League of Legends Prediction

By https://github.com/dungwoong

Table of Contents

Intro

This project analyzes data relating to matches of the video game, League of Legends. The data was collected using a combination of web scraping and API calls. The intent of this collection was to predict the outcome of single games of League of Legends based on player-related metrics that can be obtained before the game starts.

Through exploratory data analysis, processing missing values and creating new variables, I was able to generate insightful visualizations and train a classification model that predicts the outcome of high-ranking League of Legends games with ~82% accuracy.

Background: What is League of Legends?

League of Legends is a strategy game where teams of five players face each other and compete to destroy each other's base. Here is some key terminology that is used in the analysis.

Role/Champion. There are five roles that a player can play, and one is assigned to each player. These roles are: top, mid, jungle, bottom and support(as the game map has 3 lanes that are surrounded by a jungle region). Each player can play one of ~150 characters in the game, known as champions.

Winrate: a player's win rate is simply calculated by dividing his total wins of a certain category by his total games played of that category. For example, a player's role winrate would measure the winrate of all games played in a certain role.

Tier, Rank and LP: The league of legends ranked system uses a division system. I recorded games played among higher tiered players, so the tiers(in order) are: Diamond, Master, Grandmaster, Challenger. Diamond has divisions 4 through 1(with lower numbers indicating higher divisions), and master/grandmaster/challenger uses a League Point(LP) system, where higher LP implies higher ranking. Diamond players must attain 100lp to progress to the next diamond rank, and master/grandmaster/challenger is awarded to players within a certain percentile of the top players' LP.

After a single round of League of Legends, the winning team will gain LP, and the losing team will lose LP. Players must play multiple rounds in order to enter the ranked ladder, basically.
Other variables will be explained later in the report.

Data Collection: Methods

Code for data collection can be found in the src folder.

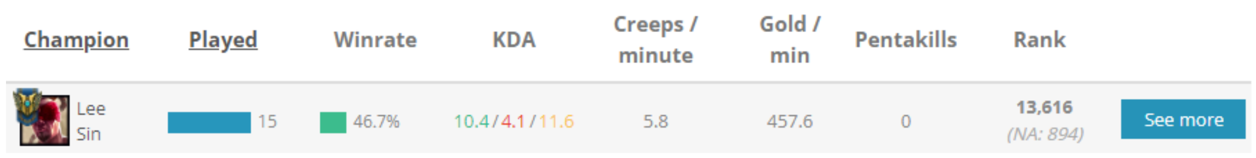
I collected observations for single League matches from the official Riot Games API, and the website League of Graphs, that calculates more informative player metrics.

Prior to data collection, I consulted some articles analyzing the most important factors of League Games, and a Twitter thread about a project similar to this one.

From the articles, I found that a player's kill/death ratio, or kill participation was highly correlated to the outcome of a game. Vision score, obtained from gaining vision of the enemy team, was also important. Destroying enemy objectives is naturally important as well. These, among other things, are a great way of predicting a game's outcome after the fact. Since I will not have access to this data before a game starts, I decided to take each player's last 10 ranked games and measure similar statistics, as they may be indicative of future performance.

From the Twitter thread, I found the League of Graphs website, and learned about their aggregated metrics. The site contains statistics for champion and role win rates as it records match data over time.

These metrics can only be otherwise calculated by aggregating all of a player's match data by making copious amounts of API calls and lots of time. Thus, this website is really helpful in creating new, informative variables.
From this website, I recorded the role and champion win rates for each player on each team using web scraping in BeautifulSoup.



Champion	Played	Winrate	KDA	Creeps / minute	Gold / min	Pentakills	Rank	
Lee Sin	15	46.7%	10.4 / 4.1 / 11.6	5.8	457.6	0	13,616 (NA: 894)	See more

An example of champion win rate on LeagueOfGraphs

Using a combination of API calls and web scraping, I collected metrics related to a player, such as their overall, champion and role win rates, and statistics related to their last 10 games. I then exported it to a spreadsheet, which can be found here(with approximate column documentation found here. I measured over 200 variables for over 2000 games, meaning around 10 variables were measured for each player on each team.

Note: one possible flaw in the data collection is how new game ids were obtained for data collection. For each player, I took their last 5 games, and took the last 5 games of each player in those games, etc. to get the game ids that data was collected for. Although I made sure game ids were unique before data was collected, this means I have multiple games containing the same player. I did, however, refresh the collector often, starting the id collection from a completely different game. I will discuss this later.

Data Analysis: Missing value basic exploration

Upon basic exploration of the data, it was noted that there are many variables that could be aggregated(eg. Average champion win rate could be created from individual champion win rate), and there was a lot of missing values, especially in champion and role winrate.

This is not good for the analysis, as those variables have been shown in previous analyses to be highly correlated with game outcomes.

Variables were recorded in the format <team>_<role>_<variable>, where the teams were numbered 100 and 200. For example: 100_TOP_champ_winrate.

After looking through all the missing values, a few variables(across each role in each team) were missing values. Since I understand the origin of the data, I could somewhat explain why each variable may be missing.

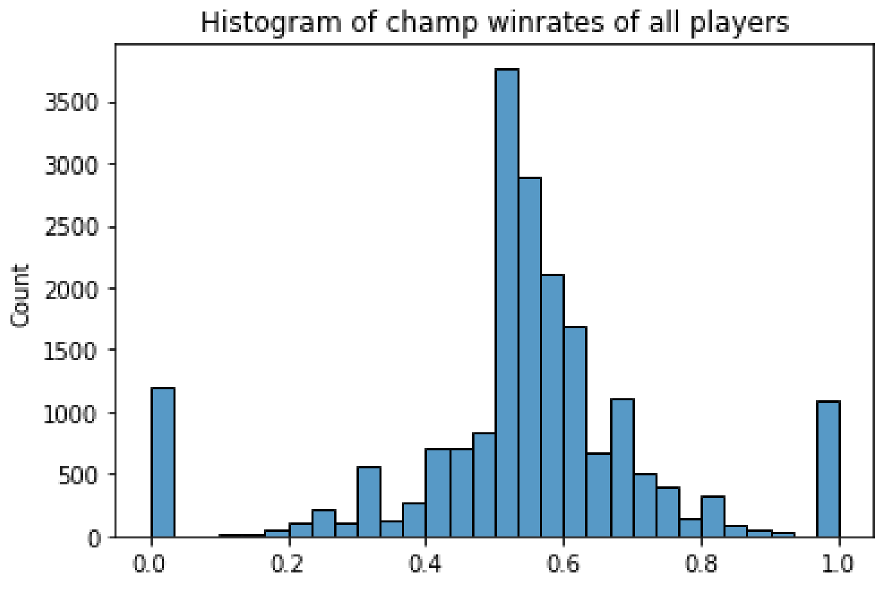
Variable	Reason for missing

champ_winrate	Player has not played said champion yet, therefore has no data shown on LeagueOfGraphs
losses	Player has not played ranked yet, has no losses
wins	See losses
role_total_played	Player has not played said role yet, therefore has no data on LeagueOfGraphs
role_wr(winrate)	See above
Rank, tier, freshblood, inactive, veteran, hotstreak lp	These variables measure a player's ranked performance(eg. Hotstreak means a player has won 3+ games in a row), and are missing because the player has no ranked info yet.

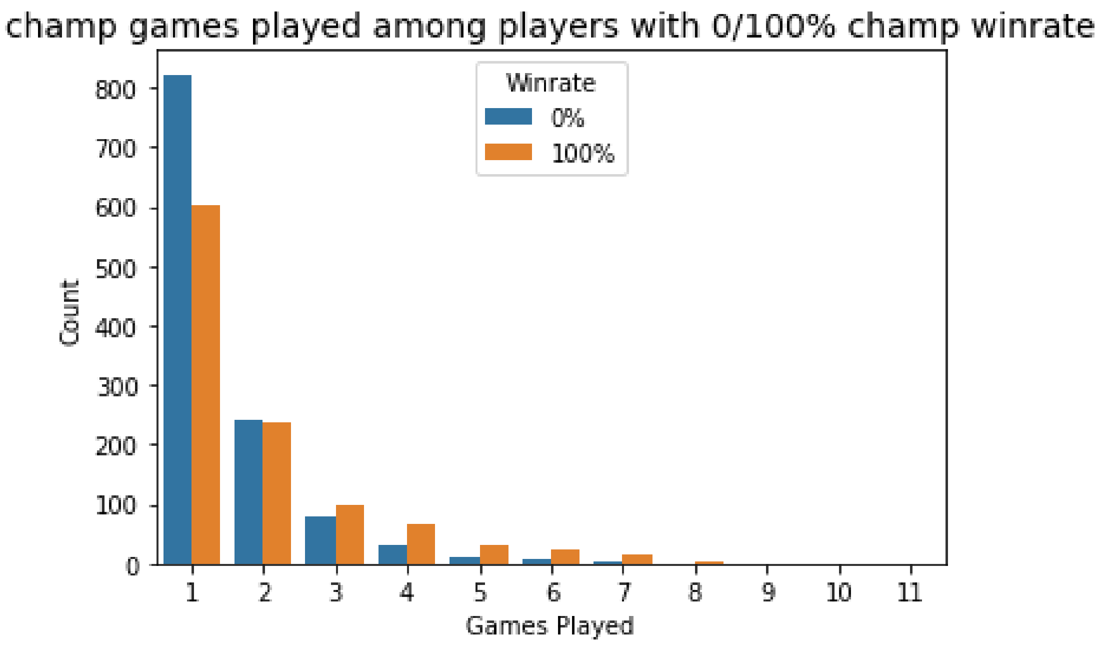
Data Analysis: Further exploration of missing values

Champ Win Rate

In addition to missing values, there were a large number of 0 and 1 values recorded for champ_winrate among all players on both teams



Upon further analysis, it was found that this was due to a low number of games played by the player. Players with 100% champion win rate were more likely to have played numerous games.



Since we collect game data after a game occurs, if a player has only played one game on a champion, their champion win rate will correspond to the outcome of that singular game they played. If a player only played a few games with their champion, their champion win rate probably has low correlation with their actual skill on their champion.

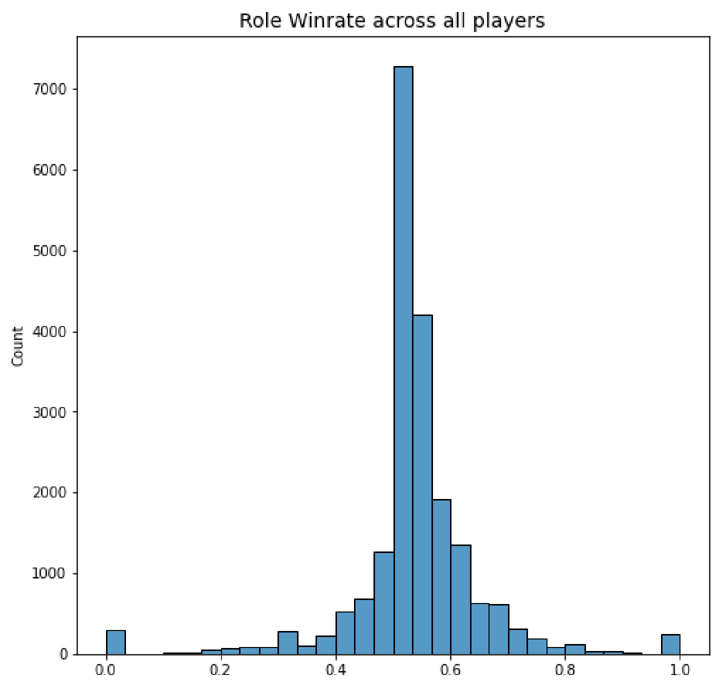
Ultimately, I decided to substitute players with < 3 games played with the median champion win rate, and keep the other values as is. This is due to the following reasons:

- Since the amount of players with 100/0% win rate and >3 games played in around 15%, most of the observations that occur due to luck, or recording winrate after a game, is corrected
- In higher ranks, it is difficult to play a champion for the first time and do good. Players are more consistent, and better players that are experienced on their champions will consistently outperform worse players(this is partly supported by data, and partly from my own experience watching high ranking games). Thus, having a 100% win rate across 3 or more games should be kept in the data.
- This may also help identify “smurfs,” experienced players who try to rank up quickly on a brand new account, and therefore have high winrates.

I then filled missing values with the median champ winrate.

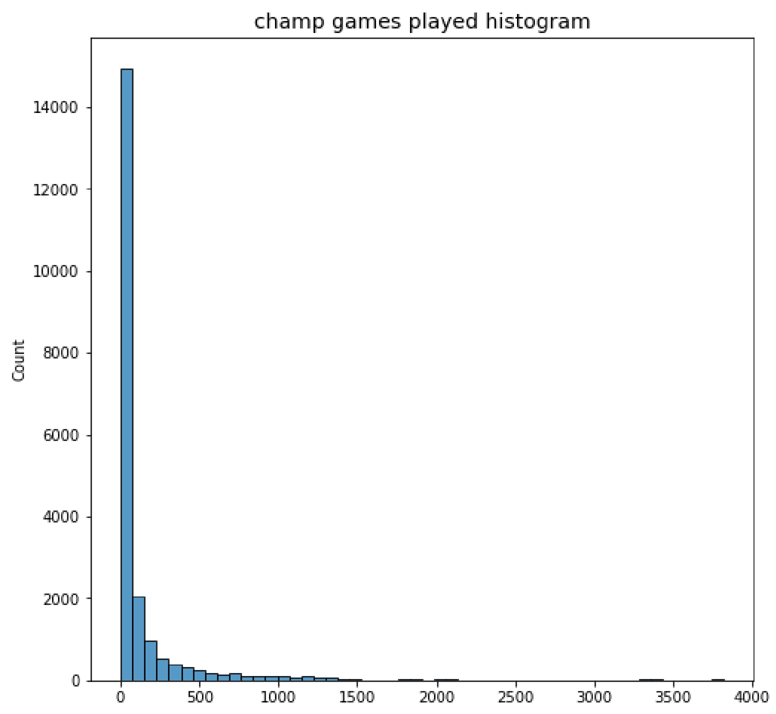
Role winrate, role total played

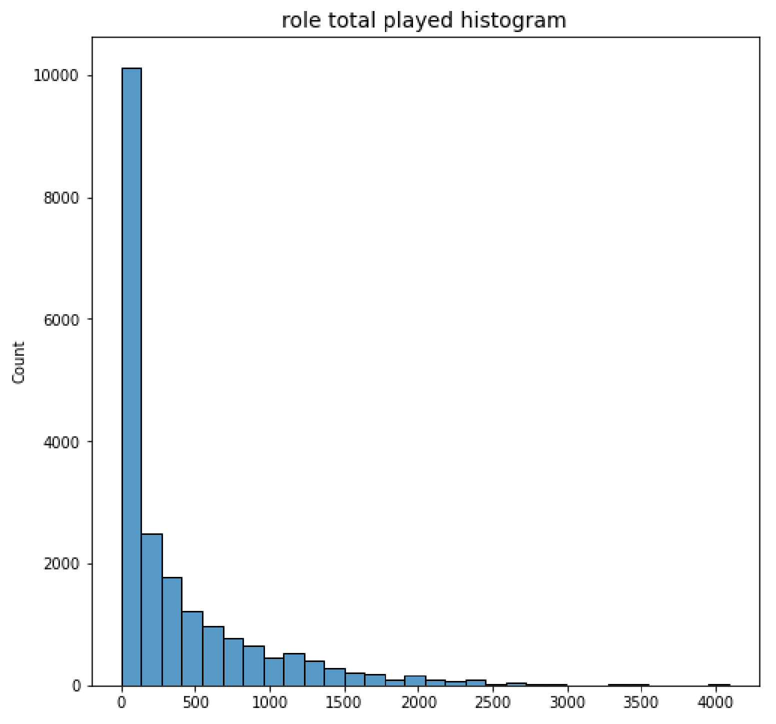
There are only 5 roles in the game, whereas there are over 158 champions. The likelihood that a player has only played a role a few times is lower. Therefore, we see a more even distribution of role winrate.



From the histograms below, it's clear that overall, players have played many more games in a role compared to on a specific champion. Thus, I decided not do do anything with the outliers, and to simply fill the missing values with the median(which is around 50%).

I also considered filling missing values with 0, but that makes no sense, as the probability of winning a game is typically around 50%(as one of the two teams must win)





Other variables

Other variables were filled logically. For example, a missing value of LP means that a player has not played ranked yet, and therefore has no LP.

A missing value of “veteran”(indicating if a player has played a lot of games without changing division) is caused when a player hasn’t played ranked, and should logically be set to “false.”

The notebook contains the code and goes into more detail about filling missing values.

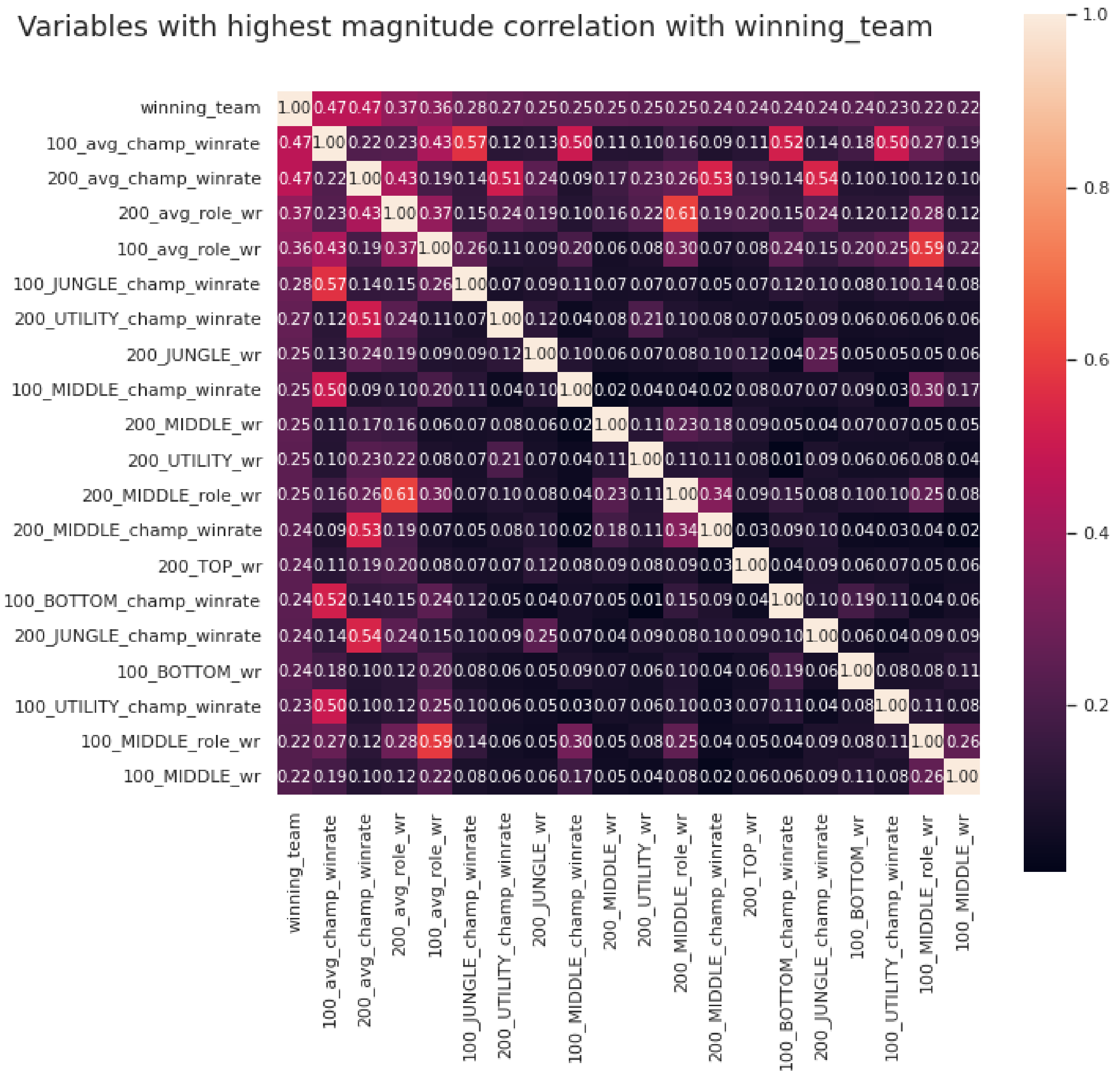
Making new variables

I also made a few new variables, namely average champion winrate and average role winrate for both teams.

Data Analysis: Visualization

Correlation Heatmap

After cleaning the data and making new variables, I plotted a heatmap of the variables with highest magnitude correlation with the winning_team target variable.
Thus, the variables with strong positive and negative correlation were plotted.



This plot shows that Twitter was correct, champion winrates are highly correlated with the target variable. Among those, team average champion and role winrates have exceptionally high correlation(compared to the other)

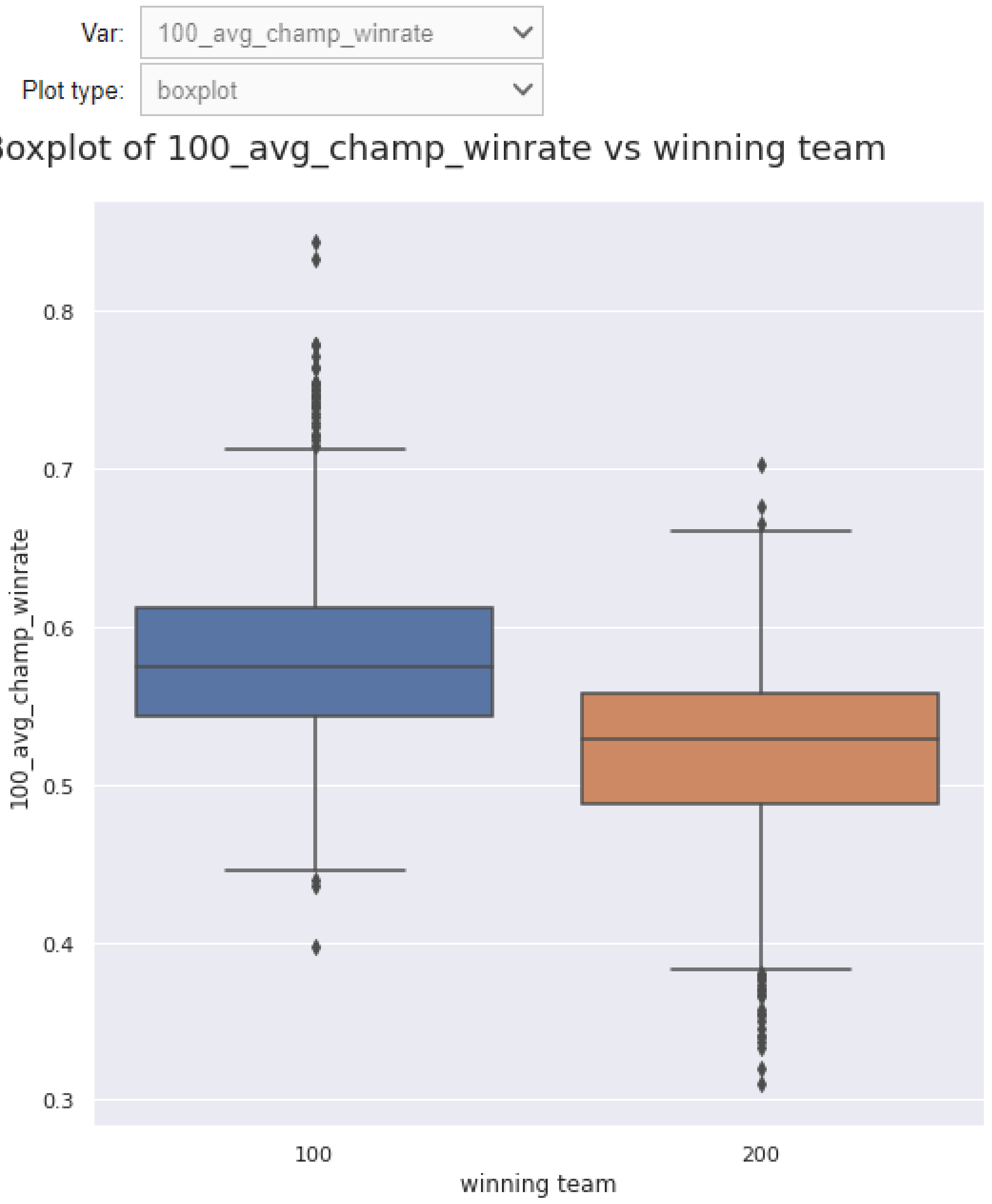
However, there is no variable with correlation magnitude > 0.5, which means that, overall, variables do not have a very strong correlation with the target. That is unfortunate.

A lot of variables are also somewhat correlated with each other. For example, individual champ win rates are correlated with average champ winrates, and individual variables are correlated with each other.

The correlation is not that high, and the logic of the correlation implies that we should not discard the variables. For example, one role may have more impact on the game outcome compared to the other ones, so the individual winrates still produce useful information, despite being highly correlated with the team average.

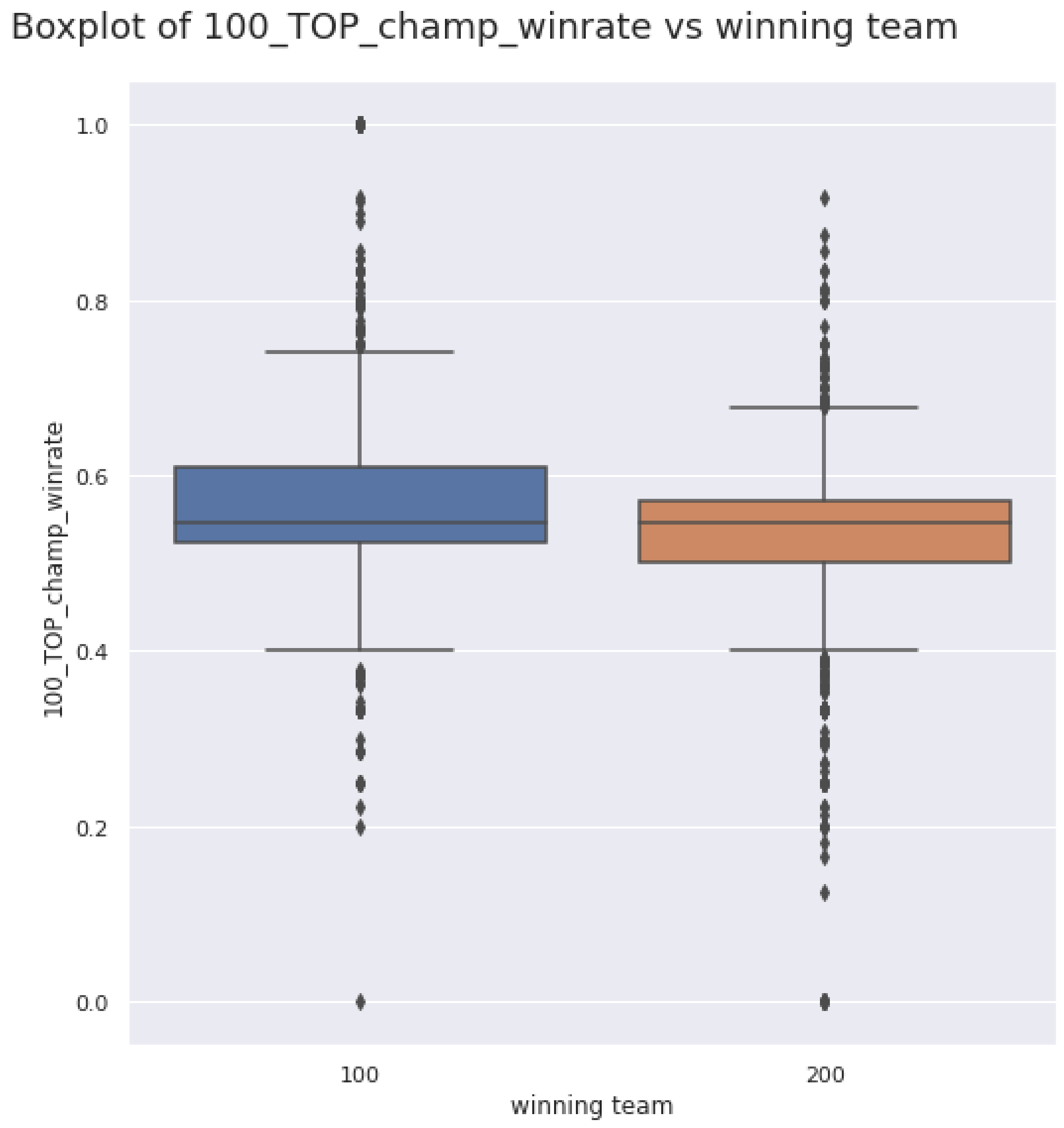
Average Champ Winrate Boxplot

On the jupyter notebook, there is a widget that can be used if it is uploaded to Google Colaboratory, that allows the user to alter the following visualization's variables, and such.


Var: 100_avg_champ_winrate
Plot type: boxplot

As shown here, the team 100 average champion win rate is typically quite higher when the winning team is team 100, compared to when team 200 wins. Also, outliers are typically above the median value when team 100 wins, and below the median value when team 200 wins.

For individual champion win rates, the boxplots are similar but are slightly closer together, and have a smaller Inter-Quartile Range.



Role winrate yielded similar plots, but were also closer together and had narrower IQRs. This supports the idea that they had a lower correlation magnitude with the target variable.

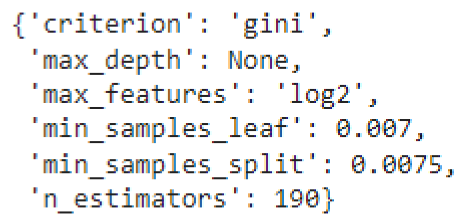
Random Forest Classification using Randomized/Grid Search Cross Validation

The code on the notebook is more informative regarding how the model was created.

Basically, I separated the data into training and testing datasets, then trained my model. I used Scikit-Learn's RandomizedSearchCV to randomly search different parameter combinations of a random forest that attempted to fit my training data.

Using the parameters of the model with the lowest ROC AUC score, I created a set of parameters that were similar to the best parameters from the Randomized Search. I then used Grid Search(which searches all possible combinations of parameters that are inputted, opposed to randomized search, which searches a set amount of random combinations) to look for better parameter combinations.

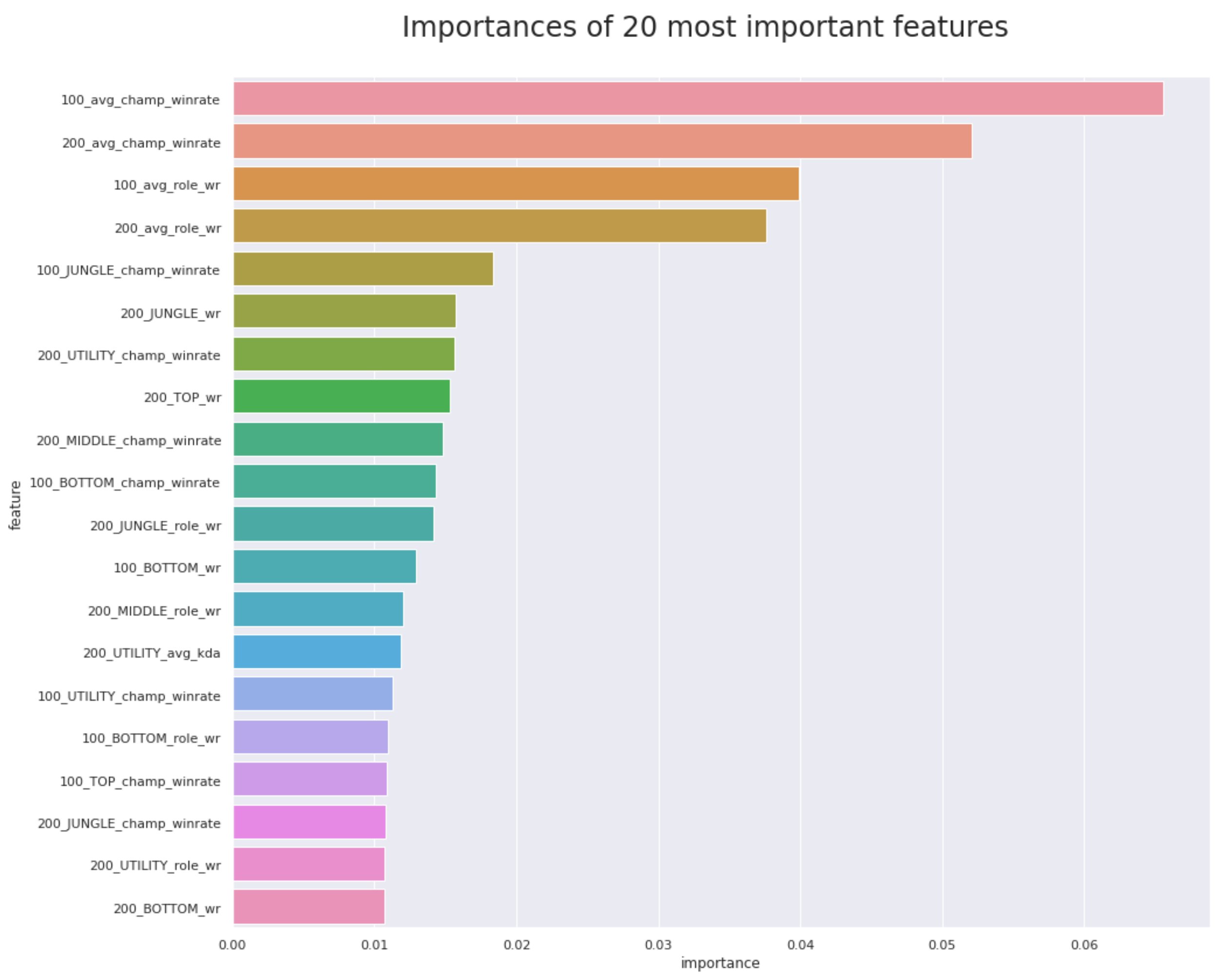
Finally, I took the best parameters from the Grid Search and used them to train a final model on the entire training dataset. The parameters I used are as follows:



```
{'criterion': 'gini',
 'max_depth': None,
 'max_features': 'log2',
 'min_samples_leaf': 0.007,
 'min_samples_split': 0.0075,
 'n_estimators': 190}
```

The AUC score of the final model was 0.813, and the test accuracy of the classifier was 0.815. The model has around an 80% true positive and true negative rate.

On the next page are the features that the model used, ordered by importance:


Importances of 20 most important features
100_avg_champ_winrate
200_avg_champ_winrate
100_avg_role_wr
200_avg_role_wr
100_JUNGLE_champ_winrate
200_JUNGLE_wr
200_UTILITY_champ_winrate
200_TOP_wr
200_MIDDLE_champ_winrate
100_BOTTOM_champ_winrate
200_JUNGLE_role_wr
100_BOTTOM_wr
200_MIDDLE_role_wr
200_UTILITY_avg_kda
100_UTILITY_champ_winrate
100_BOTTOM_role_wr
100_TOP_champ_winrate
200_JUNGLE_champ_winrate
200_UTILITY_role_wr
200_BOTTOM_wr
feature
0.00
0.01
0.02
0.03
0.04
0.05
0.06
importance

Final Thoughts

What worked well:
I think my initial research before data collection, and careful reasoning and data exploration were successful, as the new variables that I created turned out to be highly correlated with the target.

Although the model would have been more accurate(probably) if I didn't correct champ win rates of 0 or 1, it would be less accurate when applied to live games, as champ_winrate would not be distributed the same way when collected before a game begins. For this reason, I feel like my decision improved the model's real-world applicability.

Areas for improvement:
I think I should have thought more about the method by which I collected my data. I basically recorded 5-10 games for every player in every game I saw, and kind of recursively collected game ids. Thus, I may be collecting game ids for a subset of the population of high ranking league players. Although I did vary the games, constantly starting my id collector from a different high ranked player, I think I could have considered other methods of searching.

For example, if I simply took the top few players and took their past games, and went down the ranked list, rather than recursively collecting IDs. This may have broadened the subset of the challenger population that I was collecting data from.

I could also have considered collecting more data. This would allow me to discard data with missing values, rather than filling them with the median/mean of the entire column.

I could also consider making new variables, or exploring existing ones further. Aggregation seemed to extract new insights from existing data, so I could have pursued that further.

I am only a second year student, and I believe I used my current knowledge of statistics and data science to produce a rather decent project.
I will probably come back to this project in the future, with more knowledge and insights.