Group 3: Exercise 3

2023-01-31

```
df1 = read.csv("D:/MMA Material/Term 3/Talent Analytics/preprocessed_file.csv")
df1$status <- ifelse(df1$latest_date > "2017-01-01", 1, 0)
df2 <- subset(df1, select = c(2,7,8,9,17,18,19,22,23))
df3 <- aggregate(application_number ~ ., data = df2, FUN = length)

date_counts <- table(df3$latest_date)
top_dates <- names(sort(date_counts, decreasing=TRUE))[1:5]
top_counts <- sort(date_counts, decreasing=TRUE)[1:20]
top_dates
```

Data Processing

```
## NULL
```

```
top_counts
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
```

```
df3$uspc_class = as.factor(df3$uspc_class)
df3$tc = as.factor(df3$tc)
df3$gender = as.factor(df3$gender)
df3$race = as.factor(df3$race)
df3$examiner_art_unit = as.factor(df3$examiner_art_unit)
df <- df3[ -c(1,2,3) ]
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
##
```

```
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
df$tenure_days <- scale(df$tenure_days)
```

```
table(df$status)
```

Checking for the imalance in the dataset

```
##
##     0     1
##  3575 39141
```

```
set.seed(123) # for reproducibility
index <- createDataPartition(df$status, p = 0.7, list = FALSE)
train_data <- df[index, ]
test_data <- df[-index, ]
fit <- glm(status ~ ., data = train_data, family = "binomial")
```

Splitting Data into Training and Testing

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

```
predictions <- predict(fit, newdata = test_data, type = "response")
predictions_binary <- ifelse(predictions > 0.5, 1, 0)
predictions_binary <- as.factor(predictions_binary)
test_data$status <- as.factor(test_data$status)
```

```
conf_matrix <- confusionMatrix(predictions_binary, test_data$status)
conf_matrix
```

Calculating Confusion Matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0   145    70
##          1   941 11658
##
```

```
##                Accuracy : 0.9211
##                  95% CI : (0.9163, 0.9257)
##     No Information Rate : 0.9152
##     P-Value [Acc > NIR] : 0.008566
##
##                   Kappa : 0.2005
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.13352
##             Specificity : 0.99403
##          Pos Pred Value : 0.67442
##          Neg Pred Value : 0.92531
##              Prevalence : 0.08475
##          Detection Rate : 0.01132
##    Detection Prevalence : 0.01678
##       Balanced Accuracy : 0.56377
##
##        'Positive' Class : 0
##
```

```
accuracy <- conf_matrix$overall[1]
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- 2 * ((precision * recall) / (precision + recall))

cat("Accuracy:", accuracy, "\n")
```

Calculating the Accuracy, Precision, Recall, and F1 Score

```
## Accuracy: 0.9211019
```

```
cat("Precision:", precision, "\n")
```

```
## Precision: 0.1335175
```

```
cat("Recall:", recall, "\n")
```

```
## Recall: 0.9940314
```

```
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.2354143
```

```
df4_0 = df[df$status==0 ,]
df4_1 = df[df$status==1 ,]


x<-c(3000,4000,5000,6000,7000,8000,9000,10000)
for (i in x)
{
  cat("i:",i,"\n")
  df4_1_new = sample_n(df4_1, 500+i)
  newdf <- rbind(df4_0, df4_1_new)
  newdf <- newdf[sample(1:nrow(newdf)), ]
  set.seed(123) # for reproducibility
  index <- createDataPartition(newdf$status, p = 0.8, list = FALSE)
  train_data <- newdf[index, ]
  test_data <- newdf[-index, ]
  fit <- glm(status ~ ., data = train_data, family = "binomial")
  predictions <- predict(fit, newdata = test_data, type = "response")
  predictions_binary <- ifelse(predictions > 0.5, 1, 0)
  predictions_binary <- as.factor(predictions_binary)
  test_data$status <- as.factor(test_data$status)

  # Calculate confusion matrix
  conf_matrix <- confusionMatrix(predictions_binary, test_data$status)

  # Print the accuracy, precision, recall, and F1 Score
  accuracy <- conf_matrix$overall[1]
  precision <- conf_matrix$byClass[1]
  recall <- conf_matrix$byClass[2]
  f1_score <- 2 * ((precision * recall) / (precision + recall))

  cat("Accuracy:", accuracy, "\n")
  cat("Precision:", precision, "\n")
  cat("Recall:", recall, "\n")
  cat("F1 Score:", f1_score, "\n")
  cat("______________________________\n")
}
```

There are far more instances of class 0 than class 1, which makes the status class imbalanced. This can lead to bias in the classification and give low precision for high accuracy. Therefore, we need to do undersampling and try using lesser instances of class 0. Through cross validation we find the best number of sample of class 0 to take in order to balance accuracy precision trade-off

```
## i: 3000
## Accuracy: 0.7265018
## Precision: 0.7137042
## Recall: 0.740413
## F1 Score: 0.7268133
## ______________________________
## i: 4000
## Accuracy: 0.7201238
## Precision: 0.5815603
## Recall: 0.8274725
## F1 Score: 0.6830574
## ______________________________
## i: 5000
## Accuracy: 0.7179063
## Precision: 0.4725434
## Recall: 0.8691006
## F1 Score: 0.6122157
## ______________________________
## i: 6000
## Accuracy: 0.7091811
## Precision: 0.407767
## Recall: 0.8771252
## F1 Score: 0.5567202
## ______________________________
## i: 7000
## Accuracy: 0.7354402
## Precision: 0.4057971
## Recall: 0.9072802
## F1 Score: 0.5607769
## ______________________________
## i: 8000
```

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

```
## Accuracy: 0.7701863
## Precision: 0.3842857
## Recall: 0.9276968
## F1 Score: 0.5434533
## ______________________________
## i: 9000
```

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

```
## Accuracy: 0.7678776
## Precision: 0.3379501
## Recall: 0.9318542
## F1 Score: 0.4960138
## ______________________________
## i: 10000
```

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

```
## Accuracy: 0.7811723
## Precision: 0.303989
## Recall: 0.947318
## F1 Score: 0.4602775
## ______________________________
```

```
df4_1_new = sample_n(df4_1, 4500)
newdf <- rbind(df4_0, df4_1_new)
```

```r
newdf <- newdf[sample(1:nrow(newdf)), ]
set.seed(123) # for reproducibility
index <- createDataPartition(newdf$status, p = 0.8, list = FALSE)
train_data <- newdf[index, ]
test_data <- newdf[-index, ]
fit <- glm(status ~ ., data = train_data, family = "binomial")
predictions <- predict(fit, newdata = test_data, type = "response")
predictions_binary <- ifelse(predictions > 0.5, 1, 0)
predictions_binary <- as.factor(predictions_binary)
test_data$status <- as.factor(test_data$status)
```

From the previous step, we deduce that the best split is around 4500 instances of class 0.

```r
# Calculate confusion matrix
conf_matrix <- confusionMatrix(predictions_binary, test_data$status)

# Print the accuracy, precision, recall, and F1 Score
accuracy <- conf_matrix$overall[1]
precision <- conf_matrix$byClass[1]
recall <- conf_matrix$byClass[2]
f1_score <- 2 * ((precision * recall) / (precision + recall))

cat("Accuracy:", accuracy, "\n")
```

Calculating the confusion matrix to find the accuracy, precision, recall, and F1 Score

```
## Accuracy: 0.7182663
```

```r
cat("Precision:", precision, "\n")
```

```
## Precision: 0.5788732
```

```r
cat("Recall:", recall, "\n")
```

```
## Recall: 0.8276243
```

```r
cat("F1 Score:", f1_score, "\n")
```

```
## F1 Score: 0.6812519
```

```r
library(pROC)
```

Creating the object for ROC Curve

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

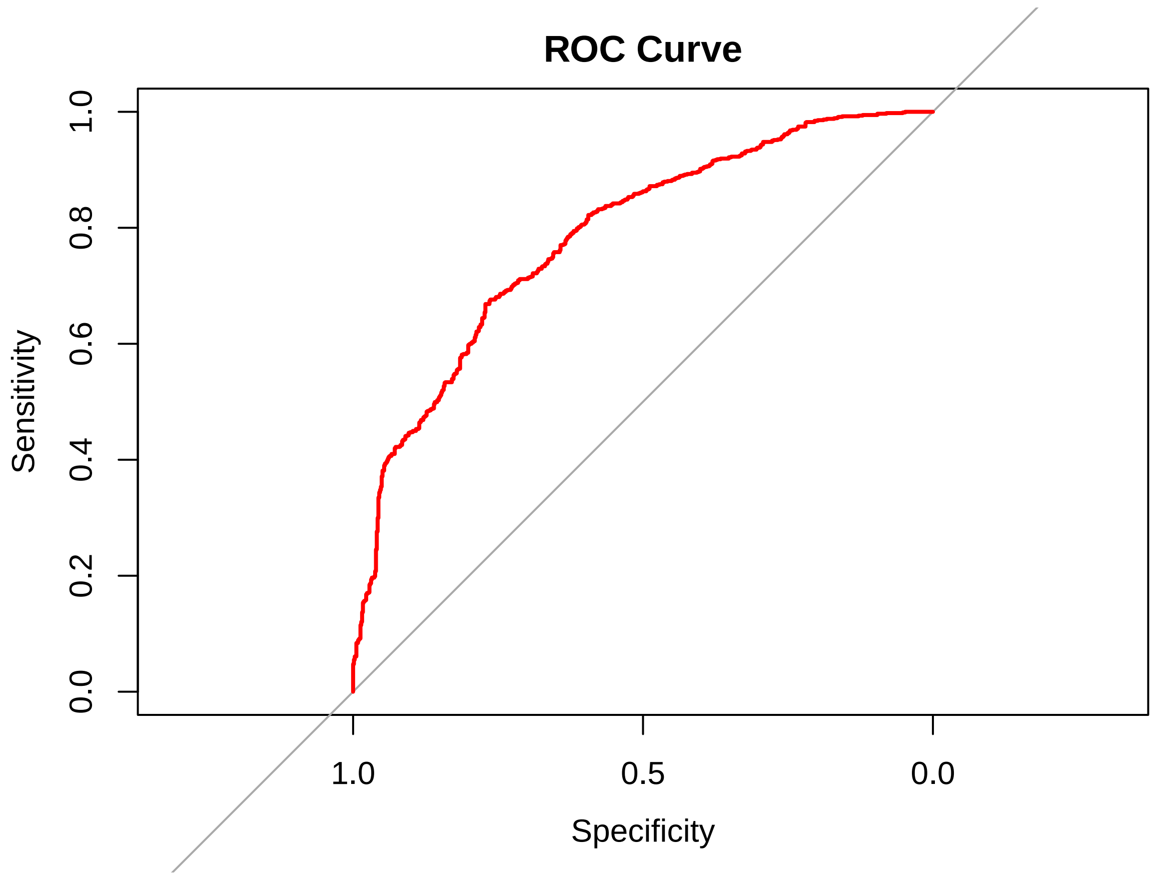
```
##
## Attaching package: 'pROC'

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

```
predictions <- predict(fit, newdata = test_data, type = "response")
predictions_numeric <- as.numeric(predictions)
test_data_status_numeric <- as.numeric(test_data$status) - 1
roc_obj <- roc(test_data_status_numeric, predictions)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



ROC Curve

```
auc_value <- auc(roc_obj)
cat("AUC:", auc_value, "\n")
```

Finding the AUC Value

```
## AUC: 0.7837009
```