**Lab Assignment-2**

library(dplyr)

library(data.tree)

library(randomForest)

library(caret) # For confusion matrix

# Load data

url <- "https://raw.githubusercontent.com/nasimm48/machine-learning/main/lab-2/data/oulad-assessments.csv"

data <- read.csv(url)

# Remove rows with NA values

data <- na.omit(data)

# Convert factors

data$code\_module <- as.factor(data$code\_module)

data$code\_presentation <- as.factor(data$code\_presentation)

data$assessment\_type <- as.factor(data$assessment\_type)

# Assuming 'score' needs to be converted to a categorical factor (e.g., High, Medium, Low)

# This is an example of binning the scores into three levels for classification

data$score <- cut(data$score, breaks=quantile(data$score, probs=0:3/3, na.rm=TRUE), include.lowest=TRUE, labels=c("Low", "Medium", "High"))

data$score <- as.factor(data$score)

set.seed(123) # for reproducibility

data\_set\_size <- floor(nrow(data) \* 0.80)

index <- sample(1:nrow(data), size = data\_set\_size)

training <- data[index, ]

testing <- data[-index, ]

# Fit random forest model for classification

rf <- randomForest(score ~ code\_module + code\_presentation + assessment\_type, data = training)

# Prediction and Result data frame

predictions <- predict(rf, newdata = testing, type = "response")

result <- data.frame(Actual = testing$score, Predicted = predictions)

# Generate and print confusion matrix

confusion\_matrix <- confusionMatrix(data = result$Predicted, reference = result$Actual)

print(confusion\_matrix)

# Optional: Create a graphical representation of the confusion matrix

library(ggplot2)

confusion\_df <- as.data.frame(confusion\_matrix$table)

confusion\_plot <- ggplot(confusion\_df, aes(x = Reference, y = Prediction, fill = Freq)) +

geom\_tile() +

geom\_text(aes(label = Freq), vjust = 1.5, color = "red") +

scale\_fill\_gradient(low = "white", high = "steelblue") +

ggtitle("Confusion Matrix") +

xlab("Actual Class") +

ylab("Predicted Class")

print(confusion\_plot)

confusion\_matrix

Console:

> library(dplyr)

> library(data.tree)

> library(randomForest)

> library(caret) # For confusion matrix

> # Load data

> url <- "https://raw.githubusercontent.com/nasimm48/machine-learning/main/lab-2/data/oulad-assessments.csv"

> data <- read.csv(url)

> # Remove rows with NA values

> data <- na.omit(data)

> # Convert factors

> data$code\_module <- as.factor(data$code\_module)

> data$code\_presentation <- as.factor(data$code\_presentation)

> data$assessment\_type <- as.factor(data$assessment\_type)

> # Assuming 'score' needs to be converted to a categorical factor (e.g., High, Medium, Low)

> # This is an example of binning the scores into three levels for classification

> data$score <- cut(data$score, breaks=quantile(data$score, probs=0:3/3, na.rm=TRUE), include.lowest=TRUE, labels=c("Low", "Medium", "High"))

> data$score <- as.factor(data$score)

> set.seed(123) # for reproducibility

> data\_set\_size <- floor(nrow(data) \* 0.80)

> index <- sample(1:nrow(data), size = data\_set\_size)

> training <- data[index, ]

> testing <- data[-index, ]

> # Fit random forest model for classification

> rf <- randomForest(score ~ code\_module + code\_presentation + assessment\_type, data = training)

Warning message:

In normalizePath(path.expand(path), winslash, mustWork) :

path[1]="": The system cannot find the path specified

> # Prediction and Result data frame

> predictions <- predict(rf, newdata = testing, type = "response")

> result <- data.frame(Actual = testing$score, Predicted = predictions)

> # Generate and print confusion matrix

> confusion\_matrix <- confusionMatrix(data = result$Predicted, reference = result$Actual)

> print(confusion\_matrix)

Confusion Matrix and Statistics

Reference

Prediction Low Medium High

Low 6686 4800 2915

Medium 1886 2312 2129

High 2881 4040 6526

Overall Statistics

Accuracy : 0.4543

95% CI : (0.449, 0.4595)

No Information Rate : 0.3386

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.1795

Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

Class: Low Class: Medium Class: High

Sensitivity 0.5838 0.20732 0.5640

Specificity 0.6605 0.82561 0.6938

Pos Pred Value 0.4643 0.36542 0.4853

Neg Pred Value 0.7589 0.68256 0.7567

Prevalence 0.3351 0.32632 0.3386

Detection Rate 0.1956 0.06765 0.1910

Detection Prevalence 0.4214 0.18514 0.3935

Balanced Accuracy 0.6221 0.51646 0.6289

> # Optional: Create a graphical representation of the confusion matrix

> library(ggplot2)

> confusion\_df <- as.data.frame(confusion\_matrix$table)

> confusion\_plot <- ggplot(confusion\_df, aes(x = Reference, y = Prediction, fill = Freq)) +

+ geom\_tile() +

+ geom\_text(aes(label = Freq), vjust = 1.5, color = "red") +

+ scale\_fill\_gradient(low = "white", high = "steelblue") +

+ ggtitle("Confusion Matrix") +

+ xlab("Actual Class") +

+ ylab("Predicted Class")

> print(confusion\_plot)

> confusion\_matrix

Confusion Matrix and Statistics

Reference

Prediction Low Medium High

Low 6686 4800 2915

Medium 1886 2312 2129

High 2881 4040 6526

Overall Statistics

Accuracy : 0.4543

95% CI : (0.449, 0.4595)

No Information Rate : 0.3386

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.1795

Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

Class: Low Class: Medium Class: High

Sensitivity 0.5838 0.20732 0.5640

Specificity 0.6605 0.82561 0.6938

Pos Pred Value 0.4643 0.36542 0.4853

Neg Pred Value 0.7589 0.68256 0.7567

Prevalence 0.3351 0.32632 0.3386

Detection Rate 0.1956 0.06765 0.1910

Detection Prevalence 0.4214 0.18514 0.3935

Balanced Accuracy 0.6221 0.51646 0.6289

Interpterion:

By using the confusion matrix and statics interpretation of the model performance

The confusion matrix shows the count the predicted classes. The model prediction LOW 6686 times when the class was low and the second row of the model predicted medium 2312 times of the actual class medium and the third-row model prediction is high with 6526 times with actual class.

And the overall statics accuracy of the predicted instance out of the total instances with approx. 45.43% with the confidence interval of 95% of the estimated accuracy. And the no information rate is 33.68% and the kappa measures the argument between actual prediction class with 0.1795.

And the statical the True Positive Rate the actual positive cases with the correctly positive predicted, True Negative Rate the and the actual negative were correctly predicted as negative and the property of negative predictions are the correct and the detection rate, detection prevalence, are the corrective prediction it shown right.

The model interpretation performance is predicting the “low” with the compared to the other two class however the overall accuracy is low and we can improve the model using the test and train.

Plot:





