PUBG Win Percentile Prediction

Ayush Yadav
IMT2017009
ayush.yadav@iiitb.org

Kaustubh Nair
IMT2017025
kaustubh.nair@iiitb.org

Sarthak Khoche
IMT2017038
sarthak.khoche@iiitb.org

I. Introduction

Given data about the ingame statistics of players who play the game PUBG (PlayerUnknown BattleGround), we tried to predict the finish placement of a player, i.e given data about the number of kills, the distance travelled, the number of assists and other such attributes, we used a random forest regressor and a linear regression model to predict the final place of the user on the leader board.

II. Data

The PUBG dataset is a public dataset hosted on Kaggle[1]. The dataset consists of train_V2.csv and test_V2.csv where the train_V2.csv is a 660MB file consisting of 4.5 million datapoints. train_V2.csv was used as the primary dataset for training and inferencing.

The dataset consists of ingame statistics of multiple players for multiple games. The parameters that were recorded for every player in a particular game were:

- **groupId** - Integer ID to identify a group within a match. If the same group of players plays in different matches, they will have a different groupId each time.
- **matchId** - Integer ID to identify match. There are no matches that are in both the training and testing set.
- **assists** - Number of enemy players this player damaged that were killed by teammates.
- **boosts** - Number of boost items used.
- **damageDealt** - Total damage dealt. Note: Self inflicted damage is subtracted.
- **DBNOs** - Number of enemy players knocked.
- **headshotKills** - Number of enemy players killed with headshots.
- **heals** - Number of healing items used.
- **killPlace—-** Ranking in match of number of enemy players killed.
- **killPoints** - Kills-based external ranking of player. (Think of this as an Elo ranking where only kills matter.)
- **kills** - Number of enemy players killed.
- **killStreaks** - Max number of enemy players killed in a short amount of time.
- **longestKill** - 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 longestKill stat.
- **maxPlace** - Worst placement we have data for in the match. This may not match with numGroups, as sometimes the data skips over placements.
- **numGroups** - Number of groups we have data for in the match.
- **revives**- Number of times this player revived teammates.
- **rideDistance** - Total distance traveled in vehicles measured in meters.
- **roadKills** - Number of kills while in a vehicle.
- **swimDistance** - Total distance traveled by swimming measured in meters.
- **teamKills** - Number of times this player killed a teammate.
- **vehicleDestroys** - Number of vehicles destroyed.
- **walkDistance** - Total distance traveled on foot measured in meters.
- **weaponsAcquired** - Number of weapons picked up.
- **winPoints** - Win-based external ranking of player. (Think of this as an Elo ranking where only winning matters.)
- **winPlacePerc** - The target of prediction. This is a percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match. It is calculated off of maxPlace, not numGroups, so it is possible to have missing chunks in a match.

III. Preprocessing

The given dataset was relatively clean and complete. Yet, a few outliers were observed in the given dataset. Hence, the following preprocessing techniques were implemented on the training dataset.

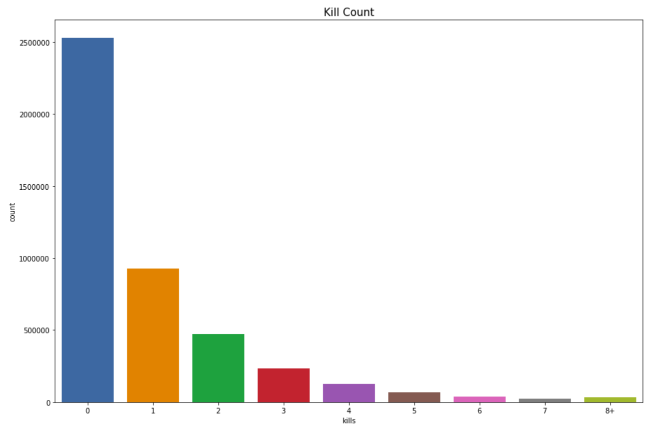
A. Null Values

There was only one row with a missing value, where the winPlacePerc value was missing. On further exploration, it turned out the match was an illegal match where the number of players in the match were only 1. Rather than replacing the null value, the row was removed because we still have ample datapoints to train on.

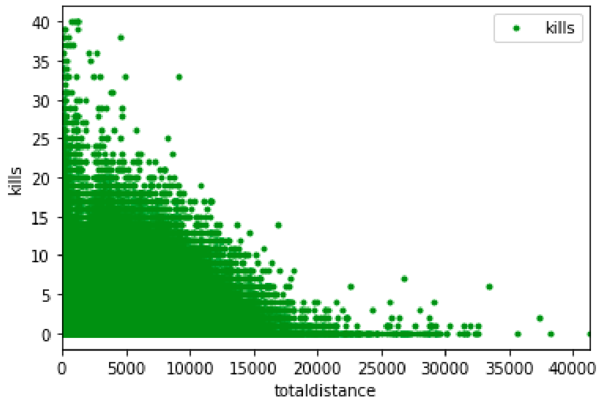
B. Handling Outliers

Outlier handling was done majorly to remove cheaters who have used mods such as noRecoil (a mod that eliminates recoil from a gun making it easier to have headshot kills), aimbots (mods that help aim), etc. These outliers were handled using the following ideas:

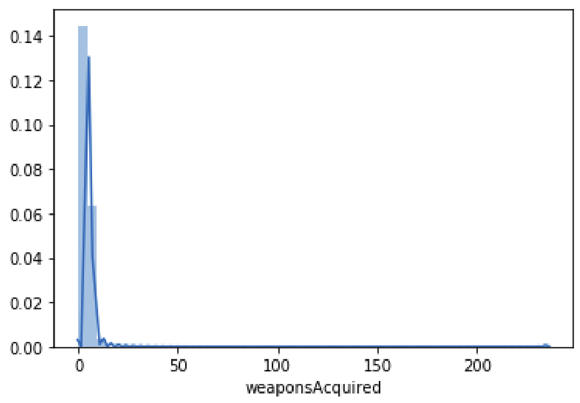
- Number of kills and headshot kills: In a game of 100 players it is highly unlikely that a player encounters 50 players and even more unlikely that a player kills more than 40 of these players. All datapoints that had more than 40 kills were removed on this account.



- Kills without moving: It is highly unlikely that a player finds any weapons without moving a single meter, it is not likely at all that a player has multiple kills without moving a single meter. All such datapoints who had more than one kill without moving a single meter were deleted. This was done using a new feature ”killswithoutmoving”(see more in the section: Feature Engineering)

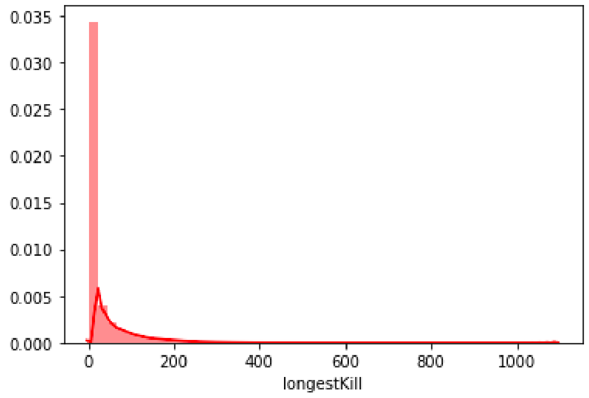


- Only swimming and still winning: All users that had only swam in the match and still managed to get a 80% rank in their match were considered to be troll players, who aren’t playing the game in the right spirit and thus would create outlier data.
- Excessive weapons acquired: Users who had acquired more than 50 weapons were considered to be cheaters(outliers) and hence were removed.



- Few General Outliers: Other outliers that were also removed were:

  LongestKillDistance: As pointed out in [2] it is not genuinely possible to kill someone from a more than 800m away. Hence all values >950m away were considered to be outliers and hence removed.

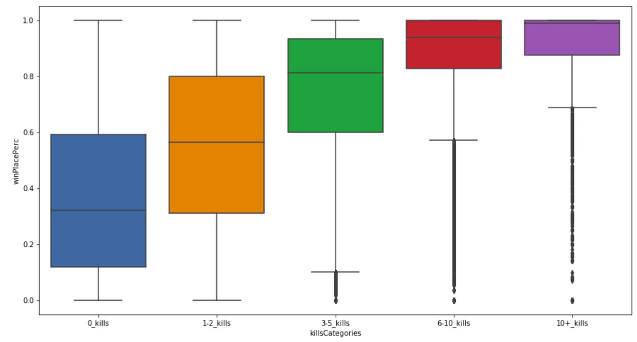


  100% headshots on many kills : All players with more than 12 kills who only had headshots were considered to be cheaters using an AimBot and hence these rows were removed.

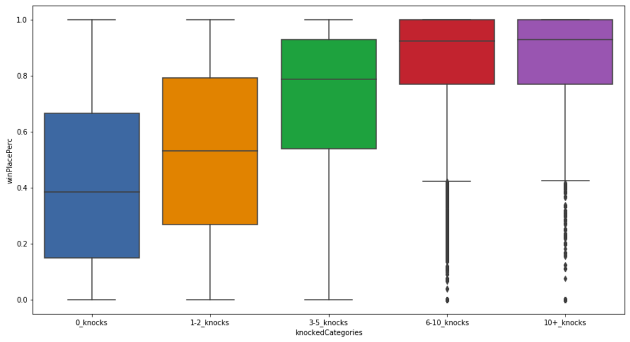
  Rambo like players: All players that had more than 30 kills and had acquired more than 30 weapons without moving more than 50m, were considered to be cheaters and hence were removed.

IV. Exploratory Data Analysis

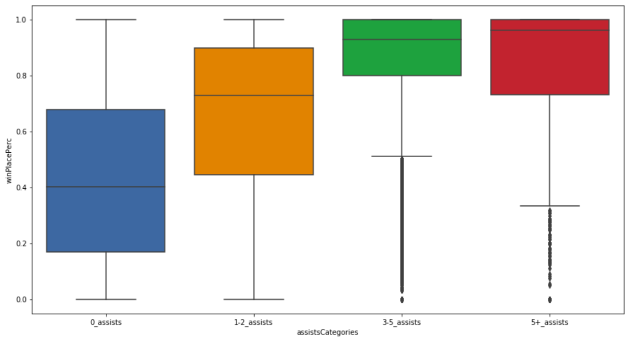
- Kills, Players knocked down and Assists: A linear relationship with win place percentile was observed in the features like number of kills, number of players knocked and number of assists. This is justified as more the damage done to the opponent, the more chances of staying in the course and winning.



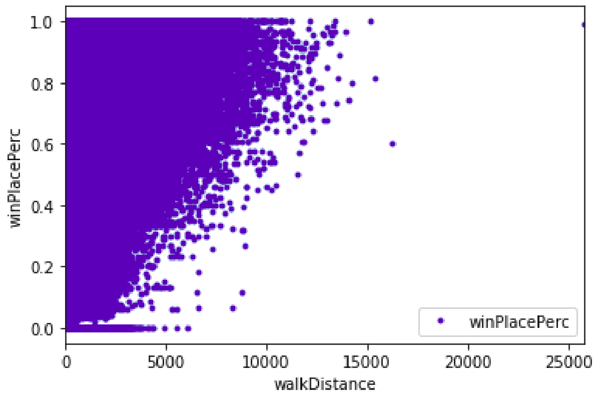
Similarly, for players knocked down



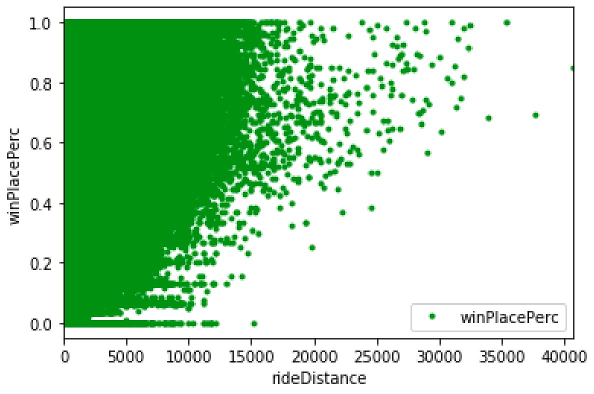
And also for assist



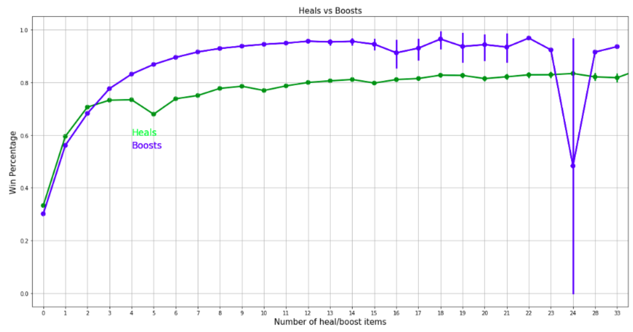
- Distance travelled, Walk distance and Ride distance : Again, a linear relation was observed between the walk distance and win place percentile.



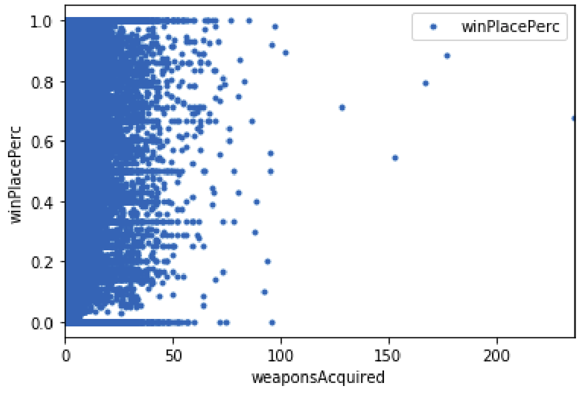
Similarly, the relation between Ride distance and win place percentile came out to be linear as well.



- Heals and Boosts :



- Weapons Acquired : Although expected, no significant relationship was observed between weapons acquired and win place percentile.



V. Feature Engineering

- playersJoined: PUBG expects a 100 people to participate in every game, but many games have less than a 100 participants. The dataset doesn't provide any attribute about the number of people that participated in the particular game, therefore a new feature was created which counts the number of players that have joined the match for for every datapoint.
- healsandboosts: This new feature sums the number of heals and boosts used by a single player. This was done because these are similar attributes and have a similar correlation to the winPlacePerc.
- totalDistance: This feature was created to find the total distance travelled by a player in a game. TotalDistance

is the sum of walkDistance, rideDistance and swimDistance.

- killswithoutmoving: The easiest way to detect cheaters is by checking whether a player had multiple kills without moving at all. This boolean feature was set to True for all those datapoints which had more than 1 kill without moving any distance on the map.
- Normalizing Features: When there are 100 players in the match, it is easier to manage atleast one kill since the number of possible targets are 99, but a match that has only 70 players will have only 69 possible targets, thus we decided to normalize the following features using the playersJoined attribute:

KillsNorm: Kills were normalized in such a way that 1 kill in a game of 100 participants counted as 1 kill, but 1 kill in a game of 50 participants counts as 1.5 kills.

damageDealtNorm: Similar to killsNorm, this attribute was adjusted to represent that a player is likely to deal more damage if he's playing a match with more players.

maxPlaceNorm: The maximum placement of a player can be worse if there are more players that are participating.

matchDurationNorm: The maximum duration of a match must be much lesser if there are only 50 players in the game since the map will shrink faster to accommodate the less number of players, whereas if there are 100 players in a match, the map shrinks at a normal rate hence increasing the matchDuration. This feature was also normalized using the KillsNorm feature.

VI. Model and Training

After Exploratory Data analysis, we concluded that the following features had a correlation (both positive and negative) with the dependent feature (winPlacePerc).

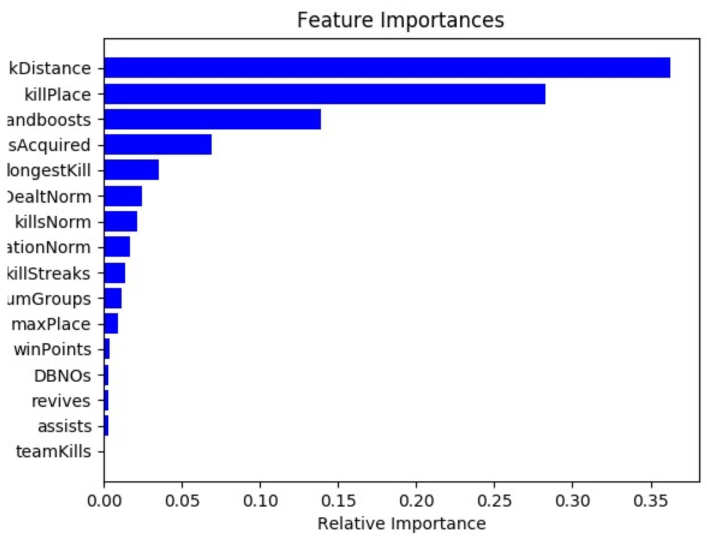
To start off, we trained a basic linear regression model, this baseline model gave us a RMSE value of 0.164 and a score of 0.70.

A. Linear Regression and Polynomial Regression

After feature engineering and outlier elimination the linear regression model gave an RMSE value of 0.12652 and a model score of 0.83044. Applying polynomial regression with degree 2 gave better results (RMSE = 0.090804, score = 0.8981), the RMSE values increased with an increase in the degree of the polynomial, indicating overfitting

B. Random Forest

Further, we used a Random Forest model, with hyperparameters as ¡depth=10, number of trees=50¿, which gave us the best RMSE value of 0.080831 and a score of 0.9030801. On plotting the feature importance given by random forrest, it can be noticed that distance moved by a player is the most important feature in determining his/her winabiliity.



C. Ridge, Lasso and ElasticNet Regression

Lasso regularization doesn't give good results since it drives some of the coefficients to zero, but as seen in *section 4*, all features considered for training have a good correlation with the target parameter. The results given by Lasso regression is an RMSE of 0.15693 and a model score of 0.73915. Ridge regression generalizes better, thus giving us a better result of a of RMSE 0.12654 and a score of 0.83043 ElasticNet is a combination of Lasso and Ridge regression, and thus gave intermediate results, i.e. a RMSE of 0.1505 and a model score of 0.75599

VII. Testing and Results

The table below shows the models that we used along with their RMSE values and scores. The model scores are based on cross validation.

Model	RMSE	Model Score
Linear Regression	0.12652	0.83044
Polynomial Regression (deg = 2)	0.09804	0.8981
Random Forest	0.080831	0.930801
Ridge Regression	0.12654	0.83043
Lasso Regression	0.15693	0.73915
ElasticNet Regression	0.1505	0.7599

The pickle files for the models can be found on this google drive.

The jupyter notebook can be viewed at this github link.

References

[1] https://www.kaggle.com/c/pubg-finish-placement-prediction/data
[2] https://www.reddit.com/r/PUBGMobile/comments/94tpiu/is_this_possible _a_800m_kill/
[3] https://scikit-learn.org/stable/documentation.html
[4] https://pandas.pydata.org/pandas-docs/stable/
[5] https://matplotlib.org/3.1.1/contents.html