Introduction

Data Overview

Exploratory Data Analysis

Splitting Data and Cross Fold Validation

Fitting the Models

Overall Performance of Models

Conclusion

Code

PSTAT131_FinalProject

Immad Ali

Introduction

What is League of Legends?

League of Legends (LoL) is a fast paced, multiplier online battle arena styled game in which two teams of five fight against each other and destroy the enemies base. At the heart of each team is the Nexus. Destroying the enemies Nexus results in you winning the game. The map, known as Summoner's Rift, consists of a roughly square map with each team in a corner. The map is perfectly symmetrical and consists of three lanes: top, middle, and bottom. There is also a jungle space between each lane. Players also level up their champions by gaining experience. Experience is gained by killing minions, killing enemies, or other neutral enemies on the map. In order to win the game, you're team must progress against the enemy's turrets, which defend the enemy nexus. The team that successful defeats the other teams Nexus, wins.

Code

Data Source

The data I am using is from the first 10 minutes of a LoL Ranked game in the Diamond Rank. I found the data set posted on Kaggle by the user Michel’s Fanboi (who now goes by Yi Lan Ma). https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min (https://www.kaggle.com/datasets/bobbyscience/league-of-legends-diamond-ranked-games-10-min)

Data Overview

Processing the Data

- In this step, I will go ahead and read the CSV file in and check the first 6 rows of the data set.

Code

```
##       gameId blueWins blueWardsPlaced blueWardsDestroyed blueFirstBlood
## 1 4519157822        0              28                  2              1
## 2 4523371949        0              12                  1              0
## 3 4521474530        0              15                  0              0
## 4 4524384067        0              43                  1              0
## 5 4436033771        0              75                  4              0
## 6 4475365709        1              18                  0              0
##   blueKills blueDeaths blueAssists blueEliteMonsters blueDragons blueHeralds
## 1         9          6          11                 0           0           0
## 2         5          5           5                 0           0           0
## 3         7         11           4                 1           1           0
## 4         4          5           5                 1           0           1
## 5         6          6           6                 0           0           0
## 6         5          3           6                 1           1           0
##   blueTowersDestroyed blueTotalGold blueAvgLevel blueTotalExperience
## 1                   0         17210          6.6               17039
## 2                   0         14712          6.6               16265
## 3                   0         16113          6.4               16221
## 4                   0         15157          7.0               17954
## 5                   0         16400          7.0               18543
## 6                   0         15899          7.0               18161
##   blueTotalMinionsKilled blueTotalJungleMinionsKilled blueGoldDiff
## 1                    195                           36          643
## 2                    174                           43        -2908
## 3                    186                           46        -1172
## 4                    201                           55        -1321
## 5                    210                           57        -1004
## 6                    225                           42          698
##   blueExperienceDiff blueCSPerMin blueGoldPerMin redWardsPlaced
## 1                 -8         19.5         1721.0             15
## 2              -1173         17.4         1471.2             12
## 3              -1033         18.6         1611.3             15
## 4                 -7         20.1         1515.7             15
## 5                230         21.0         1640.0             17
## 6                101         22.5         1589.9             36
##   redWardsDestroyed redFirstBlood redKills redDeaths redAssists
## 1                 6             0        6         9          8
## 2                 1             1        5         5          2
## 3                 3             1       11         7         14
## 4                 2             1        5         4         10
## 5                 2             1        6         6          7
## 6                 5             1        3         5          2
##   redEliteMonsters redDragons redHeralds redTowersDestroyed redTotalGold
## 1                0          0          0                  0        16567
## 2                2          1          1                  1        17620
## 3                0          0          0                  0        17285
## 4                0          0          0                  0        16478
## 5                1          1          0                  0        17404
## 6                0          0          0                  0        15201
##   redAvgLevel redTotalExperience redTotalMinionsKilled
## 1         6.8              17047                   197
## 2         6.8              17438                   240
## 3         6.8              17254                      203
## 4         7.0              17961                      235
## 5         7.0              18313                      225
## 6         7.0              18060                      221
##   redTotalJungleMinionsKilled redGoldDiff redExperienceDiff redCSPerMin
## 1                          55        -643                 8        19.7
## 2                          52        2908              1173        24.0
## 3                          28        1172              1033        20.3
## 4                          47        1321                 7        23.5
## 5                          67        1004              -230        22.5
## 6                          59        -698              -101        22.1
##   redGoldPerMin
## 1        1656.7
## 2        1762.0
## 3        1728.5
## 4        1647.8
## 5        1740.4
## 6        1520.1
```

- As we can see, the data has 9,879 observations and has 40 columns. One of those columns is the gameID, which is not helpful to our model. We will go ahead and remove that column from the data set. We will also go ahead and check for any missing values.

Code

```
##                     blueWins              blueWardsPlaced
##                            0                            0
##           blueWardsDestroyed               blueFirstBlood
##                            0                            0
##                    blueKills                   blueDeaths
##                            0                            0
##                  blueAssists            blueEliteMonsters
##                            0                            0
##                  blueDragons                  blueHeralds
##                            0                            0
##          blueTowersDestroyed                blueTotalGold
##                            0                            0
##                 blueAvgLevel          blueTotalExperience
##                            0                            0
##       blueTotalMinionsKilled blueTotalJungleMinionsKilled
##                            0                            0
##                 blueGoldDiff           blueExperienceDiff
##                            0                            0
##                 blueCSPerMin               blueGoldPerMin
##                            0                            0
##               redWardsPlaced            redWardsDestroyed
##                            0                            0
##                redFirstBlood                     redKills
##                            0                            0
##                    redDeaths                   redAssists
##                            0                            0
##             redEliteMonsters                   redDragons
##                            0                            0
##                   redHeralds           redTowersDestroyed
##                            0                            0
##                 redTotalGold                  redAvgLevel
##                            0                            0
##           redTotalExperience        redTotalMinionsKilled
##                            0                            0
##  redTotalJungleMinionsKilled                  redGoldDiff
##                            0                            0
##            redExperienceDiff                  redCSPerMin
##                            0                            0
##                redGoldPerMin
##                            0
```

- There is no missing data in our dataset. I had also addressed the issue of missing data in my data memo. Since the original data set had the gameID, I could go back and manually fill in the missing data, as long as it wasn't to many.
- I was curious about some statistics for the predictors. I went ahead and used the summary function to look at the overall statistics of each predictor.

Code

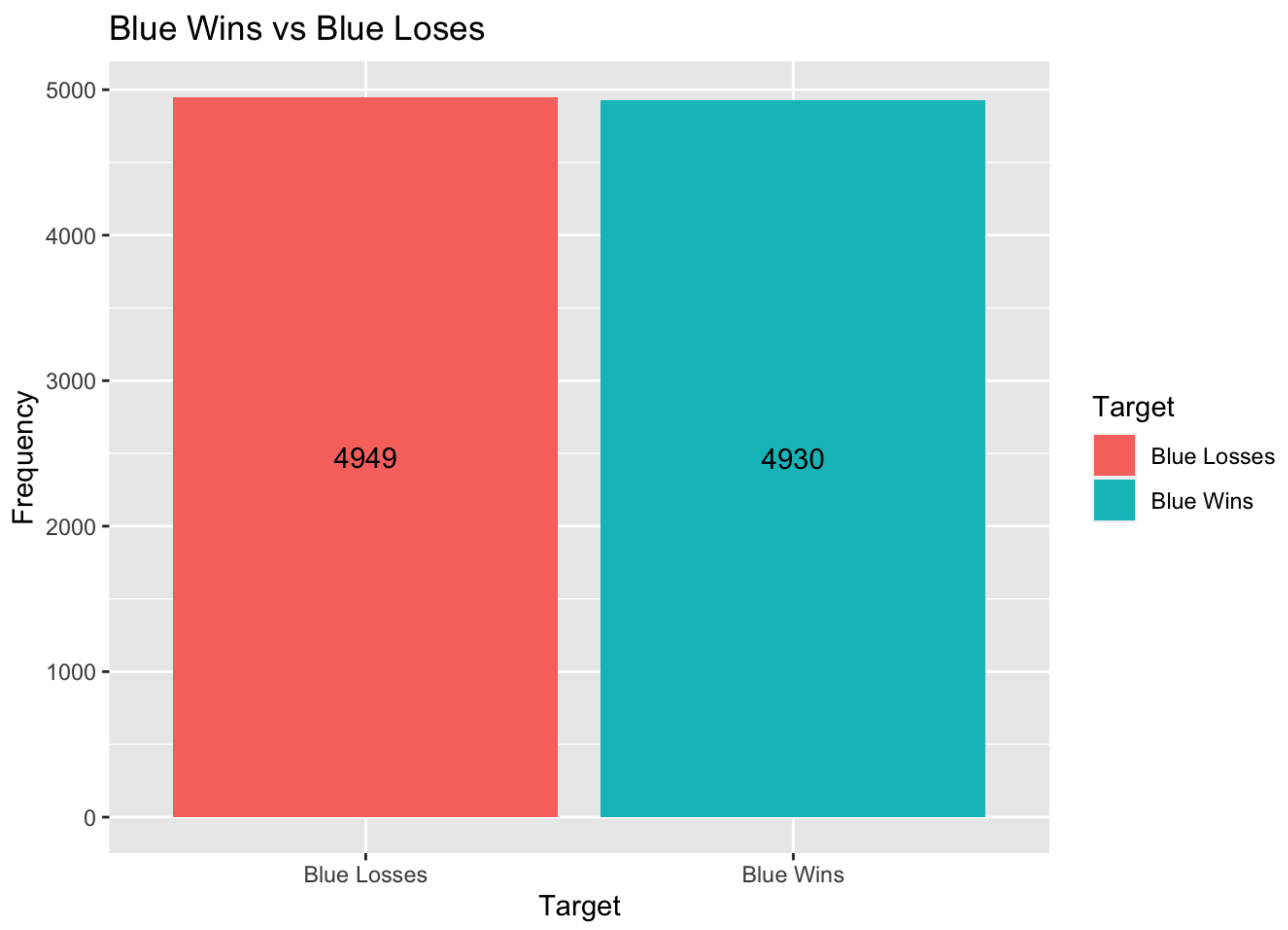
```
##     blueWins     blueWardsPlaced  blueWardsDestroyed blueFirstBlood  
##  Min.   :0.000   Min.   :  5.00   Min.   : 0.000     Min.   :0.0000  
##  1st Qu.:0.000   1st Qu.: 14.00   1st Qu.: 1.000     1st Qu.:0.0000  
##  Median :0.000   Median : 16.00   Median : 3.000     Median :1.0000  
##  Mean   :0.499   Mean   : 22.29   Mean   : 2.825     Mean   :0.5048  
##  3rd Qu.:1.000   3rd Qu.: 20.00   3rd Qu.: 4.000     3rd Qu.:1.0000  
##  Max.   :1.000   Max.   :250.00   Max.   :27.000     Max.   :1.0000  
##    blueKills        blueDeaths      blueAssists     blueEliteMonsters
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :0.00     
##  1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.:0.00     
##  Median : 6.000   Median : 6.000   Median : 6.000   Median :0.00     
##  Mean   : 6.184   Mean   : 6.138   Mean   : 6.645   Mean   :0.55     
##  3rd Qu.: 8.000   3rd Qu.: 8.000   3rd Qu.: 9.000   3rd Qu.:1.00     
##  Max.   :22.000   Max.   :22.000   Max.   :29.000   Max.   :2.00     
##   blueDragons     blueHeralds    blueTowersDestroyed blueTotalGold  
##  Min.   :0.000   Min.   :0.000   Min.   :0.00000     Min.   :10730  
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.00000     1st Qu.:15416  
##  Median :0.000   Median :0.000   Median :0.00000     Median :16398  
##  Mean   :0.362   Mean   :0.188   Mean   :0.05142     Mean   :16503  
##  3rd Qu.:1.000   3rd Qu.:0.000   3rd Qu.:0.00000     3rd Qu.:17459  
##  Max.   :1.000   Max.   :1.000   Max.   :4.00000     Max.   :23701  
##   blueAvgLevel   blueTotalExperience blueTotalMinionsKilled
##  Min.   :4.600   Min.   :10098       Min.   : 90.0         
##  1st Qu.:6.800   1st Qu.:17168       1st Qu.:202.0         
##  Median :7.000   Median :17951       Median :218.0         
##  Mean   :6.916   Mean   :17928       Mean   :216.7         
##  3rd Qu.:7.200   3rd Qu.:18724       3rd Qu.:232.0         
##  Max.   :8.000   Max.   :22224       Max.   :283.0         
##  blueTotalJungleMinionsKilled  blueGoldDiff       blueExperienceDiff
##  Min.   : 0.00                Min.   :-10830.00   Min.   :-9333.00  
##  1st Qu.:44.00                1st Qu.: -1585.50   1st Qu.:-1290.50  
##  Median :50.00                Median :    14.00   Median :  -28.00  
##  Mean   :50.51                Mean   :    14.41   Mean   :  -33.62  
##  3rd Qu.:56.00                3rd Qu.:  1596.00   3rd Qu.: 1212.00  
##  Max.   :92.00                Max.   : 11467.00   Max.   : 8348.00  
##   blueCSPerMin   blueGoldPerMin redWardsPlaced   redWardsDestroyed
##  Min.   : 9.00   Min.   :1073   Min.   :  6.00   Min.   : 0.000   
##  1st Qu.:20.20   1st Qu.:1542   1st Qu.: 14.00   1st Qu.: 1.000   
##  Median :21.80   Median :1640   Median : 16.00   Median : 2.000   
##  Mean   :21.67   Mean   :1650   Mean   : 22.37   Mean   : 2.723   
##  3rd Qu.:23.20   3rd Qu.:1746   3rd Qu.: 20.00   3rd Qu.: 4.000   
##  Max.   :28.30   Max.   :2370   Max.   :276.00   Max.   :24.000   
##  redFirstBlood       redKills         redDeaths        redAssists    
##  Min.   :0.0000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.:0.0000   1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 4.000  
##  Median :0.0000   Median : 6.000   Median : 6.000   Median : 6.000  
##  Mean   :0.4952   Mean   : 6.138   Mean   : 6.184   Mean   : 6.662  
##  3rd Qu.:1.0000   3rd Qu.: 8.000   3rd Qu.: 8.000   3rd Qu.: 9.000  
##  Max.   :1.0000   Max.   :22.000   Max.   :22.000   Max.   :28.000  
##  redEliteMonsters   redDragons       redHeralds   redTowersDestroyed
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00   Min.   :0.00000   
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:0.00000   
##  Median :0.0000   Median :0.0000   Median :0.00   Median :0.00000  
##  Mean   :0.5731   Mean   :0.4131   Mean   :0.16   Mean   :0.04302  
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.00   3rd Qu.:0.00000  
##  Max.   :2.0000   Max.   :1.0000   Max.   :1.00   Max.   :2.00000  
##   redTotalGold    redAvgLevel    redTotalExperience redTotalMinionsKilled
##  Min.   :11212   Min.   :4.800   Min.   :10465      Min.   :107.0        
##  1st Qu.:15428   1st Qu.:6.800   1st Qu.:17210      1st Qu.:203.0        
##  Median :16378   Median :7.000   Median :17974      Median :218.0        
##  Mean   :16489   Mean   :6.925   Mean   :17962      Mean   :217.3        
##  3rd Qu.:17418   3rd Qu.:7.200   3rd Qu.:18764      3rd Qu.:233.0        
##  Max.   :22732   Max.   :8.200   Max.   :22269      Max.   :289.0        
##  redTotalJungleMinionsKilled  redGoldDiff        redExperienceDiff
##  Min.   : 4.00              Min.   :-11467.00   Min.   :-8348.00  
##  1st Qu.:44.00              1st Qu.: -1596.00   1st Qu.:-1212.00  
##  Median :51.00              Median :   -14.00   Median :   28.00  
##  Mean   :51.31              Mean   :   -14.41   Mean   :   33.62  
##  3rd Qu.:57.00              3rd Qu.:  1585.50   3rd Qu.: 1290.50  
##  Max.   :92.00              Max.   : 10830.00   Max.   : 9333.00  
##   redCSPerMin    redGoldPerMin  
##  Min.   :10.70   Min.   :1121  
##  1st Qu.:20.30   1st Qu.:1543  
##  Median :21.80   Median :1638  
##  Mean   :21.73   Mean   :1649  
##  3rd Qu.:23.30   3rd Qu.:1742  
##  Max.   :28.90   Max.   :2273  
```

Exploratory Data Analysis

Win Loss Bar graph

- I decided to do a simple bar graph just to explore how the blue team was split in terms of wins versus losses.

Code

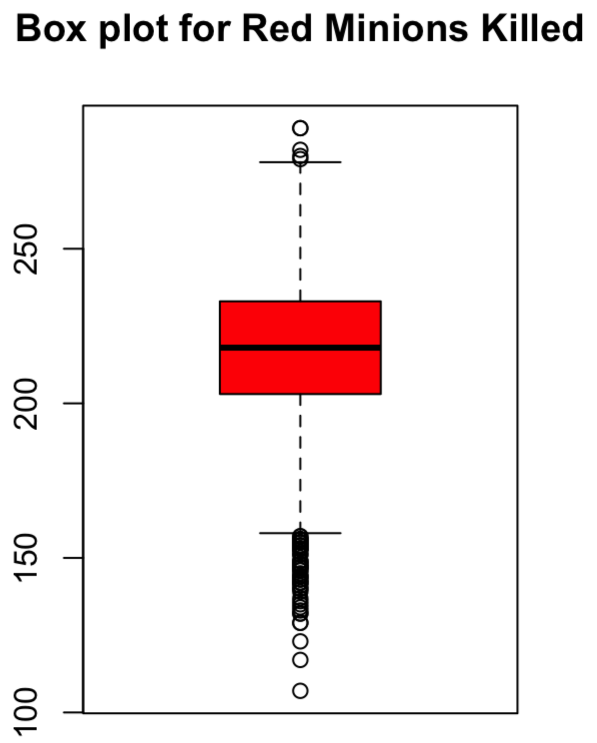


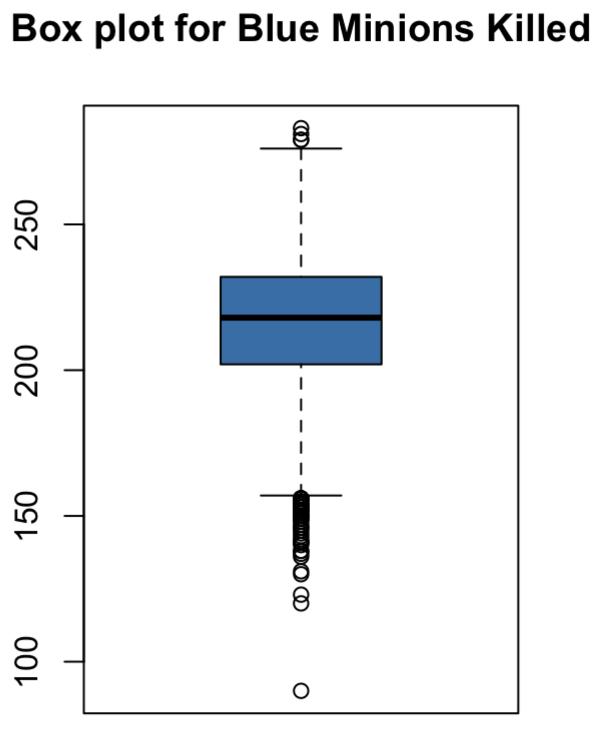
- As we can see from the bar graph above, the data seems to be split 50/50 in terms of how many games the blue team won and how many they lost. We will need to dig more in depth if we want to find out what predictors have the biggest impact on blue team winning the game.

Boxplot Red Minions Killed versus Blue Minions Killed

- To get a deeper look into the initial data set, I used a box plot to show the overall distribution of the red team and blue team minons killed.
- I wanted to get a deeper look into this because the main source of gold in this game comes from the minions killed. Therefore, we would be able to see correlations in terms how many minions were killed to the teams respective over all gold.

Code



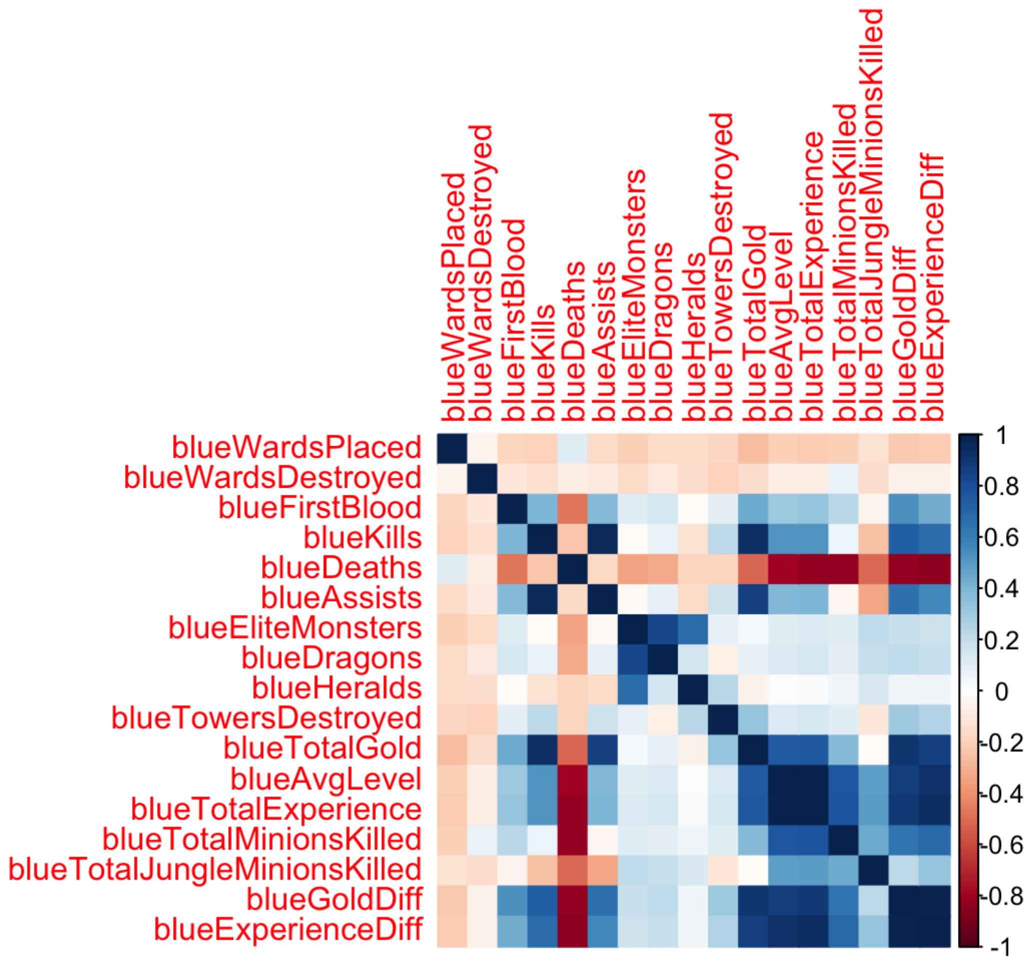


- From the two boxplots, we can see that the blue team had significantly more minions killed in each quartile. Therefore, we can say that the blue team had overall more minions killed.

Correlation Map of Blue Team Predictors

- I decided to make a blue team predictors correlation map to better visualize what certain predictors helped the blue team win.

Code

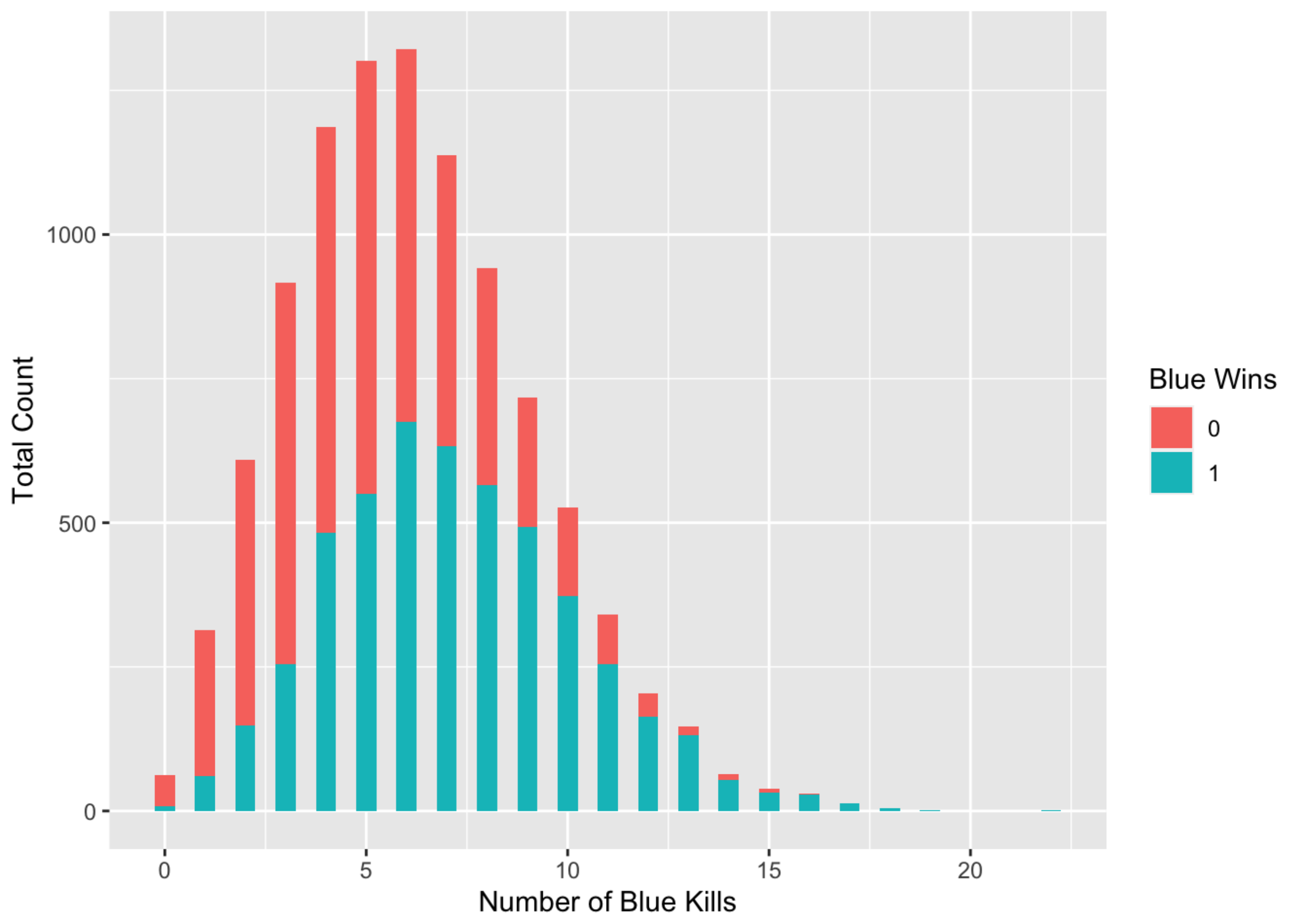


- As shown from the correlation map, there was a large correlation between total minions killed and total gold for the blue team, which confirms our findings from the boxplots. In addition to that, I found that total gold for blue team was heavily correlated to the number of blue team kills. We will explore that in our next EDA. I also wanted to point out vertain predictors that were NOT correlated. I found that the total blue team experience had a negative correlation to the number of blue team deaths. This was actually surprising to me as I thought the two would be very correlated.

Histogram of Blue Wins vs Losses including Number of Kills

- As I mentioned above, I wanted to look a bit more indepth in the total gold for blue team and the total kills for blue team.

Code



- Looking at our histogram, we can see that as the number of kills blue team acquired, the more likely they were to win the game.

Splitting Data and Cross Fold Validation

- In this step, I go ahead and split the original data set into a training and test set. I went ahead and chose a 70-30 split of the data. I also checked the dimensions of the test and training set to insure the they added up to the total observations.

Code

```
## [1] 6916   39
```

Code

```
## [1] 2963   39
```

- In the training set, there are 6916 observations and in the testing set, there are 2963 observations.

Fitting the Models

Logisitc Model

- Here I go ahead and fit the logistics regression model to our data. I also cross-validate the data.

Code

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

Code

```
## Generalized Linear Model
##
## 6916 samples
##   38 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5533, 5533, 5533, 5533, 5532
## Resampling results:
##
##   Accuracy   Kappa
##   0.7323618  0.4647228
```

- As we can see from the results, I got an accuracy rate of 73.2%. I was also getting a weird error as shown above. Unfortunately, I was unable to resolve this error after checking online and with data scientist friends.

Random Forest Model

Code

```
## Random Forest
##
## 6916 samples
##   38 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5533, 5533, 5533, 5532, 5533
## Resampling results:
##
##   Accuracy   Kappa
##   0.7129852  0.4259797
##
## Tuning parameter 'mtry' was held constant at a value of 38
```

- As you can see, we got an accuracy rating of 71.64% for the Random Forest Model.

Boosting Model

Code

```
## Stochastic Gradient Boosting
##
## 6916 samples
##   38 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5533, 5533, 5532, 5533, 5533
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7283079  0.4566357
##   1                  100      0.7252725  0.4505502
##   1                  150      0.7254169  0.4508354
##   2                   50      0.7270074  0.4540047
##   2                  100      0.7229588  0.4459260
##   2                  150      0.7226700  0.4453456
##   3                   50      0.7239713  0.4479352
##   3                  100      0.7183324  0.4366718
##   3                  150      0.7174656  0.4349311
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  1, shrinkage = 0.1 and n.minobsinnode = 10.
```

- From the results above, it seems that the accuracy rate was between 72.2% and 72.9%

K Nearest Neighbour Model

Code

```
## k-Nearest Neighbors
##
## 6916 samples
##   38 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 5533, 5532, 5533, 5533, 5533
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6912956  0.3826009
##   7  0.6989591  0.3979208
##   9  0.7057547  0.4115132
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

- From the results above, I got the highest accuracy rate at k = 9, at 71% I got the lowest accuracy rate at k = 5, at 69.12%

Overall Performance of Models

- I will now go ahead and check the overall performance of my models and determine which of the models best fits my data.

Linear Regression Model Evaluation

Code

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

Code

```
## Confusion Matrix and Statistics
##
##
## pred    0    1
##    0 1076  379
##    1  423 1085
##
##                Accuracy : 0.7293
##                  95% CI : (0.7129, 0.7453)
##     No Information Rate : 0.5059
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.4588
##
##  Mcnemar's Test P-Value : 0.1289
##
##             Sensitivity : 0.7178
##             Specificity : 0.7411
##          Pos Pred Value : 0.7395
##          Neg Pred Value : 0.7195
##              Prevalence : 0.5059
##          Detection Rate : 0.3631
##    Detection Prevalence : 0.4911
##       Balanced Accuracy : 0.7295
##
##        'Positive' Class : 0
##
```

- After fitting the logistic model to our testing data set, I computed the confusion matrix. From the prediction model, we can see that the Linear Regression model had an accuracy of 72.93%.

Random Forest Model Evaluation

Code

```
## Confusion Matrix and Statistics
##
##
## pred    0    1
##    0 1087  406
##    1  412 1058
##
##                Accuracy : 0.7239
##                  95% CI : (0.7074, 0.74)
##     No Information Rate : 0.5059
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.4478
##
##  Mcnemar's Test P-Value : 0.8612
##
##             Sensitivity : 0.7252
##             Specificity : 0.7227
##          Pos Pred Value : 0.7281
##          Neg Pred Value : 0.7197
##              Prevalence : 0.5059
##          Detection Rate : 0.3669
##    Detection Prevalence : 0.5039
##       Balanced Accuracy : 0.7239
##
##        'Positive' Class : 0
##
```

- After fitting the Random Forest Model to the testing set, I computed the confusion matrix. As we can see, we got an accuracy rate of 72.2%. This model performed slightly worse than our logistic regression model.

Boosting Model Evaluation

Code

```
## Confusion Matrix and Statistics
##
##
## pred    0    1
##    0 1078  372
##    1  421 1092
##
##                Accuracy : 0.7324
##                  95% CI : (0.716, 0.7482)
##     No Information Rate : 0.5059
##     P-Value [Acc > NIR] : < 0.0000000000000002
##
##                   Kappa : 0.4649
##
##  Mcnemar's Test P-Value : 0.08828
##
##             Sensitivity : 0.7191
##             Specificity : 0.7459
##          Pos Pred Value : 0.7434
##          Neg Pred Value : 0.7217
##              Prevalence : 0.5059
##          Detection Rate : 0.3638
##    Detection Prevalence : 0.4894
##       Balanced Accuracy : 0.7325
##
##        'Positive' Class : 0
##
```

- After fitting the Boosting Model to the testing set, I computed the confusion matrix. The boosting model gave us an accuracy rate of 73.17%, which was the highest amonghts the models.

K-Nearest Neighbour Model Evalution

Code

```
## Confusion Matrix and Statistics
##
##
## pred    0    1
##    0 1070  451
##    1  429 1013
##
##                Accuracy : 0.703
##                  95% CI : (0.6862, 0.7194)
##     No Information Rate : 0.5059
##     P-Value [Acc > NIR] : <0.0000000000000002
##
##                   Kappa : 0.4058
##
##  Mcnemar's Test P-Value : 0.479
##
##             Sensitivity : 0.7138
##             Specificity : 0.6919
##          Pos Pred Value : 0.7035
##          Neg Pred Value : 0.7025
##              Prevalence : 0.5059
##          Detection Rate : 0.3611
##    Detection Prevalence : 0.5133
##       Balanced Accuracy : 0.7029
##
##        'Positive' Class : 0
##
```

- Finally, for our last model, we fit the KNN model to our testing set and got an accuracy rate of 70.27%. This model performed the worst relative to our other models.

Conclusion

To recap what I learned from this data analysis project. I first went ahead and did exploratory data analysis on the overall data set. In my EDA, I found there are several key predictors that are highly correlated. I was also able to learn through my correlation map, that two predictors I thought would be correlated, were actually not. I also looked at the overall games won by blue team factoring in the total kills the team collected.

I then went ahead and chose four machine learning models, Logistic Regression, Random Forest, Boosting, and K-Nearest Neighbour, and used some tuning to look at their models. I then used cross-validation to test those four models and determine the overall accuracy of the model. Based on the accuracy rate of the logistic model, I can say that the that it had the best prediction effects on the testing set. I would also give the boosted model a honorary award for having the second highest accuracy rate among the other models.

I then went ahead and fit the models to our test set to determine the best performing model. I found that the boosting model ended up having the highest accuracy relative to the other models. In the end, I would say we could definitely predict the game results of the blue team winning or losing based on the data set using our machine learning model.

I would like to add a final note on how I could possibly improve this project. For one, I could get a larger data set to test and train our models. I think the biggest way I could help improve this project is to add more tuning to each model. In addition, adding some more EDA looking at other key predictors would be beneficial as well!