MGT-665
Prof. Nasim
Krishna Guguloth

Lab 1

Results
(70,30)

```
# Load necessary libraries
> library(tidyr)
> library(caret)
>
> # Read the dataset
> data <- read.csv("C:/Users/dhara/Downloads/oulad-students.csv")
>
> # Data preprocessing
> # Modify variable selection as needed
> selected_columns <- c("gender", "region",
+                       "highest_education", "imd_band", "age_band", "num_of_prev_attempts",
+                       "disability", "final_result")
>
> data <- data[selected_columns]
>
> # Convert categorical variables to factors
> factor_columns <- c("gender", "region",
+                     "highest_education", "imd_band", "age_band", "num_of_prev_attempts",
+                     "disability", "final_result")
>
> data[factor_columns] <- lapply(data[factor_columns], as.factor)
>
> # Drop rows with missing values
> data <- na.omit(data)
>
> # Split the data into training and testing sets (70% training, 30% testing)
> set.seed(123) # For reproducibility
> train_index <- createDataPartition(data$final_result, p = 0.7, list = FALSE)
>
> # Train and test datasets
> train_data <- data[train_index, ]
> test_data <- data[-train_index, ]
>
> # Train the classification model (logistic regression)
> model <- glm(final_result ~ ., data = train_data, family = "binomial")
>
> # Make predictions on the test data (probabilities)
> predictions <- predict(model, newdata = test_data, type = "response")
>
> # Convert predicted probabilities to class labels
> predicted_classes <- ifelse(predictions > 0.5, "Pass", "Fail")  # Adjust the threshold
needed
>
> # Convert predicted_classes to factor with the same levels as test_data$final_result
> predicted_classes <- factor(predicted_classes, levels = levels(test_data$final_result))
>
> # Evaluate the model
> confusionMatrix(data = predicted_classes, reference = test_data$final_result)
Confusion Matrix and Statistics

             Reference
Prediction    Distinction Fail Pass Withdrawn
  Distinction           0    0    0         0
  Fail                  0    0    0         0
  Pass                847 2072 3549      2976
```

```
  Withdrawn             0    0    0         0

Overall Statistics

               Accuracy : 0.3758
                 95% CI : (0.366, 0.3857)
    No Information Rate : 0.3758
    P-Value [Acc > NIR] : 0.5039

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Distinction Class: Fail Class: Pass Class: Withdrawn
Sensitivity                     0.00000      0.0000      1.0000           0.0000
Specificity                     1.00000      1.0000      0.0000           1.0000
Pos Pred Value                      NaN         NaN      0.3758              NaN
Neg Pred Value                  0.91031      0.7806         NaN           0.6849
Prevalence                      0.08969      0.2194      0.3758           0.3151
Detection Rate                  0.00000      0.0000      0.3758           0.0000
Detection Prevalence            0.00000      0.0000      1.0000           0.0000
Balanced Accuracy               0.50000      0.5000      0.5000           0.5000

>
```

Results

(75,25)

```
# Load necessary libraries
> library(tidyr)
> library(caret)
>
> # Read the dataset
> data <- read.csv("C:/Users/dhara/Downloads/oulad-students.csv")
>
> # Data preprocessing
> # Modify variable selection as needed
> selected_columns <- c("gender", "region",
+                       "highest_education", "imd_band", "age_band", "num_of_prev_attempts",
+                       "disability", "final_result")
>
> data <- data[selected_columns]
>
> # Convert categorical variables to factors
> factor_columns <- c("gender", "region",
+                     "highest_education", "imd_band", "age_band", "num_of_prev_attempts",
+                     "disability", "final_result")
>
> data[factor_columns] <- lapply(data[factor_columns], as.factor)
>
> # Drop rows with missing values
> data <- na.omit(data)
>
> # Split the data into training and testing sets (75% training, 25% testing)
> set.seed(123) # For reproducibility
> train_index <- createDataPartition(data$final_result, p = 0.75, list = FALSE)
>
> # Train and test datasets
> train_data <- data[train_index, ]
> test_data <- data[-train_index, ]
>
> # Train the classification model (logistic regression)
> model <- glm(final_result ~ ., data = train_data, family = "binomial")
>
```

```
> # Make predictions on the test data (probabilities)
> predictions <- predict(model, newdata = test_data, type = "response")
>
> # Convert predicted probabilities to class labels
> predicted_classes <- ifelse(predictions > 0.5, "Pass", "Fail")  # Adjust the threshold as
 needed
>
> # Convert predicted_classes to factor with the same levels as test_data$final_result
> predicted_classes <- factor(predicted_classes, levels = levels(test_data$final_result))
>
> # Evaluate the model
> confusionMatrix(data = predicted_classes, reference = test_data$final_result)
Confusion Matrix and Statistics

             Reference
Prediction    Distinction Fail Pass Withdrawn
  Distinction           0    0    0         0
  Fail                  0    0    0         0
  Pass                706 1726 2957      2480
  Withdrawn             0    0    0         0

Overall Statistics

               Accuracy : 0.3758
                 95% CI : (0.3651, 0.3866)
    No Information Rate : 0.3758
    P-Value [Acc > NIR] : 0.5043

                  Kappa : 0

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Distinction Class: Fail Class: Pass Class: Withdrawn
Sensitivity                     0.00000      0.0000      1.0000           0.0000
Specificity                     1.00000      1.0000      0.0000           1.0000
Pos Pred Value                      NaN         NaN      0.3758              NaN
Neg Pred Value                  0.91028      0.7807         NaN           0.6848
Prevalence                      0.08972      0.2193      0.3758           0.3152
Detection Rate                  0.00000      0.0000      0.3758           0.0000
Detection Prevalence            0.00000      0.0000      1.0000           0.0000
Balanced Accuracy               0.50000      0.5000      0.5000           0.5000

>
```