Will it Rain Tomorrow in Australia?

2023-07-15

Import R Libraries

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(ggplot2)
library(caret)
```

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

```
library(magrittr)
library(gridExtra)
library(scales)
library(DMwR2)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(UBL)
```

```
## Loading required package: MBA

## Loading required package: gstat

## The legacy packages maptools, rgdal, and rgeos, underpinning this package
## will retire shortly. Please refer to R-spatial evolution reports on
## https://r-spatial.org/r/2023/05/15/evolution4.html for details.
## This package is now running under evolution status 0

## Loading required package: automap

## Loading required package: sp

## Loading required package: randomForest

## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(caret)
library(MASS)
library(ipred)
library(rsample)
library(mlr)
```

```
## Loading required package: ParamHelpers

## Warning message: 'mlr' is in 'maintenance-only' mode since July 2019.
## Future development will only happen in 'mlr3'
## (<https://mlr3.mlr-org.com>). Due to the focus on 'mlr3' there might be
## uncaught bugs meanwhile in {mlr} - please consider switching.

##
## Attaching package: 'mlr'

## The following object is masked from 'package:caret':
##
##     train
```

```
library(knitr)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-7
```

```
library(outliers)
```

```
##
## Attaching package: 'outliers'

## The following object is masked from 'package:randomForest':
##
##     outlier
```

```
library(class)
```

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
```

```r
dim(rain)
```

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

##TODO: Create Feature Table of Contents | **Test** | **Encoding** | **Decoding** | **Activation Func** | **Last Layer Activation Func** | **Training Time** | **Total Loss** | **Train Recon Loss** | **Train KL Loss** | | |———-|——————|——————|———————|——————————|——————-|—————-|——————-|——————-|—-| | 1 | [256, 128, 64] | [64, 128, 256] | tanh | sigmoid | 578.813 | 0.17836289 | 0.17136258 | 0.00700030 | | | 2 | [256, 128, 64] | [64, 128, 256] | leaky_relu | sigmoid | 571.390 | 0.17361516 | 0.16642702 | 0.00718812 | | | 3 | [512, 256, 128] | [128, 256, 512] | leaky_relu | sigmoid | 755.329 | 0.17168072 | 0.16437545 | 0.00730522 | | | 4 | [256,128,64] | [64,128,256] | sigmoid | sigmoid | 343.171 | 0.19872129 | 0.19446002 | 0.00426128 | | | | 5 | [512,256,128] | [128,256,512] | sigmoid | sigmoid | 340.039 | 0.19706537 | 0.19261234 | 0.00445290 | | | 6 | [590, 310, 168] | [168, 310, 590] | leaky_relu | sigmoid | 337.860 | 0.17151794 | 0.16409160 | 0.00742632 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ## Data Preprocessing

```r
# 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
```

```r
# 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))

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

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

```
head(rain)
```

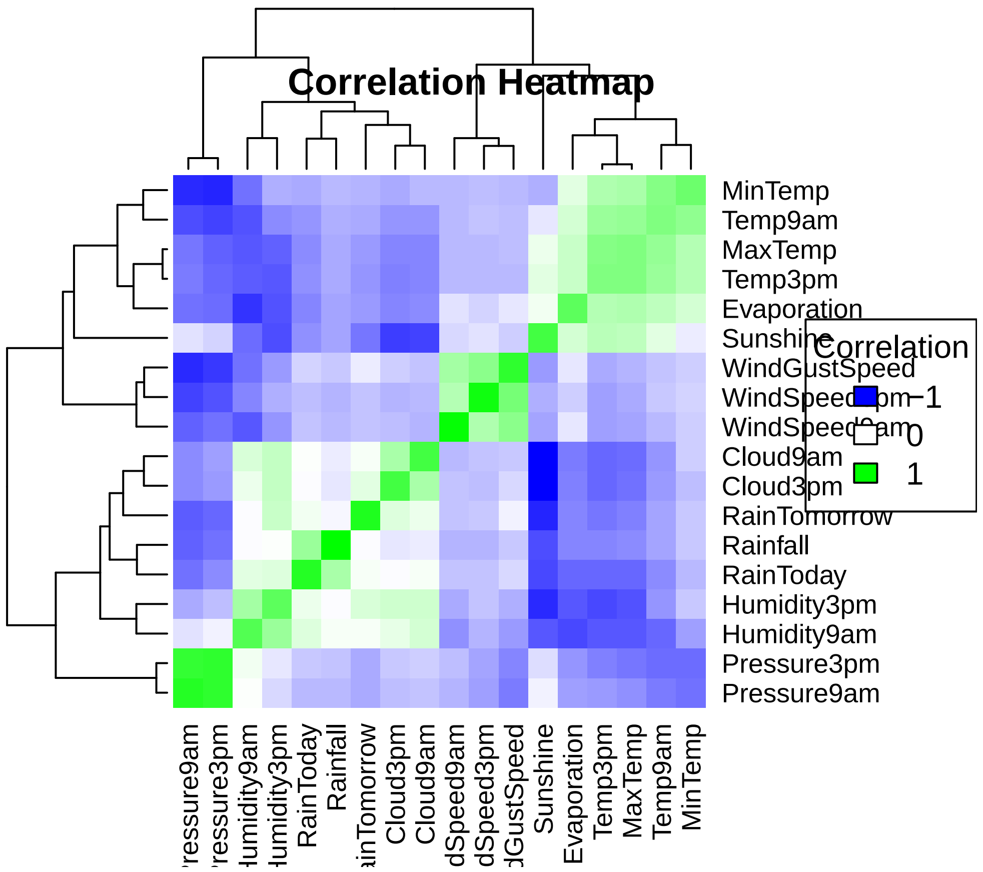
```
##      MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustSpeed WindSpeed9am
## 6050    17.9    35.2        0        12.0     12.3            48            6
## 6051    18.4    28.9        0        14.8     13.0            37           19
## 6053    19.4    37.6        0        10.8     10.6            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
```

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

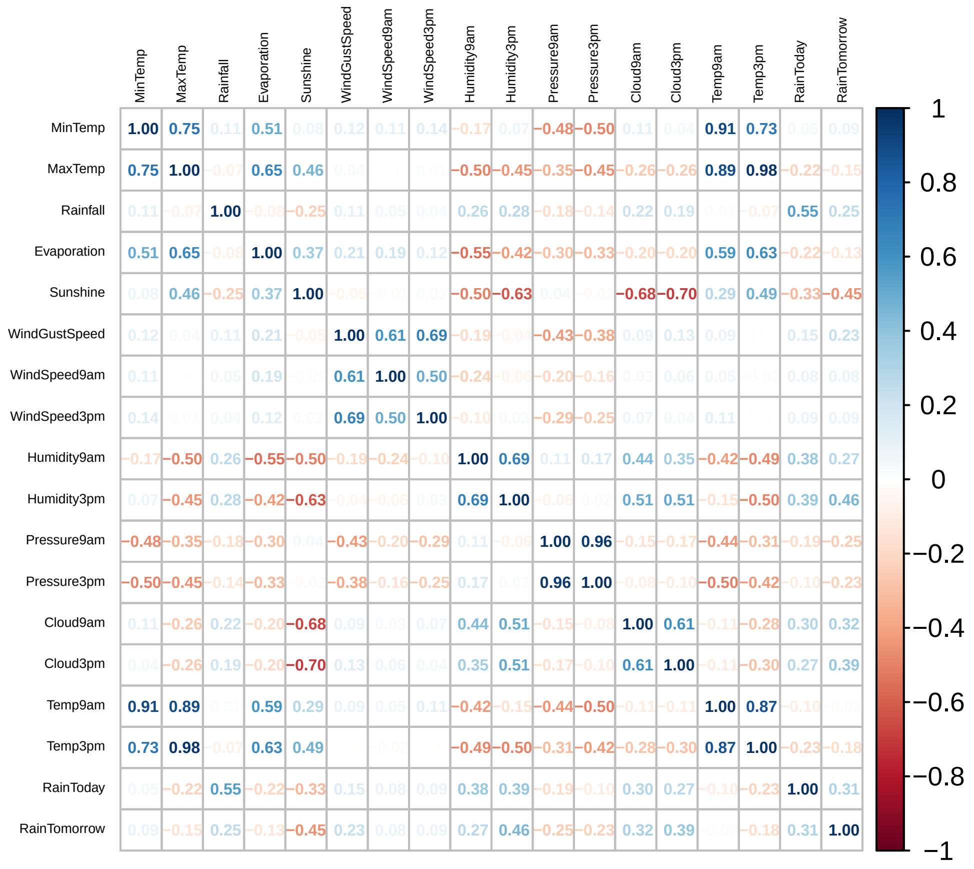
```
##     MinTemp         MaxTemp         Rainfall       Evaporation
##  Min.   :-6.70   Min.   : 4.10   Min.   :  0.00   Min.   : 0.000
##  1st Qu.: 8.60   1st Qu.:18.70   1st Qu.:  0.00   1st Qu.: 2.800
##  Median :13.20   Median :23.90   Median :  0.00   Median : 5.000
##  Mean   :13.46   Mean   :24.22   Mean   :  2.13   Mean   : 5.503
##  3rd Qu.:18.40   3rd Qu.:29.70   3rd Qu.:  0.60   3rd Qu.: 7.400
##  Max.   :31.40   Max.   :48.10   Max.   :206.20   Max.   :81.200
##     Sunshine      WindGustSpeed     WindSpeed9am    WindSpeed3pm
##  Min.   : 0.000   Min.   :  9.00   Min.   : 2.00   Min.   : 2.00
##  1st Qu.: 5.000   1st Qu.: 31.00   1st Qu.: 9.00   1st Qu.:13.00
##  Median : 8.600   Median : 39.00   Median :15.00   Median :19.00
##  Mean   : 7.736   Mean   : 40.88   Mean   :15.67   Mean   :19.79
##  3rd Qu.:10.700   3rd Qu.: 48.00   3rd Qu.:20.00   3rd Qu.:26.00
##  Max.   :14.500   Max.   :124.00   Max.   :67.00   Max.   :76.00
##   Humidity9am      Humidity3pm     Pressure9am      Pressure3pm
##  Min.   :  0.00   Min.   :  0.0   Min.   : 980.5   Min.   : 977.1
##  1st Qu.: 55.00   1st Qu.: 35.0   1st Qu.:1012.7   1st Qu.:1010.1
##  Median : 67.00   Median : 50.0   Median :1017.2   Median :1014.7
##  Mean   : 65.87   Mean   : 49.6   Mean   :1017.2   Mean   :1014.8
##  3rd Qu.: 79.00   3rd Qu.: 63.0   3rd Qu.:1021.8   3rd Qu.:1019.4
##  Max.   :100.00   Max.   :100.0   Max.   :1040.4   Max.   :1038.9
##     Cloud9am        Cloud3pm        Temp9am        Temp3pm
##  Min.   :0.000   Min.   :0.000   Min.   :-0.7   Min.   : 3.70
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:13.1   1st Qu.:17.40
##  Median :5.000   Median :5.000   Median :17.8   Median :22.40
##  Mean   :4.242   Mean   :4.327   Mean   :18.2   Mean   :22.71
##  3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:23.3   3rd Qu.:27.90
##  Max.   :8.000   Max.   :9.000   Max.   :39.4   Max.   :46.10
##    RainToday       RainTomorrow
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000
##  Mean   :0.2209   Mean   :0.2203
##  3rd Qu.:0.0000   3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000
```

Correlation

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

# Create a heatmap from the correlation matrix with blue, white, and green color scheme
heatmap(cor_matrix, col = colorRampPalette(c("blue", "white", "green"))(100))

# Add a color legend to corr_matrix
legend_colors <- c("blue", "white", "green")
legend("right", legend = c(-1, 0, 1), fill = legend_colors, title = "Correlation")

# Add a main title
title(main = "Correlation Heatmap")
```



```r
#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")
```



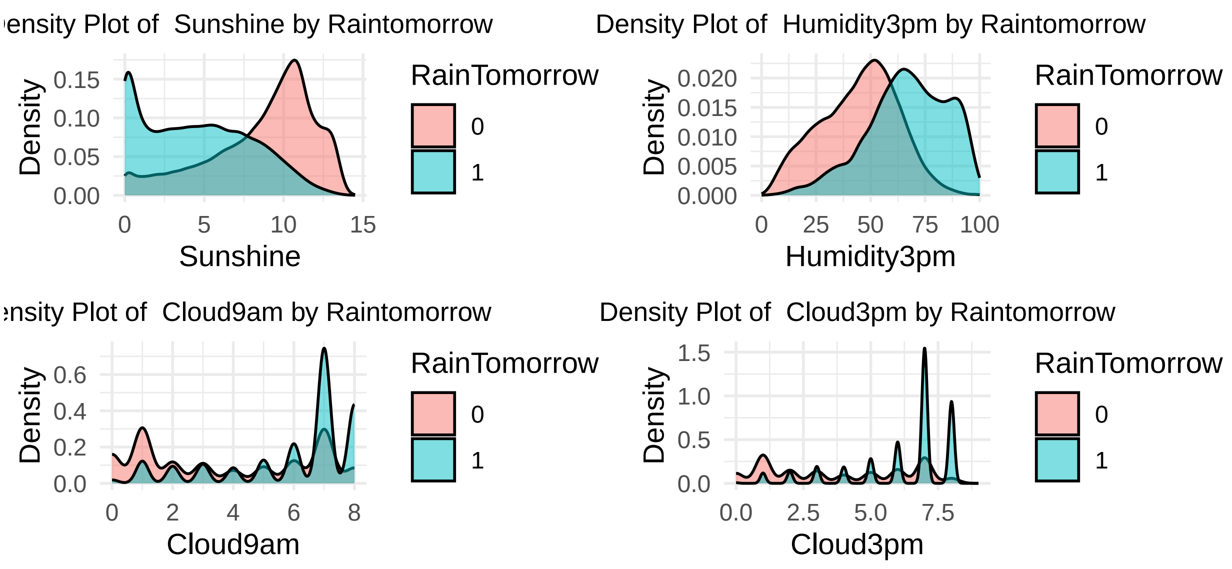
Density Plots

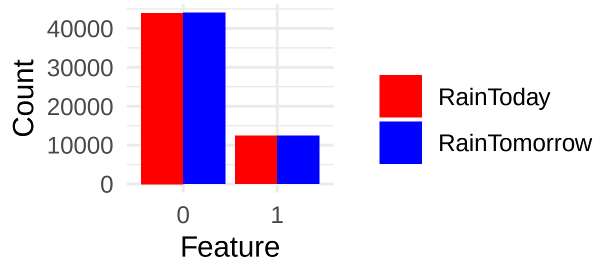
```
## Find features with highest correlation with target variable (RainTomorrow)

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 havin

# Calculate the count of each feature
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)
```

```r
# 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)
```





```
# Sunshine: fraction of total days having higher sunshine record more 0 RainTomorrow, lower sunshine, m
# Humidity3pm: overlap more but still higher humidity associated with 1 RainTomorrow and vice versa
#Cloud9am/Cloud3pm: oscillates a bit across x-axis with higher discrepancies between RainTomorrow value
#RainToday: Since RainToday is a binary variable, the density plots are concentrated around 0 and 1. Wh
```

Feature Scaling

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

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

```
# Handling balancing of data below in train/test split

# 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
```

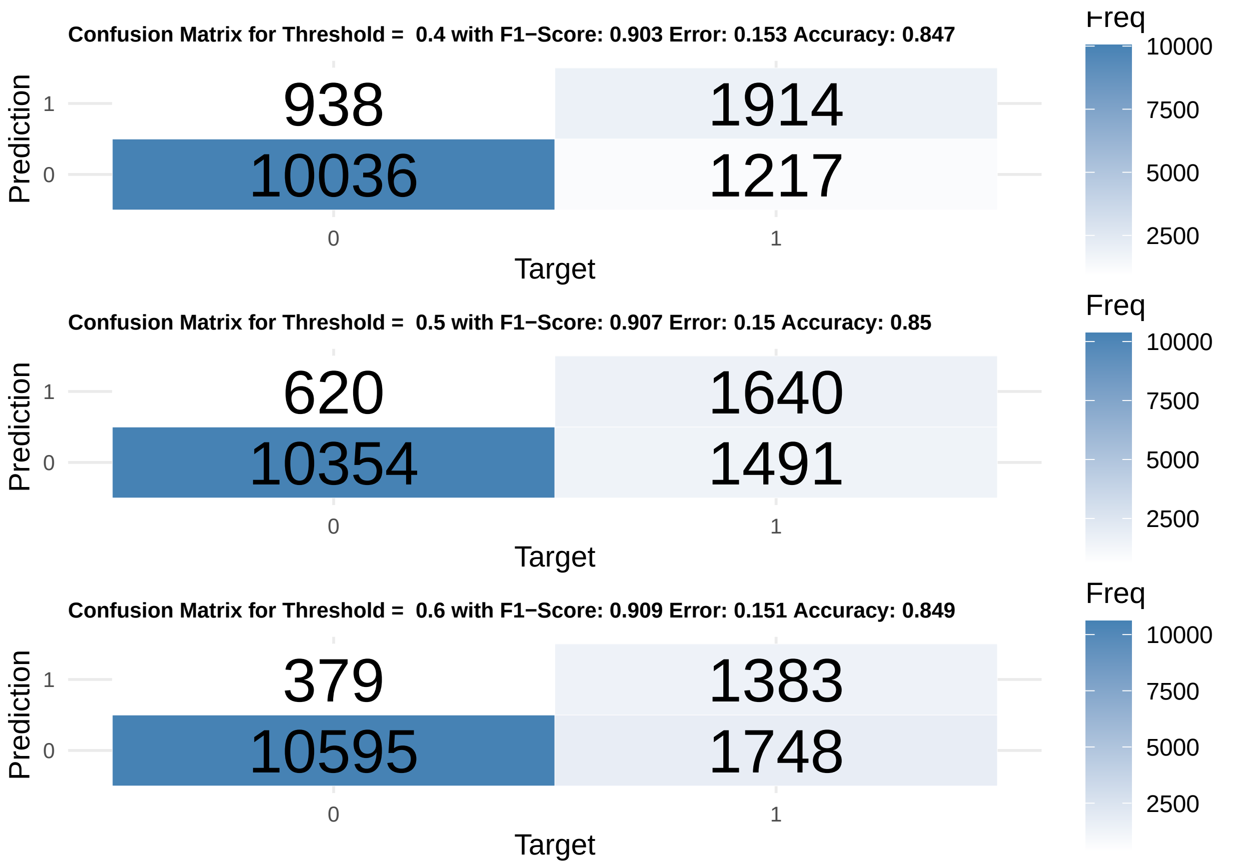
##First Logistic Regression Model: without balancing or feature selection

```
#Test/Train 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, gives the proportion of deviance explained by the model
# VIF: If the VIF value for a predictor variable is greater than 1, it indicates the presence of multic

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
```

```
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

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 mode

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
    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)
```

```r
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),

# 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),

# 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),

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)
```



## Balancing ### The data extracted wasn't balanced, in which 0 was the majority class with over 40,000 and 1 had only about 12,000 samples. To account for this imbalance, we performed a method of downsampling, 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.

```r
#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
```

```r
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
```

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

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

Logistic Regression before Feature Selection

```r
#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)

#R squared and Variance Inflation Factor (VIF)
#If the VIF value for a predictor variable is greater than 1, it indicates the presence of multicolline
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
```

```r
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
```

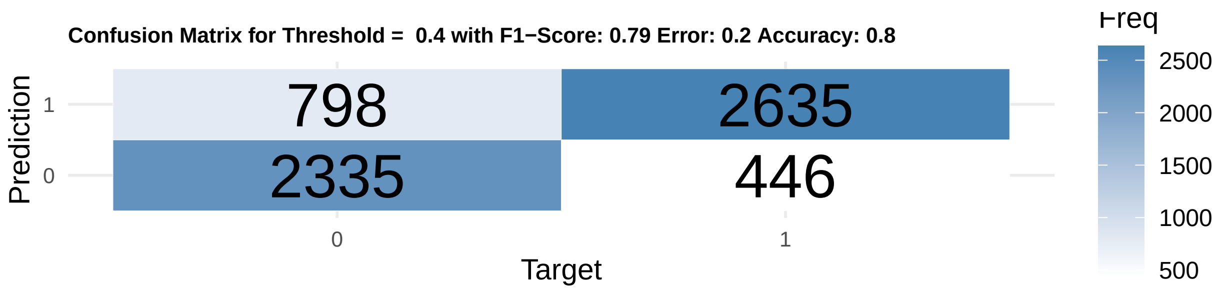
```r
error4_balanced <- mean(glm_predict_4_balanced!=rain_balanced_test$RainTomorrow)
accuracy4_balanced <- mean(glm_predict_4_balanced==rain_balanced_test$RainTomorrow)
cm4_balanced <- confusionMatrix(data = factor(glm_predict_4_balanced), reference = factor(rain_balanced

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

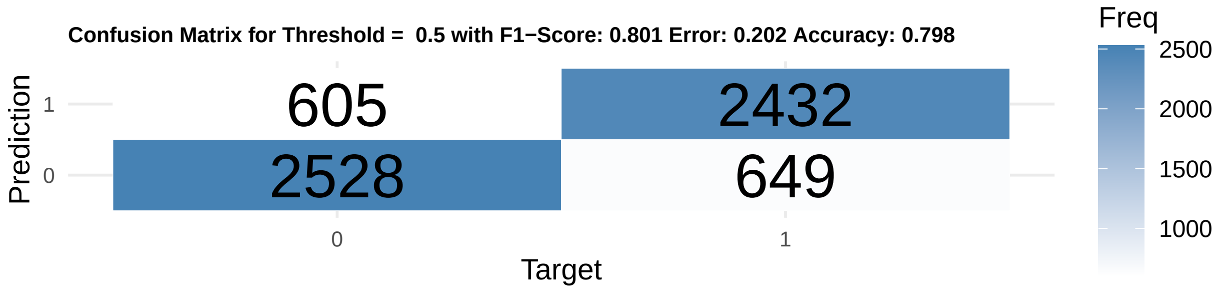
```
##    glm_predict_5_balanced
##        0    1
##   0 2528  605
##   1  649 2432
```

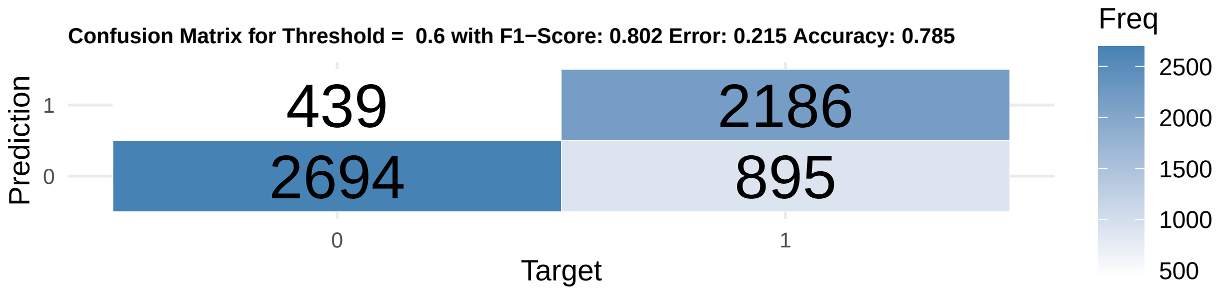
```r
error5_balanced <- mean(glm_predict_5_balanced!=rain_balanced_test$RainTomorrow)
accuracy5_balanced <- mean(glm_predict_5_balanced==rain_balanced_test$RainTomorrow)
cm5_balanced <- confusionMatrix(data = factor(glm_predict_5_balanced), reference = factor(rain_balanced

# 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
```

```r
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

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)
```







## Feature Selection (Backward Selection using BIC)

```r
# Perform logistic regression with backward stepwise selection
logit_model <- glm(rain_balanced$RainTomorrow ~ ., data = rain_balanced, family = binomial)

# Perform forward stepwise selection using AIC 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"
```

#Train/Test Split

```
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,]

head(rain_subset_train)
```

```
##         MinTemp  Sunshine WindGustSpeed WindSpeed9am WindSpeed3pm Humidity3pm
## 18847 0.4934383 0.3931034     0.3565217   0.10769231    0.2297297        0.71
## 18895 0.8188976 0.1931034     0.4869565   0.16923077    0.2432432        0.86
## 2986  0.6220472 0.2413793     0.4086957   0.33846154    0.4189189        0.18
## 1842  0.7034121 0.7793103     0.3391304   0.06153846    0.3513514        0.40
## 3371  0.4619423 0.4689655     0.1652174   0.16923077    0.2297297        0.27
## 11638 0.5170604 0.1448276     0.2086957   0.07692308    0.2702703        0.73
##       Pressure9am Pressure3pm Cloud9am  Cloud3pm   Temp3pm RainToday
## 18847   0.5392321   0.5841424    0.125 0.3333333 0.3537736         1
## 18895   0.4190317   0.4271845    0.875 0.8888889 0.5306604         1
## 2986    0.5592654   0.5695793    0.875 0.5555556 0.4811321         0
## 1842    0.5091820   0.4870550    0.250 0.3333333 0.6910377         0
## 3371    0.5876461   0.5776699    0.875 0.6666667 0.5070755         0
## 11638   0.6711185   0.7103560    0.625 0.8888889 0.3278302         1
##       RainTomorrow
## 18847            1
## 18895            1
## 2986             0
## 1842             0
## 3371             0
## 11638            0
```

```
head(rain_subset_test)
```

```
##      MinTemp  Sunshine WindGustSpeed WindSpeed9am WindSpeed3pm Humidity3pm
## 1  0.6692913 0.1172414     0.2608696    0.2307692   0.24324324        0.71
## 6  0.6141732 0.7103448     0.2782609    0.2615385   0.22972973        0.45
## 8  0.5433071 0.6827586     0.4173913    0.3384615   0.32432432        0.39
## 12 0.1863517 0.6413793     0.1304348    0.1076923   0.17567568        0.57
## 24 0.4881890 0.2000000     0.2086957    0.2307692   0.14864865        0.86
## 29 0.3569554 0.8758621     0.1913043    0.1692308   0.09459459        0.24
##    Pressure9am Pressure3pm Cloud9am  Cloud3pm   Temp3pm RainToday RainTomorrow
## 1    0.6494157   0.6521036    0.875 0.7777778 0.4363208         1            0
## 6    0.5409015   0.5679612    1.000 0.1111111 0.4740566         0            0
## 8    0.6093489   0.6682848    0.875 0.6666667 0.3844340         0            0
## 12   0.6627713   0.6423948    0.750 0.1111111 0.2948113         1            0
## 24   0.6544240   0.6100324    0.875 0.8888889 0.2735849         0            0
## 29   0.7412354   0.7103560    0.125 0.1111111 0.4599057         0            0
```

Selected Model

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

#R squared and Variance Inflation Factor (VIF)
#If the VIF value for a predictor variable is greater than 1, it indicates the presence of multicolline
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
table(rain_subset_test$RainTomorrow, glm_predict_4)
```

```
##    glm_predict_4
##        0    1
##   0 2338  795
##   1  444 2637
```

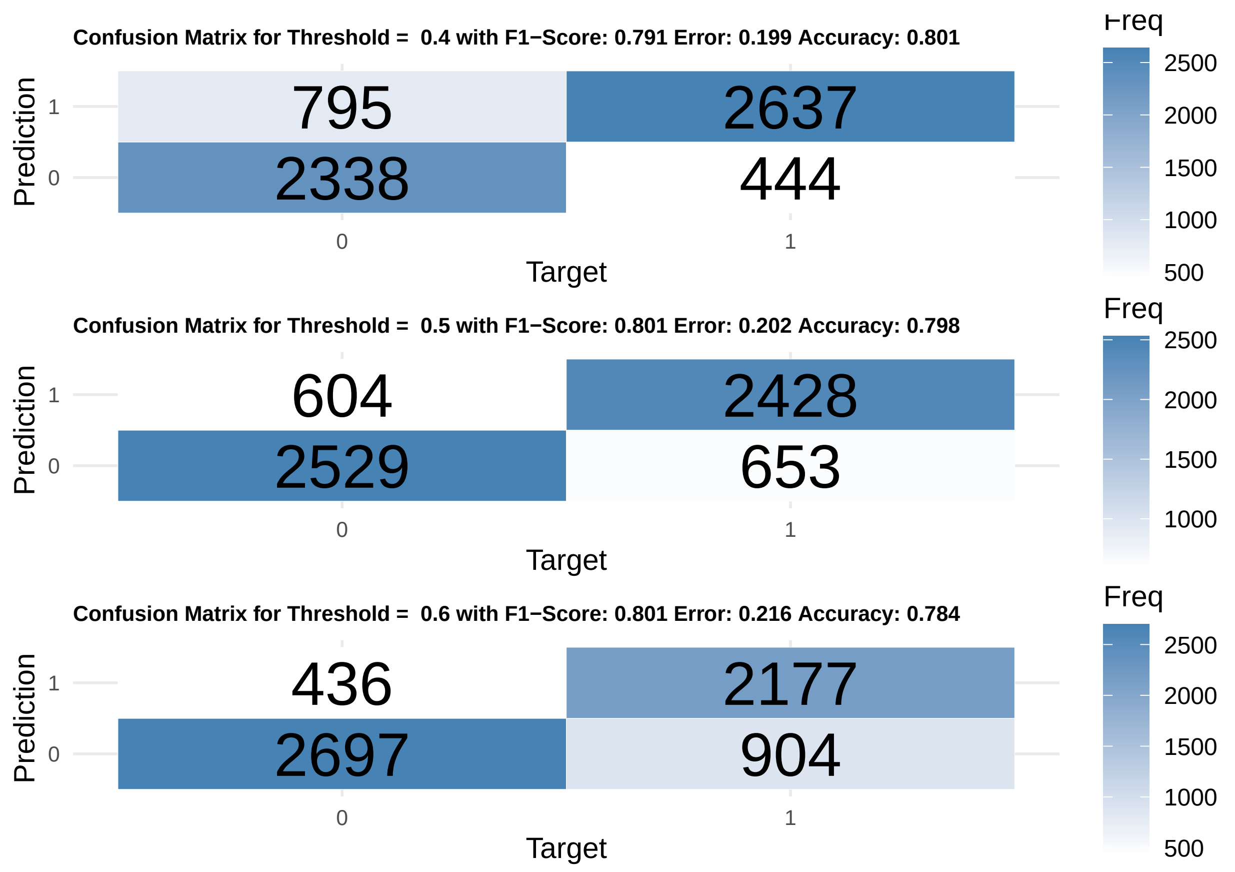
```
error4 <- mean(glm_predict_4!=rain_subset_test$RainTomorrow)
accuracy4 <- mean(glm_predict_4==rain_subset_test$RainTomorrow)
cm4 <- confusionMatrix(data = factor(glm_predict_4), reference = factor(rain_subset_test$RainTomorrow),

# Confusion matrix with threshold = 0.5
table(rain_subset_test$RainTomorrow, glm_predict_5)
```

```
##    glm_predict_5
##        0    1
##   0 2529  604
##   1  653 2428
```

```
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),

# Confusion matrix with threshold = 0.6
table(rain_subset_test$RainTomorrow, glm_predict_6)
```

```
##    glm_predict_6
##        0    1
##    0 2697  436
##    1  904 2177
```

```
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),

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)
```



##Comparison of 3 Models

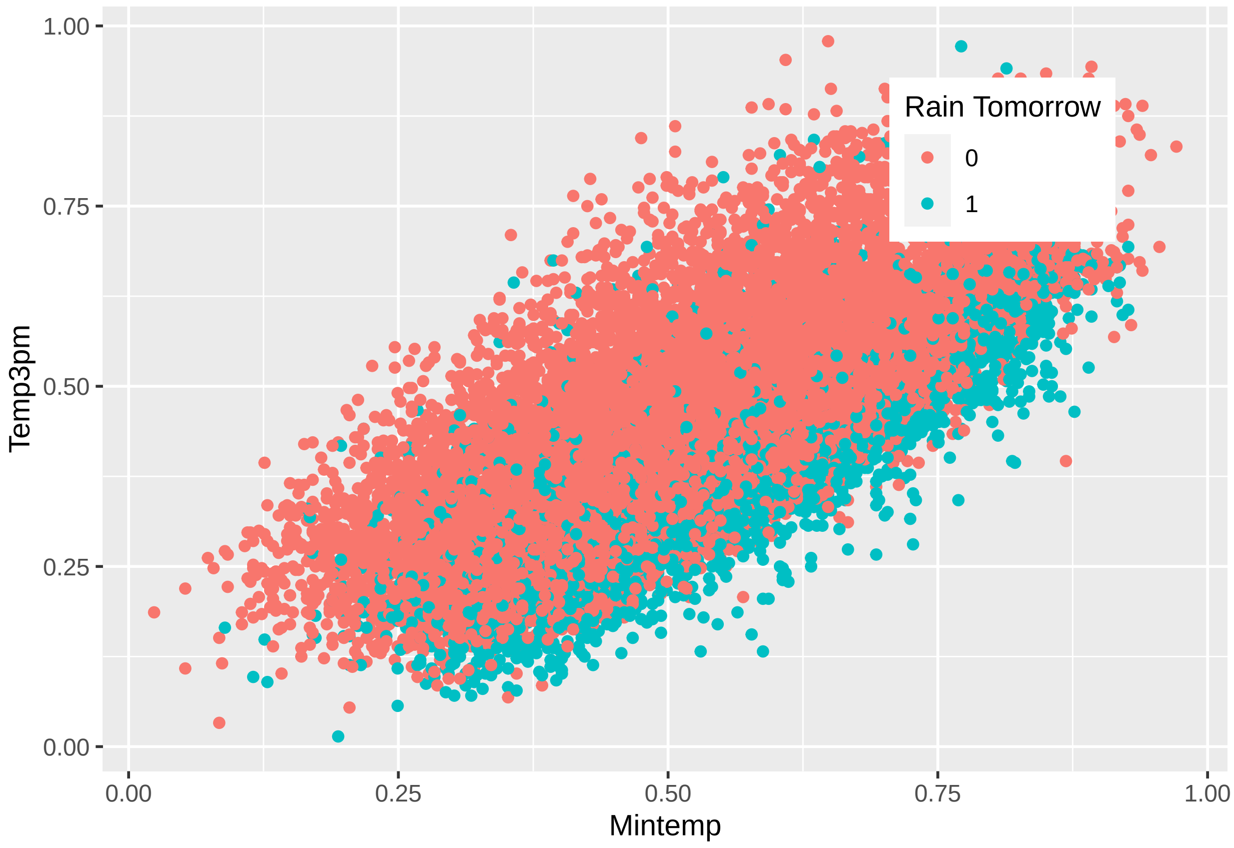
```
## Summary Statistics: F1-Score, Error, Accuracy

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

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_scor

  }

}


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
```

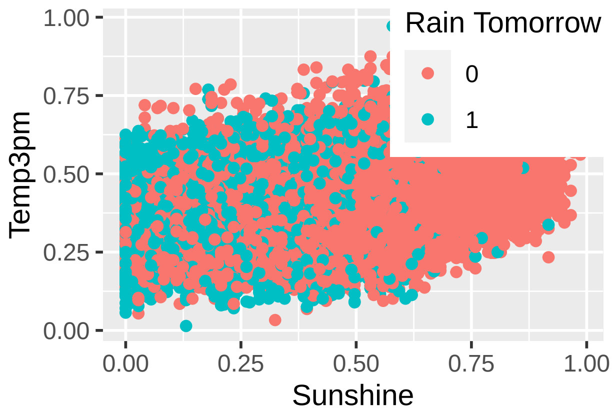
##Predictions

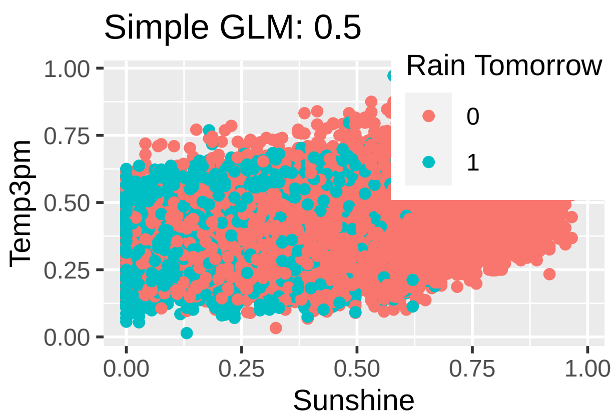
```
#for different features: mintemp an temp3pm
a00 <- ggplot(data = rain_n_test0 , aes(x = MinTemp,
y = Temp3pm,
color= as.factor(RainTomorrow) )) +
geom_point()+
labs(x = "Mintemp",
y = "Temp3pm",
color = "Rain Tomorrow") +
theme(legend.position = c(0.8, 0.8))

print(a00)
```

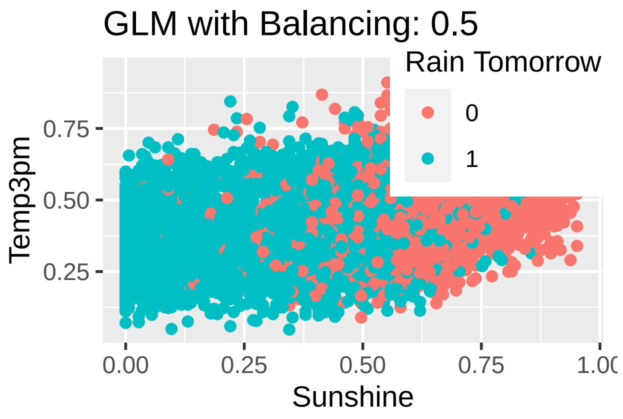


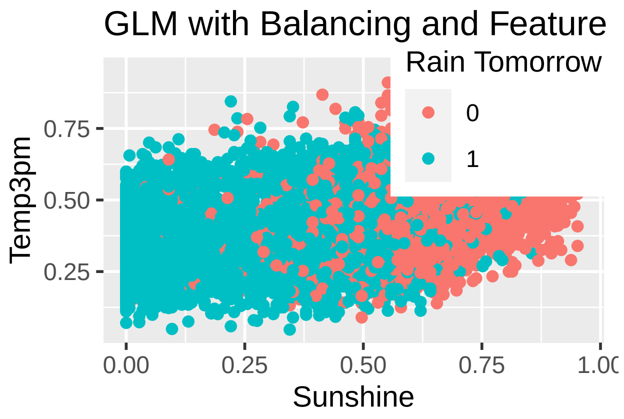
```
#since the 2 features are highly correlated, we cannot see a clear separation of the classes in the sca


# 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()+
```

```r
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)
```







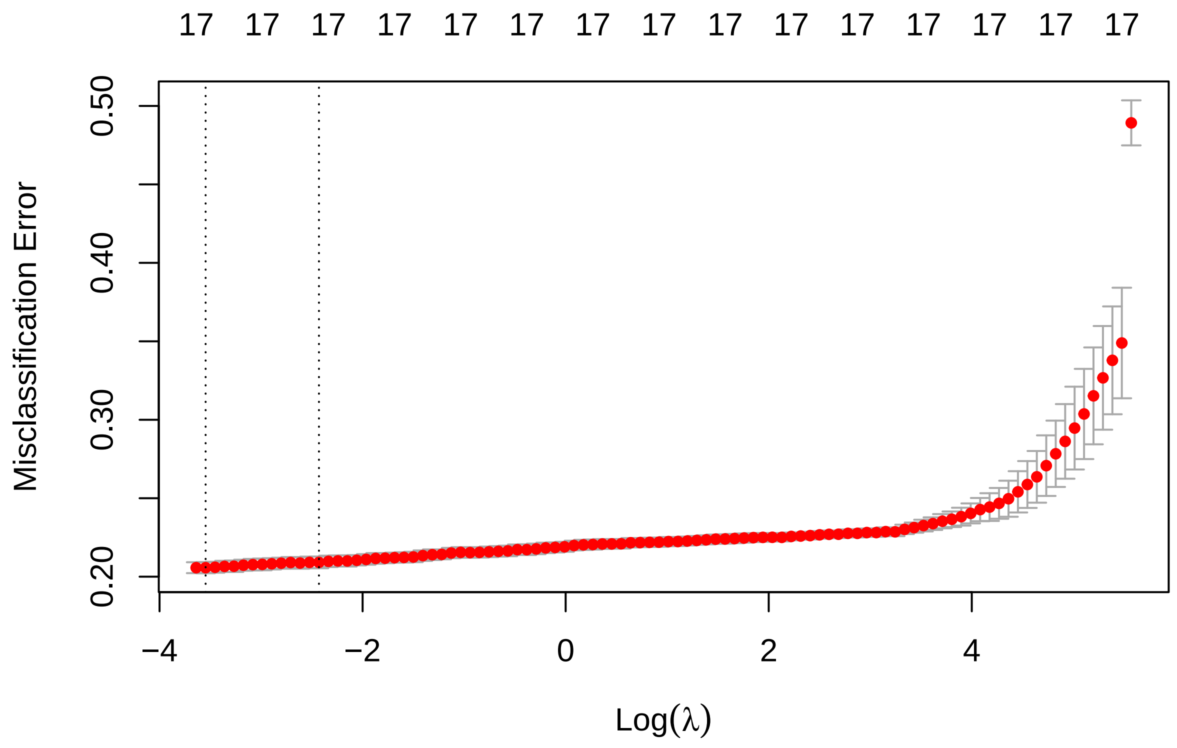


```r
X_train_subset <- model.matrix(RainTomorrow~., data=rain_subset_train)
X_test_subset <- model.matrix(RainTomorrow~., data=rain_subset_test)
X_train_subset <- X_train_subset[,-1]
X_test_subset <- X_test_subset[,-1]

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

LASSO and Ridge Regression

```r
set.seed(123)
#we're implementing lasso/ridge with balanced data with no selection ie. rain_balanced_train

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]
X_test <- X_test[,-1]

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

```r
# 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)
```



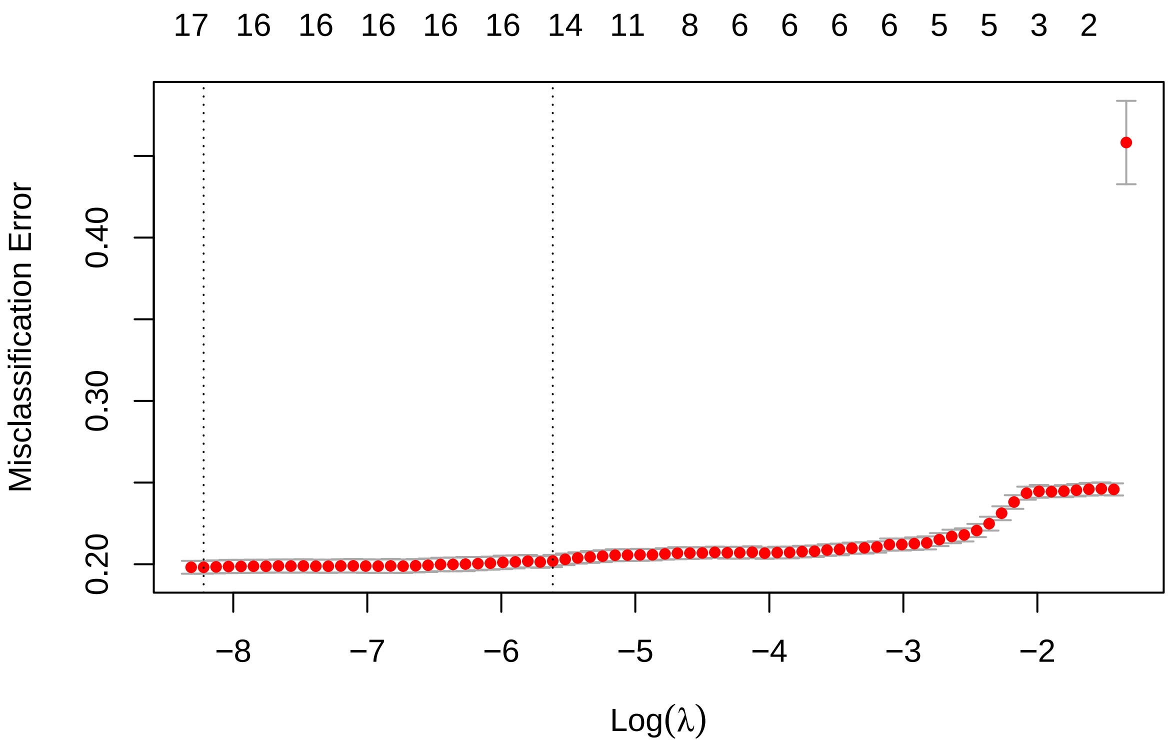
```r
# to select best lambda
lambda_opt_ridge <- ridge_cv$lambda.min
lambda_opt_ridge
```

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

```r
pred_ridge<- predict(ridge_cv, X_test, type = "class", s = lambda_opt_ridge)
table(y_test, pred_ridge)
```

```
##       pred_ridge
## y_test    0    1
##      0 2469  664
##      1  633 2448
```

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



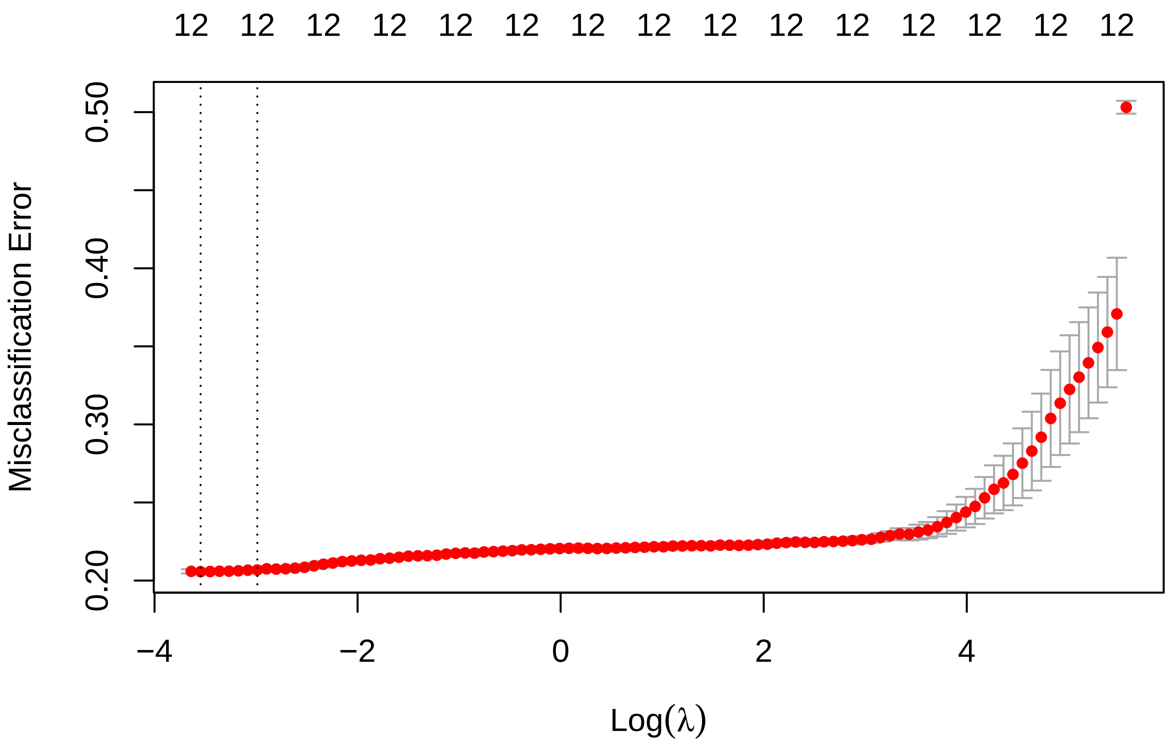
```
lambda_opt_lasso <- lasso_cv$lambda.min
lambda_opt_lasso
```

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

```
pred_lasso<- predict(lasso_cv, X_test, type = "class", s = lambda_opt_lasso)
table(y_test, pred_lasso)
```

```
##       pred_lasso
## y_test    0    1
##      0 2529  604
##      1  652 2429
```

```
#Implement 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
plot(ridge_cv_subset)
```



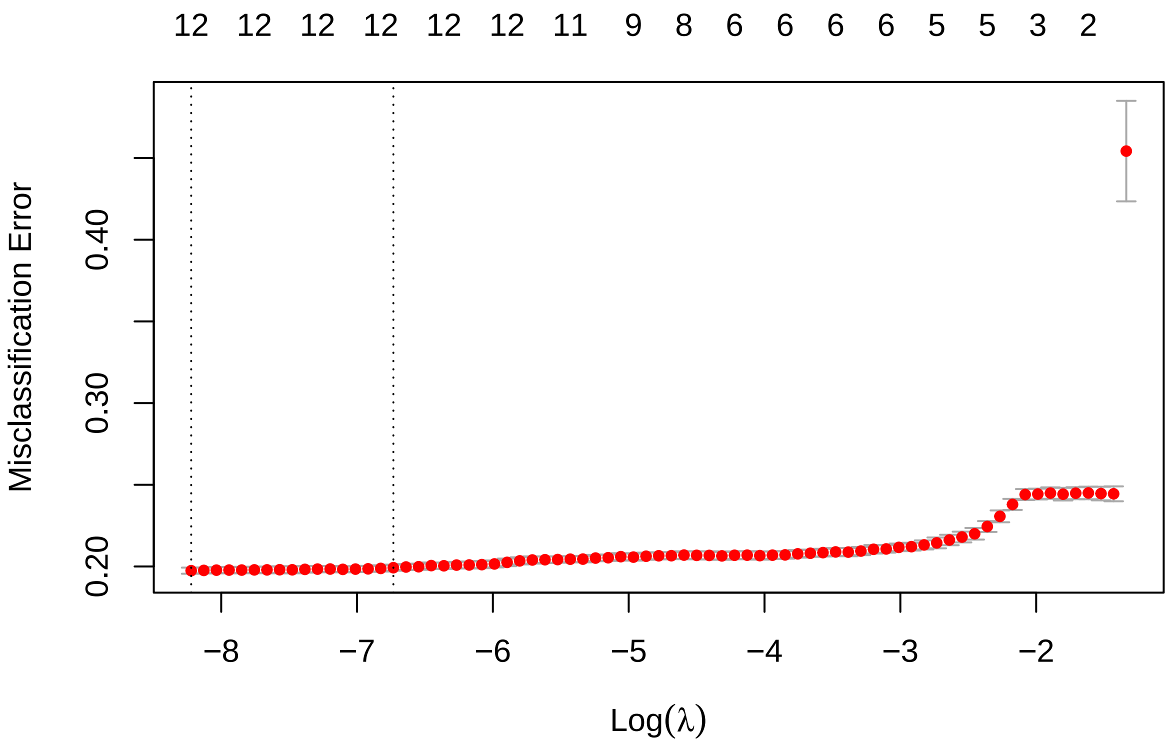
```r
# to select best lambda
lambda_opt_ridge_subset <- ridge_cv_subset$lambda.min
lambda_opt_ridge_subset
```

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

```r
pred_ridge_subset<- predict(ridge_cv_subset, X_test_subset, type = "class", s = lambda_opt_ridge_subset
table(y_test_subset, pred_ridge_subset)
```

```
##              pred_ridge_subset
## y_test_subset    0    1
##             0 2480  653
##             1  637 2444
```

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

```
##              pred_lasso_subset
## y_test_subset    0    1
##             0 2530  603
##             1  651 2430
```

##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")
#print(g1)
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
```

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

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

```r
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")
#print(g2)
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
```

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

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

```r
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")
#print(g3)
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
```

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

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

```r
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")
#print(g4)
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")
#print(g5)
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")
#print(g6)
chisq.out.test(rain_subset_train$Humidity3pm) #p-value = 0.009785, indices:3782 10189 10959 15362 16105
```

```
##
##  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
```

```
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")
#print(g7)
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
```

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

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

```r
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")
#print(g8)
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
```

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

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

```r
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")
#print(g9)
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
```

```r
which(rain_subset_train$Cloud9am ==0) # we got a p-value of 0.07 so we cannot refute the null hypothesi
```

```
##    [1]   22   60  108  154  160  183  197  201  212  227  238  292
##   [13]  319  333  359  366  391  406  411  412  425  439  440  450
##   [25]  498  501  503  514  524  530  534  550  555  569  577  618
##   [37]  654  685  687  704  707  708  721  725  736  747  748  753
##   [49]  769  771  774  787  805  842  866  890  900  919  922  937
##   [61]  953  979  997 1022 1056 1057 1060 1066 1110 1117 1125 1131
##   [73] 1151 1184 1216 1221 1231 1233 1242 1260 1261 1264 1283 1298
##   [85] 1309 1327 1337 1360 1406 1407 1422 1435 1444 1458 1477 1497
##   [97] 1533 1535 1554 1556 1577 1608 1617 1636 1653 1654 1667 1700
##  [109] 1708 1716 1737 1781 1816 1870 1880 1883 1900 1918 1923 1972
##  [121] 1991 2001 2004 2009 2022 2033 2044 2085 2086 2112 2117 2166
##  [133] 2225 2252 2273 2278 2292 2293 2296 2299 2314 2321 2334 2340
##  [145] 2343 2346 2382 2394 2403 2416 2425 2429 2442 2470 2476 2495
##  [157] 2498 2520 2528 2536 2552 2573 2587 2611 2636 2638 2644 2659
##  [169] 2661 2691 2697 2716 2723 2729 2738 2773 2792 2794 2798 2815
##  [181] 2825 2827 2869 2871 2878 2911 2922 2938 3020 3022 3030 3033
##  [193] 3040 3051 3080 3096 3127 3129 3143 3148 3155 3183 3204 3221
##   [205]  3224  3238  3247  3286  3309  3355  3359  3376  3377  3390  3454  3456
##   [217]  3470  3495  3564  3602  3603  3612  3626  3653  3655  3673  3687  3688
##   [229]  3691  3698  3723  3727  3765  3782  3811  3816  3838  3850  3861  3862
##   [241]  3873  3888  3913  3940  3959  3964  3977  3985  3990  4007  4016  4019
##   [253]  4026  4034  4044  4048  4058  4123  4194  4225  4230  4286  4305  4312
##   [265]  4342  4345  4352  4368  4371  4396  4402  4426  4477  4486  4492  4499
##   [277]  4576  4577  4586  4596  4607  4630  4674  4679  4689  4693  4716  4724
##   [289]  4742  4744  4789  4802  4840  4858  4859  4870  4873  4892  4893  4897
##   [301]  4902  4919  4921  4923  4926  4932  4933  4938  4943  4972  4983  5003
##   [313]  5018  5055  5057  5082  5096  5101  5119  5120  5138  5139  5146  5168
##   [325]  5192  5199  5231  5262  5263  5265  5267  5292  5324  5331  5333  5346
##   [337]  5397  5413  5494  5496  5538  5556  5606  5617  5626  5636  5650  5664
##   [349]  5665  5686  5690  5692  5693  5701  5705  5721  5739  5769  5784  5808
##   [361]  5812  5824  5841  5848  5849  5852  5860  5866  5901  5904  5914  5916
##   [373]  5925  5940  5946  5966  5982  6024  6031  6034  6041  6060  6068  6086
##   [385]  6118  6160  6163  6171  6186  6208  6227  6239  6250  6253  6264  6269
##   [397]  6273  6300  6301  6333  6334  6370  6374  6387  6421  6424  6434  6439
##   [409]  6479  6524  6544  6546  6575  6580  6600  6603  6663  6667  6705  6707
##   [421]  6722  6723  6754  6796  6858  6894  6899  6903  6926  6954  6965  6975
##   [433]  7040  7042  7082  7086  7099  7111  7141  7180  7199  7202  7221  7223
##   [445]  7224  7251  7258  7259  7299  7313  7323  7351  7354  7371  7376  7406
##   [457]  7407  7438  7446  7451  7489  7500  7503  7505  7510  7558  7574  7577
##   [469]  7585  7589  7594  7596  7606  7611  7614  7615  7632  7640  7648  7652
##   [481]  7661  7685  7701  7707  7714  7739  7752  7757  7761  7763  7774  7787
##   [493]  7802  7803  7829  7864  7900  7914  7920  7926  7954  7961  7962  7969
##   [505]  7978  7991  7993  8004  8011  8039  8044  8047  8048  8066  8082  8113
##   [517]  8119  8144  8148  8149  8164  8167  8174  8179  8187  8198  8211  8234
##   [529]  8244  8258  8261  8264  8274  8291  8313  8333  8336  8340  8352  8359
##   [541]  8365  8379  8536  8578  8583  8592  8601  8618  8654  8680  8699  8706
##   [553]  8737  8742  8757  8773  8776  8812  8832  8850  8855  8866  8870  8882
##   [565]  8928  8992  9009  9023  9024  9031  9052  9084  9085  9111  9178  9187
##   [577]  9190  9209  9213  9221  9228  9248  9260  9292  9293  9297  9313  9332
##   [589]  9343  9365  9381  9421  9433  9446  9459  9462  9476  9482  9493  9500
##   [601]  9528  9576  9582  9611  9618  9628  9653  9680  9708  9712  9717  9723
##   [613]  9731  9732  9735  9742  9744  9795  9852  9854  9858  9861  9879  9911
##   [625]  9928  9942  9972  9974 10037 10046 10055 10057 10060 10067 10082 10088
##   [637] 10118 10128 10162 10174 10179 10203 10208 10212 10213 10218 10226 10238
##   [649] 10247 10293 10312 10333 10351 10352 10360 10377 10393 10413 10420 10467
##   [661] 10500 10521 10532 10551 10555 10556 10572 10578 10582 10586 10592 10626
##   [673] 10641 10650 10656 10683 10698 10702 10718 10726 10737 10749 10785 10793
##   [685] 10796 10830 10831 10837 10853 10889 10904 10951 10955 10956 10958 11021
##   [697] 11024 11045 11048 11115 11120 11121 11123 11142 11156 11174 11182 11235
##   [709] 11251 11322 11333 11337 11345 11354 11369 11411 11417 11423 11444 11474
##   [721] 11475 11492 11506 11584 11599 11638 11648 11671 11678 11679 11684 11685
##   [733] 11703 11706 11712 11725 11730 11757 11759 11770 11774 11776 11777 11783
##   [745] 11796 11808 11821 11844 11886 11900 11938 11992 11997 12007 12008 12024
##   [757] 12027 12039 12067 12069 12081 12083 12123 12133 12147 12178 12182 12235
##   [769] 12240 12259 12270 12314 12343 12351 12375 12377 12378 12381 12407 12426
##   [781] 12430 12464 12469 12483 12514 12524 12534 12542 12561 12594 12617 12625
##   [793] 12649 12680 12722 12736 12739 12744 12758 12760 12767 12772 12774 12775
##   [805] 12784 12818 12829 12862 12881 12962 12989 13003 13027 13034 13051 13066
##   [817] 13076 13081 13099 13103 13116 13122 13134 13150 13160 13169 13170 13196
##   [829] 13209 13211 13227 13228 13250 13256 13272 13349 13373 13380 13383 13385
##   [841] 13386 13411 13413 13417 13432 13435 13437 13439 13452 13484 13491 13535
##  [853] 13563 13571 13584 13596 13608 13613 13638 13647 13668 13671 13752 13761
##  [865] 13774 13801 13810 13816 13848 13867 13870 13875 13879 13894 13901 13933
##  [877] 13936 13944 13962 13963 13966 13988 13989 13991 13993 14024 14026 14033
##  [889] 14046 14059 14099 14104 14150 14164 14167 14175 14180 14194 14212 14215
##  [901] 14217 14219 14231 14272 14279 14282 14293 14296 14339 14343 14359 14372
##  [913] 14374 14411 14417 14440 14449 14456 14457 14466 14478 14482 14487 14493
##  [925] 14523 14539 14561 14581 14583 14592 14710 14735 14747 14756 14776 14784
##  [937] 14801 14809 14814 14816 14840 14875 14878 14889 14896 14898 14899 14903
##  [949] 14916 14938 14995 15001 15049 15095 15097 15102 15166 15174 15186 15191
##  [961] 15216 15233 15271 15275 15281 15328 15337 15418 15426 15427 15432 15437
##  [973] 15448 15451 15460 15463 15464 15500 15527 15530 15607 15618 15628 15659
##  [985] 15663 15692 15709 15715 15737 15742 15788 15791 15795 15801 15802 15843
##  [997] 15846 15848 15849 15850 15857 15859 15883 15900 15923 15931 15964 15969
## [1009] 15994 16058 16060 16064 16073 16092 16095 16111 16133 16138 16181 16193
## [1021] 16196 16202 16205 16220 16229 16255 16271 16280 16282 16304 16309 16310
## [1033] 16319 16408 16437 16479 16494 16560 16576 16577 16588 16605 16625 16639
## [1045] 16663 16685 16718 16735 16773 16782 16799 16803 16821 16826 16837 16858
## [1057] 16870 16885 16892 16895 16901 16902 16924 16944 16949 16951 16977 16988
## [1069] 17015 17016 17025 17029 17052 17082 17084 17115 17141 17155 17164 17171
## [1081] 17190 17208 17213 17224 17251 17260 17283 17287 17290 17293 17335 17345
## [1093] 17360 17368 17401 17417 17435 17439 17471 17475 17505 17507 17508 17517
## [1105] 17518 17551 17572 17587 17590 17598 17609 17617 17634 17636 17661 17662
## [1117] 17666 17667 17677 17690 17701 17727 17775 17776 17785 17798 17801 17809
## [1129] 17824 17862 17879 17899 17945 17959 17965 17971 17992 18010 18038 18067
## [1141] 18083 18084 18090 18105 18112 18119 18159 18160 18221 18240 18245 18274
## [1153] 18288 18311 18312 18329 18359 18368 18392 18394 18449 18451 18469 18488
## [1165] 18499 18571 18589 18622 18631 18632
```

```
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")
#print(g10)
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
```

```
which(rain_subset_train$Cloud3pm ==0)
```

```
##    [1]   108   154   160   201   227   238   292   304   313   319   391   411
##   [13]   412   450   503   528   530   550   555   569   575   584   644   654
##   [25]   657   687   704   708   712   721   748   753   774   805   919   979
##   [37]   995   997  1022  1057  1066  1091  1095  1184  1191  1221  1264  1266
##   [49]  1283  1290  1298  1444  1445  1458  1497  1504  1533  1535  1617  1636
##   [61]  1643  1653  1667  1708  1716  1737  1781  1797  1865  1869  1883  1917
##   [73]  1918  1974  2001  2085  2112  2171  2224  2225  2252  2296  2321  2333
##   [85]  2343  2346  2394  2401  2416  2428  2453  2464  2475  2476  2498  2520
##   [97]  2536  2552  2587  2625  2661  2691  2697  2729  2738  2773  2780  2794
##  [109]  2811  2869  2871  2875  2878  3020  3022  3030  3033  3096  3127  3143
## [121]  3204  3245  3247  3252  3345  3355  3384  3386  3390  3424  3454  3590
## [133]  3602  3603  3637  3660  3673  3687  3688  3698  3718  3723  3727  3782
## [145]  3811  3816  3850  3854  3862  3888  3892  3942  3985  3986  4007  4021
## [157]  4026  4034  4046  4058  4123  4136  4155  4184  4198  4226  4283  4308
## [169]  4323  4371  4426  4442  4458  4477  4486  4492  4499  4510  4524  4538
## [181]  4577  4591  4633  4667  4674  4689  4716  4744  4748  4763  4766  4772
## [193]  4776  4859  4868  4873  4892  4897  4923  4932  4933  4970  4972  4986
## [205]  5003  5010  5055  5057  5082  5120  5146  5197  5213  5262  5263  5267
## [217]  5346  5367  5397  5398  5407  5412  5413  5424  5428  5496  5538  5595
## [229]  5606  5693  5723  5769  5783  5786  5824  5866  5870  5901  5925  5978
## [241]  5982  6031  6034  6041  6068  6086  6154  6171  6186  6206  6239  6253
## [253]  6319  6332  6334  6362  6374  6421  6424  6434  6439  6490  6544  6559
## [265]  6575  6600  6663  6722  6731  6754  6831  6894  6899  6954  6965  6975
## [277]  6988  7042  7082  7091  7141  7202  7227  7251  7299  7313  7323  7351
## [289]  7362  7371  7378  7406  7407  7433  7465  7489  7500  7512  7580  7594
## [301]  7606  7611  7614  7615  7637  7652  7661  7680  7685  7701  7706  7707
## [313]  7757  7763  7774  7793  7801  7829  7864  7954  7961  7969  7978  8011
## [325]  8038  8044  8047  8063  8069  8082  8120  8144  8164  8174  8192  8194
## [337]  8222  8229  8234  8264  8274  8340  8365  8379  8395  8575  8578  8652
## [349]  8654  8711  8735  8866  8870  8872  8874  8882  8916  8928  8975  9023
## [361]  9024  9084  9111  9154  9187  9190  9209  9213  9221  9228  9260  9293
## [373]  9297  9381  9412  9416  9421  9433  9500  9528  9535  9536  9582  9658
## [385]  9708  9712  9742  9762  9852  9861  9885  9903  9911  9928  9942  9974
## [397]  9987 10018 10025 10028 10055 10088 10118 10154 10174 10203 10212 10226
## [409] 10247 10293 10294 10351 10360 10387 10393 10459 10481 10500 10521 10532
## [421] 10556 10572 10578 10586 10590 10592 10595 10641 10702 10726 10737 10785
## [433] 10796 10863 10904 10951 10953 10958 10979 11021 11045 11048 11063 11066
## [445] 11123 11128 11142 11174 11185 11235 11251 11311 11353 11354 11360 11369
## [457] 11428 11433 11438 11475 11488 11506 11584 11648 11678 11679 11685 11706
## [469] 11712 11757 11774 11776 11783 11796 11827 11887 11893 11900 11905 11997
## [481] 12008 12081 12123 12135 12147 12240 12270 12339 12343 12351 12381 12407
## [493] 12456 12483 12514 12529 12534 12542 12617 12627 12633 12649 12683 12716
## [505] 12751 12752 12758 12760 12772 12774 12775 12862 12884 12989 13003 13027
## [517] 13092 13099 13111 13179 13196 13209 13262 13286 13301 13316 13349 13351
## [529] 13376 13386 13410 13417 13452 13473 13515 13535 13558 13567 13571 13584
## [541] 13613 13662 13668 13687 13752 13761 13774 13793 13801 13810 13815 13870
## [553] 13944 13972 13984 13989 14062 14096 14136 14150 14175 14194 14212 14277
## [565] 14296 14417 14456 14459 14466 14482 14540 14583 14592 14608 14747 14756
## [577] 14775 14781 14784 14790 14801 14814 14875 14889 14899 14903 14916 14966
## [589] 14978 14989 14995 15035 15037 15049 15095 15097 15102 15172 15174 15218
## [601] 15271 15275 15328 15337 15418 15432 15437 15448 15473 15479 15486 15522
## [613] 15537 15538 15556 15607 15691 15709 15715 15742 15745 15780 15846 15848
## [625] 15850 15890 15891 15965 15994 16052 16060 16091 16124 16133 16432 16576
## [637] 16577 16588 16605 16639 16659 16663 16754 16756 16773 16799 16803 16837
## [649] 16855 16870 16879 16885 16892 16895 16924 16949 16988 17001 17052 17115
## [661] 17150 17171 17213 17224 17251 17256 17259 17287 17293 17310 17320 17335
## [673] 17345 17368 17383 17401 17402 17417 17564 17572 17587 17590 17617 17618
## [685] 17634 17641 17661 17662 17718 17727 17809 17862 17899 18046 18052 18067
## [697] 18083 18090 18159 18221 18232 18240 18311 18329 18359 18361 18368 18392
## [709] 18394 18420 18469 18488 18589 18591 18622 18626 18632
```

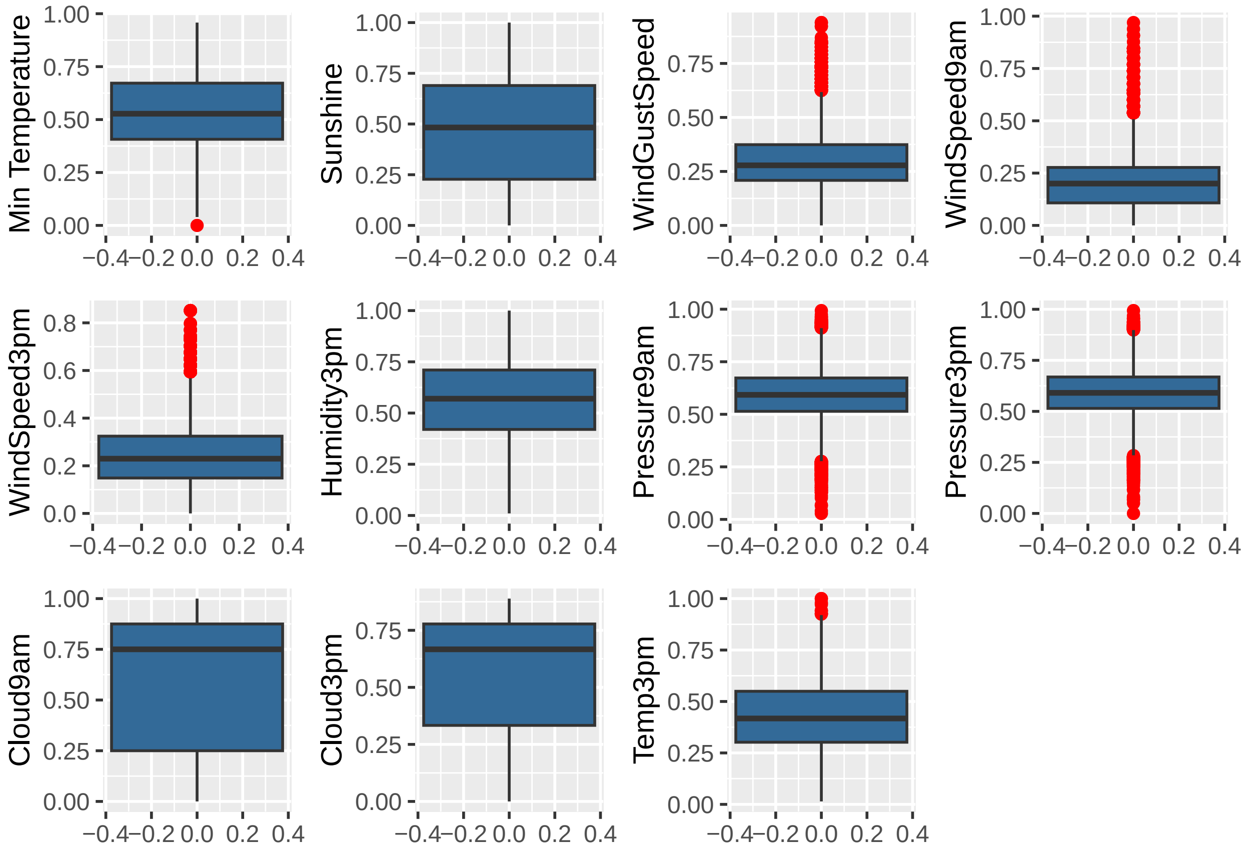
```r
g11<- ggplot(data = rain_subset_train, aes(y = Temp3pm,fill = 2)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 16,
outlier.size = 2)+
theme(legend.position="none") +
ylab("Temp3pm")
#print(g11)
chisq.out.test(rain_subset_train$Temp3pm)#p-value = 0.0003997, indices= 4596, 10679
```

```
##
##  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
```

```
which(rain_subset_train$Temp3pm ==1)
```

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

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



```
#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
```

##LDA

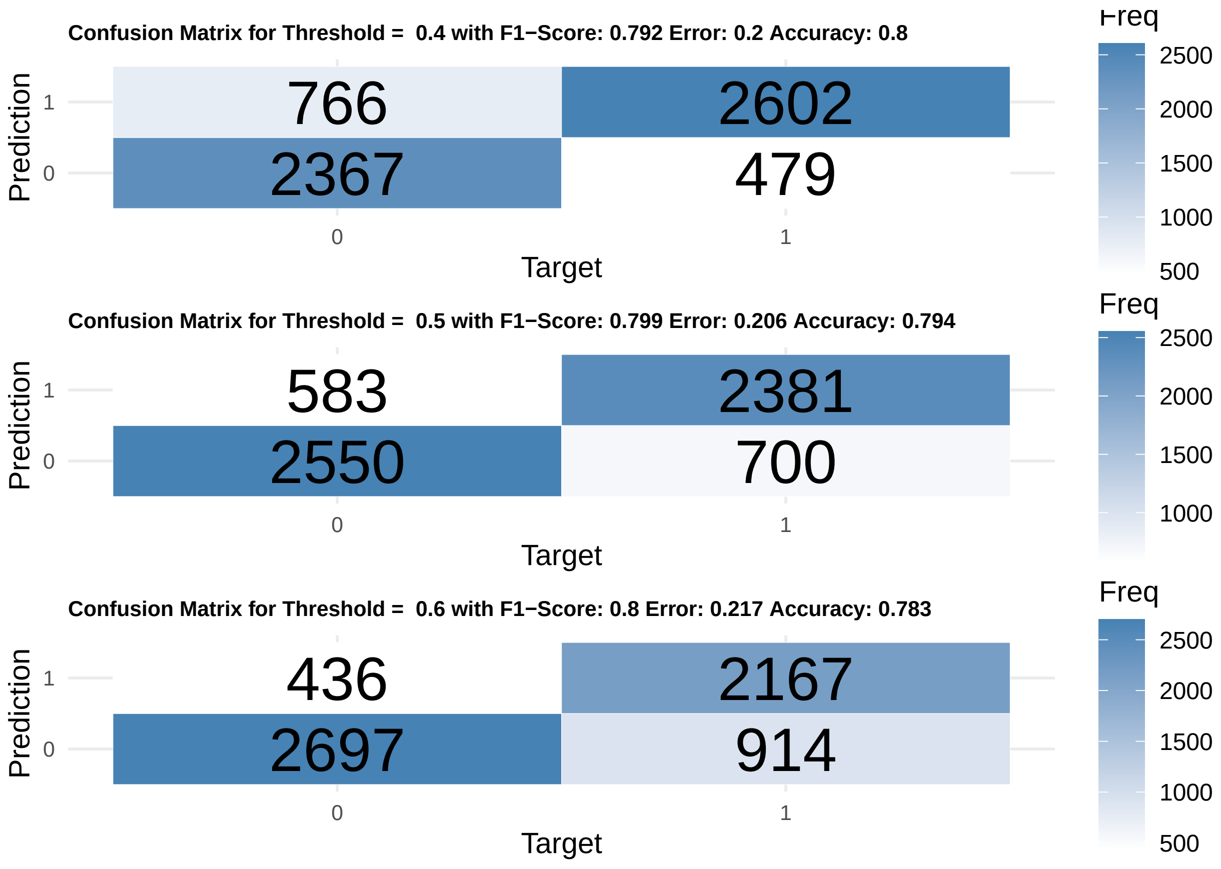
```
# Model definition starting from the previous glm_bal model:

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

```
## Call:
## lda(RainTomorrow ~ ., data = rain_subset_train_NoOutliers, family = "binomial")
```

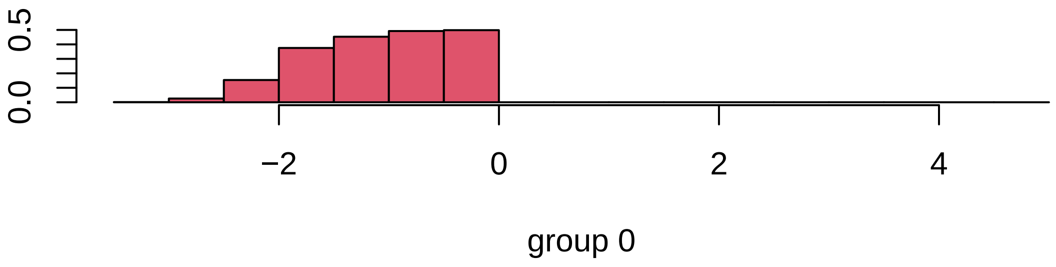
```
##
## Prior probabilities of groups:
##         0         1
## 0.4986052 0.5013948
##
## Group means:
##     MinTemp  Sunshine WindGustSpeed WindSpeed9am WindSpeed3pm Humidity3pm
## 0 0.5198474 0.5981404     0.2639337    0.2044065    0.2354919   0.4472972
## 1 0.5566944 0.3136079     0.3289740    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
```

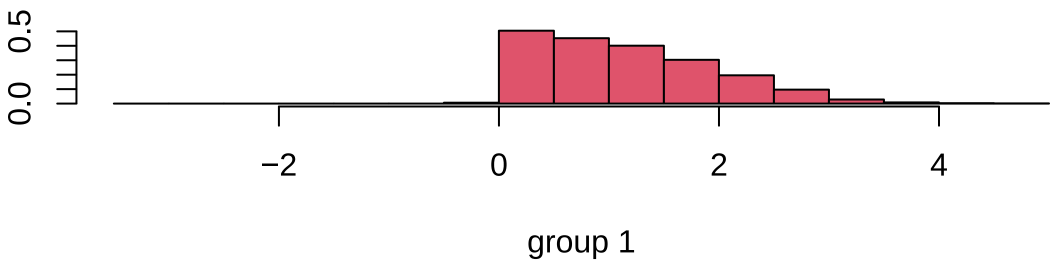
```
pred_lda<- predict(lda, rain_subset_test, type = "response")

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$RainTomorro

# 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$RainTomorro

# 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$RainTomorro
```

```r
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)
```



```r
# We use now the information given by:
# - x: linear combination of the variables that better describe the examples
# - class: assigned class

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

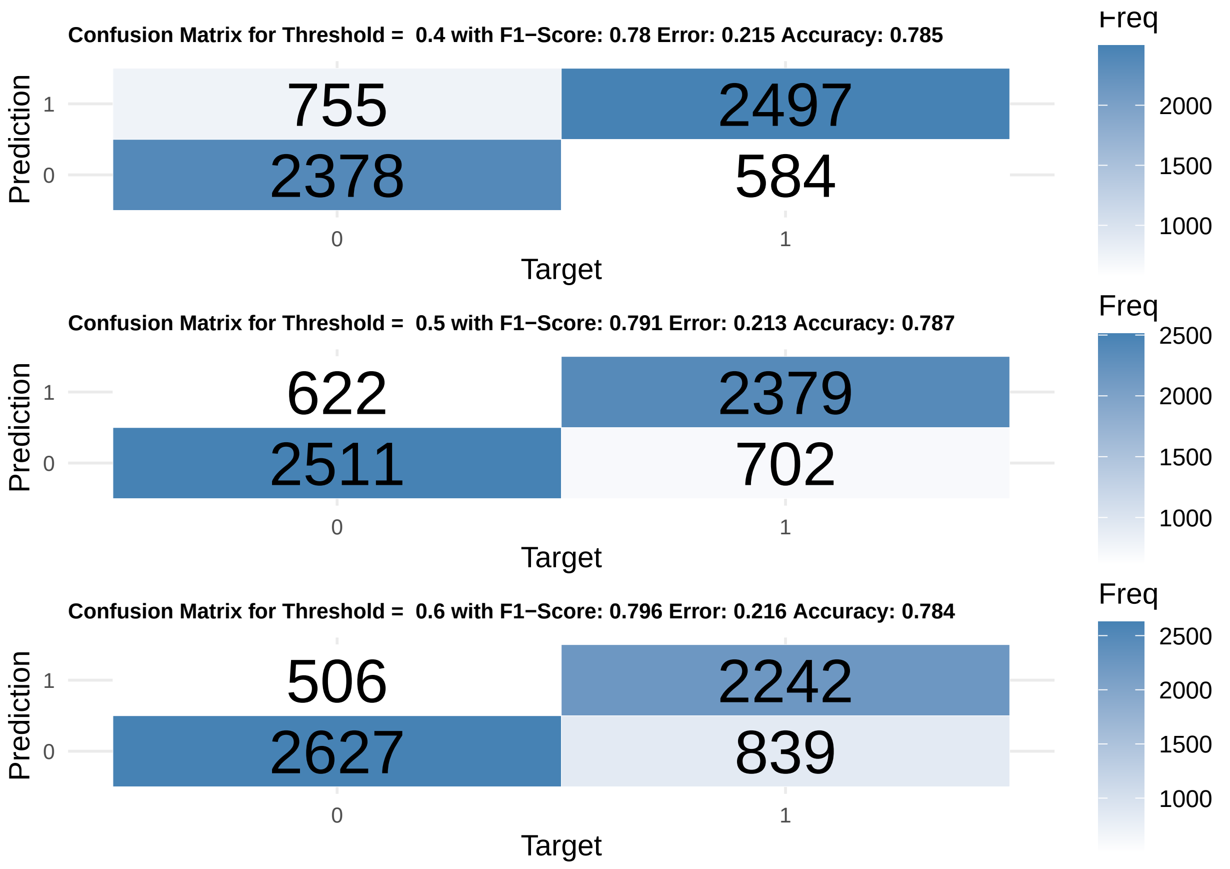




```
qda<- qda(data = rain_subset_train_NoOutliers,RainTomorrow ~.,family = "binomial")
qda
```

```
## 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 matrix with threshold = 0.4
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$RainTomorro
# Confusion matrix with threshold = 0.5
error_qda5 <- mean(pred_qda_05!=rain_subset_test$RainTomorrow)
accuracy_qda5 <- mean(pred_qda_05==rain_subset_test$RainTomorrow)
qda_CM05 <- confusionMatrix(data = factor(pred_qda_05), reference = factor(rain_subset_test$RainTomorro

# 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$RainTomorro

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)
```



```
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
```

```r
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)
k_min
```

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

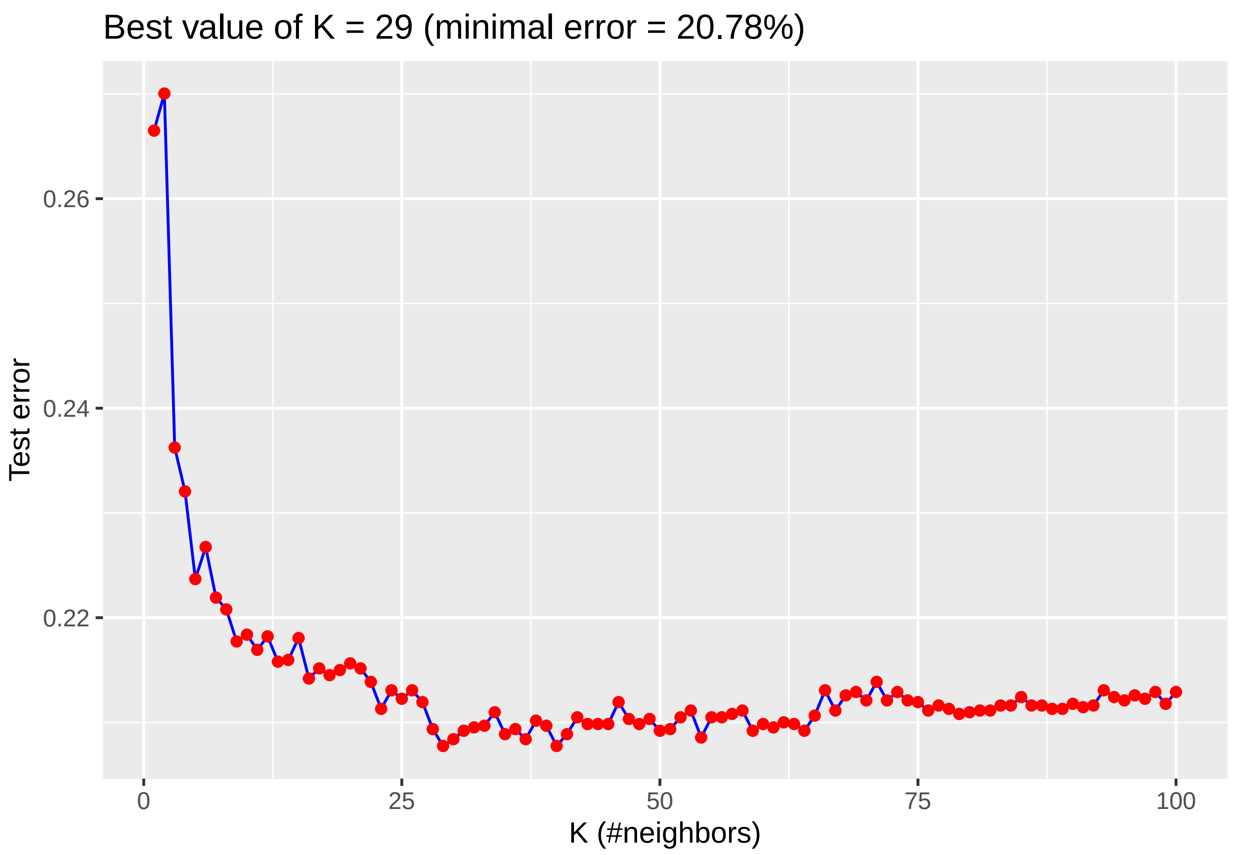
```r
# We compute now the prediction with the value of k that gives us the minimum error
knn<- knn(X_train_subset, X_test_subset,cl = y_train_subset, k = k_min)

knn_pred_min <- knn

# Confusion matrix for KNN on the test set
tab<- table(y_test_subset, knn)
tab
```

```
##               knn
## y_test_subset    0    1
##              0 2476  657
##              1  632 2449
```

```r
accuracy <-function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(tab)
```

```
## [1] 79.25652
```

```r
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),
                     "%)"))
```



#TODO: Analysis, Clean Visualizations and Code

Analysis