PUBG Finish Placement Prediction

Name: ZHOU, Ji (20583761) Email: jzhoubl@connect.ust.hk

Background

In a PUBG game, we have up to 100 players in each match. Players can be on teams which get ranked at the end of the game based on how many other teams are still alive when they are eliminated. In game, players can pick up different weapons and boosts, revive downed-but-not-out (knocked down) teammates, drive vehicles, swim, run, shoot, and experience all the consequences -- such as falling too far or running themselves over and eliminating themselves.

This Project aims to predict players' finishing placement based on their final stats in a PUBG game. The data comes from over 4 million players within over 40 thousand matches with all types: solos, duos, squads, and custom. To simplify the calculation flow, we treat all match types as the same since they didn't affect the final place too much.

Dataset

The training set contains 4446966 unique players' statistics in 2026745 groups and 47965 matches. All data are sorted with their player ids and the target values of win place percentage have been calculated. These data are stored in train_V2.csv with another test_V2.csv file containing some test cases without calculated target values. Our purpose is to use the trained model to predict the win place percentage of statistics in the test file.

To collect related data, we specify the schema definition with sentences. The details are not listed here, they can be found in the notebook file.

Data Fields 1	Data Fields 2	Data Fields 3
• **Id** • **Group Id** • **Match Id** • **Assists** • **Boosts** • **Damage Dealt** • **DBNOs** • **Headshot Kills** • **heals** • **kill Place**	• **kill Points** • **kills** • **Kill Streaks** • **Longest Kill** • **Match Duration** • **Match Type** • **Max Place** • **Num Groups** • **Rank Points** • **revives**	• **ride Distance** • **road Kills** • **swim Distance** • **team Kills** • **Vehicle Destroys** • **Walk Distance** • **Weapons Acquired** • **Win Points** • **Win Place Perc (target)**

Samples of the dataset

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
1	eef90569b9d03c	684d5656442f9e	aeb375fc57110c	0	0	91.47	0
2	1eaf90ac73de72	6a4a42c3245a74	110163d8bb94ae	1	0	68	0
3	4616d365dd2853	a930a9c79cd721	f1f1f4ef412d7e	0	0	32.9	0
4	315c96c26c9aac	de04010b3458dd	6dc8ff871e21e6	0	0	100	0
5	ff79c12f326506	289a6836a88d27	bac52627a12114	0	0	100	1
6	1a68204ccf9891	47cfbb04e1b1a2	df014fbee741c6	0	0	51.6	0
7							

Preliminary Analyses

Notice that some of the schema is not related to the win Place, so we can remove them at the beginning.

- **Id** - Player's Id
- **Group Id** - ID to identify a group within a match.
- **Match Id** - ID to identify match.

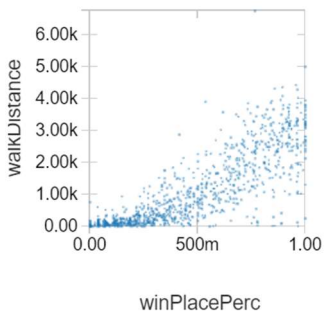
To simplify the algorithm, we decide to remove ELO statistics since it may mislead our model to highly depend on them.

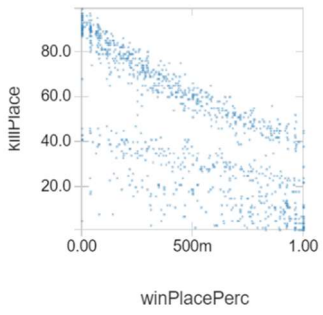
- **Kill Points** - Kills-based external ranking of player.
- **Rank Points** - Elo-like ranking of player.
- **Win Points** - Win-based external ranking of player.

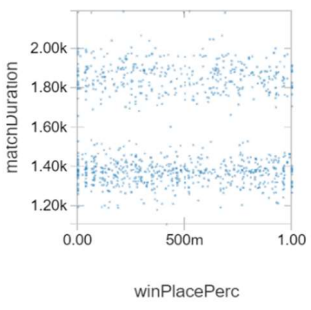
Since there are some unrelated statistics may confuse the results due to complex match cases, we remove them instead of dealing with them

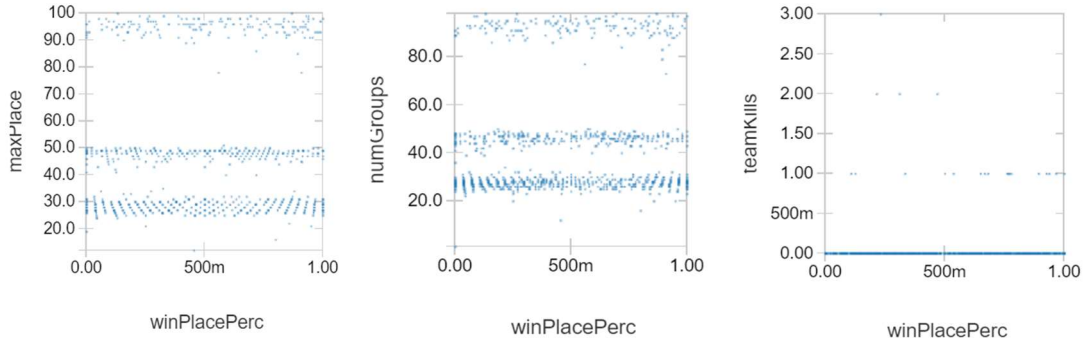
- **Longest Kill** - Longest distance between player and player killed at time of death. This may be misleading, as downing a player and driving away may lead to a large longest Kill stat.
- **Match Type** - String identifying the game mode that the data comes from.

To improve the criteria, we must do some basic statistical analyses of all the columns. Now we choose to examine some of statistics that may be unrelated to the target. We choose walk Distance, kill Place, match Duration, max Place, num Groups, team Kills as our example.









As a preliminary result, we find that match Duration, max Place, num Groups, team Kills have little effect on our target, but walk Distance and kill Place has a certain effect. So we need to drop some columns to decrease the complex of calculations. At this stage, we also check whether there is a null value in the dataset. If there is one null value, we can drop this row.

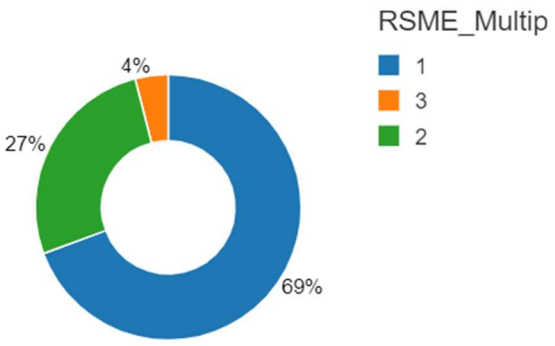
Data Preparation

Then we need to prepare the data for machine learning. To see how well the model is, we need to split the dataset into new training set and test set with 80% and 20% statistics in the original training set. Before we apply any of the algorithm, we first generate linear analyses of the dataset to see some preliminary results. The generated linear regression equation is as below:

$$
\begin{aligned}
y = {} & 0.413646647358 + (0.0230870694198 * \text{boosts}) + (0.0179682771692 * \text{assists}) \\
& + (0.0171856868531 * \text{weaponsAcquired}) + (0.014296475364 * \text{revives}) \\
& - (0.0124608091795 * \text{killStreaks}) - (0.0107073646573 * \text{DBNOs}) \\
& + (0.00597216525247 * \text{roadKills}) + (0.00481247690183 * \text{heals}) \\
& - (0.00284673719129 * \text{killPlace}) \\
& + (0.00186405039592 * \text{vehicleDestroys}) \\
& + (0.00127669784278 * \text{headshotKills}) \\
& + (0.00030038312258 * \text{swimDistance}) - (0.00013574919577 * \text{kills}) \\
& + (7.99806962651 * 10^{-5} * \text{walkDistance}) \\
& + (4.24414949638e * 10^{-5} * \text{damageDealt}) \\
& + (1.22938354063e * 10^{-5} * \text{rideDistance})
\end{aligned}
$$

```
Root Mean Squared Error: 0.16
r2: 0.74
```

And we construct the pie chart of RMSE to see if the linear regression is perfect.



Model Training

To improve the results, we can directly use decision tree and random forest to get the results. But firstly we can use a decision tree to see its accuracy. Then we use the random forest to see whether there can be better.

```
DecisionTreeRegressionModel (uid=DecisionTreeRegressor_57e1bf8f9dd4) of depth 14 with 25903 nodes
  If (feature 14 <= 847.25)
   If (feature 6 <= 74.5)
    If (feature 14 <= 306.45)
     If (feature 7 <= 0.5)
      If (feature 6 <= 62.5)
       If (feature 6 <= 29.5)
        If (feature 6 <= 3.5)
         If (feature 6 <= 1.5)
          If (feature 2 <= 27.6)
           If (feature 15 <= 9.5)

DT Root Mean Squared Error: 0.10
DT r2: 0.89
```

Notice that the depth of dt we estimated is 14 or 15 which is slightly small. So we can adopt random forest algorithm to see whether it can be improved. And instead of guessing the parameters, we use Model Selection or Hyperparameter Tuning to select the best model.

```
RandomForestRegressionModel (uid=RandomForestRegressor_9c77a9d4daa0) with 24 trees
  Tree 0 (weight 1.0):
    If (feature 14 <= 846.4)
     If (feature 6 <= 74.5)
      If (feature 15 <= 2.5)
       If (feature 14 <= 269.25)
        If (feature 8 <= 0.5)
         If (feature 0 <= 0.5)
          If (feature 15 <= 1.5)
           If (feature 6 <= 60.5)
            If (feature 14 <= 0.06665)
             If (feature 6 <= 40.5)
              Predict: 0.05531861341371514
             Else (feature 6 > 40.5)
              Predict: 0.5017071591774562

LR Root Mean Squared Error: 0.16
DT Root Mean Squared Error: 0.10
RF Root Mean Squared Error: 0.11
LR r2: 0.74
DT r2: 0.89
RF r2: 0.88
```

Unfortunately, the performance of random tree is not as good as decision tree. But we still get an optimal result with relatively low RMSE and high r^2

Conclusion

Until now, we have generated three kinds of models to evaluate winPlacePerc, which represents of win place percentage. For the last two models, we can find that feature

6 (kill Place) and feature 14 (walk Distance) has a huge effect on our target. But considering that the person who gets the champion is more likely to have a high kill

Place and walk a long distance, we may have to skip these two parameters and look at the linear regression expression to get some ideas:

$$y = 0.413646647358 + (0.0230870694198 * boosts) + (0.0179682771692 * assists) + \ldots$$

From the equation, we notice that boosts, assists and weapons Acquire have a high coefficient. We may get some conclusions saying that more boosts you throw, more assists you get, and more weapons you pick, you may have more chance to get the champion and eat the chicken!

In the end, we apply our best model of decision tree to generate the results of the test set given by the websites.

	Id	Prediction_WP
1	7f96b2f878858a	0.44290817142857136
2	eef90569b9d03c	0.5481449619771863
3	1eaf90ac73de72	0.7499144223693892
4	4616d365dd2853	0.22581101910827986
5	315c96c26c9aac	0.15800073452256047
6	ff79c12f326506	0.057620225593154474
7	95959be0e21ca3	0.007425869164183345

Dataset and Source Code

The dataset can be downloaded from the webpage below:

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

The source code is attached on the GitHub page. Or it can also be downloaded by following link:

https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2344877992759158/4406423933764510/6676117911825599/latest.html