BIN381_Project_Milestone-4_MODELLING

Group F

2024-10-21

Load the dataset

```
#Read the dataset into object "custData"
custData <- read.csv("CustData2_Prepared.csv")
custData$Eligible <- as.factor(custData$Eligible)
```

Split the dataset into training data and testing data

```
#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

Build/train Logistic Regression Model

```
logisticRegressionModel <- glm(formula = Eligible~ . -Annual_Salary,
                               data = train_data,family = 'binomial')
```

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

Make Predictions using the model

```
logisticRegressionPrediction <- predict(logisticRegressionModel,
                                        newdata = test_data, type ='response')
head(logisticRegressionPrediction)
```

```
##            3           16           29           30           37           39 
## 3.683766e-10 9.246811e-11 1.000000e+00 1.000000e+00 3.158605e-01 9.314723e-01
```

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

Confusion matrix

Create confusion matrix

```
logisticRegression_matrix <- table(actual = test_data$Eligible,
                                   predicted = logisticRegressionY_pred)
logisticRegression_matrix
```

```
##       predicted
## actual     0     1
##      0 12613   997
##      1   979 23674
```

Extract TP, TN, FP and FN from confusion matrix

```
# Extract TruePositive, TrueNegative, FalsePositive
# and FalseNegative for confusion matrix
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

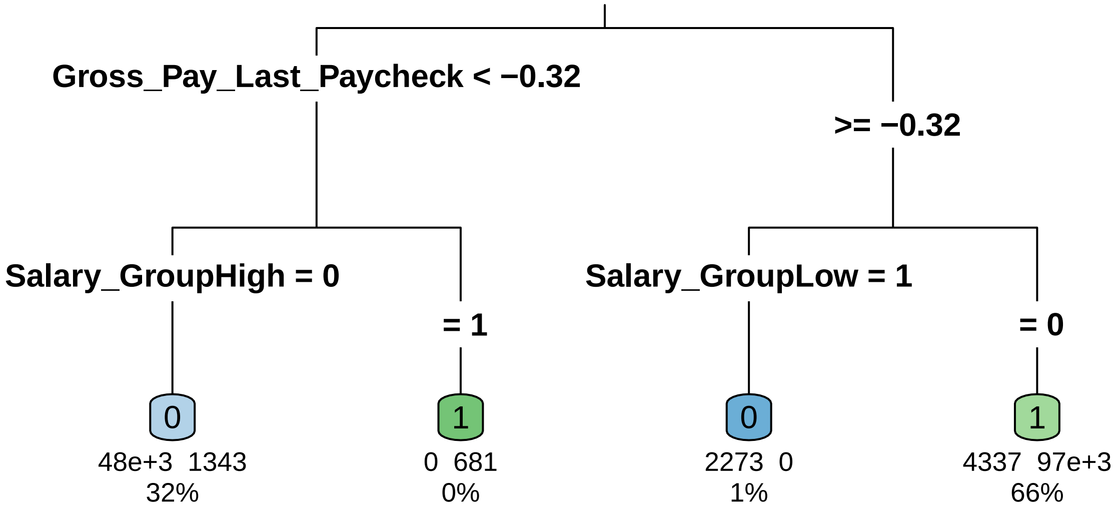
```
# Calculate Evaluation Metrics
logisticRegression_accuracy <-
  round(((sum(diag(logisticRegression_matrix)) /
            sum(logisticRegression_matrix))) * 100, 2)
logisticRegression_precision <-
  round((logisticRegression_truePositive /
           (logisticRegression_truePositive +
              logisticRegression_falsePositive)) * 100, 2)
logisticRegression_recall <-
  round((logisticRegression_truePositive /
           (logisticRegression_truePositive +
              logisticRegression_falseNegative)) * 100, 2)
logisticRegression_f1_score <-
  round(2 * (logisticRegression_precision * logisticRegression_recall) /
          (logisticRegression_precision + logisticRegression_recall), 2)
```

Decision Tree

Build/train Decision Tree Model Model

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

Visualize the model



Make Predictions using the model

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

Confusion matrix

Create confusion matrix

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

```
##    decisionTreePredictions
##         0     1
##   0 12557  1053
##   1   330 24323
```

Extract TP, TN, FP and FN from confusion matrix

```
# Extract TruePositive, TrueNegative, FalsePositive
# and FalseNegative for confusion matrix
decisionTreeTruePositive <- decisionTreeMatrix[1, 1]
decisionTreeTrueNegative <- decisionTreeMatrix[2, 2]
decisionTreeFalsePositive <- decisionTreeMatrix[1, 2]
decisionTreeFalseNegative <- decisionTreeMatrix[2, 1]
```

Claculate evaluation metrics

```
#Calculate Evaluation Metrics
decisionTreeAccuracy <-
  round((sum(diag(decisionTreeMatrix)) / sum(decisionTreeMatrix)) * 100, 2)
decisionTreePrecision <-
  round((decisionTreeTruePositive /
           (decisionTreeTruePositive + decisionTreeFalsePositive)) * 100, 2)
decisionTreeRecall <-
  round((decisionTreeTruePositive / (decisionTreeTruePositive +
                                       decisionTreeFalseNegative)) * 100, 2)
decisionTreeF1Score <-
  round(2 * (decisionTreePrecision * decisionTreeRecall) /
                            (decisionTreePrecision + decisionTreeRecall), 2)
```

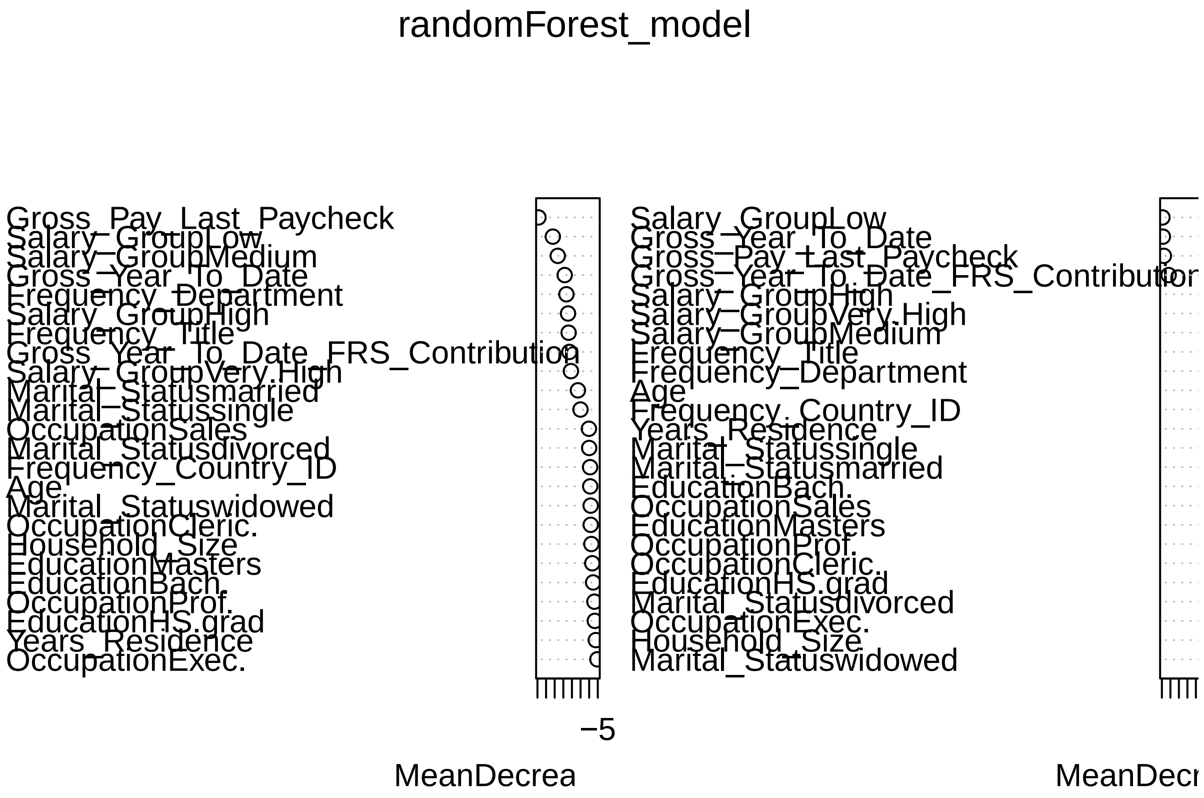
Random Forest

Build/train Random Forest Model Model

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

Visualize attribute importance

```r
#The attribute importance can be visualised using the random forest model
varImpPlot(randomForest_model)
```



Make Predictions using the model

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

Confusion matrix

Create confusion matrix

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

Extract TP, TN, FP and FN from confusion matrix

```r
# Extract TruePositive, TrueNegative, FalsePositive
# and FalseNegative for confusion matrix
randomForest_truePositive <- randomForest_matrix[1, 1]
randomForest_trueNegative <- randomForest_matrix[2, 2]
randomForest_falsePositive <- randomForest_matrix[1, 2]
randomForest_falseNegative <- randomForest_matrix[2, 1]
```

Calculate evaluation metrics

```r
#Calculate Evaluation Metrics
randomForest_accuracy <-
  round((sum(diag(randomForest_matrix)) / sum(randomForest_matrix)) * 100, 2)
randomForest_precision <-
  round((randomForest_truePositive / (randomForest_truePositive +
                                        randomForest_falsePositive)) * 100, 2)
randomForest_recall <-
  round((randomForest_truePositive / (randomForest_truePositive +
                                        randomForest_falseNegative)) * 100, 2)
randomForest_f1_score <-
  round(2 * (randomForest_precision * randomForest_recall) /
                            (randomForest_precision + randomForest_recall), 2)
```

Model Evaluation

Print evaluation metrics of Logistic Regression Model

```r
cat("Logistic Regression Accuracy:", logisticRegression_accuracy, "% \n")
```

```
## Logistic Regression Accuracy: 94.84 %
```

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

```
## Logistic Regression Precision: 92.67 %
```

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

```
## Logistic Regression Recall: 92.8 %
```

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

```
## Logistic Regression F1-score: 92.73 %
```

Print evaluation metrics of Decision Tree Model

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

```
## Decision Tree Accuracy: 96.39 %
```

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

```
## Decision Tree Precision: 92.26 %
```

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

```
## Decision Tree Recall: 97.44 %
```

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

```
## Decision Tree F1-score: 94.78 %
```

Print evaluation metrics of Random Forest Model

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

```
## Random Forest Accuracy: 96.97 %
```

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

```
## Random Forest Precision: 98.48 %
```

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

```
## Random Forest Recall: 92.92 %
```

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

```
## Random Forest F1-score: 95.62 %
```

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")
```