Luis Ruiz

League of Legends

Final Report:
League of Legends

Problem Statement

According to statista.com the revenue of the global eSport market is around 1.08 billion USD and is forecasted to grow to as much as 1.62 billion USD by 2024. League of Legends is one of the leading games for prize money with an astonishing 90 million USD paid out to eSport players since 2011. That's on average 8.2 million USD paid out to LoL players. We can expect to see that number increase due the rapid success eSports has seen in the last 10 years. Many universities have recognized the growth in eSports and have begun to offer young gamers scholarships to lure them in. The idea that competitive LoL has the potential to reach FIFA, NFL, or NBA heights is not far-fetched. Several countries including the US have recognized eSports as a sport and at some point, was in talks to be added to the upcoming Olympics.

According to Riot, the creators of LoL, there is a 50% chance of winning each game. If we can increase the win rate by at least 1% to 51% the player would have an above average win rate. Identifying the target variables is essential to increasing the win rate.

In this project, I aim to implement state of the art machine learning techniques to develop a classification model which can predict the probability of your win rate based on key factors. Given how LoL works my model can only be applied to a certain patch of the game. With each patch there comes a new meta which can drastically change the values therefore we can only apply the model to the given patch at the time of data scraping. One constraint is that our data measures only the first 10 minutes of the game which is considered the laning phase. Moreover, our data can only be applied to the NA region as each region across the world approaches the game differently. To validate my conclusions, it is imperative that I discuss my findings with a competitive LoL player.

What is League of Legends? (LoL)

CONTEXT

League of Legends is a MOBA (multiplayer online battle arena) where 2 teams (blue and red) face off. There are 3 lanes, a jungle, and 5 roles. The goal is to take down the enemy Nexus to win the game.

GLOSSARY

- Warding totem: An item that a player can put on the map to reveal the nearby area. Very useful for map/objectives control.
- Minions: NPC that belong to both teams. They give gold when killed by players.
- Jungle minions: NPC that belong to NO TEAM. They give gold and buffs when killed by players.
- Elite monsters: Monsters with high hp/damage that give a massive bonus (gold/XP/stats) when killed by a team.
- Dragons: Elite monster which gives team bonus when killed. The 4th dragon killed by a team - gives a massive stats bonus. The - - 5th dragon (Elder Dragon) offers a huge advantage to the team.
- Herald: Elite monster which gives stats bonus when killed by the player. It helps to push a lane and destroys structures.
- Towers: Structures you have to destroy to reach the enemy Nexus. They give gold.
- Level: Champion level. Start at 1. Max is 18.

Data Wrangling

The raw dataset contained about 10,000 ranked games from a high ELO (Diamond - Masters) which is according to Riot is the 99 percentiles. This data only contains the first 10 minutes of the game which is consider the laning phase. Our Data set contains 40 columns:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9879 entries, 0 to 9878
Data columns (total 40 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   gameId                        9879 non-null   int64  
 1   blueWins                      9879 non-null   int64  
 2   blueWardsPlaced               9879 non-null   int64  
 3   blueWardsDestroyed            9879 non-null   int64  
 4   blueFirstBlood                9879 non-null   int64  
 5   blueKills                     9879 non-null   int64  
 6   blueDeaths                    9879 non-null   int64  
 7   blueAssists                   9879 non-null   int64  
 8   blueEliteMonsters             9879 non-null   int64  
 9   blueDragons                   9879 non-null   int64  
 10  blueHeralds                   9879 non-null   int64  
 11  blueTowersDestroyed           9879 non-null   int64  
 12  blueTotalGold                 9879 non-null   int64  
 13  blueAvgLevel                  9879 non-null   float64
 14  blueTotalExperience           9879 non-null   int64  
 15  blueTotalMinionsKilled        9879 non-null   int64  
 16  blueTotalJungleMinionsKilled  9879 non-null   int64  
 17  blueGoldDiff                  9879 non-null   int64  
 18  blueExperienceDiff            9879 non-null   int64  
 19  blueCSPerMin                  9879 non-null   float64
 20  blueGoldPerMin                9879 non-null   float64
 21  redWardsPlaced                9879 non-null   int64  
 22  redWardsDestroyed             9879 non-null   int64  
 23  redFirstBlood                 9879 non-null   int64  
 24  redKills                      9879 non-null   int64  
 25  redDeaths                     9879 non-null   int64  
 26  redAssists                    9879 non-null   int64  
 27  redEliteMonsters              9879 non-null   int64  
 28  redDragons                    9879 non-null   int64  
 29  redHeralds                    9879 non-null   int64  
 30  redTowersDestroyed            9879 non-null   int64  
 31  redTotalGold                  9879 non-null   int64  
 32  redAvgLevel                   9879 non-null   float64
 33  redTotalExperience            9879 non-null   int64  
 34  redTotalMinionsKilled         9879 non-null   int64  
 35  redTotalJungleMinionsKilled   9879 non-null   int64  
 36  redGoldDiff                   9879 non-null   int64  
 37  redExperienceDiff             9879 non-null   int64  
 38  redCSPerMin                   9879 non-null   float64
 39  redGoldPerMin                 9879 non-null   float64
dtypes: float64(6), int64(34)
memory usage: 3.0 MB
(9879, 40)
```

Our dataset contains two target values ‘blueWins’ and ‘redWins’. Since we want to predict on one team it makes sense to drop one of the team in this case, the red team.

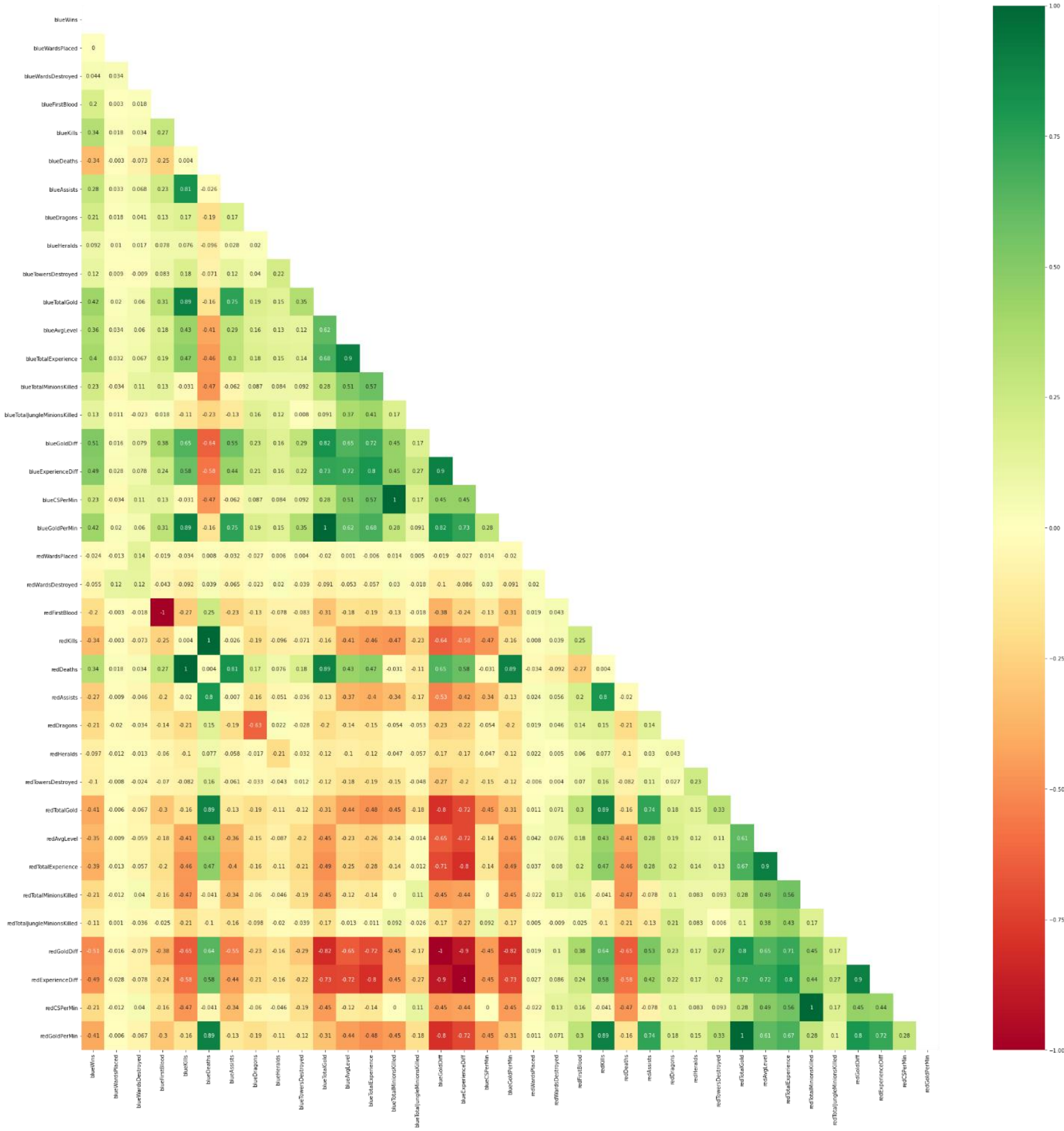
Luckily, our dataset was tidy, uniform and had no missing values.

```
# Let's check for missing values by column and mean
missing = pd.concat([df_pre.isnull().sum(), 100 * df_pre.mean()], axis=1)
missing.columns=['count', '%']
missing.sort_values(by='count', ascending=False)
```

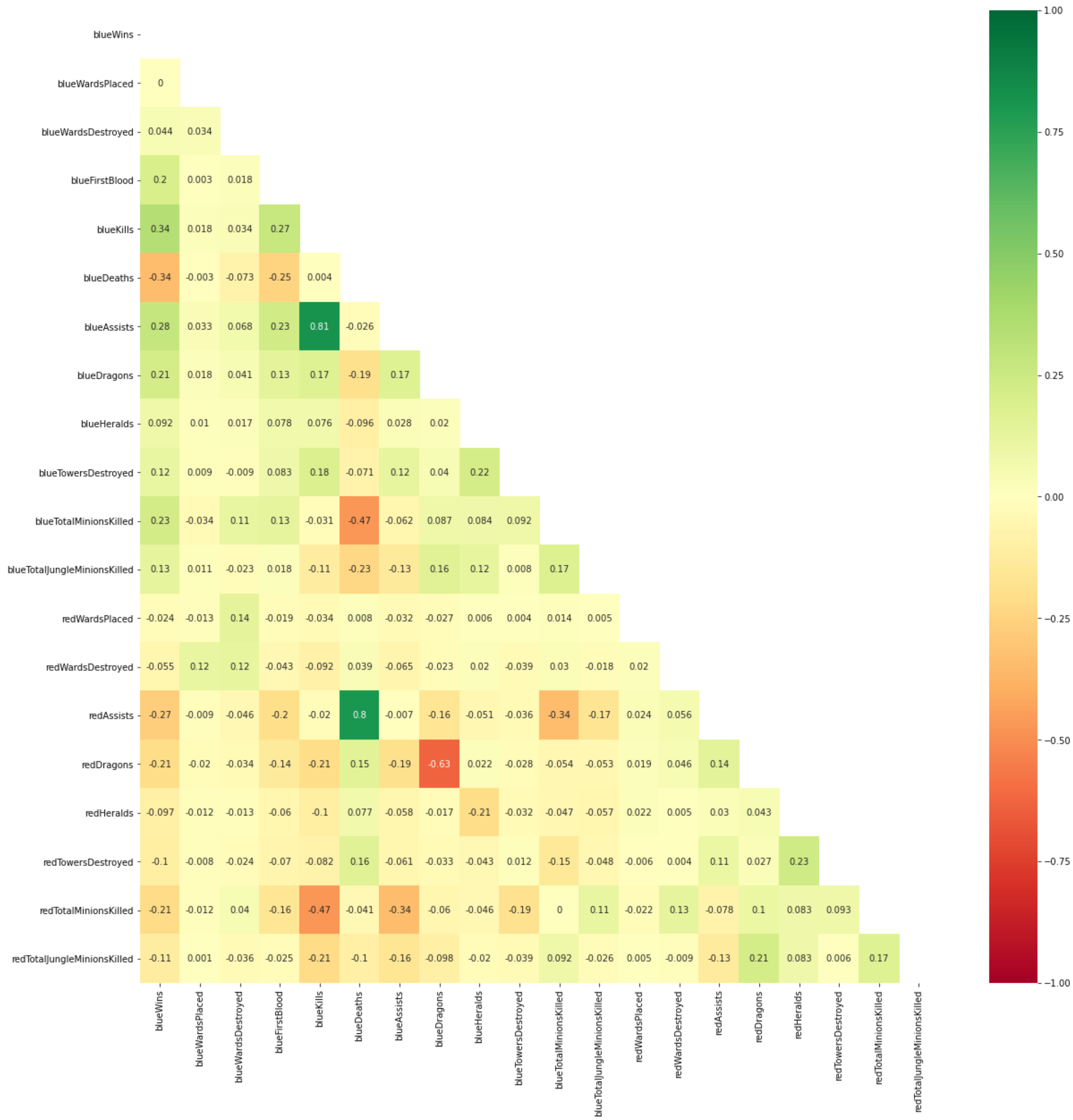
	count	%
blueWins	0	49.903836
redWardsPlaced	0	2236.795222
redTotalMinionsKilled	0	21734.922563
redTowersDestroyed	0	4.302055
redHeralds	0	16.003644
redDragons	0	41.309849
redAssists	0	666.211155
redDeaths	0	618.392550
redKills	0	613.766576
redFirstBlood	0	49.519182
redWardsDestroyed	0	272.315012
blueTotalJungleMinionsKilled	0	5050.966697
blueWardsPlaced	0	2228.828829
blueTotalMinionsKilled	0	21669.956473
blueTowersDestroyed	0	5.142221
blueHeralds	0	18.797449
blueDragons	0	36.197996
blueAssists	0	664.510578
blueDeaths	0	613.766576
blueKills	0	618.392550
blueFirstBlood	0	50.480818
blueWardsDestroyed	0	282.488106

Exploratory Data Analysis

I explored the data a bit further to try and identify redundant features by checking if there is a high level of multicollinearity.



After, checking for multicollinearity we saw several features with high multicollinearity, and some are inverse. I decided to drop those columns that had over a .75 level of collinearity. The features that we'll test on will be the ones below:



We went from having 40 features to now testing our dataset on about 20 features. We still see a high level of collinearity with ‘redAssist’ and ‘blueDeaths’ and ‘blueAssists’ and ‘blueKills’. The reason why I decided to keep those four features is because when the red team kills the blue team more than one player can be involved in the killing of that blue player but that doesn’t always happen. Keeping these features will allow us to answer the following question. Does funneling a large portion of the resources lead to victory vs having an even level team win more?

Feature Engineering & Modeling

The dataset was split into a train and test set with an 80%-20% partition. The feature engineering and modeling process is performed in the Scikit-learn environment. The data preprocessing was done using Standardization. This was used to keep our data is internally consistent; that is, each data type has the same content and format. Standardized values are useful for tracking data that isn't easy to compare otherwise. Some of our features had values in the hundreds vs towers whose max value can be 11 per team. Note, the target variable was not standardized.

blueTotalMinionsKilled	blueTotalJungleMinionsKilled	redWardsPlaced	redWardsDestroyed	redAssists	redDragons	redHeralds	redTowersDestroyed	redTotalMinionsKille
195	36	15	6	8	0	0	0	19
174	43	12	1	2	1	1	1	24
186	46	15	3	14	0	0	0	20
201	55	15	2	10	0	0	0	23
210	57	17	2	7	1	0	0	22

Since I used a Random Forest model a non-standardized dataset was used since it Random Forest model don't require numerical precision.

The trained classifiers are:

- Basic Logistic Regression
- Logistic Regression with GridSearchCV
- Logistic Regression with RandomSearchCV
- Random Forest
- XGBRF
- XGBRF with GridSearchCV

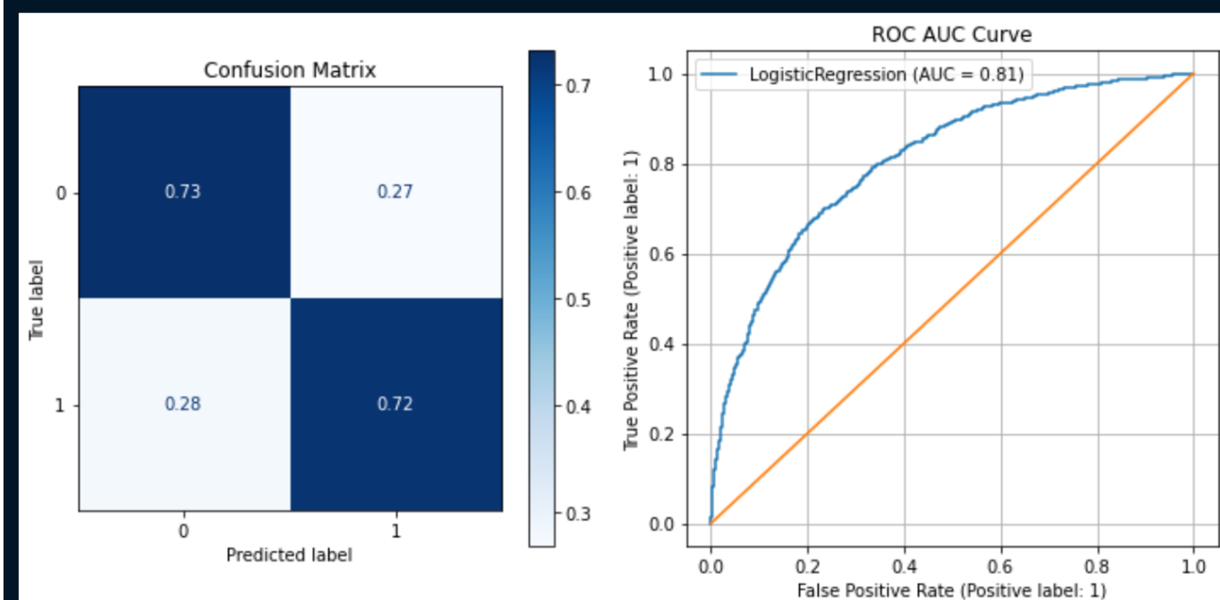
Best Performing Model

The metric of interest is the area under the curve (AUC) of the Receiver Operation Characteristic (ROC) curve with baseline performance of dummy classifier (random chance) of 50%. Tunning and overfitting were addressed by searching the best parameters through both random search with cross validation and grid search with random validation. Due to limitations in computational power and time constraints only 5 folds were used for CV.

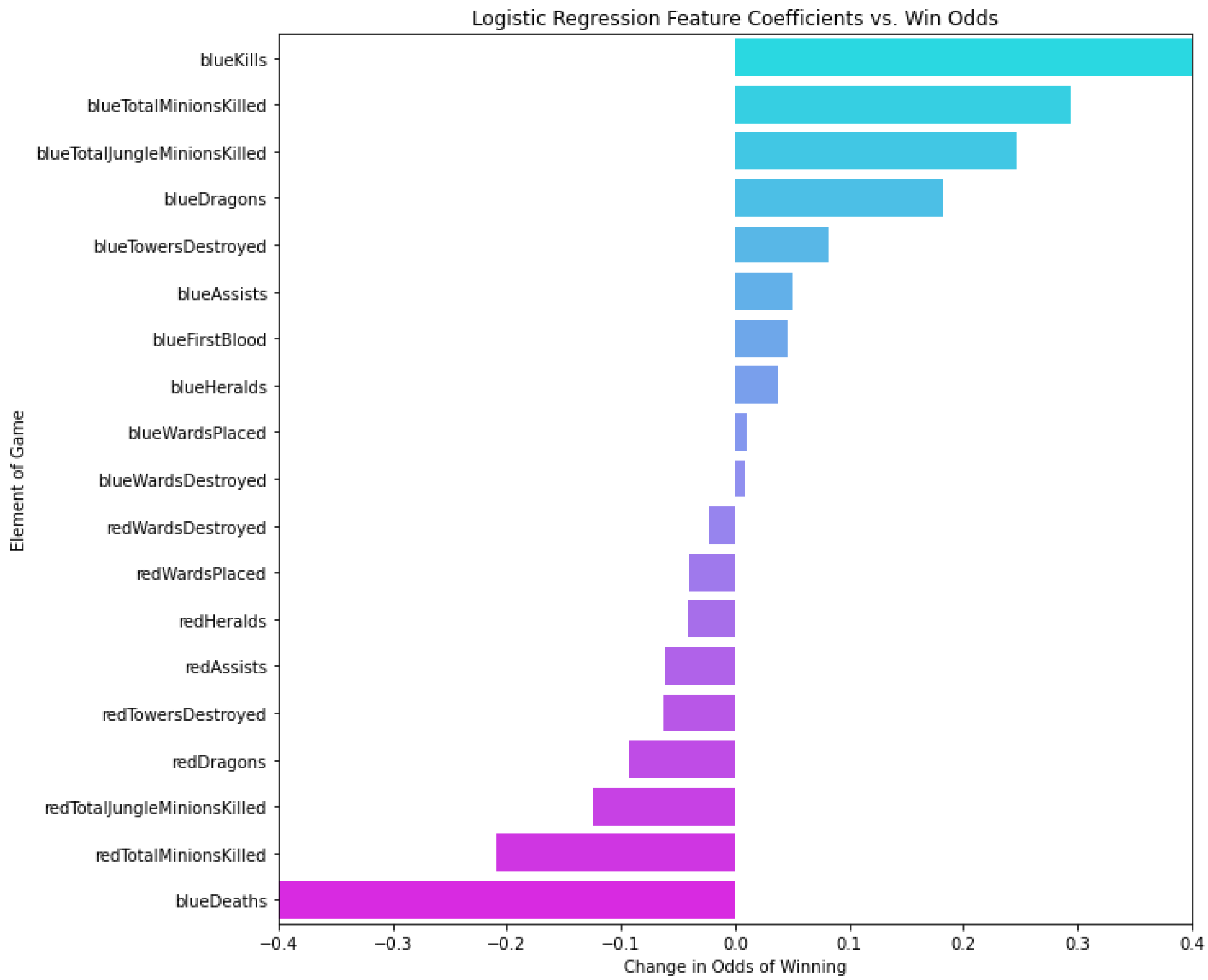
The ROC AUC results of the best performing model is:

Basic Logistic Regression

```
****CLASSIFICATION REPORT - TRAINING DATA****
              precision    recall  f1-score   support

           0     0.7184    0.7145    0.7165      3671
           1     0.7211    0.7250    0.7231      3738

    accuracy                         0.7198      7409
   macro avg     0.7198    0.7198    0.7198      7409
weighted avg     0.7198    0.7198    0.7198      7409

****CLASSIFICATION REPORT - TEST DATA****
              precision    recall  f1-score   support

           0     0.7383    0.7308    0.7346      1278
           1     0.7145    0.7223    0.7184      1192

    accuracy                         0.7267      2470
   macro avg     0.7264    0.7266    0.7265      2470
weighted avg     0.7268    0.7267    0.7268      2470

****CONFUSION MATRIX AND ROC-AUC VISUALIZATION****
```

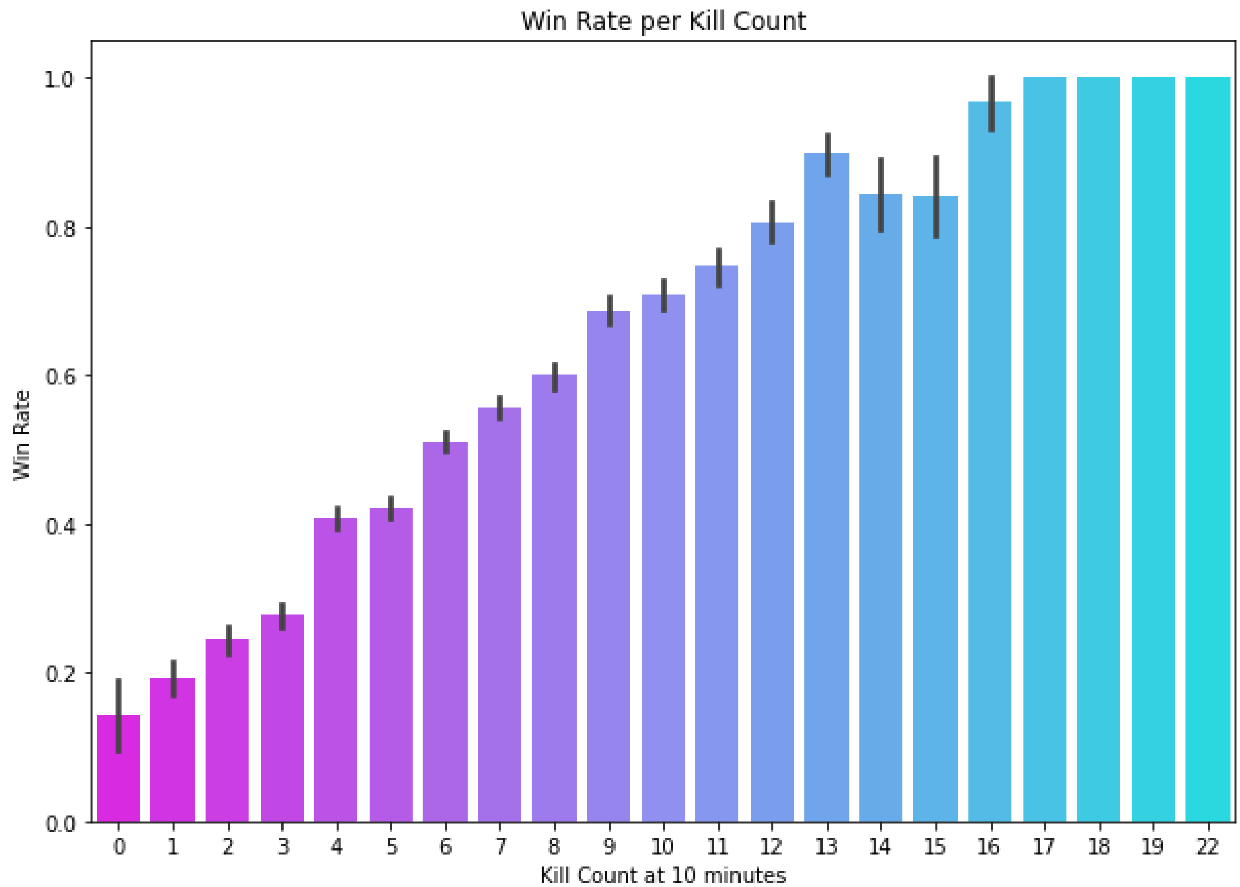


Feature Importance



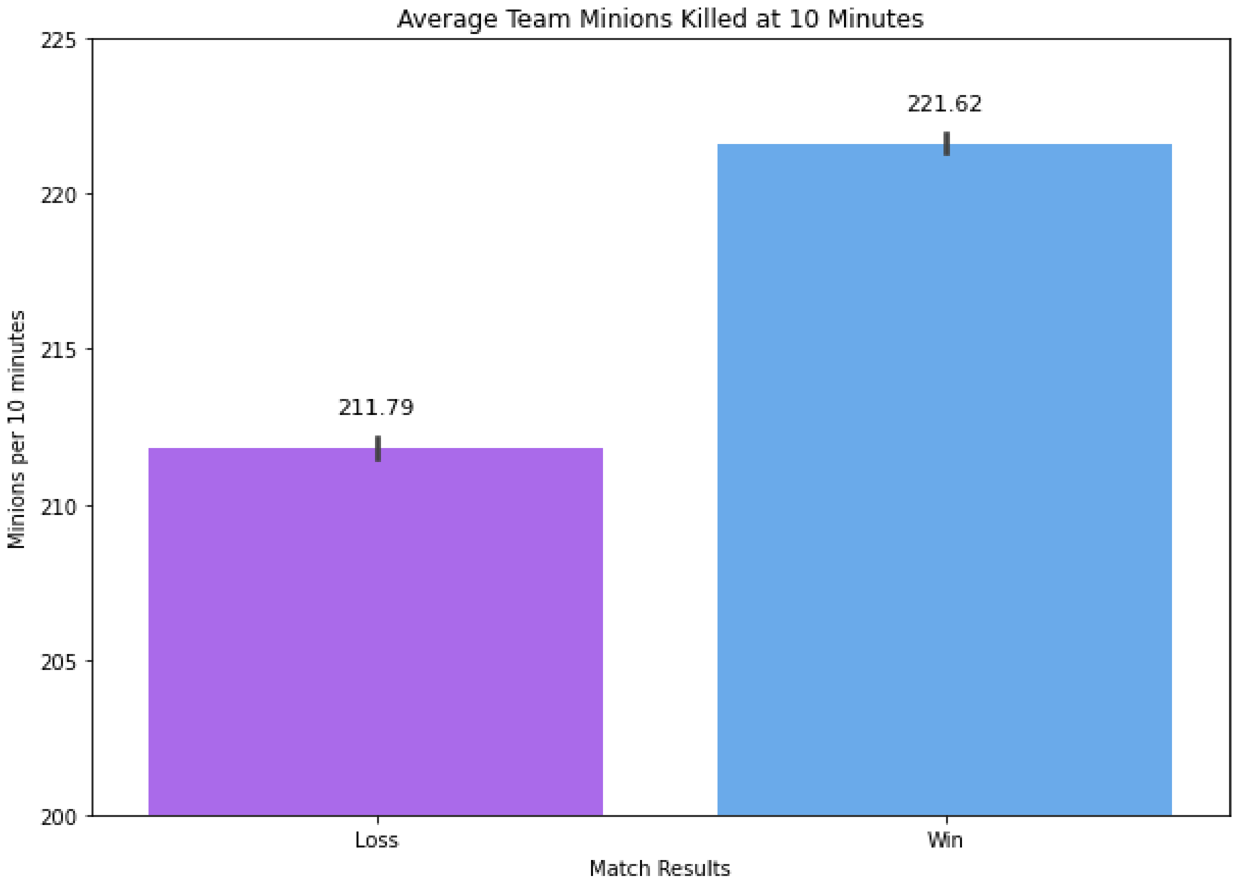
First, the most predictive features are blueKills, blueTotalMinionsKilled, and blueTotaljungleKilled. Let's take a deeper look into those features.

Killing enemy team (blueKills)



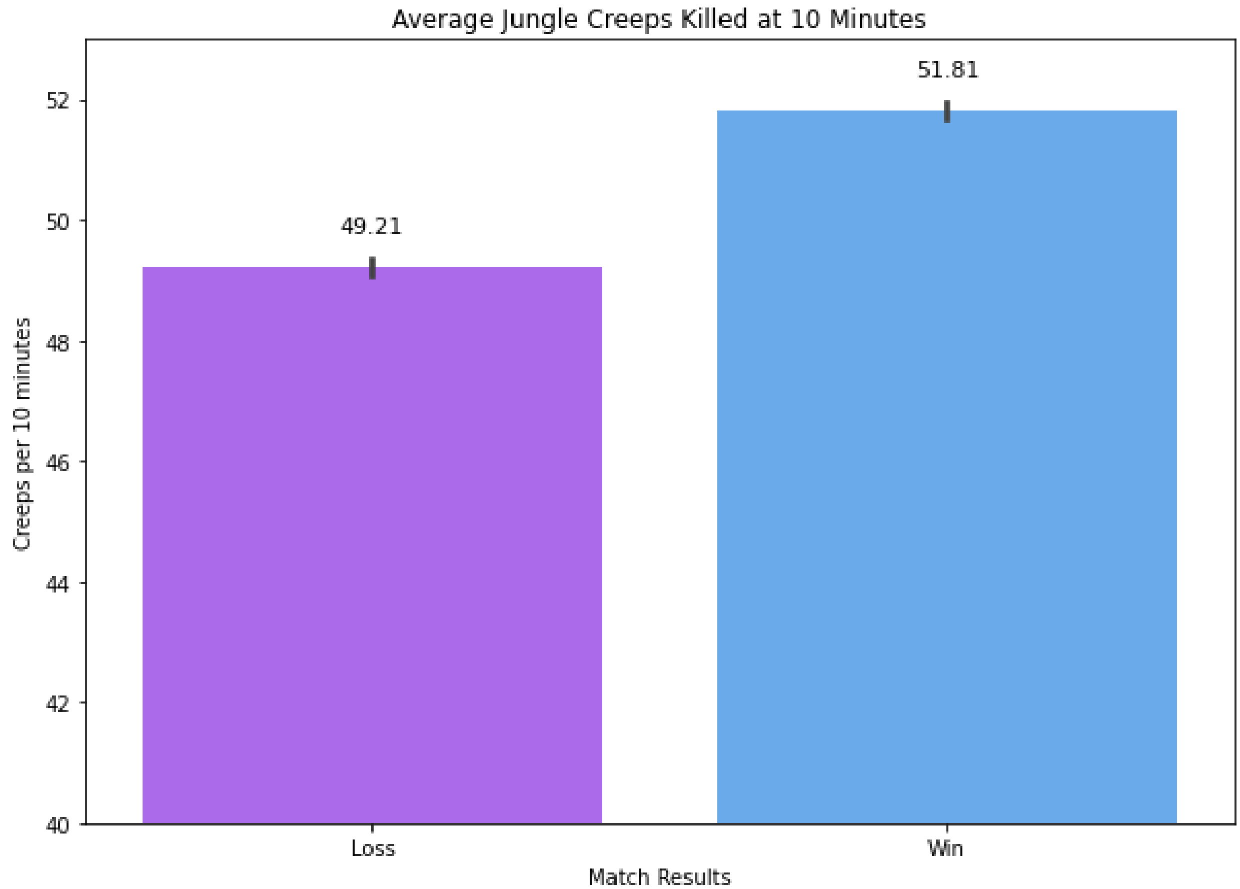
It is easy to identify that there is a very high correlation between winning and the number of kills a team has. Since we are looking to increase our win rate by at least 1% we recommend that shooting for 7 kills puts us over that 51%.

Total Lane Minions Killed



We can see that the difference between a loss and a win measuring by minions killed at the 10 min mark is about a 10 minions difference which adds up to approximately 818 gold. We should aim to farm as a team at or above 222 minions to maximize our chance at winning the game.

Total Jungle Minions Killed



Not sure why our Logistic Regression model identified this feature as one of the most influential when it comes to winning a game of LoL. This finding leaves more questions unanswered than questions answered. This can be tied to how influential the Jungle role is in the current patch.

Conclusions & Recommendations

Based on the above findings, we can see that gold, experience, and dragons kills the highest impact on the outcome of a LoL game.

My primary recommendation would be to focus heavily on gaining gold by having all the laners (top, mid, and adc) focus on last hitting minions since there are a total of 107 minions that spawn per lane within the first 10mins of the match.

My second recommendation is to focus on staying in lane and try to absorb as much XP as possible while avoiding getting killed.

Lastly, killings dragons will impact the possibility of winning. We can do this by warding the river and sweeping for wards before the dragon spawns.

Some considerations for further analysis would include:

- Analyzing data collected at the end of each match to identify what elements of the game led to a quicker vs slower victory.

- Collect data on a specific team to identify what areas need to be targeted.