PUBG Player Placement Prediction

Author(s)
Rahat Ranyal: rranyal

1 Background And Introduction

Player Unknown Battle Ground's (PUBG) is an online Multiplayer fight royale diversion created by PUBG Corporation. It is a player versus player activity amusement in which close to 100 player battle in a fight royale. Players can enter the match solo, couple or with a little collaborate to 4 individuals. In any case, there are occasions and custom amusements which permit group sizes more than 4. We will address this inconsistency amid highlight building. The last individual or the last group alive successes the match. The player needs to thump down their rivals in order to make due till the last. Amid the diversion, players seek structures, phantom towns to discover weapons, vehicles, defensive layer, and other hardware.

We previously utilized the LGBM (Light Gradient Boosting Machine) (5) Regressor to prepare our preparation dataset in the wake of doing exploratory investigation on the information. We looked at 10 different models specifically Neural Network, Multiple Linear Regression, Decision tree, and so forth with the equivalent dataset. We announced the exactness of each model that we jumped on the approval information.

1.1 Objective

To predict the final ranking and placement from the game statistics and the initial player ratings.

The overview of the steps involved in the project are:

1. Data Cleaning and Exploratory Data Analysis
2. Visualizing the data and feature engineering
3. Training Models on Training data and then drawing comparison between them using validation data
4. Choosing the best model and fitting the Test data on it5. Reporting the Accuracy of the final chosen model.

2 Method

The main objective is to predict the ranking of the player from the initial player ratings and statistics available in data. So first we are doing exploratory data analysis in which we focus primarily towards getting to know about how each attribute contribute towards the prediction. We use the help of seaborn package in python to plot various kind of graphs between attributes to get data insights. We use correlation to find out the attributes which are more related to the target attribute. We also form new attributes from the existing attributes which would give much more insight for prediction.

$$cor = \sum_{i=0}^{n} \frac{(x - x_i)(y - y_i)}{n - 1}$$

In our project we have used 11 different algorithms for our player placement prediction model. In those 11 algorithms, 4 were Regression models, 4 were ensemble models (out of which 3 were boosting models), 2 were Neural Network and a Decision Tree model. We have also used Grid Search for parameter tuning in the case of Boosting model which we will discuss further. The 4 Regression Models that we have used are:

1. Multiple Linear Regression
2. Ridge Regression
3. Lasso Regression
4. Elastic Net Regression

In Multiple Linear Regression (6) we tend to learn about multiple uncorrelated independent variable with the dependent variable or the criterion. In our project there were 28 dependent variables and 1 independent variable winPlacePerc(which gives the winning percentile of particular player). The equation is given as:

$$Y = \beta_0 + \beta_1.x_1 + \beta_2.x_2 + \beta_3.x_3 + ...$$

The other 3 Regression model that are used, uses the technique of Regularization (7) to reduce the magnitude of coefficients according to the effect they have on the target variable. They are also called as shrinkage models because they reduce the parameter coefficients, so as to prevent multicollinearity. The cost function of Ridge Regression is given as:

$$\mathrm{X} - x_i.\beta)_2 + \lambda \mathrm{X} \beta_{j2} \, (Y$$
$$i \qquad\qquad\qquad\qquad j$$

where lambda is the coefficient denoting the penalty term. higher the value of lambda more the coefficient decrease happens. Lasso Regression is also similar to the Ridge regression, but the difference here is that it also regularizes the coefficients in the way that it makes the magnitude of some coefficients as 0 according to the influence it has. This has the effect of doing automatic feature selection on the data and increasing the interpretability of the data. While ridge regression never makes the coefficient magnitude as zero. So Lasso regression performs well in the case where there are many attributes. Its equation is given as:

$$\mathrm{X} - x_i.\beta)_2 + \lambda \mathrm{X} \beta_j \, (Y$$
$$i \qquad\qquad\qquad\qquad j$$

Now Elastic Net (8) is a combination of both Ridge and Lasso Regression. Both penalty terms of Ridge and Lasso are added to the cost function and is shown as following:

$$\mathrm{X} - x_i.\beta)_2 + \lambda_1 \mathrm{X} \beta_j + \lambda_2 \mathrm{X} \beta_{j2}$$
$$(Y$$
$$i \qquad\qquad\qquad\qquad j \qquad\qquad\qquad j$$

Other than Regression we have used 3 Boosting (10) Models namely AdaBoost, Gradient Boosting and Light Gradient Boosting Machine (LGBM). Firstly Boosting is an ensemble technique to create a strong classifier from a combination of weak classifiers. This occurs by creating a model from the train data. Then creating a second model that attempts to correct the errors made in the first model. Models are added till the training set is predicted accurately. So Adaboost retrains the algorithm iteratively by choosing the training set based on accuracy of previous training set. Each training data is given a particular weight according to the accuracy achieved and the algorithm runs again.

Gradient Boosting Model (9) is a supervised technique where the main objective is to minimize the Loss function i.e., the Mean Squared Error(MSE). It uses Gradient Descent to minimize the loss and uses a learning rate alpha to find the updated value of predicted values. LGBM is a type of gradient boosting algorithm which uses decision trees as its framework. The most inherent part of this model is that it can handle a very large size of data and takes very low memory to run. Due to its

efficiency, accuracy and interpretability it has started to be widely used for very large data sets. Similarly it is not advised to use this model with small data set as it is prone to overfitting.

LGBM is an ensemble model of decision tree which learns by fitting negative gradients which are also known as residual errors. Suppose we have n identical and independent distributed (x1,x2,...xn) with dimension s in a Gradient Space. In every iteration of gradient boosting, the residuals of loss function with respect to output of the model are denoted by (g1,g2,...gn). So the model splits each node at the point where the Information Gain is the maximum.

The next model that we explored was Decision Trees. Decision Tree (11) works on the concept of dividing the tree based on the Information Gain attained from the attributes of the given dataset. Information Gain is based on the concept of entropy and information content. Entropy is the measure of randomness of an attribute with respect to target attribute. It is defined as:

$$H(t) = -\sum_{i}^{j} p_i . \log_2(p_i)$$

where H(t) is the entropy of the atttribute. p is the probability of each class present in the child node. Now Information gain of the attribute is defined as:

$$IG(T/a) = H(T) - H(T/a)$$

Here H(t) represents entropy of the parent while H(T/a) is the weighted sum of entropy of all the children in the attribute. We also evaluated our result with the Random Forest Algorithm. Random forest (12) works similar to Decision tree but the catch here is that it makes multiple decision trees and merges them together to get a more accurate and stable predictions.

We also used Neural Network. Neural network is composed of three layers, namely input layer, hidden layer and the output layer. The activation function is used to find the weighted input of each unit in the the layers. Hyper parameter tuning is done to achieve model optimization. We used Keras library to help us train the best model for our data. We extended one more step forward by using Deep learning for our dataset. Deep learning is also a type of Neural Network consisting of hierarchy of layers in which each layer further transforms the data into more abstract representation. In deep learning more the layers, higher will be the features learned by the model. The output layer combines all these features and make a prediction. This way it differs from Neural Network. While simple Neural Network uses only one hidden layer which is not suitable for learning complex features, deep learning uses multiple hidden layers to learn these complex features which might risk overfitting. To avoid this, we use Batch Normalization which normalizes and scales the output of the activation at each layer to avoid values going to the extremes. This allows each layer to learn differently from other layers, and hence avoids overfitting. It also has the effect of reducing the training time. We also use dropout at each hidden layer to avoid overfitting by ignoring a set amount of neurons output at each layer. Hence, deep learning can be very expensive and requires massive dataset to train itself on. The architecture of the model can be seen in figure 1.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	22528
batch_normalization_1 (Batch	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
batch_normalization_2 (Batch	(None, 256)	1024
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
batch_normalization_3 (Batch	(None, 128)	512
dropout_3(Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 1)	129

Total params: 190,465
Trainable params: 188,673
Non-trainable params: 1,792

Figure 1: Architecture of Deep Learning Model

3 Experiment Setup

3.1 Dataset Description

We are provided with training and test datasets consisting of 4.45million and 1.93million records respectively, each having 28 features. The data we got was from the Kaggle Competition PUBG Finish Placement Prediction (1). The attributes of the dataset is as follows:

ATTRIBUTES	DESCRIPTION	DATATYPE
Id	Player's ID	Category
groupId	ID to identify group in match	Category
DBNOs	Enemy Players Knocked	uint8
Assists	Enemy Player Damaged that were killed By Teammates	uint8
Boosts	Total Boosts item used	uint8
Damage Dealt	Total Damage Dealt	float16
Headshot Kills	Total headshots taken by the player	uint8
heals	No. of healing items used	uint8
kill Place	Ranking in match of number of enemy players killed	uint8
kill Points	Kills-based external ranking of player	uint8
killstreaks	Maximum enemy killed in a short duration	uint8
kills	No. of enemy player killed	uint8
longest Kill	Longest distance between player and player killed	float16
numerous	Total groups in a match	uint8
match Duration	Duration of match in seconds	uint8
matched	The Match ID to trace match	Category
match Type	Types of match like solo, duo etc.	Category
misplaced	The max rank a team/player got in that match	uint8
rank Points	Elo-like ranking of player	uint8
revives	No. of times player revived teammates	uint8

ride Distance	Total distance covered in the form of ride	float16
roadkill's	No. of kills while in a vehicle	uint8
swim Distance	Total distance in form of swimming	float16
teamkills	Total teammates killed in a game	uint8
vehicle Destroys	Total vehicle destroyed	uint8
walk Distance	Total distance walked in a game	float16
weapons Acquired	Total weapons acquired throughout the game	uint8
win Points	Win-based external ranking of player. (Elo-like)	uint8
winPlacePerc	The winning percentage of a player	float16

3.2 Data Insights

So as to discover designs in the information and figure out what highlights might be helpful, We chose to perform Exploratory Data Analysis on the information. This would likewise enable us to make highlights which better portray the aptitude dimension of the players. We begin with the appropriation of winning percentile over all recreations. From Figure 2, we see that our percentile dispersion isn't actually as uniform as we anticipated. There is by all accounts more cases happening at low percentiles. This demonstrates by and large, we have a bigger number of washouts than victors.

The higher tallies at the percentile limits is not out of the ordinary. There is in every case some somebody getting the percentiles scores of zero and 1, whatever remains of the scores fluctuating crosswise over various matches. How about we consider the normal winning percentile per match and take a gander at the dispersion. As referenced before, we ought to anticipate an ordinary dispersion with mean 0.5. In any case, what we find is that the mean conveyance is fairly right-tailed. We can derive that a few diversions have lower normal percentiles. The explanation behind this can be players leaving the amusement before it wraps up. This prompts more individuals and groups having lower positions.

Next, we examine the relationship. From the plot in Figure 4, we can see that a few highlights are exceptionally connected with one another and in this way can either be joined or dropped. We additionally can extricate ascribes connected to our objective variable, win place rate.

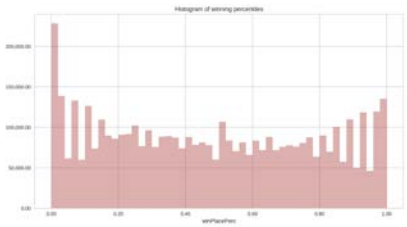


Figure 2: Histogram of Winning Percentiles

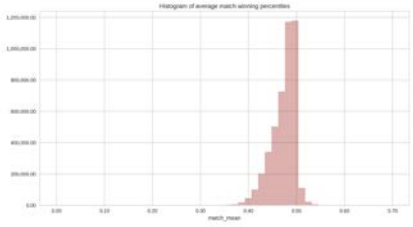


Figure 3: Histogram of Average match Winning Percentiles

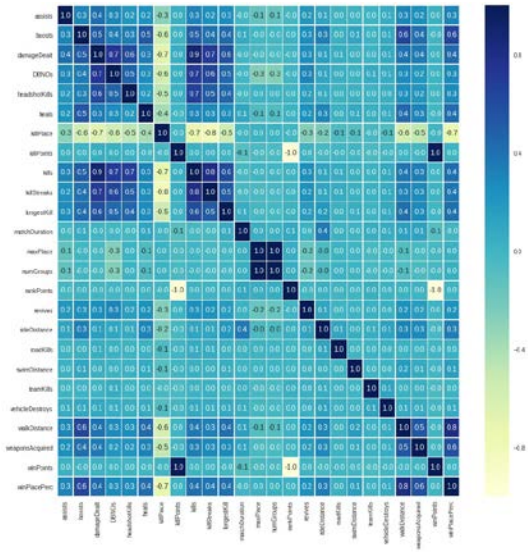


Figure 4: Correlation matrix

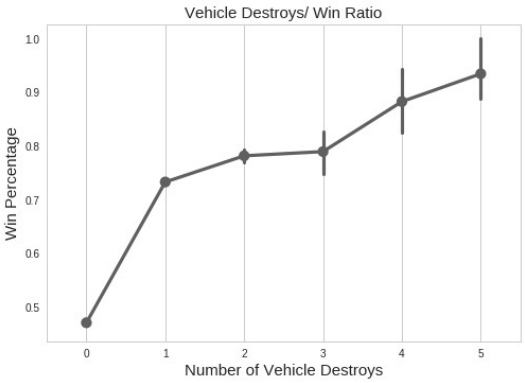


Figure 5: Vehicles Destroyed vs Win Place

A Principal Component Analysis validates this speculation. We likewise observe that traits like No. of vehicles obliterated, which have just about zero connection with the objective variable, really impacts it. We likewise plot the quantity of Kills against the success place expectation to see a general pattern of proportionality which stagnates after a specific dimension of slaughters.

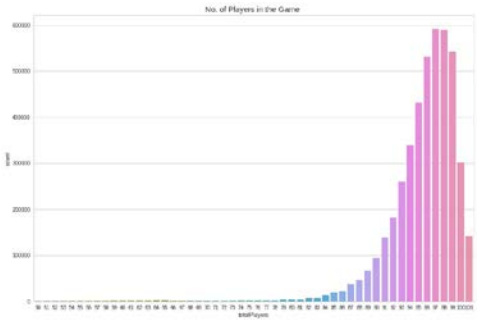


Figure 6: Number of Players

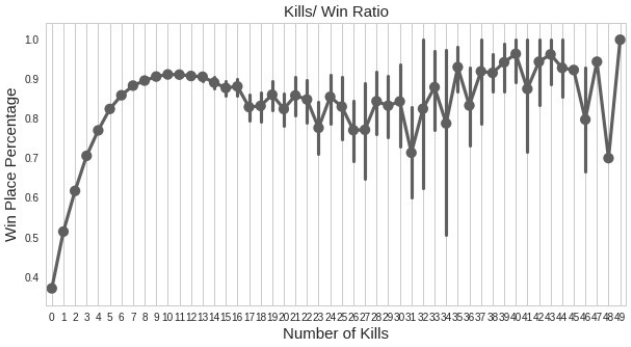


Figure 7: Kills vs Win Place

Next, we view the all-out number of players in an amusement. From Figure 6, We see that the quantity of players shifts in each diversion. At the point when there are 100 players in the amusement it may be simpler to discover and execute somebody, than when there are 70 players. Consequently, we have to standardize certain traits which connote a player's aptitude level with the goal that we can look at them

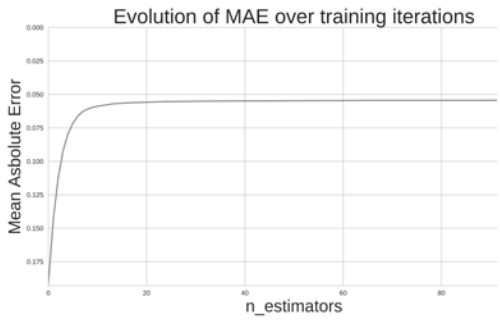
4 Results

For our training dataset we tried fitting onto 11 different models as shown in table 2. The MAE values and R2 values that we got were from the validation data that we applied on each model. We found that boosting models outperformed Regression and Neural Network. Linear Gradient Boosting machine, Table 2: Model Results

Model	MAE	R_2	Parameters
Linear Multiple Regression	0.08755	84.83%	default
Ridge Regression	0.08755	84.83%	default
LASSO Regression	0.12084	74.42%	default
Elastic Net Regression	0.113	77.06%	default
AdaBoost	0.0974	82.79%	learning rate = 0.8
Gradient Boosting	0.05861	93.01%	learning rate = 0.8
Random Forest	0.05751	93.00%	estimators = 0.8
Decision Tree	0.07711	86.92%	default
LGBM	0.05392	93.70%	learning rate = 0.3, estimators = 250, numerates = 200, objective = made
Simple MLP	0.08630	85.42%	learning rate = adaptive, epochs = 23,
Deep Learning	0.0623	90.71%	hidden layers = 4, Optimizer -Adam, learning rate = 0.01, epsilon = 1e-8, decay = 1e-4, epochs = 23, Activation - REL for hidden and sigmoid for output layer

which is an extremely fast and efficient gradient boosting algorithm based on decision trees gave the best result out of the boosting models of 93.01% Mean Absolute Error. The parameters that we used for LGBMRegressor were, [learningrate:0.3, nestimators:250, numleaves:200, objective:'mae', earlystoppingrounds:10] and which are the best parameters we found by applying Grid Search Cross

Validation with 3 folds and parameter grid: gridParams = ['learning_rate': [0.05,0.1,0.3,0.002], 'n_estimators': [50,250], 'num_leaves': [6,10,16,200], 'boosting_type' : ['gbdt','dart','goss','rf'], 'objective' : ['mae'],] The training took 624 seconds with the Mean Absolute Error = 0.05437 and R2 Score=93.22%. The MAE vs epoch graph and the feature scores is shown below in figure 9 and figure 10 respectively.



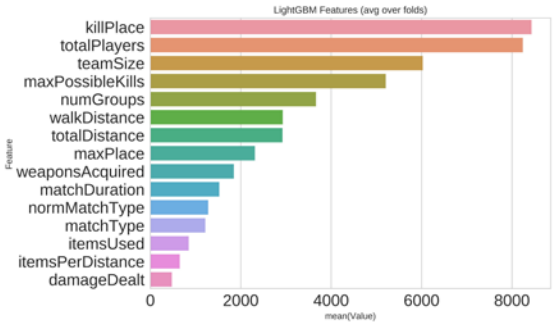


Figure 9: MAE vs Training Iterations Figure 10: LGBM Top 15 Feature Scores

The MAE Score without our feature engineering was 0.0607 proving the importance of our new features which is further attested by the feature scores of LGBM. Out of the top 15 important features determined by LGBM, 7 of them are the new features we added. We then ran our final model on the test dataset and uploaded the predictions to the kaggle competition and the final Test MAE returned was 0.0539.

5 Conclusion

For this dataset, we have seen that Boosting models like LGBM, AdaBoost and Random Forest perform superior to anything conventional Multiple Regression models and Neural Networks. This is maybe in light of the fact that boosting calculations are strong to commotion and exceptions. They enhance the model by means of inclination plunge utilizing nonexclusive differentiable misfortune capacities and furthermore has different advantages like programmed invalid taking care of, in-constructed include determination and scale invariance. Since our information isn't really straightly distinguishable, numerous direct relapse models performed more terrible. There are numerous highlights with non-direct connections between them, which boosting calculations can perceive better utilizing the outfit of various frail students and tuning the parameters to control display intricacy (abstain from overfitting). Likewise, while neural systems are useful for learning complex highlights at a larger amount by doing programmed include designing at each layer which would be useful for information including picture, content or sound where we don't have numerous highlights (only the information itself), our concern has unthinkable information with different highlights effectively given. With unthinkable information, regularly the connection between highlights is shallow and there truly is no compelling reason to investigate complex highlights. Thus, we find that the boosting calculations can learn non-direct connections superior to anything straight relapse and work preferable on unthinkable information over neural systems. Alongside giving better outcomes, they are quicker to prepare when contrasted with neural or profound neural systems. This makes them an unquestionable requirement use approach for relapse issues of this nature. We additionally comprehended the procedure and significance of hyperparameter improvement and accomplishing it through network seek.

We likewise took in the significance of Exploratory Data Analysis and Feature building in Machine Learning. EDA encourages us discover the patterns in the information and comprehend what sort of pre-handling is required. We can likewise utilize these experiences to add new highlights to upgrade the heartiness of our model.

The profound learning model utilized could have performed better on the off chance that it was given more opportunity to prepare. We could likewise have improved it by hyper tuning other important parameters to accomplish better outcomes. Notwithstanding for the best performing model LGBM, a randomized matrix inquiry could have given preferred outcomes over giving network parameters.

We confined ourselves to the kaggle issue explanation of foreseeing the Win place percentile for every one of the players in the dataset. We could have anticipated different measurements like No. of executes, Walk separate, headshots given past information. This expectation should be possible for the entire dataset or at a better scale, for example, per group or per coordinate

Help From: -Microsoft Virtual Academy, The Content Available on the website and from GitHub and pubg competition video present on YouTube