BIN381_Project_Milestone 3_MODELLING

Group F

2024-10-15

#Read and Split the Dataset

```
#Load Libraries
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

library(rpart)
library(rpart.plot)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(caTools)
library(randomForest)

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine

#Load the dataset
custData <- read.csv("CustData2_Prepared.csv")
```

```r
#Explore the structure of the dataset
str(custData)
```

```
## 'data.frame':    191317 obs. of  26 variables:
##  $ Annual_Salary                       : num  0.872 -0.156 -0.455 0.266 -
0.621 ...
##  $ Gross_Pay_Last_Paycheck             : num  0.591 -0.267 -0.511 0.432 -
0.637 ...
##  $ Gross_Year_To_Date                  : num  0.827 -0.267 -0.884 0.602 -
0.417 ...
##  $ Gross_Year_To_Date_FRS_Contribution: num  0.823 -0.268 -0.884 0.599 -
0.417 ...
##  $ Age                                 : num  2.088 0.021 0.688 2.022 -
0.779 ...
##  $ Household_Size                      : int  2 3 2 2 2 2 2 2 3 2 ...
##  $ Years_Residence                     : int  2 5 2 3 4 4 4 4 5 3 ...
##  $ Marital_Statusdivorced              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Marital_Statusmarried               : int  1 0 1 0 0 0 0 0 1 0 ...
##  $ Marital_Statussingle                : int  0 1 0 1 1 1 1 1 0 1 ...
##  $ Marital_Statuswidowed               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ EducationBach.                      : int  0 1 0 1 0 0 0 0 1 1 ...
##  $ EducationHS.grad                    : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ EducationMasters                    : int  0 0 0 0 1 1 1 1 0 0 ...
##  $ OccupationCleric.                   : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ OccupationExec.                     : int  0 1 0 0 0 0 0 0 1 0 ...
##  $ OccupationProf.                     : int  0 0 0 0 1 1 1 1 0 0 ...
##  $ OccupationSales                     : int  0 0 0 1 0 0 0 0 0 1 ...
##  $ Salary_GroupLow                     : int  0 0 1 0 1 0 0 0 0 0 ...
##  $ Salary_GroupMedium                  : int  0 1 0 0 0 0 0 1 0 0 ...
##  $ Salary_GroupHigh                    : int  0 0 0 1 0 1 0 0 0 1 ...
##  $ Salary_GroupVery.High               : int  1 0 0 0 0 0 1 0 1 0 ...
##  $ Frequency_Title                     : int  8206 1030 28 67 1368 10809 7
723 1030 101 ...
##  $ Frequency_Department                : int  16988 16925 6138 16925 8895
18158 14 29331 8895 16925 ...
##  $ Frequency_Country_ID                : int  2079 2079 2079 2079 2079 2079
2079 2079 2079 2079 ...
##  $ Eligible                            : int  1 1 0 1 0 1 1 1 1 1 ...
```

```r
#Split the dataset to 80% training data and 20% testing data
set.seed(123)
train_index <- createDataPartition(custData$Eligible, p = 0.8, list = FALSE)
train_data <- custData[train_index, ]
test_data <- custData[-train_index, ]
```

#Logistic Regression

```r
#Build Logistic Regression Model
logisticRegressionModel <- glm(formula = Eligible~ . -Annual_Salary,
                               data = train_data,family = 'binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(logisticRegressionModel)
```

```
##
## Call:
## glm(formula = Eligible ~ . - Annual_Salary, family = "binomial",
##     data = train_data)
##
## Coefficients: (7 not defined because of singularities)
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -4.234e+08  4.712e+11  -0.001  0.99928
## Gross_Pay_Last_Paycheck             1.546e+00  6.487e-02  23.839  < 2e-16
***
## Gross_Year_To_Date                 -1.912e+00  1.674e+00  -1.142  0.25347
## Gross_Year_To_Date_FRS_Contribution  5.049e+00  1.674e+00   3.017  0.00256
**
## Age                                 2.924e-02  1.174e-02   2.490  0.01278
*
## Household_Size                      3.623e-02  4.979e-02   0.728  0.46681
## Years_Residence                    -1.179e-02  1.545e-02  -0.763  0.44538
## Marital_Statusdivorced             -1.326e-01  2.265e-01  -0.585  0.55833
## Marital_Statusmarried              -2.633e-01  2.054e-01  -1.282  0.19997
## Marital_Statussingle               -2.746e-01  2.048e-01  -1.341  0.18006
## Marital_Statuswidowed                      NA         NA      NA       NA
## EducationBach.                     -1.487e-02  2.672e-02  -0.556  0.57790
## EducationHS.grad                           NA         NA      NA       NA
## EducationMasters                           NA         NA      NA       NA
## OccupationCleric.                          NA         NA      NA       NA
## OccupationExec.                            NA         NA      NA       NA
## OccupationProf.                            NA         NA      NA       NA
## OccupationSales                            NA         NA      NA       NA
## Salary_GroupLow                     4.234e+08  4.712e+11   0.001  0.99928
## Salary_GroupMedium                  4.234e+08  4.712e+11   0.001  0.99928
## Salary_GroupHigh                    4.234e+08  4.712e+11   0.001  0.99928
## Salary_GroupVery.High               4.234e+08  4.712e+11   0.001  0.99928
## Frequency_Title                     1.202e-04  3.372e-06  35.637  < 2e-16
***
## Frequency_Department               -8.632e-06  1.416e-06  -6.095  1.1e-09
***
## Frequency_Country_ID                8.212e-07  5.429e-07   1.512  0.13042
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 199391  on 153053  degrees of freedom
## Residual deviance:  42371  on 153036  degrees of freedom
## AIC: 42407
```

```
##
## Number of Fisher Scoring iterations: 20
```

```
#Make Predictions using Logistic Regression
logisticRegressionPrediction <- predict(logisticRegressionModel, newdata =
test_data, type ='response')
head(logisticRegressionPrediction)
```

```
##            2            4            5            6           14
25
## 4.681861e-01 1.000000e+00 4.870882e-10 1.000000e+00 1.000000e+00
1.000000e+00
```

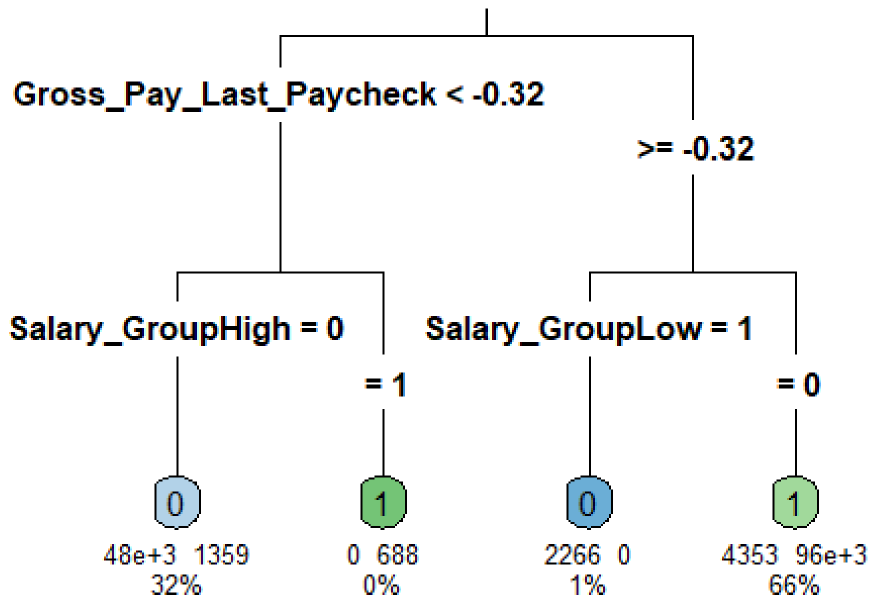
```
logisticRegressionY_pred = ifelse(logisticRegressionPrediction >0.5, 1, 0)

#Confusion matrix of Logistic Regression
logisticRegression_matrix <- table(actual = test_data$Eligible, predicted =
logisticRegressionY_pred)
logisticRegression_matrix
```

```
##       predicted
## actual     0     1
##      0 12483  1004
##      1   985 23791
```

```
logisticRegression_truePositive <- logisticRegression_matrix[1, 1]
logisticRegression_trueNegative <- logisticRegression_matrix[2, 2]
logisticRegression_falsePositive <- logisticRegression_matrix[1, 2]
logisticRegression_falseNegative <- logisticRegression_matrix[2, 1]

# Calculate Evaluation Metrics
logisticRegression_accuracy <- round((sum(diag(logisticRegression_matrix)) /
sum(logisticRegression_matrix)), 2)
logisticRegression_precision <- round(logisticRegression_truePositive /
(logisticRegression_truePositive + logisticRegression_falsePositive), 2)
logisticRegression_recall <- round(logisticRegression_truePositive /
(logisticRegression_truePositive + logisticRegression_falseNegative), 2)
logisticRegression_f1_score <- round(2 * (logisticRegression_precision *
logisticRegression_recall) / (logisticRegression_precision +
logisticRegression_recall), 2)
```

#Decision Tree

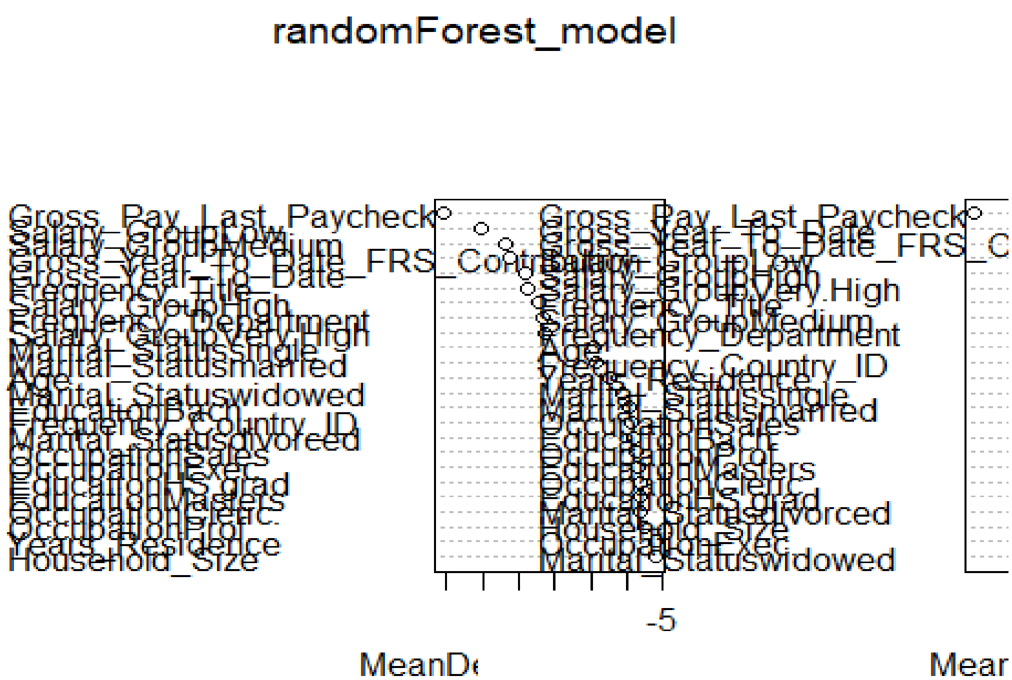
```
#Build Decision Tree Model
decisionTreeModel <- rpart(Eligible ~ . -Annual_Salary, data = train_data,
method = 'class')
```




```r
# Make predictions on the test data
decisionTreePredictions <- predict(decisionTreeModel, newdata = test_data,
type = 'class')

#Confusion matrix
decisionTreeMatrix <- table(test_data$Eligible, decisionTreePredictions)
print(decisionTreeMatrix)

##    decisionTreePredictions
##         0     1
##   0 12450  1037
##   1   314 24462

decisionTreeTruePositive <- decisionTreeMatrix[1, 1]
decisionTreeTrueNegative <- decisionTreeMatrix[2, 2]
decisionTreeFalsePositive <- decisionTreeMatrix[1, 2]
decisionTreeFalseNegative <- decisionTreeMatrix[2, 1]

#Calculate Evaluation Metrics
decisionTreeAccuracy <- round((sum(diag(decisionTreeMatrix)) /
sum(decisionTreeMatrix)), 2)
decisionTreePrecision <- round(decisionTreeTruePositive /
(decisionTreeTruePositive + decisionTreeFalsePositive), 2)
decisionTreeRecall <- round(decisionTreeTruePositive /
(decisionTreeTruePositive + decisionTreeFalseNegative), 2)
decisionTreeF1Score <- round(2 * (decisionTreePrecision * decisionTreeRecall)
/ (decisionTreePrecision + decisionTreeRecall), 2)
```

#Random Forest

```r
#Reload the dataset
custData <- read.csv("CustData2_Prepared.csv")
custData$Eligible <- as.factor(custData$Eligible)

#Split the dataset to 80% training data and 20% testing data
set.seed(123)
train_index <- createDataPartition(custData$Eligible, p = 0.8, list = FALSE)
train_data <- custData[train_index, ]
test_data <- custData[-train_index, ]

#Build Random Forest Model
randomForest_model <- randomForest(Eligible ~ . -Annual_Salary, data =
train_data, ntree = 100, mtry = 3, importance = TRUE)
```




```r
#Make Predictions Using Random Forest
randomForest_predictions <- predict(randomForest_model, newdata = test_data)

#Matrix for Random Forest
randomForest_cm <- confusionMatrix(as.factor(randomForest_predictions),
as.factor(test_data$Eligible))
randomForest_matrix <- randomForest_cm$table

randomForest_truePositive <- randomForest_matrix[1, 1]
randomForest_trueNegative <- randomForest_matrix[2, 2]
randomForest_falsePositive <- randomForest_matrix[1, 2]
```

```r
randomForest_falseNegative <- randomForest_matrix[2, 1]

#Calculate Evaluation Metrics
randomForest_accuracy <- round((sum(diag(randomForest_matrix)) /
sum(randomForest_matrix)), 2)
randomForest_precision <- round(randomForest_truePositive /
(randomForest_truePositive + randomForest_falsePositive), 2)
randomForest_recall <- round(randomForest_truePositive /
(randomForest_truePositive + randomForest_falseNegative), 2)
randomForest_f1_score <- round(2 * (randomForest_precision *
randomForest_recall) / (randomForest_precision + randomForest_recall), 2)
```

#Model Evaluation

```r
##Print Evaluation Metrics of All Models
cat("Logistic Regression Accuracy:", logisticRegression_accuracy, "\n")

## Logistic Regression Accuracy: 0.95

cat("Logistic Regression Precision:", logisticRegression_precision, "\n")

## Logistic Regression Precision: 0.93

cat("Logistic Regression Recall:", logisticRegression_recall, "\n")

## Logistic Regression Recall: 0.93

cat("Logistic Regression F1-score:", logisticRegression_f1_score, "\n")

## Logistic Regression F1-score: 0.93

cat("Decision Tree Accuracy:", decisionTreeAccuracy, "\n")

## Decision Tree Accuracy: 0.96

cat("Decision Tree Precision:", decisionTreePrecision, "\n")

## Decision Tree Precision: 0.92

cat("Decision Tree Recall:", decisionTreeRecall, "\n")

## Decision Tree Recall: 0.98

cat("Decision Tree F1-score:", decisionTreeF1Score, "\n")

## Decision Tree F1-score: 0.95

cat("Random Forest Accuracy:", randomForest_accuracy, "\n")

## Random Forest Accuracy: 0.97

cat("Random Forest Precision:", randomForest_precision, "\n")

## Random Forest Precision: 0.98
```

```r
cat("Random Forest Recall:", randomForest_recall, "\n")
```

```
## Random Forest Recall: 0.93
```

```r
cat("Random Forest F1-score:", randomForest_f1_score, "\n")
```

```
## Random Forest F1-score: 0.95
```

#Save Models

```r
#Save the logistic regression model
saveRDS(logisticRegressionModel, file = "logistic_regression_model.rds")

#Save the decision tree model
saveRDS(decisionTreeModel, file = "decision_tree_model.rds")

#Save the random forest model
saveRDS(randomForest_model, file = "random_forest_model.rds")
```