LEAGUE OF LEGENDS REAL-TIME WINNING RATE ANALYSIS

**Hongyi Liu Houze Liu**

[hl2638@nyu.edu](mailto:hl2638@nyu.edu) [hl2979@nyu.edu](mailto:hl2979@nyu.edu)

ABSTRACT

League of Legends and its professional games have been popular for almost a decade. We tried to recreate the functionality of Captain KI, a real-time winning rate predicting robot, without any open-source code from them. Acquiring the data through Riot Games’ API with some necessary manual processing on the data, we trained logistic regression, random forest and support vector machine and eventually selected random forest as our prediction model. The results are satisfying to us, not only in predicting game results but also winning rates, which is meaningful to real-life scenarios.

1 INTRODUCTION

For many years League of Legends has been a popular video game worldwide. A large variety of techniques and tactics have been developed and replaced as the game itself updates with new versions and patches. The game is now of significant commercial value and popularity, which is partly contributed to by the professional leagues and tournaments. Consequently, game analysis has earned increasing importance and attention meanwhile. Inspired by “Captain KI (KI上校)”, we seek to recreate a similar winning-rate analysis model.

1.1 What is Captain KI and how is it related to our project?

Captain KI is an AI figure sponsored by KFC China (hence the name) based on big data analysis algorithms. During S8, he predicted real-time winning rate as the matches were being played. The model is trained with a huge amount of games statistics from tournaments matches and high-elo (high-level) games on the Korean server (“PentaQ” 2018). While we do not have any official support, we try to realize this functionality as much as possible. Our goal is: given game time and statistics, we will be able to predict how likely one team is going to win.

2 OBTAINING DATA

While Captain KI has support from Riot LA, Riot China, and Tencent Esports Team (“PentaQ” 2018), and we have neither funding nor database available, we still can fulfill our little machine learning project starting with requesting data from Riot Games. Through their API we could get data ranging from player ID to match timelines. We also employed an open-source package, Riotwatcher, to manage API requests in Python.

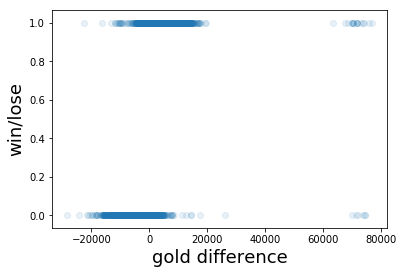
Our aim was to obtain enough matches that: 1) are of a high level (ranked games of master/challenger level on KR/NA server), 2) are played under Patch 8.19 which was the patch used in the S8 World Finals, 3) are “regular” (no games lasting less than 16 minutes, which implies exceptional situations).

We used a breadth-first-search method to get the games: first start from Faker’s (legendary Korean player) account. Once we have his match history, search the time range of the games in Patch 8.19. From the games Faker played on Patch 8.19, pick one latest game as the seed. Get the other 9 players from that game and keep doing the BFS search until we have got enough matches. In reality, the requests took a considerable amount of time. We changed to pick 5 games per player’s game history instead of 1, so as to improve the efficiency of each request.

3 DATA ANALYSIS AND PREPROCESSING

The raw data we obtained came in 2 types: matches and timelines. Both contained complicated, nested data. See reference list for more information. In short, matches contain general information and statistics of the match while timelines record events and some of the statistics by time frames.

Take a look at the relationship between game result and gold difference at different game lengths:

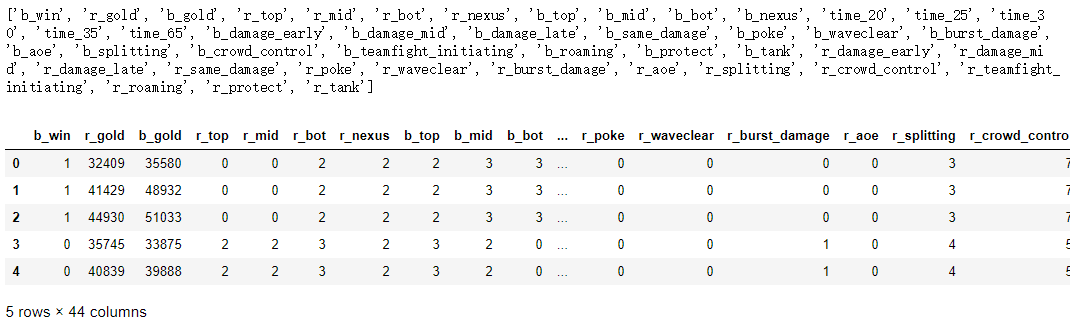


One can see the majority of the games are won with a gold lead. There are some cases where the team behind turned the tide. When it comes to late-game, gold difference has a slightly lesser effect on game results but they are still correlated. We also assume that turrets destroyed has an influence as it not only increases gold (existent feature) but also represses enemy team’s vision.

The features we need for training is mainly: current time, gold and gold difference between the two teams, turrets destroyed and team composition. All above can be extracted from the raw data except team composition, which needs to draw champion features while some champion features cannot be measured by statistics.

Some of the champion features, such as damage per minute, can be calculated from raw data. Others, such as ability to protect or roam (travel to assist allies) could only be manually rated. We consulted Yikun Zhou, S7 CN server challenger for necessary information and manually generated a set of champion-feature data. To avoid linear influence, all abilities are either rated as 0/1 or 0/1/2. For example: Ziggs is tier 1 at wave clearing so he is rated 2 at this field. Xayah has almost no roaming ability so rated 0 at roaming.

The final pandas.Dataframe is too “wide” to display so below is a screenshot of the keys and part of the Dataframe.head(). Prefixes “b” and “r” represent blue and red team. “Same damage” indicates whether the team deals physical (or magical) damage only. “Time\_x” means whether the game is at a certain period. “top/mid/bot” means how many turrets are left on each lane.



Observing the dataset we find that since the team composition, gold and tower are all continuous (though not necessarily linear) and not normally distributed. Thus it seems a good choice to apply the min-max scaler on these columns can keep the features. Meanwhile, game time has been one-hot-encoded into 5 frames: whether it is before 20 minutes, 20-25 minutes, etc.

We have also attempted to use standardization and PCA method due to the high dimensions. However, the variance of normalized data which did not concentrate implied that most features cannot be reduced to a lower dimension.

4 METHODS

Defining blue team’s victory as 1 and the opposite, 0, this has become a classification problem, which leads us to logistic regression. What it also provides is a sigmoid function value which indicates the probability of winning. It perfectly serves our demand.

In addition, we tried Random Forest and Support Vector Machine. The probability output of random forest is trickier: If 7 trees voted yes and 3 trees voted no, the probability is 0.7. Therefore we set the number of estimators to 1000 to get a precision with three digits of decimals. For SVM, the probability function is another sigmoid function: “In the binary case, the probabilities are calibrated using Platt scaling: logistic regression on the SVM’s scores, fit by an additional cross-validation on the training data.” (Scikit-learn documentation)

(Platt)

We also attempted to train a model using keras, to see if it can do better. Similarly, the probability outcome is determined by a sigmoid function on the last layer.

We realized that we did not need to modify the loss functions, since essentially we were to predict a binary outcome with both results equally important i.e. it makes no difference whether it is a false-1 or a false-0. However, we used a new metric for accuracy, so we can have better sense of the performance and real-life meaning: in their answer to the question regarding Captain KI, the KI developer team showed their predicting accuracy as the picture below:

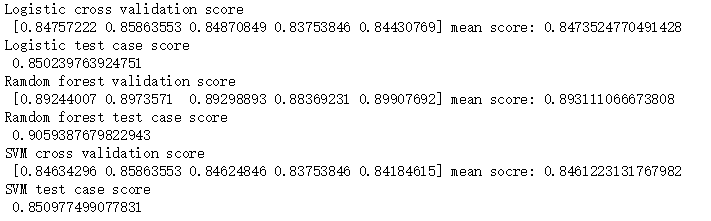


The columns “x%胜率” indicate whether the team actually won when a winning rate >= x% was predicted. From this we came up with a new metric for accuracy: x+% accuracy, meaning the accuracy of the predictions with estimate >= x%.

Another point we noticed is that for humans usually it is hard to tell which game will win in the early/mid phase of the game. Therefore we also made a metric for this: 15-min accuracy, namely the accuracy of predictions at 15 minutes after game started.

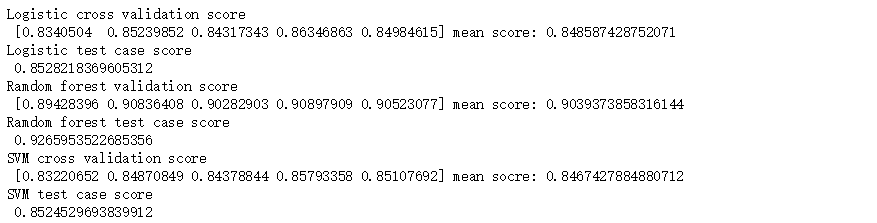
5 RESULTS

We tried training our models without team composition data, that is, the models only cared about the situations in the game, having no idea which champions are in the game.



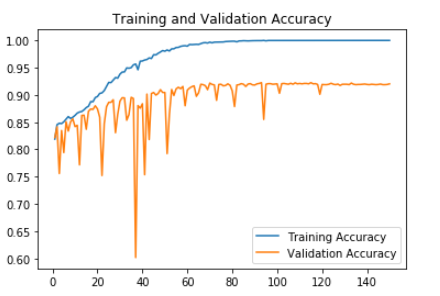
This result is delighting, since even without knowledge of the team composition, we are still able to be 80+% right in general about which team is going to win.

Add the team composition feature and we train again:



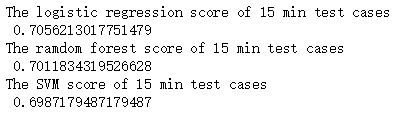
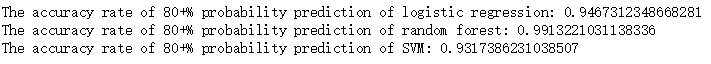
Each model improved by a little bit. Technically, team composition has a large effect in the game result. As is often mentioned by professional players and commentators a good combination of champions is worth several thousands of gold. Obviously, our interpretation of champion features and team composition was not comprehensive and accurate enough; such features could have improved the model much more if processed properly and if given sufficiently large data.

For neural network, after more layers and nodes added, we eventually achieved about 92.03% accuracy.



Random forest performs slightly better than neural network and is much faster to train. Therefore we decide to employ random forest.

Take a look at the 80+% accuracy and the 15-min accuracy of the 3 models:



The results are generally very good and confirms random forest as our choice. Even at 15 minutes we are roughly 70% right. Why is 70% a good result? Below is the mathematical proof.

This means that if we predicted all winning rates completely correct, the best accuracy we can get is the average of the real winning rates of the leading team. We estimated this value by calculating the average predicted winning rate of the leading team.

https://lh3.googleusercontent.com/9_xRd59-mz5tdyvP4l_y3Y99R5yUMu3NdIb0rmr4SAHAp8goPC8qFdY-CnXQ3TqVPdbtjHNhzheuVEZ6FGpprLjFm2UOl5RBPITgvEl1_IFio1PbYuxFfuBw6Ie21VBl0jqZGdHd

The average predicted winning rate is 0.678. Assume that our model is not robust enough and that the average(P) is higher. We have the reason to believe that average(P) is far from 80%, because otherwise the accuracy would approach 99%. Therefore we have that , average(P) being equal to the best we can do in theory, while our accuracy is 70%, implying that though our model is (of course) flawed, we managed to acquire a good result.

6 CONCLUSION, REMARKS AND DISCUSSION

Our model, to say the least, can fulfill our primary goal: given game time, team compositions, gold and turret data, it can predict a value which represents the winning rate of the blue team (and thus predict the winning rate of the red team). Moreover, in terms of the binary result: victory or loss, it can be 90% of the time correct. Whenever there is a prediction made that one team is more than 80% likely to win, 99% of the time it is correct. We also have reason to believe that at 15 minutes, when the results of most games are not so easy to predict, it is still doing a good job (70%).

We discovered that one can be quite accurate when they predict the game result simply by gold difference and turrets destroyed. This corresponds to our preliminary observation as well. We could have done better, though, if we had more access to champion data and had wiser ways to process them. A more advanced team-composition analysis would enable us to achieve even more: Captain KI came up with more functionalities such as pre-game team-strength ratio and team-strength curves over time.

ACKNOWLEDGEMENT

I (Hongyi) would like to thank Yikun Zhou, NYU Shanghai class of 2021, and my roommate, who ranked challenger in Season 7’s Chinese server and whom I asked about champion ratings.

Though we did not receive any sponsorship from either PentaQ Team or KFC, we should thank them for making Captain KI which inspired us.

REFERENCES

“PentaQ刺猬电竞社.” Answer on “如何评价英雄联盟S8全球总决赛当中KI上校的预测？”(How to comment Captain KI’s predictions on League of Legends S8 World Finals), zhihu.com. <https://www.zhihu.com/question/296926812/answer/508711738>. Accessed on Dec 6 2018.

Riotwatcher Documentation. <https://riot-watcher.readthedocs.io/en/latest/>. Accessed Dec 6 2018.

Riot Developer Portal. <https://developer.riotgames.com/>. Accessed Dec 6 2018.

Scikit-learn documentation on support vector machine. <https://scikit-learn.org/stable/modules/svm.html#scores-probabilities>. Accessed Dec 6 2018.

Pedregosa *et al.* [Scikit-learn: Machine Learning in Python](http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html), JMLR 12, pp. 2825-2830, 2011.

John C. Platt. *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*. 26 March, 1999

Chollet, Fran et al. *Keras*, 2015