STATS 202 Final Project

"FF at 20?": Diving into the Factors that Dictate League of Legends High-Elo Ranked Matches

Author
Taehyung (Kevin) Kim | takim@stanford.edu

All project code, data, and data scraping scripts available at:
github.com/kevintaehyungkim/lol-match-predictor

1 INTRODUCTION

League of Legends (LoL) is a team-oriented strategy game developed by Riot Games in 2009 that evolved from a small North American community to an international gaming powerhouse that now boasts over 115 million monthly players worldwide. To put this number into perspective, the next most popular game is Counter Strike: Global Offensive (CS:GO) which has 26 million monthly users, followed by Dota 2 at 11 million and Minecraft at 10 million.

There are several game modes offered, but by far the most popularly played map is Summoner's Rift, where two teams of 5 players maneuvering their selected champions collectively work together to destroy the enemy nexus. In short, there is one player assigned to each of the following positions on both teams: top, jungle, mid, AD carry, and support, and each position carries a vital role to contribute to the team's success. A player can elect to play in either ranked games or normal games. Ranked queue is different from normal games in the sense that each player has a Elo rank rating similar to chess and only compete with or against similarly rated players. Naturally, since there is more at stake, ranked solo-queue games are typically more intense and competitive. With a new set of players on different positions every game, 151 different possible champions, countless permutations of item choices and team compositions, it's impossible to confidently predict which team will emerge victorious. The goal of this project is to utilize regression and classifying techniques combined with my understanding of the game to construct a model that best understands the win conditions and strategies in the current meta. I will then assess the model's consistency in correctly selecting the winning team of any single high-elo ranked game of League of Legends given numbers that represent the game state.

2 RELATED WORKS

Due to the rapidly growing nature of e-sports and the gaming industry as a whole, there are prior studies on League of Legends game modeling and prediction algorithms not yet publicized due to the competitive nature of e-sports teams. There is also growing interest in incorporating artificial intelligence and deep learning to delve deep into winning strategies and counterstrategies to outperform the top human players in real time strategy games such as Dota 2 and Starcraft 2. Additionally, I came across a paper titled *Dota 2 Win Prediction* that incorporate interesting features such as team synergies and counters to boast over a 70% prediction accuracy. But what I did realize was that there was heavy emphasis on individual gameplay and match-up knowledge. With an unprecedented number of new champions released in League of Legends recently, I'm curious to explore how team performances and gameplay centered around objectives affect the outcome of a match in addition to champion counters and historical match up data.

3 DATA SET & FEATURES

3.1 Data Set

Prior to data set selection three key guidelines had to be established —

1. **All games must have been played on the same patch and region**
   League of Legends is a rapidly evolving game with semi-monthly game patches to balance the *meta* (most effective tactics available), apply bug fixes, and potentially release new champions. In order to ensure consistency throughout the games played and also have ample window for sources to collect raw data, all games will be from patch version 10.20. Since each region's understanding of the meta varies ever so slightly, this project only analyzes matches played in the North America (NA) region.

2. **Analyze games from diamond rank and higher in terms of Elo rating**
   There are presently 9 ranked-tiers in ascending order - iron, bronze, silver, gold, platinum, diamond, master, grandmaster, and challenger, with the challenger tier harboring the top 0.011% of the regional player base. At the highest levels of gameplay, there is much less variance among players' skill levels and understanding of the meta and strategies whereas the lower ranked tiers contain more players that play for entertainment value without necessarily understanding the meta or efficient strategies on how to win the game. This often results in figurative coin flips to crown the winning team and thus more difficult to correctly predict consistently.

3. **Factor both individual & team performance**
   In any competitive setting, a team of more-skilled players can easily lose a match to a team of slightly less-skilled players that possess great synergy. But as the skill gap also widens, there is less likelihood of an upset. Thus, I wanted to examine the significance of each position, a player's individual performance, in conjunction with meticulous team-fighting and rotations around the map in the final result.

3.1.1 — Riot Games Summoner & Match APIs

Provided key information about a specific match. In my research process, I collected game data from 300 matches played in diamond and higher-elo. More specifically, I wanted to examine matches evenly across the board and compiled closely spread out the matches amongst each of the four ranked tiers — diamond, master, grandmaster, and challenger. Due to API access restrictions without a professional license, the number of match data I could access with my account were unfortunately limited.

70% of the match data (~210 games) will serve as training data, while the remaining 30% of the matches (~90 games) will be used as test data.

3.2 Features

There were a total of 50 input features (and 1 output column for win/lose) — 1 feature to indicate whether the team starts on blue side or red side, 4 features to assess early-mid game objective team play (which can snowball or help bring a team back into the game) and 9 features for each player on a team to analyze individual performance. In the output vector, the winning team is marked with a value of 1 while the losing team is marked with 0. In summary, I prioritized feature selection on trying to determine the most important factors in order to get a lead into the late game, and how performances of each lane contribute to the team's overall success.

3.2.1 — General Features

- Map Side: 0 for blue side and 1 for red side.
- Win: 0 for loss and 1 for win (output)

3.2.2 — Player Specific Features: Each Player Statistics per Game (x5 players)

- Damage Dealt to Champions
- Damage Dealt to Objectives
- Vision Score
- Number of Kills
- Number of Deaths
- Number of Assists
- Gold Earned
- Minions Killed
- Champion Level

3.2.3 — Team Specific Features: Early-Mid Game Objectives (v. 10.20)

- First Tower: 1 for the team that succeeds; 0 otherwise
- First Blood: 1 for the team that succeeds; 0 otherwise
- First Rift Herald: 1 for the Amount of damage dealt throughout a game.
- First Inhibitor: 1 for the team that succeeds; 0 otherwise

Note: There were a handful of matches in which Riot could not determine which position (role and lane) one or more players were responsible for at the start of the game. I removed these matches as two separate positions may result in drastically different feature values and I wanted to dismiss any avertible variance. This ultimately brought the unique number of matches to 266 (match_ids.txt), and having a row for each team resulted in 532 rows.

4 MODEL SELECTION AND METHODS

4.1 Logistic Regression

My first idea was to apply logistic regression techniques since it is one of the more basic regression algorithms in predicting categorical outcomes, whereas linear regression is used more for continuous sequences. Since the ultimate goal is to best understand the multitude of factors that contribute in either a win or loss for a team, there are two discrete possibilities. I also felt that logistic regression would serve as a crucial benchmark with the other regression and classifying techniques, allowing me to better tune the algorithms and hyper parameters and realize when the results go too far astray.

We model the joint probability as:

$$P(Y = 1 \mid X) \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$
$$P(Y = 0 \mid X) = \frac{1}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}.$$

This is the same as using a linear model for the log odds:

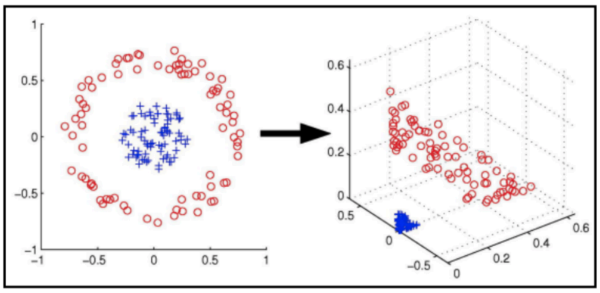
$$\log\left[\frac{P(Y = 1 \mid X)}{P(Y = 0 \mid X)}\right] = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

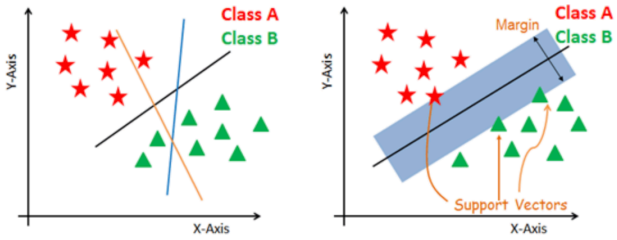
4.2 Principal Component Analysis (PCA)

There were around 50 features that factor into the model, mostly from each individual player's contributions of 9 features each. League of Legends is a team-based game, but with a lot of individual decision making components tied together. Although I am not confident that there will be strongly correlated variables that will aid in data simplification, I did not want to leave out the slight possibility that there are dormant pockets that in my dataset that are derivatives and simultaneously influence other variables as the game plays out. The use of Principal Component Analysis (PCA) may reveal underlying variance is attributed to each of the principal components, and although I do not have an immense set of training and test data, it provides the added benefit of reducing the dimensions into more digestible components for use in other algorithms. In the spirit of

4.3 SVM Classifier

SVM is a well-known supervised machine learning algorithm that can be used for both regression and classification problems. The main objective is to segregate the given dataset in the most optimal way possible, and it does this via finding a hyperplane the furthest distance from support vectors in the dataset. It is intrinsically two-class, works well with high dimensional space even with non-linear data, and offers more flexibility in how you want to interpret your findings. Due to these reasons I thought my use case would be a good reason to utilize this algorithm especially since I don't have an expansive test/training data and want to avoid overfitting. Although results vary widely depending on the kernel used, I also came across GridSearchCV from the sklearn library and incorporated it into the hyper parameter tuning process to assist with tuning for optimal parameters, gamma, and kernel to improve accuracy.





4.4 Random Forest

Random forest is a supervised learning algorithm that can also be applied to both classification and regression problems. One key benefit of using random forest is that it adds randomness to the model as the decision trees expand. It searches for the best feature among a random subset of features as opposed to focusing on the most important feature when splitting a node. In essence, the algorithm creates multiple decision trees and ultimately merges them together to get a more accurate but not overfitted, and stable prediction. As my dataset has a handful of features, I decided that incorporating the random forest algorithm can help demystify some of more complex interactions between the feature variables.

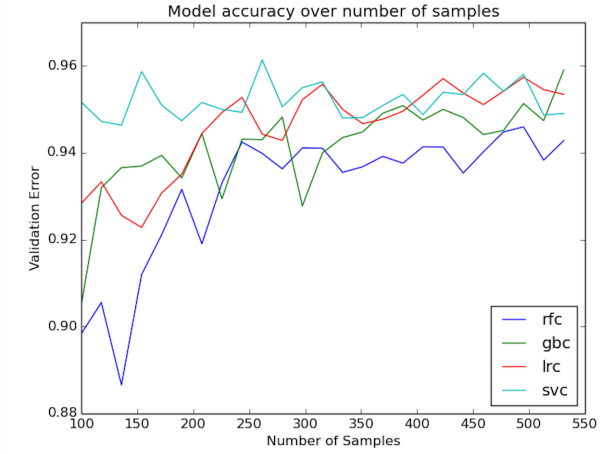
4.4 Gradient Boosting

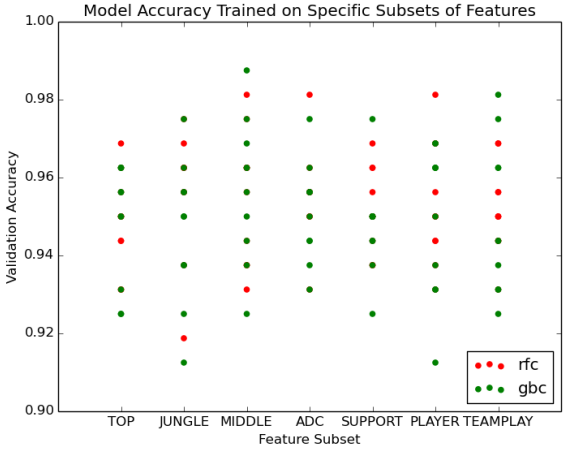
Gradient boosting essentially is a method to sequentially convert weak learners into strong learners. It identifies shortcoming in decision trees by applying gradients in the loss function $y = ax + b + e$ and combines the efforts of weak models such as decision trees to build a stronger one, each of the additional weak models reducing the mean squared error overall. Since we are using gradient descent and updating our predictions at the learning rate, it is much easier to find values for which the Mean Squared Error is minimum. In the case I have data points leading to errors that are difficult to fit with a simpler model (such as logistic regression), this approach seemed like a nice balance between logistic regression and SVM to start simple and iterate more later down the road.

4.5 GridSearch

GridSearch is an exhaustive process that results in optimal values for model hyper-parameters. This method is particularly useful when fine tuning algorithms such as SVM where there's heavy dependence on kernels and tuning values to classify correctly. Since I don't have an overly large amount of parameters, I decided that it would be worth the effort to search through the parameter combinations to find the best parameters for all the models used.

5 RESULTS





5.1 Result Analysis

During the hands-on phase, I was able to capture 4 models given the set of features that can calculate the prediction accuracy for a given ranked game of League of Legends using logistic regression, SVM, random forest, and gradient boosting. Although 532 samples is not an expansive training and test set by any means, it provided just enough of a sample size to witness a slow plateau near the 400 sample mark. The validation error for gradient boosting is nearing 96%, while the other models are hovering the 95% mark. One interesting note is that the random forest approach is slower to reach the same validation error as the other algorithms, but since it selects random subsets of features as opposed to immediately recognizing the most crucial features, it was interesting to see it behave as expected.

I wanted to take a better look at the gradient boosting and random forest models — specifically which features were more influential. I categorized the features into subsets, namely each position, features pertinent to an individual player's performance, and objectives resulting from teamwork and coordination. Although I expected specific player contributions to make the

largest impact, mid lane priority appears to have the most significant impact on the game. This implies that having the strongest player in the mid role is most conducive to a winning strategy, which makes sense since mid lane runs across the center of the map and have the potential to swing fights for objectives or vision more than top lane or bot lane.

Overall, I think that although the models proved to be highly accurate, I needed a larger sample size to deduce more concrete conclusions on which models ultimately perform the best given enough training time. I also realize that there would have been far more interesting results had I not included certain features such as the gold earned, number of kills, or the number of deaths, which is reflective of results at the end of the game and makes it many times easier for the models to accurately conclude that the teams with more kills and less deaths will be the victor. The intent was to explore how much winning one particular lane can dictate the whole game, and to some degree proved that mid lane is the dominant position to influence the outcome of a match as a skilled player. However, it would have also been interesting to analyze based off data snapshots once near the start of the game and another during the mid game, since the data captured in those stages are less indicative of the actual outcome.

6 CONCLUSION AND FUTURE WORK

6.1 Areas for Improvement

Although the results were more accurate than I had anticipated, I'm still excited to explore more options in the near future that I believe will ultimately enhance this model and result in better understanding of the meta which will lead to more accurate predictions.

6.1.1 — Champion Synergies and Team Compositions

I unfortunately didn't have the time to delve into champion synergies and how team compositions play a crucial role in the mid and late game team fights that can potentially dictate the result of a match. Given that there are countless champions with a multitude of different abilities that synchronize well especially at the highest competitive level of League of Legends.

6.1.2 — Item Builds

I believe item builds can also play an important role in how team fights play out, especially given that one or two auto attacks have the potential to flip fights around, and I would love to investigate how more offensive or defensive builds for a champion in a given meta factors into a team's overall success.

6.1.3 — Player Form

Any gamer has definitely experienced the highs of continuous win-streaks and also the lows of underperforming in crucial matches. Confidence is a major factor in both sports and e-sports and I will be looking out for ways to seamless integrate player psyche into future iterations.

6.2 Special Shoutout

Special shout out to Riot Games API for allowing me to continually bombard with parallel requests for the data needed to build this model (I guess the game is pretty fun as well). I'm also indebted to the countless but unnamed pages I came across on Google that helped clarify topics and better understand others in more detail throughout the duration of this course. And last but not least a thank you to Professor Jonathan Taylor and everyone in the STATS 202 course staff. It was a welcoming feeling to pop in to office hours any day of the week and to have learned invaluable knowledge that excites me about the field. I hope to continue building upon my enthusiasm and develop these foundations for the better.

7 WORKS CITED

[1] *Dota 2 Win Prediction.*
http://jmcauley.ucsd.edu/cse258/projects/fa15/018.pdf

[2] *How many people play League of Legends?*
https://leaguefeed.net/did-you-know-total-league-of-legends-player-count-updated-2020/#:~:text=League%20of%20Legends%20has%20a,1%20Million%20Daily%20player%20count.

[3] *Support Vector Machine Tutorial.*
http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf