No                  No               1
14  13  31            Government Sector           Yes      1500000             6               0           Yes                 Yes               1
15  14  31            Government Sector           Yes       400000             3               0            No                  No               0
16  15  34 Private Sector/Self Employed           Yes       700000             7               0            No                  No               0
17  16  28 Private Sector/Self Employed           Yes      1150000             4               1            No                  No               0
18  17  28 Private Sector/Self Employed           Yes       800000             7               0            No                  No               1
19  18  29 Private Sector/Self Employed           Yes      1050000             5               1            No                  No               1
20  19  34 Private Sector/Self Employed           Yes      1500000             2               0           Yes                 Yes               1
21  20  28 Private Sector/Self Employed           Yes      1150000             6               0           Yes                  No               0
```

2. Boruta is used as the dimensionality reduction technique.Boruta is a feature selection method in R that is designed to find all relevant features in a dataset. It uses a random forest algorithm to assess the importance of each feature and compares it to shadow features (randomly created noise variables) to determine significance.




```
> attStats(Boruta.srx)
                      meanImp medianImp    minImp    maxImp  normHits  decision
Age                 47.666180 47.701018 44.118257 50.210767 1.0000000 Confirmed
Employment.Type     14.301254 14.420035 10.854561 16.939984 1.0000000 Confirmed
GraduateOrNot        7.229653  7.259723  3.886254  9.248540 0.9615385 Confirmed
AnnualIncome        71.582406 71.540345 66.693296 77.196946 1.0000000 Confirmed
FamilyMembers       38.855629 38.708474 35.581719 42.686618 1.0000000 Confirmed
ChronicDiseases      1.458719  1.494206 -1.837167  4.754579 0.1923077  Rejected
FrequentFlyer       15.841959 15.797392 13.819831 18.419146 1.0000000 Confirmed
EverTravelledAbroad 29.754704 29.682124 27.633963 32.168789 1.0000000 Confirmed
> print(Boruta.srx)
Boruta performed 26 iterations in 4.53133 secs.
 7 attributes confirmed important: Age, AnnualIncome, Employment.Type,
EverTravelledAbroad, FamilyMembers and 2 more;
 1 attributes confirmed unimportant: ChronicDiseases;
```

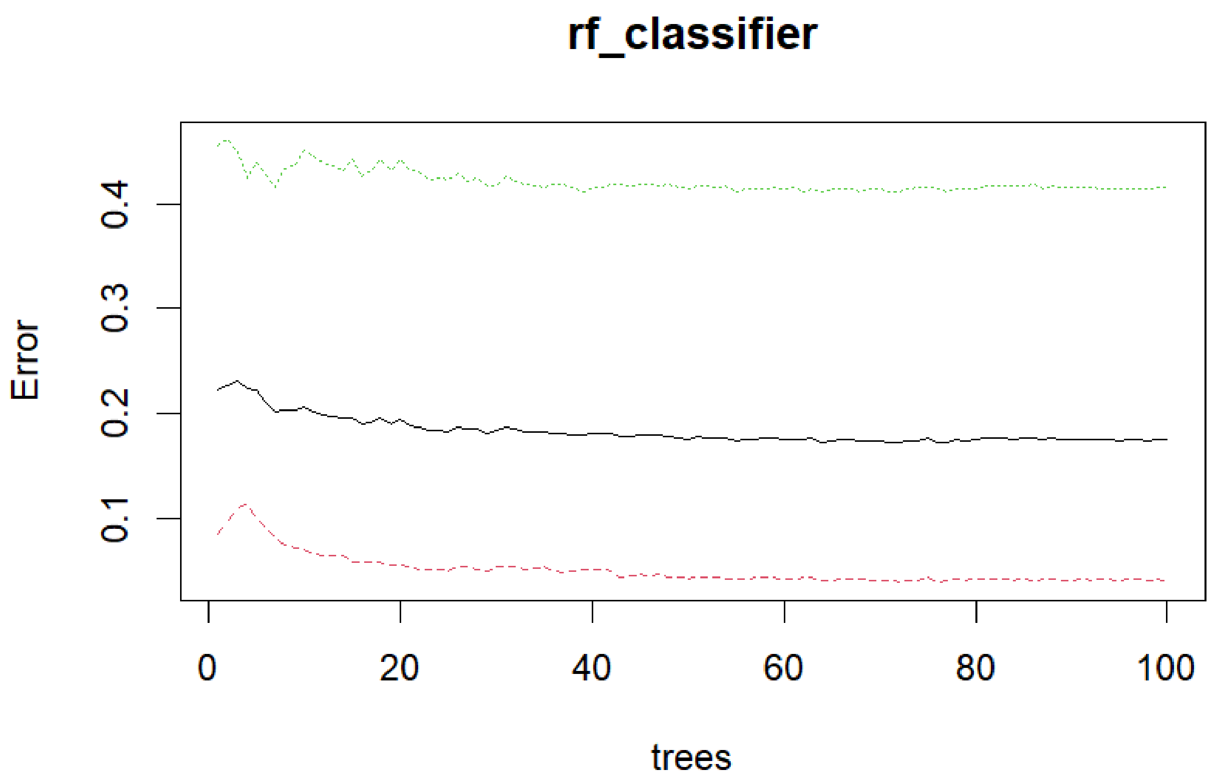
3. After the dimensionality reduction process, we got 7 features as important and one column as not important.So we take only the important features for building the model.

```
> #SELECTED FEATURES
> print(column_names <- names(ds))
[1] "Age"                 "Employment.Type"     "GraduateOrNot"
[4] "AnnualIncome"        "FamilyMembers"       "FrequentFlyer"
[7] "EverTravelledAbroad" "TravelInsurance"
```

By performing Boruta feature selection we got the result that the column named ChronicDiseases is not important according to this dataset.So this column is dropped from the dataset.And the remaining columns were used for building the model.

4. Random Forest Classifier was used for building the model. A Random Forest classifier is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

```
> summary(rf_classifier)
                Length Class  Mode
call               4   -none- call
type               1   -none- character
predicted       1490   factor numeric
err.rate         300   -none- numeric
confusion          6   -none- numeric
votes           2980   matrix numeric
oob.times       1490   -none- numeric
classes            2   -none- character
importance         7   -none- numeric
importanceSD       0   -none- NULL
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            14   -none- list
y               1490   factor numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
```



5. Results

```
> print(cm_rf)
   y_pred_rf
      0   1
  0 312   7
  1  75 103
```

True Positive (TP): The model correctly predicted 312 instances where the true class was 0.
True Negative (TN): The model correctly predicted 103 instances where the true class was 1.
False Positive (FP): The model incorrectly predicted 7 instances as class 1 when the true class was 0.
False Negative (FN): The model incorrectly predicted 75 instances as class 0 when the true class was 1.

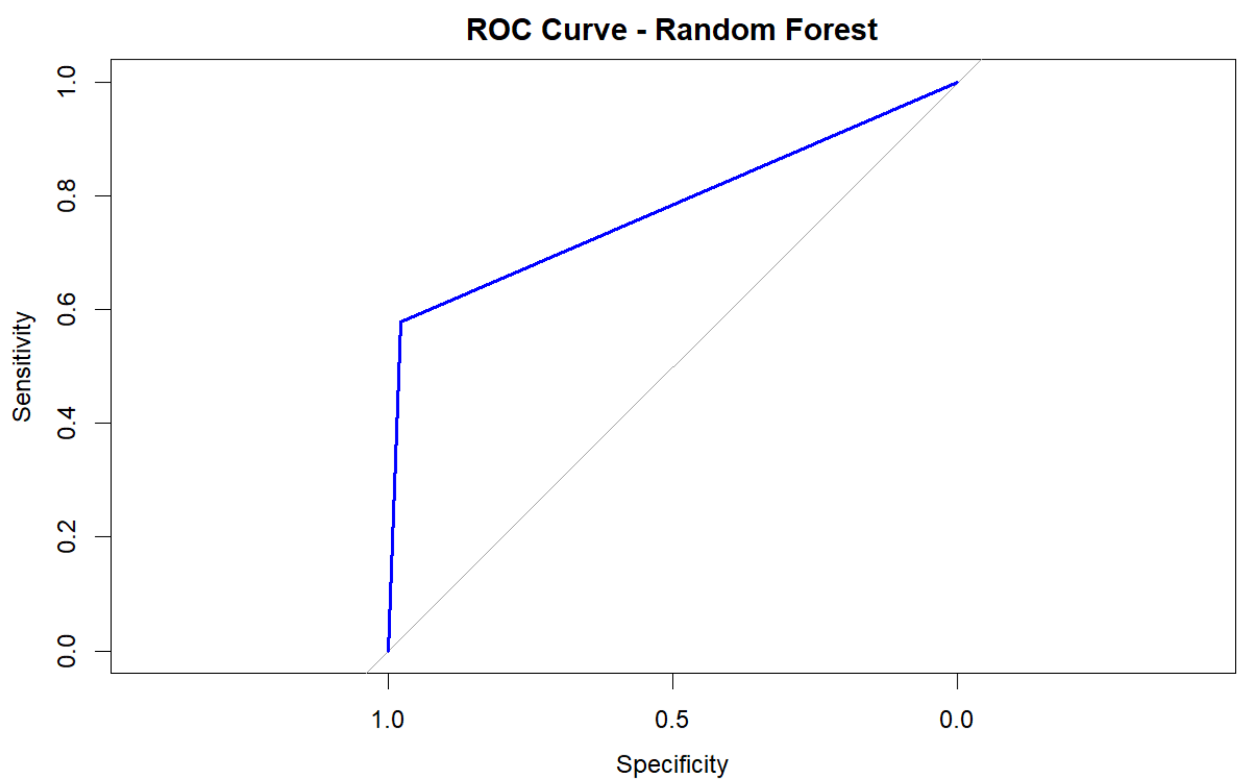
```
> accuracy <- sum(diag(cm_rf)) / sum(cm_rf)
> print(paste("Accuracy:", round(accuracy, 4)))
[1] "Accuracy: 0.835"
```

83.5% Accuracy: The model is correct about the travel insurance status (either 0 or 1) for approximately 83.5% of the instances in the test set.

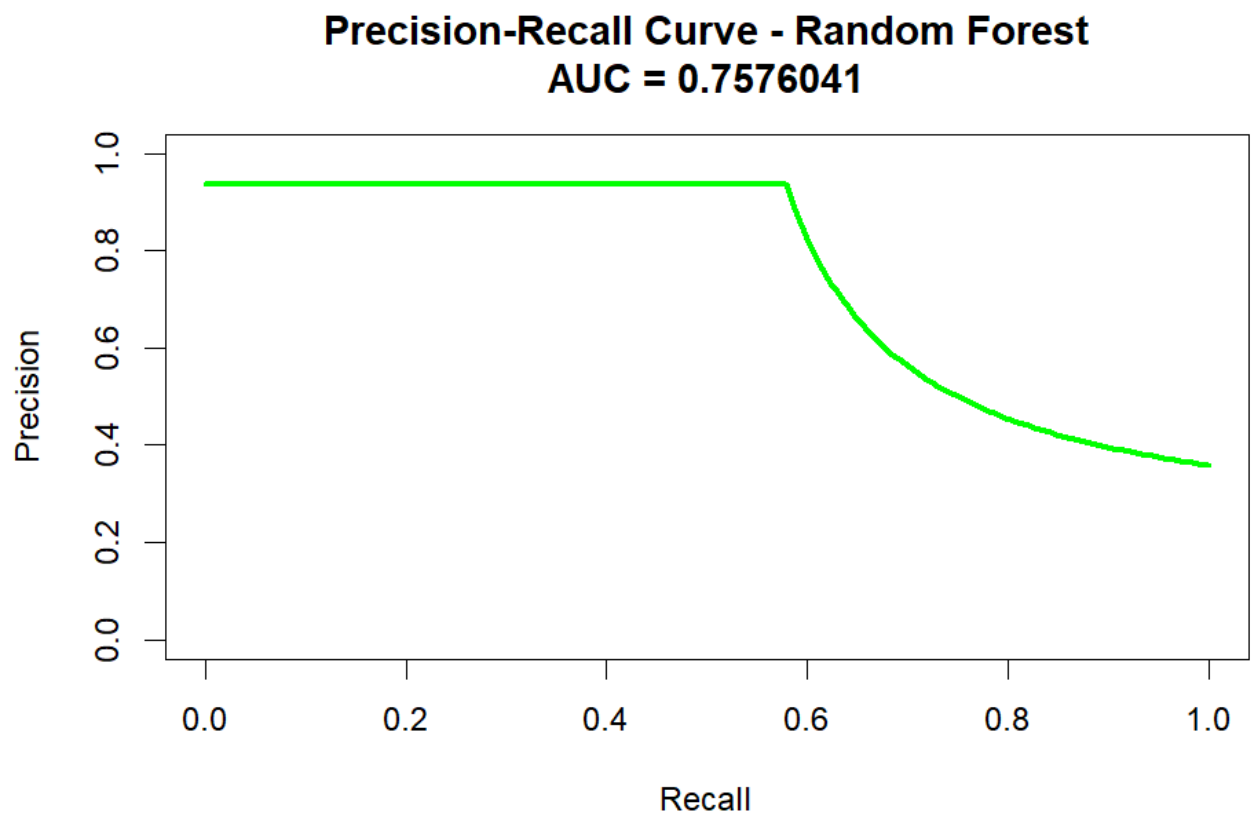
```
> recall <- cm_rf[2, 2] / sum(cm_rf[2, ])
> print(paste("Recall:", recall))
[1] "Recall: 0.578651685393258"
```

Out of all the instances that truly belong to the positive class, the classifier successfully identified 57.9% of them. The remaining 42.1% of actual positive instances were classified as false negatives,they were incorrectly predicted as the negative class.

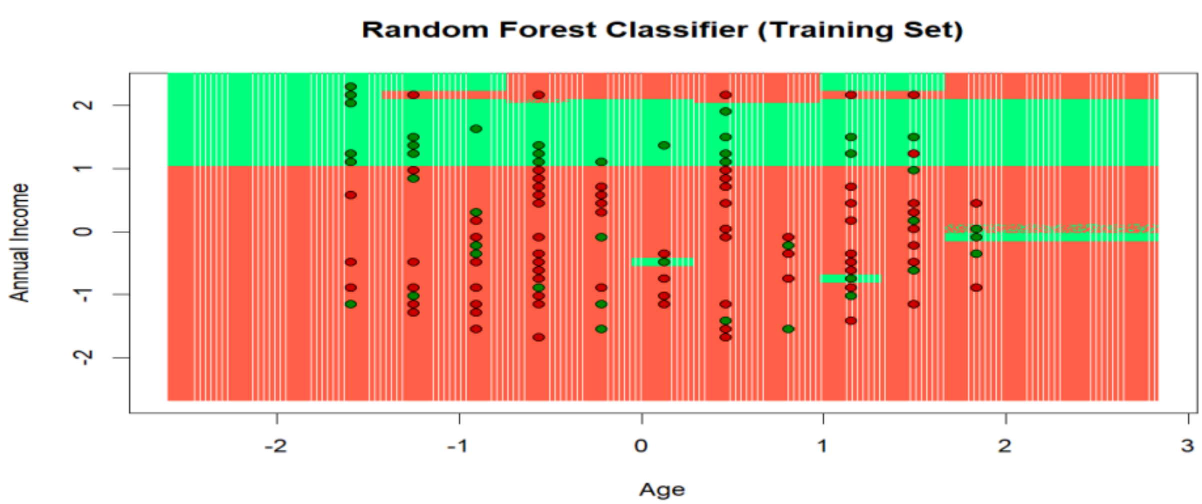
6.



The area under the ROC curve (AUC) is a summary measure of the model's performance.A curve above the diagonal indicates better-than-random performance. The steeper the curve, the better the model distinguishes between classes.



Points on the upper-right corner of the curve represent models with both high precision and high recall.In this case,there are very few false positives that is 7.



This is the graph showing the results of random forest classifier between the features Age and Annual Income on the Training Set.

This is the graph showing the results of random forest classifier between the features Age and Annual Income on the Test Set.

R CODE

```
ds=read.csv("TravelInsurancePrediction.csv")
summary(ds)
ds
ds<- ds[, -1]
ds$Employment.Type <- ifelse(grepl("Government", ds$Employment.Type), 1, 0)
ds$GraduateOrNot <- ifelse(ds$GraduateOrNot == "Yes", 1, 0)
ds$FrequentFlyer <- ifelse(ds$FrequentFlyer == "Yes", 1, 0)
ds$EverTravelledAbroad <- ifelse(ds$EverTravelledAbroad == "Yes", 1, 0)
ds
column_to_scale <- "AnnualIncome"
ds[[column_to_scale]] <- scale(ds[[column_to_scale]])
column_to_scale <- "Age"
ds[[column_to_scale]] <- scale(ds[[column_to_scale]])
column_to_scale <- "FamilyMembers"
ds[[column_to_scale]] <- scale(ds[[column_to_scale]])
```

```
ds
ds$TravelInsurance <- as.factor(ds$TravelInsurance)
ds$Age <- as.factor(ds$Age)
ds$Employment.Type<- as.factor(ds$Employment.Type)
ds$AnnualIncome<- as.factor(ds$AnnualIncome)
ds$FamilyMembers<- as.factor(ds$FamilyMembers)
ds$ChronicDiseases<- as.factor(ds$ChronicDiseases)
ds$FrequentFlyer<- as.factor(ds$FrequentFlyer)
ds
library(Boruta)
set.seed(777)
Boruta(TravelInsurance~.,data=ds)->Boruta.srx
print(Boruta.srx)
plot(Boruta.srx)
attStats(Boruta.srx)
Boruta(TravelInsurance~.,data=ds,getImp=getImpFerns)->Boruta.srx.ferns
print(Boruta.srx.ferns)
Boruta(TravelInsurance~.,data=ds,doTrace=2)->Boruta.srx
print(Boruta.srx)
ds<- ds[, -6]
ds
print(column_names <- names(ds))
ds[] <- lapply(ds, function(x) as.numeric(as.character(x)))
ds$TravelInsurance = factor(ds$TravelInsurance, levels = c(0, 1))
library(caTools)
set.seed(123)
split = sample.split(ds$TravelInsurance, SplitRatio = 0.75)
training_set = subset(ds, split == TRUE)
test_set = subset(ds, split == FALSE)
names(test_set)
library(randomForest)
rf_classifier = randomForest(TravelInsurance ~ ., data = training_set, ntree = 100)
summary(rf_classifier)
var_imp_plot <- plot(rf_classifier)
y_pred_rf = predict(rf_classifier, newdata = test_set[-8], type = 'response')
cm_rf = table(test_set[, 8], y_pred_rf)
print(cm_rf)
accuracy <- sum(diag(cm_rf)) / sum(cm_rf)
print(paste("Accuracy:", round(accuracy, 4)))
```

```
recall <- cm_rf[2, 2] / sum(cm_rf[2, ])
print(paste("Recall:", recall))

install.packages("pROC")
library(pROC)
roc_curve <- roc(test_set[, 8], as.numeric(y_pred_rf))
plot(roc_curve, col = "blue", main = "ROC Curve - Random Forest")
install.packages("PRROC")
library(PRROC)
test_set[, 8] <- as.numeric(as.character(test_set[, 8]))
pr_curve <- pr.curve(scores.class0 = as.numeric(y_pred_rf), weights.class0 = test_set[,
8], curve = TRUE)
plot(pr_curve, col = "green", main = "Precision-Recall Curve - Random Forest")
library(ElemStatLearn)
set=training_set
names(set)
x1=seq(min(set[,1])-1, max(set[,1])+1, by=0.01)
x2=seq(min(set[,2])-1, max(set[,2])+1, by=0.01)
grid_set=expand.grid(x1,x2)
colnames(grid_set) <- c("Age","AnnualIncome")
y_grid_rf = predict(rfe_classifier, newdata = grid_set, type = 'response')
plot(set[, -3],
    main = "Random Forest Classifier (Training Set)",
    xlab = "Age",
    ylab = "Annual Income",
    xlim = range(x1),
    ylim = range(x4))
contour(x1, x4,matrix(as.numeric(y_grid_rf), length(x1), length(x4)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid_rf == 1, 'springgreen', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
library(ElemStatLearn)
set=test_set
names(set)
x1=seq(min(set[,1])-1, max(set[,1])+1, by=0.01)
x2=seq(min(set[,2])-1, max(set[,2])+1, by=0.01)
grid_set=expand.grid(x1,x2)
colnames(grid_set) <- c("Age","AnnualIncome")
y_grid_rf = predict(rfe_classifier, newdata = grid_set, type = 'response')
plot(set[, -3],
```

```
    main = "Random Forest Classifier (Test Set)",
    xlab = "Age",
    ylab = "Annual Income",
    xlim = range(x1),
    ylim = range(x4))
contour(x1, x4,matrix(as.numeric(y_grid_rf), length(x1), length(x4)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid_rf == 1, 'springgreen', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```