Global Power Ranking Hackathon

This is an entry for the DevPost Power Ranking Hackathon.

https://docs.google.com/document/d/1wFRehKMJkkRR5zyjEZyaVL9H3ZbhP7_wP0FBE5ID40c

The PDF Version can be found here.
The video showcase of this entry can be be found here.

Introduction

This is an entry to the Devpost Power Rankings Hackathon 2023. I am a Software Engineer based in Berlin and have been a long time player of league and follower of it's esports scene. Discussion of the game from a technical and analytical view point and found it a great starting point for expanding my own knowledge of statistics and data analysis.

League of legends offers a lot of variations in micro decision and skill expression through fast, reactive gameplay and cool combos, and also big picture, strategic thinking and organized team work which make it a fascinating environment to analyze.

Entry

My Entry into this hackathon is in the form of an API. This API is available under https://usm38g8rwj.execute-api.eu-central-1.amazonaws.com/api where all the required endpoints are available. The first 20 teams of the global rankings system can be fetched with the following curl command: here is a curl command for each endpoitn

```
# get the global rankind (first 20 teams)
curl --location 'https://usm38g8rwj.execute-api.eu-central-
1.amazonaws.com/api/global_rankings?number_of_teams=20'

# get rankings for a list of team_ids
curl --location 'https://usm38g8rwj.execute-api.eu-central-
1.amazonaws.com/api/team_rankings?
team_ids=105593182576105603%2C102787200145472495'

# get the rankings for teams in a tournament by tournament_id
curl --location 'https://usm38g8rwj.execute-api.eu-central-
1.amazonaws.com/api/tournament_rankings/110733838935136200'
```

Tech Stack

- The API is written in Python with the Chalice framework from AWS
- AWS Lambda and ApiGateway for hosting the API

- S3 for accessing hackathon data and hosting data for the API
- Athena for exploring and exporting data
- Terraform to configure required infrastructure and permission

Python was chosen because my familiarity with the language and because it's ecosystem offers good libraries for both api development and data science tasks. Chalice is a Python library that allows us to easily deploy out code to AWS with lambda and API Gateway, using syntax that is similar to other common python api libraries like flask. The Athena data prepared

Index

- Intro
- building simple elo model
- k-factor tuning?
- trying simple prediction model
    - simple statistics
    -
- explanation why machine leanring prediction poor
- exploration of more complex systemic indocators
- idk lol?

WIN stats per Side

Total wins 25255 Blue wins 13352 Red wins 11903 Blue win rate 0.5286873886359137

First Blood win Stats

Blue wins 13350 Red wins 11903 Blue first blood 12886 Red first blood 12325 Blue first blood win 8241 Red first blood win 7234 Blue first blood win rate 0.6395312742511252 Red first blood win rate 0.5869371196754564 Average game time 318835.30970577756

Basic Statistical Indicators

To begin building a predictive model we can start by looking at some basic statistics indicators. A few somewhat obvious choices are first blood, first tower and first inhibitor. Out of the 25255 games in the dataset, Blue won 13350 games and red 11903. Therefore the blue side has a 64% winrate after getting first blood, while the red side has 0.59. The following table also includes the winrates per for first tower and first inhibitor.

Side	Total Game	Wins	First Blood WR	First Tower WR	First Inhib WR
Blue	25255	13352	0.63953	0.69689	0.94578
Red	25255	11903	0.58693	0.67633	0.94511

These stats might be interesting, but are only use for predicting the outcome of the game live as it is being played out. And event like the first inhibitor only happen quite late into the game, ideally we would like to make predictions earlier, even before the games starts

Historic Performance through rolling averages

When trying to predict the outcome of a game

Problem is difference between predicting live and predicting ahead of time? can u use eval from live prediction as a performance indicator for team rankings? how?

Tech Stack

- Python and Chalice for writing the API
- Terraform to configure required infrastructure
- AWS Lambda for hosting the API
- S3 for accessing hackathon data and hosting data for the API
- Athena for SQL queries

Python was chosen because my familiarity with the language and because it's ecosystem offers good libraries for both api development and data science tasks. Chalice is a Python library that allows us to easily deploy out code to AWS with lambda and API Gateway, using syntax that is similar to other common python api libraries like flask. The Athena data prepared

Elo System

The rating system is implemented using an Elo system. The reason for this is the process directness and simplicity. While the specific formulas or parameters can differ between elo system, the most important part is that their purely result based systems. It makes no assumptions about which strategies, mid term goals or game play patters are preferable or desirable.
The attribution of points depends only on a teams ability to win, and nothing else. This approach is a convenient way to cut through all the complexity and variance of a competitive game like league and easily derive and objective rating the accurate reflects relative strength base on past performance.
No doubt this is the reason these kinds of system are across many competitive games and sports such as chess, tennis and football.

K_Factor = 32 K_Factor_Scaling = expected_score_blue = 1/(1+10**((red_elo-blue_elo)/480)) Elo = Old_elo + (K_FACTOR*(1-expected_score_blue)*blue_game_wins)

K-Factor Scaling

- account for importance of different kinds of matches
- scaling based on bo5 nr

Testing The Elo System

Predictability

Out of the 2488 matches included in the dataset, with this elo system 50% of matches have an elo difference of less then 103. In all these matches, blue side has a win rate of 55%. In all matches with an elo diff greater then 103, blue side has a wr of 75%. In all red side favoured matches with an elo diff greater then 103, the

blue side win rate is 31%. The cutoff of 103 for even matched is picked as the point where half of all matches have an elo difference lower then this value.

total games 2478 half_match_cutoff 50 There are a total of 1248 "even" matches where the elo difference between the two teams is less then 103. EVEN MATCHES WR 0.54 1250 in those matches, the blue side has a win rate of 55% Out of the 600 matches with a elo difference of over 103, blue side has a 75% win rate when the bluefav_matches WR 0.77 657 redfav_matches WR 0.31 554 These win rates / elo difference are consistent with what is observed in other rating systems used by FIDE USCF.

Limitations

- elo system is not a predictive model, it is not supposed to predict failures or upsets, it is supposed to reflect them
- consider player injury
- predictive power can still be used to verify basic functionality of an elo system

Match Based Elo

Problem:

- difference in preparation between matches and games in a match
- difference of importance depending on nr of matches

Base elo on match performance: elo adapt for the score of specific match results

Analizing Statistical Indicators

First Blood win percentage base first blood win perc team frist blood win perc

Blue Team Base Winrate = 0.5284001981178801 Blue Team With First Blood Winrate = 0.5305619585088626 Red Team Base Winrate = 0.4715998018821199 Red Team WR With First Blood = 0.4736004456292524

Basically no difference, diff per team not much different

100 Gold Difference @15 mins

total_games = 25195

games_with_blue_adv = 13241 blue_win_with_adv = 10025 blue_loss_with_adv = 3216 blue_wr_with_adv = 0.75711

games_with_red_adv = 11950 red_win_with_adv = 8661 red_loss_with_adv = 3289 red_wr_with_adv = 0.72476

- small gold diff
- much better indicator then first tower or kill (that actually should give way more gold?)
- better because it is compounded statistical indicator, it is the result of all the other actions being played out, and tradeoffs between different plays and event is evened out through the games systems into a mor useful statistical indicator
- how much does it have to do with being a statistic taken later in the game?
- how does it compare to 10 min?

100 Gold Difference @10 mins

games_with_blue_adv = 12703 blue_win_with_adv = 9158 blue_loss_with_adv = 3545 blue_wr_with_adv = 0.72093

games_with_red_adv = 12482 red_win_with_adv = 8329 red_loss_with_adv = 4153 red_wr_with_adv = 0.66728

team total wr

- MOst statistical indicators are not very useful
- compare to other statistical indicators

Modeling the Game

- modeling the games systems at a high level, gameflow and comback systems -> comback systems result for difference in time based on skill level
- use reinforcement learning https://www.youtube.com/watch?v=bD6V3rcr_54 https://www.andrew.cmu.edu/course/10-403/slides/aws_gym_tutorial.pdf

Enhancing Prediciton from ELO model

IDEA: Update elo not per game but per match with increasing importance according to games in a match win bo5 boX are good numerical indicator of match importance since it's unlikely you'd want important titles decided by a small nr of matches how do we make this so the code isnt shit :thinking_face:

RESERACH:

Assing k factor to different leagues not as effective since teams dont usually move between league except for msi and worlds. so teams in major region both win and lose more elo

COMPARING RANKING SYSTEMS: There are many ways to compare two power ranking systems. Here are a few common methods:

```
Direct comparison: This is the simplest method, and it simply involves comparing
the rankings of each system for each team. The system with the higher rankings for
the most teams is generally considered to be the better system.
Predictive accuracy: This method compares the ability of the two systems to
predict the outcomes of future matches. This can be done by using a historical
dataset of match results and comparing the predictions of the two systems to the
actual outcomes.
Sensitivity to recent results: This method compares how sensitive the two systems
are to recent results. A system that is more sensitive to recent results will tend
to change rankings more quickly in response to changes in team performance.
Complexity: This method compares the complexity of the two systems. A more complex
system may be more accurate, but it may also be more difficult to understand and
use.
Robustness: This method compares how robust the two systems are to changes in the
data. A more robust system will be less likely to produce inaccurate rankings if
the data is noisy or incomplete.
```

Elo rating system, the Glicko rating system, and the Sagarin rating system.

https://wismuth.com/elo/calculator.html#best_of=5&score=0-0&system=tennis-men&e_score=0.6

CHANGELOG:

13/09/2023: Implemented a basic elo system based on games wins 14/09/2023 Save report and images to daily folder