

Neural Networks

EE-258/Fall 2018

A Project Report on
PUBG Finish Placement Prediction

- A Regression problem to predict the win percentages using Multilayer Perceptron (MLP) Algorithm.

Under the Guidance of

Prof. Birsen Sirkeci

Submitted By

Deepak Shivani

012560340

Introduction:

PlayerUnknown's Battlegrounds (PUBG) is an online multiplayer battle royale game developed and published by PUBG Corporation. In the game, up to one hundred players using a parachute land onto an island and scavenge for weapons and equipment to kill others while avoiding getting killed themselves. The available safe area of the game's map decreases in size over time, directing surviving players into tighter areas to force encounters. The last player or team standing wins the round.

The team at PUBG has made official game data available for the public to play with. We are given over 65000 games worth of anonymous player data, split into training and testing sets. The main objective of this project is to predict final placement from the final in game stats and initial player ratings. This is a regression problem since we are predicting the win percentages of each player. The problem is worked on with the Multilayer perceptron that is a fully connected feed forward neural network.

Dataset Visualization:

As discussed we are provided with a large number of anonymous PUBG game stats, formatted so that each row contains one player's post-game stats. The data comes from matches of all types: solos (single play), duos (in a pair), squads (a team of four), and custom; there is no guarantee of there being 100 players per match, nor at most 4 player per group. In a PUBG game, up to 100 players start in each match (matchId). Players can be on teams (groupId) which get ranked at the end of the game (winPlacePerc) based on how many other teams are still alive when they are eliminated. In game, players can pick up different munitions, revive downed-but-not-out (DBNO - knocked) teammates, drive vehicles, swim, run, shoot, and experience all of the consequences -- such as falling too far or running themselves over and eliminating themselves.

The dataset consists of 29 features each having a unique dependency on the target value. In this problem 'winPlacePerc' is considered to be our target variable.

```
In [3]: train_df.info()

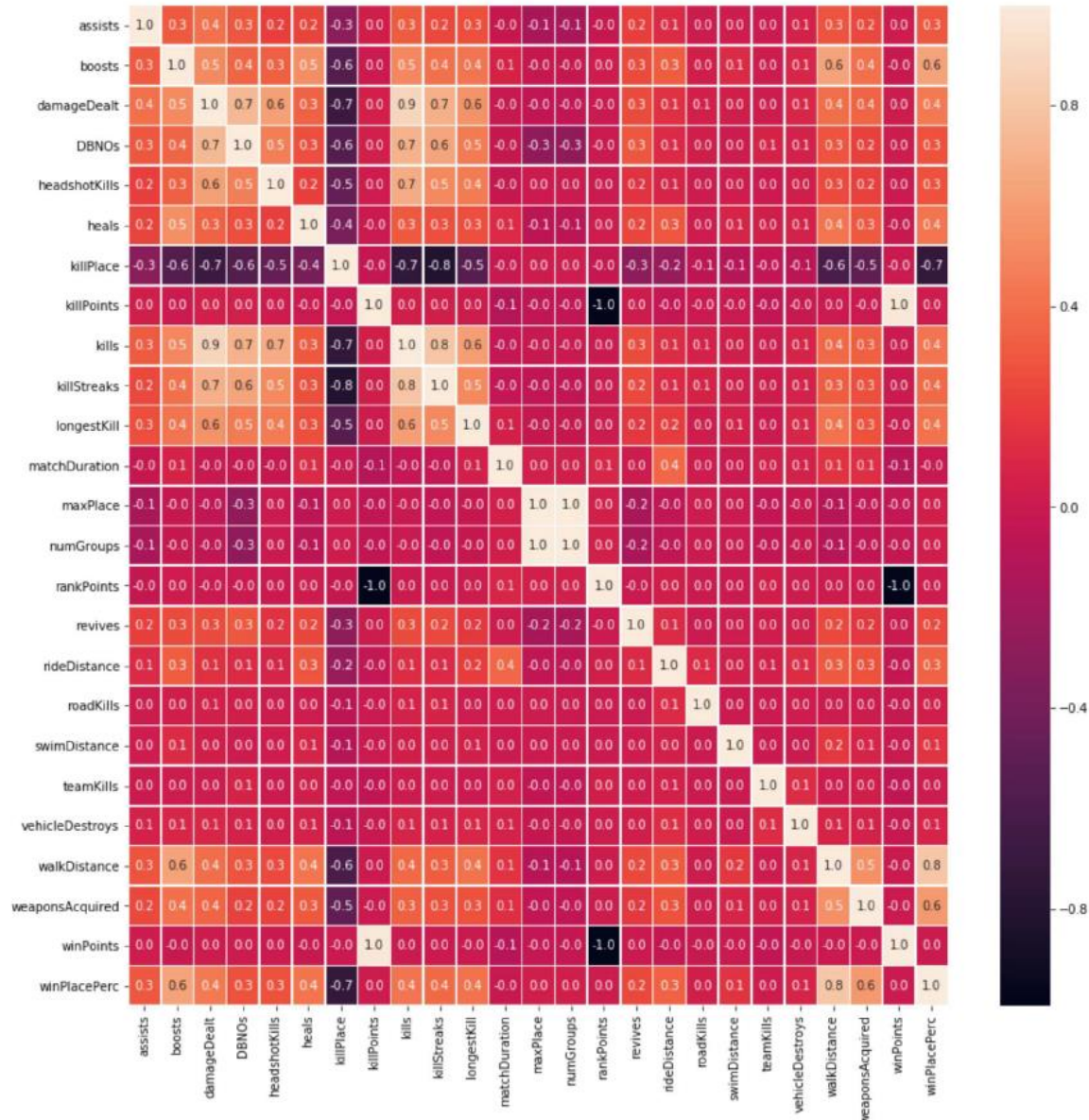
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4446966 entries, 0 to 4446965
Data columns (total 29 columns):
Id                object
groupId           object
matchId           object
assists           int64
boosts            int64
damageDealt       float64
DBNOs             int64
headshotKills     int64
heals             int64
killPlace         int64
killPoints        int64
kills             int64
killStreaks       int64
longestKill       float64
matchDuration     int64
matchType         object
maxPlace          int64
numGroups         int64
rankPoints        int64
revives           int64
rideDistance      float64
roadKills         int64
swimDistance      float64
teamKills         int64
vehicleDestroys   int64
walkDistance      float64
weaponsAcquired   int64
winPoints         int64
winPlacePerc      float64
dtypes: float64(6), int64(19), object(4)
memory usage: 983.9+ MB
```

The description of all the available features is discussed below:

- **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.

A correct choice or combination features leads to better solutions in both the test and the training sample. There are no definite path for feature engineering. It is very much guided by available data. Hence data exploration is very important before feature selection.

The simplest way to see relation between features is correlation. Here with the help of the seaborn library we are plotting the heat map which gives us a better view of correlation. A heat map is a graphical representation of data that uses a system of color coding to represent different values.



- As it is seen from the graph above that there is a very high negative correlation between 'killPoints' and 'rankPoints' and there is high positive correlation between winPoints and killPoints, maxPlace and numGroups, kills and damageDealt.
- Highly correlated variables with 'winPlacePerc' are 'walkDistance' (0.8), killPlace (-0.7), boosts (0.6) and weaponsAcquired (0.6).

Data Preprocessing:

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Below mentioned are some the preprocessing tricks that were used in this project.

- Samples that have matches with more than one player are only considered, there are matches with no players or just one player those samples could affect our model badly.
- Since any kind of distance covered by the player has some effect on 'winPlacePerc' so we can make a new feature by adding the 'rideDistance', 'walkDistance' and 'swimDistance' as 'totalDistance'.
- Replacing all the negative 'rankPoints' value to be 0, to avoid corrupted results.
- The features like Id, groupId, matchId and matchType are categorical and thus not play a crucial role in our model, therefore are removed from the dataset.
- We are making new features by indicating the mean of the features and putting them into a rank form such that max value will have the highest rank.

Methodology:

In this project we are using the Multilayer perceptron algorithm implemented using keras python library. Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot): \mathbb{R}^m \rightarrow \mathbb{R}^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X=x_1, x_2, \dots, x_m$ and a target y , it can learn a non-linear function approximator in our case it is regression. Below mentioned are some of the details of model used in this project.

- Our model implements a multi-layer perceptron (MLP) that trains using backpropagation with ReLu as the activation function.
- MLP uses parameter alpha for regularization (L2 regularization) term which helps in avoiding overfitting by penalizing weights with large magnitudes.

- MLP trains using Adam (Gradient Descent) which updates parameters using the gradient of the loss function with respect to a parameter that needs adaptation where η is the learning rate which controls the step-size in the parameter space search. Loss is the loss function used for the network. Adam is similar to SGD in a sense that it is a stochastic optimizer, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments. With Adam, training supports online and mini-batch learning.
- The neural network consists of :
Number of neurons in each layer: Input layer = 170, Hidden layer1 = 136, Hidden layer2 = 136, Hidden layer3 = 136, Hidden layer4 = 136, Output layer = 1.
- Performance Metric Used: Mean Absolute Error (MAE) is the simplest measure of forecast accuracy, it is simply, as the name suggests, the mean of the absolute errors. The absolute error is the absolute value of the difference between the forecasted value and the actual value.

The advantages of Multi-layer Perceptron are:

- Capability to learn non-linear models.
- Capability to learn models in real-time (on-line learning) using partial fit.

The disadvantages of Multi-layer Perceptron (MLP) include:

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

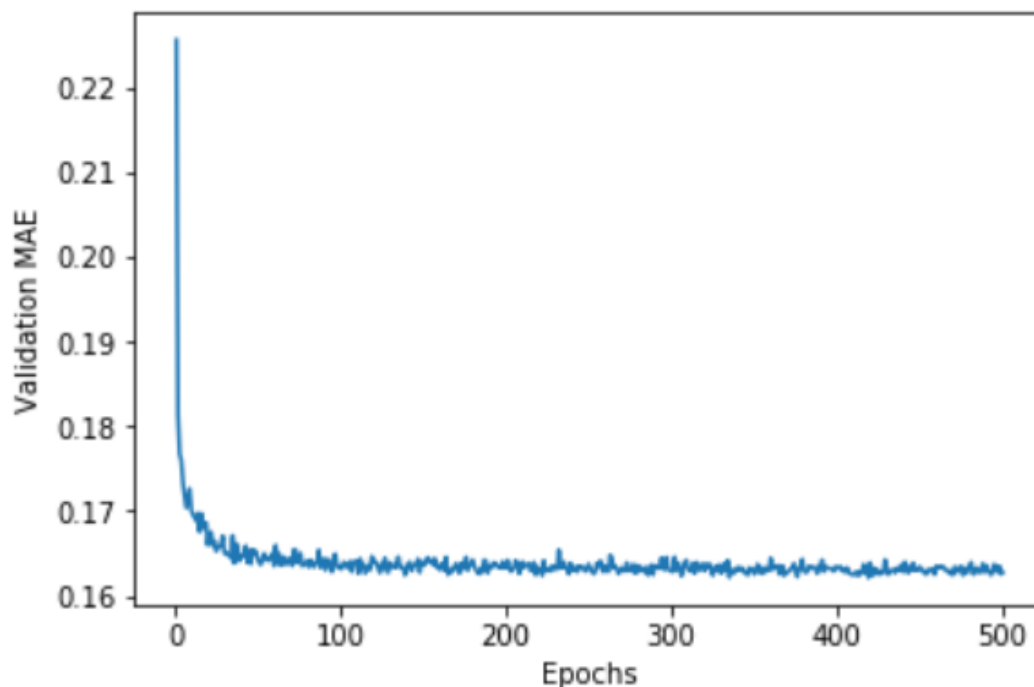
Conclusion:

Changes made as compared to the Baseline Model:

- As compared to the baseline model, a lot of feature engineering was involved in this project. It resulted in the enhancement of the model by manipulating the features based on their correlation values.

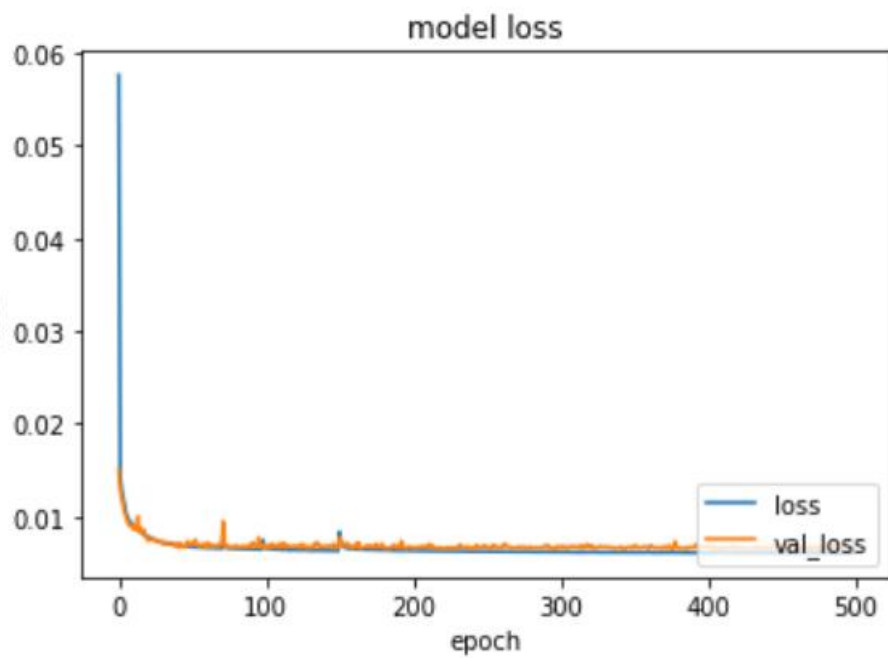
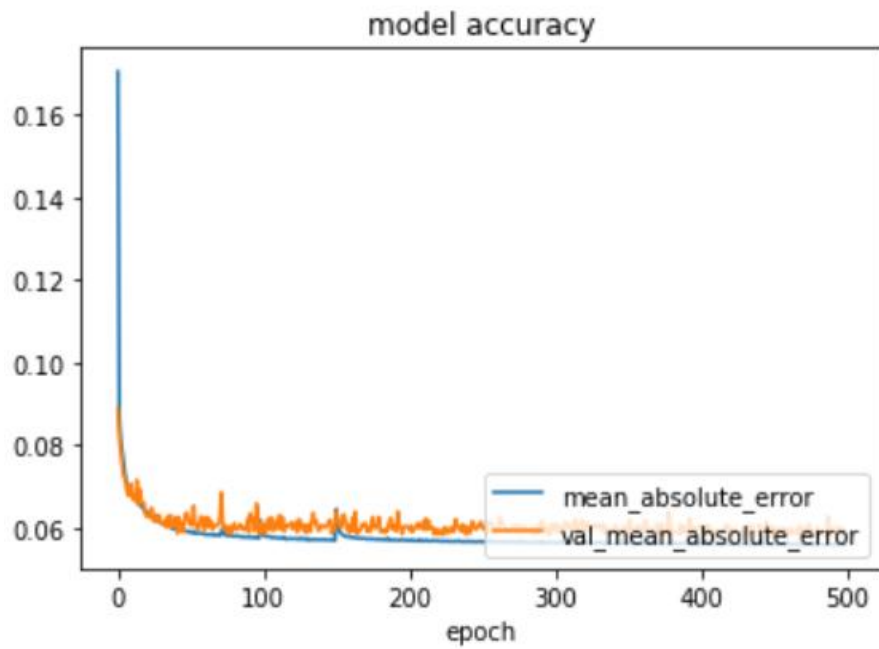
- To avoid overfitting the model was made simpler by reducing the number of hidden layers in the network. Also the number of neurons were made consistent in each layer.
- The dropout regularization technique was removed from the model since it was not playing a crucial role in determining the 'winPlacePerc'.
- Since we are using MLP as our algorithm the data was scaled before training as MLP is sensitive to feature scaling.

Performance:



- The figure is plotted between the validation MAE and the total number of epochs.
- At the start of training, the validation MAE is very high but reduces exponentially as the number of epochs increases.
- The validation MAE saturates to the lowest MAE value once the number of epochs are about to end.
- The error was consistently reduced as compared to the baseline model.
- The figure model accuracy is the comparison between the MAE of validation dataset and training dataset.
- It gives us the rough idea about the accuracy of our model based on MAE.
- The figure model accuracy is the comparison between the MAE of validation dataset and training dataset.

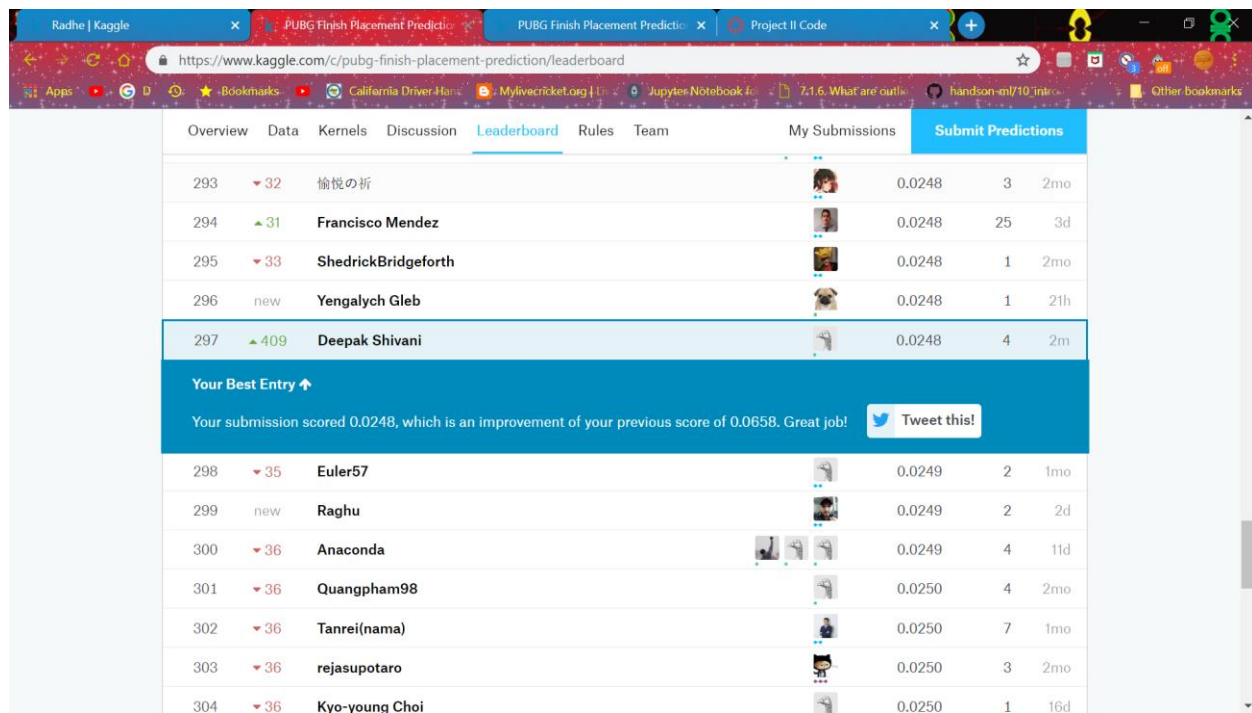
- It gives us the rough idea about the accuracy of our model based on MAE.



- After training the model, it is evaluated on the given test dataset to predict the winPlacePer of each player.
- The prediction made is stored in a .csv file and later uploaded on kaggle to get a rank on competition leader board.
- The performance can be improved by the use of effective feature engineering i.e. using some techniques on the data set such as combining highly correlated features, effective feature selection etc.

RESULT: The Mean Absolute Error for this problem turned out to be 0.0248.

Kaggle Rank:



| Rank | Change | Player | MAE | Submissions | Time |
|---|--------|---------------------|--------|-------------|------|
| 293 | ▼ 32 | 愉悦の祈 | 0.0248 | 3 | 2mo |
| 294 | ▲ 31 | Francisco Mendez | 0.0248 | 25 | 3d |
| 295 | ▼ 33 | ShedrickBridgeforth | 0.0248 | 1 | 2mo |
| 296 | new | Yengalych Gleb | 0.0248 | 1 | 21h |
| 297 | ▲ 409 | Deepak Shivani | 0.0248 | 4 | 2m |
| Your Best Entry ↑ Your submission scored 0.0248, which is an improvement of your previous score of 0.0658. Great job! Tweet this! | | | | | |
| 298 | ▼ 35 | Euler57 | 0.0249 | 2 | 1mo |
| 299 | new | Raghu | 0.0249 | 2 | 2d |
| 300 | ▼ 36 | Anaconda | 0.0249 | 4 | 11d |
| 301 | ▼ 36 | Quangpham98 | 0.0250 | 4 | 2mo |
| 302 | ▼ 36 | Tanrei(nama) | 0.0250 | 7 | 1mo |
| 303 | ▼ 36 | rejasupotaro | 0.0250 | 3 | 2mo |
| 304 | ▼ 36 | Kyo-young Choi | 0.0250 | 1 | 16d |