Data Analysis and Machine Learning on League of Legends

Yiyi Zhou, Chang Liu
zhou.Yiy@husky.neu.edu
liu.chang12@husky.neu.edu
INFO 7390 Advances Data Sci/Architecture, Fall 2017 Northeastern University

Abstract

Based on selected world tournament match records dataset, this project aims to make analysis about region gaming styles, strategies, and the win rates behind it in the game of League of Legends. This project made a logic regression to analyze the effect of different parameter on the match result and used Log5 formula to find the win rate between each two champions. Furthermore, from these match records, with various champions selected by different players to content, a model was trained to predict match result based on champion selection. With the analysis in this project we answered how to pick champions according to the champion picked on the other side.

Introduction

League of Legends (LOL) has been a typical and popular multiplayer online battle arena (MOBA) video game developed and published by Riot Games. Released in 2009, it has been a great popular MOBA game around the world.

During a classic match, players will be involved in a 5v5 team game to work with teammates and compete with the enemy team. The ultimate goal is to destroy the main base of the enemy's team. For each match, players can select a different champion as the character to operate, and each champion has unique abilities and game roles. As shown in the figure, the entire map is divided into blue side and red side with two teams spawned in two corners. Right in the left bottom and right top are the blue side and

Fig 1. A classic MOBA game map

the red side main base respectively, and there are three lanes connecting two bases in total, called top lane, middle lane, and bottom lane.

The other diagonal vertical to the middle lane is a path called the river, and these two diagonals divide the rest part of the map into 4 parts, and they are all called jungles. Generally, a 5 players' team would assign one to the top, one to the middle, one to the jungle, and two to the bottom, their position is called top, middle, jungle, bottom and support.

Usually the bottom is an archer (a champion's role, usually called AD carry/ADC) who needs a long time of farming, so a support is assigned for him/her to protect this process, and they are assigned to the bottom lane to protect the dragon to be stolen from the enemy, which is an important strategic resource in the early game.

Dataset

The datasets used in this project are from kaggle.com and other game statistic website.

Kaggle.com:
https://www.kaggle.com/chuckephron/leagueoflegends (Professional Tournament)
https://www.kaggle.com/lanls1/matchidv1 (Rank Matches)
https://www.kaggle.com/xenogearcap/league2016 (Rank Matches)

Official Match History site:
http://matchhistory.na.leagueoflegends.com/en/#page/landing-page

Official stats API:
https://developer.riotgames.com

Data Analysis

1. The effect of different coefficients for the match result and logic regression.

The match results of LOL depend on many different coefficients, like the gold gain of the team, the dragons it killed, the barons it killed etc. We made a logical regression to see the effect of different coefficients on our match result. We chose the value of game length, the dragon killed, the gold in 10 minutes, the gold in 20 minutes, the herald number, the baron value, the inhibits value, the tower knocked down and the kill times in a match as the independent variable.
The following is the regression result we got.

	coef	std err	z	P>|z|	[0.025	0.975]
gamelength	-0.1703	0.010	-17.691	0.000	-0.189	-0.151
DragonValue	0.0594	0.054	1.103	0.270	-0.046	0.165
GoldIn10	-0.0003	6.49e-05	-4.448	0.000	-0.000	-0.000
GoldIn20	-6.053e-05	2.71e-05	-2.235	0.025	-0.000	-7.44e-06
Herald	-0.1321	0.084	-1.568	0.117	-0.297	0.033
BaronValue	0.2153	0.094	2.300	0.021	0.032	0.399
InhibitsValue	-0.2859	0.074	-3.878	0.000	-0.430	-0.141
TowerValue	1.8170	0.064	28.447	0.000	1.692	1.942
kill	0.0003	0.010	0.033	0.974	-0.020	0.020

Fig 2. Logical regression

To win a game, gold in 10, gold in 20 are important features, and so are the Baron value, the inhibits value, and the tower value. For every unit change in Baron, the log odds of being a winner will decrease by 0.2153 For every unit change in Tower I knocked down, the log odds of being a winner will decrease by 1.8170.
We used the model we trained to predict the result of our test data. And we got the accuracy of 0.96. That's very high.

To see if there are some relationship between the independent values, we

calculated for the correlation between different independent variables.

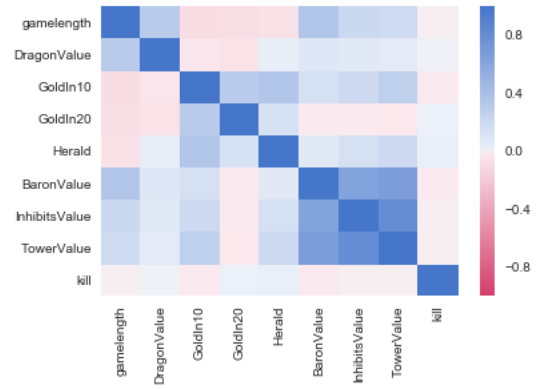


Fig 3. Heat map for different coefficients in this game

Here we can see, the baron value, the inhibits value and the tower value have some effect on each other but not large. There is no noticeable relationship between the other independent values.

2. The blue team's advantage according to time

A popular belief is that, the blue team have some advantages over the red team at the beginning of the match, so they are more likely to win at the beginning. We calculated the win rate of the blue team in different matches of different game length and compared it with 0.5. The result is shown as below.

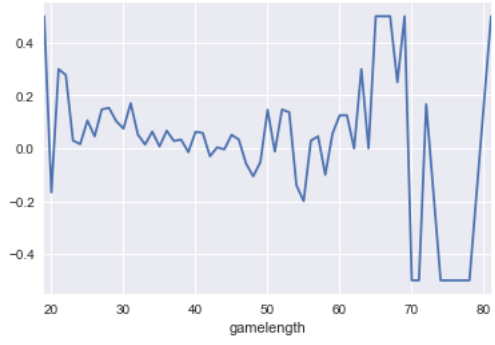


Fig 4. Blue win rate according to time

As we can see, during the beginning 50 minutes, the winning rate of blue team is slightly higher than 0.5, so it does have an advantage over the red team at the beginning. As time increases, this advantage disappears.

3. Pick Champions

We first made the statistic of the most popular champions in different location of the game(*Top,Jungle,Middle,ADC,Support*).

Top:

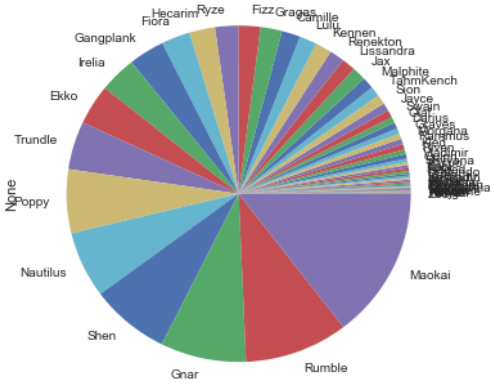


Fig 5. Pie chart of popular champions in Top

As we can see *Maokai, Rumble, Gnar, Shen, Nautilus* are the top five popular champions for Top.

We did the following steps to find the champion that has some advantage over the five champions above.

We first selected all the matches that *Maokai* takes part and grouped them by the Top champion name of the other team. And then we selected all the matches that *Maokai* took part when the team lose the match and again grouped them by the champion name of the other team. Later we divided them to calculate for the win rate of these champions over *Maokai.*

Considering the rigor of statistic, we filtered the champions that fight with *Maokai* for less than five times.

The result we got is shown below.

	count(*)	win_rate
Malphite	6	0.833333
Sion	18	0.666667
Gragas	27	0.592593
Illaoi	7	0.571429
Singed	9	0.555556
Swain	9	0.555556
Ekko	74	0.540541
Irelia	30	0.533333

Fig 6. Champions win rate over Maokai in Top

According to the result, *Malphite* and *Sion* are powerful rivals of *Maokai*. In the step of picking champions, if one side choose to use *Maokai*, the other side will have an advantage if them pick *Malphite* or *Sion*.

We did the similar work for all of the top five champions in Top. And the results are shown below.

	count(*)	win_rate
Kassadin	6	0.833333
Riven	10	0.700000
Camille	14	0.642857
Ryze	14	0.571429
Jax	7	0.571429
Hecarim	27	0.555556
Poppy	9	0.555556
Fizz	21	0.523810

Fig 7. Champions win rate over Rumble in Top

	count(*)	win_rate
Yasuo	7	0.714286
Poppy	12	0.666667
Jax	8	0.625000
Gangplank	34	0.617647
Fiora	10	0.600000
Lulu	20	0.600000
Maokai	104	0.557692
Hecarim	29	0.551724

Fig 8. Champions win rate over Gnar in Top

	count(*)	win_rate
Ryze	6	0.666667
Ekko	16	0.625000
Trundle	41	0.609756
Fizz	7	0.571429
Rumble	38	0.552632
Nautilus	62	0.548387
Fiora	13	0.538462
Gangplank	58	0.534483

Fig 9. Champions win rate over Shen in Top

	count(*)	win_rate
Gnar	8	0.750000
JarvanIV	6	0.666667
Trundle	9	0.666667
Maokai	135	0.511111
Gangplank	10	0.500000
Rumble	46	0.500000
Gragas	35	0.485714
Poppy	75	0.480000

Fig 10. Champions win rate over Nautilus in Top

Jungle:

We made similar analysis for Jungle as we did for Top.

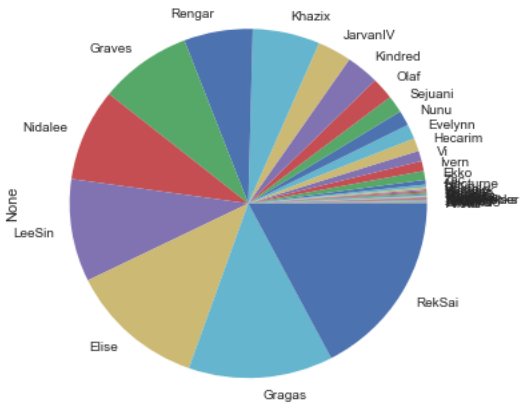


Fig 11. Pie chart of popular champions in Jungle

RekSai, Gragas, Elist ,LeeSin and *Nidalee* are popular champions for jungle.

	count(*)	win_rate
Kindred	47	0.553191
Graves	38	0.552632
Nidalee	87	0.517241
Rengar	18	0.500000
Olaf	8	0.500000
LeeSin	56	0.482143
RekSai	386	0.474093
Khazix	11	0.454545

Fig 12. Champions win rate over Gragas in Jungle

	count(*)	win_rate
Ivern	6	0.833333
Rengar	30	0.600000
Olaf	21	0.571429
Hecarim	16	0.562500
Gragas	167	0.550898
RekSai	245	0.546939
LeeSin	110	0.527273
Ekko	6	0.500000

Fig 13. Champions win rate over Elise in Jungle

	count(*)	win_rate
Sejuani	10	0.800000
Evelynn	14	0.642857
JarvanIV	80	0.575000
Olaf	27	0.555556
Nidalee	43	0.534884
Rengar	62	0.532258
Gragas	56	0.517857
Khazix	114	0.517544

Fig 14. Champions win rate over LeeSin in Jungle

	count(*)	win_rate
Elise	104	0.576923
Evelynn	8	0.500000
Gragas	87	0.482759
LeeSin	43	0.465116
Hecarim	9	0.444444
Kindred	40	0.425000
RekSai	114	0.377193
Graves	56	0.303571

Fig 15. Champions win rate over Nidalee in Jungle

Middle:

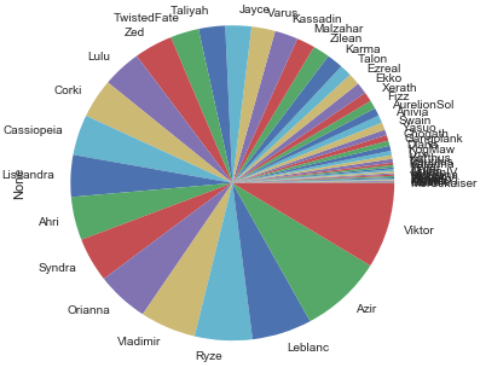


Fig 16. Pie chart of popular champions in Middle

As we can see, *Viktor, Azir, Leblanc, Ryze* and *Vladmir* are the popular champions in this match.

	count(*)	win_rate
KogMaw	6	1.000000
Zed	10	0.900000
Corki	22	0.727273
Syndra	10	0.700000
Zilean	28	0.678571
Malzahar	15	0.666667
Quinn	6	0.666667
Swain	12	0.666667

Fig 17. Champions win rate over Victor in Middle

	count(*)	win_rate
Fizz	9	0.666667
Vladimir	24	0.666667
Varus	44	0.590909
Leblanc	68	0.573529
Cassiopeia	25	0.560000
Ryze	34	0.558824
KogMaw	11	0.545455
Zed	15	0.533333

Fig 18. Champions win rate over Azir in Middle

	count(*)	win_rate
Kassadin	28	0.750000
Lulu	24	0.625000
Zed	18	0.611111
Karma	7	0.571429
Morgana	7	0.571429
Cassiopeia	7	0.571429
Corki	29	0.551724
Zilean	11	0.545455

Fig 19. Champions win rate over Leblanc in Middle

	count(*)	win_rate
Ahri	8	1.000000
AurelionSol	8	0.875000
Malzahar	10	0.800000
Talon	7	0.714286
Zed	6	0.666667
Lulu	7	0.571429
Syndra	26	0.538462
Taliyah	13	0.538462

Fig 20. Champions win rate over Ryze in Middle

	count(*)	win_rate
Talon	6	0.666667
Malzahar	21	0.619048
Anivia	38	0.605263
Orianna	17	0.588235
Syndra	19	0.578947
Taliyah	14	0.571429
Cassiopeia	32	0.562500
Ryze	27	0.555556

Fig 21. Champions win rate over Vladimir in Middle

ADC:

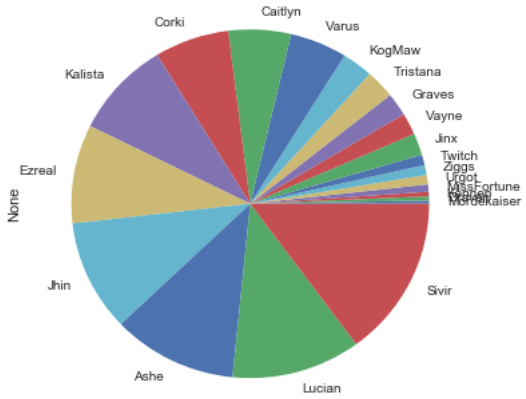


Fig 22. Pie chart of popular champions in ADC

As we can see, *Sirvir, Lucian, Ashe, Jhin*, and *Ezreal* are the popular champions at ADC in this match.

	count(*)	win_rate
Twitch	18	0.611111
Kalista	103	0.514563
Vayne	66	0.500000
Jhin	114	0.500000
Urgot	20	0.500000
Varus	16	0.500000
Ashe	132	0.484848
Jinx	51	0.470588

Fig 23. Champions win rate over Sivir in ADC

	count(*)	win_rate
Graves	14	0.714286
Jhin	60	0.600000
Kalista	133	0.593985
Twitch	17	0.588235
Vayne	12	0.583333
MissFortune	7	0.571429
Sivir	229	0.563319
Corki	102	0.529412

Fig 24. Champions win rate over Lucian in ADC

	count(*)	win_rate
Kalista	28	0.571429
Ziggs	9	0.555556
Varus	137	0.540146
Sivir	132	0.515152
Ezreal	146	0.486301
Lucian	65	0.476923
Caitlyn	64	0.468750
Jhin	224	0.450893

Fig 25. Champions win rate over Ashe in ADC

	count(*)	win_rate
Kalista	10	0.600000
Ashe	224	0.549107
Caitlyn	63	0.507937
Tristana	8	0.500000
Ziggs	16	0.500000
Sivir	114	0.500000
KogMaw	6	0.500000
Ezreal	121	0.421488

Fig 26. Champions win rate over Jhin in ADC

	count(*)	win_rate
Corki	43	0.581395
Jhin	121	0.578512
Kalista	59	0.576271
Caitlyn	54	0.555556
Ziggs	11	0.545455
Sivir	81	0.543210
Varus	26	0.538462
Lucian	159	0.534591

Fig 27. Champions win rate over Ezreal in ADC

Support:

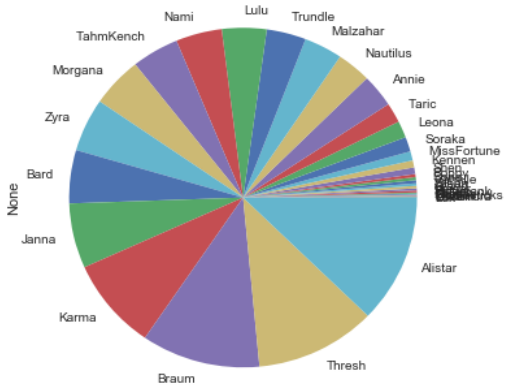


Fig 28. Pie chart of popular champions in Support

As we can see, *Alistar, Thresh, Brarum, Karma, Janna* are the popular champions in this match.

	count(*)	win_rate
Malzahar	6	1.000000
Zyra	8	0.875000
TahmKench	45	0.666667
Nami	11	0.636364
Shen	16	0.625000
Leona	12	0.583333
Lulu	7	0.571429
Thresh	164	0.548780

Fig 29. Champions win rate over Alistar in Support

	count(*)	win_rate
TahmKench	26	0.692308
Shen	11	0.636364
Kennen	8	0.625000
Morgana	59	0.593220
Bard	39	0.564103
Trundle	19	0.526316
Annie	63	0.523810
Leona	27	0.481481

Fig 30. Champions win rate over Thresh in Support

	count(*)	win_rate
Taric	36	0.666667
Nami	30	0.666667
Nautilus	18	0.666667
Janna	54	0.666667
Soraka	26	0.615385
Morgana	28	0.571429
Bard	97	0.546392
Alistar	222	0.545045

Fig 31. Champions win rate over Brarum in Support

	count(*)	win_rate
Taric	8	0.750000
Nautilus	16	0.625000
TahmKench	31	0.612903
Lulu	97	0.608247
Thresh	63	0.587302
Alistar	31	0.580645
Sona	9	0.555556
Bard	50	0.540000

Fig 32. Champions win rate over Karma in Support

	count(*)	win_rate
Annie	39	0.666667
Karma	16	0.625000
Thresh	95	0.547368
Morgana	43	0.534884
Lulu	10	0.500000
Alistar	91	0.472527
Nautilus	38	0.447368
Leona	16	0.437500

Fig 33. Champions win rate over Janna in Support

The aim of the above analysis is to find a proper choice for players to get a counter pick against the enemy's choice, as the last enemy's selected champion can be seen in the ban/pick phase before the game begins. To implement this, here follows the example.

	count(*)	win_rate
Elise	104	0.576923
Evelynn	8	0.500000
Gragas	87	0.482759
LeeSin	43	0.465116
Hecarim	9	0.444444
Kindred	40	0.425000
RekSai	114	0.377193
Graves	56	0.303571
JarvanIV	15	0.200000

Fig 34. Champions win rate over Nidalee in Jungle

During the match in tournaments matches, *Elise* can be a good choice to counter *Nidalee* as a jungle position pick. *Elise* is a great popular Jungle champion with a high rate of selection as well. She is quite strong and aggressive in the early game phase, especially in scenarios like quick-gank and jungle encounter fight. She is able to kill other champions before level of 6 due to her special skill mechanisms of high outbreak of damage, which is obviously easy to take advantage to champions willing to farm and get rid of early encountering such as *Nidalee.*

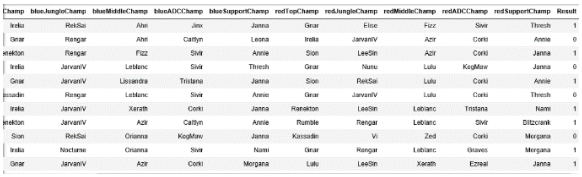
4. Using Log5 to find the possibility a character wins another character:

Log 5 is a formula invented by Bill James to estimate the probability that team A will win a game, based on the true winning percentage of Team A and Team B.
The Log5 estimate for the probability of A defeating B is :

$$p_{A,B} = \frac{p_A - p_A \times p_B}{p_A + p_B - 2 \times p_A \times p_B}.$$

In our analysis we used Log5 to find the wining rate between two champions by first calculating for the winning rate of the specific champion among all (we defined the champion to win by the team result, like if Rumble showed in two matches in blue side, and in these two matches the blue side wins then the winning rate of Rumble is 100%) and later calculate their win rate between each other.

We first got the Champion participated in every match from both sides and the result of these matches. The graph is showing as below (Columns contain the champion in each location and the result).



Champ	blueJungleChamp	blueMiddleChamp	blueADCChamp	blueSupportChamp	redTopChamp	redJungleChamp	redMiddleChamp	redADCChamp	redSupportChamp	Result
Irelia	RekSai	Ahri	Jinx	Janna	Gnar	Elise	Fizz	Sivir	Thresh	1
Gnar	Rengar	Ahri	Caitlyn	Leona	Irelia	JarvanIV	Azir	Corki	Annie	0
enekton	Rengar	Fizz	Sivir	Annie	Sion	LeeSin	Azir	Corki	Janna	1
Irelia	JarvanIV	Leblanc	Sivir	Thresh	Gnar	Nunu	Lulu	KogMaw	Janna	0
Gnar	JarvanIV	Lissandra	Tristana	Janna	Sion	RekSai	Lulu	Corki	Annie	1
assadin	Rengar	Leblanc	Sivir	Annie	Gnar	JarvanIV	Lulu	Corki	Thresh	0
Irelia	JarvanIV	Xerath	Corki	Janna	Renekton	LeeSin	Leblanc	Tristana	Nami	1
enekton	JarvanIV	Azir	Caitlyn	Annie	Rumble	Rengar	Leblanc	Sivir	Blitzcrank	1
Sion	RekSai	Orianna	KogMaw	Janna	Kassadin	Vi	Zed	Corki	Morgana	0
Irelia	Nocturne	Orianna	Sivir	Nami	Gnar	Rengar	Leblanc	Graves	Morgana	1
Gnar	JarvanIV	Azir	Corki	Morgana	Lulu	LeeSin	Xerath	Ezreal	Janna	1

Fig 35. Match records with champion information

We suppose that champions appear in matches for less than 30 times are champions with low using frequencies, and we remove the champions from our analysis.

Here is part of the win rate of every champion.

Rammus	0.666667
Aurelionsol	0.647059
Zed	0.639810
Zac	0.612903
Taric	0.590909
Camille	0.590476
Vi	0.587302
Malzahar	0.585227
Riven	0.567568
Quinn	0.561404
Urgot	0.560976
Soraka	0.558140
Twitch	0.558140
Kalista	0.551959
Kassadin	0.546729
Nidalee	0.546314
Fizz	0.541126
Ivern	0.540541
Tahmkench	0.540059

Fig 36. Champions win rate of all the champions

We chose the top five champions with the highest winning rate, and the last five champions with the lowest wining rate and calculated for the winning rate of them between each other using Log5:

	Rammus	Aurelionsol	Zed	Zac	Taric	Renekton	Xerath	Evelynn	Lux	Drmundo
Rammus	0.5	0.521739	0.529617	0.55814	0.580645	0.736364	0.740157	0.745763	0.75	0.871287
Aurelionsol	0.478261	0.5	0.507898	0.536585	0.559322	0.719128	0.723077	0.728916	0.733333	0.86121
Zed	0.470383	0.492102	0.5	0.528721	0.551521	0.712703	0.716706	0.722628	0.72711	0.85739
Zac	0.44186	0.463415	0.471279	0.5	0.522936	0.688591	0.692785	0.698997	0.703704	0.842742
Taric	0.419355	0.440678	0.448479	0.477064	0.5	0.668571	0.672907	0.679335	0.684211	0.830189
Renekton	0.263636	0.280872	0.287297	0.311409	0.331429	0.5	0.504908	0.512244	0.517857	0.707906
Xerath	0.259843	0.276923	0.283294	0.307215	0.327093	0.495092	0.5	0.507338	0.512953	0.703829
Evelynn	0.254237	0.271084	0.277372	0.301003	0.320665	0.487756	0.492662	0.5	0.505618	0.697674
Lux	0.25	0.266667	0.27289	0.296296	0.315789	0.482143	0.487047	0.494382	0.5	0.692913
Drmundo	0.128713	0.13879	0.14261	0.157258	0.169811	0.292094	0.296171	0.302326	0.307087	0.5

Fig 37. Winning rate between the strongest five champions and the weakest five champions.

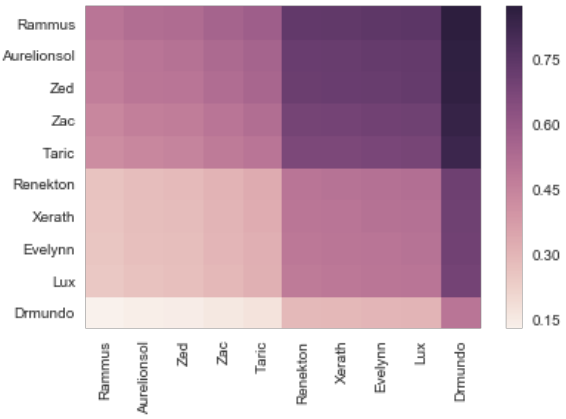


Fig 38. Heat map of winning rate between each pair of the five strongest champions and the five weakest champions

Here we can see that the possibility of *Rammus* to win *Drmundo* is higher than 0.85 which is large. The winning rates of *Renekton, Xerath, Evelynn* and *Lux* between each other are around 0.5 which means that the level difference among these champions are not significant. *Drmundo* is a weak champion, the power difference between

Drmundo and the other four last powerful champions is not neglectable.

5. Using K-Nearest Neighbors to predict Match Results

A popular belief among LOL players is that the champions chosen in a match have effect on the match result. A team with a good composition is more likely to win the game. We used the K-nearest neighbors for classification and regression that predicts the match result. We chose to use the KNN in the consider that champions may have some talents and qualities complementary towards other champions so we need to take them as a group instead of merely calculating about the attendance of each champion.

To achieve our goal, we built an array with 262 columns with each column representing a champion in the game as there were 131 champions occurred in our dataset and we used different columns to represent if this champion was in the red side or in the blue side (like if the first character participated in a match in the blue side, then the first column in that row was set to '1', and if it participated in the red side, then the 1+131 = 132 column in this row was set to '1'. The array has 3802 rows representing the 3802 matches in our dataset. In total, each row had 10 cells set to 1 with five of them on the left side and five of them on the right side.

How our array looks like:

```
[array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  1.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,
         0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
         0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
```

Fig 39. Example of the array in KNN

The following is the code we used to build our array.

```
def fun(row):
    new_row = np.zeros(263)
    new_row[heros.index(row[0])] = 1
    new_row[heros.index(row[1])] = 1
    new_row[heros.index(row[2])] = 1
    new_row[heros.index(row[3])] = 1
    new_row[heros.index(row[4])] = 1
    new_row[heros.index(row[5]) + 131] = 1
    new_row[heros.index(row[6]) + 131] = 1
    new_row[heros.index(row[7]) + 131] = 1
    new_row[heros.index(row[8]) + 131] = 1
    new_row[heros.index(row[9]) + 131] = 1
    return new_row
```

Fig 40. Function we used in KNN

We divided our data into training data and testing data.

We trained our model using K-Neighbors Classifier in sklearn.neighbors. We used the uniform weight and auto algorithm to train the model.

The following is the code we used to get our model.

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(weights='uniform', algorithm='auto').fit(X_train, y_train)
score=model.score(X_test, y_test)
score
```

Fig 41. The model get from KNN

To test our model, we used our test data. The following is our accuracy.

The accuracy of our model to predict our test model is about 0.5, and the accuracy to predict our training data itself is 0.7.

```
score1=model.score(X_array_test, Y_array_test)
score1
```

```
0.49014454664914586
```

```
score2=model.score(X_array_train, Y_array_train)
score2
```

```
0.70009865175928976
```

Fig 42. Accuracy of our KNN model

We came to the conclusion that LOL was a well-designed game. The fighting ability of different composition of champions are at a balanced level. The champions picked don't have a noticeable effect on the match result.

Conclusion

As the data analysis part shown, some obvious trend and results of analysis could be addressed by comparing specific attributes of dataset. Such as the gold proportion rate to each position and their win rate. The above analysis could be informative to players focusing on how to win a game more effectively, especially for those professional e-sport teams.

The other part of prediction/pick is to make a counter pick selection guide against a specific champion selected by the enemy. This could be quite practical and useful for players at ban/pick phase, especially those who are not that familiar with all the champions' features and abilities.

We tried to train a model to predict the match result according to the composition of both side of the team and thus help our players to make decisions. However, this model turned out to be not very helpful with an accuracy of about 0.5. We think part of the reason is that our data comes from professional games, so the team composition are more likely to be well-organized in both side. Besides, the LOL is a well-designed game so the power levels of the champions are balanced at an acceptable level.

Acknowledgment

We would like to show our gratitude to professor Nik Bear Brown for guiding us and encouraging us during the course of this project.

References

[1]. kaggle.com League of Legends: https://www.kaggle.com/chuckephron/leagueoflegends (Professional Tournament) https://www.kaggle.com/lanls1/matchidv1
[2]. Riot developer API: https://developer.riotgames.com
[3]. KNN algorithm-wiki: https://en.wikipedia.org/wiki/Knearest_neighbors_algorithm
[4]. KNN algorithm specification: http://blog.csdn.net/u011067360/article/details/23941577
[5]. LOL stats: http://www.lolking.net
[6]. LOL champions ranking: http://champion.gg/statistics/