Machine Learning Work for Final Project

Taylor Prince, Emily Pham, and Chloe Kim

Predictive Modeling for Wildfire Occurrence

Cleaning the Data Set:

```r
# install necessary packages
library(tidyverse)
library(dplyr)
library(ggplot2)
```

```r
# read in the CSV file and convert to tibble
dataset <- read.csv("forestfires_off.csv") %>%
  as_tibble()


# select relevant variables
dataset <- dataset %>%
  select(X, Y, month, FFMC, DMC, DC, FSS, temp, RH, wind, rain, area)

# convert 'month' column to numeric
month_names <- c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug",
                 "sep", "oct", "nov", "dec")
dataset$month <- match(dataset$month, month_names)

montesinho <- dataset

# replace 'area' variable with a binary variable called 'fire' that represents whether
# or not a wildfire occurred
dataset <- dataset %>%
  mutate(fire = ifelse(area == 0, 0, 1))
# (fire=0 if the burned area of the forest recorded is 0 hectares, else fire = 1)
dataset <- dataset %>%
  select(X, Y, month, FFMC, DMC, DC, FSS, temp, RH, wind, rain, fire)
```

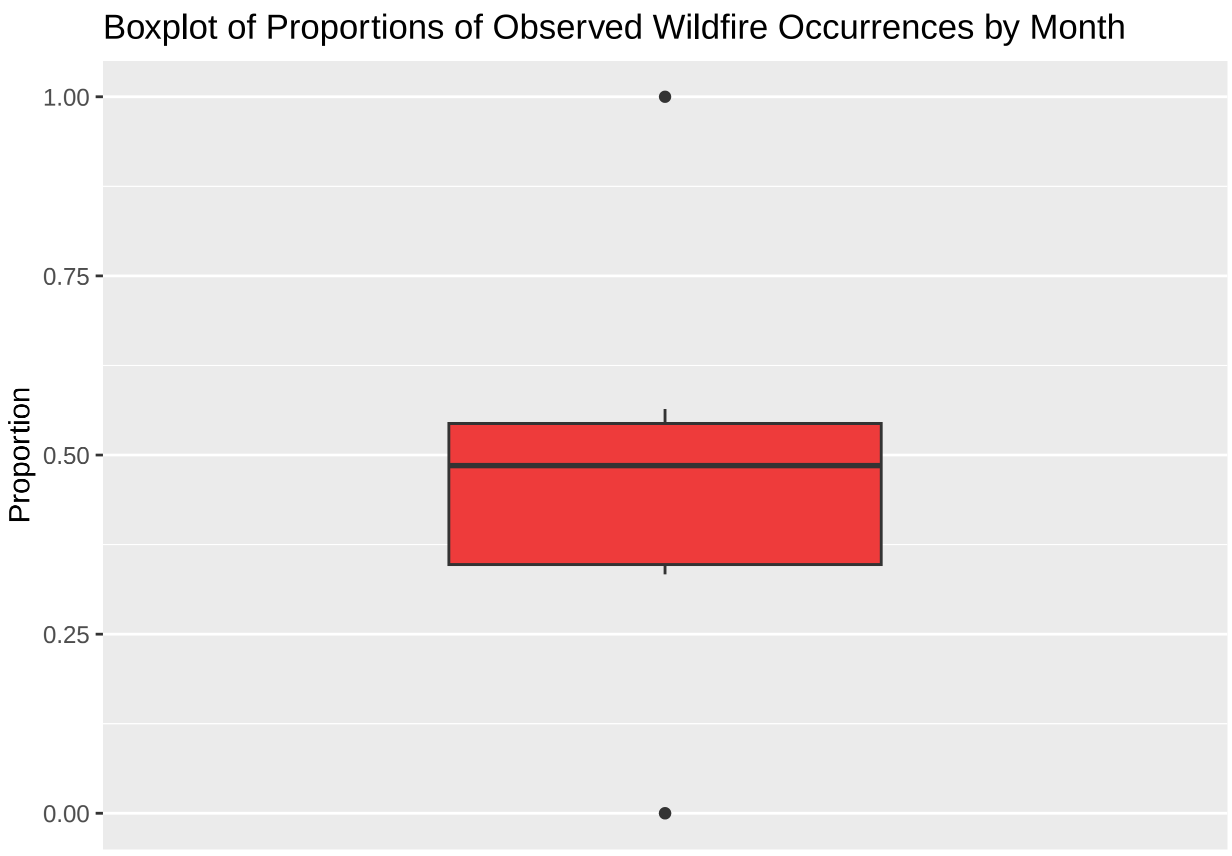
```r
# display a frequency distribution of each month in the dataset
table(dataset$month)
```

```

  1   2   3   4   5   6   7   8   9  10  11  12
  2  20  54   9   2  17  32 184 172  15   1   9
```

```r
# compile the proportion of observed wildfire occurrences per month in the data set
occurrence_by_month  <- data.frame(month = numeric(0), prop = numeric(0))
for (m in 1:12) {
  month_data <- dataset %>%
    filter(month == m)
  occurrence_by_month <- rbind(occurrence_by_month,
                               data.frame(month = m,
                                          prop = mean(month_data$fire)))
}
# display the results
print(occurrence_by_month)
```

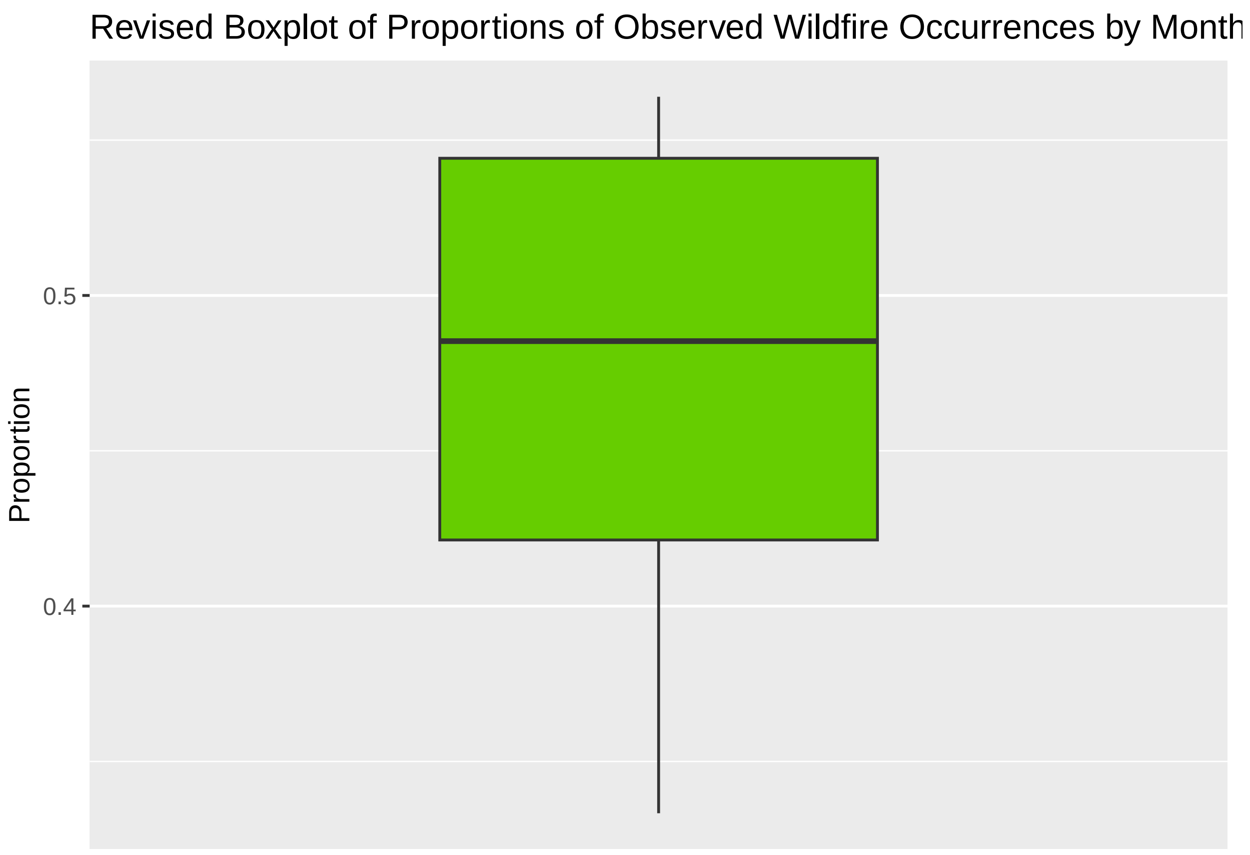
```
##    month      prop
## 1      1 0.0000000
## 2      2 0.5000000
## 3      3 0.3518519
## 4      4 0.4444444
## 5      5 0.5000000
## 6      6 0.4705882
## 7      7 0.5625000
## 8      8 0.5380435
## 9      9 0.5639535
## 10    10 0.3333333
## 11    11 0.0000000
## 12    12 1.0000000
```

```
# display a boxplot of the proportions of observed wildfire occurrences for each month
month_boxplot <- ggplot(data = occurrence_by_month, aes(y = prop)) +
  geom_boxplot(fill = "brown2") +
  scale_x_discrete() +
  labs(title = "Boxplot of Proportions of Observed Wildfire Occurrences by Month",
       y = "Proportion")
print(month_boxplot)
```



Based on the frequency distribution table above, the data sampling for each month was not conducted uniformly/evenly. There is too little data for months 1 (jan), 5 (may), and 11 (nov). Also, the proportion value for month 12 (dec) is a significant outlier. Observations were only recorded/reported when wildfires occurred in that month, contributing to a potential skewness in the data distribution. Therefore, we will exclude months 1 (jan), 5 (may), 11 (nov), and 12 (dec) from the dataset to build a relatively more accurate predictive model.

```r
# remove the months with limited data / outliers (i.e. jan, mar, nov, and dec)
dataset_with_months_removed <- dataset %>%
  filter(month != 1 & month != 5 & month != 11 & month != 12)

montesinho_with_months_removed <- montesinho %>%
  filter(month != 1 & month != 5 & month != 11 & month != 12)

# display a revised boxplot of the proportions of observed wildfire occurrences for
# each month
occurrence_by_month <- occurrence_by_month[-c(1, 5, 11, 12), ]
revised_month_boxplot <- ggplot(data = occurrence_by_month, aes(y = prop)) +
  geom_boxplot(fill = "chartreuse3") +
  scale_x_discrete() +
  labs(title = "Revised Boxplot of Proportions of Observed Wildfire Occurrences by Month",
       y = "Proportion")
print(revised_month_boxplot)
```



```r
# compile the proportion of observed wildfire occurrences per location (i.e. each X and Y
# combination)
x_values <- sort(as.vector(unique(dataset$X)))
y_values <- sort(as.vector(unique(dataset$Y)))
occurrence_by_location <- data.frame(X = numeric(0), Y = numeric(0), prop = numeric(0))
for (x in x_values) {
  for (y in y_values) {
    xy_combination <- dataset %>%
      filter(X == x, Y == y)
    if (nrow(xy_combination) > 0) {
      occurrence_by_location <- rbind(occurrence_by_location,
                                      data.frame(X = x, Y = y,
                                                 prop = mean(xy_combination$fire)))
    }
  }
}
# display the results
occurrence_by_location_table <-  as_tibble(occurrence_by_location) %>%
  pivot_wider(names_from = Y, values_from = prop) %>%
  as.data.frame
print(occurrence_by_location_table[, -1])
```

```
          2         3         4         5         6  8   9
1 0.2105263 0.7000000 0.6666667 1.0000000        NA NA  NA
2 0.4400000 0.0000000 0.6296296 0.7000000        NA NA  NA
3        NA 1.0000000 0.3488372 0.1428571 0.0000000 NA  NA
4        NA 0.5909091 0.5555556 0.4000000 0.5000000 NA  NA
```

```
5        NA        NA 0.5652174 0.0000000 0.5000000 NA  NA
6        NA 0.3600000 0.6666667 0.6530612 0.3333333 NA  NA
7        NA 0.5000000 0.5555556 0.2727273 0.5000000 NA  NA
8        NA 1.0000000 1.0000000 0.7500000 0.5576923  1  NA
9        NA        NA 1.0000000 0.5000000 1.0000000 NA 0.5
```

```
# display a frequency distribution of each location (X and Y combination) in the dataset
table(dataset$X, dataset$Y)
```

```
     2  3  4  5  6  8  9
  1 19 10 15  4  0  0  0
  2 25  1 27 20  0  0  0
  3  0  1 43  7  4  0  0
  4  0 22 36 25  8  0  0
  5  0  0 23  3  4  0  0
  6  0 25  9 49  3  0  0
  7  0  2 45 11  2  0  0
  8  0  3  1  4 52  1  0
  9  0  0  4  2  1  0  6
```

```
# remove the locations (X and Y combinations) with limited data (i.e. less than 3
# observations recorded/reported)
x_and_y_combinations_to_remove <- data.frame(X = numeric(0), Y = numeric(0))
for (x in seq_along(x_values)) {
  for (y in seq_along(y_values)) {
    if (table(dataset$X, dataset$Y)[x, y] < 3) {
      x_and_y_combinations_to_remove <- rbind(x_and_y_combinations_to_remove,
                                              data.frame(X = x_values[x],
                                                         Y = y_values[y]))
      entry <- which(occurrence_by_location$X == x_values[x] &
                       occurrence_by_location$Y == y_values[y])
      if (length(entry) > 0) {
        occurrence_by_location <- occurrence_by_location[-entry, ]
      }
    }
  }
}
# display the first few rows of the dataframe containing the locations with limited data
print(head(x_and_y_combinations_to_remove))
```

```
  X Y
1 1 6
2 1 8
3 1 9
4 2 3
5 2 6
6 2 8
```

```
x_and_y_combinations_to_remove <- as.matrix(x_and_y_combinations_to_remove)
colnames(x_and_y_combinations_to_remove) <- NULL
dataset_with_months_and_locations_removed <- dataset_with_months_removed
for (i in seq_len(nrow(x_and_y_combinations_to_remove))) {
  row <- x_and_y_combinations_to_remove[i, ]
  dataset_with_months_and_locations_removed <-
    dataset_with_months_and_locations_removed %>%
    filter(!(X == row[1] & Y == row[2]))
}

montesinho_with_months_and_locations_removed <- montesinho_with_months_removed
for (i in seq_len(nrow(x_and_y_combinations_to_remove))) {
  row <- x_and_y_combinations_to_remove[i, ]
  montesinho_with_months_and_locations_removed <-
    montesinho_with_months_and_locations_removed %>%
    filter(!(X == row[1] & Y == row[2]))
}
```

Training and Testing Logistic Regression Model:

After cleaning the data set, our objective is to now construct a predictive model that will enable us to estimate the probability of wildfire occurrence in Montesinho Park and other representative areas. Specifically, we will randomly cut the cleaned data into a “training” (80% of data) and “testing” (20% of data) set. We will train a machine learning model on the data from Montesinho Park by fitting the logistic regression model to the training data through R’s built-in *glm*() function. We will repeat this process 20 times to use the trial with the best model in the sense that its predicted proportion of wildfire occurrences is closest to the actual, observed proportion of wildfire occurrences from the training set. Through this, we can derive the coefficients that most accurately represent the relationship between the predictor variables and the binary outcome. Since the predictor variables have very different scales, normalizing the coefficients allows us to determine each variable’s relative importance/effect in estimating the probability of a wildfire occurring. Then, we can understand which factors have the strongest impact on the occurrence of wildfires and, thus, help prioritize mitigation strategies based on our results.

```
cleaned_dataset <- dataset_with_months_and_locations_removed
cleaned_montesinho <- montesinho_with_months_and_locations_removed

set.seed(1) # for reproducibility
results <- list()
num_trials <- 20
for (i in seq_len(num_trials)) {
  # split up the data set randomly
  train_data <- cleaned_dataset %>% sample_frac(0.8) # (training = 80% of data)
  test_data <- cleaned_dataset %>% anti_join(train_data) # (testing = 20% of data)
  results <- c(results, list(list(training = train_data, testing = test_data)))
}
title <- paste0("trial", 1:num_trials)
names(results) <- title
# display an element of a list called 'results' containing the training and
# testing set for each trial
print(head(results, 1))
```

```
$trial1
$trial1$training
# A tibble: 394 x 12
        X     Y month  FFMC   DMC    DC   FSS  temp    RH  wind  rain  fire
    <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1      2     5     9  91.6 108.  764    2.63  19.3    44   2.2     0     1
 2      6     5     3  91.2  48.3  97.8  3.64  14.6    26   9.4     0     1
 3      4     6     2  68.2  21.5  87.2  0     15.4    40   2.7     0     0
 4      5     4     9  90.3 290   855.   2.89  16.2    58   3.6     0     0
 5      6     6     8  96   164   643    3.81  30.8    30   4.9     0     1
 6     6     5     9  92.6 115.   777.   3.14  24.3    27   4.9      0     0
 7     2     5     8  92.1 153.   658.   3.84  20.2    47   4        0     1
 8     4     4     8  95.1 141.   606.   4.15  20.6    58   1.3      0     0
 9     6     4     3  90.8  41.9   89.4  2.98  13.3    42   0.9      0     1
10     1     4     9  92.8 119    784.   2.91  16.8    28   4        0     1
# ... with 384 more rows

$trial1$testing
# A tibble: 95 x 12
       X     Y month  FFMC   DMC    DC   FSS  temp    RH  wind  rain  fire
   <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1     8     6     8  92.3  88.9  496.  3.09  24.1    27   3.1      0     0
 2     8     6     9  91   130.   693.  2.81  13.1    63   5.4      0     0
 3     7     5     9  92.5  88    699.  2.83  22.8    40   4        0     0
 4     7     5     9  92.5  88    699.  2.83  17.8    51   7.2      0     0
 5     7     5     9  92.8  73.2  713   4.50  19.3    38   4        0     0
 6     7     4    10  90    41.5  683.  3.12  11.3    60   5.4      0     0
 7     5     6     9  90.9 126.   686.  2.81  14.7    70   3.6      0     0
 8     6     6     7  94.2  62.3  443.  3.46  23      36   3.1      0     0
 9     4     3     8  91.7 114.   661.  2.66  17.6    45   3.6      0     0
10     4     3    10  92.6  46.5  692.  3.14  13.8    50   2.7      0     0
# ... with 85 more rows
```

```r
log_reg <- function(z) {
  # perform logistic regression on training set
  model = glm(formula = fire ~ X + Y + month + FFMC + DMC + DC + FSS +
                temp + RH + wind + rain, data =  z[[1]], family = binomial)
  predict(model, type = "response")
}
probs <- suppressWarnings(lapply(results, log_reg)) # (probability of wildfire occurrence
# for each observation in the training set of every trial based on fitted logistic
# regression model)

predict_prop_fire <- function(values) {
  # calculate the proportion of probabilities greater than 1/2 (i.e. proportion of
  # predicted wildfire occurrences)
  mean(values > 0.5) # (where 1/2 is the classification threshold such that if the
  # probability of the target 'fire' variable being equal to 1 is greater than 1/2,
  # meaning a wildfire likely occurred, then we declare the predicted class as 1 to
  # represent a wildfire occurrence)
}
pred_prop <- sapply(probs, predict_prop_fire) # (predicted/estimated proportion of
# wildfire occurrences for each trial from training set)

observe_prop_fire <- function(z) {
  # calculate the proportion of observed wildfire occurrences in the training data
  mean(z[[1]]$fire)
}
obs_prop <- sapply(results, observe_prop_fire) # (observed/actual proportion of wildfire
# occurrences for each trial from training set)

# choose the best out of 20 trials
best_trial <- names(which(abs(pred_prop - obs_prop) == min(abs(pred_prop - obs_prop))))
```

```r
# display the summary of the best model
best_model = glm(formula = fire ~ X + Y + month + FFMC + DMC + DC + FSS +
                   temp + RH + wind + rain, data =  results[[best_trial]]$training,
                 family = binomial)
summary(best_model)
```

```

Call:
glm(formula = fire ~ X + Y + month + FFMC + DMC + DC + FSS +
    temp + RH + wind + rain, family = binomial, data = results[[best_trial]]$training)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.69205  -1.12993  -0.00862   1.10707   1.85076

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.107075   3.857117  -0.287   0.7741
X            0.111963   0.055754   2.008   0.0446 *
Y            0.041729   0.108192   0.386   0.6997
month       -0.295549   0.175498  -1.684   0.0922 .
FFMC        -0.011641   0.046782  -0.249   0.8035
DMC         -0.004001   0.002962  -1.351   0.1768
DC           0.003704   0.001742   2.126   0.0335 *
FSS          0.159776   0.262621   0.608   0.5429
temp         0.064097   0.032117   1.996   0.0460 *
RH           0.005099   0.009422   0.541   0.5884
wind         0.032285   0.064090   0.504   0.6144
rain         0.006221   0.349005   0.018   0.9858
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 546.20  on 393  degrees of freedom
Residual deviance: 524.04  on 382  degrees of freedom
AIC: 548.04

Number of Fisher Scoring iterations: 4
```

```r
estimates <- summary(best_model)$coefficients[, "Estimate"][-1] # (coefficients for
# each predictor variable from the best model)

# normalize the coefficients to compare them on an even playing field
normalized_coefficients <- estimates
for (variable in names(estimates)) {
  coefficent <- estimates[variable]
  normalized_coefficients[variable] <- coefficent * sqrt(sum(dataset[variable]^2))
}
sorted_indices <- order(-abs(normalized_coefficients))
# display the normalized coefficients in descending order of magnitude
print(normalized_coefficients[sorted_indices])
```

```
      month            DC         temp         FFMC            X          DMC
```

```
-52.51048511  50.64275084  28.79838553 -24.03687844  13.26374173 -11.64631455
         FSS           RH            Y         wind         rain
 11.23353697   5.47186449   4.24308446   3.22867304   0.04193349
```

(converted to a table in LaTeX for paper)

From these results, we can identify the variables of most and least importance for predicting wildfire occurrence. It appears that the month, Drought Code (DC) index, temperature, and Fine Fuel Moisture Code (FFMC) index are the most influential factors, while rain has the least association with wildfire occurrences. However, it is crucial to note that rainfall is taken into account when calculating the DC index and FFMC index values, which raises the possibility of confounding effects.

Now, we will test how well our predictive model called 'best_model' performs on the testing set to see how accurately it can make predictions on new data.

```r
experimental <- mean(predict(best_model, newdata = results[[best_trial]]$testing,
                             type = "response") > 0.5) # (predicted proportion of
                             # wildfire occurrences in testing set based on model)

actual <- mean(results[[best_trial]]$testing$fire) # (observed proportion wildfire
# occurrences in testing set)

# calculate and display the the percentage error
percent_error <- abs((experimental - actual) / actual) * 100
cat(paste0("The percentage error is ", percent_error, "%."))
```

```
The percentage error is 1.81818181818183%.
```

Since the percentage error of approximately 1.82% is very low, we can conclude that our model is effective in predicting wildfire occurrence.

However, upon statistical analysis, our trained logistic regression model may not be the most suitable for predicting wildfire occurrence. We observed from the model summary that the residual deviance is 524.04, which is quite high. Although it is slightly lower than the null deviance, suggesting it is a better fit than the null model, this value would ideally be as close to 0 as possible. Moreover, the AIC is also high with a value of 548.2, especially considering that it is near the AIC of the null model. One cause for both of these issues is again, confounding.

```r
# calculate confusion matrix and statistics for further evaluation
library(caret)
caret::confusionMatrix(data = factor(as.numeric(unname(predict(
  best_model, newdata = results[[best_trial]]$testing,
  type = "response") > 0.5))),
  reference = factor(results[[best_trial]]$testing$fire))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 14 29
         1 28 26

               Accuracy : 0.4124
                 95% CI : (0.3133, 0.5169)
    No Information Rate : 0.567
    P-Value [Acc > NIR] : 0.9992

                  Kappa : -0.1934
```

```
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.3333
            Specificity : 0.4727
         Pos Pred Value : 0.3256
         Neg Pred Value : 0.4815
             Prevalence : 0.4330
         Detection Rate : 0.1443
   Detection Prevalence : 0.4433
      Balanced Accuracy : 0.4030

       'Positive' Class : 0
```

Comparing Model Predictions to Observations - Temporal and Spatial Analysis:

Temporal Comparison

```r
# create a function that finds the average probability of wildfire occurrence using
# our model for a given data set
predict_avg_prob <- function(df) {
  probabilities <- predict(best_model, newdata = df, type = "response")
  return(mean(probabilities > 0.5))
}

# compute the predicted proportion (i.e. average probability) based on our model and
# observed proportion of wildfire occurrences for each month in the (cleaned) data set
# and compile the values into two dataframes
month_values <- sort(unique(cleaned_dataset$month))
pred_month_data <- data.frame(month = numeric(0), prob = numeric(0))
actual_month_data  <- data.frame(month = numeric(0), prop = numeric(0))
for (m in month_values) {
  month_data <- cleaned_dataset %>%
    filter(month == m) # (data set containing the observations exclusively for each month)
  pred_month_data <- rbind(pred_month_data,
                           data.frame(month = m,
                                      prob = predict_avg_prob(month_data)))
  actual_month_data <- rbind(actual_month_data,
                             data.frame(month = m,
                                        prop = mean(month_data$fire)))
}

# display the two resulting dataframes
print(pred_month_data)
```

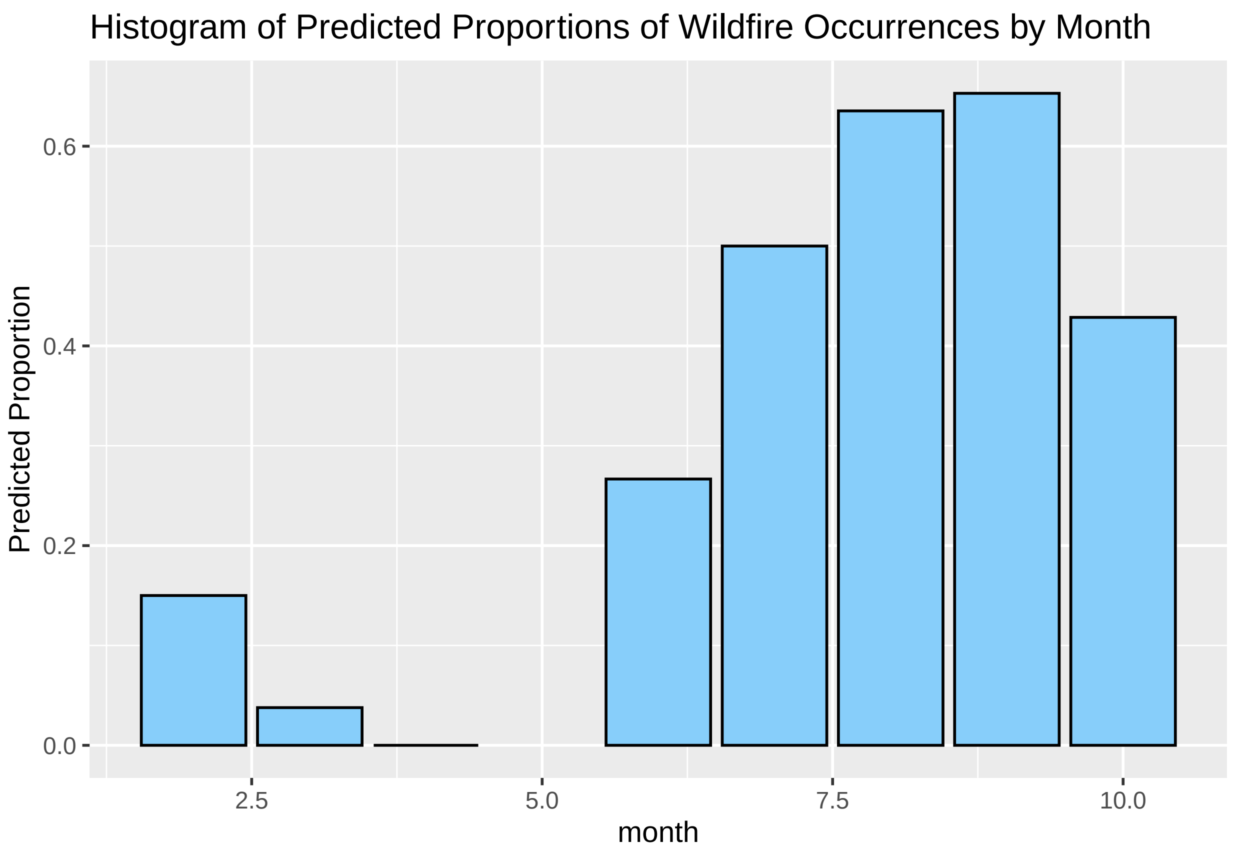
```
  month       prob
1     2 0.15000000
2     3 0.03773585
3     4 0.00000000
4     6 0.26666667
5     7 0.50000000
6     8 0.63535912
7     9 0.65294118
```

```
8    10 0.42857143
```

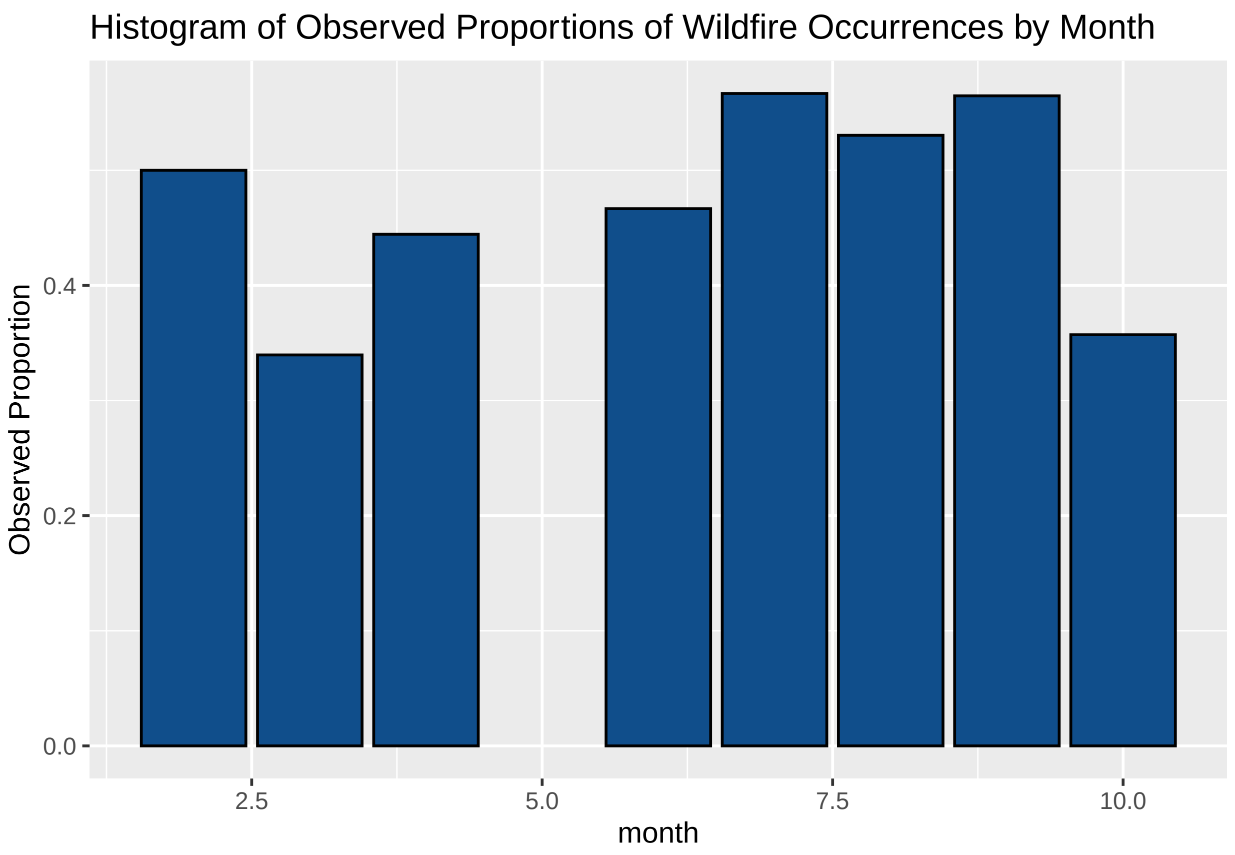
```
print(actual_month_data)
```

```
  month      prop
1     2 0.5000000
2     3 0.3396226
3     4 0.4444444
4     6 0.4666667
5     7 0.5666667
6     8 0.5303867
7     9 0.5647059
8    10 0.3571429
```

```
# display a histogram of the average probabilities of wildfire occurrences for each month
ggplot(pred_month_data, aes(x = month, y = prob)) +
  geom_bar(stat = "identity", color = "black", fill = "lightskyblue") +
  labs(title = "Histogram of Predicted Proportions of Wildfire Occurrences by Month",
       y = "Predicted Proportion")
```



```
# display a histogram of the observed proportions of wildfire occurrences for each month
ggplot(actual_month_data, aes(x = month, y = prop)) +
  geom_bar(stat = "identity", color = "black", fill = "dodgerblue4") +
  labs(title = "Histogram of Observed Proportions of Wildfire Occurrences by Month",
       y = "Observed Proportion")
```



```r
# calculate the correlation coefficient between the predicted and observed
# proportions of wildfire occurrences for each month
month_cor_coef <- cor(pred_month_data$prob, actual_month_data$prop)
cat(paste0("The correlation coefficient is ", month_cor_coef, "."))
```

```
The correlation coefficient is 0.606683481735162.
```

Analyzing the histograms above, it is evident that their data distributions for the last three months (August, September, and October) have a similar shape. However, for the other five months, the predicted proportions differ significantly from the observed proportions in terms of value. Nevertheless, despite this discrepancy, there are general trends/patterns captured by our predictive model that mirror the observed data, such as the proportion of wildfire occurrences being greater in February (month 2) than March and April (months 3 and 4) as well as less in June (month 6) than July (month 7). Moreover, the correlation coefficient of approximately 0.607 indicates a moderately strong, positive relationship between predicted and observed proportions of wildfire occurrences for every month. From this, we can conclude that our model's predictions align well with the observed data regarding the temporal distribution of wildfire occurrences in terms of month.

It is worth noting that the relatively higher accuracy in predicting the proportion of wildfire occurrences for months 8 and 9 in particular could be attributed to having significantly larger sample sizes [**see page 1 for values in the outputted frequency distribution**] by the Central Limit Theorem.

Spatial Comparison

```r
# compute the predicted proportion (i.e. average probability) based on our model and
# observed proportion of wildfire occurrences for each location (X and Y combination)
# in the (cleaned) data set and compile the values into two dataframes
x_values <- sort(as.vector(unique(dataset$X)))
```

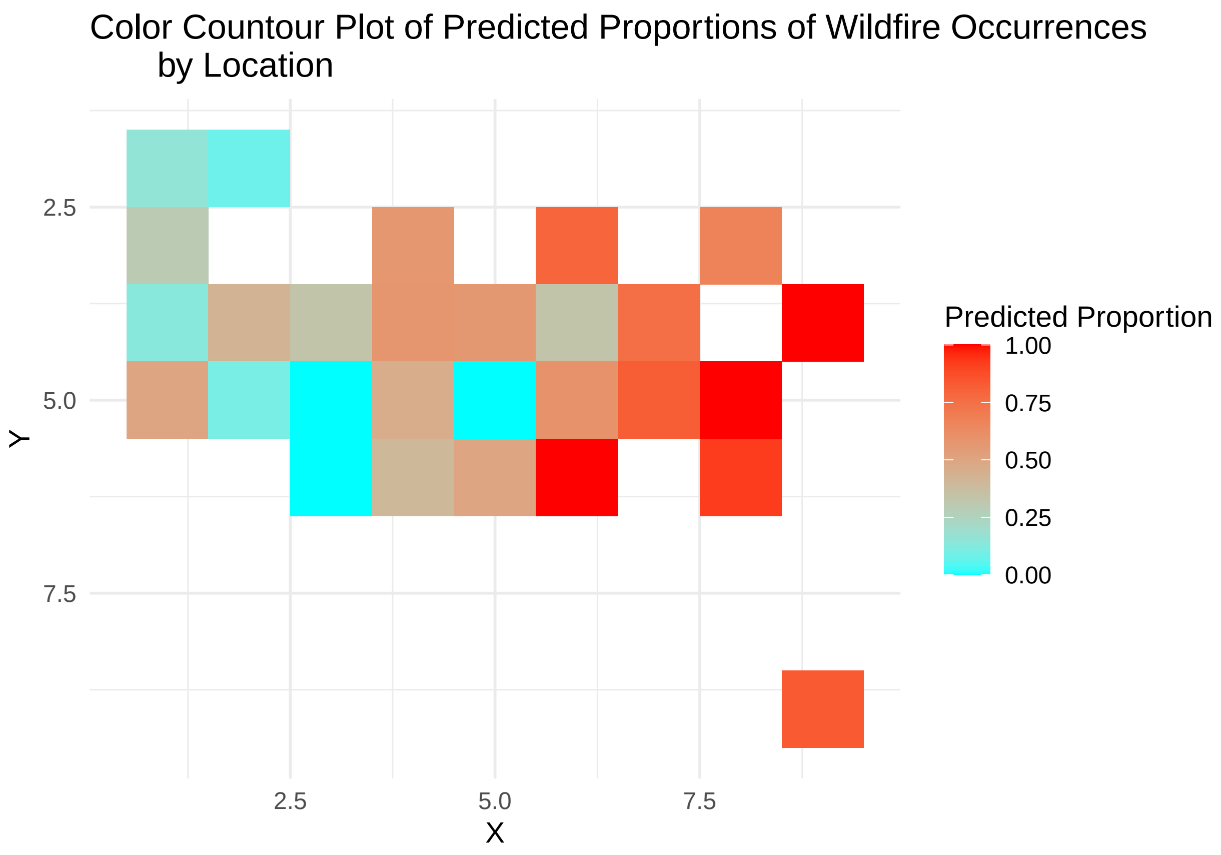
```r
y_values <- sort(as.vector(unique(dataset$Y)))
pred_xy_combinations_data <- data.frame(X = numeric(0), Y = numeric(0),
                                        prob = numeric(0))
actual_xy_combinations_data <- data.frame(X = numeric(0), Y = numeric(0),
                                          prop = numeric(0))
for (x in x_values) {
  for (y in y_values) {
    xy_combination <- cleaned_dataset %>%
      filter(X == x, Y == y)
    if (nrow(xy_combination) > 0) {
      pred_xy_combinations_data <-
        rbind(pred_xy_combinations_data,
              data.frame(X = x, Y = y,
                         prob = predict_avg_prob(xy_combination)))
      actual_xy_combinations_data <- rbind(actual_xy_combinations_data,
                                           data.frame(X = x, Y = y,
                                                      prop = mean(xy_combination$fire)))
    }
  }
}

# display the first few rows of the two resulting dataframes
print(head(pred_xy_combinations_data))
```

```
  X Y      prob
1 1 2 0.1578947
2 1 3 0.3000000
3 1 4 0.1333333
4 1 5 0.5000000
5 2 2 0.0800000
6 2 4 0.4230769
```

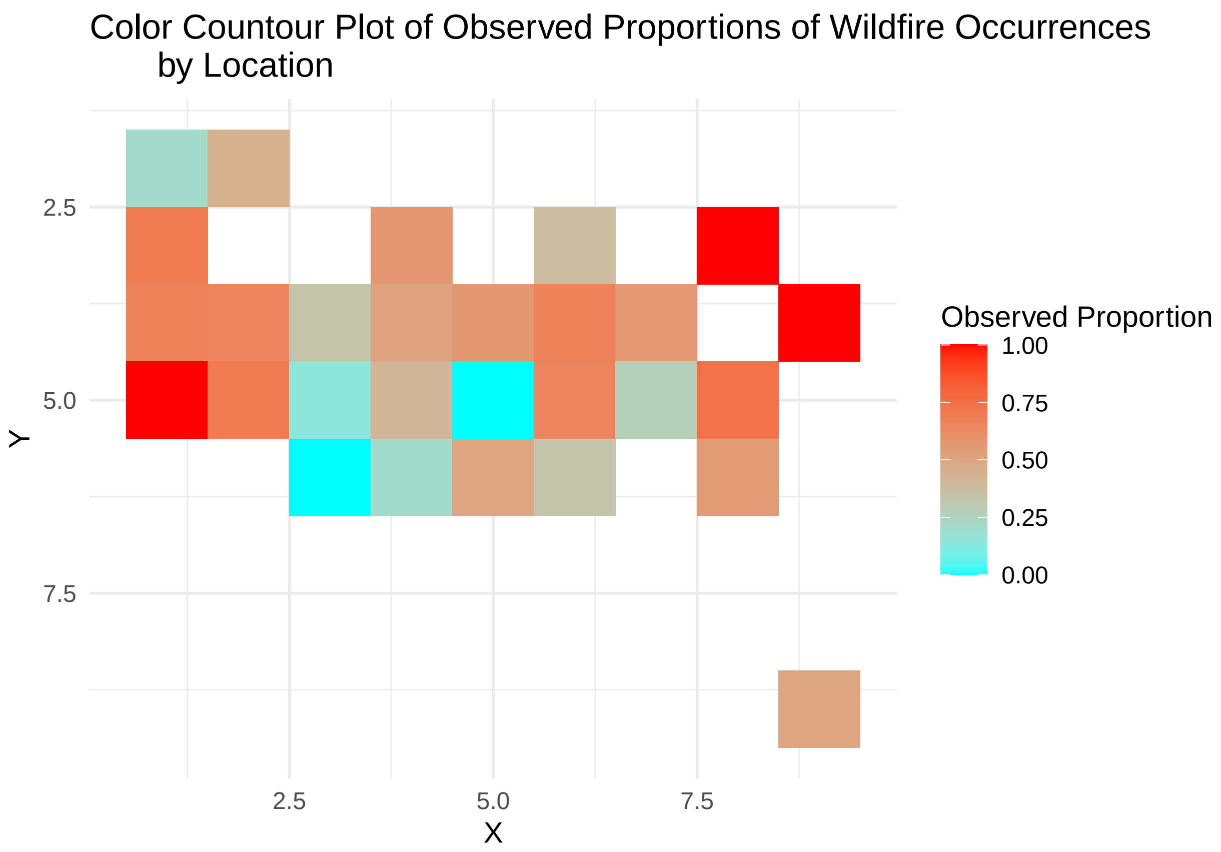
```r
print(head(actual_xy_combinations_data))
```

```
  X Y      prop
1 1 2 0.2105263
2 1 3 0.7000000
3 1 4 0.6666667
4 1 5 1.0000000
5 2 2 0.4400000
6 2 4 0.6538462
```

```r
# display a color contour plot of the average probabilities of wildfire occurrences
# for each location (X and Y combination)
ggplot(pred_xy_combinations_data, aes(X, Y, fill = prob)) +
  geom_tile() +
  scale_fill_gradientn(colours = rev(rainbow(2))) +
  labs(title = "Color Countour Plot of Predicted Proportions of Wildfire Occurrences
     by Location", x = "X", y = "Y", fill = "Predicted Proportion") +
  theme_minimal() +
  scale_y_reverse()
```



```r
# display a color contour plot of the observed proportions of wildfire occurrences
# for each location (X and Y combination)
ggplot(actual_xy_combinations_data, aes(X, Y, fill = prop)) +
  geom_tile() +
  scale_fill_gradientn(colours = rev(rainbow(2))) +
  labs(title = "Color Countour Plot of Observed Proportions of Wildfire Occurrences
      by Location", x = "X", y = "Y", fill = "Observed Proportion") +
  theme_minimal()  +
  scale_y_reverse()
```



```r
# calculate the correlation coefficient between the predicted and observed
# proportions of wildfire occurrences for each location (X and Y combination)
location_cor_coef <- cor(pred_xy_combinations_data$prob,
                         actual_xy_combinations_data$prop)
cat(paste0("The correlation coefficient is ", location_cor_coef, "."))
```

```
The correlation coefficient is 0.417181582502191.
```

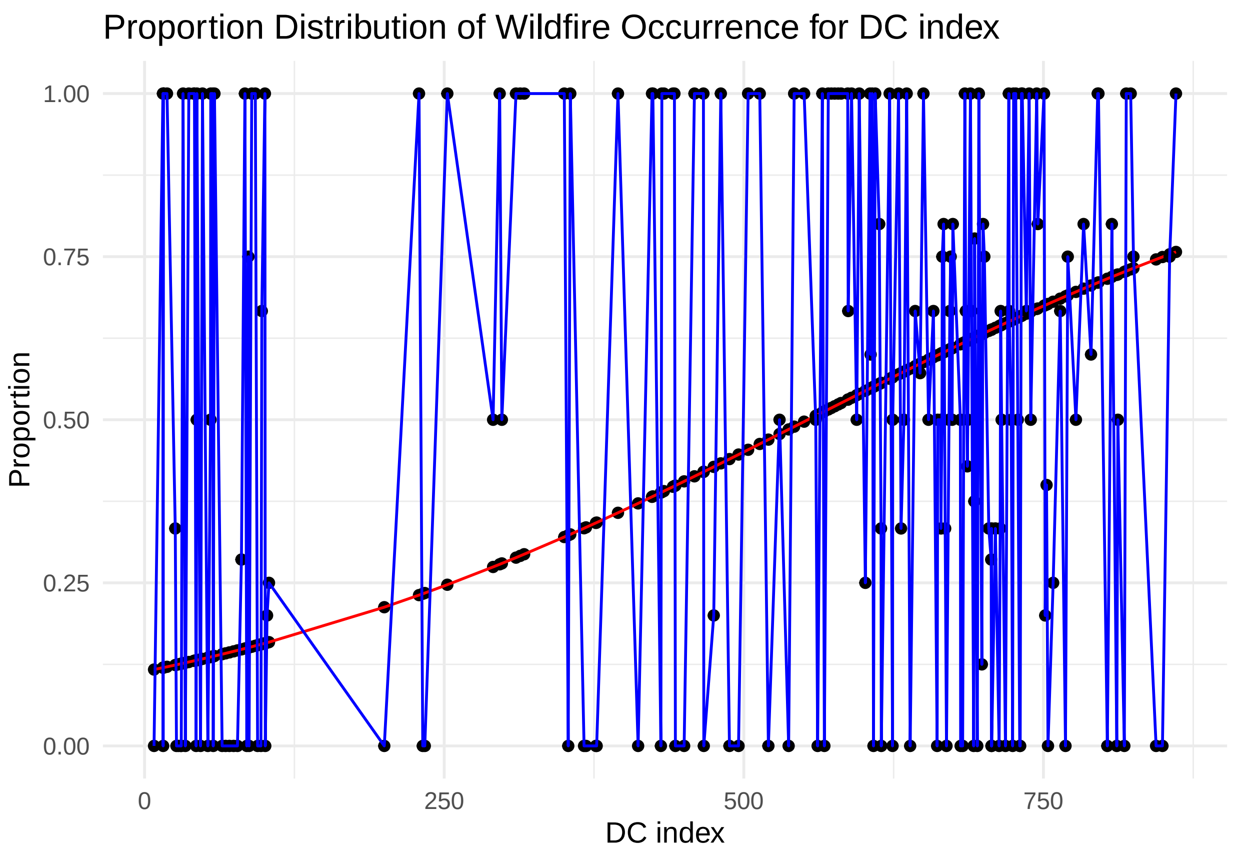
Comparing the two color contour plots, the majority of their squares seem to resemble each other in regards to color shade. This suggests that most of the model's predictions for wildfire occurrence in specific locations (i.e. X and Y values) is right. Therefore, we can conclude that our predictive model is generally accurate in capturing the spatial distribution of wildfires in Montesinho Park (and similar/representative areas) in terms of X and Y coordinates within the park map.

In addition, we can interpret from the correlation coefficient of approximately 0.417 that there is a moderate positive relationship between the predicted and observed proportions of wildfire occurrences for every location. This implies that as the predicted proportions increase or decrease for a particular location (i.e. X and Y combination), the observed proportions from the data set tend to exhibit a similar behavior. However, since this positive relationship is not very strong based on the correlation coefficient, there may be other factors influencing the observed proportions of wildfire occurrences in different locations that are not taken into account by our predictive model.

Visualizing the Predicted and Observed Proportion Distribution of Wildfire Occurrence for Important Environmental Factors:

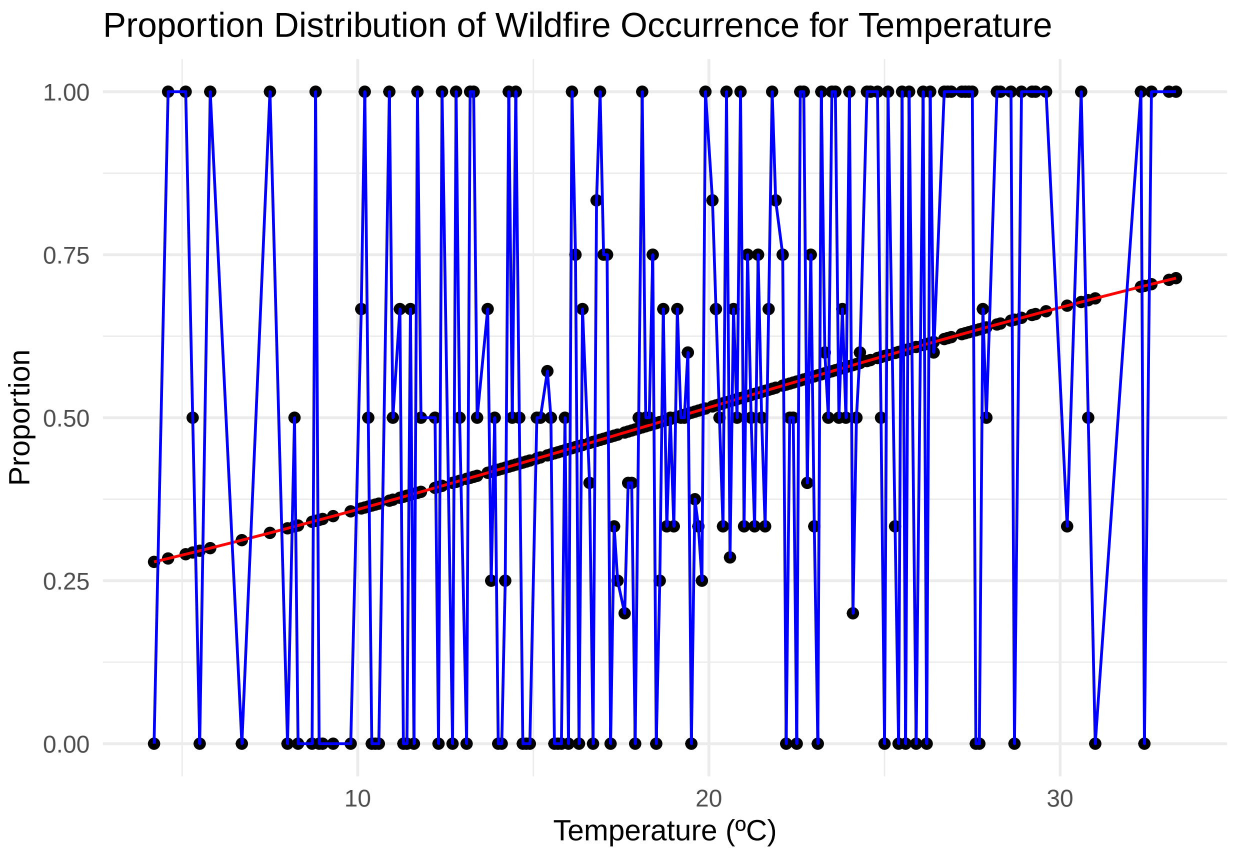
Drought Code (DC) index

```r
# compute the mean of each column in the data set to set it as a constant value
# for each predictor variable respectively (excluding DC and fire)
means <- apply(cleaned_dataset, 2, mean)
means <- means[names(means) != "DC" & names(means) != "fire"]
fixed_variables <- names(means)
names(means) <- NULL

# create a dataframe with the unique values of DC and the fixed variables
pred_DC_data <- data.frame(DC = sort(unique(cleaned_dataset$DC)))
for (i in seq_along(means)) {
  pred_DC_data <- cbind(pred_DC_data,
                        data.frame(means[i]))
}
colnames(pred_DC_data) <- c("DC", fixed_variables)

# predict the proportion of wildfire occurrences for each unique value of DC
predict_prob <- function(row) {
  predict(best_model, newdata = row, type = "response")
}
pred_DC_data <- as_tibble(pred_DC_data) %>%
  group_by(DC) %>% rowwise() %>%
  summarise(prob = predict_prob(data.frame('DC' = DC, 'X' = X, 'Y' = Y,
                                           'month' = month, 'FFMC' = FFMC,
                                           'DMC' = DMC, 'FSS' = FSS, 'temp' = temp,
                                           'RH' = RH, 'wind' = wind, 'rain' = rain))) %>%
  as.data.frame()

# calculate the observed proportion of wildfire occurrences for each unique
# value of DC
actual_DC_data <- cleaned_dataset %>% group_by(DC) %>%
  summarise(prop = mean(fire)) %>% as.data.frame()

# merge the actual and predicted data into one dataframe
merged_df <- merge(pred_DC_data, actual_DC_data, by = "DC")

# display a plot of the predicted proportion (i.e. probability) and observed
# proportion of wildfire occurrences for each unique value of FFMC in the data set
ggplot(data = merged_df, aes(x = DC)) +
  xlab("DC index") + ylab("Proportion") +
  ggtitle("Proportion Distribution of Wildfire Occurrence for DC index") +
  geom_point(aes(y = prob)) + geom_line(aes(y = prob), color = "red") +
  geom_point(aes(y = prop)) + geom_line(aes(y = prop), color = "blue") +
  theme_minimal()
```



Temperature

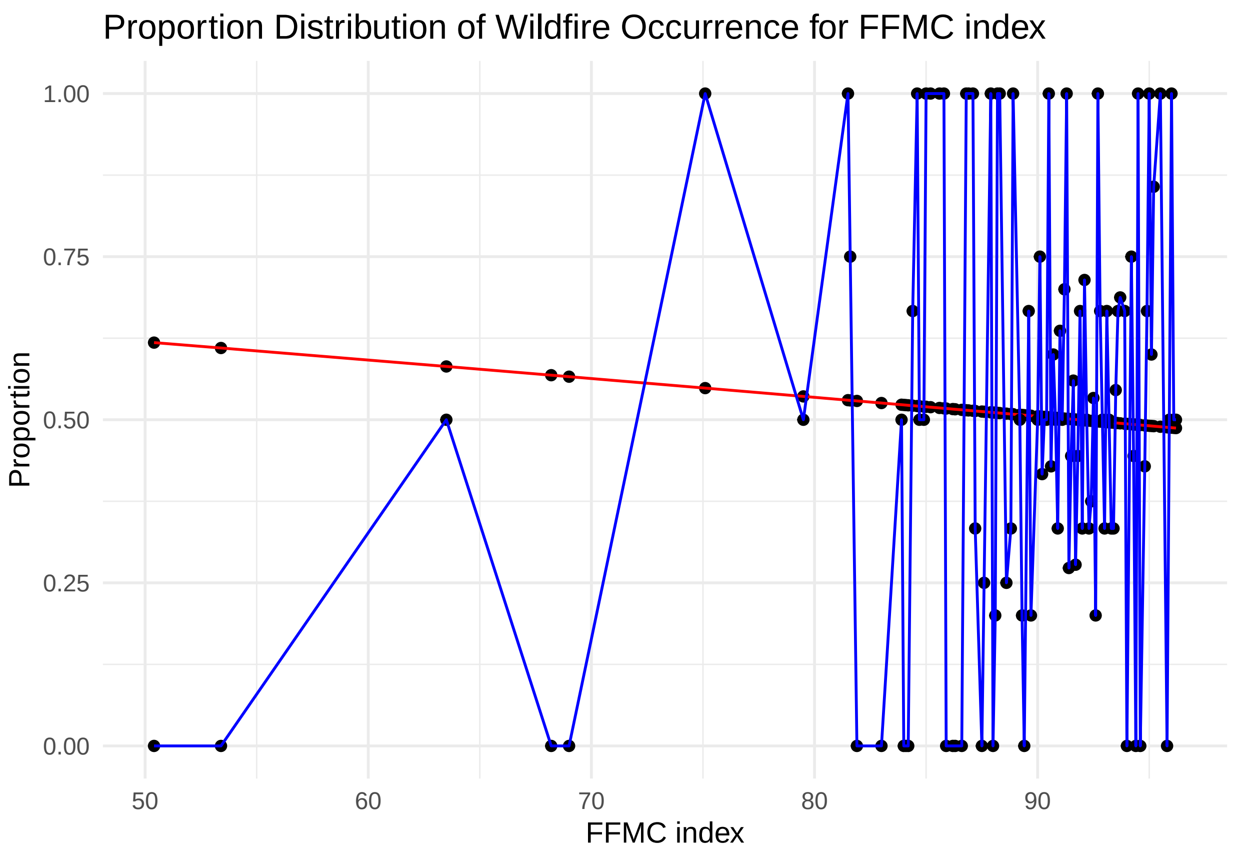
```
# compute the mean of each column in the data set to set it as a constant value
# for each predictor variable respectively (excluding temp and fire)
means <- apply(cleaned_dataset, 2, mean)
means <- means[names(means) != "temp" & names(means) != "fire"]
fixed_variables <- names(means)
names(means) <- NULL

# create a dataframe with the unique values of temp and the fixed variables
pred_temp_data <- data.frame(temp = sort(unique(cleaned_dataset$temp)))
for (i in seq_along(means)) {
  pred_temp_data <- cbind(pred_temp_data,
                          data.frame(means[i]))
}
colnames(pred_temp_data) <- c("temp", fixed_variables)

# predict the proportion of wildfire occurrences for each unique value of temp
predict_prob <- function(row) {
  predict(best_model, newdata = row, type = "response")
}
pred_temp_data <- as_tibble(pred_temp_data) %>%
  group_by(temp) %>% rowwise() %>%
  summarise(prob = predict_prob(data.frame('temp' = temp, 'X' = X, 'Y' = Y,
                                           'month' = month, 'FFMC' = FFMC,
                                           'DMC' = DMC, 'DC' = DC, 'FSS' = FSS,
                                              'RH' = RH, 'wind' = wind, 'rain' = rain))) %>%
  as.data.frame()

# calculate the observed proportion of wildfire occurrences for each unique
# value of temp
actual_temp_data <- cleaned_dataset %>% group_by(temp) %>%
  summarise(prop = mean(fire)) %>% as.data.frame()

# merge the actual and predicted data into one dataframe
merged_df <- merge(pred_temp_data, actual_temp_data, by = "temp")

# display a plot of the predicted proportion (i.e. probability) and observed
# proportion of wildfire occurrences for each unique value of FFMC in the data set
ggplot(data = merged_df, aes(x = temp)) +
  xlab("Temperature (ºC)") + ylab("Proportion") +
  ggtitle("Proportion Distribution of Wildfire Occurrence for Temperature") +
  geom_point(aes(y = prob)) + geom_line(aes(y = prob), color = "red") +
  geom_point(aes(y = prop)) + geom_line(aes(y = prop), color = "blue") +
  theme_minimal()
```



Fine Fuel Moisture Code (FFMC) index

```
# compute the mean of each column in the data set to set it as a constant value
# for each predictor variable respectively (excluding FFMC and fire)
means <- apply(cleaned_dataset, 2, mean)
means <- means[names(means) != "FFMC" & names(means) != "fire"]
```

```r
fixed_variables <- names(means)
names(means) <- NULL

# create a dataframe with the unique values of FFMC and the fixed variables
pred_FFMC_data <- data.frame(FFMC = sort(unique(cleaned_dataset$FFMC)))
for (i in seq_along(means)) {
  pred_FFMC_data <- cbind(pred_FFMC_data,
                          data.frame(means[i]))
}
colnames(pred_FFMC_data) <- c("FFMC", fixed_variables)

# predict the proportion of wildfire occurrences for each unique value of FFMC
predict_prob <- function(row) {
  predict(best_model, newdata = row, type = "response")
}
pred_FFMC_data <- as_tibble(pred_FFMC_data) %>%
  group_by(FFMC) %>% rowwise() %>%
  summarise(prob = predict_prob(data.frame('FFMC' = FFMC, 'X' = X, 'Y' = Y,
                                           'month' = month, 'DMC' = DMC, 'DC' = DC,
                                           'FSS' = FSS, 'temp' = temp, 'RH' = RH,
                                           'wind' = wind, 'rain' = rain))) %>%
  as.data.frame()

# calculate the observed proportion of wildfire occurrences for each unique
# value of FFMC
actual_FFMC_data <- cleaned_dataset %>% group_by(FFMC) %>%
  summarise(prop = mean(fire)) %>% as.data.frame()

# merge the actual and predicted data into one dataframe
merged_df <- merge(pred_FFMC_data, actual_FFMC_data, by = "FFMC")

# display a plot of the predicted proportion (i.e. probability) and observed
# proportion of wildfire occurrences for each unique value of FFMC in the data set
ggplot(data = merged_df, aes(x = FFMC)) +
  xlab("FFMC index") + ylab("Proportion") +
  ggtitle("Proportion Distribution of Wildfire Occurrence for FFMC index") +
  geom_point(aes(y = prob)) + geom_line(aes(y = prob), color = "red") +
  geom_point(aes(y = prop)) + geom_line(aes(y = prop), color = "blue") +
  theme_minimal()
```


Proportion Distribution of Wildfire Occurrence for FFMC index
Proportion
1.00
0.75
0.50
0.25
0.00
50
60
70
80
90
FFMC index

Predictive Modeling for Wildfire Burned Area

We will perform LASSO regression with the Montesinho Park data with the number of acres burned as our response variable. By doing so, we will be able to compare our results to the actual data to evaluate the accuracy of the model.

LASSO regression is a method that is used to fit a regression model when multicollinearity is present in the data. It seeks to minimize the sum of squared residuals summed with the shrinkage penalty (i.e. $\sum(y_i - \hat{y}_i)^2 + \lambda \sum |\beta_j|$ where j ranges from 1 to p predictor variables and $\lambda \geq 0$) where we select a value for λ that produces the lowest possible test mean squared error (MSE).

Initial Steps

```
# Install necessary packages.
install.packages("glmnet", repos = "https://cran.us.r-project.org")
install.packages("readr")
install.packages("caret")
```

```
# Load in necessary libraries.
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-7
```

```
library(readr)
library(caret)
```

Training and Testing LASSO Regression Model:

Prepare the data for LASSO regression.

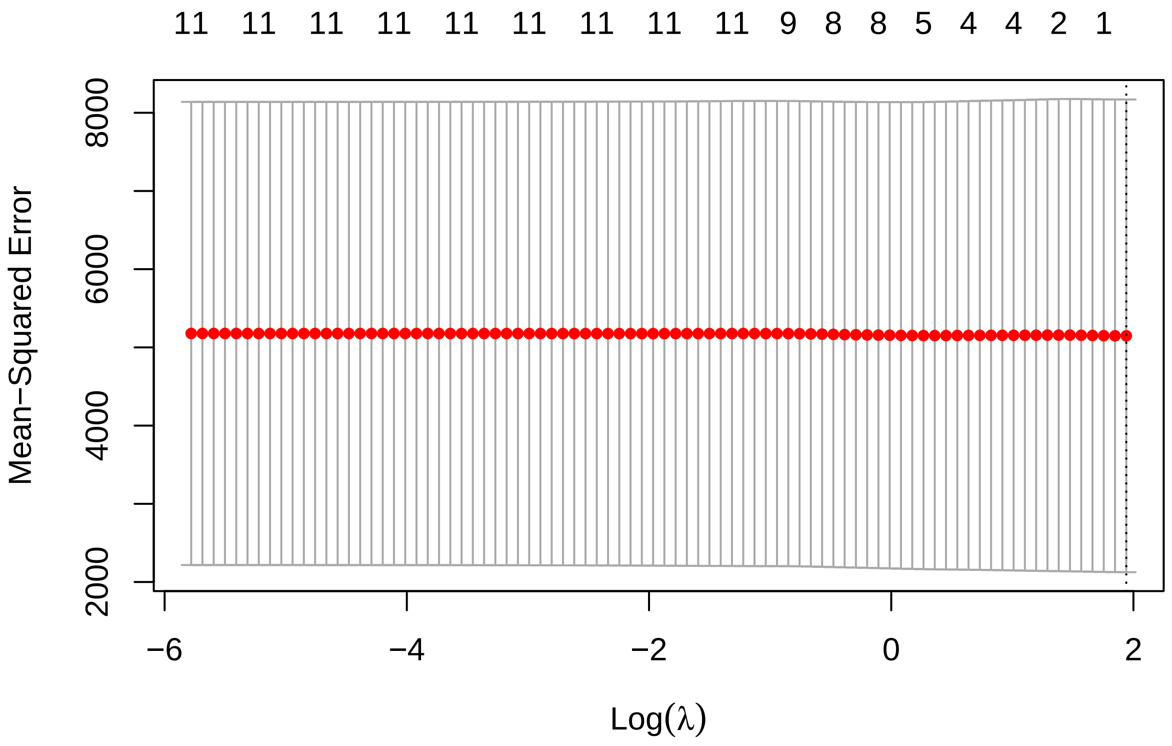
```
# set seed (so we can reproduce this result)
set.seed(1)

# partition (split) and create index matrix of selected values
index <- createDataPartition(cleaned_montesinho$area, p=0.8, list=FALSE, times=1)

# to address error message, convert df to data frame object
cleaned_montesinho <- as.data.frame(cleaned_montesinho)

# create test and training data frames
train_montesinho <- cleaned_montesinho[index,]
test_montesinho <- cleaned_montesinho[-index,]

# Define response variable
response <- train_montesinho$area

# Define matrix of predictor variables.
predictor_variables <- data.matrix(train_montesinho[, c('X', 'Y', 'month', 'FFMC', 'DMC', 'DC', 'FSS',
```

Fit the LASSO regression model.

```r
# perform k-fold cross-validation to find optimal lambda value
# note here that the function cv.glmnet() automatically performs k-fold cross validation using k=10 fol
cv_model <- cv.glmnet(predictor_variables, response, alpha=1)

# find optimal lambda value that minimizes test MSE
best_lambda_lasso <-cv_model$lambda.min
best_lambda_lasso
```

```
## [1] 6.965441
```

```r
# produce plot of test MSE by lambda value
plot(cv_model)
```



```r
#find coefficients of best model
best_model_lasso <- glmnet(predictor_variables, response, alpha = 1, lambda = best_lambda_lasso)
coef(best_model_lasso)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) 1.380714e+01
## X           .
## Y           .
## month       .
## FFMC        .
## DMC         .
## DC          .
## FSS         .
## temp        1.828647e-16
## RH          .
## wind        .
## rain        .
```

Using the model on the testing data.

```
#model prediction
predictions1 <- predict(best_model_lasso, newx=data.matrix(test_montesinho[, c('X', 'Y', 'month', 'FFMC

#model performance/accuracy
mod1perf <- data.frame(RMSE=RMSE(predictions1, test_montesinho$area), Rsquared=R2(predictions1, test_mon
colnames(mod1perf) <- c("RMSE", "R2")
mod1perf
```

```
##       RMSE         R2
## 1 16.80416 0.02690776
```

```
modpercentage <- mod1perf$R2
```

RMSE is a valuable performance indicator for regression models, as it provides an estimation of how well the model is able to predict the target value. It considers the average difference between predicted and actual values, taking into account both the direction and magnitude of the errors. Lower RMSE values indicate higher accuracy and better model performance. As shown in the results, the RMSE is a higher value so we can conclude that the model is not very accurate.

The R-squared turns out to be 0.0269078. That is, the best model was able to explain 0.0269078 of the variation in the response values of the testing data.