ML Hackathon Report

- Problem Statement: We are provided with a large number of anonymized PUBG game stats, formatted so that each row contains one player's post-game stats. The data comes from matches of all types: solos, duos, squads, and custom; there is no guarantee of there being 100 players per match, nor at most 4 player per group. Model should predict players' finishing placement based on their final stats, on a scale from 1 (first place) to 0 (last place).

- Dataset used: Dataset is taken from Kaggle (https://www.kaggle.com/c/pubg-finish-placement-prediction/data).It consist of only train set (train_V2.csv) as it consisted of about 4.4 million datapoint so used it to create both train and test set. Data fields are:
  - DBNOs - Number of enemy players knocked.
  - 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.
  - headshotKills - Number of enemy players killed with headshots.
  - heals - Number of healing items used.
  - Id - Player’s Id
  - 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.) If there is a value other than -1 in rankPoints, then any 0 in killPoints should be treated as a “None”.
  - killStreaks - Max number of enemy players killed in a short amount of time.
  - kills - Number of enemy players killed.

- 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.
- matchDuration - Duration of match in seconds.
- matchId - ID to identify match. There are no matches that are in both the training and testing set.
- matchType - String identifying the game mode that the data comes from. The standard modes are “solo”, “duo”, “squad”, “solo-fpp”, “duo-fpp”, and “squad-fpp”; other modes are from events or custom matches.
- rankPoints - Elo-like ranking of player. This ranking is inconsistent and is being deprecated in the API’s next version, so use with caution. Value of -1 takes place of “None”.
- 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.) If there is a value other than -1 in rankPoints, then any 0 in winPoints should be treated as a “None”.
- groupId - 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.
- numGroups - Number of groups we have data for in the match.
- maxPlace - Worst placement we have data for in the match. This may not match with numGroups, as sometimes the data skips over placements.
- 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.

- Approach:

  As we a have to predict the winning percentile placement so we decided Regression model.
  then went with data preprocessing as discussed bellow in analysis of dataset
  then Data Visualization as it will helot study the different distributions and to tell the need of feature engineering
  then feature engineering is needed if we there is need for removing any column or adding extra features or normalizing data.
  Model training will be started with linear regression then stochastic gradient Decent, Decision Tree, Random Forest and finally XGBoost.
  Depending on the test metric value will decide which model is better for the Prediction.

- Analysis of Dataset:
- First we have dropped all the unnecessary columns like id, matchid and grouped.
- Then we started with **Data Preprocessing**:

A. **Missing data**: As data can be missing during data extraction or collection and they needed to be handled because they reduce quality of our test metrics. As they were very less compared to the entire dataset so we decided to drop them.

B. **Lower frequency of data**: In matchType column because some categories were having lower frequency so those rows were also dropped.

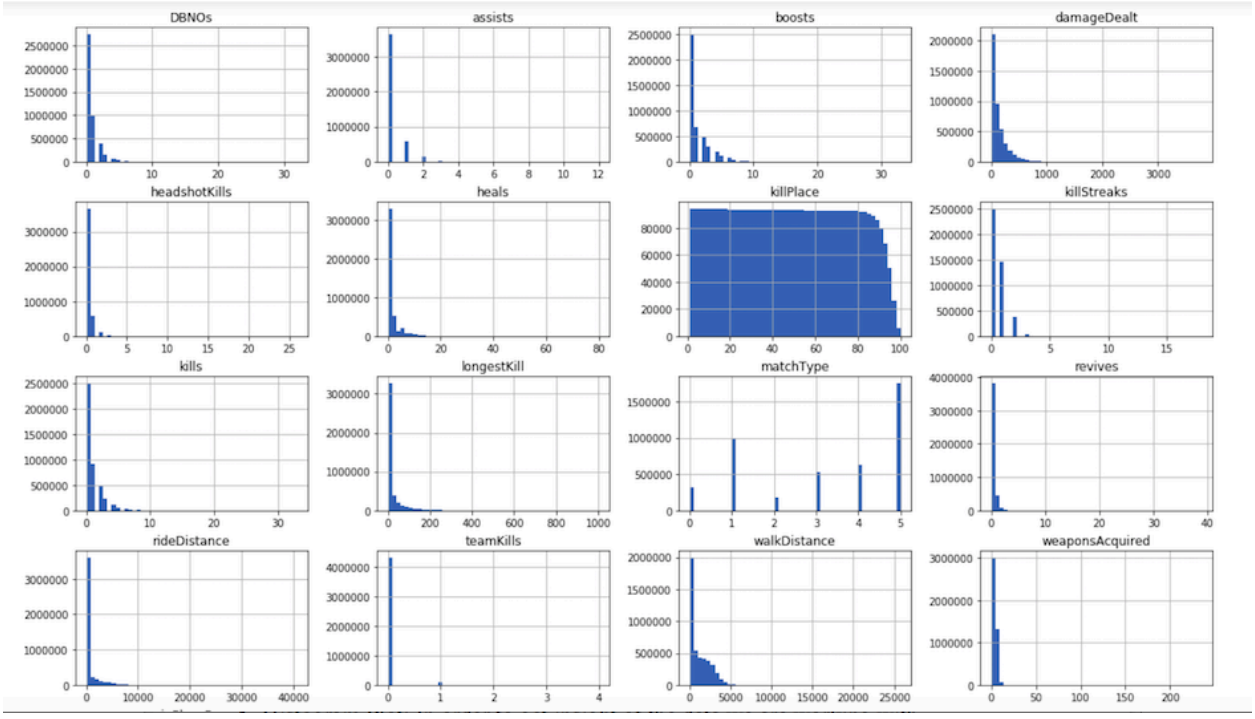
C. **Correlation**: Correlation is the statistical measure that describes association between different random variable. Bellow table gives the correlation between the winPlacePerc column and all other. thus columns with lower correlation where dropped. as we know correlation is not the right criteria to decide whether columns have relation or not. So, we used Pandas scatter_matrix function to find correlation, which plots every Every numerical attribute against every other numerical attribute which in our case will be 100 graph which will not fit in a page. So we plotted for some important attributes like killPoints, matchDuration and teamKills. Which again showed no correlation. Hence we finally dropped all the Undesirable columns.

```
assists            0.305667
boosts             0.636463
damageDealt        0.450645
DBNOs              0.284162
headshotKills      0.282640
heals              0.430058
killPlace         -0.720910
killPoints         0.013001
kills              0.431588
killStreaks        0.377973
longestKill        0.412435
matchDuration     -0.004197
maxPlace           0.038446
numGroups          0.040720
rankPoints         0.013751
revives            0.241897
rideDistance       0.345415
roadKills          0.031816
swimDistance       0.150905
teamKills          0.016416
vehicleDestroys    0.072860
walkDistance       0.814679
weaponsAcquired    0.617926
winPoints          0.007000
winPlacePerc       1.000000
Name: winPlacePerc, dtype: float64
```

D. **Label encoding**: Used label encoder to convert categorical data into numbers (like incase of matchType column). As in case of One hot coding number of columns will unnecessary will get increased.

- Visualization:

To understand patterns in the given data we need to perform EDA(Exploratory Data Analysis)



- Feature Engineering:

A. Overcome Tailheaviness:

As we can see a number of features are skewed to the right side. This is not a good fit for our models as models will be unable to detect patterns in such kind of data.

There are 3 popular ways to deal with skewness in data.

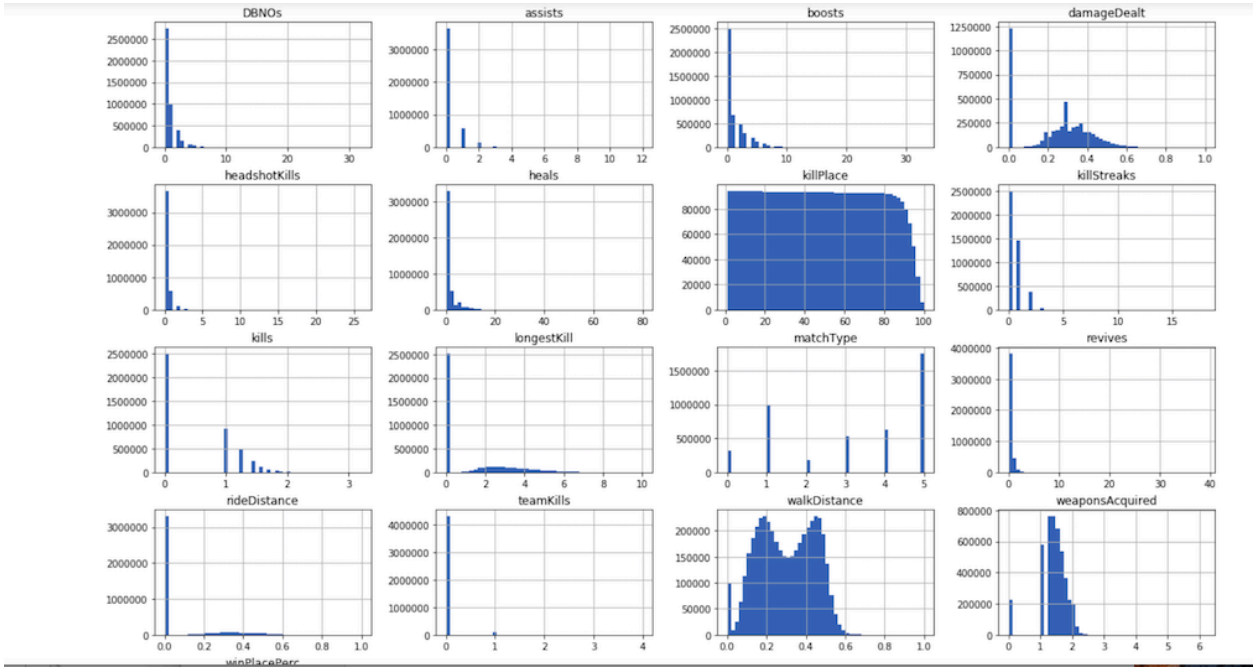
1. Convert to log scale

2. Convert to cube root scale

3. Convert to square root scale

Ofcourse the log scale performs best. But the log scale does do well if the data contains a lot of 0's.

So we converted our data to cube root scale and divided the data columns by their maximum value. This will ensure that the data appears gaussian and has a range of 0 to 1.



B. Normalization: their are features with high magnitude which will weigh high in Euclidean distance calculation than the features with low magnitude. So, we converted values of each column between 0 to 1.

C. Banding: Some column like matchPlace has values between 1 to 100. So, converted it into bands of 10 depending its correlation with WinPlacePerc column. Then label encoded it for machine to understand.

- ## Model Building:

We started our model building by feeding our data to linear regression model. But a linear regression model is a fairly simple model and does not perfom so well. Our MAE is pretty high and we move towards other options.
We then train a Stochiastic Gradient Descent Model. A SGD will never converge to the minimum but will jump around the minima point. This meant that true minima was out of reach.
We moved towards trees next. Specifically decision trees and random forests. Random forest is a colletive of multiple decision trees. All of which are weak learners. We take 10 such decision trees. The random forest gives us a better result than the previous 3 models.
We next try our hands with gradient boosting technique,
specifically XGBoost Regressor.
XGBoost regressor gives us the best results.We use it as our primary model.

- ## Training and Validation :

In order to train this model we have removed a part of data for testing like 20% and are training the model in the available 70% of data.
For validation as holdout method gives high variance for the unseen data . So, we are using K-Fold cross validation technique. In this entire data is divided in 10 folds out of which 9 folds are used as train set and remaining one is used as test set and each time different subset is selected. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set.

- Test Metrics:

  In case of regression , test metrics used are MAE, RMSE, R square and Adjusted R square. Only MAE and RMSE are useful to compare between different models. We are using MAE as our test metric because we are only interested in knowing the difference between ground truth and predicted value From the bellow table we decided to use XGBoost as the model for our dataset as it was giving highest accuracy and lowest MAE.

	Model	MAE
1	XGBoost Regressor	0.078348
3	Random Forest	0.079446
2	Decision Tree	0.084815
0	Linear Regression	0.093490
4	Stochastic Gradient Decent	0.093553