Data Analysis on League of Legends for Champion Selection

Zixiao Wang
zw4258@nyu.edu

1. Introduction

League of Legends, developed by Riot Games, is a global popular multiplayer online battle arena game. The players compete in teams of five, each controlling a unique champion (There are large sets of Champion to choose from) with distinct abilities, fighting to destroy the enemy's Nexus. The game map is mainly composed of three lanes and four jungle areas, which create five positions for each team (Top, Jungle, Middle, Bottom, and Support/Utility).

Figure 1. Champion Selection Page of League of Legends.

The game is highly dynamic based on champion selection with corresponding position selection at the beginning of the game. This project focuses on the design and development of a data processing structure to create a scalable recommendation system to optimize champion selection, utilizing Spark, Panda, Matlab, Spark.ML library, RiotWatcher [2]. The analysis of the proposed structure and system demonstrates positive outcomes, indicating strong potential in this domain.

Figure 2. Game map indicated with five positions

2. Related Work

In the article by Kenneth Mann, the author tries to start a project focusing on analyzing the data of League of Legends [5]. The author illustrates the method he used to collect the match data using the official Riot API with Python. The author used pandas for data manipulation, and Matplotlib for visualizing the data set. In the end, he tries to use the machine learning model to predict the outcome of the match.

In the article by Brendon Kirouac, the author shows how to create a database to match data using Python [3]. Especially, he presents the key step about how to use Riot API to collect match data as the Riot official API does not offer pure match data directly. The author uses PostgreSQL as the database, which I might not use, but the idea he used to clean and store the data is similar to the idea of this project. Also, the work demonstrates some practical approaches to analyzing large datasets in the context of gaming.

3. Data Collection Pipeline

The data set collection is both easy and difficult. The official server contains countless match data and keeps updated in real-time. The match data can be collected by sending requests to the official League of Legends server using the API key per developer account, which expires every 24 hours. The server limits 100 requests per 2 minutes per API key. Therefore, for continuously collecting large amounts

of match data in real-time from the official Server, multiple API keys with different accounts are recommended. However, in this project, one API is used, as the goal of the project to ensure the collection pipeline and analysis structure [3].

The players of the game are ranked with ten divisions, so the higher the division the closer the game is to the professional e-sport match. The diamond division (top 4% players in the US) is the target of the project. To get the data needed, it is necessary to scrape the data through division data first. Utilizing the diamond player user ID to request the match ID they participated in, I can locate the exact match data with the match ID sent with API to the server. Each user can be associated with all the matches they participated in, so pulling a large amount of users from high division is necessary for better results [7].

4. Data Cleaning

The next step would be data cleaning by removing redundant items and selecting data such as Champion selected, items bought, and banned champions. The data is scrapped into five CSV files during the collection process:

- summID: contains the players' ID from Diamond division.
- PUUIDs: contains the special ID for each player to pull matchs.
- matchID: contains the ID for each match.
- championData: contains champion selection, champion bans, and game duration for the winning team [6].
- champion_withLost_Data: contains lost team champion selection.

The Cleaning structure ensures the structure is able to filter essential information to fit different purpose projects. The initial metadata from players contains not only indirect access to the match data that can be used for winning matches but also contains the player's account information that is necessary to design any sale events in the game or potential marketing chance.

5. Data Analysis

5.1. Win Rate for Different Positions

Champion	TOP_count
Aatrox	61
Garen	43
Camille	40
Renekton	32
Darius	30
TahmKench	27
Jax	27
Sett	26
Malphite	25
Volibear	25
Nasus	24
Ornn	24
Riven	24
KSante	20
DrMundo	20
Yone	19
Mordekaiser	19
Pantheon	18
Kennen	18
TwistedFate	17

Figure 3. Top 20 champions in Top position with number of wins in 1000 matches

Champion	TOP_count
Aatrox	60
Camille	50
Garen	39
Sett	39
Yone	30
Darius	29
TahmKench	28
Volibear	27
Tryndamere	26
Jax	25
Mordekaiser	24
DrMundo	23
Renekton	23
Urgot	22
Nasus	22
KSante	22
Ornn	22
Fiora	21
Gnar	20
Malphite	19

Figure 4. Top 20 champions in Top position with number of lost in 1000 matches

Then start to analyze the Champion Selection. The win rate is the most efficient number to evaluate whether a champion for a certain position is a good pick or not, since in Figure 3, at the TOP position, the most number of wins is achieved by Aatrox, but he also has the highest lost count. Visualizing these data through multiple methods can be done by the pyspark's basic functionality [4]. To eliminate distraction, the champion with a total match number less than ten is excluded from the calculation.

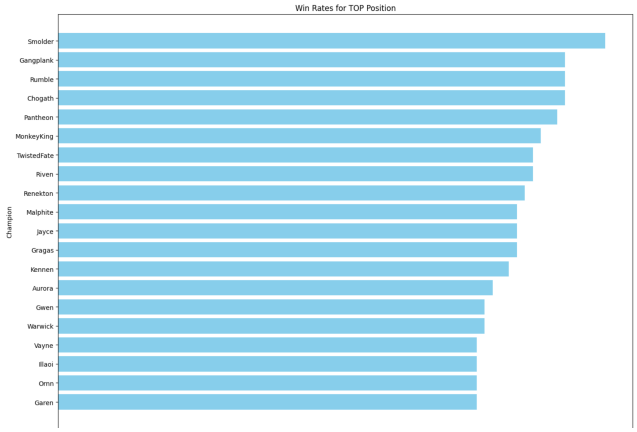

Figure 5. Top 20 champions in Top position with their win rate in Diamond Division

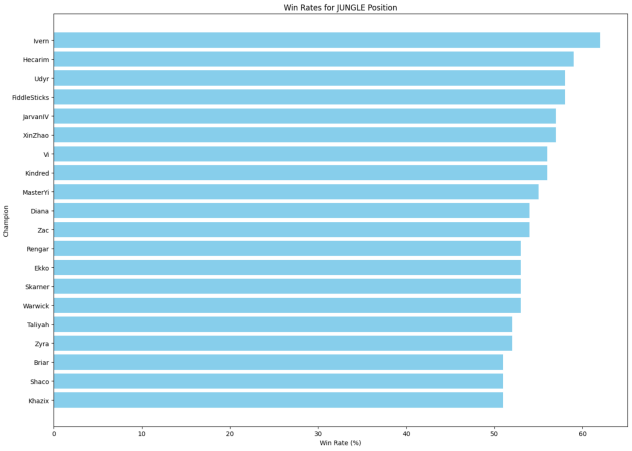

Figure 6. Top 20 champions in JUG position with their win rate in Diamond Division

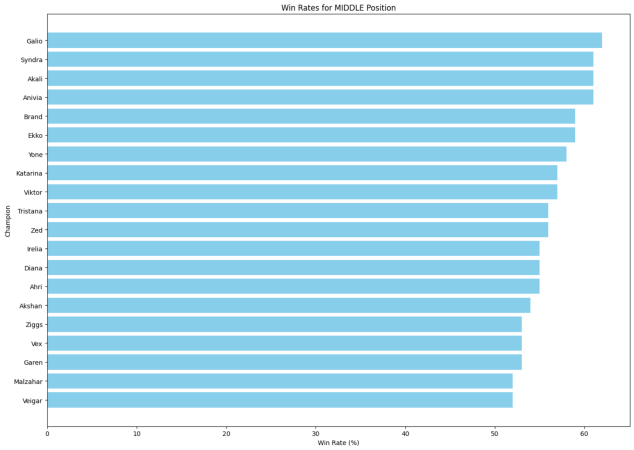

Figure 7. Top 20 champions in MID position with their win rate in Diamond Division

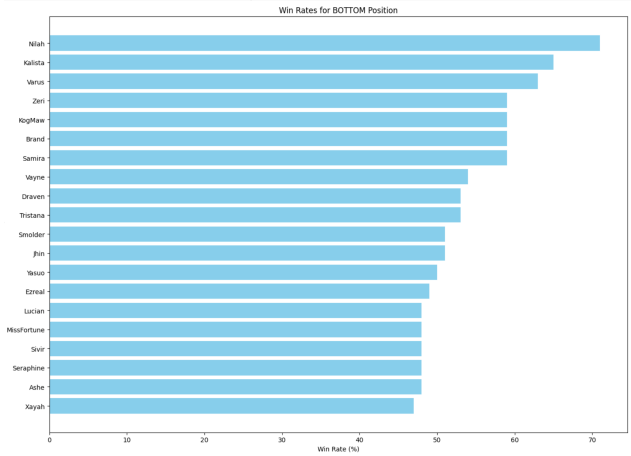

Figure 8. Top 20 champions in BOT position with their win rate in Diamond Division

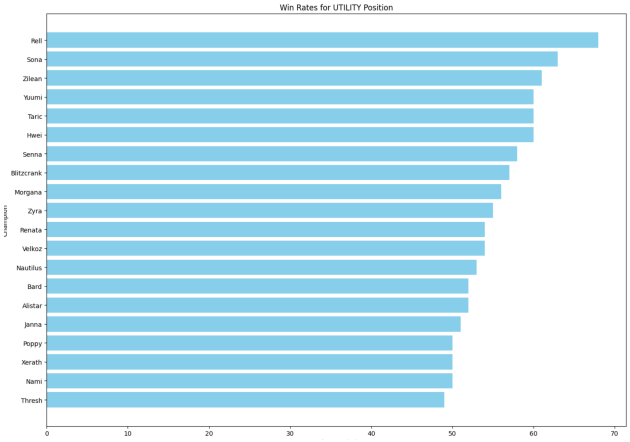

Figure 9. Top 20 champions in SUP position with their win rate in Diamond Division

Surprisingly, the win rate analysis produces more than champion selection on a single position. Instead, the visualizations present a game pattern in the professional matches. The project considers it as the winning pattern. As shown in the last two graphs, the winning tendency of several champions creates a larger gap, compared with TOP, JUG, and MID graphs. The top five champions for BOT graph are called mid-late game champions by players, who tend to be weak in the early game, while SUP graph suggests some champions labeled as powerfully supportive. At the same time, the TOP, MID, and JUG graphs favorite champions who are extremely aggressive in the early game. The pattern is about winning the early game with aggressive TOP, JUG, and MID to ensure BOT and SUP can end the game in the middle stage of the game. Surprisingly, such a pattern is widely used by most strong professional players in international tournaments currently, which is summarized by

observing a large amount of matches in person.

The alignment between our analysis and the observed game patterns in professional matches lands the foundation of this project. This strong correlation demonstrates the potential for professionals to use this data-driven structure to predict winning strategies and refine their gameplay prior to tournaments. Given that professional tournaments typically occur after the latest game version updates, players and teams can use this system to adapt to changes more effectively. This predictive capability offers a significant competitive advantage.

5.2. Time Duration as Winning Team

While a winning strategy often represents the best choice, it's important to recognize that player gameplay is crucial, and every player brings a unique style and performance to the game. Because of this variability, a single winning pattern may not suit all players, so providing alternative strategies is essential. This project also examines optimal team compositions at different stages of the game. The analysis is segmented into 15-minute intervals, up to 60 minutes, as most games conclude within this time frame.

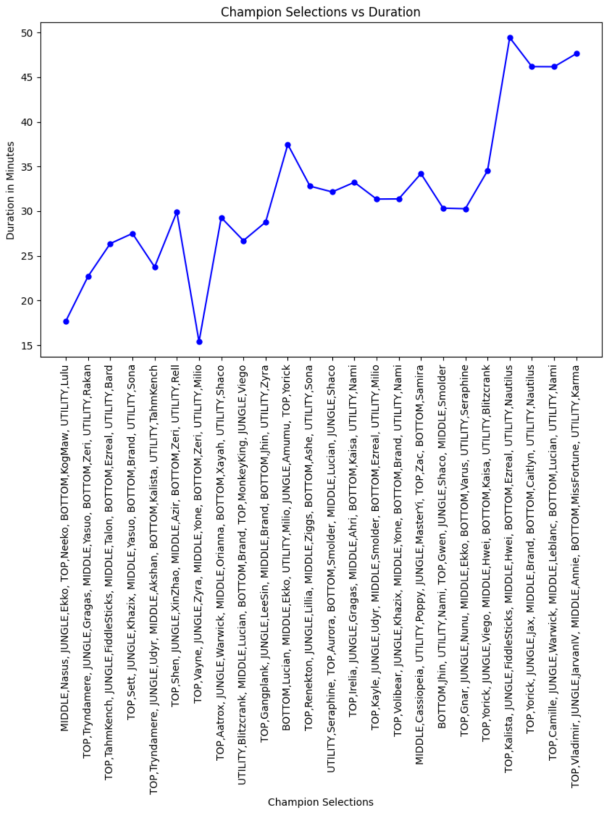

Figure 10. randomly pick 5 winning team compositions within each stage of the game

The graph further validates the specific winning patterns discussed earlier in this report, highlighting a critical trend: most games are won around the 30-minute mark, which corresponds to the mid-stage of gameplay. This observation reinforces the importance of mid-game strategies. However, the data-driven structure also uncovers valuable insights into team compositions that are particularly effective in extending the game beyond this typical time frame. Specifically, it identifies compositions that can sustain competitive play for at least 35 minutes, ultimately securing victory in the later stages of the game. This ability to prolong the match allows teams to capitalize on their strengths in the late game, which players specializing in late-game champions can be more comfortable with. These findings suggest that while early and mid-game strategies are crucial, there is also significant potential in developing compositions and tactics that are geared toward success in late matches, providing teams with a broader array of strategic options depending on their player styles.

6. Model Analysis and Potential Improvement

In this section, it focus on the development and evaluation of predictive models that aim to understand and optimize team compositions and their impact on game duration and win rates. It utilizes Apache Spark's MLlib to build these models, utilizing Random Forest algorithms for both regression and classification tasks.

6.1. Team Composition and Game Duration Model

To analyze the relationship between team composition and the duration of the game, the champion selections are then converted into indexed numerical representations using StringIndexer, which allows categorical data to be used as input for the model. A VectorAssembler is employed to assemble these indexed features into a single feature vector and the data set is split into 80% used for training the model and 20% for testing it. It applies a RandomForestRegressor [1] to predict the average duration of the game based on the team composition. The Random Forest model is chosen due to its ability to handle the interactions between different champions. The model's performance is evaluated using the Root Mean Squared Error (RMSE). The model yielded an RMSE of 14.739, which may appear quite poor at first glance. However, considering the constraints of the dataset—only 1,000 matches were analyzed and countless possible combinations of five champions from a pool of 168—the result is not as unfavorable as it seems. Initially, due to the limitations of the Riot Server, it was assumed that 1,000 matches would provide sufficient data. However, the results indicate that a much larger dataset is necessary to achieve more accurate predictions. Despite this, the underlying data-driven framework has been successfully built and tested, and improvements can be made by spending more time on data collection and utilizing a more powerful server

for processing. With these enhancements, it is expected that the model's performance will significantly improve.

6.2. Top Lane Win Rate Prediction Model

Similar to the previous model, a RandomForestRegressor [1] is employed to predict the win rate, with the data split into training and testing sets for model evaluation. The model is trained on 80% of the data, and its performance is assessed using the RMSE metric on the test data. This model is evaluated with RMSE, 7.7652. The RMSE still appears less than ideal, but when compared to the previous model, there's a huge improvement as the complexity is reduced from predicting with five champions to just one. This significant improvement underscores the effectiveness of the infrastructure and pipeline, demonstrating that they are functioning as intended. Further enhancements can be readily achieved by incorporating more match data into the pipeline, which will help refine the model and yield more accurate results.

7. Potential User Case

As a player, first to choose a position to play in the game. Through the data analysis portion of the structure, the player notices certain champions he can play with a high win rate. Then he further check the champions he needs to ban to get more advantages in the gameplay.

BannedChampion	count	SelectedChampion
Aurora	20	Smolder
Draven	17	Smolder

Figure 11. Top 2 most banned champions if selected Smolder as player's champion

Then use the model to input the team composition to see whether the team can win in early-game or mid-stage, even late-stage game. According to that, set up appropriate strategy through out the game.

8. Conclusion

In this project, we designed and developed a data infrastructure and pipeline for the League of Legends players to optimize their champion selection and lead them to win eventually. The data analysis conducted through this structure produces insights into winning strategies for different players and teams. Different from other data analysis tools for League of Legends, this project not only analyzes existing data but also applies predictive models to estimate winning chances and game durations based on specific champion selections. While the initial model results may not be ideal, the foundational structure is sound, and significant improvements in accuracy can be expected as the model is further refined and more match data is continuously processed through the pipeline.

References

[1] Apache Spark Developers. Pyspark randomforestregressor api documentation. https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.regression.RandomForestRegressor.html, 2024. 4, 5

[2] RiotWatcher Developers. Riotwatcher: A python wrapper for the riot games api. https://riot-watcher.readthedocs.io/en/latest/riotwatcher/LeagueOfLegends/index.html, 2024. Version 3.0. 1

[3] Brendon Kirouac. Building a league of legends match database with python. *Medium*, 2023. https://medium.com/@brendonkirouac/building-a-league-of-legends-match-database-with-python-7eee033090b9. 1, 2

[4] Brendon Kirouac. Feature engineering for a league of legends item recommender in python, 2024. https://medium.com/@brendonkirouac/feature-engineering-for-a-league-of-legends-item-recommender-in-python-49b3079a1903. 2

[5] Kenneth Mann. League of legends data science project. *Medium*, 2023. https://medium.com/@mannkenn/league-of-legends-data-science-project-42bd0933ddb4. 1

[6] Riot Games. League of legends data dragon json package: Champion data. https://ddragon.leagueoflegends.com/cdn/14.15.1/data/en_US/champion.json, 2024. 2

[7] Riot Games. Riot games api documentation. https://developer.riotgames.com/apis/, 2024. 2