Will it Rain Tomorrow in Australia?

Sofia Pope Trogu Fairouz Baz Radwan Auriane Mahfouz

17-07-2023

Contents

1 Import R Libraries

```
library(corrplot)
library(ggplot2)
library(caret)
library(magrittr)
library(gridExtra)
library(scales)
library(DMwR2)
library(UBL)
library(caret)
```

```
library(MASS)
library(ipred)
library(rsample)
library(mlr)
library(knitr)
library(glmnet)
library(outliers)
library(class)
library(kableExtra)
library(tidyverse)
library(pROC)
library(dplyr)
```

2 Data Preparation

2.1 Download the Rain Dataset

We downloaded the Rain Australia dataset as a CSV from the following link: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package/code?datasetId=6012&searchQuery=visual. The dataset is originally composed of 23 columns and 145,460 examples. The aim of the data is to use available information about today's weather, i.e. Temperature, Humidity, Pressure, to predict whether it will rain tomorrow. Therefore, we originally start with 22 features and 1 target variable, RainTomorrow with binary classification (Yes=1, No=0).

```
file_path <- "/Users/Sofia/Desktop/Rain_Australia/weatherAUS.csv"
rain <- read.csv(file_path)
head(rain)
```

```
##         Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir
## 1 2008-12-01   Albury    13.4    22.9      0.6          NA       NA           W
## 2 2008-12-02   Albury     7.4    25.1      0.0          NA       NA         WNW
## 3 2008-12-03   Albury    12.9    25.7      0.0          NA       NA         WSW
## 4 2008-12-04   Albury     9.2    28.0      0.0          NA       NA          NE
## 5 2008-12-05   Albury    17.5    32.3      1.0          NA       NA           W
## 6 2008-12-06   Albury    14.6    29.7      0.2          NA       NA         WNW
##   WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am
## 1            44          W        WNW           20           24          71
## 2            44        NNW        WSW            4           22          44
## 3            46          W        WSW           19           26          38
## 4            24         SE          E           11            9          45
## 5            41        ENE         NW            7           20          82
## 6            56          W          W           19           24          55
##   Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm
## 1          22      1007.7      1007.1        8       NA    16.9    21.8
## 2          25      1010.6      1007.8       NA       NA    17.2    24.3
## 3          30      1007.6      1008.7       NA        2    21.0    23.2
## 4          16      1017.6      1012.8       NA       NA    18.1    26.5
## 5          33      1010.8      1006.0        7        8    17.8    29.7
## 6          23      1009.2      1005.4       NA       NA    20.6    28.9
##   RainToday RainTomorrow
## 1        No           No
## 2        No           No
## 3        No           No
## 4        No           No
## 5        No           No
## 6        No           No
```

```
summary(rain)
```

```
##      Date              Location            MinTemp         MaxTemp      
##  Length:145460      Length:145460      Min.   :-8.50   Min.   :-4.80  
##  Class :character   Class :character   1st Qu.: 7.60   1st Qu.:17.90  
##  Mode  :character   Mode  :character   Median :12.00   Median :22.60  
##                                        Mean   :12.19   Mean   :23.22  
##                                        3rd Qu.:16.90   3rd Qu.:28.20  
##                                        Max.   :33.90   Max.   :48.10  
##                                        NA's   :1485    NA's   :1261   
##     Rainfall        Evaporation        Sunshine     WindGustDir       
##  Min.   :  0.000   Min.   :  0.00   Min.   : 0.00   Length:145460     
##  1st Qu.:  0.000   1st Qu.:  2.60   1st Qu.: 4.80   Class :character  
##  Median :  0.000   Median :  4.80   Median : 8.40   Mode  :character  
##  Mean   :  2.361   Mean   :  5.47   Mean   : 7.61                     
##  3rd Qu.:  0.800   3rd Qu.:  7.40   3rd Qu.:10.60                     
##  Max.   :371.000   Max.   :145.00   Max.   :14.50                     
##  NA's   :3261      NA's   :62790    NA's   :69835                     
##  WindGustSpeed     WindDir9am         WindDir3pm         WindSpeed9am    
##  Min.   :  6.00   Length:145460      Length:145460      Min.   :  0.00  
##  1st Qu.: 31.00   Class :character   Class :character   1st Qu.:  7.00  
##  Median : 39.00   Mode  :character   Mode  :character   Median : 13.00  
##  Mean   : 40.03                                        Mean   : 14.04  
##  3rd Qu.: 48.00                                        3rd Qu.: 19.00  
##  Max.   :135.00                                        Max.   :130.00  
##  NA's   :10263                                         NA's   :1767    
##   WindSpeed3pm    Humidity9am      Humidity3pm      Pressure9am    
##  Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 980.5  
##  1st Qu.:13.00   1st Qu.: 57.00   1st Qu.: 37.00   1st Qu.:1012.9  
##  Median :19.00   Median : 70.00   Median : 52.00   Median :1017.6  
##  Mean   :18.66   Mean   : 68.88   Mean   : 51.54   Mean   :1017.6  
##  3rd Qu.:24.00   3rd Qu.: 83.00   3rd Qu.: 66.00   3rd Qu.:1022.4  
##  Max.   :87.00   Max.   :100.00   Max.   :100.00   Max.   :1041.0  
##  NA's   :3062    NA's   :2654     NA's   :4507     NA's   :15065   
##   Pressure3pm        Cloud9am        Cloud3pm        Temp9am      
##  Min.   : 977.1   Min.   :0.00    Min.   :0.00    Min.   :-7.20  
##  1st Qu.:1010.4   1st Qu.:1.00    1st Qu.:2.00    1st Qu.:12.30  
##  Median :1015.2   Median :5.00    Median :5.00    Median :16.70  
##  Mean   :1015.3   Mean   :4.45    Mean   :4.51    Mean   :16.99  
##  3rd Qu.:1020.0   3rd Qu.:7.00    3rd Qu.:7.00    3rd Qu.:21.60  
##  Max.   :1039.6   Max.   :9.00    Max.   :9.00    Max.   :40.20  
##  NA's   :15028    NA's   :55888   NA's   :59358   NA's   :1767   
##     Temp3pm       RainToday         RainTomorrow      
##  Min.   :-5.40   Length:145460      Length:145460      
##  1st Qu.:16.60   Class :character   Class :character   
##  Median :21.10   Mode  :character   Mode  :character   
##  Mean   :21.68                                         
##  3rd Qu.:26.40                                         
##  Max.   :46.70                                         
##  NA's   :3609
```

```
dim(rain)
```

```
## [1] 145460     23
```

2.2 Variable Table of Contents

Table 1: Table of features with description and units.

Heading	Meaning	Units
Date	Day of Month	
Location	City in Austrailia	
MinTemp	Minimum temperature in the 24 hours to 9am.	degrees Celsius
MaxTemp	Maximum temperaure in the 24 hours from 9am.	degrees Celsius
Rainfall	Precipitation (rainfall) in the 24 hours to 9am.	millimetres
Evaporation	Class A pan evaporation in the 24 hours to 9am	millimetres
Sunshine	Bright sunshine in the 24 hours to midnight	hours
WindGustDir	Direction of strongest gust in the 24 hours to midnight	16 compass points
WindGustSpeed	Speed of strongest wind gust in the 24 hours to midnight	kilometres per hour
Temp9am	Temperature at 9am	degrees Celsius
Humidity9am	Relative humidity at 9am	percent
Cloud9am	Fraction of sky obscured by cloud at 9am	eighths
WindDir9am	Wind direction averaged over 10 minutes prior to 9am	16 compass points
WindSpeed9am	Wind speed averaged over 10 minutes prior to 9am	kilometres per hour
Pressure9am	Atmospheric pressure reduced to mean sea level at 9am	hectopascals
Temp3pm	Temperature at 3pm	degrees Celsius
Humidity3pm	Relative humidity at 3pm	percent
Cloud3pm	Fraction of sky obscured by cloud at 3pm	eighths
WindDir3pm	Wind direction averaged over 10 minutes prior to 3pm	16 compass points
WindSpeed3pm	Wind speed averaged over 10 minutes prior to 3pm	kilometres per hour
Pressure3pm	Atmospheric pressure reduced to mean sea level at 3pm	hectopascals
RainToday	Yes if the rain for today is 1mm or more, No if otherwise	1 if Yes- 0 if No
RainTomorrow	Yes if the rain for tomorrow is >= 1mm	1 if Yes- 0 if No

2.3 Data Pre-processing

Once we've loaded the dataset, we executed a series of pre-processing steps required prior to running our subsequent predictive models and data analysis. First, we found and removed any columns and rows that were completely empty. Next, we re-encoded the binary columns, RainToday and RainTomorrow to have numerical values: 1,0 instead of "Yes", "No", respectively. Then, we removed unneeded categorical variables, such as date, location, windgustdir, and winddir at 9 am and 3 pm. Finally, we reformated the dataframe to be all numeric values to be compatible with our further analysis methods. This process of data pre-processing led to a new dataset dimension of 56,420 × 18.

```
# Find Empty Columns
empty_columns <- which(colSums(is.na(rain)) == nrow(rain))
names_of_empty_col<- names(rain)[empty_columns]

dim(rain)
```

```
## [1] 145460     23
```

```
# Omit rows with NAs. We are left with 56,420 rows and 23 columns
rain <- na.omit(rain)

# Set Yes/No values to 1, 0, respectively for RainToday and RainTomorrow
rain$RainToday <- ifelse(rain$RainToday == "Yes", 1,
                                    ifelse(rain$RainToday == "No", 0,
                                           rain$RainToday))

#RainTomorrow is our Target variable
rain$RainTomorrow <- ifelse(rain$RainTomorrow == "Yes", 1,
                                    ifelse(rain$RainTomorrow == "No", 0,
                                           rain$RainToday))
```

```r
#Remove date, location columns, and wind direction columns
rain <- rain[, !(names(rain) %in% c('Date', 'Location', 'WindGustDir',
                                    'WindDir9am', 'WindDir3pm'))]

# New dimension of rain: 56,420 × 18
head(rain)
```

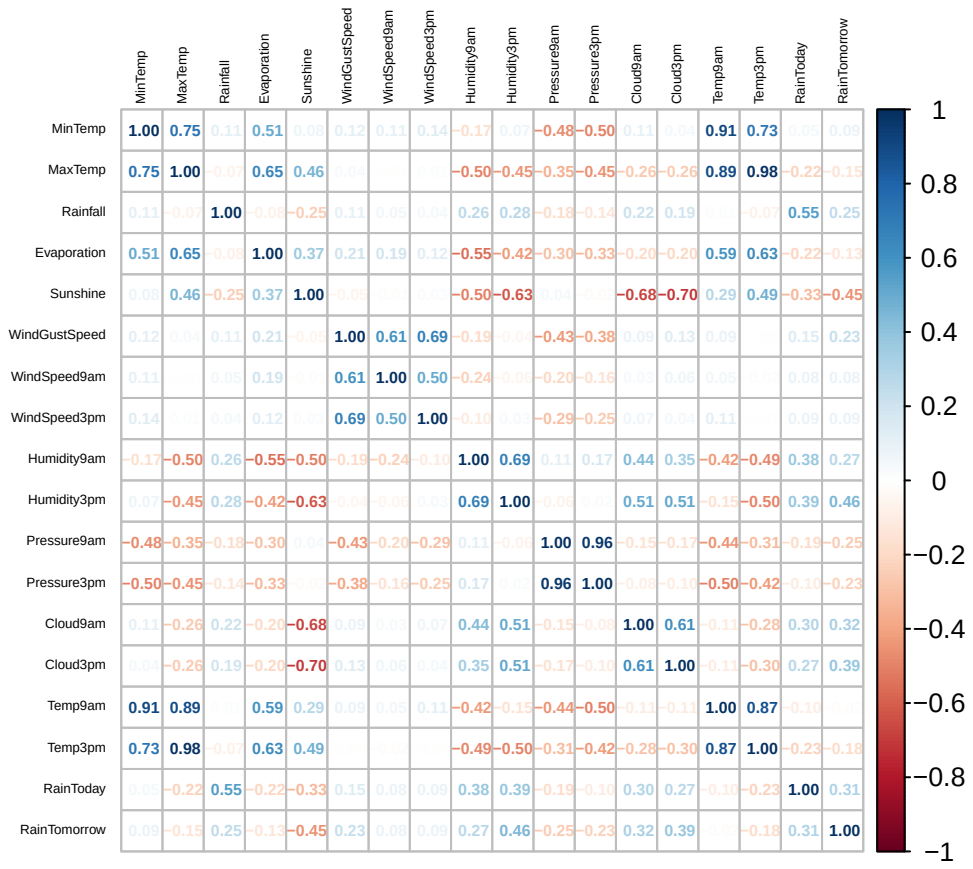
```
##      MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustSpeed WindSpeed9am
## 6050    17.9    35.2        0        12.0     12.3            48            6
## 6051    18.4    28.9        0        14.8     13.0            37           19
## 6053    19.4    37.6        0        10.8     10.6            46           30
## 6054    21.9    38.4        0        11.4     12.2            31            6
## 6055    24.2    41.0        0        11.2      8.4            35           17
## 6056    27.1    36.1        0        13.0      0.0            43            7
##      WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm Cloud9am
## 6050           20          20          13      1006.3      1004.4        2
## 6051           19          30           8      1012.9      1012.1        1
## 6053           15          42          22      1012.3      1009.2        1
## 6054            6          37          22      1012.7      1009.1        1
## 6055           13          19          15      1010.7      1007.4        1
## 6056           20          26          19      1007.7      1007.4        8
##      Cloud3pm Temp9am Temp3pm RainToday RainTomorrow
## 6050        5    26.6    33.4         0            0
## 6051        1    20.3    27.0         0            0
## 6053        6    28.7    34.9         0            0
## 6054        5    29.1    35.6         0            0
## 6055        6    33.6    37.6         0            0
## 6056        8    30.7    34.3         0            0
```

```r
rain <- as.data.frame(lapply(rain, as.numeric))
```

2.4 Correlation

After data pre-processing, we began to investigate the feature values and how each feature correlated with the other respective features. To perform this analysis, we built a correlation matrix that calculates the correlation values between each feature, so it results in a n_feature x n_feature matrix. From the correlation matrix, we visualized the correlations in both a heatmap and a numerical correlation plot.These plots provide a good context for the project and helped motivate our feature selection later on.

```r
# Build a Correlation Matrix
cor_matrix <- cor(rain)

#Plot correlation matrix with numerical values
corrplot <- corrplot(cor(rain[,-19]),
             method = "number",
             diag = TRUE,
             tl.cex = 0.4,
             number.cex = 0.5,
             tl.col = "black")
```



2.5 Density Plots

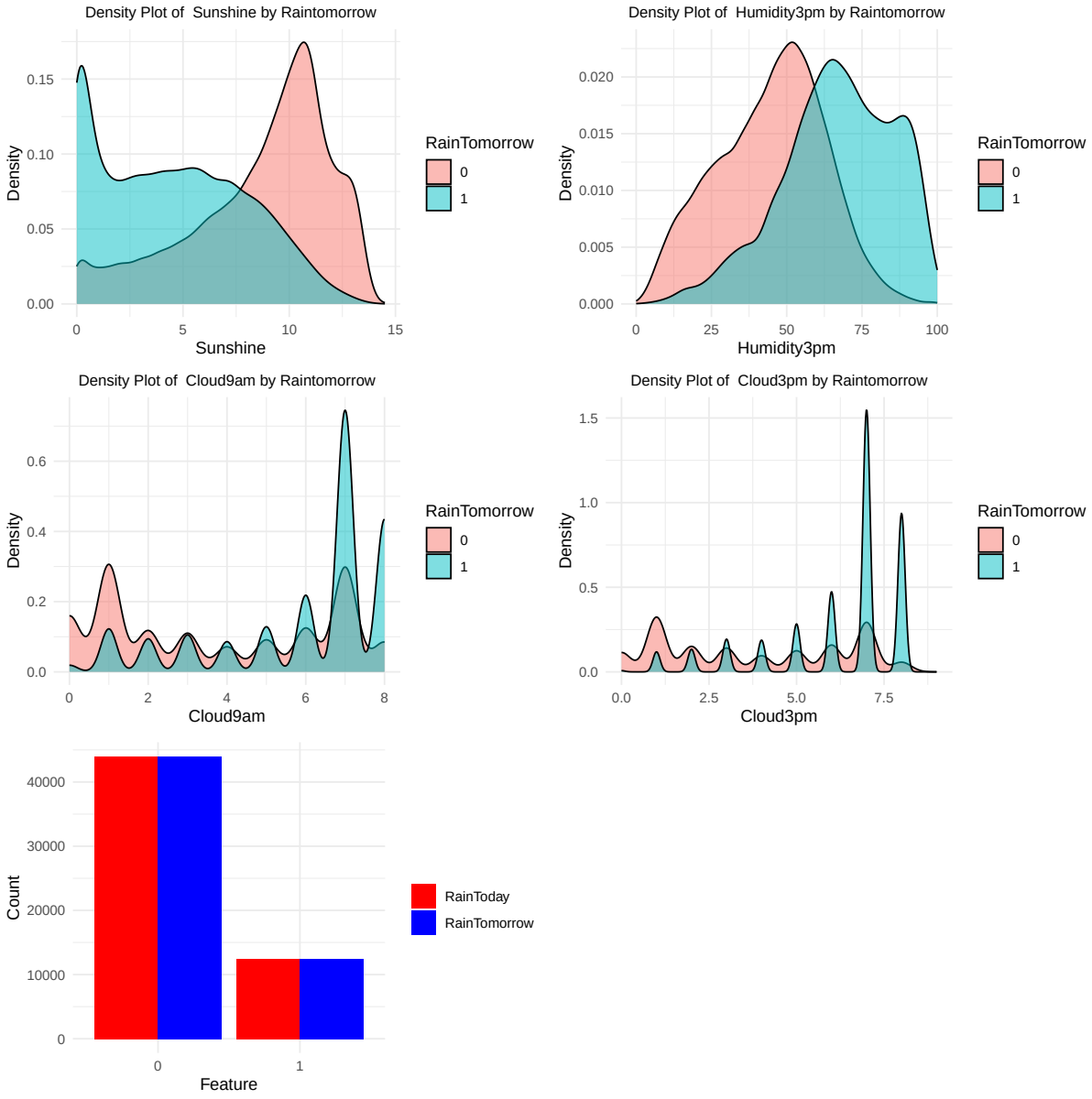
Based on the results of the correlation matrix, we further illustrated the features exhibiting the highest correlation using density and bar plots for the continuous and categorical variables, respectively against the target variable. From these plots, we identified the following trends: In the density plot with Sushine, we noted that the fraction of total days having higher sunshine are associated with more 0 RainTomorrow, and lower sunshine levels with more 1 RainTomorrow; Whereas, the Humidity3pm density plot appeared to have higher overlap between the two classes, but still higher humidity associated with 1 RainTomorrow and vice-versa. In the density plots with the two Cloud variables (9 am and 3 pm), we observed more oscillation across the x-axis with higher discrepancies in the target class at the two extremes. Finally, the representation of the RainToday binary variable against the target variable in a bar chart expresses the strong correlation between the classes of the two variables.We found a similar number of samples with both RainToday and RainTomorrow equal to 0 and vice-versa.

```
## Find features with highest correlation with target variable (RainTomorrow)

correlations <- cor_matrix['RainTomorrow',]
highly_correlated_columns <- correlations[abs(correlations) > 0.3 &
                                           correlations != 1]
column_names <- names(highly_correlated_columns)
print(column_names)
```

```
## [1] "Sunshine"    "Humidity3pm" "Cloud9am"    "Cloud3pm"    "RainToday"
```

```
rain_subset <- rain[,c(column_names)]

# We are trying to visualize relationship between Target Variable, RainTomorrow
# with the features having the highest correlation

# Calculate the count of each feature
```

```r
count_rain_today <- sum(rain$RainToday == 1)
count_no_rain_today <- sum(rain$RainToday == 0)
count_rain_tomorrow <- sum(rain$RainTomorrow == 1)
count_no_rain_tomorrow <- sum(rain$RainTomorrow == 0)

# Create a data frame with the counts
count_df <- data.frame(
  Feature = c("RainToday", "RainTomorrow", "RainToday","RainTomorrow"),
  Value = c("1", "1", "0", "0"),
  Count = c(count_rain_today, count_rain_tomorrow, count_no_rain_today,
            count_no_rain_tomorrow)
)

plot_list <- list()

for (col in column_names) {
  if (col == "RainToday") {
    # Plot the barplot
    bar_plot <- ggplot(count_df, aes(x = Value, y = Count, fill = Feature)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Feature", y = "Count", fill = "") +
    scale_fill_manual(values = c("red", "blue"), labels = c("RainToday",
                                                            "RainTomorrow")) +
    theme_minimal()
    plot_list <- append(plot_list, list(bar_plot))
  }
  else {
    density_plot <- rain%>% ggplot(aes(x = .data[[col]] ,
                                       fill = factor(RainTomorrow))) +
    geom_density(alpha = 0.5) +
    labs(x = col, y = "Density", fill = "RainTomorrow") +
    ggtitle(paste("Density Plot of ", col, "by Raintomorrow")) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5, size =10))
  plot_list <- append(plot_list, list(density_plot))
  }

}

# Visualize density and bar plots
grid.arrange(grobs = plot_list, nrow = 3, ncol = 2)
```



2.6 Feature Scaling

One additional pre-processing step that is vital to statistical learning methodologies is the application of feature scaling. Particularly when we are creating similarity measures, i.e. Euclidean distances, correlations, it is important to ensure your feature values fall within a similar range. There are different feature scaling techniques to standardize the range of feature values, but we chose min/max normalization. The min/max scaling takes each feature value, subtracts the minimum feature value and divides the result by the difference between the maximum and minimum values. In our project, we applied the min/max scaling to all the continuous variables, but kept the two binary variables, RainToday and RainTomorrow (our target) the same.

```
# Feature Scaling: Scale feature values using min/max scaling
min_max_norm <- function(x) {(x - min(x)) / (max(x) - min(x))}

rain_n <- as.data.frame(lapply(rain[,1:16], min_max_norm))

#Add back in Binary Features: RainToday and target variable, RainTomorrow
rain_n$RainToday <- rain$RainToday
rain_n$RainTomorrow <- rain$RainTomorrow
```

3 Regression Models

Our primary goal for this project was to build predictive models in order to implement a binary classification on our dataset. An intuitive approach to achieve that is the Logistic Regression model, which is a statistical modeling technique used to predict binary outcomes given a set of variables. In our report, we chose to implement several versions of this model, where we incorporated the pre-processing techniques discussed in this report. We first started by applying a logistic regression on the original dataset, split into a train and test sets, with neither balancing nor feature selection. We then implemented two other models, the first with balancing alone and the second with both balancing and feature selection. Furthermore, to clearly see how effective each model is, we ran predictions on the test set, plotted the confusion matrices, and visualized different performance metrics such as f1-score, accuracy and error. For each of these predictions, we tried out different threshold values (0.4,0.5,0.6) for the final classification.

3.1 First Logistic Regression Model: without balancing or feature selection

Prior to feeding our data to our model, we implemented the train/test split which is a common machine learning technique which involves splitting the original dataset into two subsets: the training set and the test set. The training set is usually made up of the majority of the dataset and is used for training, where the model learns the best hyperparameters, without playing any part in the evaluation phase. On the other hand, the test set is exclusively used for testing our model's performance at predicting classes.

```
#Train/Test Split

#Set Seed for Reproducibility
set.seed(123)

# Set Training Set Size to 75% of Total Dataset
train0 <- sample(1:nrow(rain_n), nrow(rain_n) * 0.75)

# Calculate the test indices
test0 <- setdiff(1:nrow(rain_n), train0)

# Split the target variable into train and test sets
rain_n_train0 <- rain_n[train0, ]
rain_n_test0 <- rain_n[test0 , ]

# Run GLM Logistic Regression Model using Training Set
glm_model0 <- glm(data = rain_n_train0,
                  rain_n_train0$RainTomorrow ~ .,
                  family = binomial)

# R squared and Variance Inflation Factor (VIF)

#R-squared (Coefficient of Determination): R-squared is a statistical metric
#that measures the proportion of the variance in the dependent variable
#(target variable) that can be explained by the independent variables
#(predictor variables) in a regression model. It provides an indication of how
#well the regression model fits the observed data.

# VIF: is a measure used to detect multicollinearity in a regression model.
# If the VIF value for a predictor variable is greater than 1, it indicates the
#presence of multicollinearity, suggesting that the predictor variable is
# highly correlated with other predictor variables in the model which makes it
#hard to assess the individual effects of each predictor on the dependent
#variable accurately.
```

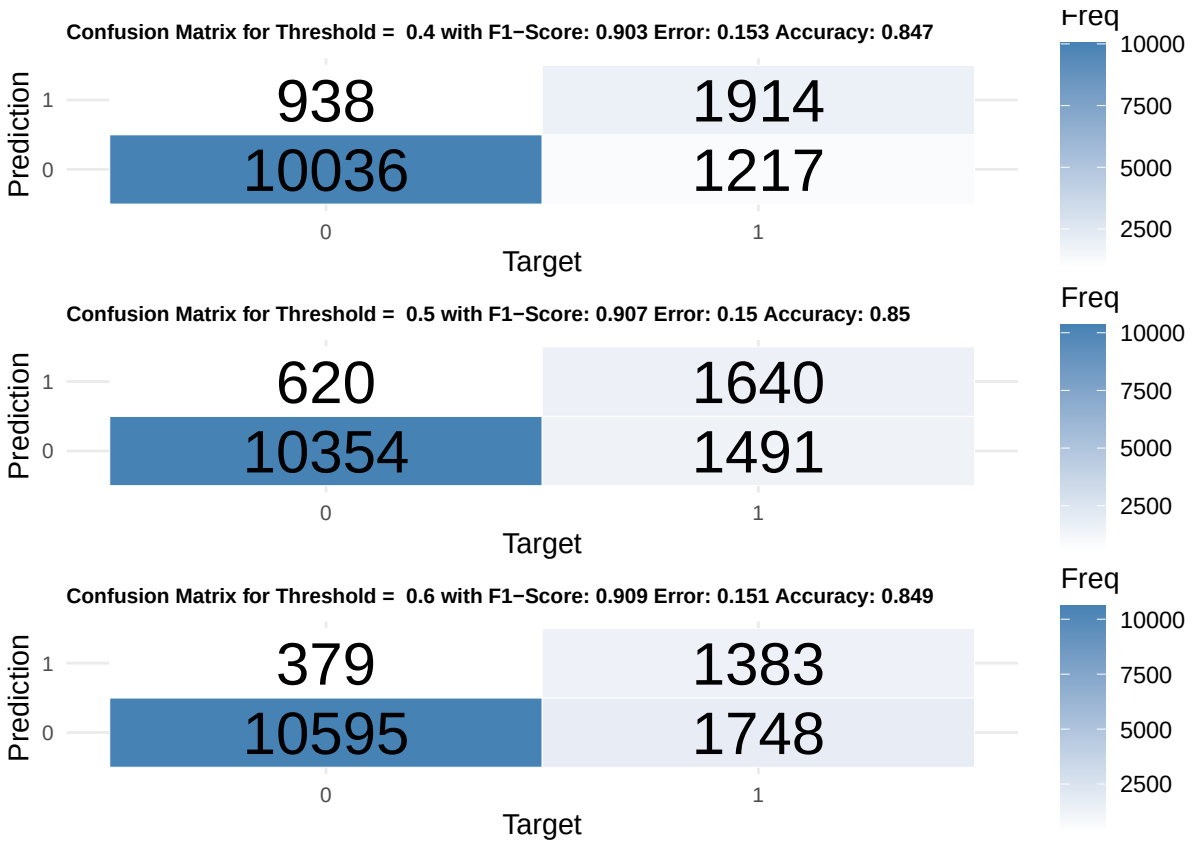
```r
model_summary0 <- summary(glm_model0)
summary(glm_model0)
```

```
##
## Call:
## glm(formula = rain_n_train0$RainTomorrow ~ ., family = binomial,
##     data = rain_n_train0)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0778  -0.5047  -0.2791  -0.1248   3.1273
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.27671    0.21755 -15.062  < 2e-16 ***
## MinTemp        -1.75799    0.32913  -5.341 9.23e-08 ***
## MaxTemp         0.90140    0.60726   1.484  0.13771
## Rainfall        2.00857    0.50309   3.992 6.54e-05 ***
## Evaporation    -0.68500    0.55112  -1.243  0.21390
## Sunshine       -2.12269    0.10235 -20.739  < 2e-16 ***
## WindGustSpeed   7.07457    0.21145  33.458  < 2e-16 ***
## WindSpeed9am   -0.77965    0.15902  -4.903 9.44e-07 ***
## WindSpeed3pm   -2.14228    0.18900 -11.335  < 2e-16 ***
## Humidity9am     0.25419    0.18214   1.396  0.16285
## Humidity3pm     5.62834    0.19259  29.225  < 2e-16 ***
## Pressure9am     8.41492    0.56060  15.011  < 2e-16 ***
## Pressure3pm   -12.37656    0.58288 -21.233  < 2e-16 ***
## Cloud9am       -0.10464    0.06917  -1.513  0.13036
## Cloud3pm        0.99910    0.08537  11.703  < 2e-16 ***
## Temp9am         1.58034    0.50982   3.100  0.00194 **
## Temp3pm        -0.28768    0.65560  -0.439  0.66080
## RainToday       0.47364    0.04201  11.273  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 44559  on 42314  degrees of freedom
## Residual deviance: 28080  on 42297  degrees of freedom
## AIC: 28116
##
## Number of Fisher Scoring iterations: 6
```

```r
r2_0 <-
  1 - (model_summary0$deviance / model_summary0$null.deviance) # 0.3698141
vif0 <- 1 / (1 - r2_0) # 1.586833

#Predict test using glm model
glm_predict0 <- predict(glm_model0, rain_n_test0, type = "response")

#Convert predictions into 0,1 based on different thresholds

threshold4 <- 0.4
threshold5 <- 0.5
threshold6 <- 0.6
```

```r
glm_predict_4_0 <- ifelse(glm_predict0 > threshold4, 1, 0)
glm_predict_5_0 <- ifelse(glm_predict0 > threshold5, 1, 0)
glm_predict_6_0 <- ifelse(glm_predict0 > threshold6, 1, 0)

# Function to create a confusion matrix with F1 score, error and accuracy rate
# A confusion matrix is a table that is often used to evaluate the performance
# of a classification model. It provides a summary of the predicted and actual
# values for a set of data points. The matrix allows us to assess how well the
# model is performing in terms of correctly and incorrectly classifying the data.
# The rows correspond to the predicted classes and the columns represent the
# actual targets.

create_confusion_matrix <-
  function(confusion_matrix,
           threshold,
           error,
           accuracy) {
    # Extract the confusion matrix table
    cm_table <- as.data.frame(confusion_matrix$table)

    #Extract F1 score
    f1_score <- confusion_matrix$byClass["F1"]

    # Plot the confusion matrix using ggplot2
    ggplot(data = cm_table, aes(x = Reference, y = Prediction, fill = Freq)) +
      geom_tile(color = "white") +
      geom_text(aes(label = Freq), color = "black", size = 8) +
      scale_fill_gradient(low = "white", high = "steelblue") +
      labs(
        title = paste(
          "Confusion Matrix for Threshold = ",
          threshold,
          "with F1-Score:",
          round(f1_score, 3),
          "Error:",
          round(error, 3) ,
          "Accuracy:",
          round(accuracy, 3)
        ),
        x = "Target",
        y = "Prediction"
      ) +
      theme_minimal() +
      theme(axis.text = element_text(size = 8),
            plot.title = element_text(size = 8, face = "bold"))

  }

# Confusion matrix with threshold = 0.4
error4_0 <- mean(glm_predict_4_0 != rain_n_test0$RainTomorrow)
accuracy4_0 <- mean(glm_predict_4_0 == rain_n_test0$RainTomorrow)
cm4_0 <-
  confusionMatrix(
    data = factor(glm_predict_4_0),
    reference = factor(rain_n_test0$RainTomorrow),
```

```
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.5
error5_0 <- mean(glm_predict_5_0 != rain_n_test0$RainTomorrow)
accuracy5_0 <- mean(glm_predict_5_0 == rain_n_test0$RainTomorrow)
cm5_0 <-
  confusionMatrix(
    data = factor(glm_predict_5_0),
    reference = factor(rain_n_test0$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.6
error6_0 <- mean(glm_predict_6_0 != rain_n_test0$RainTomorrow)
accuracy6_0 <- mean(glm_predict_6_0 == rain_n_test0$RainTomorrow)
cm6_0 <-
  confusionMatrix(
    data = factor(glm_predict_6_0),
    reference = factor(rain_n_test0$RainTomorrow),
    mode = 'everything'
  )

a0 <- create_confusion_matrix(cm4_0, 0.4, error4_0, accuracy4_0)
b0 <- create_confusion_matrix(cm5_0, 0.5, error5_0, accuracy5_0)
c0 <- create_confusion_matrix(cm6_0, 0.6, error6_0, accuracy6_0)

# Threshold of 0.05 is the best among thresholds in terms of accuracy,
#sensitivity, and specificity
cm_all0 = list(a0, b0, c0)
plot_width <- c(4, 4, 4)
grid.arrange(grobs = cm_all0,
             nrow = 3,
             width = plot_width)
```



3.2 Logistic Regression with Balancing

In addition to feature scaling, balancing our data is another technique used to prepare data before applying predictive modeling. Balancing reduces the effect of the dominating class which results in high bias and unreliable predictions. In order to balance data, there are several under-sampling and over-sampling methods that can be used depending on the characteristics of the dataset. As we will observe in the code below, our original data isn't balanced, in which 0 is the majority class with over 40,000 and 1 has only about 12,000 samples. To account for this imbalance, we performed a method of under-sampling, whereby we reduced the number of samples in the majority class (0) to the same number of samples of the minority class (1). This methodology may reduce the pool of data we have available, but it removes the bias in our model predictions toward one dominating class. We opted against using over-sampling as that might add unwanted noise to our data which might also lead to unreliable results.

```
# Check distribution of RainTomorrow values to see how balanced the data is
# 0: 43993; 1: 12427
table(rain$RainTomorrow)
```

```
##
##     0     1
## 43993 12427
```

```
#Downsamples majority class(0)
#Added yname to specify the target variable in downSample function, ow it
#assumes first col is target
rain_balanced <-
  downSample(x = rain_n[,-which(names(rain_n) == "RainTomorrow")],
             y = factor(rain_n$RainTomorrow),
             yname = "RainTomorrow")
table(rain_balanced$RainTomorrow)
```

```
##
##     0     1
## 12427 12427
```

```
head(rain_balanced)
```

```
##     MinTemp   MaxTemp    Rainfall Evaporation  Sunshine WindGustSpeed
## 1 0.6692913 0.4409091 0.007759457  0.06896552 0.1172414     0.2608696
## 2 0.8661417 0.6772727 0.000000000  0.08374384 0.7034483     0.2260870
## 3 0.4094488 0.5204545 0.000000000  0.06403941 0.5931034     0.2434783
## 4 0.7086614 0.6750000 0.000000000  0.09852217 0.6275862     0.3565217
## 5 0.5301837 0.6795455 0.000000000  0.11822660 0.6275862     0.2782609
## 6 0.6141732 0.4886364 0.000000000  0.08128079 0.7103448     0.2782609
##   WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm
## 1    0.2307692    0.2432432        0.64        0.71   0.6494157   0.6521036
## 2    0.2000000    0.3513514        0.72        0.59   0.5358932   0.5129450
## 3    0.1076923    0.2702703        0.72        0.45   0.6961603   0.6618123
## 4    0.3076923    0.2297297        0.60        0.45   0.4741235   0.4854369
## 5    0.1076923    0.1486486        0.32        0.18   0.5225376   0.5339806
## 6    0.2615385    0.2297297        0.76        0.45   0.5409015   0.5679612
##   Cloud9am  Cloud3pm   Temp9am   Temp3pm RainToday RainTomorrow
## 1    0.875 0.7777778 0.5810474 0.4363208         1            0
## 2    0.250 0.2222222 0.7605985 0.6816038         0            0
## 3    0.625 0.3333333 0.4738155 0.5188679         0            0
## 4    0.625 0.6666667 0.6832918 0.6698113         0            0
## 5    0.750 0.5555556 0.5960100 0.6297170         0            0
## 6    1.000 0.1111111 0.4588529 0.4740566         0            0
```

```
#Check if there are any NAs
sum(is.na(rain_balanced$RainTomorrow))
```

```
## [1] 0
```

```
#Test/Train Split
set.seed(123)

train_balanced <-
  sample(1:nrow(rain_balanced), nrow(rain_balanced) * 0.75)

# Calculate the test indices
test_balanced <- setdiff(1:nrow(rain_balanced), train_balanced)

# Split the target variable into train and test sets
rain_balanced_train <- rain_balanced[train_balanced, ]
rain_balanced_test <- rain_balanced[test_balanced , ]
glm_model_balanced <- glm(data = rain_balanced_train,
                          rain_balanced_train$RainTomorrow ~ .,
                          family = binomial)

model_summary_balanced <- summary(glm_model_balanced)
summary(glm_model_balanced)
```

```
##
## Call:
## glm(formula = rain_balanced_train$RainTomorrow ~ ., family = binomial,
##     data = rain_balanced_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -3.5064  -0.6447   0.0516   0.6418   2.8163
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.78587    0.28387  -6.291 3.15e-10 ***
## MinTemp        -1.45819    0.41703  -3.497 0.000471 ***
## MaxTemp         1.04710    0.84911   1.233 0.217513
## Rainfall        1.47028    0.77973   1.886 0.059346 .
## Evaporation    -1.42725    0.70721  -2.018 0.043577 *
## Sunshine       -2.57528    0.13469 -19.119  < 2e-16 ***
## WindGustSpeed   6.91036    0.29248  23.627  < 2e-16 ***
## WindSpeed9am   -0.64221    0.21128  -3.040 0.002369 **
## WindSpeed3pm   -1.72241    0.25163  -6.845 7.65e-12 ***
## Humidity9am     0.19478    0.23371   0.833 0.404623
## Humidity3pm     5.68413    0.25584  22.217  < 2e-16 ***
## Pressure9am     8.31613    0.75726  10.982  < 2e-16 ***
## Pressure3pm   -12.46658    0.79195 -15.742  < 2e-16 ***
## Cloud9am       -0.27995    0.08624  -3.246 0.001169 **
## Cloud3pm        1.07058    0.10368  10.325  < 2e-16 ***
## Temp9am         0.49761    0.65435   0.760 0.446979
## Temp3pm         0.73571    0.89965   0.818 0.413484
## RainToday       0.46663    0.05825   8.011 1.14e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25840  on 18639  degrees of freedom
## Residual deviance: 15893  on 18622  degrees of freedom
## AIC: 15929
##
## Number of Fisher Scoring iterations: 5
```

```
r2_balanced <-
  1 - (model_summary_balanced$deviance / model_summary_balanced$null.deviance)
# 0.3849359
vif_balanced <- 1 / (1 - r2_balanced) # 1.625847

#Predict test with model
glm_predict_balanced <-
  predict(glm_model_balanced, rain_balanced_test, type = "response")

#Convert predictions into 0,1 based on different thresholds

glm_predict_4_balanced <-
  ifelse(glm_predict_balanced > threshold4, 1, 0)
glm_predict_5_balanced <-
  ifelse(glm_predict_balanced > threshold5, 1, 0)
glm_predict_6_balanced <-
  ifelse(glm_predict_balanced > threshold6, 1, 0)

# Confusion matrix with threshold = 0.4
table(rain_balanced_test$RainTomorrow, glm_predict_4_balanced)
```

```
##    glm_predict_4_balanced
##        0    1
##   0 2335  798
```

```
##   1  446 2635
```

```
error4_balanced <-
  mean(glm_predict_4_balanced != rain_balanced_test$RainTomorrow)
accuracy4_balanced <-
  mean(glm_predict_4_balanced == rain_balanced_test$RainTomorrow)
cm4_balanced <-
  confusionMatrix(
    data = factor(glm_predict_4_balanced),
    reference = factor(rain_balanced_test$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.5
table(rain_balanced_test$RainTomorrow, glm_predict_5_balanced)
```

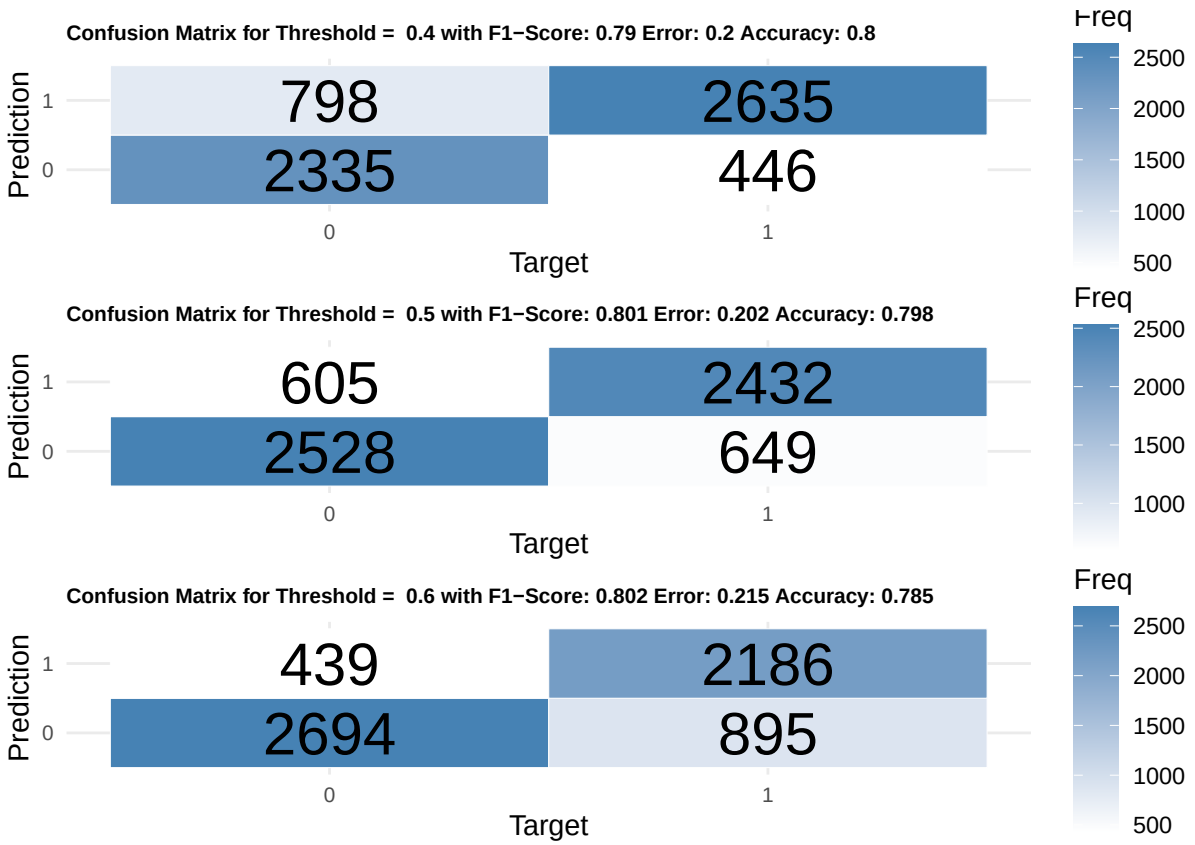
```
##    glm_predict_5_balanced
##        0    1
##   0 2528  605
##   1  649 2432
```

```
error5_balanced <-
  mean(glm_predict_5_balanced != rain_balanced_test$RainTomorrow)
accuracy5_balanced <-
  mean(glm_predict_5_balanced == rain_balanced_test$RainTomorrow)
cm5_balanced <-
  confusionMatrix(
    data = factor(glm_predict_5_balanced),
    reference = factor(rain_balanced_test$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.6
table(rain_balanced_test$RainTomorrow, glm_predict_6_balanced)
```

```
##    glm_predict_6_balanced
##        0    1
##   0 2694  439
##   1  895 2186
```

```
error6_balanced <-
  mean(glm_predict_6_balanced != rain_balanced_test$RainTomorrow)
accuracy6_balanced <-
  mean(glm_predict_6_balanced == rain_balanced_test$RainTomorrow)
cm6_balanced <-
  confusionMatrix(
    data = factor(glm_predict_6_balanced),
    reference = factor(rain_balanced_test$RainTomorrow),
    mode = 'everything'
  )

a_balanced <-
  create_confusion_matrix(cm4_balanced, 0.4, error4_balanced, accuracy4_balanced)
b_balanced <-
  create_confusion_matrix(cm5_balanced, 0.5, error5_balanced, accuracy5_balanced)
c_balanced <-
  create_confusion_matrix(cm6_balanced, 0.6, error6_balanced, accuracy6_balanced)

# Threshold of 0.5 is the best among thresholds in terms of accuracy,
```

```
#sensitivity, and specificity
cm_all_balanced = list(a_balanced, b_balanced, c_balanced)
plot_width <- c(4, 4, 4)
grid.arrange(grobs = cm_all_balanced,
             nrow = 3,
             width = plot_width)
```



3.3 Logistic Regression with Feature Selection

Feature selection is another key step in predictive model definition. The selection method involves evaluating all the features provided in the original dataset and selecting the most relevant features that best describe and predict our data. By identifying a smaller subset of important features, we are ensuring that the model doesn't overfit on the training data and is able to generalize on new data. Several techniques can be implemented to apply feature selection. Forward and backward selection iterate through all the features to determine each feature's effect on the overall model by cumulatively adding features from an empty set or removing features one at a time from the full set, respectively. Moreover, the two methods can be integrated to obtain a more exhaustive feature selection. Within the configuration of forward and backward selection using Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC), you can select the degree of penalization for the amount of features through the choice of k. The penalty term increases proportionally with the number of parameters set by k, which ensures that the complexity of the model is taken into account and to avoid overfitting. We chose to configure backward selection using BIC, which incorporates a log term and penalizes complex models more than AIC.

3.3.1 Feature Selection (Backward Selection using BIC)

```
# Perform logistic regression with backward stepwise selection
logit_model <-
  glm(rain_balanced$RainTomorrow ~ .,
```

```
      data = rain_balanced,
      family = binomial)

# Perform forward stepwise selection using BIC with log(n)
logit_model <-
  stepAIC(
    logit_model,
    direction = "backward",
    k = log(nrow(rain_balanced)),
    trace = FALSE
  )

# Print the summary of the logistic regression model
summary(logit_model)
```

```
##
## Call:
## glm(formula = rain_balanced$RainTomorrow ~ MinTemp + MaxTemp +
##     Rainfall + Sunshine + WindGustSpeed + WindSpeed9am + WindSpeed3pm +
##     Humidity3pm + Pressure9am + Pressure3pm + Cloud9am + Cloud3pm +
##     RainToday, family = binomial, data = rain_balanced)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5377  -0.6440  -0.0412   0.6411   2.8180
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.63412    0.22495  -7.264 3.74e-13 ***
## MinTemp        -1.01052    0.24797  -4.075 4.60e-05 ***
## MaxTemp         1.39711    0.30809   4.535 5.77e-06 ***
## Rainfall        2.29815    0.69305   3.316 0.000913 ***
## Sunshine       -2.36985    0.11567 -20.488  < 2e-16 ***
## WindGustSpeed   7.04107    0.24971  28.197  < 2e-16 ***
## WindSpeed9am   -0.81429    0.17622  -4.621 3.82e-06 ***
## WindSpeed3pm   -1.98642    0.21483  -9.246  < 2e-16 ***
## Humidity3pm     5.76180    0.15339  37.562  < 2e-16 ***
## Pressure9am     8.78896    0.61661  14.254  < 2e-16 ***
## Pressure3pm   -13.07527    0.64902 -20.146  < 2e-16 ***
## Cloud9am       -0.25366    0.07282  -3.483 0.000495 ***
## Cloud3pm        1.11010    0.08919  12.447  < 2e-16 ***
## RainToday       0.47366    0.04919   9.629  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 34455  on 24853  degrees of freedom
## Residual deviance: 21168  on 24840  degrees of freedom
## AIC: 21196
##
## Number of Fisher Scoring iterations: 5
```

```
# Subset the dataframe with the chosen features based on stepwise selection
rain_subset <-
  rain_balanced[, c(
    "MinTemp",
```

```
    "Sunshine",
    "WindGustSpeed",
    "WindSpeed9am",
    "WindSpeed3pm",
    "Humidity3pm",
    "Pressure9am",
    "Pressure3pm",
    "Cloud9am",
    "Cloud3pm",
    "Temp3pm",
    "RainToday",
    "RainTomorrow"
  )]
```

3.3.2 Feature Selection Results and Train/Test Split

In the summary of the feature selection model, we noticed that all the features chosen had a very significant p-value (***: p close to 0). Therefore, we decided to move forward with all the selected features we received from backward stepwise selection in subsequent areas of the project.

```
set.seed(123)

train <- sample(1:nrow(rain_subset), nrow(rain_subset) * 0.75)

# Calculate the test indices
test <- setdiff(1:nrow(rain_subset), train)

# Split the target variable into train and test sets
rain_subset_train <- rain_subset[train,]
rain_subset_test <- rain_subset[test,]
```

3.3.3 Logistic Regression using Selected Model

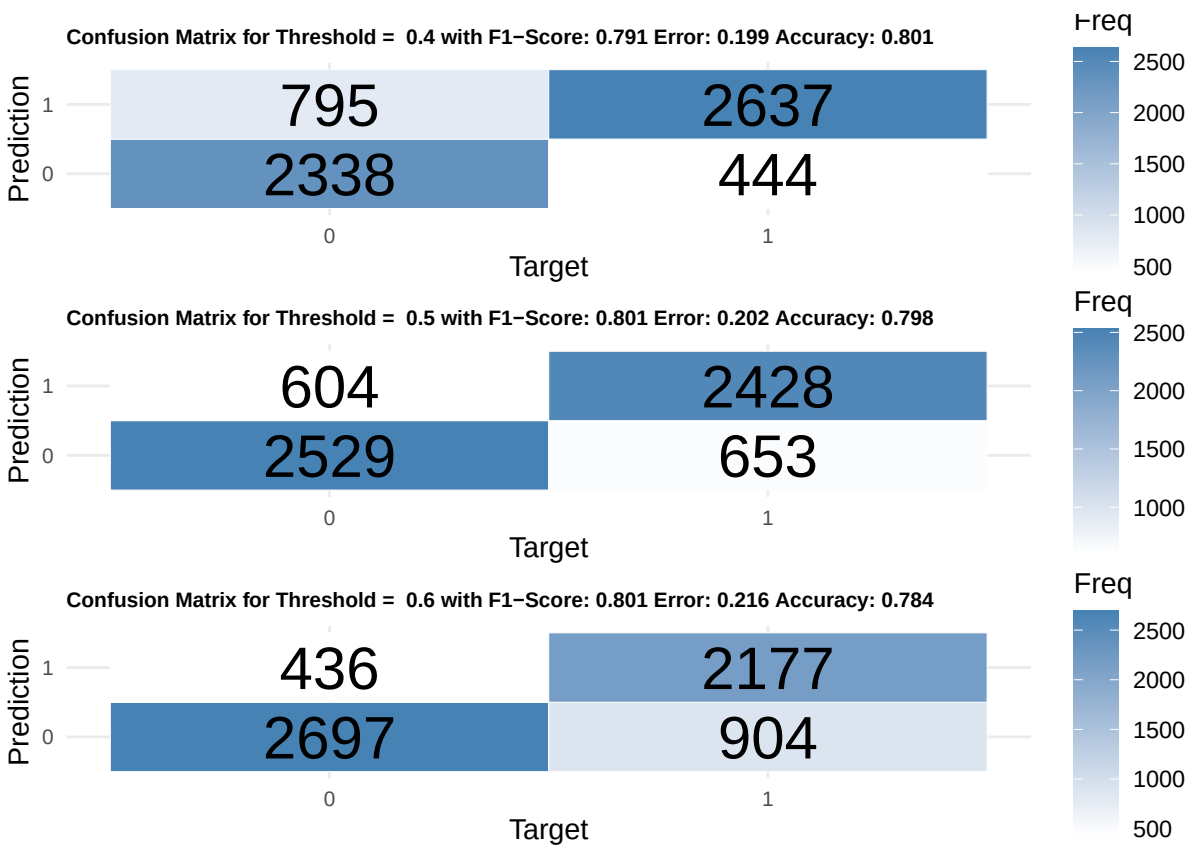
```
# Model Definition
glm_model <- glm(data = rain_subset_train,
        rain_subset_train$RainTomorrow ~ .,
        family = binomial)

model_summary <- summary(glm_model)
summary(glm_model)
```

```
##
## Call:
## glm(formula = rain_subset_train$RainTomorrow ~ ., family = binomial,
##     data = rain_subset_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.5021  -0.6462   0.0540   0.6430   2.8063
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.73364    0.26519  -6.537 6.26e-11 ***
## MinTemp        -1.38220    0.30132  -4.587 4.49e-06 ***
## Sunshine       -2.57348    0.13375 -19.242  < 2e-16 ***
## WindGustSpeed   6.89157    0.28504  24.177  < 2e-16 ***
## WindSpeed9am   -0.74847    0.20273  -3.692 0.000222 ***
```

```
## WindSpeed3pm   -1.67831    0.24595  -6.824 8.87e-12 ***
## Humidity3pm     5.97653    0.19279  31.000  < 2e-16 ***
## Pressure9am     8.34452    0.73462  11.359  < 2e-16 ***
## Pressure3pm   -12.54847    0.77299 -16.234  < 2e-16 ***
## Cloud9am       -0.28562    0.08367  -3.414 0.000641 ***
## Cloud3pm        1.08002    0.10194  10.595  < 2e-16 ***
## Temp3pm         1.97120    0.37774   5.218 1.80e-07 ***
## RainToday       0.54230    0.04751  11.414  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 25840  on 18639  degrees of freedom
## Residual deviance: 15903  on 18627  degrees of freedom
## AIC: 15929
##
## Number of Fisher Scoring iterations: 5
```

```
r2 <- 1 - (model_summary$deviance/model_summary$null.deviance) # 0.3845522
vif <- 1/(1-r2) # 1.624833

#Predict test with model
glm_predict <- predict(glm_model, rain_subset_test, type = "response")

#Convert predictions into 0,1 based on different thresholds

glm_predict_4<- ifelse(glm_predict > threshold4, 1, 0)
glm_predict_5<- ifelse(glm_predict > threshold5, 1, 0)
glm_predict_6<- ifelse(glm_predict > threshold6, 1, 0)

# Confusion matrix with threshold = 0.4
error4 <- mean(glm_predict_4!=rain_subset_test$RainTomorrow)
accuracy4 <- mean(glm_predict_4==rain_subset_test$RainTomorrow)
cm4 <- confusionMatrix(data = factor(glm_predict_4),
                       reference = factor(rain_subset_test$RainTomorrow),
                       mode ='everything')

# Confusion matrix with threshold = 0.5
error5 <- mean(glm_predict_5!=rain_subset_test$RainTomorrow)
accuracy5 <- mean(glm_predict_5==rain_subset_test$RainTomorrow)
cm5 <- confusionMatrix(data = factor(glm_predict_5),
                       reference = factor(rain_subset_test$RainTomorrow),
                       mode ='everything')

# Confusion matrix with threshold = 0.6
error6 <- mean(glm_predict_6!=rain_subset_test$RainTomorrow)
accuracy6 <- mean(glm_predict_6==rain_subset_test$RainTomorrow)
cm6 <- confusionMatrix(data = factor(glm_predict_6),
                       reference = factor(rain_subset_test$RainTomorrow),
                       mode ='everything')

a <- create_confusion_matrix(cm4, 0.4, error4, accuracy4)
b <- create_confusion_matrix(cm5, 0.5, error5, accuracy5)
c <- create_confusion_matrix(cm6, 0.6, error6, accuracy6)

cm_all = list(a, b, c)
```

```
plot_width <- c(4, 4, 4)
grid.arrange(grobs = cm_all, nrow = 3, width = plot_width)
```



3.4 Comparison of 3 Models

In the following plots, we compared each of the performance metrics previously computed across all the logistic regression models for each prediction threshold. By observing the results, we noticed close performance across all the models, with the simplest model before balancing and feature selection showing a slightly higher accuracy and F1 score, and lower error rate.

```
## Summary Statistics: F1-Score, Error, Accuracy

models <-
  c("Simple GLM",
    "GLM with Balancing",
    "GLM with Balancing and Feature Selection")
model_suffix <- c("_0", "_balanced", "")

thresholds <- c(4, 5, 6)
threshold_values <- c(0.4, 0.5, 0.6)

metrics <-
  data.frame(
    Model = character(),
    Threshold = numeric(),
    F1_Score = numeric(),
    Error = numeric(),
    Accuracy = numeric(),
    stringsAsFactors = FALSE
  )
```

```r
j <-  1

for (model in models) {
  model_suff <- model_suffix[j]
  j <-  j + 1
  for (i in 1:length(thresholds)) {
    threshold <- thresholds[i]
    threshold_value <- threshold_values[i]

    # Calculate the F1 score for each combination of model and threshold
    cm_name <- paste0("cm", threshold, model_suff)
    cm <- get(cm_name)
    f1_score <- cm$byClass["F1"]

    error_name <- paste0("error", threshold, model_suff)
    error <- get(error_name)

    accuracy_name <- paste0("accuracy", threshold, model_suff)
    accuracy <- get(accuracy_name)

    # Add the F1 score to the data frame
    metrics <-
      rbind(
        metrics,
        data.frame(
          Model = model,
          Threshold = threshold_value,
          F1_Score = f1_score,
          Error = error,
          Accuracy = accuracy,
          stringsAsFactors = FALSE,
          row.names = NULL
        )
      )
  }
}

print(metrics)
```

```
##                                     Model Threshold  F1_Score     Error
## 1                             Simple GLM       0.4 0.9030458 0.1527827
## 2                             Simple GLM       0.5 0.9074894 0.1496632
## 3                             Simple GLM       0.6 0.9087790 0.1507976
## 4                      GLM with Balancing       0.4 0.7896517 0.2001931
## 5                      GLM with Balancing       0.5 0.8012678 0.2018024
## 6                      GLM with Balancing       0.6 0.8015472 0.2146765
## 7 GLM with Balancing and Feature Selection       0.4 0.7905325 0.1993885
## 8 GLM with Balancing and Feature Selection       0.5 0.8009501 0.2022852
## 9 GLM with Balancing and Feature Selection       0.6 0.8010098 0.2156421
##    Accuracy
## 1 0.8472173
## 2 0.8503368
## 3 0.8492024
## 4 0.7998069
## 5 0.7981976
## 6 0.7853235
```

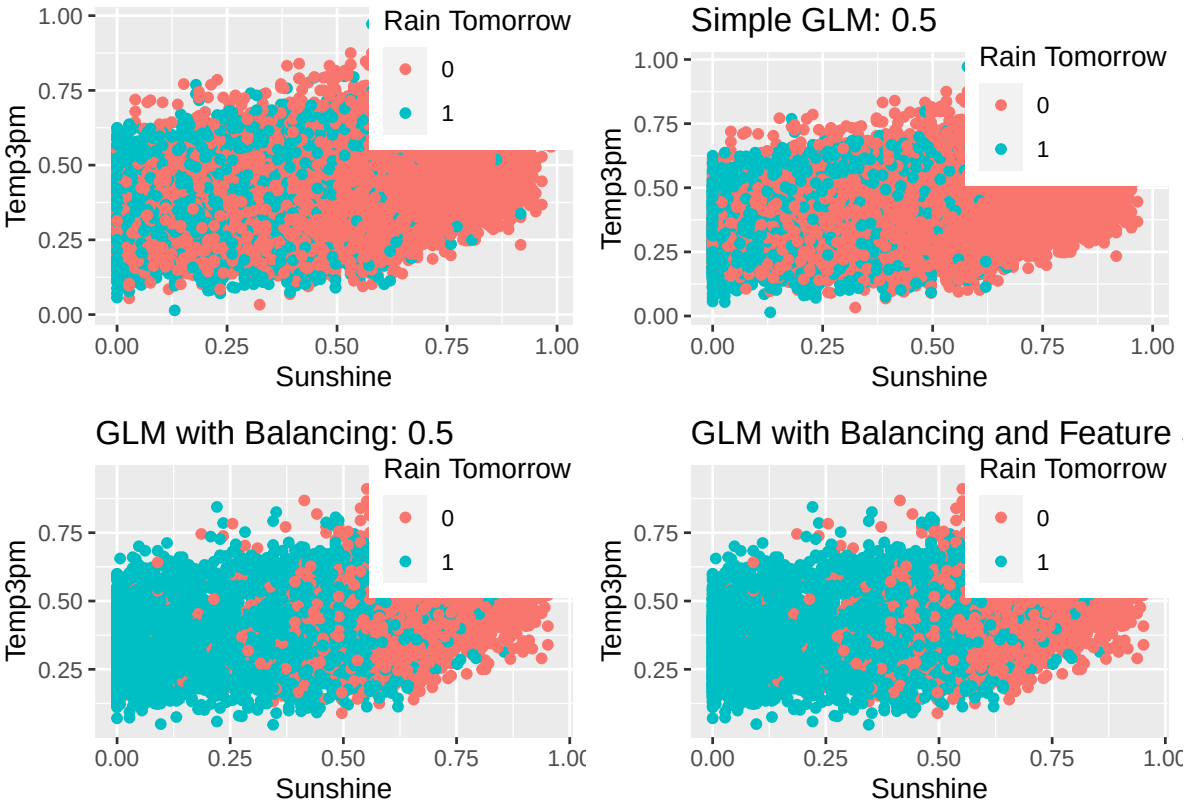
```
## 7 0.8006115
## 8 0.7977148
## 9 0.7843579
```

3.5 Visualizing data classifications from predictions

In addition to comparing performance metrics across the three models, we also sought to visualize the separation of classes in scatter plots between two predictive features: Sunshine and Temp3pm. As seen for the majority of the features, we are unable to completely separate the classes in the scatterplots since the features are highly correlated. To pair with these scatter plots, we created logistic curves for each of our logistic regresison models that plot the predicted probability of the RainTomorrow classification of samples against the true classification. As expected, points with predicted probabilities closer to zero are being classified as 0 for RainTomorrow, whereas points with probabilities closer to one are being classified as 1. In comparing the three models, the Simple GLM appears to be more skewed to the right due to zero being the dominating class. While in the other two modified GLM models, undersampling reduced the number of samples of the 0 class to a point at which the classes had an equal number of samples. This likely led to less skewing of the logistic curve, but also a more even distribution of points around the middle of the plots, indicating less polarity in the classification.

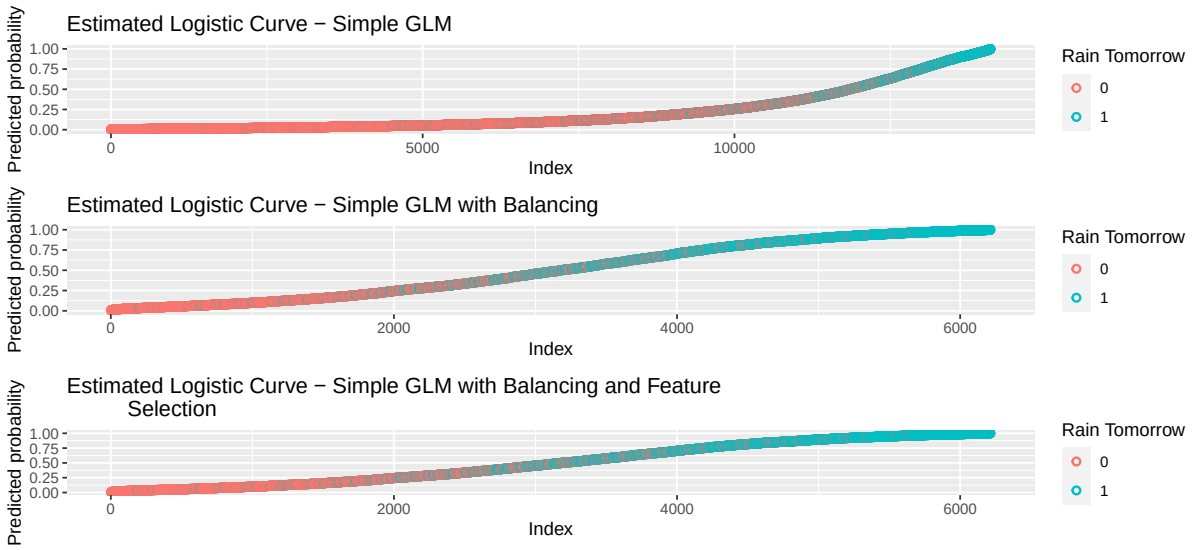
```
#for different features: mintemp an temp3pm
a00 <- ggplot(data = rain_n_test0 , aes(x = MinTemp,
y = Temp3pm,
color= as.factor(RainTomorrow) )) +
geom_point()+
labs(x = "Mintemp",
y = "Temp3pm",
color = "Rain Tomorrow") +
theme(legend.position = c(0.8, 0.8))

# original classification
a0 <- ggplot(data = rain_n_test0 , aes(x = Sunshine,
y = Temp3pm,
color= as.factor(RainTomorrow) )) +
geom_point()+
labs(x = "Sunshine",
y = "Temp3pm",
color = "Rain Tomorrow") +
theme(legend.position = c(0.8, 0.8))

# Simple GLM: Threshold of 0.5
a2 <- ggplot(data = rain_n_test0 , aes(x = Sunshine,
y = Temp3pm,
color= as.factor(glm_predict_5_0) )) +
geom_point()+
labs(x = "Sunshine",
y = "Temp3pm",
color = "Rain Tomorrow",
title = "Simple GLM: 0.5") +
theme(legend.position = c(0.8, 0.8))

# GLM with Balancing: Threshold of 0.5
b2 <- ggplot(data = rain_balanced_test , aes(x = Sunshine,
y = Temp3pm,
color= as.factor(glm_predict_5_balanced) )) +
geom_point()+
labs(x = "Sunshine",
y = "Temp3pm",
```

```
color = "Rain Tomorrow",
title = "GLM with Balancing: 0.5") +
theme(legend.position = c(0.8, 0.8))

# GLM with Balancing and Feature Selection
c2 <- ggplot(data = rain_balanced_test , aes(x = Sunshine,
y = Temp3pm,
color= as.factor(glm_predict_5) )) +
geom_point()+
labs(x = "Sunshine",
y = "Temp3pm",
color = "Rain Tomorrow",
title = "GLM with Balancing and Feature Selection: 0.5") +
theme(legend.position = c(0.8, 0.8))

grid.arrange(a0, a2, b2, c2, nrow = 2)
```




```
predicted_data <-
  data.frame(prob_of_rain = glm_predict0,
             RainTomorrow = rain_n_test0$RainTomorrow)
predicted_data <-
  predicted_data[order(predicted_data$prob_of_rain, decreasing = FALSE), ]
predicted_data$rank <-  1:nrow(predicted_data)

a3 <-
  ggplot(data = predicted_data, aes(x = rank, y = prob_of_rain)) +
  geom_point(
    aes(color = as.factor(RainTomorrow)),
    alpha = 1,
    shape = 1,
    stroke = 1
```

```
  ) +
  xlab("Index") +
  ylab("Predicted probability") +
  ggtitle("Estimated Logistic Curve - Simple GLM") +
  labs(color = "Rain Tomorrow")

predicted_data <-
  data.frame(prob_of_rain = glm_predict_balanced,
             RainTomorrow = rain_balanced_test$RainTomorrow)
predicted_data <-
  predicted_data[order(predicted_data$prob_of_rain, decreasing = FALSE), ]
predicted_data$rank <-  1:nrow(predicted_data)

b3 <-
  ggplot(data = predicted_data, aes(x = rank, y = prob_of_rain)) +
  geom_point(
    aes(color = as.factor(RainTomorrow)),
    alpha = 1,
    shape = 1,
    stroke = 1
  ) +
  xlab("Index") +
  ylab("Predicted probability") +
  ggtitle("Estimated Logistic Curve - Simple GLM with Balancing") +
  labs(color = "Rain Tomorrow")

predicted_data <-
  data.frame(prob_of_rain = glm_predict,
             RainTomorrow = rain_balanced_test$RainTomorrow)
predicted_data <-
  predicted_data[order(predicted_data$prob_of_rain, decreasing = FALSE), ]
predicted_data$rank <-  1:nrow(predicted_data)

c3 <-
  ggplot(data = predicted_data, aes(x = rank, y = prob_of_rain)) +
  geom_point(
    aes(color = as.factor(RainTomorrow)),
    alpha = 1,
    shape = 1,
    stroke = 1
  ) +
  xlab("Index") +
  ylab("Predicted probability") +
  ggtitle("Estimated Logistic Curve - Simple GLM with Balancing and Feature
          Selection") +
  labs(color = "Rain Tomorrow")

grid.arrange(a3, b3, c3, nrow = 3)
```




```
X_train_subset <- model.matrix(RainTomorrow~., data=rain_subset_train)
X_test_subset <- model.matrix(RainTomorrow~., data=rain_subset_test)
X_train_subset <- X_train_subset[,-1]
X_test_subset <- X_test_subset[,-1]

# Target vector
y_train_subset <- rain_subset_train$RainTomorrow
y_test_subset <- rain_subset_test$RainTomorrow
```

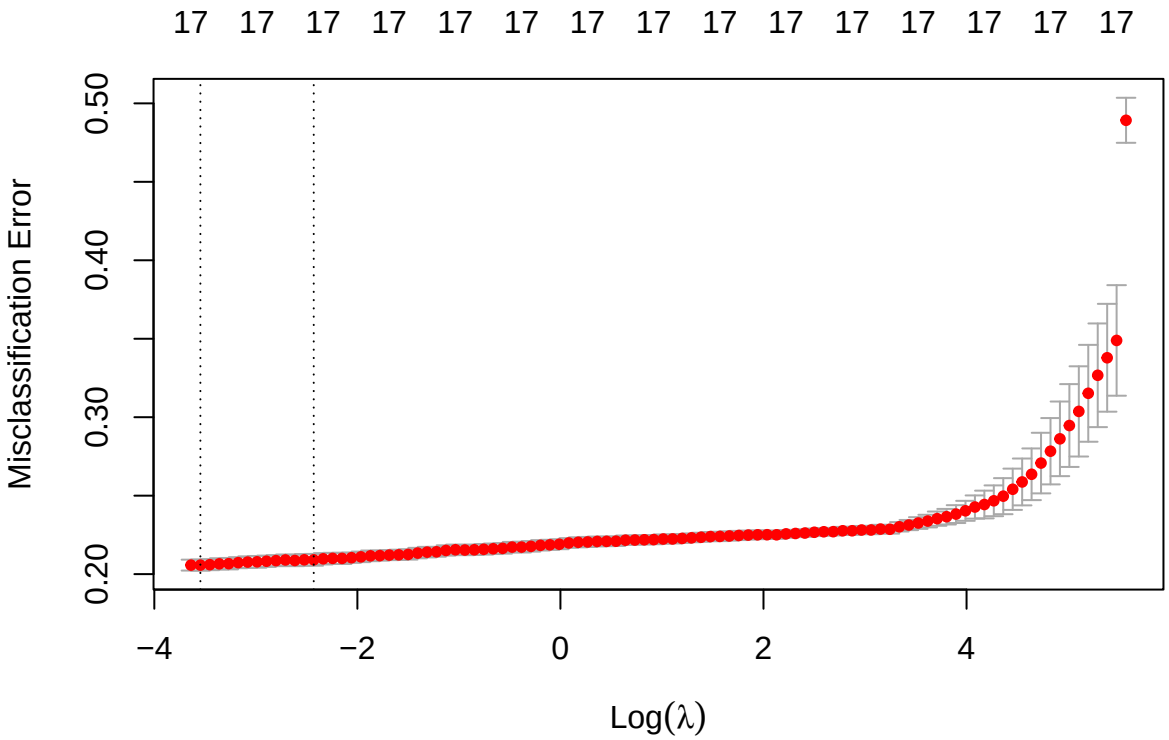
4 Regularized Regression Models

Beyond the standard logistic regression models, we chose to implement two alternative regularization models: LASSO and Ridge Regression. Both regularization techniques introduce a penalty team to the standard regression model; Least Absolute Shrinkage and Selection Operator (LASSO) regression utilizes the L1 norm, which is the sum of the absolute value of the parameters multiplied by a hyperparamter, lambda, whereas Ridge Regression uses the L2 norm, which is the square root of the sum of squared parameters multiplied by the lambda hyperparameter. The lambda hyperparameter plays a major role in determining the level of regularization introduced in the model; the higher the lambda value is, the higher the regularization and vice-versa. Moreover, LASSO regression is suitable for performing another form of feature selection as it introduces sparsity in our model and prioritizes fewer parameters. Ridge regression, on the other hand, is effective for dealing with multicollinearity (identified by VIF), when features are highly correlated, by shrinking the magnitude of the coefficients. According to the VIF calculated previously (greater than 1), our data has high multicollinearity, which can be potentially addressed using the ridge regression.

4.1 LASSO and Ridge Balanced w/o Feature Selection

```
set.seed(123)

#We're first implementing lasso/ridge with balanced data with no feature
#selection

X <- model.matrix(RainTomorrow ~ ., data = rain_balanced)
X <- X[, -1]

X_train <- model.matrix(RainTomorrow ~ ., data = rain_balanced_train)
X_test <- model.matrix(RainTomorrow ~ ., data = rain_balanced_test)
X_train <- X_train[, -1]
```

```r
X_test <- X_test[, -1]

# Target vector
y <- rain_balanced$RainTomorrow
y_train <- rain_balanced_train$RainTomorrow
y_test <- rain_balanced_test$RainTomorrow

# alpha=0 for ridge, alpha=1 (default) for lasso

# Ridge Regression (L2)

ridge_cv <-
  cv.glmnet(
    X_train,
    y_train,
    alpha = 0,
    family = "binomial",
    type.measure = "class"
  )
plot(ridge_cv)
```

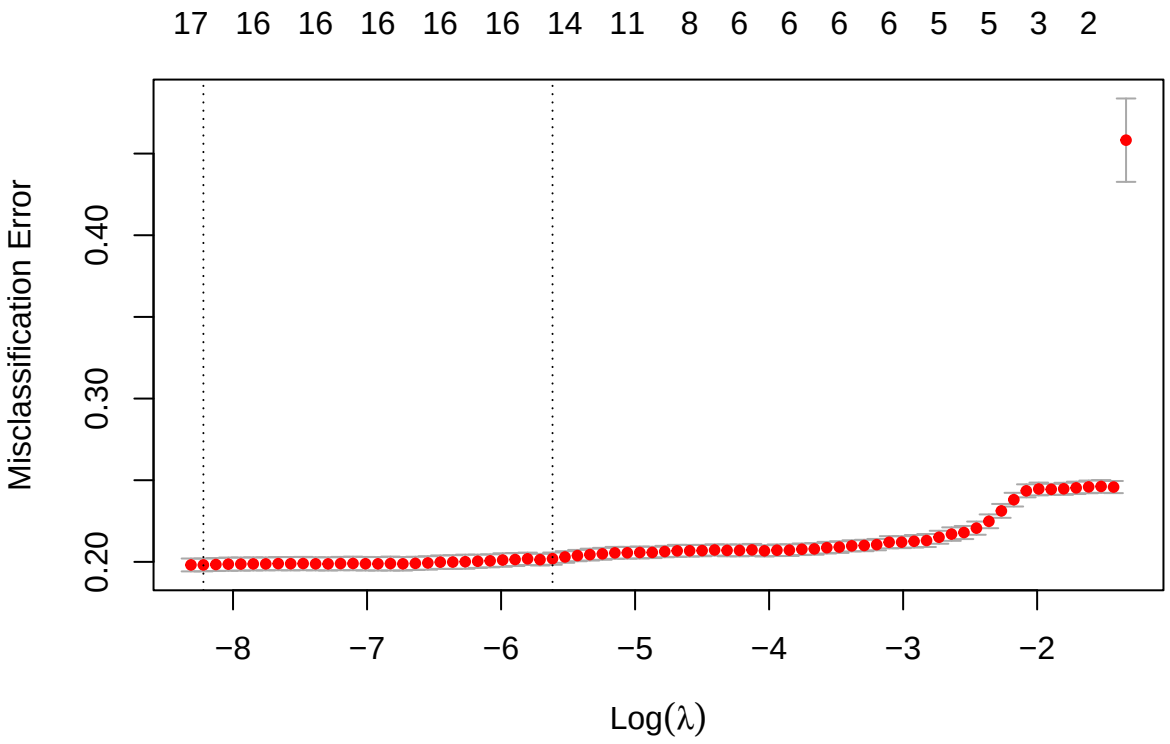



```r
# To select best lambda
lambda_opt_ridge <- ridge_cv$lambda.min
lambda_opt_ridge
```

```
## [1] 0.0288348
```

```r
pred_ridge <-
  predict(ridge_cv, X_test, type = "class", s = lambda_opt_ridge)
error_ridge <- mean(pred_ridge != y_test) # 0.2133891
accuracy_ridge <- mean(pred_ridge == y_test) # 0.7866109
cm_ridge <-
  confusionMatrix(
    data = factor(pred_ridge),
```

```
    reference = factor(y_test),
    mode = 'everything'
  ) # F1 : 0.7856

#Lasso Regression (L1)
lasso_cv <-
  cv.glmnet(
    X_train,
    y_train,
    alpha = 1,
    family = "binomial",
    type.measure = "class"
  )
plot(lasso_cv)
```



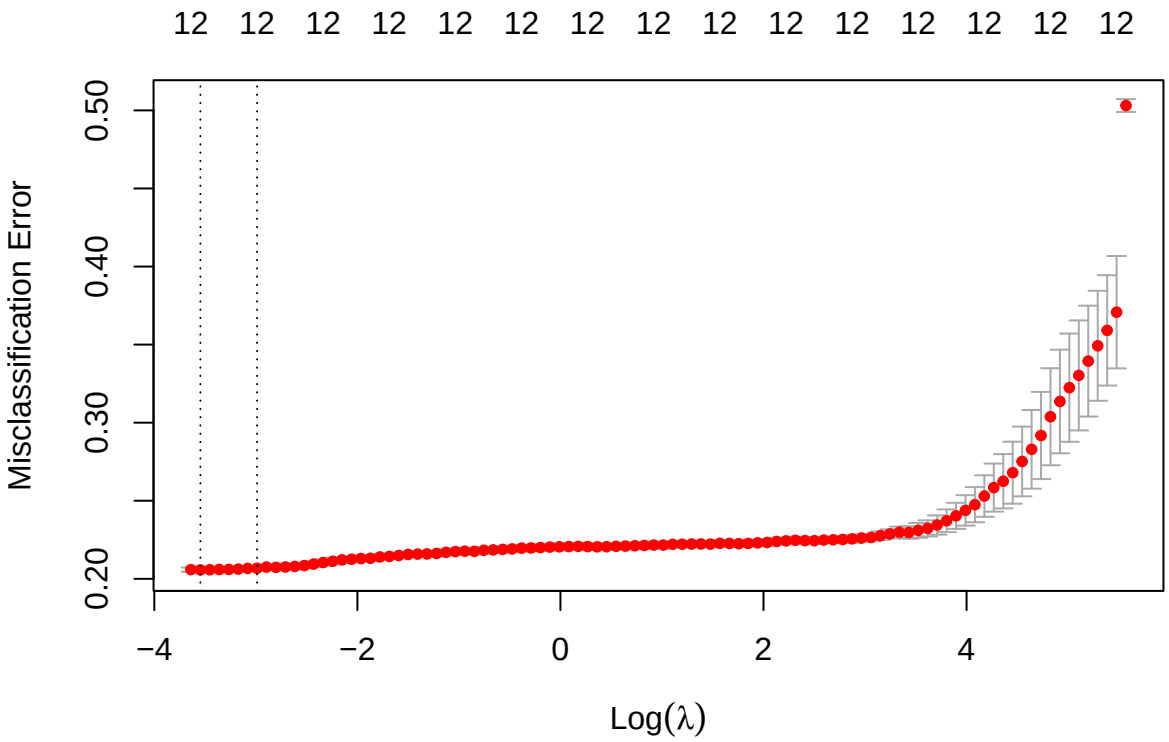

```
lambda_opt_lasso <- lasso_cv$lambda.min
lambda_opt_lasso
```

```
## [1] 0.0002689143
```

```
pred_lasso <-
  predict(lasso_cv, X_test, type = "class", s = lambda_opt_lasso)
error_lasso <- mean(pred_lasso != y_test) # 0.2046991
accuracy_lasso <- mean(pred_lasso == y_test) #  0.7953009
cm_lasso <-
  confusionMatrix(
    data = factor(pred_lasso),
    reference = factor(y_test),
    mode = 'everything'
  ) # F1 : 0.7956
```

4.2 LASSO and Ridge Balanced w/ Feature Selection

```
#Then, we implemented Ridge and LASSO after feature selection

# Ridge Regression (L2)

ridge_cv_subset <-
  cv.glmnet(
    X_train_subset,
    y_train_subset,
    alpha = 0,
    family = "binomial",
    type.measure = "class"
  )
plot(ridge_cv_subset)
```

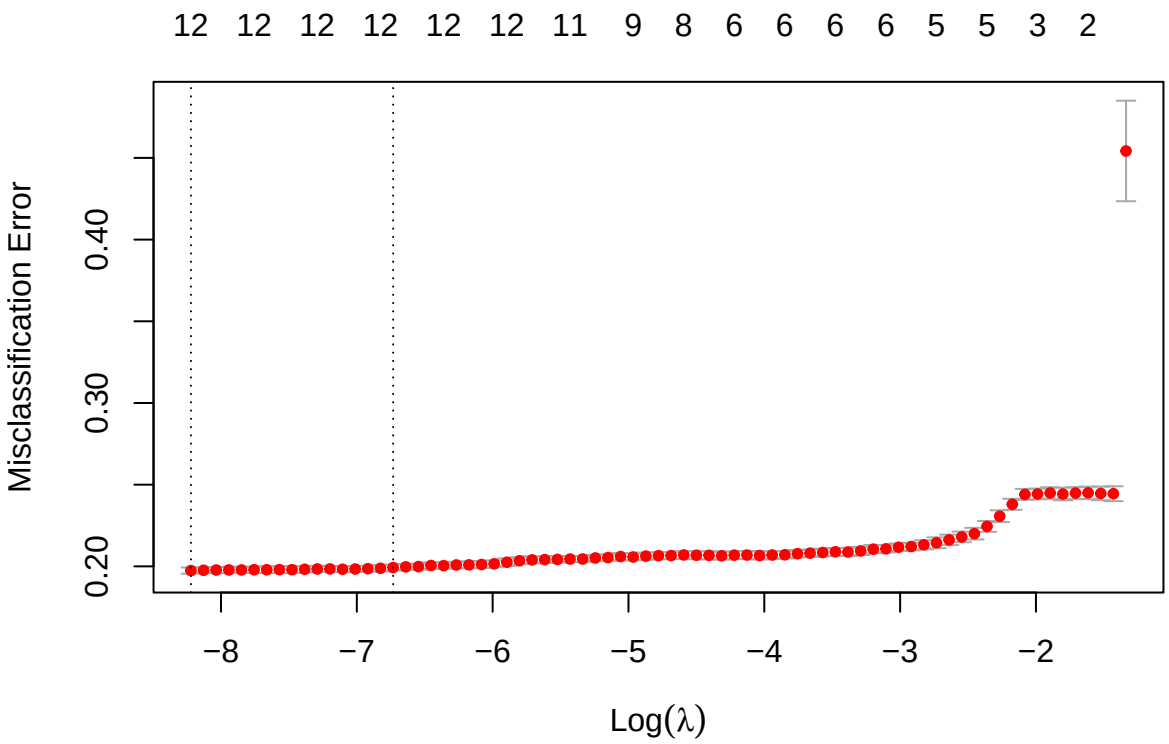



```
# To select best lambda
lambda_opt_ridge_subset <- ridge_cv_subset$lambda.min
lambda_opt_ridge_subset
```

```
## [1] 0.0288348
```

```
pred_ridge_subset <-
  predict(ridge_cv_subset, X_test_subset, type = "class",
          s = lambda_opt_ridge_subset)
error_ridge_subset <-
  mean(pred_ridge_subset != y_test_subset) # 0.2125845
accuracy_ridge_subset <-
  mean(pred_ridge_subset == y_test_subset) # 0.7874155
cm_ridge_subset <-
  confusionMatrix(
    data = factor(pred_ridge_subset),
    reference = factor(y_test),
    mode = 'everything'
```

```
  ) # F1 : 0.7864

#Lasso Regression (L1)
lasso_cv_subset <-
  cv.glmnet(
    X_train_subset,
    y_train_subset,
    alpha = 1,
    family = "binomial",
    type.measure = "class"
  )
plot(lasso_cv_subset)
```




```
lambda_opt_lasso_subset <- lasso_cv_subset$lambda.min
lambda_opt_lasso_subset
```

```
## [1] 0.0002689143
```

```
pred_lasso_subset <-
  predict(lasso_cv_subset, X_test_subset, type = "class",
          s = lambda_opt_lasso_subset)
error_lasso_subset <-
  mean(pred_lasso_subset != y_test_subset) # 0.2059865
accuracy_lasso_subset <-
  mean(pred_lasso_subset == y_test_subset) # 0.7940135
cm_lasso_subset <-
  confusionMatrix(
    data = factor(pred_lasso_subset),
    reference = factor(y_test),
    mode = 'everything'
  ) # F1 : 0.7941
```

5 Linear and Quadratic Discriminant Analysis

Linear and Quadratic Discriminant Analysis, also known as LDA and QDA, fall into another class of statistical modeling techniques that can be used in supervised learning. LDA projects high-dimensional data into a lower-dimensional space while still preserving the original classifications of the training set. The reducing technique separates the data linearly to deduce a linear combination of the features that best generalize the data. LDA is best suited for data with clear spatial separation of classes and assumes the equality of covariance matrices between the classes. Meanwhile, QDA is an extension of LDA that can capture non-linear relationships across model features. Unlike LDA, QDA is still suitable when the classes have unique covariance matrices. In our project, we performed both LDA and QDA on our balanced and feature selected data. However, since LDA and QDA are sensitive to outliers, we performed outlier removal prior to running the models.

5.1 Remove outliers from data

```
#looking for outliers in our data after balancing and feature selection

g1<- ggplot(data = rain_subset_train, aes(y = MinTemp,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Min Temperature")
chisq.out.test(rain_subset_train$MinTemp)  #p-value = 0.00151, remove 376
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$MinTemp
## X-squared = 10.066, p-value = 0.00151
## alternative hypothesis: lowest value 0 is an outlier
```

```
which(rain_subset_train$MinTemp == 0)
```

```
## [1] 376
```

```
g2<- ggplot(data = rain_subset_train, aes(y = Sunshine,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Sunshine")
chisq.out.test(rain_subset_train$Sunshine)  #p-value = 0.04431, index 6965
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Sunshine
## X-squared = 4.0448, p-value = 0.04431
## alternative hypothesis: highest value 1 is an outlier
```

```
which(rain_subset_train$Sunshine == 1)
```

```
## [1] 6965
```

```
g3<- ggplot(data = rain_subset_train, aes(y = WindGustSpeed,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("WindGustSpeed")
chisq.out.test(rain_subset_train$WindGustSpeed) #p-value = 3.071e-07, no values
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$WindGustSpeed
## X-squared = 26.204, p-value = 3.071e-07
## alternative hypothesis: highest value 0.939130434782609 is an outlier
```

```
which(rain_subset_train$WindGustSpeed == 0.939130434782609)
```

```
## integer(0)
```

```
g4<- ggplot(data = rain_subset_train, aes(y = WindSpeed9am,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("WindSpeed9am")
chisq.out.test(rain_subset_train$WindSpeed9am)  #p-value = 1.43e-08 , no values
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$WindSpeed9am
## X-squared = 32.146, p-value = 1.43e-08
## alternative hypothesis: highest value 0.969230769230769 is an outlier
```

```
which(rain_subset_train$WindSpeed9am == 0.969230769230769)
```

```
## integer(0)
```

```
g5<- ggplot(data = rain_subset_train, aes(y = WindSpeed3pm,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("WindSpeed3pm")
chisq.out.test(rain_subset_train$WindSpeed3pm)  #p-value = 4.733e-07 , no values
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$WindSpeed3pm
## X-squared = 25.37, p-value = 4.733e-07
## alternative hypothesis: highest value 0.851351351351351 is an outlier
```

```
which(rain_subset_train$WindSpeed3pm == 0.851351351351351)
```

```
## integer(0)
```

```
g6<- ggplot(data = rain_subset_train, aes(y = Humidity3pm,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Humidity3pm")
chisq.out.test(rain_subset_train$Humidity3pm)
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Humidity3pm
## X-squared = 6.6737, p-value = 0.009785
## alternative hypothesis: lowest value 0.01 is an outlier
```

```
#p-value = 0.009785, #indices:3782 10189 10959 15362 16105 18240
which(rain_subset_train$Humidity3pm ==0.01)
```

```
## [1]  3782 10189 10959 15362 16105 18240
```

```
g7<- ggplot(data = rain_subset_train, aes(y = Pressure9am,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Pressure9am")
chisq.out.test(rain_subset_train$Pressure9am) # p-value = 2.435e-06, index= 1935
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Pressure9am
## X-squared = 22.217, p-value = 2.435e-06
## alternative hypothesis: lowest value 0.0283806343906518 is an outlier
```

```
which(rain_subset_train$Pressure9am ==0.0283806343906518)
```

```
## [1] 1935
```

```
g8<- ggplot(data = rain_subset_train, aes(y = Pressure3pm,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Pressure3pm")
chisq.out.test(rain_subset_train$Pressure3pm)  # p-value = 2.756e-07, index=15369
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Pressure3pm
## X-squared = 26.413, p-value = 2.756e-07
## alternative hypothesis: lowest value 0 is an outlier
```

```
which(rain_subset_train$Pressure3pm ==0)
```

```
## [1] 15369
```

```
g9<- ggplot(data = rain_subset_train, aes(y = Cloud9am,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Cloud9am")
chisq.out.test(rain_subset_train$Cloud9am)
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Cloud9am
## X-squared = 3.2178, p-value = 0.07284
## alternative hypothesis: lowest value 0 is an outlier
```

```
# we got a p-value of 0.07 so we cannot refute the null hypothesis


g10<- ggplot(data = rain_subset_train, aes(y = Cloud3pm,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
```

```
outlier.size = 2)+
theme(legend.position="none") +
ylab("Cloud3pm")
chisq.out.test(rain_subset_train$Cloud3pm) #p-value = 0.04988
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Cloud3pm
## X-squared = 3.8456, p-value = 0.04988
## alternative hypothesis: lowest value 0 is an outlier
```

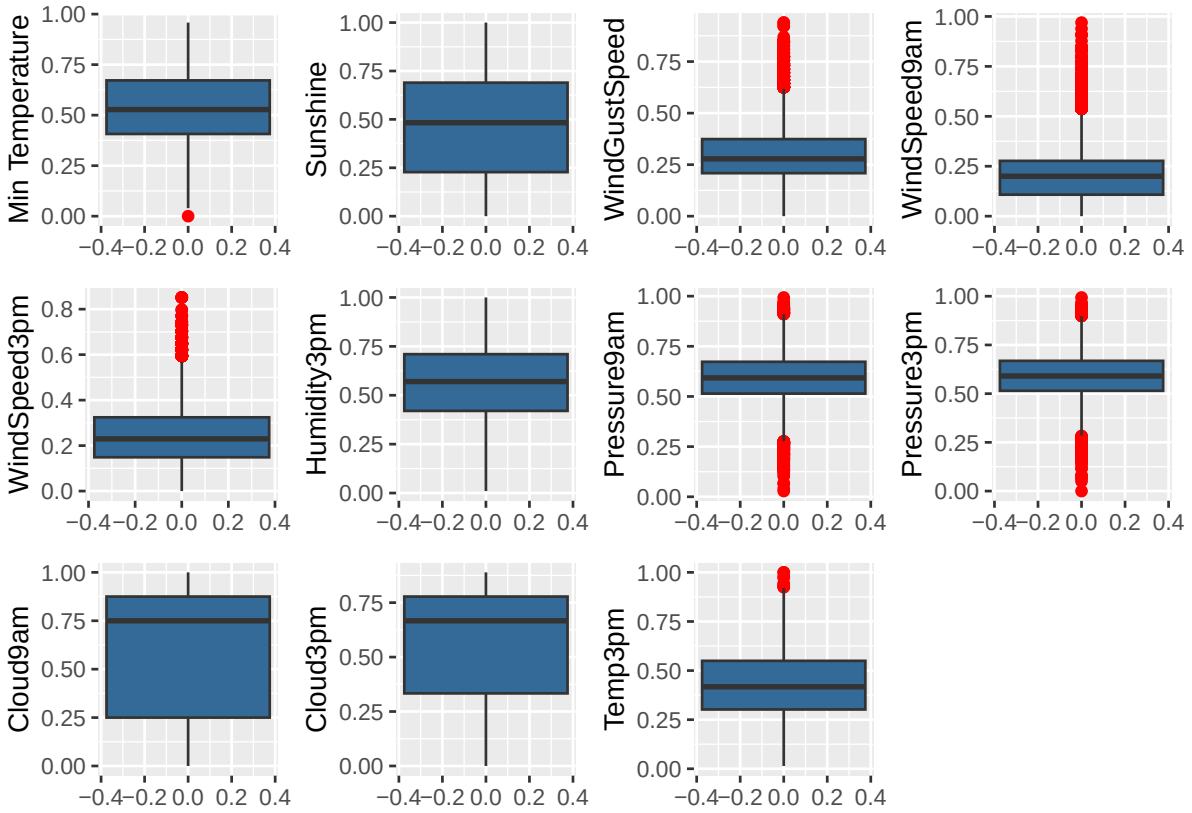
```
g11<- ggplot(data = rain_subset_train, aes(y = Temp3pm,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Temp3pm")
chisq.out.test(rain_subset_train$Temp3pm)
```

```
##
##  chi-squared test for outlier
##
## data:  rain_subset_train$Temp3pm
## X-squared = 12.534, p-value = 0.0003997
## alternative hypothesis: highest value 1 is an outlier
```

```
#p-value = 0.0003997, indices= 4596, 10679
which(rain_subset_train$Temp3pm ==1)
```

```
## [1]  4596 10679
```

```
grid.arrange(g1, g2, g3,g4,g5,g6,g7,g8,g9,g10,g11,  nrow = 3)
```



```r
#remove outliers for p_values less that 0.05
rain_subset_train_NoOutliers <- rain_subset_train[-c(4596,10679,15369,1935,3782,
                                                     10189,10959,15362,16105,
                                                     18240,376,6965)]
```

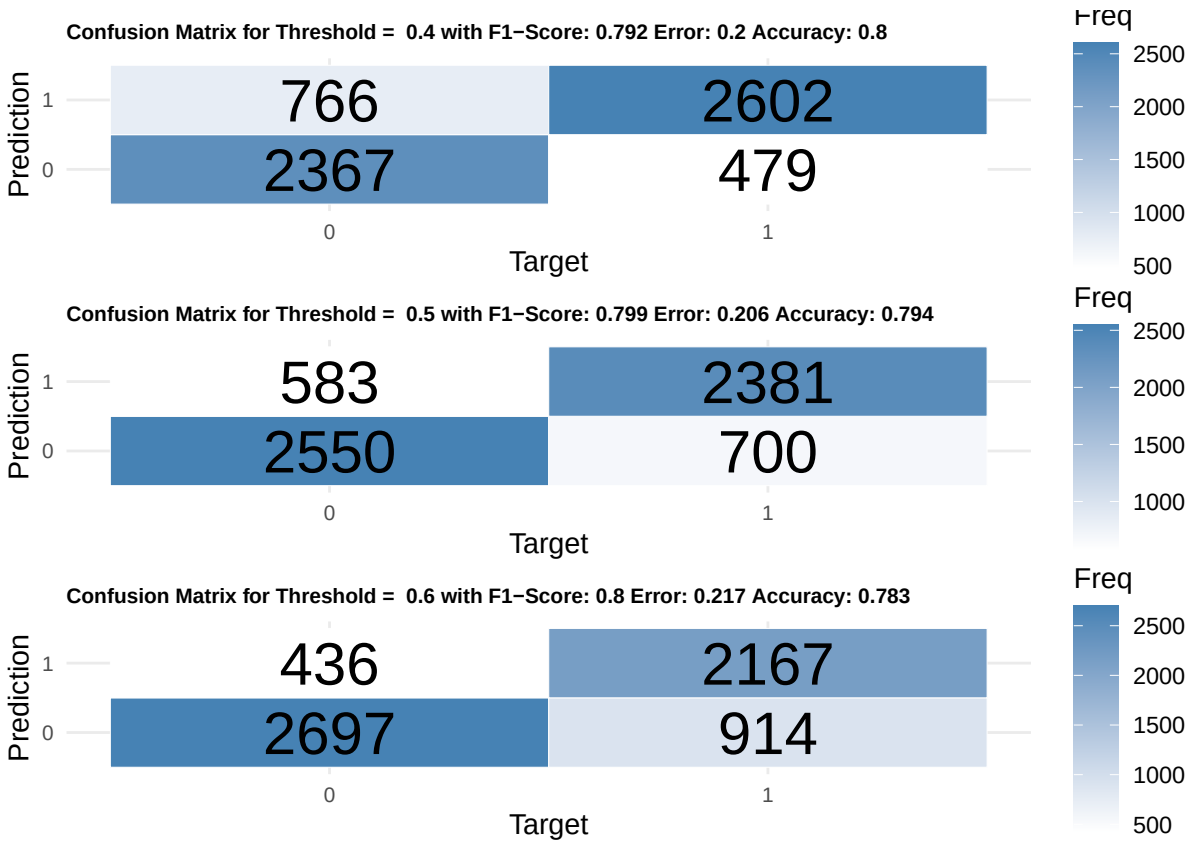
5.2 Linear Discriminant Analysis (LDA)

```r
# Model definition starting from the previous glm_bal model:

lda <-
  lda(data = rain_subset_train_NoOutliers, RainTomorrow ~ ., family = "binomial")
lda
```

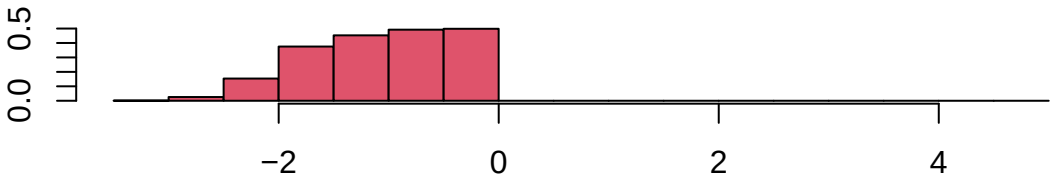
```
## Call:
## lda(RainTomorrow ~ ., data = rain_subset_train_NoOutliers, family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.4986052 0.5013948
##
## Group means:
##     MinTemp  Sunshine WindGustSpeed WindSpeed9am WindSpeed3pm Humidity3pm
## 0 0.5198474 0.5981404     0.2639337    0.2044065    0.2354919   0.4472972
## 1 0.5566944 0.3136079     0.3289740    0.2318952    0.2608863   0.6693281
##   Pressure9am Pressure3pm  Cloud9am  Cloud3pm   Temp3pm RainToday
## 0   0.6284978   0.6232747 0.4703169 0.4187170 0.4626215 0.1561222
## 1   0.5583618   0.5622434 0.7413733 0.6944504 0.3922791 0.4574149
##
## Coefficients of linear discriminants:
##                      LD1
## MinTemp       -0.8224766
## Sunshine      -1.7816536
## WindGustSpeed  3.7461372
## WindSpeed9am  -0.4589668
## WindSpeed3pm  -0.7932450
## Humidity3pm    3.3868009
## Pressure9am    3.5262631
## Pressure3pm   -5.8402214
## Cloud9am      -0.1977520
## Cloud3pm       0.7803935
## Temp3pm        1.3252533
## RainToday      0.3534121
```

```r
pred_lda <- predict(lda, rain_subset_test, type = "response")

post_lda <- pred_lda$posterior

pred_lda_04 <- as.factor(ifelse(post_lda[, 2] > threshold4, 1, 0))
pred_lda_05 <- as.factor(ifelse(post_lda[, 2] > threshold5, 1, 0))
pred_lda_06 <- as.factor(ifelse(post_lda[, 2] > threshold6, 1, 0))


# Confusion matrix with threshold = 0.4
error_lda4 <- mean(pred_lda_04 != rain_subset_test$RainTomorrow)
accuracy_lda4 <- mean(pred_lda_04 == rain_subset_test$RainTomorrow)
lda_CM04 <-
```

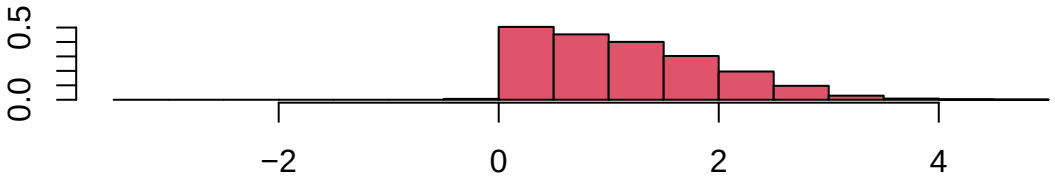
```
  confusionMatrix(
    data = factor(pred_lda_04),
    reference = factor(rain_subset_test$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.5
error_lda5 <- mean(pred_lda_05 != rain_subset_test$RainTomorrow)
accuracy_lda5 <- mean(pred_lda_05 == rain_subset_test$RainTomorrow)
lda_CM05 <-
  confusionMatrix(
    data = factor(pred_lda_05),
    reference = factor(rain_subset_test$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.6
error_lda6 <- mean(pred_lda_06 != rain_subset_test$RainTomorrow)
accuracy_lda6 <- mean(pred_lda_06 == rain_subset_test$RainTomorrow)
lda_CM06 <-
  confusionMatrix(
    data = factor(pred_lda_06),
    reference = factor(rain_subset_test$RainTomorrow),
    mode = 'everything'
  )



A <-
  create_confusion_matrix(lda_CM04, 0.4, error_lda4, accuracy_lda4)
B <-
  create_confusion_matrix(lda_CM05, 0.5, error_lda5, accuracy_lda5)
C <-
  create_confusion_matrix(lda_CM06, 0.6, error_lda6, accuracy_lda6)

# Threshold of 0.6 is the best among thresholds in terms of accuracy,
#sensitivity, and specificity
CM_all_lda = list(A, B, C)
plot_width <- c(4, 4, 4)
grid.arrange(grobs = CM_all_lda,
             nrow = 3,
             width = plot_width)
```




```
# Here we plot stacked histograms representing the linear combination of
# variable that best represent the samples with their discriminant scores, split
#by each assigned class. The y-axis is the frequency of examples falling into
# each range of discriminant scores. From these plots, we observed little to no
# overlap across the two classes, which is an indicator of good separation
# between the two classes.

ldahist(pred_lda$x[, 1], g = pred_lda$class, col = 2)
```



group 0



group 1

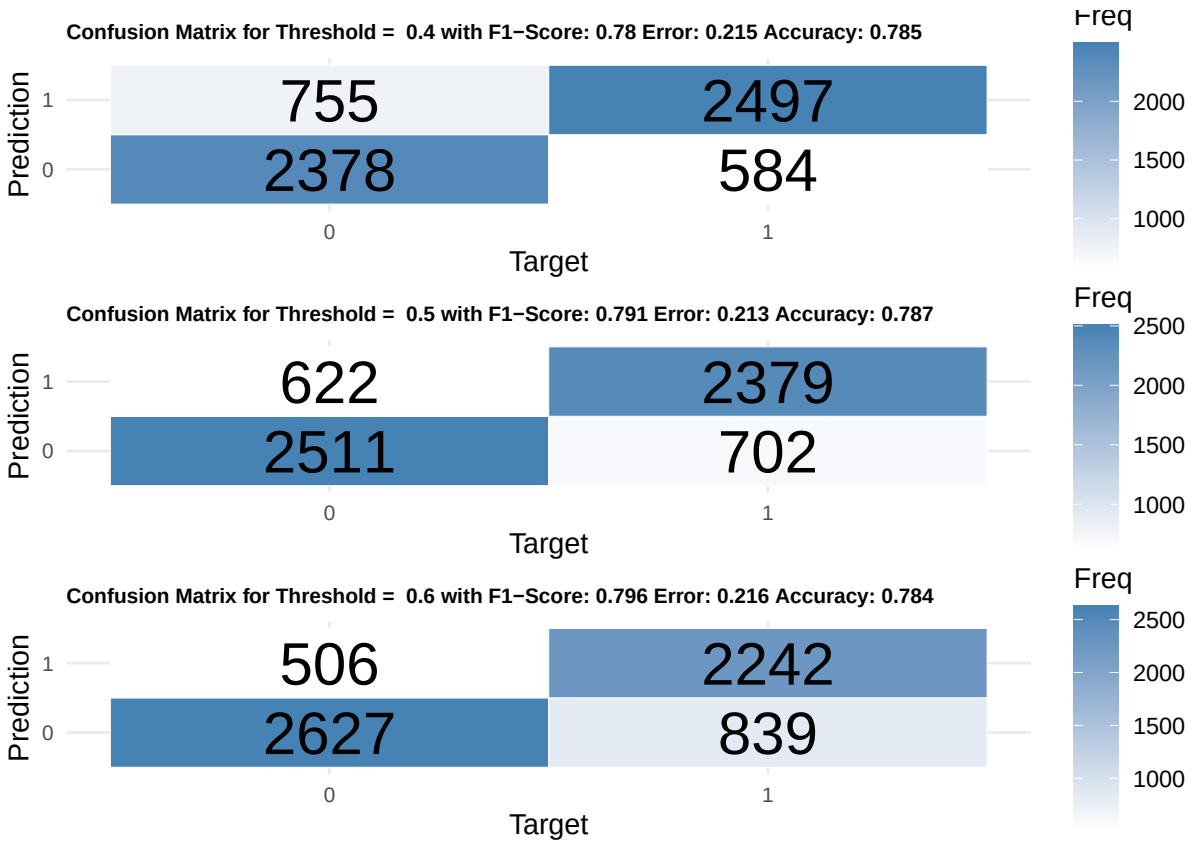
5.3 Quadratic Discriminant Analysis (QDA)

```
qda <-
  qda(data = rain_subset_train_NoOutliers, RainTomorrow ~ ., family = "binomial")
qda
```

```
## Call:
## qda(RainTomorrow ~ ., data = rain_subset_train_NoOutliers, family = "binomial")
##
## Prior probabilities of groups:
##         0         1
## 0.4986052 0.5013948
##
## Group means:
##     MinTemp  Sunshine WindGustSpeed WindSpeed9am WindSpeed3pm Humidity3pm
## 0 0.5198474 0.5981404     0.2639337    0.2044065    0.2354919   0.4472972
## 1 0.5566944 0.3136079     0.3289740    0.2318952    0.2608863   0.6693281
##   Pressure9am Pressure3pm  Cloud9am  Cloud3pm   Temp3pm RainToday
## 0   0.6284978   0.6232747 0.4703169 0.4187170 0.4626215 0.1561222
## 1   0.5583618   0.5622434 0.7413733 0.6944504 0.3922791 0.4574149
```

```
pred_qda <- predict(qda, rain_subset_test, type = "response")

post_qda <- pred_qda$posterior

pred_qda_04 <- as.factor(ifelse(post_qda[, 2] > threshold4, 1, 0))
pred_qda_05 <- as.factor(ifelse(post_qda[, 2] > threshold5, 1, 0))
pred_qda_06 <- as.factor(ifelse(post_qda[, 2] > threshold6, 1, 0))


# Confusion matrices and performance metrics below indicate very little
#difference across the QDA models with the different thresholds.

# Confusion matrix with threshold = 0.4
```

```r
error_qda4 <- mean(pred_qda_04 != rain_subset_test$RainTomorrow)
accuracy_qda4 <- mean(pred_qda_04 == rain_subset_test$RainTomorrow)
qda_CM04 <-
  confusionMatrix(
    data = factor(pred_qda_04),
    reference = factor(rain_subset_test$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.5
error_qda5 <- mean(pred_qda_05 != rain_subset_test$RainTomorrow)
accuracy_qda5 <- mean(pred_qda_05 == rain_subset_test$RainTomorrow)
qda_CM05 <-
  confusionMatrix(
    data = factor(pred_qda_05),
    reference = factor(rain_subset_test$RainTomorrow),
    mode = 'everything'
  )

# Confusion matrix with threshold = 0.6
error_qda6 <- mean(pred_qda_06 != rain_subset_test$RainTomorrow)
accuracy_qda6 <- mean(pred_qda_06 == rain_subset_test$RainTomorrow)
qda_CM06 <-
  confusionMatrix(
    data = factor(pred_qda_06),
    reference = factor(rain_subset_test$RainTomorrow),
    mode = 'everything'
  )

A <-
  create_confusion_matrix(qda_CM04, 0.4, error_qda4, accuracy_qda4)
B <-
  create_confusion_matrix(qda_CM05, 0.5, error_qda5, accuracy_qda5)
C <-
  create_confusion_matrix(qda_CM06, 0.6, error_qda6, accuracy_qda6)

# Threshold of 0.05 is the best among thresholds in terms of accuracy,
# sensitivity, and specificity
CM_all_qda = list(A, B, C)
plot_width <- c(4, 4, 4)
grid.arrange(grobs = CM_all_qda,
             nrow = 3,
             width = plot_width)
```



6 K-Nearest Neighbors (kNN)

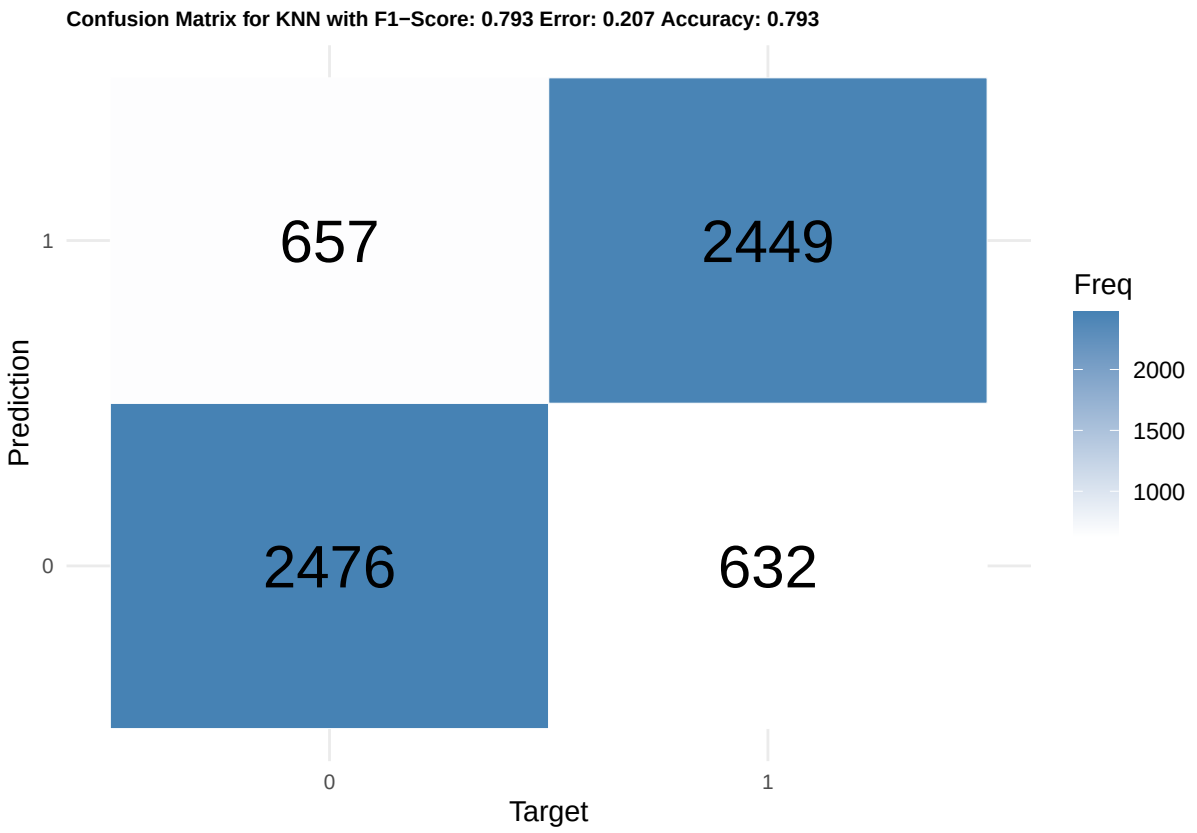
K-Nearest Neighbors (KNN) is a non-parametric, supervised learning technique, which assumes that features that are close in the feature space belong to the same class. The model identifies the labeled K-closest points to an observed point, and assigns the point to the majority class of the said-K-nearest neighbors. KNN is very sensitive to feature scaling and choice of K. If K is too small, KNN might lead to overfitting as each point is classified according to a small set of training examples. Choosing a K that is too large, however, may lead to less overfitting, but with the tradeoff of higher bias and too simple of a model.

```
set.seed(2531)

# We look now for the best value of the parameter
kmax <- 100
knn_test_error <- numeric(kmax)

# For each possible value of k we consider the obtained accuracy of the model
for (k in 1:kmax)
{
  knn_pred <-
    as.factor(knn(X_train_subset, X_test_subset, cl = y_train_subset, k = k))

  cm <- confusionMatrix(data = knn_pred, reference = y_test_subset)

  knn_test_error[k] <- 1 - cm$overall[1]
}

# We took the minimum value of the error
k_min <- which.min(knn_test_error)
```

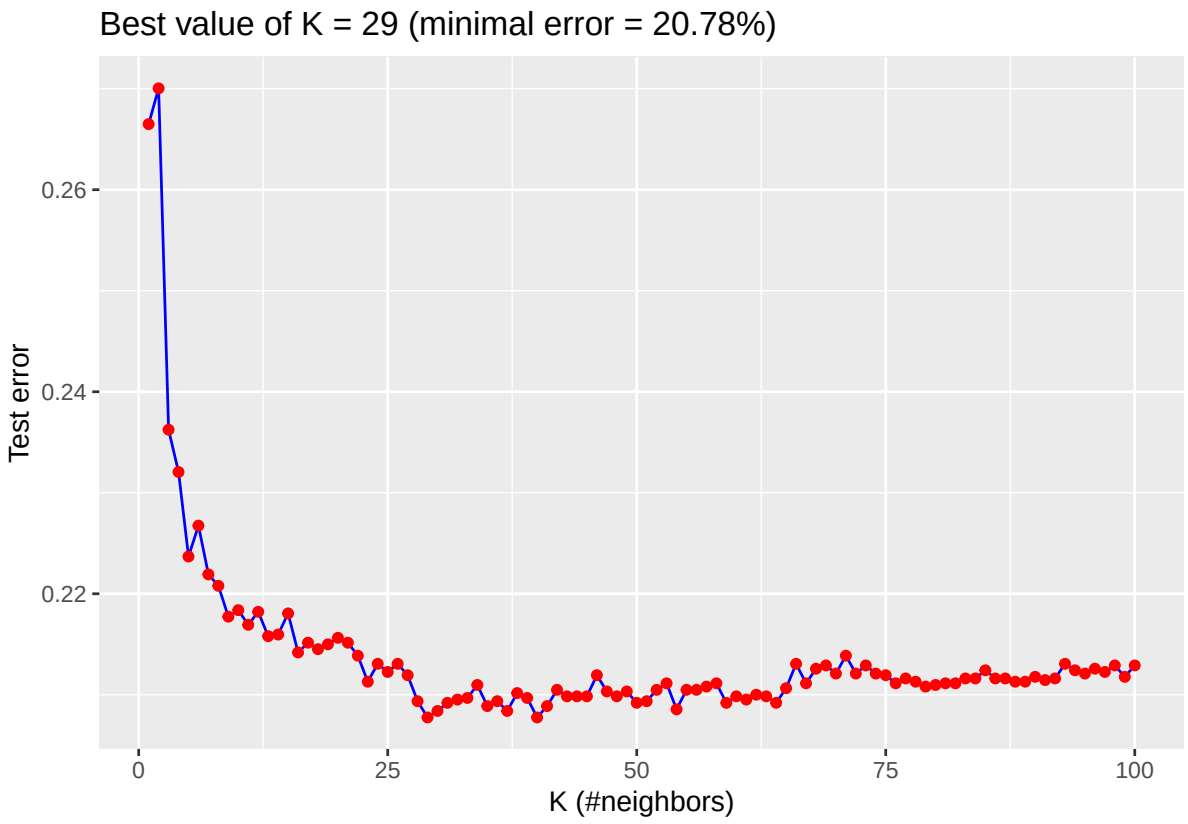
```r
k_min
```

```
## [1] 29
```

```r
# We compute now the prediction with the value of k that gives us the minimum error
knn <-
  knn(X_train_subset, X_test_subset, cl = y_train_subset, k = k_min)

knn_pred_min <- knn

# Confusion matrix for KNN on the test set
cm_knn <-
  confusionMatrix(data = knn_pred_min ,
                  reference = rain_subset_test$RainTomorrow,
                  mode = 'everything')

cm_table_knn <- as.data.frame(cm_knn$table)

f1_score_knn <- cm_knn$byClass["F1"]
error_knn <- mean(knn_pred_min != rain_subset_test$RainTomorrow)
accuracy_knn <- mean(knn_pred_min == rain_subset_test$RainTomorrow)

ggplot(data = cm_table_knn, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), color = "black", size = 8) +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(
    title = paste(
      "Confusion Matrix for KNN with F1-Score:",
      round(f1_score_knn, 3),
      "Error:",
      round(error_knn, 3) ,
      "Accuracy:",
      round(accuracy_knn, 3)
    ),
    x = "Target",
    y = "Prediction"
  ) +
  theme_minimal() +
  theme(axis.text = element_text(size = 8),
        plot.title = element_text(size = 8, face = "bold"))
```




```
#Plot test error against different levels of K
ggplot(data.frame(knn_test_error),
       aes(x = 1:kmax, y = knn_test_error)) +
       geom_line(colour="blue") +
       geom_point(colour="red") +
       xlab("K (#neighbors)") +
       ylab("Test error") +
       ggtitle(paste0("Best value of K = ", k_min,
                      " (minimal error = ",
                     format((knn_test_error[k_min])*100, digits = 4),
                     "%)"))
```



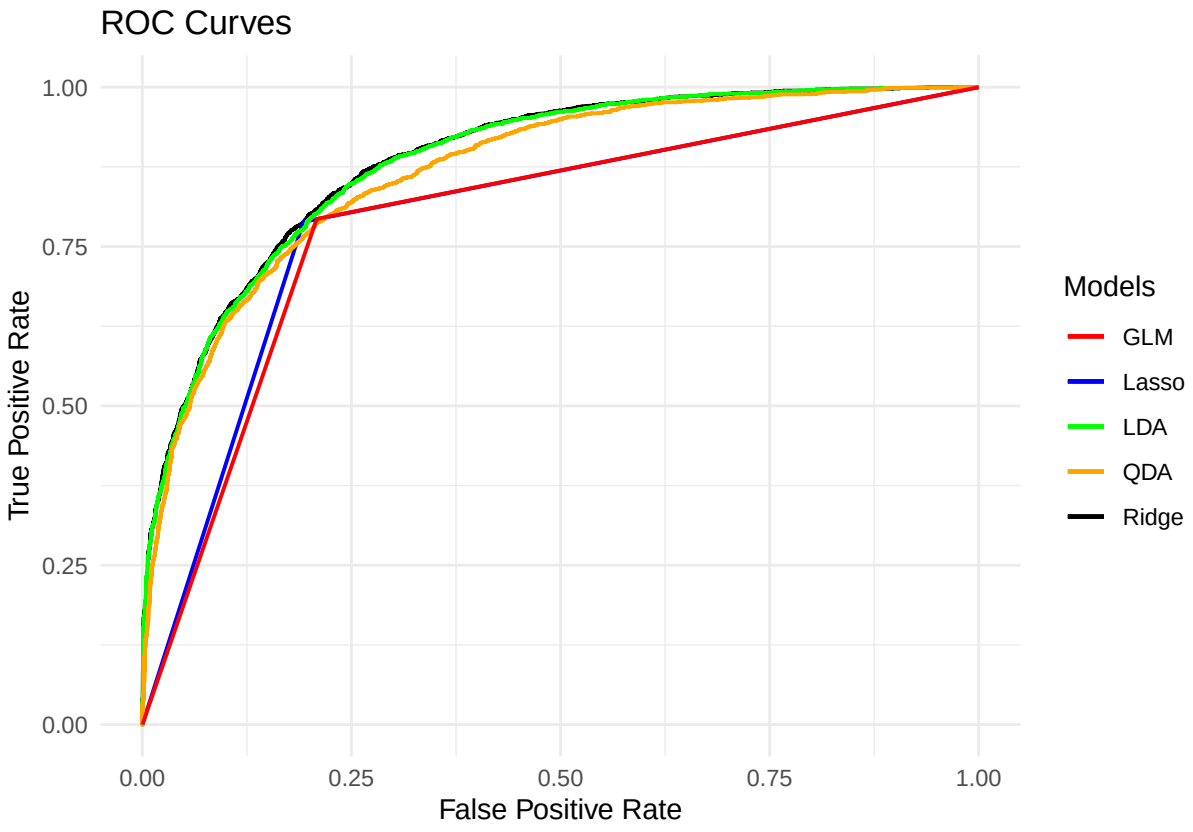
7 ROC Curves

In this final section, we are going to compare all the models applied to balanced and subsetted data and generate a ROC curve to plot the False Positive rate (x-axis) against the True Positive rate (y-axis). This ultimately allows us to visualize the specificity, deduced from the false positives (1-false positive = specificity), and sensitivity from the true positives of each model and between models. We excluded KNN from this analysis, since we are unable to analyze thresholds for predicting classes since the model generates strict 0/1 classifications.

```
# Best Models:

set.seed(125)

# GLM Model
best_glm_model <- glm_model

best_glm_predict <- glm_predict

# LDA Model
best_lda_model <-
  lda(data = rain_subset_train, RainTomorrow ~ ., family = "binomial")

best_lda_predict <-
  predict(lda, rain_subset_test, type = "response")
best_post_lda <- best_lda_predict$posterior

# QDA Model
best_qda_model <-
  qda(data = rain_subset_train, RainTomorrow ~ ., family = "binomial")
```

```
best_qda_predict <-
  predict(qda, rain_subset_test, type = "response")
best_post_qda <- best_qda_predict$posterior

# LASSO Model
best_lasso_model <- lasso_cv_subset

best_lasso_predict <- pred_lasso_subset

# Ridge Model
best_ridge_model <- ridge_cv_subset

best_ridge_predict <- pred_ridge_subset

prediction <-
  tibble(truth = as.factor(rain_subset_test$RainTomorrow))
prediction <-
  prediction %>% mutate(pred = as.numeric(best_glm_predict)) %>%
  mutate(model = "GLM") %>%
  add_row(
    truth = as.factor(rain_subset_test$RainTomorrow),
    pred = as.numeric(best_post_lda[, 2]),
    model = "LDA"
  ) %>%
  add_row(
    truth = as.factor(rain_subset_test$RainTomorrow),
    pred = as.numeric(best_post_qda[, 2]),
    model = "QDA"
  ) %>%
  add_row(
    truth = as.factor(rain_subset_test$RainTomorrow),
    pred = as.numeric(best_ridge_predict),
    model = "Ridge"
  ) %>%
  add_row(
    truth = as.factor(rain_subset_test$RainTomorrow),
    pred = as.numeric(best_lasso_predict),
    model = "Lasso"
  )

# Calculate ROC curves for each model
roc_list <-
  lapply(split(prediction, prediction$model), function(df) {
    roc.out <- roc(df$truth, df$pred, levels = c(0, 1))
    roc.df <- data.frame(specificity = roc.out$specificities,
                         sensitivity = roc.out$sensitivities)
    return(roc.df)
  })

model_colors <- c("red", "blue", "green", "orange", "black")
#List of Model Names
model_names <- names(roc_list)

ggplot() +
  lapply(seq_along(roc_list), function(i) {
    geom_line(
```

```
    data = roc_list[[i]],
    aes(
      x = 1 - specificity,
      y = sensitivity,
      color = model_colors[i]
    ),
    linetype = "solid",
    linewidth = 0.7
  )
}) +
lapply(seq_along(roc_list), function(i) {
  labs(
    title = "ROC Curves",
    x = "False Positive Rate",
    y = "True Positive Rate",
    color = "Models"
  )
}) +
scale_color_manual(values = model_colors, labels = model_names) +
theme_minimal()
```



8 Discussion

The area under each of the ROC curves for each model serves as an indicator for how well each model predicts RainTomorrow.The larger the area under the curve (AUC), the higher the recall of the model is (positivity rate). We can observe that the ridge, LDA, and QDA show a better AUC compared to GLM and LASSO models. In some cases concerning fatalities or hazardous situations, we care more about achieving a tradeoff between false positives and true positives in order to minimize false negatives. Thus, the ROC curves are suitable for modeling these cases. However, in our case, we care

more generally about a model that predicts well than having this tradeoff between false positives and true positives. On the other hand, other projects using this data may give higher consideration to positive rain predictions, such as projects in the agriculture sector, where having rain tomorrow may impact the harvest season. To address our concern with prioritizing more precise predictions over the positivity rate, we extended our final analysis of the models by considering the key performance metrics, i.e. F1 Score, accuracy, error rate, for each model.

```
file_path3 <- "/Users/Sofia/Desktop/Rain_Australia/metrics_summary.csv"
metrics_summary <- read.csv(file_path3)

options(knitr.kable.NA = '')
metrics_summary %>%
  knitr::kable(
    format = "latex",
    align = "l",
    booktabs = TRUE,
    longtable = TRUE,
    linesep = "",
    caption = "Table of performance metrics by model."
    ) %>% row_spec(0,bold=TRUE) %>%
  kableExtra::kable_styling(
      position = "left",
      latex_options = c("striped", "repeat_header"),
      stripe_color = "gray!15"
    )
```

Table 2: Table of performance metrics by model.

Model	F1.Score	Error	Accuracy
Simple GLM 0.4	0.903	0.153	0.847
Simple GLM 0.5	0.907	0.149	0.851
Simple GLM 0.6	0.907	0.153	0.847
GLM with Balanced Dataset-0.4	0.781	0.205	0.795
GLM with Balanced Dataset-0.5	0.796	0.205	0.795
GLM with Balanced Dataset-0.6	0.798	0.215	0.785
GLM with balancing and feature selection-0.4	0.781	0.206	0.794
GLM with balancing and feature selection-0.5	0.795	0.205	0.795
GLM with balancing and feature selection-0.6	0.797	0.216	0.784
LDA with balancing and feature seletion-0.4	0.783	0.207	0.793
LDA with balancing and feature seletion-0.5	0.792	0.210	0.790
LDA with balancing and feature seletion-0.6	0.795	0.220	0.780
QDA with balancing and feature seletion-0.4	0.776	0.217	0.783
QDA with balancing and feature seletion-0.5	0.785	0.216	0.784
QDA with balancing and feature seletion-0.6	0.788	0.220	0.780
Ridge Regression with balancing	0.786	0.213	0.787
Ridge Regression with balancing and feature selection	0.786	0.213	0.787
Lasso Regression with balancing	0.796	0.205	0.795
Lasso Regression with balancing and feature selection	0.794	0.206	0.794
KNN	0.782	0.216	0.784

While in the ROC curves, Ridge, LDA and QDA performed better, the GLM models had the best performance metrics overall. These results suggest that compared to other models, the logistic regression model is the most precise in predicting RainTomorrow.