IST 652 Final Project Paper  
How to win the game in League of Legends?

Kun Yang

School of information studies

Syracuse University

Email: kyang23@syr.edu

Abstract

League of Legends is the most popular computer game in the world, and I am a crazy fan of it. The dataset I am going to use contains the first 10 minutes detailed data of approximate 10k ranked games from a high level, so players are roughly having the same gaming technique. I am going to use several machine learning algorithms like Naive Bayes, Decision Tree, Logic Regression and so on to determine which factor determines the game's victory or defeat to the greatest extent.

Keywords: Machine learning; league of legends

IST 652 Final Project Paper  
How to win the game in League of Legends?

# Introduction

League of Legends is a free 5v5 online MOBA game created and published by Riot Games. The end-game objective of the game is to destroy the enemy team's nexus, located deep into their base. Before doing so, players split into roles and head into lanes where they must kill minions for gold, battle enemy players and destroy their structures. Players choose champions in order to play the game, with all champions having unique abilities and roles. There is also a jungle with a wide variety of jungle monsters, as well as neutral monsters that can be killed by both teams such as Baron Nashor, Elemental Drakes or Elder Drake. The killing of these monsters grants different bonuses and rewards.

The dataset I use is from Kaggle: <https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min>. This dataset contains the first 10 minutes detailed data of approximate 10k ranked games from a high level, so players are roughly having the same gaming technique. There are total 21 columns and 9879 rows in it and all about the blue team. I will explore them more in the future sector.

# Define the problem

The purpose of players playing League of Legends is to win, and this is a typical classification problem because the final result of the game is win or loss. Besides, League of Legends is one of the largest esports with various annual tournaments taking place worldwide. In terms of esports professional gaming as of June 2016, League of Legends has had $29,203,916 USD in prize money, 4,083 Players, and 1,718 tournaments, compared to Dota 2's $64,397,286 USD of prize money, 1,495 players, and 613 tournaments. My goal is to help players to find out the crucial rules to win the match.

# Understanding the data

There are total 21 columns in the dataset. From data.types function(data is a pandas dataframe with the dataset I use), it seems that there are 8 categorical colmuns and 13 numerical columns. ‘gameID’ is unique RIOT ID of the game that can be used with the Riot Games API. ‘blueWins’ is the target column, yes means blue team win and no means loss. ‘blueWardsPlaced’ is the number of warding totems placed by the blue team on the map. ‘blueWardsDestroyed’ is the number of enemy warding totems the blue team has destroyed. ‘blueFirstBlood’ is if the blue team get the first kill of the game. ‘blueKills’ is number of enemies killed by the blue team. ‘blueDeaths’ is number of blue team deaths. ‘blueAssists’ is number of blue team kill assists. ‘blueEliteMonsters’ is number of elite monsters killed by blue team. ‘blueTotalGold’ is blue team total gold and there are five levels: normal, few, very few, many, very many.

# Exploratory data analysis (EDA)

I use two methods to achieve Exploratory data analysis, the first one is a package called pandas\_profiling and another is coding by myself.

pandas\_profiling is an open source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. Besides, if this is not enough to convince us to use this tool, it also generates interactive reports in web format that can be presented to any person, even if they don’t know programming. I intercept part of the report of column ‘blueWardsPlaced’ as Figure 1, the full html report has uploaded to my github: <https://github.com/Kun97/ist652finalproject>. We can easily find out almost all the information we want, like missing value, mean, minimum, maximum and also a chart of distribution, it is very convenient to explore the data.

A screenshot of a cell phone

Description automatically generated

Figure 1: part of pandas\_profiling report

In my own code, I create some figures to show the distribution of each column like figure 2 and figure 3. Besides, I also create a correlation chart for each of the numerical column in figure 4. A picture containing bird

Description automatically generated

Figure 2: visualization of numerical column

A screenshot of a cell phone

Description automatically generated

Figure 3: visualization of categorical column A screenshot of a cell phone

Description automatically generated

Figure 4: correlation chart

# Data Preprocessing and Preparation

The first column, ‘gameID’, has no relationship with the result, so I decide to delete this column. After checking, this data set has a good condition that there are no missing values and duplicated values. Based on the report, there are no outliers for all numerical columns except ‘blueWardsPlaced’. For this column, low records can be existing but too many can’t, so I decide to delete the records which above 99-th percentile.

The last step is transformation which is covert all the categorical columns to numerical columns, and what I use is map function like data['blueTotalGold'].map({'Very Few': 0, 'Few': 1, 'Normal': 2, 'Many': 3, 'Very Many': 4}).

# Machine Learning and Statistical Modeling

There are two main parts of machine learning: supervised learning and unsupervised learning. For supervised learning, input data is called training data and has a known label or result such as spam/not-spam or a stock price at a time. Actually in my dataset, it already has a label for the result which is ‘blueWins’, but I still implement an unsupervised learning algorithm, I will talk it later.

I use five supervised learning algorithms which are Naïve Bayes, Decision Tree, Random Forests, Logistic Regression and K-nearest neighbors. The package I use to implement algorithms is scikit-learn. scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

The first step is create training and testing data. I use train\_test\_split method from sklearn.model\_selection to automatically split the origin dataset, the code is X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 25). Next step is training the model, I will use Naive Bayes as an example. The code is very simple, what I do is import Naive Bayes and then training the model. Then, using test data to predict the result. In the end, I calculate the accurate for each algorithm using accuracy\_score() under sklearn.metrics.

For unsupervised learning, input data is not labeled and does not have a known result. So the algorithm I choose is kmeans and I set cluster as two. What I want to achieve is it will be divided to win and loss group. The accurate for testing result is 0.28374, but what I think is that it just divided to two groups but doesn’t labeling them. In other words, the accurate is 0.71626. and this result really surprised me.

# Model Performance Evaluation and Algorithm Fine-tuning

A close up of a sign

Description automatically generatedAs is said before, I calculate the accurate for every algorithm and I create a table (see Figure 5) to demonstrate it.

Figure 5: accurate of all algorithms

We can find that the logistic regression is the most accurate. So, for the next step algorithm fine-tuning, I will only choose logistic regression to move forward.

The method I use for algorithm fine-tuning is GridSearchCV under sklearn.model\_selection. GridSearchCV exhaustive search over specified parameter values for an estimator. I set two parameters for tuning, the first one is dual: true or false, the second one is max\_iter: 100, 150, 200, 500 and 1000. After training, I find out the best parameter is 'dual': False, 'max\_iter': 100 and the accurate is 0.732679.

# Model interpretation & Conclusion

I list the coefficient of each attribute in figure 6.

A close up of a sign

Description automatically generated

Figure 6: coefficient of each attribute

From figure 6 we can find out the ‘blueDragons’ has the most positive correlation and ‘blueHeralds’ has the most negative correlation which means if you want win the match, you must control as many dragons and less heralds as you can. Besides, ‘first blood’, ‘elite monster’ and ‘jungle minions killed’ are also very important to the result of the game and you need avoid gain lots of experience for the whole team.