**The Impact of First Objects Obtaining on the Win Rate**

**of League of Legends**

**Term Project Final Report for Machine Learning Ⅰ, 1st Semester 2022**

**21102398**

**Daeil Han**

Department of Applied Artificial Intelligence

dalehan0606@seoultech.ac.kr

Abstraction

Since humankind first started playing games, predicting the outcome of games has been a major concern of humankind. League of Legends is now the most popular e-sports in the world. This study is motivated by curiosity about who will be the winner. This study analyzes how getting objects first affects the win rate of the game through logistic classification model and Naive Bayes model, and reveals the limitations. In conclusion, the model of this work is very predictable, but it implies very fatal flaws.

1. **Introduction and Rules**
   1. **Introduction**

League of Legends is one of the most popular games among male college students. In this game, two teams compete against each other for destroying the opponent’s Nexus. Nexus is the most important target in this game, and the team is defeated when their Nexus is destroyed by the enemy team.

Teams fight against each other in small-scale battles to achieve their final goal. Depending on the victories or defeats of the battles, each team's win rate may increase or decrease. The team that won a small-scale battle gets some reward. Players use these rewards best and prepare to win the next slightly larger battle. The most representative strategy of the League of Legends is to make a kind of virtuous cycle that connects small victory to small reward, and small reward to larger victory.

Considering the structure of this game, the most important moment in this game will be the time the virtuous cycle mentioned above is begun by some small rewards. In other words, our purpose is to abstract these small rewards into several objects and design a model that predicts the probability that a team will win when they first acquire the object. We will use part of a dataset *“League Of Legends High elo Ranked Games (2020)”* [1]. We will use logistic regression for the win rate prediction.

* 1. **League of Legends Rules**

League of Legends is a very complicated game. It is impossible to understand all rules of the game at once. Therefore, will deal with only simplified rules for discussion.

There are two teams: the blue team and the red team. Each team has to destroy the opponent's Nexus or get the opponent's surrender to win the game. There is no tie. For these purposes, players usually try hard to get some small rewards mentioned above. It's like chess players trying to break the balance of power by letting out even one more of the opponent's pieces. We define these rewards as ‘first (opponent) player kill’, ‘first turret demolition’, ‘first dragon kill, ‘first inhibitor demolition.’, and ‘first Nashor Baron-kill.’ Players can get some gold to purchase stronger items by expelling the opponent player or demolishing a turret. When they destroy the enemy’s inhibitor, the team can get stronger troopers. Players can get some special abilities by killing a dragon or Nashor Baron (another more valuable monster).

1. **Related Work**

So many related works exist. Specifically, this study was directly inspired by *Predicting In-Game Events from Game Event Streams*.

1. **Methods**
   1. **Data Set**

The data set is collected in Kaggle. It has three game data set by tier: Challenger, Grand Master, and Master. We only use the master tier data set because the upper tier players’ data sets are relatively small and mixing two data sets can contaminate our task. In addition, the data set also has so many unnecessary features because our purpose is to analyze the impact on the win rate of getting objects before the opponent team. So, we slice them for it to better suit our purpose.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **blueWin** | **blueFirstBlood** | **blueFirstTurret** | **blueFirstDragon** | **blueFirstInhibitor** | **blueFirstBaron** |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
|  |  |  |  |  |  |

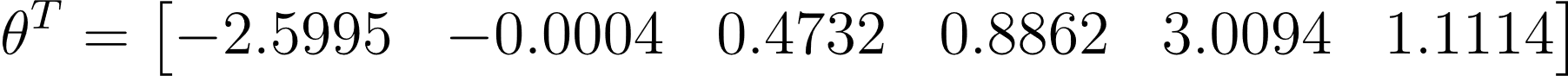
Table 1: Head data set of Master ranked games

If a blue team won, the value is 1. As mentioned earlier, there is no tie. If a blue team first got an object, its value becomes 1.

* 1. **Logistic Regression**

At first, multivariate linear regression is used to find the decision boundary. The hypothesis function is as follows.[2]

|  |  |
| --- | --- |
|  | *Eq. 1* |



We can find the coefficients by using Python and Scikit-learn. They are derived by the linear regression algorithm. However, we cannot use linear regression because our goal is to predict whether a team will win a game or not. In other words, the predictions should come out as ‘yes’ or ‘no’. To solve this problem, we will use logistic regression. In the linear regression process, predictions can even exceed ‘1’, and it cannot be interpreted in our task. The logistic regression will regularize these values. The Sigmoid function for the logistic regression is as follows.

|  |  |
| --- | --- |
|  | *Eq. 2* |

To classify each case, put Eq. 1 to Eq. 2, which is the function used to predict the result of games.

|  |  |
| --- | --- |
|  | *Eq. 3* |

When a row data is assigned in , the model calculates the win rate of the blue team, and if it is more than 0.5, it decides the blue team will win. If not, it decides the blue team will lose.

1. **Result and Conclusion**

All analysis is conducted by Python libraries such as NumPy, Pandas, Matplotlib, Seaborn, and Scikit-learn. The evaluation of the performance of the model is as follows. [3, 4, 5]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Accuracy** | **Precision** | **Sensitivity** | **Specificity** | **F1 Score** |
| 0.847 | 0.864 | 0.820 | 0.874 | 0.841 |

Table 2: Evaluating the performance of the model

The accuracy of the model is about 0.847, which means the error rate is about 0.153. The precision is 0.864, a little bit higher than the accuracy. The sensitivity is 0.82, whereas the specificity is 0.874. It looks like the model is better in the cases of defeat, about 6.6%. The confusion matrix and ROC curve are as follows. The AUC is 0.911.

|  |  |
| --- | --- |
| Figure 1: Confusion Matrix | Figure 2: ROC Curve |

Fortunately, the result shows that the hypothesis, predicting the win rate from the first gain of the objects, seems reliable.

1. **Regrets**

The first regret is that this model cannot predict an outcome of a game until quite a time later because some objects are not decided at the end phase of the game. Therefore, for the model to become more useful, those objects are removed from consideration. However then, the accuracy of the model will decrease.

Second, the probability of obtaining objects in the early phase of the game is equally considered with the probability of obtaining objects in the later phase. In other words, when a team obtained an object from the early phase, it influences the probability that the team will obtain later phase objects. Therefore, the events considered in the task are not independent.

Third, there are some omissions in the model. The case of surrender was omitted. In addition, this model assumed that if a blue team didn’t get some object, it belongs to the red team. This is a fragile assumption because some objects can not belong to any team.

1. **Contributions**

I did everything by myself except for advice on the table of contents, which was advised by assistant Jin Chang-Gyun and Professor Kim Sung-Eun of BrAIn Lab.

GitHub link https://github.com/moordo91/Machine-Learning-1/tree/main/TermProject

**References**

[1] Minyoung Shin. League Of Legends High elo Ranked Games (2020). https://www.kaggle.com/datasets/gyejr95/league-of-legends-challenger-ranked-games2020?resource=download.

[2] @\_@..... Scikit-learn으로 로지스틱 회귀 (클래스 분류편). https://dprdmr.tistory.com/34.

[3] 테디노트. 로지스틱회귀(Logistic Regression)와 분류 평가 지표 (Precision, Recall, F1 Score). https://teddylee777.github.io/scikit-learn/scikit-learn-logistic-regression-and-metrics.

[4] DelftStack. Python에서 ROC 곡선 그리기. https://www.delftstack.com/ko/howto/python/plot-roc-curve-python/.

[5] 통계의 본질 (유튜브 : 통계의 본질). ROC curve 이해하기 ① 직접 그려보기. https://hsm-edu.tistory.com/1033.