PUBG Final Placement Prediction
STA561/ECE682

Jiayi Liu(jl843) Yimin Xu(yx142) Zeyu Liu(zl194)
ECE Department
Duke University
yx142@duke.edu

Abstract

For a e-sport game, what people is seeking is a best strategy for a better final placement. In game PUBG, there are tons of elements will affect the results. In this project, a model for final placement will be derived based on Random Forest algorithm and GBDT algorithm in the LightGBM framework.

1 Introduction

1.1 Background

PUBG(aka PLAYERUNKNOWNs BATTLEGROUNDS) is one of the most popular games around the world. It is an online multiplayer battle royale game made by Korean game company Bluehole. Each player can choose to enter the match solo, duo or with a team up to four players and players have to fight with other players/teams to eliminate other teams. Every player is supposed to enter the game map with nothing and has to search the map to find their equipment. As the game going on, the available game map will shrink and players who are outside the available area will be judged as death. Finally, the last living team will be the winner.
There are several popular strategies to win a PUBG game. Some players incline to hide in a secret place and avoid battles, while others may be more likely to involve in battles and loot the death players equipment. Our project intends to use the game data collected from real games to predict the rankings inside a single game and therefore analyze the importance of each statistic to provide a more efficient approach to win the game.

1.2 Data Description

This dataset comes from the official game data of PUBG which is open to public. The data was collected through the PUBG Developer API by Kaggle.
There are more than 6 million anonymized player data in this dataset, which are separated into training and testing sets, and we wish to predict the final placement from final in-game stats and initial player ratings to discover the best strategy to get higher placement in a PUBG game.
This dataset contains 28 properties which can be classified into 2 parts. The first part is the final in-game stats which contains damageDealt, assists, kills, etc. And the second part is the initial player rating, that is, the winPoints, the win-based external ranking of player. All rows contain missing values(like *Nan*) are deleted.
The correlation between the final winPoints and the features are listed as following.

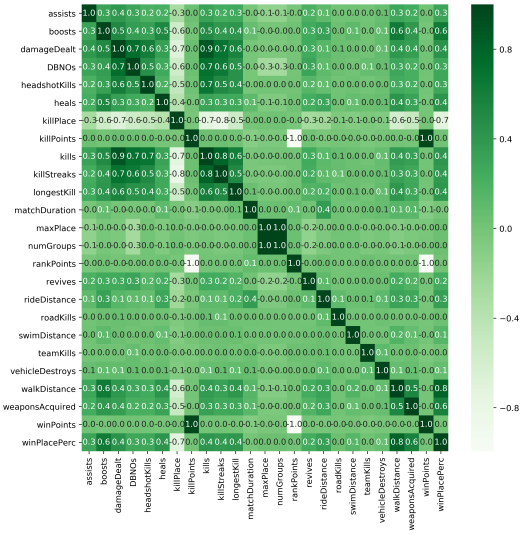

Figure 1: Correlation heatmap

2 Algorithm/Method

In this section, we will discuss two common approaches to do the regression work in Machine Learning. In addition, a newly established framework for large data will also be mentioned.

2.1 Random Forest[1]

Random forest is a supervising machine learining algorithm which builds a ”Forest” ensembling a group of ”trees”. The tree it uses is actually the decsion tree, most of the time trained with bagging model. The diagrammatic sketch of random forest is listed as following.

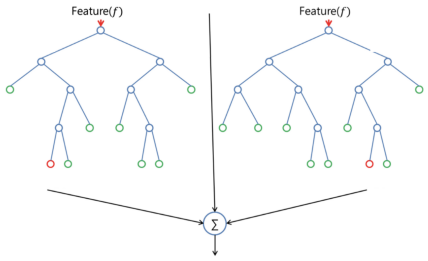

Figure 2: Diagrammatic sketch for Random forest

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples: For $b = 1, ..., B$:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After training, After training, predictions for unseen samples $x^{'}$ can be made by averaging the predictions from all the individual regression trees on $x^{'}$:

$$\hat{f} = \sum_{b=1}^{B} f_b(x^{'}) \tag{1}$$

2.2 GBDT[2]

Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners. For this special case, Friedman proposes a modification to gradient boosting method which improves the quality of fit of each base learner.

Generic gradient boosting at the m-th step would fit a decision tree $h_m(x)$ to pseudo-residuals. Let J_m be the number of its leaves. The tree partitions the input space into J_m disjoint regions $R_{1m}, \ldots, R_{J_m m}$ and predicts a constant value in each region. Using the indicator notation, the output of $h_m(x)$ for input x can be written as the sum: $h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x)$,

where b_{jm} is the value predicted in the region R_{jm}.

Then the coefficients b_{jm} are multiplied by some value γ_m, chosen using line search so as to minimize the loss function, and the model is updated as follows:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \quad \gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)).$$

Friedman proposes to modify this algorithm so that it chooses a separate optimal value γ_{jm} for each of the tree's regions, instead of a single γ_m for the whole tree. He calls the modified algorithm "TreeBoost". The coefficients b_{jm} from the tree-fitting procedure can be then simply discarded and the model update rule becomes:

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma).$$

2.3 LightGBM[3]

LightGBM is a gradient boosting framework developed by Microsoft for tree based learning algorithms. It is designed to offer a higher efficiency and faster training speed. In this project, we use the LightGBM python package to conduct both Random Forest and GBDT models, as LightGBM offers easy-to-use APIs.

3 Results

Three regression models, the normal linear regression, random forest and GBDT, are used in this project and MSE error are used to evaluate these models. The general MSE error of these models are listed below.

Table 1: Errors of different models

MODEL	ERROR
Linear Regression	0.127
Random Forest	0.116
GBDT	0.085

3.1 Random Forest

As to show the model performance, the error distribution of random forest model are plotted. One thing to notice that the error here is just the difference between prediction and real value.

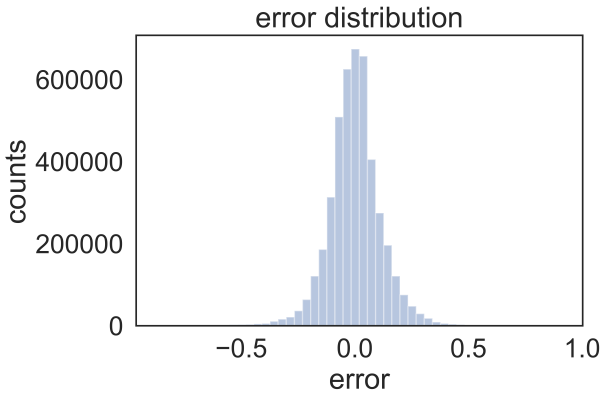

Figure 3: Random Forest Error Distribution

As a tree-based models, random forest is able to calculate the importance of each feature.

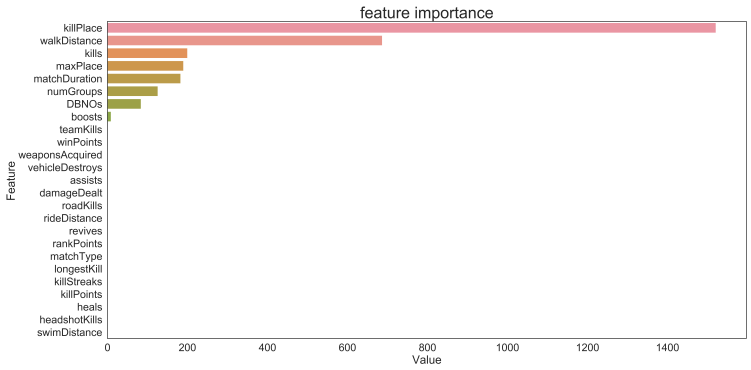

Figure 4: Random Forest Feature Importance

3.2 GBDT

The error distribution and feature importance figure are also plotted for GBDT model. Compare to random forest, the error distributed more concentrated near zero. Also, the feature importance figure is more uniform and more features are taken into account.

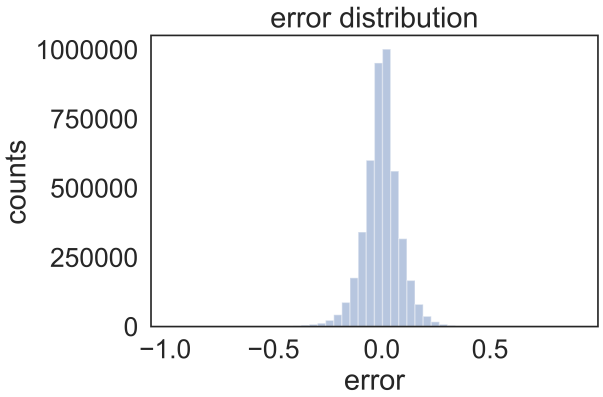

Figure 5: GBDT Error Distribution

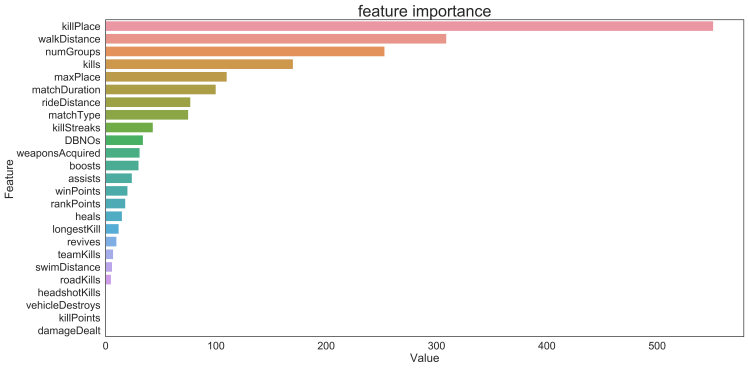

Figure 6: GBDT Feature Importance

4 Conclusion

4.1 Best classifier

From the mean square error derived by the classifier we use, GBDT under lightgbm framework shows the best performance.

4.2 Strategy for higher placement

To get the highest final placement in the PUBG game, player need to kill other players to acquire what they have. By doing such things, they are more likely to ensure their items to be sufficient enough for the entire game. Secondly, people with more walking distance will be more likely to get higher placement in the last. With longer walking distance, people are more likely to avoid others' focus and survive longer.

References

[1] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[2] Wikipedia. Gradient boosting. http://en.wikipedia.org/w/index.php?title=Gradient%20boostingoldid=893919603, 2019.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.