Predictive Modeling_Capstone Project

2024-03-06

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
# Load the 'Pre-processed' data through readxl package:
library(readxl)

# Read the Excel file
Preprocessed_dataset <- read_excel("preprocessed_dataset.xlsx")
```

```
Preprocessed_dataset$Delay_Severity<-as.factor(Preprocessed_dataset$Delay_Severity)
```

```
 # Adjusting column names appropriately
colnames(Preprocessed_dataset) <- c("Delay_Severity", "Route", "Day", "Incident", "Direction","Vehicle"
```

```
str(Preprocessed_dataset)
```

```
## tibble [150,737 x 7] (S3: tbl_df/tbl/data.frame)
##  $ Delay_Severity: Factor w/ 3 levels "Borderline Late (<10 Min)",..: 3 3 3 3 1 1 3 3 3 3 ...
##  $ Route         : num [1:150737] 0.0901 0.0681 0.034 0.8999 0.0841 ...
##  $ Day           : num [1:150737] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ Incident      : num [1:150737] 0 0.333 0 0.333 0.333 ...
##  $ Direction     : num [1:150737] 1 0.75 0.5 0.5 0.5 1 0.5 0.25 1 1 ...
##  $ Vehicle       : num [1:150737] 0.0886 0.0849 0.0106 0.0337 0.0157 ...
##  $ Time_Period   : num [1:150737] 0.667 0.667 0.667 0.667 0.667 ...
```

```
# Load required packages:
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```r
library(caret)
```

```
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.3.1

##
## Attaching package: 'ggplot2'

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

## Loading required package: lattice
```

```r
library(class)
```

#EXPERIMENTAL DESIGN: TRAIN:TEST SPLIT:

```r
set.seed(123) # Set a random seed for reproducibility

# Create indices for the training set (85% of the dataset)
trainIndex <- createDataPartition(Preprocessed_dataset$Delay_Severity, p = .85,
                                  list = FALSE,
                                  times = 1)

# Split the data into training and testing sets
train <- Preprocessed_dataset[trainIndex, ]
test <- Preprocessed_dataset[-trainIndex, ]
```

MACHINE LEARNING CLASSIFICATION MODELS TO BE TESTED

```r
# We'll train-test 4 separate classification algorithms - Random Forest (RF), Logistic Regression (LR),

# All classification models will make "TTC Delay prediction" in 3 class (Delay Severity) - 1. Borderlin

# To ensure accuracy and reliability, 10 fold cross validation will be conducted for each models

# Based on the outcome, final model will be selected. Based on the final model selection, we'll try to
```

ALGORITHM 1: RANDOM FOREST:

```r
# Train the Random Forest model
rf_model <- randomForest(Delay_Severity ~ ., data = train, ntree = 100)

# Print the model summary
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Delay_Severity ~ ., data = train, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 34.64%
## Confusion matrix:
##                              Borderline Late (<10 Min)
## Borderline Late (<10 Min)                        23097
## Considerably Late (11-15 Min)                     7248
## Extremely Late (>15 Min)                          3682
##                              Considerably Late (11-15 Min)
## Borderline Late (<10 Min)                             9161
## Considerably Late (11-15 Min)                        28082
## Extremely Late (>15 Min)                              6806
##                              Extremely Late (>15 Min) class.error
## Borderline Late (<10 Min)                        7621   0.4208230
## Considerably Late (11-15 Min)                    9862   0.3786068
## Extremely Late (>15 Min)                        32568   0.2435897
```

```r
# Predict on the test set
predictions <- predict(rf_model, newdata = test)

# Assuming caret package is installed for confusionMatrix
library(caret)

# Generate the confusion matrix
conf_matrix <- confusionMatrix(predictions, test$Delay_Severity)

# Print the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##                                Reference
## Prediction                       Borderline Late (<10 Min)
##   Borderline Late (<10 Min)                           4129
##   Considerably Late (11-15 Min)                       1596
##   Extremely Late (>15 Min)                            1312
##                                Reference
## Prediction                       Considerably Late (11-15 Min)
##   Borderline Late (<10 Min)                               1201
##   Considerably Late (11-15 Min)                           4983
##   Extremely Late (>15 Min)                                1791
```

```
##                                  Reference
## Prediction                        Extremely Late (>15 Min)
##   Borderline Late (<10 Min)                            595
##   Considerably Late (11-15 Min)                       1183
##   Extremely Late (>15 Min)                            5820
## 
## Overall Statistics
## 
##                Accuracy : 0.6604
##                  95% CI : (0.6542, 0.6666)
##     No Information Rate : 0.3527
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.4891
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: Borderline Late (<10 Min)
## Sensitivity                                    0.5868
## Specificity                                    0.8847
## Pos Pred Value                                 0.6969
## Neg Pred Value                                 0.8257
## Prevalence                                     0.3112
## Detection Rate                                 0.1826
## Detection Prevalence                           0.2621
## Balanced Accuracy                              0.7357
##                      Class: Considerably Late (11-15 Min)
## Sensitivity                                        0.6248
## Specificity                                        0.8101
## Pos Pred Value                                     0.6420
## Neg Pred Value                                     0.7985
## Prevalence                                         0.3527
## Detection Rate                                     0.2204
## Detection Prevalence                               0.3433
## Balanced Accuracy                                  0.7175
##                      Class: Extremely Late (>15 Min)
## Sensitivity                                   0.7660
## Specificity                                   0.7933
## Pos Pred Value                                0.6522
## Neg Pred Value                                0.8701
## Prevalence                                    0.3360
## Detection Rate                                0.2574
## Detection Prevalence                          0.3946
## Balanced Accuracy                             0.7796
```

```
#Overall Accuracy:The Random Forest model achieves an accuracy of 66.04% [cross validation ~65.86%], in

#Class-Specific Performance:Sensitivity values range from 58.68% to 76.60%, showing the model's ability

#Predictive Values:Positive predictive values (PPV) range from 64.20% to 69.69%, indicating the proport

#Negative predictive values (NPV) range from 79.85% to 87.01%, indicating the proportion of correct neg
```

```
#Balanced Accuracy:The balanced accuracy ranges from 71.75% to 77.96% across different classes, providi
```

#10 FOLD CROSS-VALIDATION FOR RANDOM FOREST ALGORITHM:

```r
library(caret)

# Create 10-fold cross-validation folds
folds <- createFolds(train$Delay_Severity, k = 10)

# Initialize an empty vector to store accuracies
accuracies <- numeric(length(folds))

# Perform 10-fold cross-validation
for (i in 1:length(folds)) {
  # Extract training and test data for current fold
  training_fold <- train[-folds[[i]], ]
  test_fold <- train[folds[[i]], ]

  # Train the Random Forest model
  rf_model <- randomForest(Delay_Severity ~ ., data = training_fold, ntree = 100)

  # Predict on the test set
  predictions <- predict(rf_model, newdata = test_fold)

  # Generate the confusion matrix
  conf_matrix <- confusionMatrix(predictions, test_fold$Delay_Severity)

  # Calculate accuracy and store it
  accuracies[i] <- conf_matrix$overall['Accuracy']
}

# Print accuracies of each fold
print(accuracies)
```

```
##  [1] 0.6561037 0.6618531 0.6541013 0.6587574 0.6552720 0.6582891 0.6658342
##  [8] 0.6596426 0.6553500 0.6613860
```

```r
# Cross-validation output for Random Forest dataset
RF_cv_output <- c(0.656, 0.662, 0.654, 0.659, 0.655, 0.658, 0.666, 0.660, 0.655, 0.661)

# Compute the mean of the cross-validation output
mean_RF_cv_output <- mean(RF_cv_output)

# Print the mean
print(mean_RF_cv_output)
```

```
## [1] 0.6586
```

```r
library(caret)
library(nnet)
```

#ALGORITHM 2: MULTINOMIAL LOGISTIC REGRESSION:

```r
# Train the multinomial logistic regression model
multinom_model <- multinom(Delay_Severity ~ ., data = train)
```

```
## # weights:  24 (14 variable)
## initial  value 140761.896710
## iter  10 value 137661.895481
## final  value 137174.041403
## converged
```

```r
# Predict on the test set
test$predicted_severity <- predict(multinom_model, newdata = test)

# Generate the confusion matrix
conf_matrix <- confusionMatrix(test$predicted_severity, test$Delay_Severity)

# Print the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##                                  Reference
## Prediction                         Borderline Late (<10 Min)
##   Borderline Late (<10 Min)                              446
##   Considerably Late (11-15 Min)                         3290
##   Extremely Late (>15 Min)                              3301
##                                  Reference
## Prediction                         Considerably Late (11-15 Min)
##   Borderline Late (<10 Min)                                  553
##   Considerably Late (11-15 Min)                             3550
##   Extremely Late (>15 Min)                                  3872
##                                  Reference
## Prediction                         Extremely Late (>15 Min)
##   Borderline Late (<10 Min)                             132
##   Considerably Late (11-15 Min)                        2983
##   Extremely Late (>15 Min)                             4483
##
## Overall Statistics
##
##                Accuracy : 0.375
##                  95% CI : (0.3687, 0.3814)
##     No Information Rate : 0.3527
##     P-Value [Acc > NIR] : 1.512e-12
##
##                   Kappa : 0.0501
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                        Class: Borderline Late (<10 Min)
## Sensitivity                                     0.06338
## Specificity                                     0.95601
## Pos Pred Value                                  0.39434
```

```
## Neg Pred Value                                0.69314
## Prevalence                                    0.31123
## Detection Rate                                0.01973
## Detection Prevalence                          0.05002
## Balanced Accuracy                             0.50970
##                      Class: Considerably Late (11-15 Min)
## Sensitivity                                        0.4451
## Specificity                                        0.5714
## Pos Pred Value                                     0.3614
## Neg Pred Value                                     0.6539
## Prevalence                                         0.3527
## Detection Rate                                     0.1570
## Detection Prevalence                               0.4345
## Balanced Accuracy                                  0.5083
##                      Class: Extremely Late (>15 Min)
## Sensitivity                                   0.5900
## Specificity                                   0.5222
## Pos Pred Value                                0.3846
## Neg Pred Value                                0.7156
## Prevalence                                    0.3360
## Detection Rate                                0.1983
## Detection Prevalence                          0.5155
## Balanced Accuracy                             0.5561
```

```r
# Model Comparison LR Vs RF:

#Overall Accuracy:LR: Achieves an accuracy of 37.5%; RF: Outperforms LR with an accuracy of 66.04%.

#Sensitivity and Specificity:LR>Sensitivity ranges from 6.38% to 59.00% and Specificity ranges from 52.

#Balanced Accuracy:LR> Balanced accuracy ranges from 50.97% to 55.61% across different classes; RF>Bala

#In summary, Random Forest significantly outperforms Logistic Regression across all metrics, including
```

#10 FOLD CROSS VALIDATION FOR MULTINOMIAL LOGISTIC REGRESSION

```r
library(caret)
library(nnet) # For multinom model

# Create 10-fold cross-validation folds
folds <- createFolds(train$Delay_Severity, k = 10)

# Initialize an empty vector to store accuracies
accuracies <- numeric(length(folds))

# Perform 10-fold cross-validation
for (i in 1:length(folds)) {
  # Extract training and test data for current fold
  training_fold <- train[-folds[[i]], ]
  test_fold <- train[folds[[i]], ]

  # Train the multinomial logistic regression model
  multinom_model <- multinom(Delay_Severity ~ ., data = training_fold)
```

```r
  # Predict on the test set
  test_fold$predicted_severity <- predict(multinom_model, newdata = test_fold)

  # Generate the confusion matrix
  conf_matrix <- confusionMatrix(test_fold$predicted_severity, test_fold$Delay_Severity)

  # Calculate accuracy and store it
  accuracies[i] <- conf_matrix$overall['Accuracy']

  # Print accuracy of the current fold
  cat("Accuracy of Fold", i, ":", accuracies[i], "\n")
}
```

```
## # weights:  24 (14 variable)
## initial  value 126686.476068
## iter  10 value 123923.895344
## final  value 123455.379162
## converged
## Accuracy of Fold 1 : 0.3768342
## # weights:  24 (14 variable)
## initial  value 126684.278843
## iter  10 value 123926.416811
## final  value 123474.527740
## converged
## Accuracy of Fold 2 : 0.3786484
## # weights:  24 (14 variable)
## initial  value 126686.476068
## iter  10 value 123875.929773
## final  value 123406.715731
## converged
## Accuracy of Fold 3 : 0.373478
## # weights:  24 (14 variable)
## initial  value 126685.377455
## iter  10 value 123910.749168
## final  value 123451.336685
## converged
## Accuracy of Fold 4 : 0.3785218
## # weights:  24 (14 variable)
## initial  value 126684.278843
## iter  10 value 123956.223923
## final  value 123494.793969
## converged
## Accuracy of Fold 5 : 0.3784142
## # weights:  24 (14 variable)
## initial  value 126686.476068
## iter  10 value 123924.616394
## final  value 123460.134530
## converged
## Accuracy of Fold 6 : 0.3797221
## # weights:  24 (14 variable)
## initial  value 126686.476068
## iter  10 value 123922.356591
## final  value 123460.994152
```

```
## converged
## Accuracy of Fold 7 : 0.3814393
## # weights:  24 (14 variable)
## initial  value 126685.377455
## iter  10 value 123933.892139
## final  value 123472.695363
## converged
## Accuracy of Fold 8 : 0.3817217
## # weights:  24 (14 variable)
## initial  value 126686.476068
## iter  10 value 123900.008420
## final  value 123436.414882
## converged
## Accuracy of Fold 9 : 0.3726194
## # weights:  24 (14 variable)
## initial  value 126685.377455
## iter  10 value 123915.057927
## final  value 123449.051871
## converged
## Accuracy of Fold 10 : 0.3798486
```

```r
# Print accuracies of each fold
print(accuracies)
```

```
##  [1] 0.3768342 0.3786484 0.3734780 0.3785218 0.3784142 0.3797221 0.3814393
##  [8] 0.3817217 0.3726194 0.3798486
```

```r
# Cross-validation output for Logistic Regression dataset
LR_cv_output <- c(0.377, 0.379, 0.373, 0.379, 0.378, 0.380, 0.381, 0.382, 0.373, 0.380)

# Compute the mean of the cross-validation output
mean_LR_cv_output <- mean(LR_cv_output)

# Print the mean
print(mean_LR_cv_output)
```

```
## [1] 0.3782
```

```r
library(caret)
library(class)
# Excluding columns with NA names
Preprocessed_dataset <- Preprocessed_dataset[, !is.na(names(Preprocessed_dataset))]

# Splitting the dataset again after excluding NA named columns
set.seed(123) # Ensuring reproducibility

trainIndex <- createDataPartition(Preprocessed_dataset$Incident, p = .85,
                                  list = FALSE, times = 1)

train <- Preprocessed_dataset[trainIndex, ]
test <- Preprocessed_dataset[-trainIndex, ]
```

ALGORITHM 3: APPLYING KNN ALGORITHM ON TTC DELAY DATASET:

```
# Apply KNN
knn_pred <- knn(train = train[, -1], # Exclude the dependent variable from training data
                test = test[, -1],   # Exclude the dependent variable from testing data
                cl = train$Delay_Severity, # Training labels
                k = 10)              # Number of neighbors

# View the confusion matrix
confusionMatrix <- table(Actual = test$Delay_Severity, Predicted = knn_pred)
print(confusionMatrix)
```

```
##                                  Predicted
## Actual                            Borderline Late (<10 Min)
##   Borderline Late (<10 Min)                            3302
##   Considerably Late (11-15 Min)                        1835
##   Extremely Late (>15 Min)                             1367
##                                  Predicted
## Actual                            Considerably Late (11-15 Min)
##   Borderline Late (<10 Min)                                2086
##   Considerably Late (11-15 Min)                            4083
##   Extremely Late (>15 Min)                                 1981
##                                  Predicted
## Actual                            Extremely Late (>15 Min)
##   Borderline Late (<10 Min)                           1665
##   Considerably Late (11-15 Min)                       2086
##   Extremely Late (>15 Min)                            4204
```

CALCULATING ACCURACY, PRECISION AND RECALL FOR KNN ALGORITHM:

```
# Given confusion matrix values
confusionMatrix <- matrix(c(3302, 2086, 1665,
                            1835, 4083, 2086,
                            1367, 1981, 4204),
                          nrow = 3, byrow = TRUE,
                          dimnames = list(Actual = c("Borderline Late (<10 Min)", "Considerably Late (1
                                          Predicted = c("Borderline Late (<10 Min)", "Considerably Late

# Calculate sensitivity (recall) for each class
sensitivity_A <- confusionMatrix[1,1] / sum(confusionMatrix[1,])
sensitivity_B <- confusionMatrix[2,2] / sum(confusionMatrix[2,])
sensitivity_C <- confusionMatrix[3,3] / sum(confusionMatrix[3,])

# Calculate specificity for each class
total = sum(confusionMatrix)
true_negatives_A <- total - sum(confusionMatrix[1,]) - sum(confusionMatrix[,1]) + confusionMatrix[1,1]
specificity_A <- true_negatives_A / (total - sum(confusionMatrix[1,]))
```

```r
true_negatives_B <- total - sum(confusionMatrix[2,]) - sum(confusionMatrix[,2]) + confusionMatrix[2,2]
specificity_B <- true_negatives_B / (total - sum(confusionMatrix[2,]))

true_negatives_C <- total - sum(confusionMatrix[3,]) - sum(confusionMatrix[,3]) + confusionMatrix[3,3]
specificity_C <- true_negatives_C / (total - sum(confusionMatrix[3,]))

# Calculate overall accuracy
accuracy <- sum(diag(confusionMatrix)) / sum(confusionMatrix)

# Print the results
print(paste("Sensitivity for Borderline Late (<10 Min):", sensitivity_A))
```

```
## [1] "Sensitivity for Borderline Late (<10 Min): 0.468169573231249"
```

```r
print(paste("Sensitivity for Considerably Late (11-15 Min):", sensitivity_B))
```

```
## [1] "Sensitivity for Considerably Late (11-15 Min): 0.510119940029985"
```

```r
print(paste("Sensitivity for Extremely Late (>15 Min):", sensitivity_C))
```

```
## [1] "Sensitivity for Extremely Late (>15 Min): 0.556673728813559"
```

```r
print(paste("Specificity for Borderline Late (<10 Min):", specificity_A))
```

```
## [1] "Specificity for Borderline Late (<10 Min): 0.794163023913602"
```

```r
print(paste("Specificity for Considerably Late (11-15 Min):", specificity_B))
```

```
## [1] "Specificity for Considerably Late (11-15 Min): 0.721533721328312"
```

```r
print(paste("Specificity for Extremely Late (>15 Min):", specificity_C))
```

```
## [1] "Specificity for Extremely Late (>15 Min): 0.750879989373713"
```

```r
print(paste("Overall Accuracy:", accuracy))
```

```
## [1] "Overall Accuracy: 0.512583484453094"
```

```r
#KNN Output Comparison with Random Forest:

#Overall Accuracy: Random Forest achieved an overall accuracy of 66.04%, significantly outperforming KN

#Sensitivity and Specificity:Across all classes, Random Forest exhibits higher sensitivity and specific
#For Borderline Late (<10 Min):Random Forest Sensitivity: 58.68% vs. KNN Sensitivity: 46.82%.Random For

#For Considerably Late (11-15 Min):Random Forest Sensitivity: 62.48% vs. KNN Sensitivity: 51.01%. Rando
```

```
#For Extremely Late (>15 Min):Random Forest Sensitivity: 76.60% vs. KNN Sensitivity: 55.67%. Random For

#Predictive Values:Random Forest also demonstrates higher positive predictive values (PPV) and negative

#Balanced Accuracy:Balanced accuracy values are generally higher for Random Forest across all classes c
```

10 FOLD CROSS VALIDATION FOR KNN ALGORITHM:

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

# Excluding columns with NA names
Preprocessed_dataset <- Preprocessed_dataset[, !is.na(names(Preprocessed_dataset))]

# Splitting the dataset again after excluding NA named columns
set.seed(123) # Ensuring reproducibility
folds <- createFolds(Preprocessed_dataset$Delay_Severity, k = 10)

# Initialize an empty vector to store accuracies
accuracies <- numeric(length(folds))

# Perform 10-fold cross-validation
for (i in seq_along(folds)) {
  # Extract training and test data for current fold
  train_fold <- Preprocessed_dataset[-folds[[i]], ]
  test_fold <- Preprocessed_dataset[folds[[i]], ]

  # Apply KNN
  knn_pred <- knn(train = train_fold[, -1],  # Exclude the dependent variable from training data
                  test = test_fold[, -1],    # Exclude the dependent variable from testing data
                  cl = train_fold$Delay_Severity, # Training labels
                  k = 10)                    # Number of neighbors

  # Calculate accuracy
  accuracy <- mean(knn_pred == test_fold$Delay_Severity)

  # Store accuracy of the current fold
  accuracies[i] <- accuracy

  # Print accuracy of the current fold
  cat("Accuracy of Fold", i, ":", accuracy, "\n")
}
```

```
## Accuracy of Fold 1 : 0.5069993
## Accuracy of Fold 2 : 0.5089884
## Accuracy of Fold 3 : 0.5130366
## Accuracy of Fold 4 : 0.5090559
## Accuracy of Fold 5 : 0.5043121
## Accuracy of Fold 6 : 0.5101499
## Accuracy of Fold 7 : 0.5132356
```

```
## Accuracy of Fold 8 : 0.5063686
## Accuracy of Fold 9 : 0.514263
## Accuracy of Fold 10 : 0.5119411
```

```r
# Print accuracies of each fold
print(accuracies)
```

```
##  [1] 0.5069993 0.5089884 0.5130366 0.5090559 0.5043121 0.5101499 0.5132356
##  [8] 0.5063686 0.5142630 0.5119411
```

```r
# Cross-validation output for KNN Algorithm dataset
KNN_cv_output <- c(0.507, 0.509, 0.513, 0.509, 0.504, 0.510, 0.513, 0.506, 0.514, 0.512)

# Compute the mean of the cross-validation output
mean_KNN_cv_output <- mean(KNN_cv_output)

# Print the mean
print(mean_KNN_cv_output)
```

```
## [1] 0.5097
```

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.1
```

```
## Loaded gbm 2.1.9
```

```
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.c
```

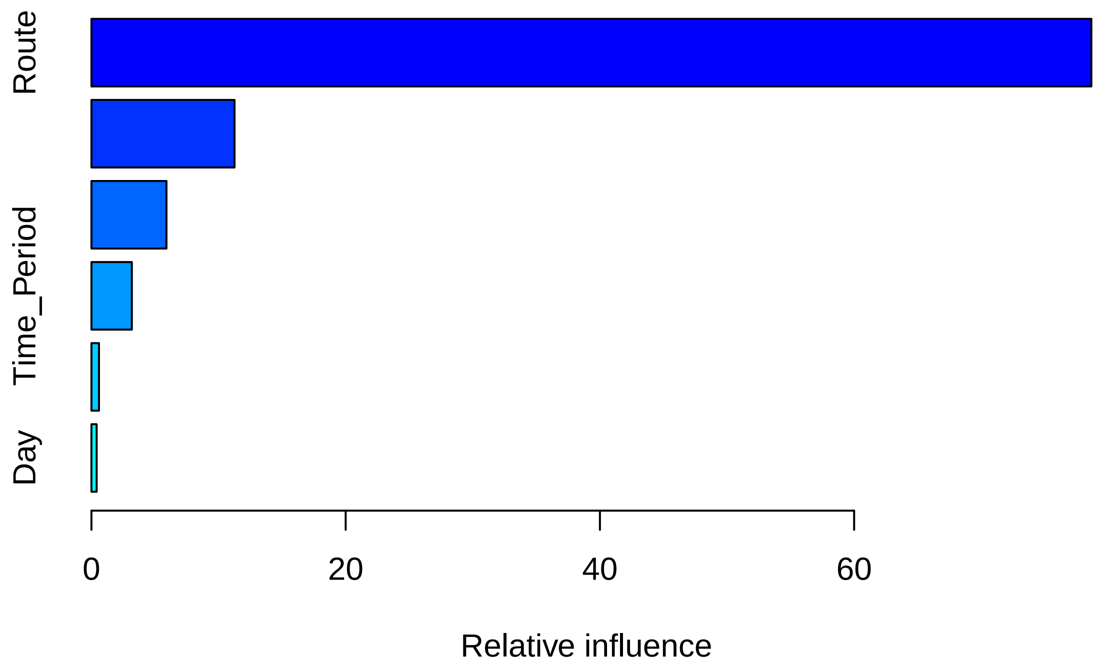
```r
library(caret)
```

```r
set.seed(123) # For reproducibility

# Training the GBM model
gbm_model <- gbm(Delay_Severity ~ .,
                 data = train,
                 distribution = "multinomial",
                 n.trees = 100,
                 interaction.depth = 3,
                 shrinkage = 0.1,
                 n.minobsinnode = 10,
                 cv.folds = 5)
```

```
## Warning: Setting `distribution = "multinomial"` is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```r
# Summarize the model
summary(gbm_model)
```




```
##                    var    rel.inf
## Route            Route 78.6613477
## Vehicle        Vehicle 11.2560785
## Incident      Incident  5.9015552
## Time_Period Time_Period  3.1779548
## Direction    Direction  0.5936231
## Day                Day  0.4094405
```

```r
# Predicting on the test set
predictions <- predict(gbm_model, newdata = test, n.trees = 100, type = "response")

# Converting probabilities to factor levels
max_probs <- apply(predictions, 1, which.max)
predicted_labels <- factor(max_probs, labels = levels(train$Delay_Severity))

# Generate the confusion matrix
conf_matrix <- confusionMatrix(predicted_labels, test$Delay_Severity)

# Print the confusion matrix and statistics
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##                                  Reference
## Prediction                        Borderline Late (<10 Min)
```

```
##   Borderline Late (<10 Min)                          3347
##   Considerably Late (11-15 Min)                      2214
##   Extremely Late (>15 Min)                           1492
##                                  Reference
## Prediction                        Considerably Late (11-15 Min)
##   Borderline Late (<10 Min)                              1158
##   Considerably Late (11-15 Min)                          4966
##   Extremely Late (>15 Min)                               1880
##                                  Reference
## Prediction                        Extremely Late (>15 Min)
##   Borderline Late (<10 Min)                           554
##   Considerably Late (11-15 Min)                      1555
##   Extremely Late (>15 Min)                           5443
##
## Overall Statistics
##
##                Accuracy : 0.6084
##                  95% CI : (0.602, 0.6148)
##     No Information Rate : 0.354
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4096
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Borderline Late (<10 Min)
## Sensitivity                                    0.4745
## Specificity                                    0.8899
## Pos Pred Value                                 0.6616
## Neg Pred Value                                 0.7888
## Prevalence                                     0.3120
## Detection Rate                                 0.1480
## Detection Prevalence                           0.2238
## Balanced Accuracy                              0.6822
##                      Class: Considerably Late (11-15 Min)
## Sensitivity                                        0.6204
## Specificity                                        0.7419
## Pos Pred Value                                     0.5685
## Neg Pred Value                                     0.7810
## Prevalence                                         0.3540
## Detection Rate                                     0.2196
## Detection Prevalence                               0.3864
## Balanced Accuracy                                  0.6812
##                      Class: Extremely Late (>15 Min)
## Sensitivity                                   0.7207
## Specificity                                   0.7761
## Pos Pred Value                                0.6175
## Neg Pred Value                                0.8471
## Prevalence                                    0.3340
## Detection Rate                                0.2407
## Detection Prevalence                          0.3899
## Balanced Accuracy                             0.7484
```

```
#Comparison- GBM Vs RF:

#Overall Accuracy: Random Forest achieves an accuracy of 66.04%, outperforming GBM's accuracy of 60.84%

#Sensitivity and Specificity:Sensitivity values for Random Forest range from 58.68% to 76.60% across di

#Positive Predictive Values (PPV):Random Forest's PPV ranges from 64.20% to 69.69%, and GBM's PPV range
#Negative Predictive Values (NPV):Random Forest's NPV ranges from 79.85% to 87.01%, and GBM's NPV range

#Balanced Accuracy:Random Forest achieves balanced accuracy values ranging from 73.57% to 77.96% across

#In summary, Random Forest generally outperforms GBM in terms of overall accuracy, sensitivity, PPV, NP
```

10 FOLD CROSS-VALIDATION FOR GBM ALGORITHM

```r
library(caret)
library(gbm)

# Set seed for reproducibility
set.seed(123)

# Create 10-fold cross-validation folds
folds <- createFolds(Preprocessed_dataset$Delay_Severity, k = 10)

# Initialize an empty vector to store accuracies
accuracies <- numeric(length(folds))

# Perform 10-fold cross-validation
for (i in seq_along(folds)) {
  # Extract training and test data for current fold
  train_fold <- Preprocessed_dataset[-folds[[i]], ]
  test_fold <- Preprocessed_dataset[folds[[i]], ]

  # Train the GBM model
  gbm_model <- gbm(Delay_Severity ~ .,
                   data = train_fold,
                   distribution = "multinomial",
                   n.trees = 100,
                   interaction.depth = 3,
                   shrinkage = 0.1,
                   n.minobsinnode = 10)

  # Predict on the test set
  predictions <- predict(gbm_model, newdata = test_fold, n.trees = 100, type = "response")

  # Convert probabilities to factor levels
  max_probs <- apply(predictions, 1, which.max)
  predicted_labels <- factor(max_probs, labels = levels(train_fold$Delay_Severity))

  # Calculate accuracy
  accuracy <- mean(predicted_labels == test_fold$Delay_Severity)
```

```r
  # Store accuracy of the current fold
  accuracies[i] <- accuracy

  # Print accuracy of the current fold
  cat("Accuracy of Fold", i, ":", accuracy, "\n")
}
```

```
## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 1 : 0.5995489

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 2 : 0.6072968

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 3 : 0.6105619

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 4 : 0.6055198

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 5 : 0.6052806

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 6 : 0.6028261

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 7 : 0.6142772
```

```
## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 8 : 0.60946

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 9 : 0.6088629

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.

## Accuracy of Fold 10 : 0.6040202
```

```r
# Print accuracies of each fold
print(accuracies)
```

```
##  [1] 0.5995489 0.6072968 0.6105619 0.6055198 0.6052806 0.6028261 0.6142772
##  [8] 0.6094600 0.6088629 0.6040202
```

```r
# Cross-validation output for GBM Algorithm dataset
GBM_cv_output <- c(0.600, 0.607, 0.610, 0.606, 0.605, 0.603, 0.614, 0.609, 0.609, 0.604)

# Compute the mean of the cross-validation output
mean_GBM_cv_output <- mean(GBM_cv_output)

# Print the mean
print(mean_GBM_cv_output)
```

```
## [1] 0.6067
```

#MODEL SELECTION:

```r
# Based on the comparative analysis, Random Forest Classification model was the clear winner across Acc

#In summary, Random Forest classification model was a better fit classification model over GBM, LR, and
```

OVERSAMPLING DATASET TO CHECK ON MODEL IMPROVEMENT

```r
library(shiny)
```

```
## Warning: package ’shiny’ was built under R version 4.3.1
```

```r
# Assuming 'Preprocessed_dataset' is your preprocessed dataset
# Replace 'Preprocessed_dataset' with the name of your dataset
dataset <- train

# UI for downloading the dataset
ui <- fluidPage(
  downloadButton("downloadData", "Download Preprocessed Dataset")
)

# Server function
server <- function(input, output) {
  output$downloadData <- downloadHandler(
    filename = function() {
      "train.csv"  # Change the filename as needed
    },
    content = function(file) {
      write.csv(dataset, file, row.names = FALSE)
    }
  )
}

# Run the Shiny application
shinyApp(ui, server)
```

#RELOAD OVERSAMPLED BALANCED DATASET

```r
# Load the TTC Delay data readxl package:
library(readxl)

# Read the Excel file
Oversampled_Dataset <- read_excel("train_Oversampling.xlsx")
Oversampled_Dataset$Delay_Severity <- as.factor(Oversampled_Dataset$Delay_Severity)
# View the data (optional)
str(Oversampled_Dataset)
```

```
## tibble [125,590 x 7] (S3: tbl_df/tbl/data.frame)
##  $ Delay_Severity: Factor w/ 3 levels "Borderline Late (<10 Min)",..: 3 3 1 1 3 3 3 3 3 2 ...
##  $ Route         : num [1:125590] 0.0681 0.034 0.0841 0.039 0.3353 ...
##  $ Day           : num [1:125590] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ Incident      : num [1:125590] 0.333 0 0.333 0.333 0 ...
##  $ Direction     : num [1:125590] 0.75 0.5 0.5 1 0.5 0.25 1 1 0.25 0.5 ...
##  $ Vehicle       : num [1:125590] 0.0849 0.0106 0.0157 0 0.0931 ...
##  $ Time_Period   : num [1:125590] 0.667 0.667 0.667 0.667 0.667 ...
```

```r
# Identifying Presence of Missing Data
missing_values1 <- colSums(is.na(Oversampled_Dataset))

# Print the count of missing values for each column
print(missing_values1)
```

```
## Delay_Severity          Route            Day       Incident      Direction
##              0              0              0              0              0
##        Vehicle    Time_Period
##              0              0
```

CHECK RANDOM FOREST MODEL ACCURACY IMPROVEMENT OPPORTUNITY THROUGH OVERSAMPLING:

```r
# Split the data into training and testing sets
train1 <- Oversampled_Dataset
test1 <- test
```

```r
library(randomForest)
library(caret)
# Train the Random Forest model
rf_model1 <- randomForest(Delay_Severity ~ ., data = train1, ntree = 100)

# Print the model summary
print(rf_model1)
```

```
##
## Call:
##  randomForest(formula = Delay_Severity ~ ., data = train1, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 33.14%
## Confusion matrix:
##                               Borderline Late (<10 Min)
## Borderline Late (<10 Min)                         28519
## Considerably Late (11-15 Min)                      8247
## Extremely Late (>15 Min)                           4481
##                               Considerably Late (11-15 Min)
## Borderline Late (<10 Min)                              7390
## Considerably Late (11-15 Min)                         25504
## Extremely Late (>15 Min)                               6045
##                               Extremely Late (>15 Min) class.error
## Borderline Late (<10 Min)                         6648   0.3298635
## Considerably Late (11-15 Min)                     8807   0.4007237
## Extremely Late (>15 Min)                         29949   0.2600618
```

```r
# Predict on the test set
predictions1 <- predict(rf_model1, newdata = test1)

# Assuming caret package is installed for confusionMatrix
library(caret)

# Generate the confusion matrix
conf_matrix1 <- confusionMatrix(predictions1, test1$Delay_Severity)

# Print the confusion matrix
print(conf_matrix1)
```

```
## Confusion Matrix and Statistics
##
```

```
##                                Reference
## Prediction                      Borderline Late (<10 Min)
##   Borderline Late (<10 Min)                          4921
##   Considerably Late (11-15 Min)                      1041
##   Extremely Late (>15 Min)                           1091
##                                Reference
## Prediction                      Considerably Late (11-15 Min)
##   Borderline Late (<10 Min)                              1092
##   Considerably Late (11-15 Min)                          5617
##   Extremely Late (>15 Min)                               1295
##                                Reference
## Prediction                      Extremely Late (>15 Min)
##   Borderline Late (<10 Min)                          660
##   Considerably Late (11-15 Min)                      809
##   Extremely Late (>15 Min)                          6083
##
## Overall Statistics
##
##                Accuracy : 0.7351
##                  95% CI : (0.7293, 0.7409)
##     No Information Rate : 0.354
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6023
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Borderline Late (<10 Min)
## Sensitivity                                    0.6977
## Specificity                                    0.8874
## Pos Pred Value                                 0.7374
## Neg Pred Value                                 0.8662
## Prevalence                                     0.3120
## Detection Rate                                 0.2177
## Detection Prevalence                           0.2951
## Balanced Accuracy                              0.7925
##                      Class: Considerably Late (11-15 Min)
## Sensitivity                                        0.7018
## Specificity                                        0.8733
## Pos Pred Value                                     0.7522
## Neg Pred Value                                     0.8424
## Prevalence                                         0.3540
## Detection Rate                                     0.2484
## Detection Prevalence                               0.3303
## Balanced Accuracy                                  0.7876
##                      Class: Extremely Late (>15 Min)
## Sensitivity                                   0.8055
## Specificity                                   0.8415
## Pos Pred Value                                0.7183
## Neg Pred Value                                0.8961
## Prevalence                                    0.3340
## Detection Rate                                0.2691
```

```
## Detection Prevalence                           0.3746
## Balanced Accuracy                              0.8235
```

```
#The overall accuracy slightly improved after oversampling 'Borderline Late' data
```