A League of Data: Analyzing the choices and skills that allow League of Legends Players to Win Games

Mark JP Sanchez

21/04/2022

Introduction

League of Legends is a hugely popular video game that rakes in billions of dollars a year and is easily one of the most popular and influential games in the world. The success of the game has even spawned a successful e-sports scene, a k-pop band, and even a critically acclaimed Netflix show. All of this, spawning from an online 5 vs. 5 competitive MOBA game. A MOBA game effectively being a game such that two teams face off head-to-head in a match. Each player chooses a character, in League of Legends they are called champions, to play as and each character has a unique set of skills. During the game, each player attempts to level up their character, accrue money to spend on items, destroy enemy towers and escort their minions to the enemy base. The objective of the game is to destroy your opponent's base but to do so, you need to power up and work with your team to both infiltrate their base and protect yours over the course of 20 minutes to an hour.

Evidently. one thing that makes the game so much fun is all the moving parts and decisions a player can make to influence a match. Some decisions include what champion to choose, what items to buy, and what objective to focus on. Unfortunately, all of these factors and decisions can overwhelm new and existing players, and with the main objective obviously being to win the game, it would be nice to now what kind of decisions actually influence those wins.

Methods

Data Wrangling and Cleaning Methods

The data set was taken from the RIOT api and contains data from League of Legends matches. The data collected contains info such as how long the matches were, what champions each player played, how much damage each player did, and what side objectives the players took. The data was given in JSON format which I had to turn into nice R data frames.

One issue that arose from gathering match data from the RIOT api was that there was no proper endpoint for gathering random matches. The main way to gather matches was to call an endpoint that gave you a list of past matches of a specific player. The problem with this is that if I collected all of my match data from this list of matches, then I would have a confounding variable being this one player. I want to make my data between matches as independent as possible. Luckily, each match consists of 10 players so I can recursively gather matches of each player in the current match. Using this method, I gathered the my match data by first gathering a few thousand player ids by recursively checking the players in a given player's latest match. I then get the latest match before a previous date of each player. My chosen date was Valentines day which was a week before I gathered my data. There was also a large update to the game on January 7th, 2022 so we need to make sure our matches don't come before then. This should give me a list of matches that has little overlap in players.

Calling the match data api from RIOT gives you a lot of variables to work with. Especially since each player in the match have tens of different variable attached to them. With 10 players per game, that is a lot of data to work with. Since we are looking at the decisions each player makes on their own, we will take the data and split it up into 2 different datasets. The first dataset will contain match metadata such as when the match was created and how long the match was. The second dataset contains data on each player during a match. So each observation in that player dataset will be corresponding to data from a player in a match. This data will include things like how much damage they did, how many kills they got and how much gold they made.

With the methodologies I listed above I ran my code and let the program run for a few hours. This long waiting time was due to the rate limit imposed by the RIOT api of a max of 100 requests per 2 minutes. In the end we collected a dataset of 9400 matches.

Feature Importance Analysis

There are a lot of decisions to make within the game. For new players especially, the number of choices to make can be overwhelming. To help with this issue, we will be looking at what decisions are the most important so that new players and maybe even veteran players can prioritize what skills they need to work on.

To achieve this, we will fit a xgboost model where our response variable is the likelihood that the team will win. We will then check the top most important features and those will be the most influential features/decisions a player can make

Champion Selection Analysis

A big choice each player has to make is which champion they will choose for game. Each team chooses 5 champions to play as and this choice is so important, there is even a phase in the game where you ban certain champions so that your opponent can't play it.

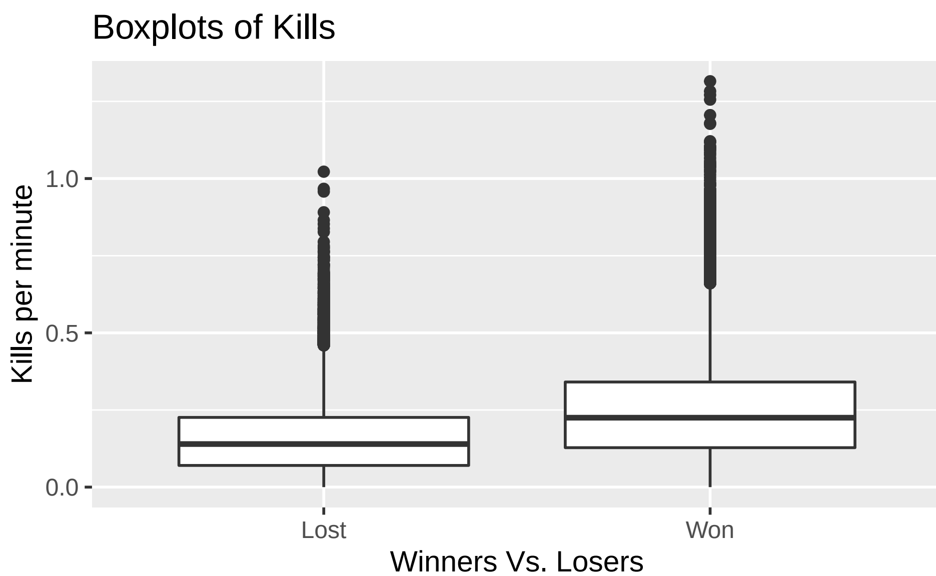
To understand each champion better and each role better, we will perform some matrix factorization. A big part of the game is dealing damage to your opponents. So we will construct a matrix where each row represents one of the 158 champions and each column will represent another champion. The value of the cell will be an important stat that we will discover that the row champion is responsible for in games where the column champion is on the other team. So for example if our row champion is Thresh, our column champion is Vex and our stat is total damage dealt, then the cell will contain the average damage dealt by Thresh in games where Vex is on the opposing team. By looking at the latent factors the factorization generates, we can gather some insight on how the game works and if choosing a champion really matters.

Results

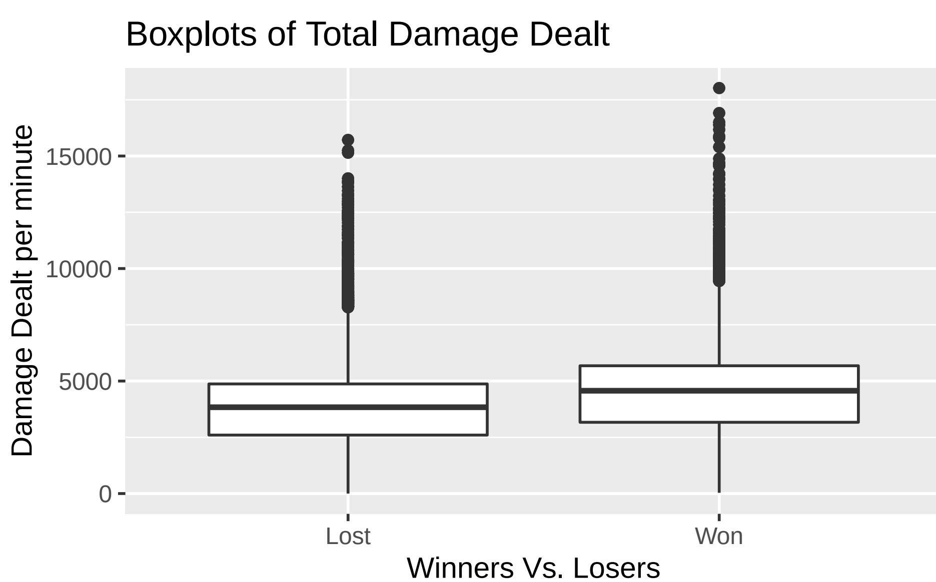
Exploratory Data Analysis

Player Stats and Skills

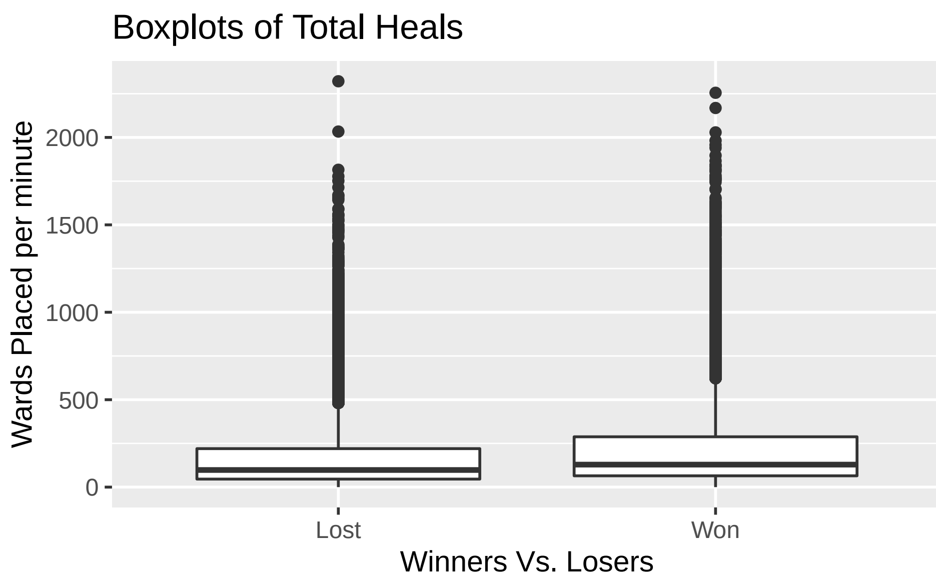
We started by looking at our data to see what kind of decision and skills players who won had. There are a multitude of stats and skills players must polish. Here below, we can see some box plots of some of these stats split by winners and losers. This will help us with understanding what decisions and skills will be useful for our xgboost model.



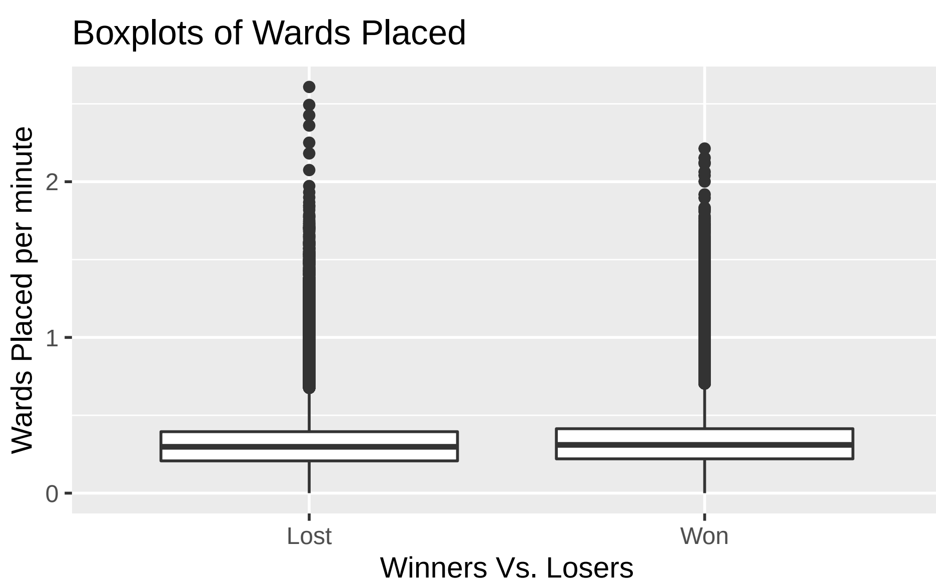
Kills are self explanatory. A kill dictates a player being able to reduce an opposing player to 0 health. Once a player dies, they must wait before being able to aid their team and their killer gets a reward. So it is easy to see that getting more kills would imply that you are winning and above we can see that there is evidence of this relationship.



To get a kill, a player must deal damage. There are also other non-player characters such as minions and towers which a player can deal damage to. Just like kills, there seems to be an effect where dealing more damage increases the likelihood of a player winning.



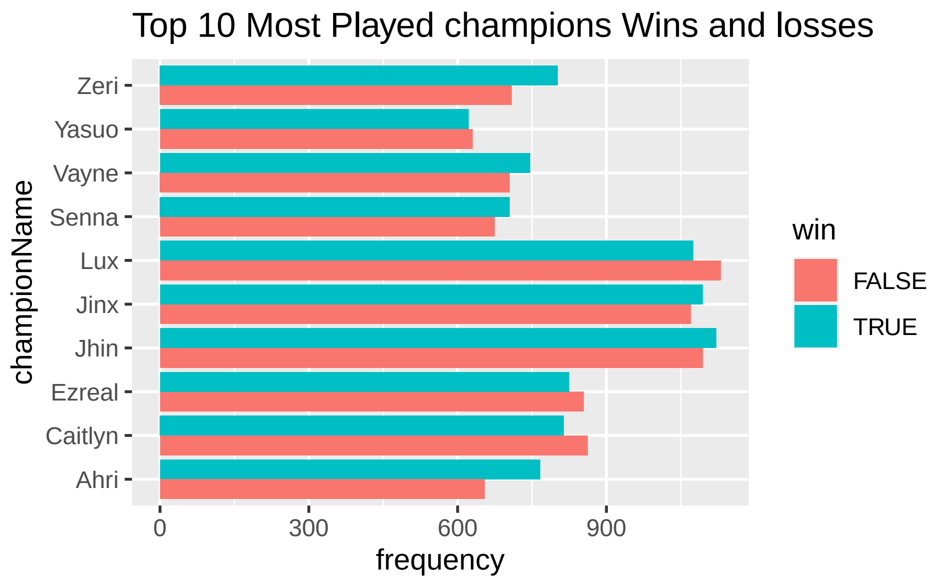
On the flip side of dealing damage is healing. While there seems to be less of a relationship when compared to getting kills, healing more per minute seem to correlate to an increase chance of winning.

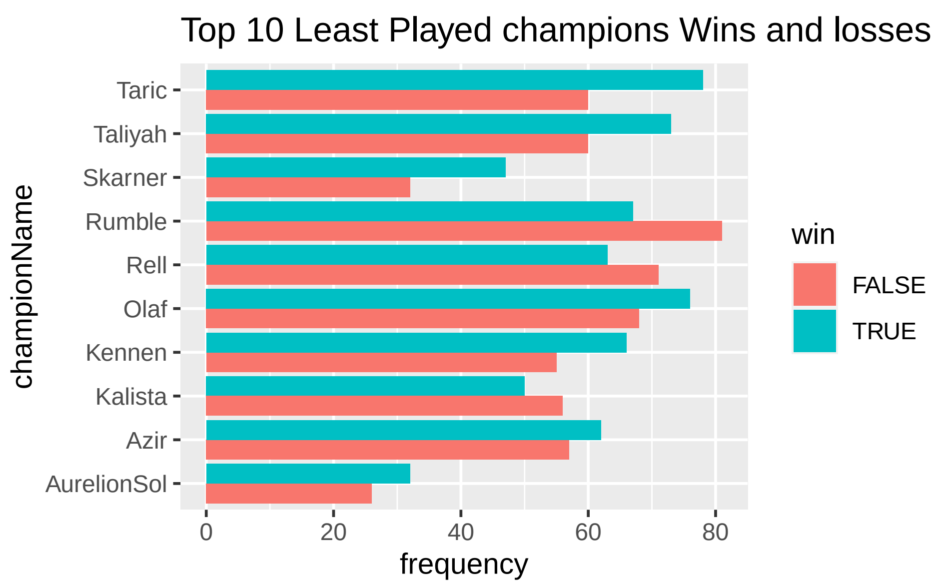


Within the game, there are items called wards which you can buy. These ward provide you vision on the map where you have no other team mates since that area would be covered by a fog of war. As we can see above, there seems to be little correlation between the number of wards you buy and the likelihood of you winning

champion Statistics

Another big decision players have to make is what character or champion they are going to play. Since they can't switch champions in between matches and since each match is around 30 minutes to an hour long, players must choose their champions wisely. Here we will look at some basic info surrounding the champions and the roles that they play. This insight will be useful in motivating us on our procedure with matrix factorization.





Here we can see that the top 10 most popular champions seem to have a smaller win loss disparity than the top 10 least played champions. This might just be because of the smaller number of times the least played champions are played. We can perform some tests to check.

Our test consists of a simple check as to whether a champion has a 50% win rate or not. Our null hypothesis is that the champion has a 50% win rate. Our alternate hypothesis is that the win rate of the champion is not 50%. For this test our p-value threshold will be 0.05/158. This is because there are 158 champions so we would be performing 158 tests. We want a 5% significance threshold but since we are performing so many tests we have a high chance of getting a type 2 error. If we can find a champion that has a higher winrate then right choice to make would be choosing that champion.

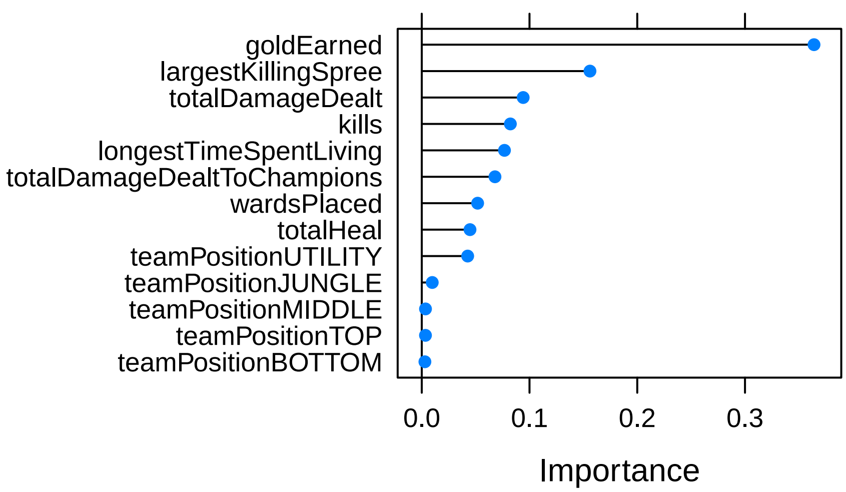
After performing our tests the only champion that strong evidence against the null hypothesis was Vex who had a win rate of 43%. So overall League of Legend champions seem to be balanced from this basic p-value test

Feature Importance Analsys

Just like we said, we will be fitting our data to a xgboost model and then we will be extracting the most important features.

We split our data into a 70/30 train test split with our response variable being the likelihood a player will win. Our predictors will be the position the player is playing, the amount of damage they do in total, the amount of damage they do to other champions, number of kills, largest killing spree, longest time spent living, gold earned, total amount healed and number of wards placed. Since each match varies in length, we will divide each value by the match length to get a value per minute variable.

After performing 5-fold validation for our xgboost model, we arrive with a model that has an accuracy of 73%. In other words, our model shows that the result of a match can be predicted with 73% accuracy by looking at the performance of 1 of the 10 players in the match. So each player has a large influence on the state of the game.



Looking at our variable importance graph we can see that the most influential and important variable is gold earned. Largest killing spree also seems to be placed over kills which is interesting. Lastly, it is worth it to note that the 2 least important variables are number of wards placed and total heals the player has dealt.

The two most important variable actually make a lot of sense. The more gold you earn, the more items you can buys which makes you stronger. If you have a large killing spree then that means that you were able to kill a lot of enemy champions without dying. It shows that you are actively participating in the game while not allowing your opponents to gain any benefit from punishing you.

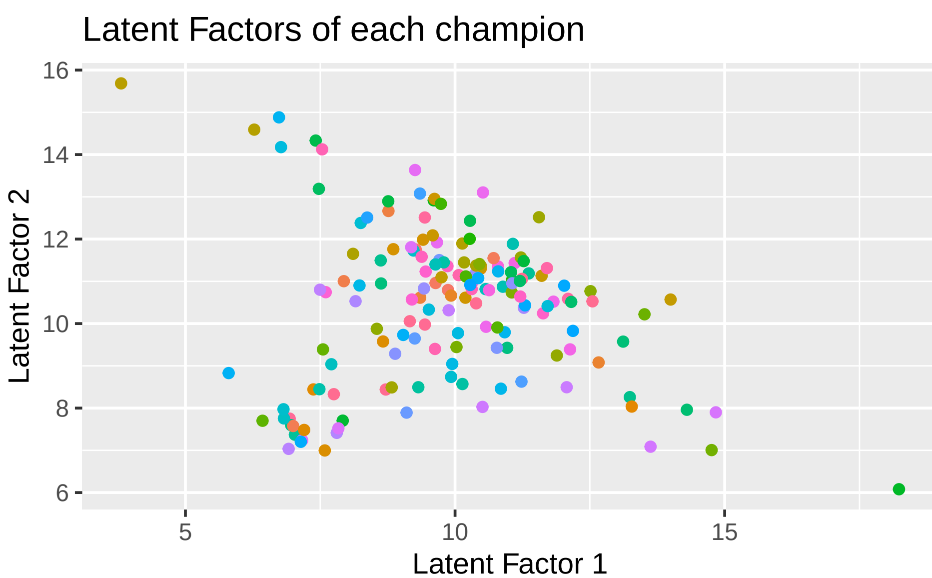
champion Selection Analysis

As was said before, choosing your champion is very important in League of Legends but exactly how important is it? Our p-value test on the winrate of each champion shows that it is not unreasonable to believe that each champion, aside from Vex, has a winrate of 50%. So maybe the choice of choosing a champion isn't that important. Here is where we will utilize some matrix factorization to discover some latent factors of each champion. Whats really powerful about matrix factorization is that our model will be able to learn and show us some continuous features that are baked into our categorical variable, here being what champion we are looking at.

From our xgboost model, we know that the most influential stat is the amount of gold earned per minute a player makes. So we will be using that as our value in each cell. So each cell is the average gold per minute the row champion earns in games where the column champion is on the opposing team. Note that since there are 158 champions in the game, we may not have a lot of match data for certain match ups. For example, it might be rare for Vex to face off against a Skarner. So we will be only using data that comes from at least 8 games. So each match up has to exist at least 8 times in our data set to be used. That way we aren't too weighted by a single match.

We will be factoring our matrix into 2 matrices that are 158x2 and 2x158. The reason why we are creating 2 latent factors is because it is the easiest to visualize and we are focusing on interpret ability.

After fitting our model and applying validation, we are left with a matrix factorization model that has an r-squared value of 0.88 on our test set. So our model explains 88% of the variation in our test set which is pretty good. Now that we have generated our latent factors, lets plot them.

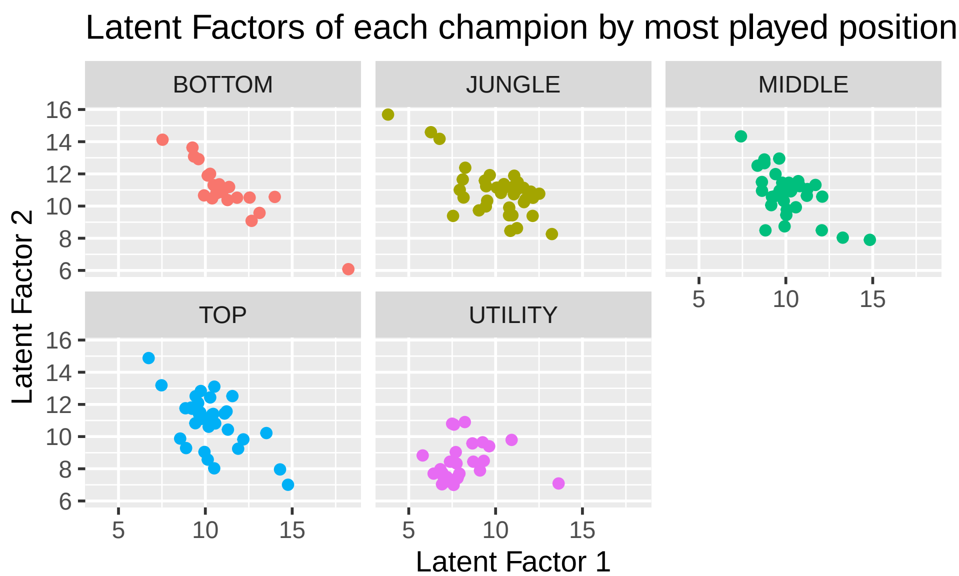


From here we can see that our 158 champions are mapped all over our feature space. There are some interesting shapes that can be seen in our chart which might be indicative of some clusters of similar kinds of champions. The most notable thing is how somewhat sparse the points are. This shows that there are some noticeable difference between each champion and how they perform in the pursuit of earning gold. If all of the champions were effectively the same, then the points would be much more densed and more of a ball would be formed. We even see some outliers which can be considered unique champions.

One of the top most point relates to Draven while the bottom most point is Lulu. Draven is a champion built purely to dish out damage while Lulu is made specifically to be support champions. It is interesting to see that the model has decided that these two should be very far apart as they have very little overlapping skills and play styles.

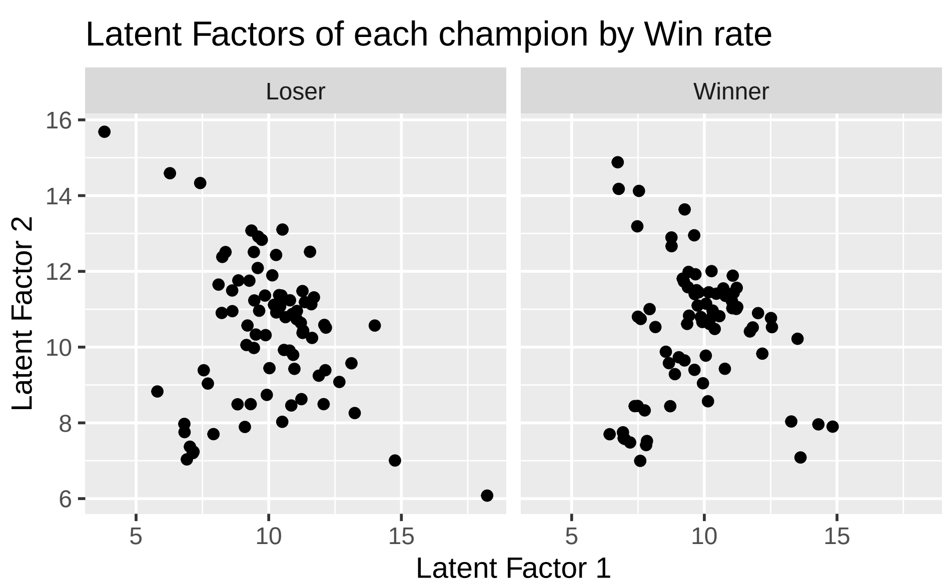
An interactive version of this graph can be found on the associated website at: https://mark-of-jp.github.io/jsc370-project/

Each champion is usually made with a role/position in mind. Let's see how this graph looks when we break it up by the most played position of each champion.



We can actually see a clear separation between all of the groups. It seems like our model, without even knowing that roles/positions exist, was able to seperate the champions into these groups based solely on their gold earned through differing match ups. All the champions in the middle are more versatile as they can somewhat succeed in playing in an off-role. For example, Annie can be found in the middle of the chart and can see success in the Utility and Middle roles. Draven and Soraka on the other hand, the top most and bottom most points respectively, excel in their positions but suffer anywhere else.

Now most importantly, we will see how these points relate to the champion's winrate. Here we will designate a winner as a champion who has an over 50% winrate.



Here we can see that there really isn't a difference between the shapes of those who have a higher than a 50% winrate and those who have lower than a 50% win rate. This might be evident that the game is for, all intents and purposes, balanced which is a great state for a game to be in. That means that you can choose your favourite champion and have a fair chance of winning. That doesn't mean though that you can choose any champion without thinking about the current state of the game. Looking at the model we can see that it actually learned roles/positions of each champion. The fairness is predicated on the fact taht you play the champion for the role it was intended to play. So feel free to choose any champion that you want but don't be disappointed if you are playing Soraka in the Bottom role and feel underwhelmed.

Conclusion and Summary

From fitting our model and looking at the data, we can gather a lot about the game of League of Legends. Our initial exploratory graphs gave us insight on the relationship of variables, our xgboost model showed us what skills are important and what aren't, and our matrix factorization gave us an understanding of how to make your choice on what champion to choose.

There are many decisions to be made and many skills to polish which can be overwhelming for a new player. So to aid new and veteran players, here is our recommendation on how you can win a game in League of Legends,

- You should polish your skills in earning as much gold as possible. This means killing and last-hitting minions, killing enemies and reducing any downtime you have. Use gold, and the difference of gold between you and your opponents, as a benchmark on who is currently doing better in the game
- Focus on getting kills and staying alive. This seems pretty evident but there are some nuances. Focus on getting high killing sprees. This means that just because you can get a kill doesn't mean you should. How likely is death if you attempt to secure a kill? If you can guarantee a kill at the expense of your own life, you should reconsider your options. That doesn't mean never trade your lives for another player but if you already have a number of kills under your belt, be wary of dangerous situations.
- Kills are more important than doing damage. While doing damage is useful, you are far more likely to win a game the more kills you have. So even if it feels like overkill, unless you leave yourself defenseless, do not be ashamed of unloading more than enough abilities on an enemy to ensure they die.
- Do not feel too pressured to choose the "best" or most "meta" champion to play. All of the champions seem to be fairly balanced and more notably distinct from each other. Choose a role you want to play and a champion that fits that role. Since there is some distinction between each champion it is better to choose specific champion and learn to get good with them. While some skills are transferable between champions, each one of them has their own flavour.If you are new and attempt to learn whatever is currently the "best" current champion, then you might find that your progress will be slowed down due to you constantly switching between differing characters.
- While being a team game, your personal achievements and actions within the game are very influential in the outcome. Even if you feel like your team is horrible, you are not guaranteed to lose. If you try your best, you might find that victory is possible.

Our last recommendation is one that needs no machine learning model support it. The last recommendation being have fun! At the end of the day, League of Legends is a video game and the goal of a video game is enjoyment. While it is cool to win, it is impossible to win every game so just have fun as you improve your skills.