League of Legends – Strategy Analysis

Team members:

- Michael Merzlyak (email: michael.merzlyak@mail.huji.ac.il, csid: mikemerzl)
- Guy Ben - Eliezer (email: guy.beneliezer@mail.huji.ac.il, csid: guybene96)
- Yarin Schiller (email: yarin.schiller@mail.huji.ac.il, csid: yarinsch)

Problem description:

The Problem:

In Chess, when you learn the game, it is very common to learn from the greats. It is very common to take some state of the game and study a pro move and take it as your own (Like: "Albin countergambit", "Amar opening", etc.). We decided to take that and see if it transferable. We decided to take the game "League of Legends" and using algorithms from the course to see if a low-level team, using strategies from an experienced one, has a better chance of winning the game and what kind of parameters were important (SPOILER ALERT: We succeeded).

We wanted to look at this from a recommendation point of view, we tried to build a recommendation system for a low-level team (hence: "user"), considering their choices in the game.

The Game:

League of Legends is a team-based game with over 140 champions to make epic plays with. In the game, two teams of five players battle in player versus player combat, each team occupying and defending their half of the map. Each of the ten players controls a character, known as a "champion". During a match, champions become more powerful by collecting experience points, earning gold, and purchasing items to defeat the opposing team. In the game's main mode, Summoner's Rift, a team wins by pushing through to the enemy base and destroying their "Nexus", a large structure located within.

Summoner's Rift is the flagship game mode of *League of Legends* and the most prominent in professional-level play. The mode has a ranked competitive ladder; a matchmaking system determines a player's skill level and generates a starting rank from which they can climb. There are nine tiers; the least skilled are Iron, Bronze, and Silver, and the highest are Master, Grandmaster, and Challenger. Summoner's Rift matches can last from as little as 15 minutes to over an hour.

Data:

We gathered data on games from 2 different leagues, the bronze league, and the grandmaster league. Our goal will be to look at games from bronze league and obtain strategies from the grandmaster league.

By using several methods from the RIOT games developer API (LOL's company) we managed to get players names, from both bronze and grandmaster leagues. We didn't enter any parameters and tried to gather data from different part of the globe to increase independence between the teams. After that we obtain game id's and took the "timeline" of each game.

From the data gathering we obtained bronze players and grandmaster players, with a size of around 100 kb, this data was a list of tuples with each tuple containing a player name and its region. These are "summoners_grandmaster.pkl" and "summoners_bronze.pkl" data.

Then we built a list of games from those players, the list contained game id's weighing at around 45 kb. These are "matches_bronze.pkl" and "matches_grandmaster.pkl".

Finally, we gathered the timelines of each game and ended with 2,000 bronze timelines and 15,000 grandmaster timelines. This data is in a Json format containing plenty of information, the important part is that every timeline contained a list for every minute in the game containing a list of "events". Every event is a Json with keys according to the event type. Events like "Elite monster kill", "Item purchased", "champion kill", "ward placed", etc.

Here is an example to see a structure for a single game. Each frame is a minute, containing several events:

We have a lot more grandmaster games than bronze for this question, hence we have 450 mb for the bronze and 1.2 GB for the grandmaster. This is "timelines_bronze.pkl" and "timelines_granmaster.pkl".

Solution:

To solve our problem of getting a good strategy to a team of beginners based on the choices of veterans we used a recommendation system. We basically wanted to recommend a user a strategy of he's liking based on he's choices in the game so far. To do so we knew we could get data on other players and hence the obvious candidate was to use collaborative filtering and user-user based model. (The entire algorithm is in the "DataAnalysis" notebook at the second cell)

Pre-processing the data -

To get the answer, we needed to preprocess our data. We started with building a class ("Match") that holds the data of a single game and calculates two main things. First, we gathered the champions for each team, and second for each minute we built 2 dictionaries, one per team. The keys were events that we saw relevant for the strategy (for example "level up" is automatic so we didn't count it) and considering some parameters per event type (this can be seen in the "data_types.py" script looking at the "ALL_TYPES"). The dictionary values were the number of times that event occurred. After that for simplicity we built the "Teams" class that held only the relevant dictionary and the champions of the game.

Finally, if the data was invalid for any reason, bad request, missing champions, missing a winning team, we skipped it. We took only 1 team from each game so the users will remain independent. We ended with around 850 bronze teams and 8,500 grandmaster teams.

The algorithm –

We wanted to get the cosine and Jaccard similarity for a bronze team compared with every grandmaster team and choose the most similar. But first we needed to consider that the basic algorithm ignores the main factor of time. To deal with this we calculated the similarity of every minute, then taking the mean to gain one number that considers the time of the events in the game.

We did these calculations with 3 parameters in mind, which metric will give the best result, how long from the start of the game should we take before we give our recommendation, should we consider the champions of a given team when searching for a similarity?

To answer the questions, we calculated 12 different times the similarity for each bronze team. First, we took 5, 10 and 15 minutes of each game to obtain the teams events then doing the same for the grandmaster teams we

calculated those decision similarities using the metrices (Like explained above, by taking the mean of the first 5/10/15 minutes of the game). After that we wanted to see if we should consider the champions of the strategy, we took the first 10 minutes of the game (take the middle of the before parameter) and compared the same way as before only now we only compared between teams that had 1/2/3 champions in common (we tried 4 and 5 but we simply couldn't find enough games to match that). All in all we have 12 different calculations for each game, which were 5/10/15 minutes and then 1/2/3 champions in common each calculated with the Jaccard and the cosine metric.

Evaluation:

Evaluation Criteria:

We looked at the similarity between our recommendation's strategy and the real strategy played in the game. We compared all the minutes in the game (taking the minimum length of both games to compare) using the relevant metric (games that were calculated with the Jaccard similarity at the first place were now being evaluated with the same metric). Now we checked if a game won. We wanted to see a game that "didn't follow" our recommendation were less likely to win.

After calculating the entire similarity between a bronze user and it's chosen grandmaster user, we grouped the scores into 10 quantiles so we could group the games according to who "listened" to our recommendation the most and the least.

After obtaining the groups we calculated their rate of success, hoping to find the group with the smallest similarity to lose the most, getting a metric to see if our recommendations were fruitful.

Moreover, in order to make sure that our results were statistically significant we noticed that because we are looking at teams that either win (:=1) or lose (:=0) we actually wanted to estimate the parameter for a Bernoulli random variable with a parameter of θ. Where each group were somewhat independently distributed with there own parameter. To check that our results were indeed significant we calculated a CI with 95% confidence level for the parameter. Then by plotting with the CI we saw if the connection we saw earlier was indeed 95% true.

Setup:

As I explained earlier, we took 1,000 bronze games, trying to take from different regions of the world, and then we took a single team from each game while dropping problematic data, ending with around 850 games ("users").

We compared those games to around 8,500 grandmaster games, finding there most similar game (a recommendation for each metric) based on some parameters (times and champions) and recommending that one.

Then, we saw if the games "listend" to us by finding the entire similarity between the game and the recommended game. Finally seeing if there was some connection between games that didn't play like there grandmaster counterpart and there win rate.

Results:

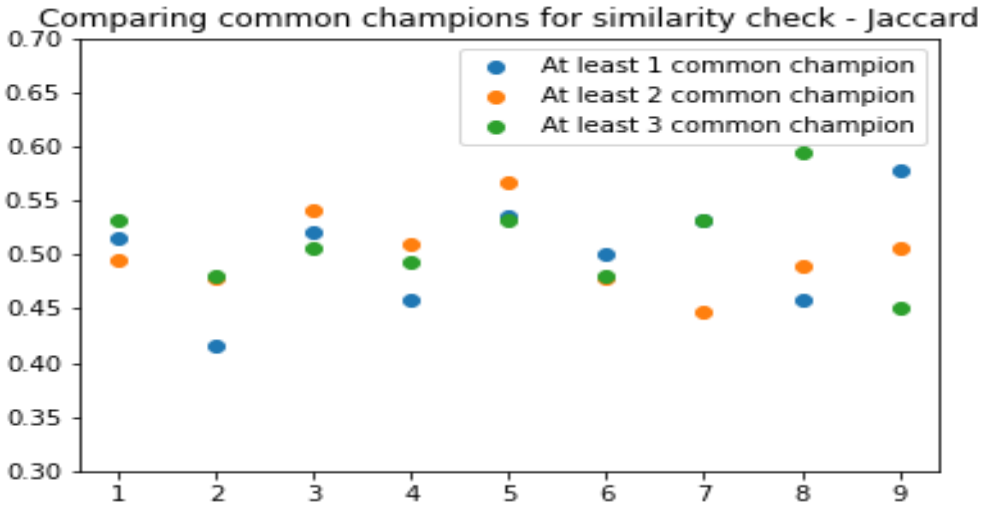
We saw several interesting results:

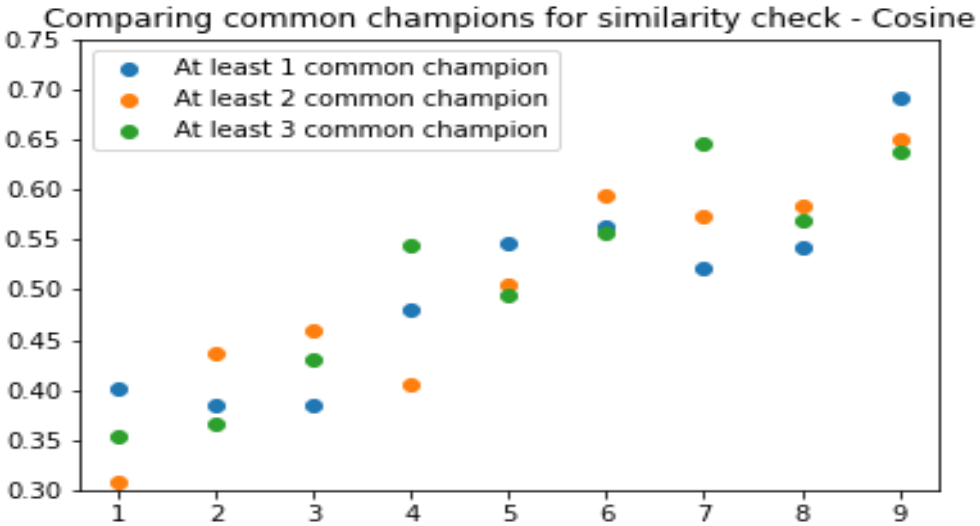
1. There **is** a positive connection between the games that played like there grandmaster counterpart and there win rate. We saw this connection with all our parameters when taking the cosine metric in account.
2. When taking the cosine metric and not the Jaccard metric the connection exists between the strategies and the win rates.
3. Taking the parameter's extreme (5 and 15 minutes) and (1 and 3 common champions) affects the variance of the results. When taking the larger constriction, there seemed to be a more consistent connection.

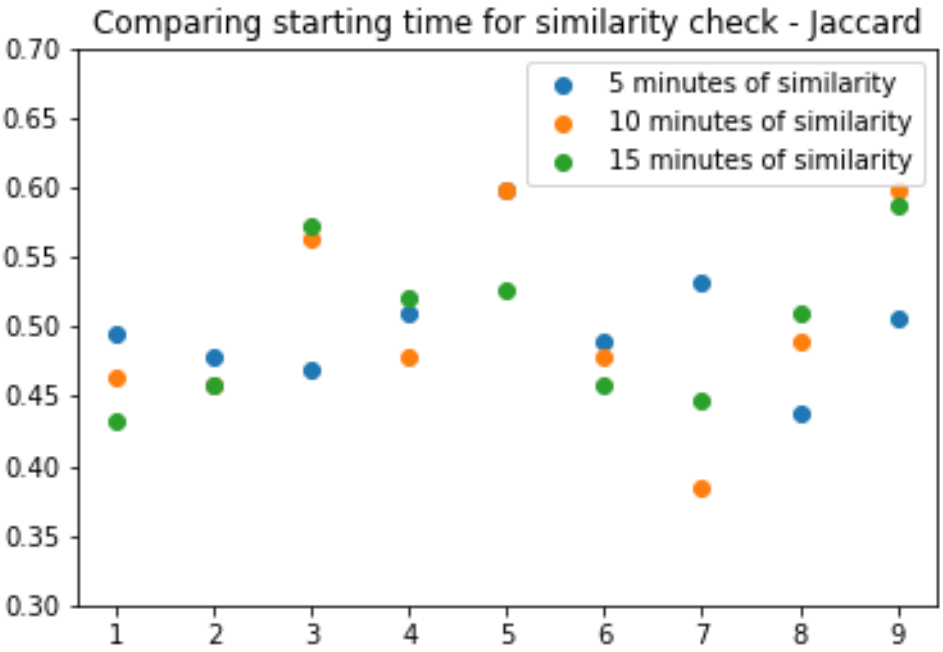
Visualization:

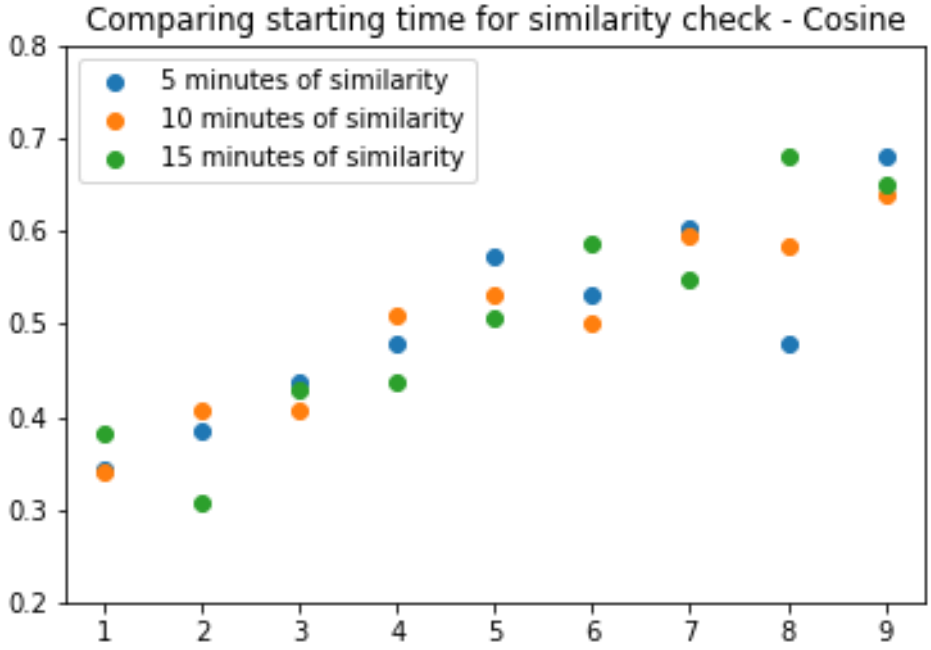
All the graphs (scatter and line plots) were calculated such that the x-axis was the groups grade (as explained above) and the y-axis was the groups win rate. All the groups were approximately equal in size (because of the way we built them). When noting the Jaccard or cosine metric it means that all the calculation were done with that metric, the choice of the recommendation and the final evaluation.

- We first wanted to see a general trend while comparing our parameters values. From the graphs below we can see that the cosine metric has a more consistent and larger impact on the results. The trend seems more consistent and maximum at 9 and is higher while the minimum at 1 is lower than with the Jaccard metric.

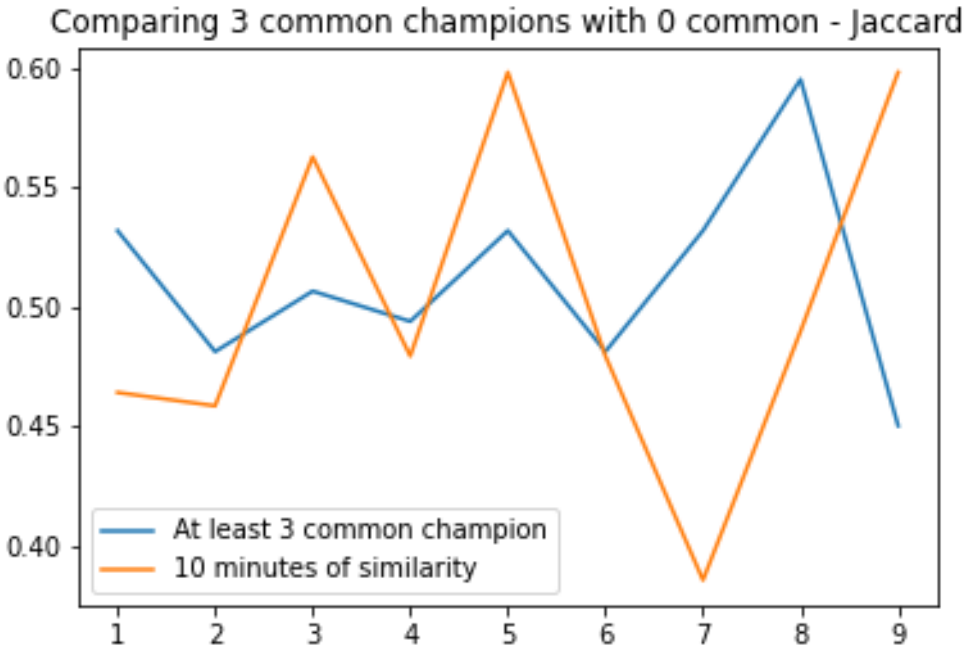


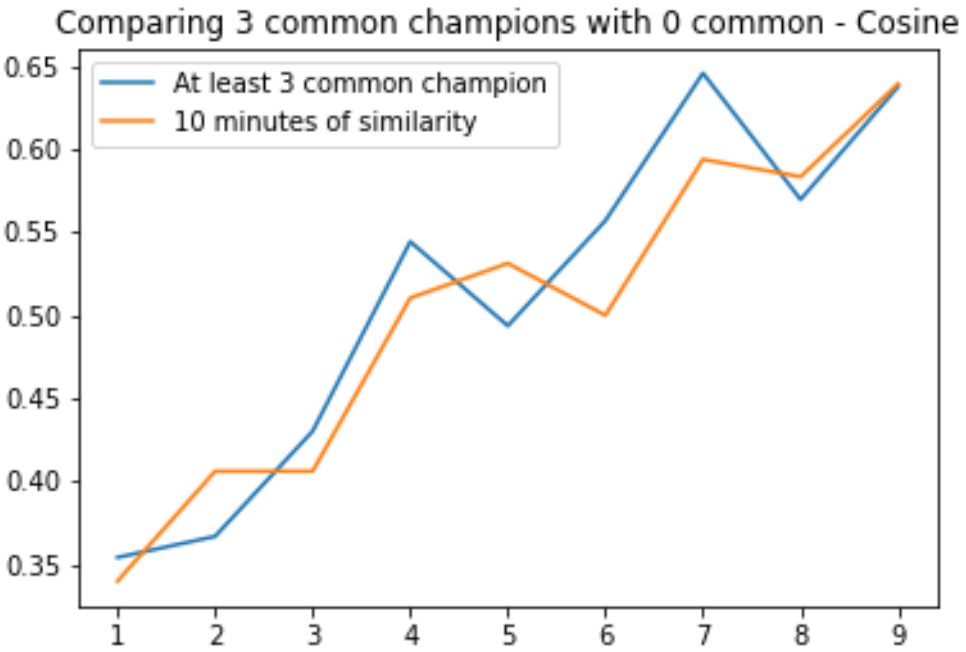



Comparing starting time for similarity check - Jaccard
5 minutes of similarity
10 minutes of similarity
15 minutes of similarity
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
1
2
3
4
5
6
7
8
9

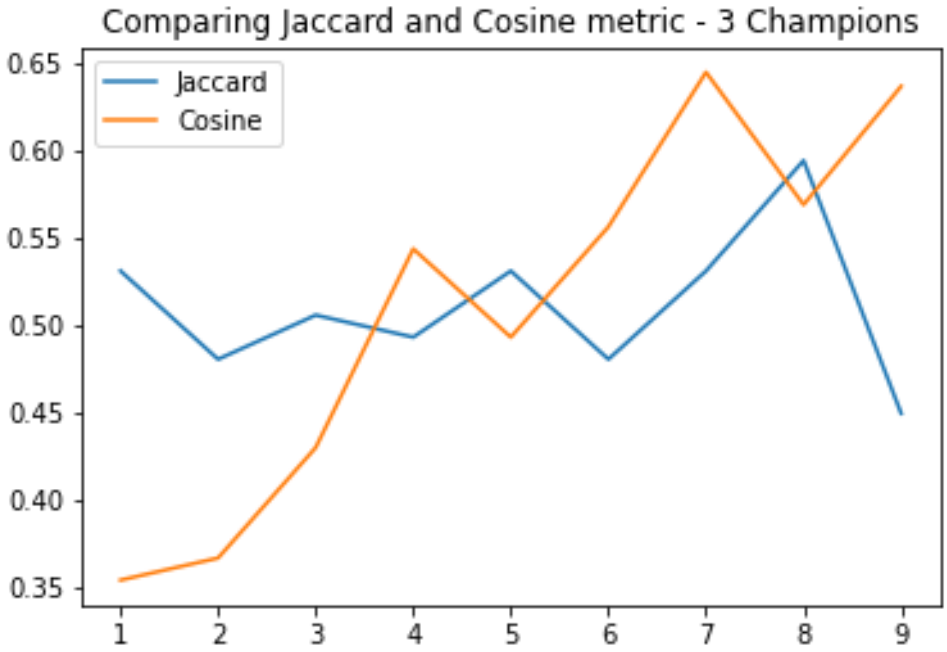

Comparing starting time for similarity check - Cosine
5 minutes of similarity
10 minutes of similarity
15 minutes of similarity
0.8
0.7
0.6
0.5
0.4
0.3
0.2
1
2
3
4
5
6
7
8
9

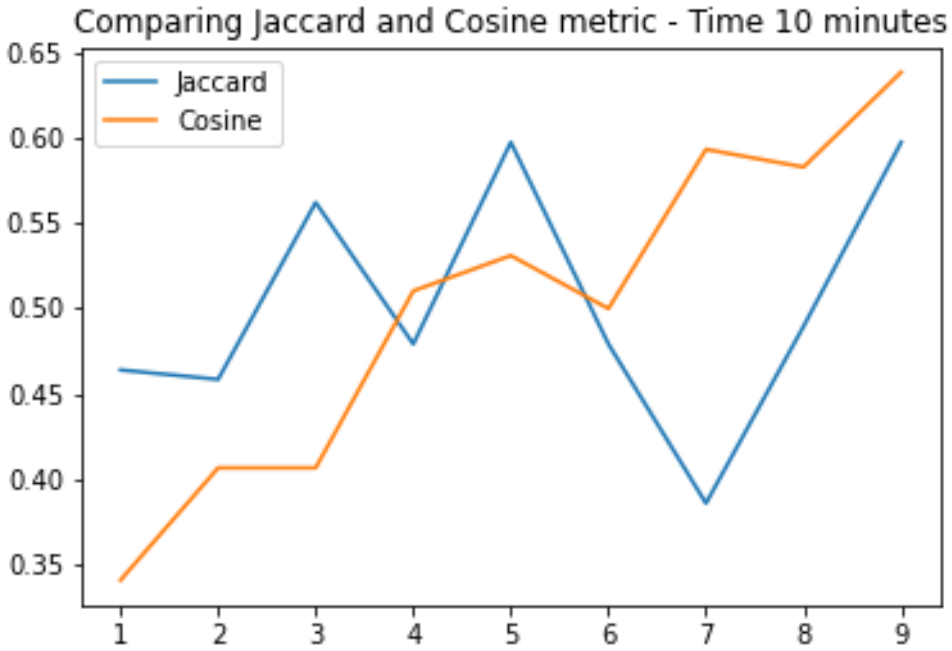
- When looking at the graphs, we don't see a big difference between the parameter's values. We wanted to see a difference between the extremes of the champions parameter, so we plotted with 3 common champions and no common champion:



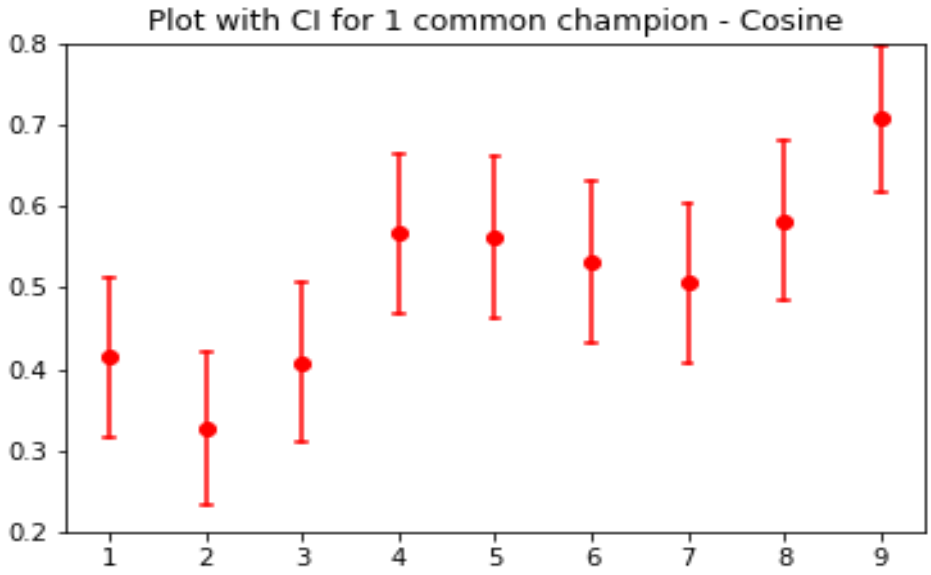


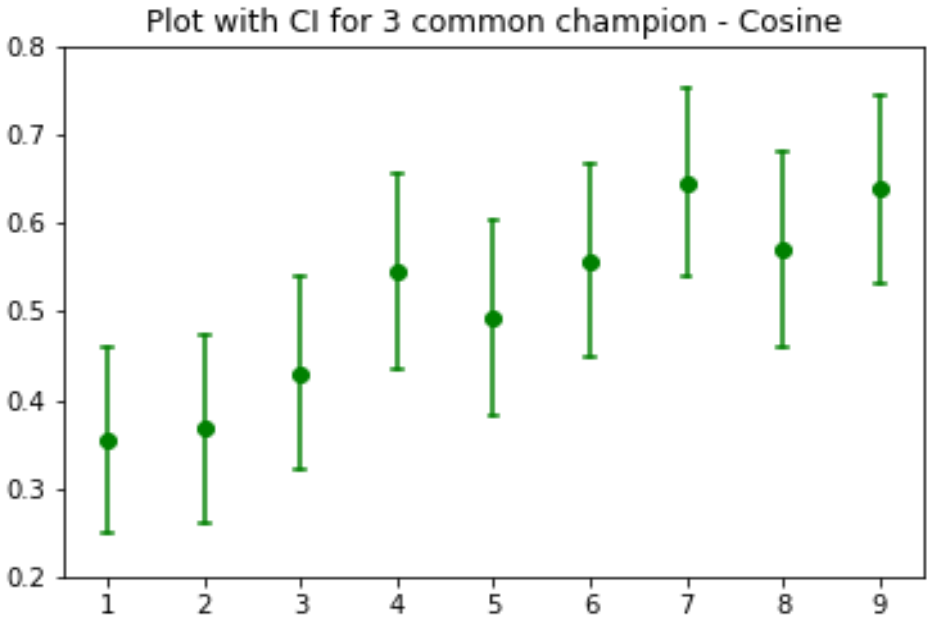
- It seemed that the cosine metric was a better choice than the Jaccard metric in all our graphs, so we decided to plot the same parameters with the different metrics to compare. By the graphs it seems that indeed the cosine metric gives a greater effect on the win rate, and it seems the win rate is very affected by the change in the Jaccard metric:

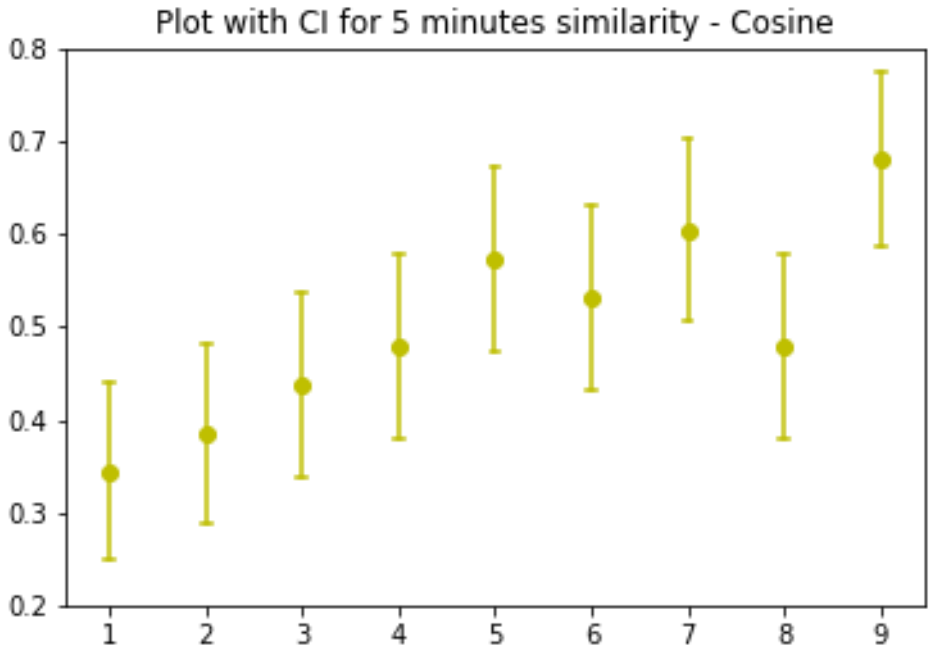


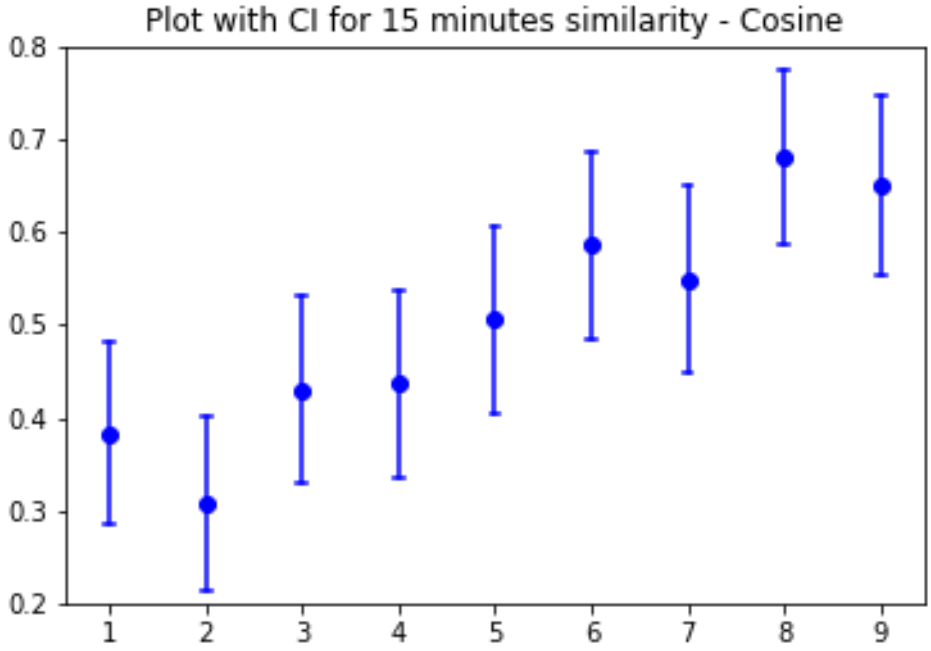


- In order to see that our results are statistically significant we plotted the CI to see a trend even with the 95% CI. We also wanted to see a difference between the parameter values, we hoped to see a difference in the CI between the parameters:






Plot with CI for 5 minutes similarity - Cosine
0.8
0.7
0.6
0.5
0.4
0.3
0.2
1
2
3
4
5
6
7
8
9


Plot with CI for 15 minutes similarity - Cosine
0.8
0.7
0.6
0.5
0.4
0.3
0.2
1
2
3
4
5
6
7
8
9

Impediments:

The main problem was how to evaluate the output and return a numeric value to be compared. At the beginning we had scores of similarities and win and lose, plotting that we got something resembling the Dirichlet function (dots all over the x axis and $y = 1$). We had to make our scores from there continues space into a discrete space.

Future Work:

For future work we could do:

- Check the same algorithm only with individual champions and not a team
- Given more games we wanted to check the same algorithm with a perfect match of all 10 champions, 5 from each team, and compare those strategies

Brief Conclusion:

We managed to find a connection between new teams win rate and how close to veteran teams they are playing. We learnt a lot about how API's work, how to get data from companies and that data can sometimes be corrupted.

We learned a lot about data analysis and that you need to be creative sometimes to understand what you really got.

At the end of project, we had fun on the one hand and came up with real results on the other.

Code:

All the data is in
https://drive.google.com/drive/folders/13v0RDy0Am6guIRwnPi45idVX-z5sNZBG?usp=sharing

The code is on GIT

https://github.com/yarinsch/Group23Needle.git

The main part in DataAnlysis.ipynb. In order to run the notebook you only need the two final files we worked in the notebooks folder, grand_teams.pkl and test_bronze_teams.pkl.