Northwestern University - EECS 349 Machine Learning
Project Members: Yunil Seo, Katie Shin, Ethan Suh
Final Project Report

League of Legends (LoL) is played between 2 teams, 5 players each. Each player chooses a champion (character) in order so that there are no overlapping champions. Each player within a team decides which sections of the game they want to defend, and there are gold that can be earned for items and minions killed for experience. There are a lot of other factors of the game and players that determine which team wins. We attempted to gain the appropriate information about matches through the League of Legends API. Overall data is organized by playerID, matchID, championsID, and other IDs. We were able to pull match-specific data for each player, however information on averages statistics, such as the average minions killed per game, were only available by computing across multiple API calls. We soon discovered that this task is very difficult and time consuming because of the limit on how many API calls we can make (60 calls per minute).

We decided to parse information on a per-game basis only and we redetermined which attributes we should consider. The RiotAPI had 10 json files, each containing information about 100 matches, which gave us data for a total of 1,000 matches. We used javascript to parse the data and make csv file, which we converted to arff. Once the arff file was created, we ran the file through weka, specifically through classifiers such as ZeroR, Naive Bayes, IBk, J48, and Multilayer Perceptron. Each of the classifiers were ran with a 10-fold cross validation, which means that the data was divided into 10 equal subsets, and each subset was tested as the testing set while the other 9 were used as training sets.

We ran our data through weka, specifically using ZeroR, Naive Bayes, IBk, J48, and Multilayer Perceptron. One interesting finding is that the average win rate across the board is approximately 50%. This statistics makes sense because while the highest ranked players have a higher win rate, the lower ranked players have a lower win rate. Our task produced meaningful results, in that we were able to achieve high accuracy across various models through our selection of features. Even J48, though the tree splits on feature in illogical ways given knowledge about the game, was able to yield a solid accuracy of 85%. Furthermore, the classifiers such as Naive Bayes and Multilayer Perceptron were able to produce even better results. This is reasonable because Naive Bayes uses

conditional probability which takes into account the likelihood and probability of an attribute happening using priors and posteriors. Multilayer Perceptron uses backpropagation to edit the classified results to make it as accurate as possible, and has multiple activation layers which adds in more complexity to the calculation process. A note about the features we chose: we initially wished to compute the averages statistics of the current champion for each player--the win rate on that champion, average minions killed, average gold earned, etc.--since these values would have offered great insight in each team's' performance against each other. However, due to the rate limits of the Riot API, we simply used the per game statistics for each of these metrics and were still able to find meaningful results.

Potential improvements that can be made to this project include more attributes and team data analysis. Because we were having difficulty obtaining data from the API due to the constraint on the number of calls per minute, obtaining the full set of data per match would have taken us weeks to gather 1,000 matches' worth of data. More time to run our API rate-limited code, or access to the production API key with much higher limits, would have allowed us to gather the exact data we were aiming for. Also it doesn't matter which player plays which champion, but rather what the team composition is across each team of five players. Therefore, a bit-wise representation of the selected champions for each team might be a more logical process that would have offered different results.

Although each member of this team had different main roles within the project, we all contributed to the entirety of the progress. Ethan came up with the algorithms and code base for the javascript file we used, Yunil researched a lot of ideas for efficient coding and LoL database access, and Katie wrote reports and write-ups to keep track of the progress throughout the quarter. Despite the trial and error we went through, the results and facts discovered have been interesting and show a lot of potential for future improvement.