Brown University
DATA 1030

December 7, 2021

PUBG Finish Placement Prediction Project Final Report

Danica(Hui) Shen

Github repository: https://github.com/DS-134340/Data1030-Project

Introduction

The PUBG (Player Unknown's Battlegrounds) game has become increasingly popular on various platforms all over the world in recent years. At most 100 players are matched in a single game, and players are given the choice of playing on their own (“solo”), playing with a partner (“duo”), playing with a team (“squad”), and some other customized or event-designed forms. All the players have a collection of multiple behaviors throughout a game, with the purpose of moving towards a safe place, acquiring varieties of munitions and weapons, implementing teamwork such as assists and revivals, shooting or eliminating other players by other means, and ultimately winning the game when the player(s) become the final survivor(s). With the help of API, data regarding those behaviors and the performance of the players are collected into an integrated data set, which can provide some meaningful insights to both the players and the company who offers the game. As for the players, they will be able to learn from the relationships between particular behaviors during and the end result of the game, thus they can improve their own behaviors and strategies in order to enhance their performance and increase the chance of their survival. When it comes to the game provider, such data set might help them identify potential fraudulent players who violate the rules or take advantage of external support, as well as how they can improve, optimize, and innovate the game setting so that they can realize a high player retention rate by keeping players motivated.

The purpose of this project focuses on the players’ perspective, which is to examine the relationship between players’ behaviors and their performance. The data set is provided by the Kaggle website as one of their Playground Code Competitions. [1] There are 4,446,966 data points along with 29 columns in the original data set. The target variable is called “winPlacePerc”, representing the percentile winning placement at the end of each game. As the name implies, all the values are within the range between 0 and 1, with 1 indicating the first place and 0 indicating the last place. Since the target variable includes numbers describing meaningful values but not categories, regression techniques should be used in this project.

There is one Kaggle user called “bjebert” won the first place in this competition, and he gives out three key steps that should be considered and implemented in order to receive a better and more reasonable prediction result. [2] Firstly, he emphasizes the importance of the variable “killPlace” and points out that every player from a group shares the same final placement of the game. Secondly, he discusses an approach utilizing an XGB regression model to find the correct placement for those “unknown” groups of each match. Thirdly, he talks about how his group implements post-processing on the data set in order to deal with the missing groups and adjust the ranking system accordingly.

Exploratory Data Analysis

After some explorations on the original data set, one record with missing value for the target variable has been discovered and dropped. There is no missing value in other attributes, so the data set has 4,446,965 records at this point.

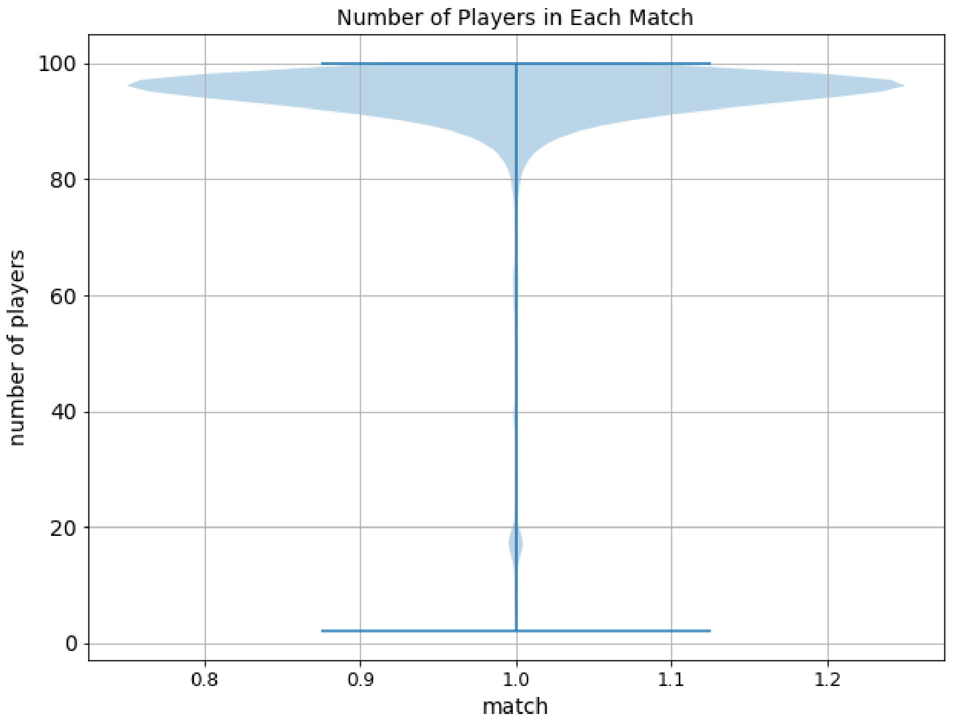

Figure 1 This figure shows the distribution of the number of players whose data has been collected into this data set, in each match, which can be alternatively described as each single game. It shows that there are more than 80 players' data that have been collected for the majority of the matches/games, while a small group has only around 20 players of each match/game.

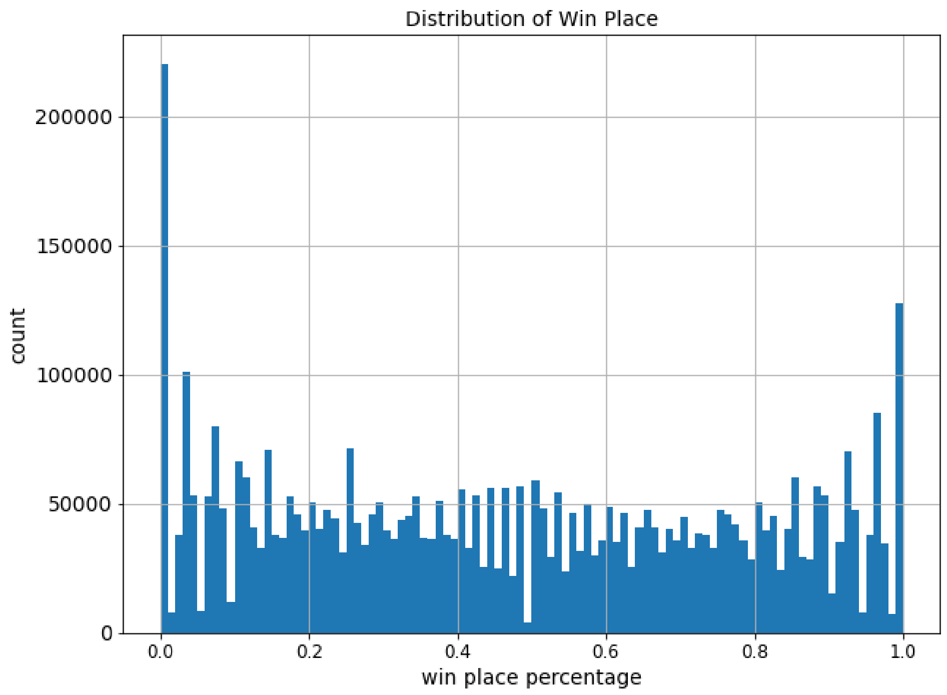

Figure 2 This figure describes the distribution of the target variable of this project. As the values of the "winPlacePerc" variable are in the percentile format that is obtained by calculation instead of an exact number indicating the ranking, it seems plausible that the middle point of the placement that represented by 0.5 has least distribution after binning. Similarly, the top and bottom placements have a higher density.

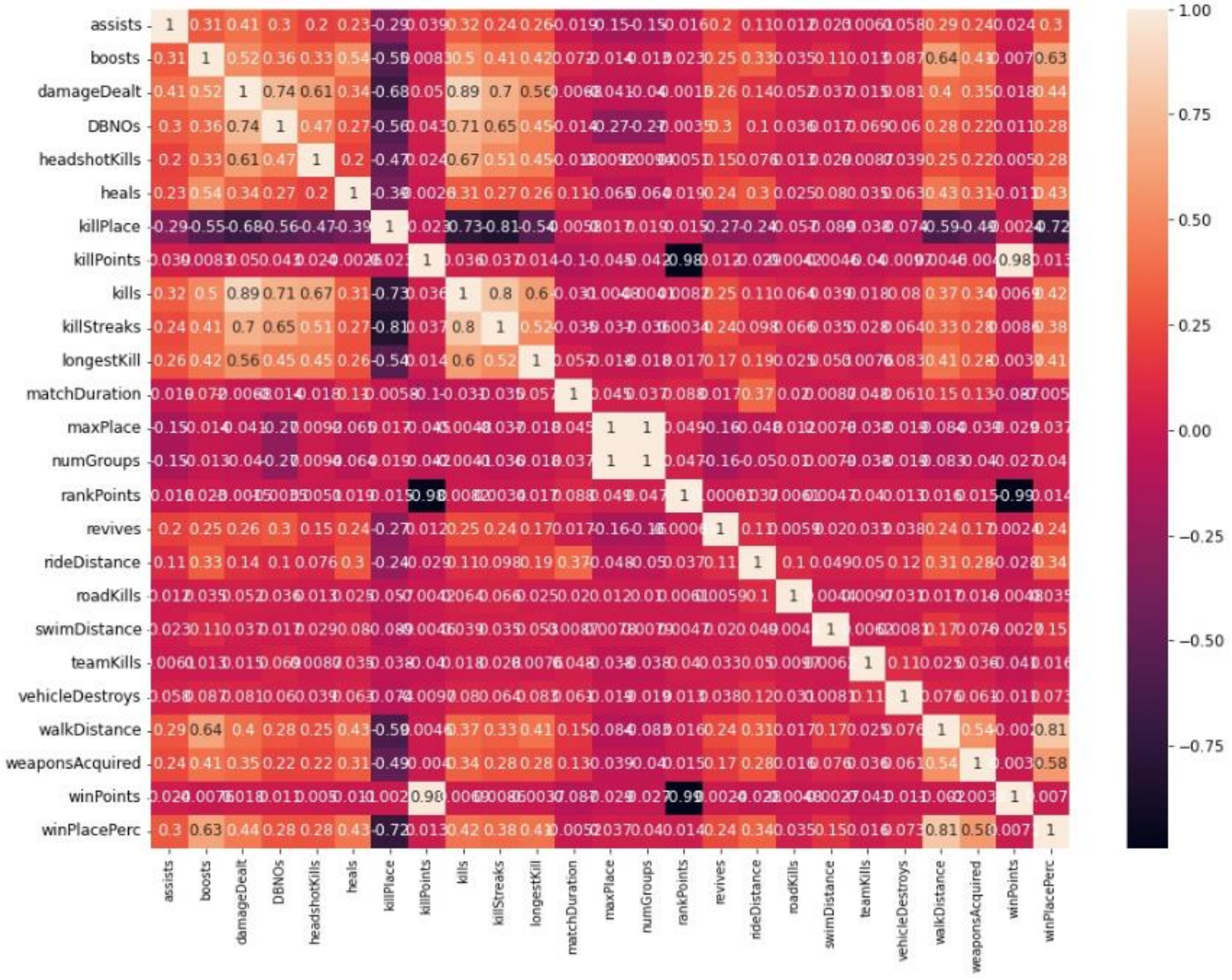

Figure 3 This figure interprets the correlation between pair-wise numeric variables in the data set. It is evident that, among those continuous features, “killPlace”, “walkDistance”, “boosts”, and “weaponAcquired” are the top ones that have strong correlation with the target variable.

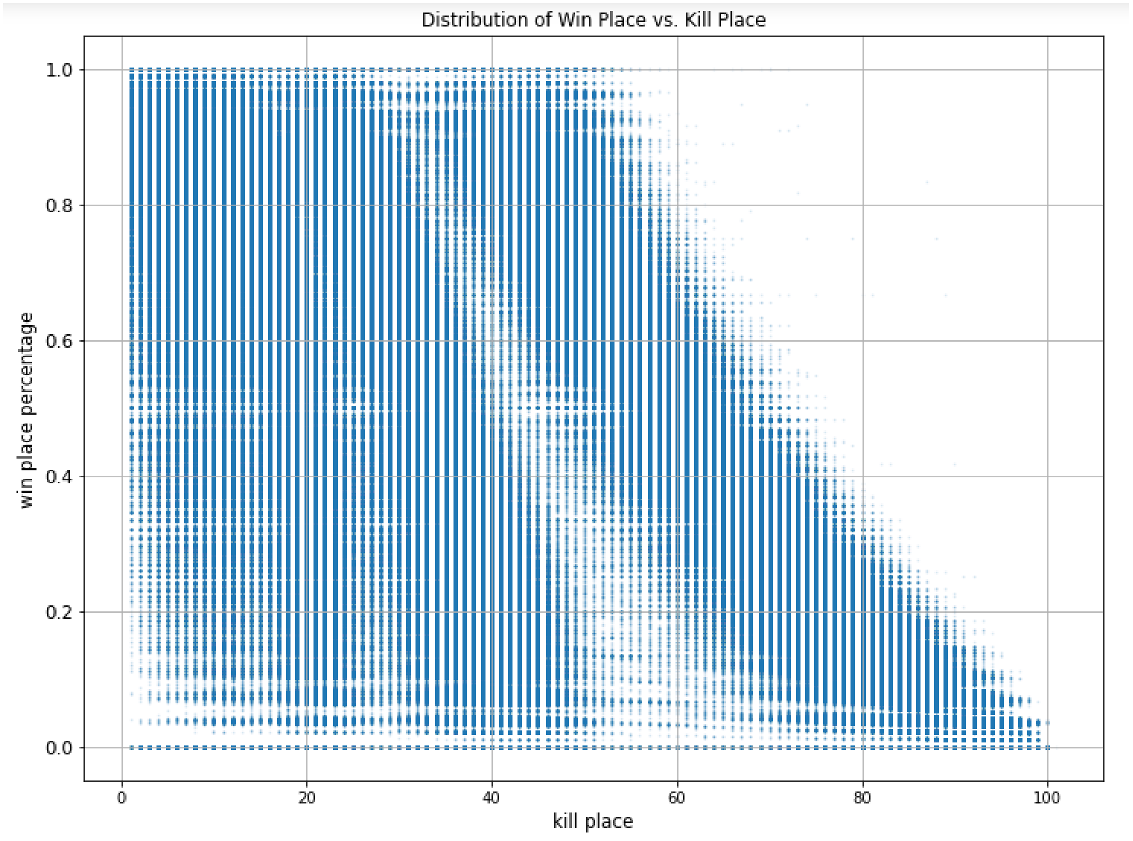

Figure 4 This figure examines the relationship between "killPlace" and the target variable. It describes that the ranking in match of number of enemies a player killed can to some extent partially explain that player's final placement of a game. A sparse distribution in the top-right of the figure indicates that players with worse kill-rankings are less likely to gain better final winning placement in each game.

Methods

Data Splitting

Although this data set may include multiple records from the same player, all the games that one player played are independent and each data point has indeed a unique "Id", thus this data set should be considered as an IID data set and there is neither a group structure nor a time series in this data set. There are more than 4 million data points in the original data set, from which 44,470 records (around 1%) were randomly sampled for modeling due to storage and processing speed limitations. For the sample data set, the basic random split approach was implemented with sizes of 60%, 20%, and 20% for training, validation, and testing data sets, respectively, that would best mimic future use when machine learning models are deployed.

Data Preprocessing

As discussed in previous section, the three id-related columns, including “Id”, “matchId”, and “groupId” were dropped as they could help nothing for predicting the target variable. In addition, “numGroups” and “rankPoints” were also dropped since there are some inconsistency issues with these two columns that mentioned in the data dictionary of this data set on the Kaggle website.

The column “matchType” is the only categorical variable and there is no ordinal logic within it, thus the OneHotEncoder was applied. Given the nature of the PUBG games, the maximum number of players of each single match is 100, so the variables regarding placements should have a solid range between 1 and 100. Therefore, the MinMaxScaler should be utilized to transform the “killPlace” and “maxPlace” variables. All other independent variables were transformed by implementing the StandardScaler. After preprocessing the data, there are 38 features in total that can be fed into machine learning algorithms in further steps.

Model Tuning

After the above splitting and preprocessing steps, four different machine learning algorithms were trained and compared based on the selected evaluation metric, mean absolute error (MAE). The baseline model was implemented by simply using the median value of the training data as the prediction for the testing data. Those four machine learning models, including Lasso Regression, Ridge Regression, Decision Tree Regressor, and Random Forest Regressor, were all hyperparameter tuned by searching for the optimal hyperparameter combination according to the minimum validation MAE. In order to get a more accurate MAE, all the predictions that are below 0 or above 1 were set to 0 and 1, respectively, since the target variable “winPlacePerc” has a solid range with a maximum of 1 and a minimum of 0. The training and tuning process was implemented 6 times regarding 6 different random seeds used for splitting. The table below shows the values of the hyperparameters that were tuned for the four models.

ML Model	Hyperparameters
Lasso	alpha: 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1
Ridge	alpha: 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1
Decision Tree	splitter: best, random max_depth: 7, 8, 9, 10, 11 max_features: auto, log2, sqrt
Random Forest	max_depth: 10, 11, 12, 13, 14 n_estimators: 500, 600, 700, 800, 900

Figure 5 Hyperparameters used in model tuning

Results

Model Selection

As for each machine learning algorithm, the optimal hyperparameter combination can be determined for each random seed. The table and the plot below compare the tuned models by listing out the optimal combination of the hyperparameter(s) for each model and the average MAE score on the testing data.

ML Model	Hyperparameters	Average Testing MAE
Lasso	alpha: 1e-7	0.088
Ridge	alpha: 1e-7	0.088
Decision Tree	splitter: best max_depth: 10 max_features: auto	0.076
Random Forest	max_depth: 14 n_estimators: 600	0.066

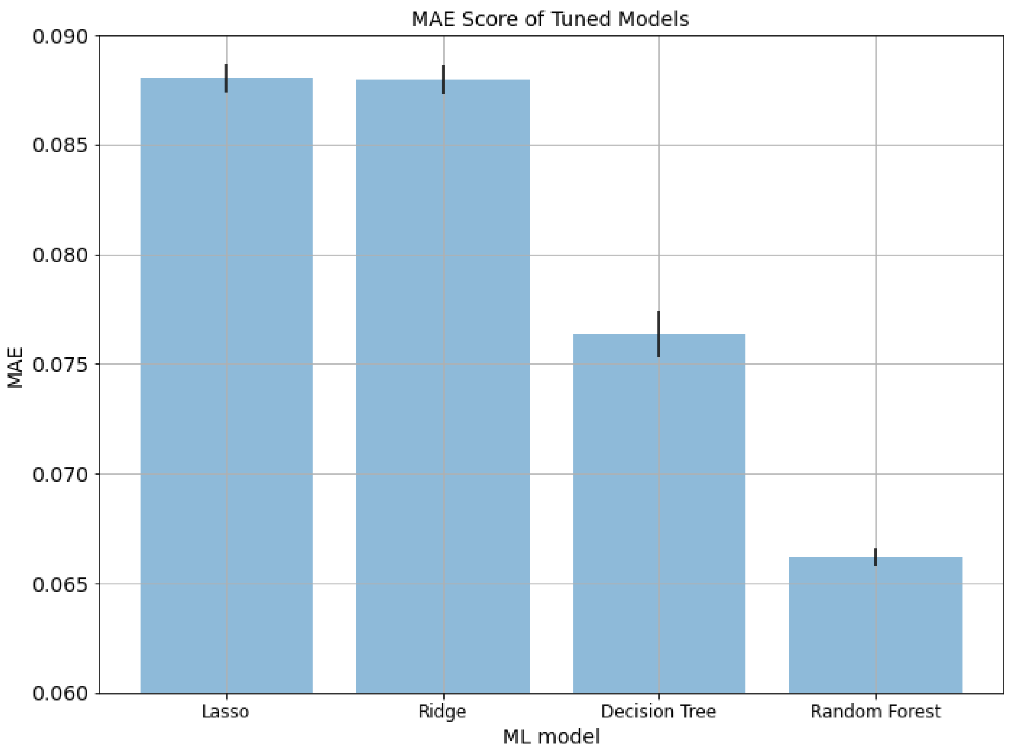

Figure 6 Model comparison

As we can conclude from the table and the plot above, the Random Forest model is the best performing model with a hyperparameter combination of the max_depth at 14 and the n_estimators at 600, thus it was selected as the final model used for making the prediction and evaluating the model.

Model Evaluation

In order to evaluate the best performing model described above, it was trained 20 times based on different random splits defined by 20 different random states. With the purpose of evaluating the model, the data was split into training for 80% and testing for 20% each time. The MAE scores for both the baseline and the selected models were stored for comparison. The baseline models have an average MAE score at 0.268 with a standard deviation of 0.002, while the Random Forest models have an average MAE score at 0.066 with a standard deviation of 0.0004. Therefore, the MAE of the Random Forest models is about 101 standard deviations below the baseline and the MAE of the baseline is about 505 standard deviations above the best performing Random Forest model.

Model Interpretation

Outlook

As mentioned in the previous sections, this data set is considerably large, so only 1% of the original data set was used to train the model and interpret the findings. Whether the sampled data can represent and speak for the entire original data might be doubted as it is only a small portion. In this case, taking advantage of a device, an environment, or a platform that has considerably stronger computing power should be a better choice. Therefore, more data or different samples might be used for modeling and a more comprehensive model tuning process with more hyperparameter combinations might be implemented in order to get a model with more predictive power and achieve an even more accurate prediction.

References

[1]: https://www.kaggle.com/c/pubg-finish-placement-prediction/data

[2]: https://www.kaggle.com/c/pubg-finish-placement-prediction/discussion/79161