Can we predict LoL Game Results?
Final Report

Yuhan Li, Anqi Ren, Jiang Zhu

1. Introduction

This project is about game result analysis with data of League of Legends. League of Legends (LoL) is a multiplayer online battle arena video game, which provides hundreds of champions and each with unique ability and battle skills. There are two teams with five members each in one play, and the ultimate goal of the play is to destroy the nexus of the opposing team. LoL is a very popular video game, which is released in 2009. Until 2016, there are over 100 million active players estimated each month. [1] Today, even the number of active players is slightly dropped, there are still over 80 million people every month playing this game. [2] Therefore, the project would have a very broad range of audiences considering the large amount of players worldwide. The project could help general game players to prioritize their strategies.

The project aims to achieve two major goals. Firstly, the team would like to identify and rank the most relevant features that lead to a win in a LoL game match using a dataset from Kaggle. There are multiple variables in a game that can affect the game result. Sometimes, players need to choose their priorities among different variables in order to win the game, and this project can help them make quick decisions during the play. Secondly, the team would like to choose the most accurate model that can predict the final result with the smallest error rate.

The team mainly focuses on proximal gradient method with different loss functions and regularizers, classification tree, random forest, and support vector machine (SVM). The proximal gradient method selects the best combination of feature based on the misclassification rate of game result. Other models are used to predict the game result with training and test data, and the team will choose the best model with the smallest prediction error.

2. Data Description

The dataset consists of 51,536 ranked EUW (European Server) games from the LoL along with two json files containing the champion IDs, names, and summoner spell IDs. During data cleaning process, there is no missing data or duplicate entries found in the dataset. The dataset is prepared to be used in further analysis. Below is a table describing features in the dataset after cleaning process.

Table 1: Data description After data cleaning

Features	Data Type	Explanation
win	Binary	Win (1) or lose (0)
firstBlood/firstTower/firstInhibitor/firstBaron/firstDragon/firstRiftHerald	Binary	Whether the team got first blood, first tower, first inhibitor, first baron, first dragon, first rift herald (1) or not(0)

towerKills/inhibitorKills/baron Kills/dragonKills/riftHeraldKills	Integer	Number of towers/ inhibitor/ baron/ dragonKills/ riftHeraldKills killed in the game
t_champ_id	Integer	Which champions the 5 team members used
t_ban	Integer	ChampIDs banned by the team

The data column **win** in Table 1 above is the dependent variable and the rest of variables except **t_champ_id a**nd **t_ban** are possible features used to predict the response. Therefore, there are 11 features that are used as independent variables. The original data set stores data per row using gameID as the primary key, which means each row has game data for both winning team and losing team. In this case, gameID, game duration, and seasonID are irrelevant, which are ignored in further analysis. The team separates corresponding variables for winning and losing team and then store them in rows as team, so each row in the data set after cleaning only has game data for winning team or losing team. The cleaned dataset consists of **102980** rows and **31** columns.

In LoL, first blood can be obtained by a champion if its player makes the first kill in the game. It brings 400 extra gold bonus to the champion, which is a great advantage at the beginning of the game. Similarly, first tower/inhibitor/baron/riftHerald/dragon refers to the first kill of tower/inhibitor/baron/riftHerald/dragon in the game regardless of team. After first kill, every kill of tower, inhibitor, baron, dragon, riftHerald, and champion in opposite team brings gold to the teams and champions. The gold earned in the game by team or by champion can be used to purchase items in shop, which provide champions with bonus stats and ability. [3] So gold is supposed to be an important feature. However, the dataset does not include the team gold for every game match, which makes the calculation very difficult. Thus, the team does not include gold relevant analysis in this project.

3. Descriptive statistics

3.1 First Statistics

First statistics indicates whether the team achieve fist blood /tower /inhibitor /baron /riftHerald /dragon in the game. The diagram (Figure 1) shows the average frequency of winning team achieving firstBlood, firstTower, firstInhibitor, firstBaron, firstRiftHerald, and firstDragon. The six independent features are binary variables (1/0) with value 1 indicating the winning team achieves it, and 0 otherwise. Winning team seems to get first inhibitor most frequently, and then first tower, first dragon, and first blood. First baron and first riftHerald appear to be least likely obtained by the winning team.

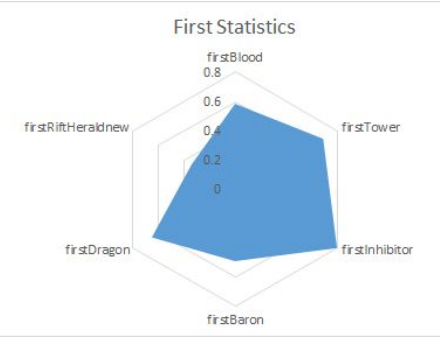


Figure 1: Frequency of winning team achieving first statistics

Figure 2: Winning Probabilities of Teams achieving first statistics

Another way to look at this problem is to calculate the winning probabilities of teams achieving certain first statistics. The formula of Win Probability is:

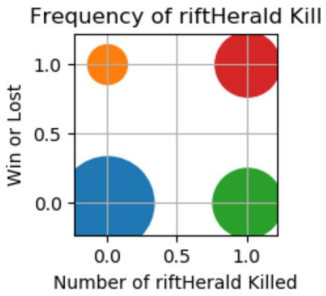
$$P(win|first\ blood) = frequency(win = 1|firstBlood = 1)/frequency(firstBlood = 1)$$

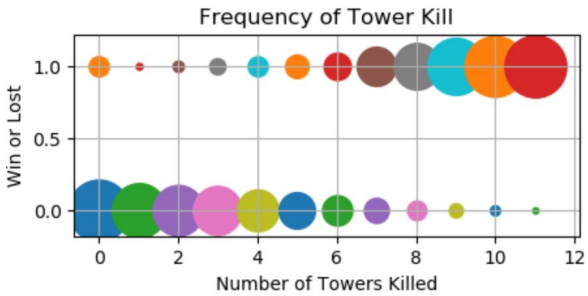
The result (Figure 2) shows that getting the firstInhibitor leads to the maximum win probability of 81%. Other ranked statistics are firstBaron, firstTower, firstRiftHearld, firstDragon. firstBlood has the least probability.

The above analysis can draw a conclusion that firstInhibitor is the most important factor that leads to game wins compared to other first statistics.

3.2 Kill Statistics

Another part of the features is kill statistics which indicates how many kills of certain items does a team achieve in a game. The following plots shows the frequency of kills numbers. The x-axis is number of items killed and the y-axis indicates whether the team wins. The size of the plot shows how many points are located at the values. The more frequently a value appears, the bigger the circle is. The colors has no meanings. From the plots we can concluded that our data is highly classified and there are many entries have the same values.





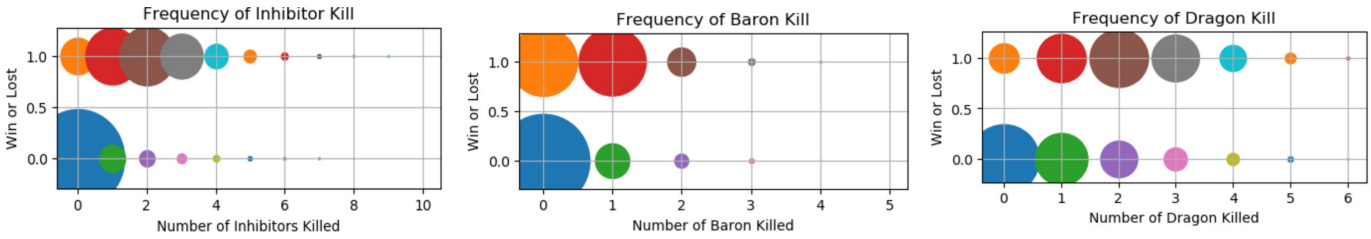


Figure 3: Number of Kills VS Game Win

From the above figures (Figure 3), we can get a general idea of how each variable in game affect the team wins. Winning teams tend to destroy as many towers as possible while many teams kill a lot of towers still lost. For inhibitor, baron and dragon, winning teams kill more of these elements than losing team, but some teams still win when they kill none of inhibitors, barons and dragons. Killing the riftHerald does not guarantee victories either.

3.3 Champion Selection

Another important factor that players might be interested is the champion selection. The purpose of this analysis is that players can know what are the most likely champions selected or banned by their opponents so they can develop appropriate strategies. Although the selections depends on players' preferences, we may still summarize which champions are most favored by players. The following bar chart ranks the top 10 champions selected or banned by players.

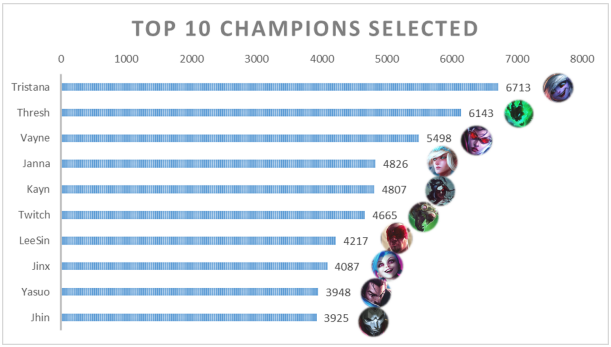


Figure 4: Top 10 Champions Selected

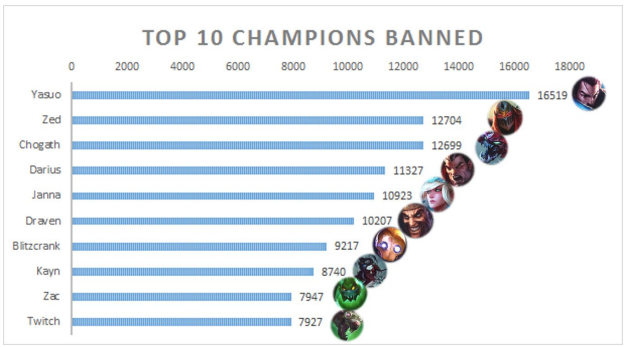


Figure 5: Top 10 Champions Banned

4. Data Modeling

The first goal of project is to find features that will most likely lead to game wins. In our models, the response variable is binary suggesting whether the team win or lose. The other 11 features (first statistic and kill statistics) are independent variables. We use proximal gradient method to select features that are most likely related to response variable. Then we use those features to fit classification tree, random forest, and support vector machine (SVM) models that yield good predictions and use cross validation to show how reliable the models predictions are and show the most suitable model.

4.1 Proximal Gradient Method

We choose proximal gradient method to solve regularized empirical risk minimization problem. The result of coefficient would suggest how each feature is related to the response variable. Adding a regularizers can better generalize our model, prevent overfitting and stabilize estimates. Since our data is highly classified, in order to find the best model, we tried 4 different loss functions (Quadratic Loss, L1 Loss, Hinge Loss and Logistic Loss), and 4 different regularizers (No Regularizer, L1 Regularizer, Quadratic Regularizer, Non-negative Regularizer). In total, we fit 16 models combining each of loss functions and regularizers.

We select the combination of loss function and regularizer that yields the highest accuracy of prediction. To estimate the accuracy of our model prediction, we used misclassification rate. Since the response value is either 0 or 1, we classify the prediction to 1 if the prediction value is closer to 1 than to 0, and otherwise. The misclassification rate indicates how much portion of the prediction is incorrect. We select the model with the least misclassification rate. Cross-validation is used to estimate error rates. The data is split into halves for test set and training set. We use different random seeds to split the data and take the average of error rates as the final error rate.

Table 2: Misclassification Rates with different loss functions and regularizers

	Quadratic Loss	L1 Loss	Hinge Loss	Logistic Loss
No Regularizer	0.1178	0.1198	0.1371	0.1745
L1 Regularizer	0.1127	0.1177	0.1177	0.5007
Quadratic Regularizer	0.1169	0.1193	0.1344	0.1902
Nonnegative Constraint	0.1178	0.1199	0.1381	0.1718

By fitting the 16 models, we obtain misclassification rates as above. The least rate 11.27% is modeled with Quadratic Loss and L1 Regularizer. However, the coefficients of the model is [0, 0, 0, 0, 0, 0, 0.081463, 0, 0, 0, 0, 0]. It means tower_kills dominates in the model. We would like to see more features that are highly related to the response, so we look at the second lowest rate 11.69%, which is modeled with Quadratic Loss and Quadratic Regularizer. The coefficients are [0.00507602, 0.00666604, 0.00932582, 0.00464379, 0.00593433, 0.00294128, 0.0795685, 0.0175619, 0.0053342, 0.01586, 0.00294128, 0.00739838]. Table 3 shows the top 8 features with the highest coefficients. We will use these 8 features in the following models.

Table 3: Coefficients with the best model (ranked)

team_towerKills	team_inhibitorKills	team_dragonKills	first Inhibitor	first Tower	first Dragon	team_baronKills	first Blood	first Baron	first RiftHerald	team_riftHeraldKills
0.07957	0.01756	0.01586	0.00933	0.00667	0.00593	0.00533	0.00508	0.00464	0.00294	0.00294

From the above result, we consider **team_towerKills, team_inhibitorKills, team_dragonKills, firstInhibitor, firstTower, firstDragon, team_baronKills, and firstBlood** as the most relevant features to wins. They are selected for further analysis.

In sections 4.2-4.4, we use k-fold cross validation (k=5) to split training and test set, we also use graphs of Confusion Matrix of true y vs predicted y to show the classifier performance for each class. The misclassification rates are calculated using the same method as in section 4.1.

Before implementing other models, we use the Proximal Gradient Method again for our best combination of regularizers and loss functions (Quadratic Loss and Quadratic Regularizer) with 8 selected models. The overall misclassification rate for Proximal Gradient is **11.0%**. Figure 6 shows the Confusion Matrix for Proximal Gradient. To understand the matrix, we look at the upper left box in Figure 4 as an example (which is the green square box with a number 89%). This is the percentage representing how much of y that are actually 0 are classified as 0, which is a right prediction and is represented in green. Similarly, the pink box next to it represents how much of y that are actually 0 are classified as 1, which is a wrong prediction. (The right part of the graph basically shows the same result as the left part but just in vertical form. Since we use Matlab to do the plot, the two parts are shown together by default.)

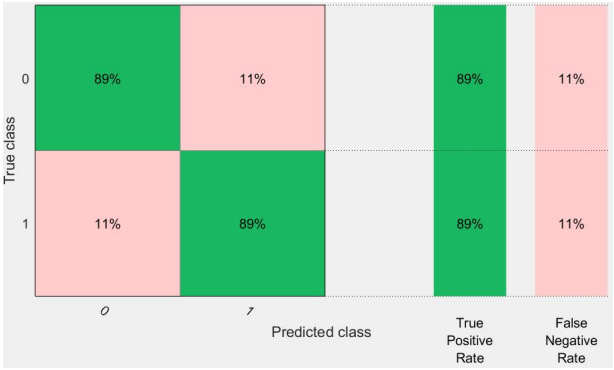


Figure 6: Confusion Matrix for Proximal Gradient with Quadratic Loss and Quadratic Regularizer using 8 features

4.2 Classification Tree

Since our problem is a classification problem, after selecting features, the first model we wanted to try was the Classification Tree. Figure 8 shows the Confusion Matrix for Classification Tree. Classification Tree gives us a misclassification rate of **11.8%.**

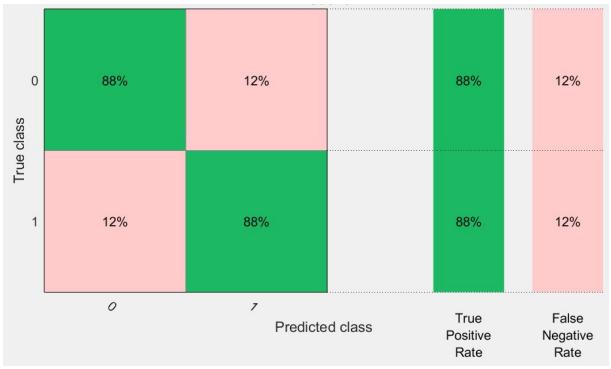


Figure 7: Confusion Matrix for Classification Tree using 8 features

4.3 Random Forest

Since Random Forest corrects for decision trees' tendency of overfitting for training set, we use Random Forest as our second model. Figure 8 shows the Confusion Matrix for Random Forest. Random Forest gives us a misclassification rate of **10.2%**.

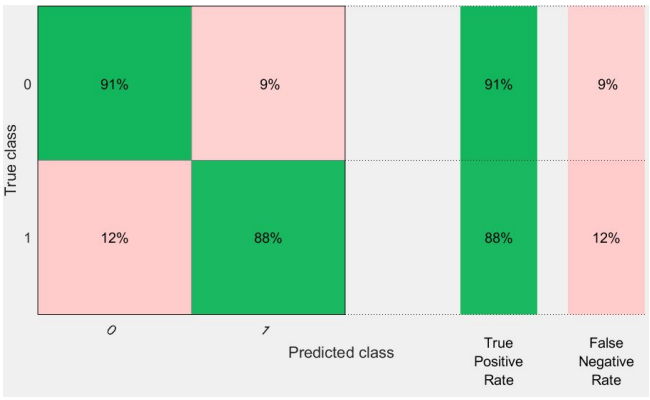


Figure 8: Confusion Matrix for Random Forest using 8 features

4.4 SVM

Another model frequently used in machine learning for classification problem is the Support Vector Machine (SVM), which allows some mistakes and trades off the severity of mistakes with the safety margin. We are using a Linear SVM here. Figure 8 shows the Confusion Matrix Linear SVM. SVM gives us a misclassification rate of **12.9%.**

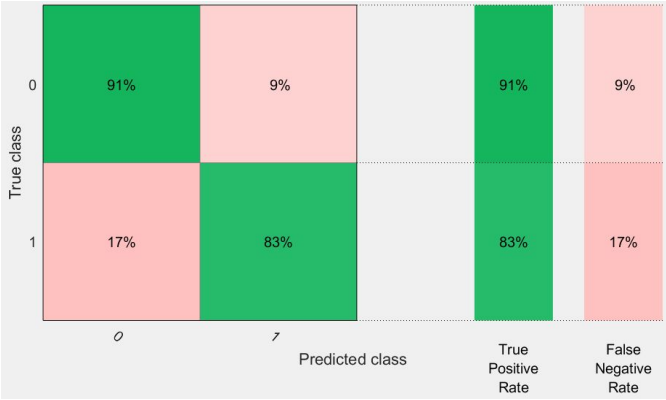


Figure 8: Confusion Matrix for Linear SVM using 8 features

5. Conclusions

5.1 Feature selection interpretations

We concluded from section 4.1 that in decreasing order of priority, the most relevant features that will lead a win in LoL games are team_towerKills, team_inhibitorKills, team_dragonKills, firstInhibitor, firstTower, firstDragon, team_baronKills, firstBlood, firstBaron,

firstRiftHerald and team_riftHeraldKills. Although it may be obvious for LoL players that, for example, they should kill more towers to win the game, the purpose of our study is to show that when players have the opportunity for multiple features, how they should choose to prioritize strategies. For example, firstInhibitor is a more important feature than team_baronKills, which means that when a player or the team has the opportunity to destroy the first inhibitor and to kill a baron, the player or the team should choose to destroy inhibitor rather than killing the baron.

5.2 Model analysis and interpretations

We compare the misclassification rate of Proximal Gradient with Quadratic Loss and Quadratic Regularizer, Classification Tree, Random Forest and Linear SVM to see which model fits the data best. Table 3 shows the results. We can see from Table 3 that Random Forest gives us the best prediction of a misclassification rate of **10.2%**.

Table 3: Misclassification Rates for Proximal Gradient with Quadratic Loss and Quadratic Regularizer, Classification Tree, Random Forest and Linear SVM with 8 features.

Model	Proximal Gradient	Classification Tree	Random Forest	Linear SVM
Misclassification	11%	11.8%	10.2%	12.9%

Overall, we are confident about our results for the following reasons. First, we used methods to avoid overfitting and underfitting To prevent underfitting, we are using a large dataset of 102980 columns. To prevent overfitting, we used a k-fold cross validation (k=2 for feature selection and k=5 for model predictions). We also use random forest trying to improve the overfitting problem for classification tree. Second, the final four models we used had similar misclassification errors, which means they all give reliable results. Third, the interpretations of feature selections basically make sense in terms of real world setting.

6. Bibliography

[1] League of Legends. (2017, September 18). Retrieved November 29, 2017, from https://en.wikipedia.org/wiki/League_of_Legends

[2]You'll Never Guess How Many People Play League of Legends. (n.d.). Retrieved November 29, 2017, from https://www.unrankedsmurfs.com/blog/players-2017

[3]Gold. (n.d.). Retrieved November 29, 2017, from http://leagueoflegends.wikia.com/wiki/Gold