Machine Learning Engineer Nanodegree

Capstone Project

PUBG Finish Placement Prediction

Can you predict the battle royale finish of PUBG Players?

Himanshu Singh
March 07th, 2019

I. Definition

Project Overview

The gaming industry is estimated to be worth $ 100 billion globally and is the largest segment within the entertainment industry; standalone, it exceeds the movies or music segments. The Indian gaming industry is currently valued at over $300 million or approximately Rs. 2000 Crores. With a population of 1.3 billion people and two-thirds of them under the age of 35, India has the world's largest youth population. With this distinct advantage, India has the potential to become one of the world's leading markets for gaming. Gaming is not only one of the fastest growing markets in India, but it is also becoming a career option of choice for a vast number of talented young students from diverse backgrounds. Game Art professionals are in great demand across the globe. The result is that there is huge employment potential for young aspirants within India as it develops as an outsourcing hub for international gaming companies. This is the one reason that motivated me to apply my skills of machine learning to build my career in the gaming industry by knowing how in reality user collected data is used in gaming industries. PlayerUnknown's BattleGrounds (PUBG) has enjoyed massive popularity. With over 50 million copies sold, it's the fifth best selling game of all time and has millions of active monthly players. The team at PUBG has made official game data available for the public to explore and scavenge outside of "The Blue Circle." This competition is not an official or affiliated PUBG site - Kaggle collected data made possible through the PUBG Developer API. The interest in finding best winning strategies is another reason that motivated me to work on this project as udacity capstone project. By knowing winning strategies helps PUBG developers to make winning strategies more complicated, challenging and interesting to attract more and more people to play their game.

Research on operational model of PUBG paper is really helpful in knowing the mathematics behind the converging circle and to know the best time to parachute and also to improve attack hit ratio.

There are some more datasets (2) (3) on PUBG are available, which helps in understanding current datasets features as all datasets are made by same PUB developer API.

Problem Statement

We are by no means experts on the game or military tactics in general for that matter, so, I decided to take an approach using machine learning that is a more familiar to me: trying to explore trends and patterns using provided data by PUBG developer API to know best strategies and decisions to win the game. Let's talk

on problem statement more- We have 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 players per group.
Detailed strategy here to solve the problem-

1. Import important libraries and define optimization function for this large dataset.
2. Loading the dataset and pass through optimization function to reduce memory usage of dataset
3. Explore the dataset for understanding different features and correlation between them.
4. Pre-process the dataset to make it accept by machine learning regressor algorithms.
5. Selected best three regressor algorithm which are suitable for this large dataset to overcome overfitting.
6. Training done with different parameters with K-Fold cross validation.
7. Analysis the score based on given evaluation metric i.e.-mean absolute error as it is a more natural measure of average error, and (unlike RMSE) is unambiguous. Dimensioned evaluations and inter-comparisons of average model-performance error, therefore, should be based on MAE.
8. Founded the relation between the MAE score on testing set and visual plot of predicting on validation set.
9. Feature importance given by different models are averaged to get important feature which will help in making winning strategy.

Our aim is to determine the player's finishing placement based on their final stats, on a scale from 1 (first place) to 0 (last place). A winning player will get finishing placement closes to one and looser will get close to zero. After that, the winning player, who got finishing placement closes to one, will be analysed by looking at the features that are important to him to win the game. By that only we will come to know the best strategy to win the game.

Metrics

Since this is a Kaggle Challenge, we already have an evaluation metric, i.e. "Mean Absolute Error". This will be between ourpredicted winPlacePerc and the observed winPlacePerc.

Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

If the absolute value is not taken (the signs of the errors are not removed), the average error becomes the Mean Bias Error (MBE) and is usually intended to measure average model bias. MBE can convey useful information but should be interpreted cautiously because positive and negative errors will cancel out.

II. Analysis

Data Exploration

Exploratory Visualization

I provided a detailed explanation with visualizations of following features-

1. Current mode (MatchType)
2. Match duration
3. Players in team(added)
4. The killers
5. Heals and boosts
6. Weapons acquired
7. Knocking and reviving
8. The travelers

Current modes

According to PUBG Wiki

Game mode Descrition : https://pubg.gamepedia.com/Game_Modes

Classic mode

- sqaud : Squad is a gameplay type where players are organized into teams which are then pitted against each other. This is distinct from the classic Free For All (FFA) gameplay mode, where players are free to kill whomever they wish.SQUAD, a game mode where you can team up in groups of 2, 3 or 4 players, or if you prefer, you can still play solo and take on everyone alone in the match.Note: No matter the size of your party when you enter a match, you'll be matched with teams of 4.
- sqaud-fpp : Squad FPP(First Person Perspective) is based around your normal Squad game mode (as read above), but instead of being in 3rd person this is strictly in 1st person.
- duo : For this game mode, you will be paired up with another individual and will compete to be the last ones alive.
- duo-fpp : Duo FPP is based around your normal Duo game mode (as read above), but instead of being in 3rd person this is strictly in 1st person.
- solo : Solo, a game mode where you spawn into the world alone and you just rely on your own tactics and skill to push you to the end and be the last player alive.
- solo-fpp : Solo FPP is based around your normal Solo game mode (as read above), but instead of being in 3rd person this is strictly in 1st person.
- Normal matches : I checked with the API's documentation, and it appears that games prepended with "normal" are custom games (such as faceit games, Zombies and War(arcade))

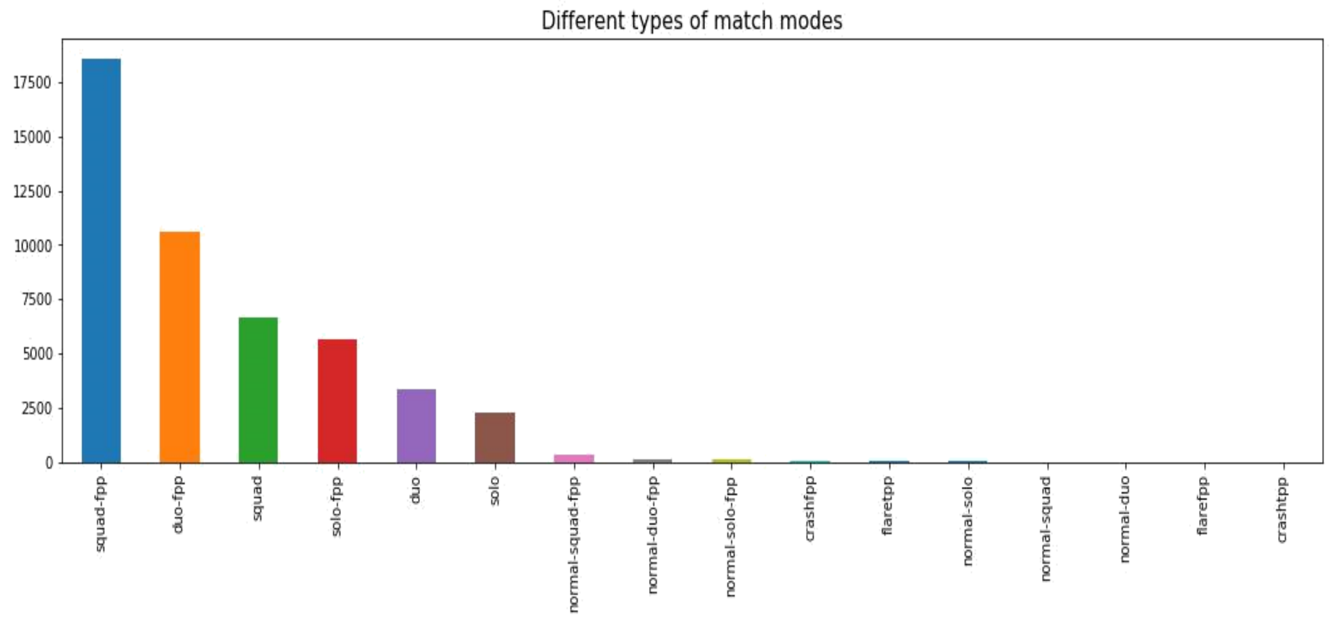
Event mode

Event modes are PUBG’s version of custom games available to all players.

- flare (flare gun) : Flare Gun can be obtained. A flare gun is a kind of signal that shoots toward the sky. It is a special firearm that supplies a special package (Care Package) when the airplane is deployed to that position when in use.
- crash (Crash Carnage) : " Road scary is the subject of this week's event mode, where fuel and fire are king." In Crash carnage, there are no firearms. You have to concentrate on driving skills that can move the Duo to the final round. In this event, the circle moves quite fast, so it quickly crashes on the way to the loot, the vehicle, and the warrior glory of the road. " i.e. It means to win only with a melee weapon without a gun. Goto link 5 &7 at reference.

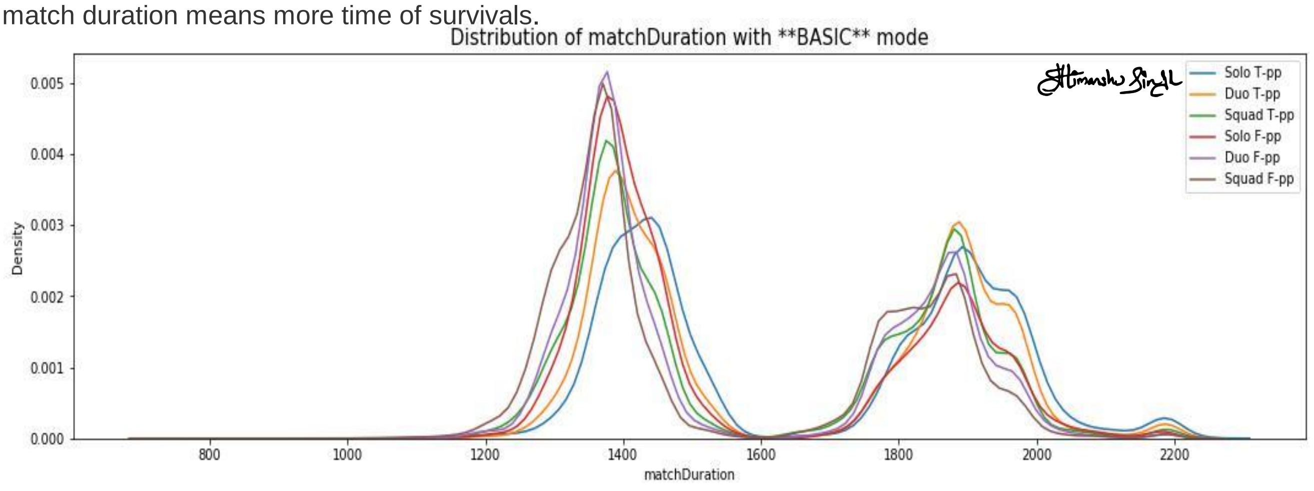
```
squad-fpp         18576
duo-fpp           10620
squad             6658
solo-fpp          5678
duo               3356
solo              2297
normal-squad-fpp  358
normal-duo-fpp    158
normal-solo-fpp   96
crashfpp          73
flaretpp          29
normal-solo       23
normal-squad      16
normal-duo        12
flarefpp          9
crashtpp          5
```

Out of 4446967 players of training data, following are the no. of players playing in different modes-

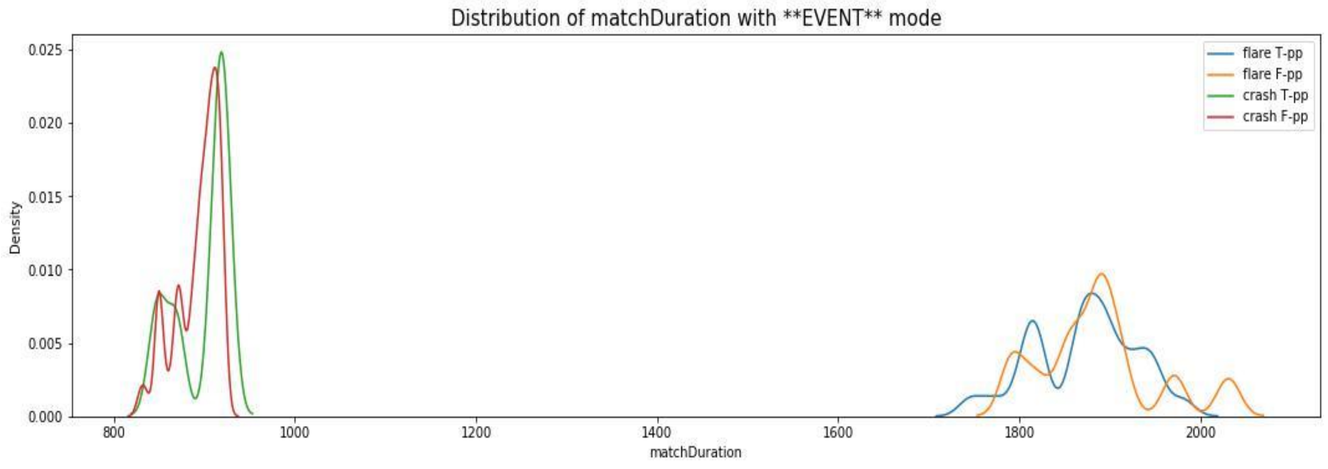


Match Duration

This variable indicates how long the game took. The match duration is relevant to winning the game.. like high match duration means more time of survivals.

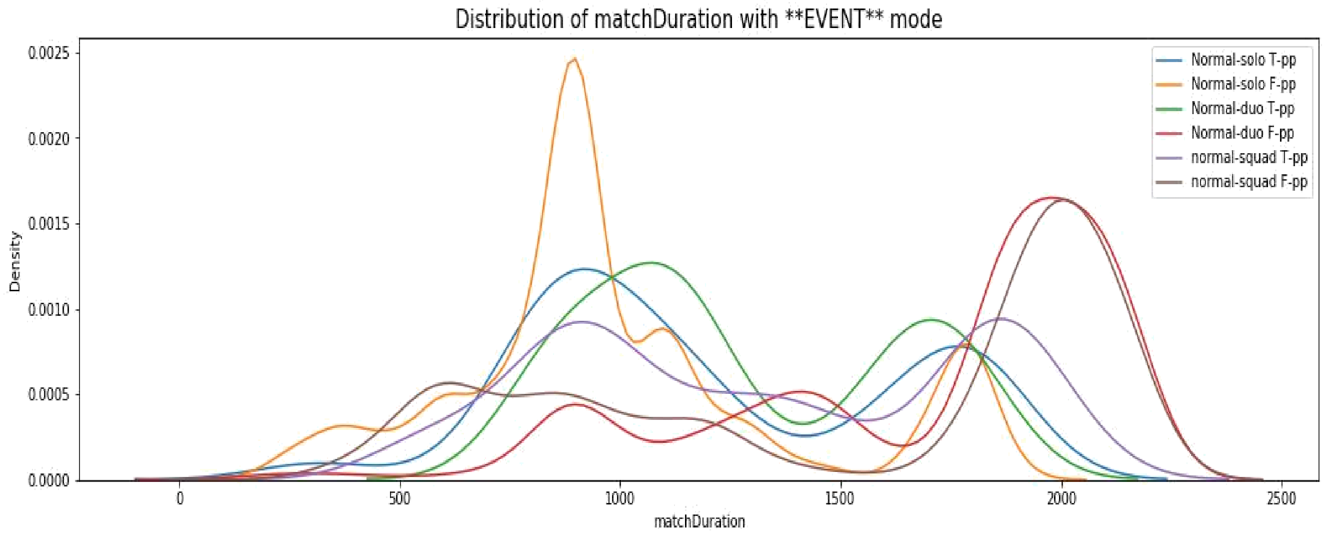


From above figure, it is seen that there are two peeks, the 1st peek is between 1200sec to 1600sec and 2nd peek is between 1600sec to 2200sec.



From above figure, it is seen that crash first and third person matches are played between

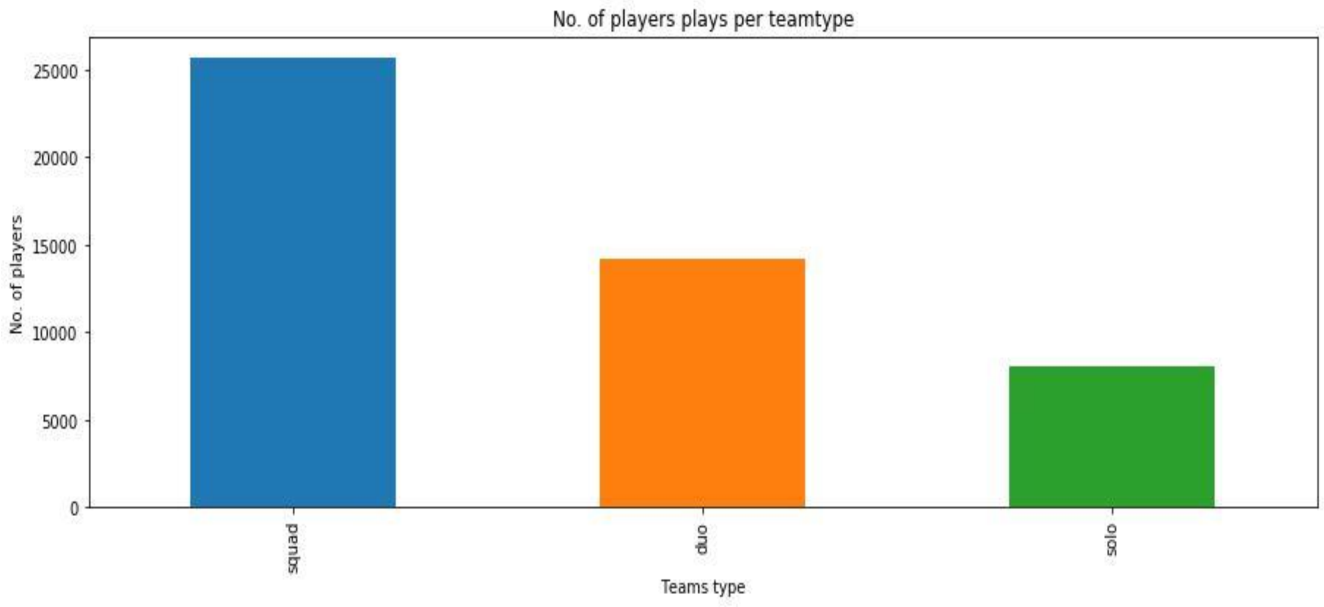
800sec to 950sec and flare first and third person matches are played between 1700sec to >2000sec. Means that flare gun matches are shorter then vehicle with melee weapon matches.



From above figure, it is seen that Arcade mode or zombie mode are played with different time duration. But most players plays in solo FPP, duo FPP and squad FPP.

Players in a team

There are 25646,14224 and 8095 players plays in squad, duo and solo matches respectively.



I seen that group id='289a6836a88d27' has 10 players in a team.

There are 115580 teams has more then 4 players in a team..

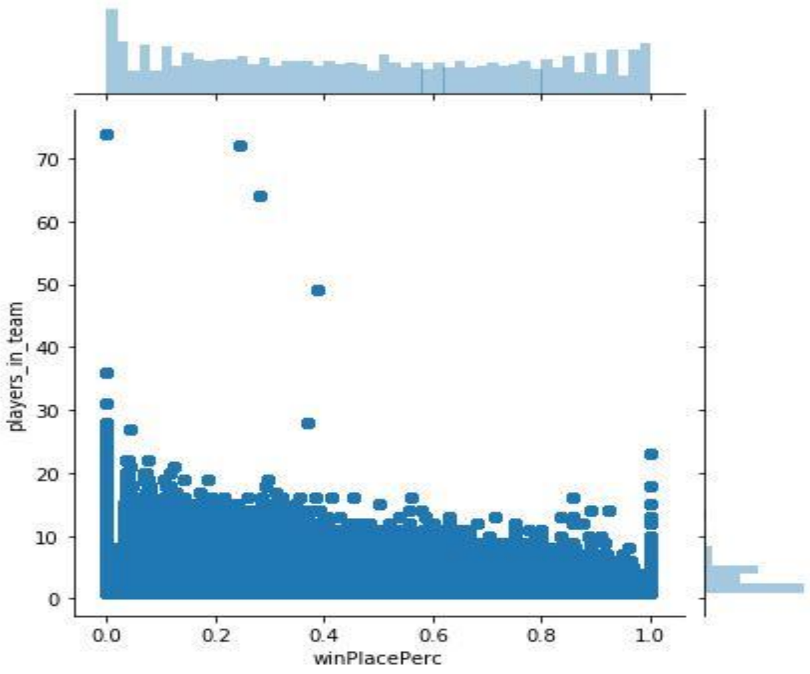
As i research a lot on how this can be possible.. because a max u can play in Sqad which takes 4 players in one team..Then how is this possible??

Then i found out that this can be possible by any of this following ways-

1. The match is from a custom game/event (If there are more than 8 in a group, it is almost definitely this one)
2. The API mistakenly reported two groups as placing in the same place, and in creating groupId these groups were clumped as one larger group. This is also a contributor to the difference between "numGroups" and "maxPlace" (Teams leaving the game may also contribute).
3. There is a very rare bug in the game in which more than 4 people end up in one group.
4. A zombie game can be there..

So, if this will be the situation then can we say it as outliers??
..As by outlier I mean a bot .. by above point 2 and 3 show its an API mistake by putting real players in one groupId.. and point 1 and 2 shows a special type of mode but all these somehow a real players, no bot ..

Correlation of players in team with winPlaceperc



```
Number of players in a team impact on wining the game = -0.29311116498305745
```

As, we can see the impact is negative..So, we need not to worry about number of players in team

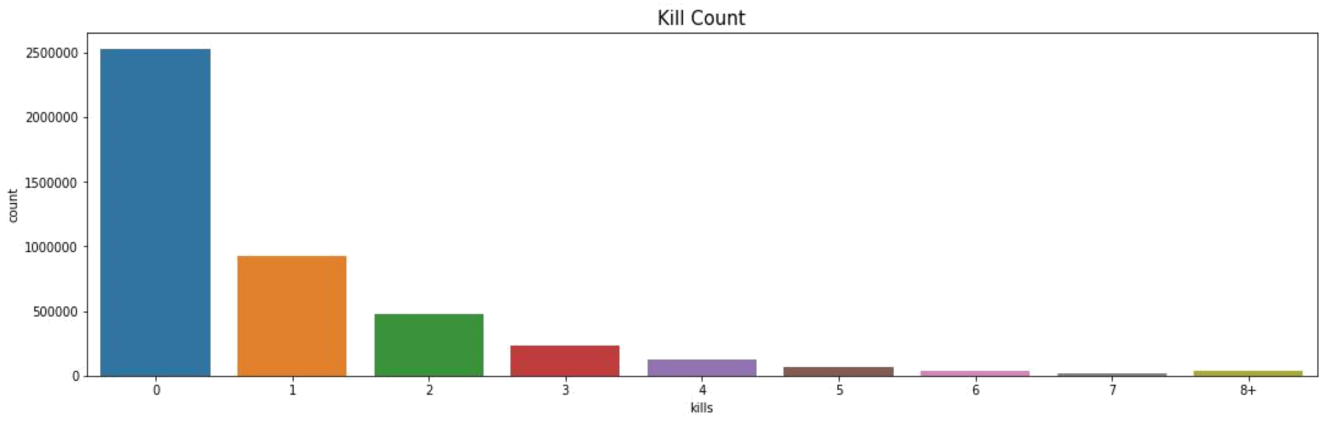
The Killers

Pubg is a shooting game. So, lets starts with killing features then we will see its correlation with target WinPlacePerc

There are Three types of kills here-

1. Headshot kills- killing by aiming on head.
2. Road Kills - killing from vehicles.
3. Longestkills- killing from long distance,generally by snipers.
4. Regular kills- without road kills and headshots.

Lets explore them to find outliers to remove.



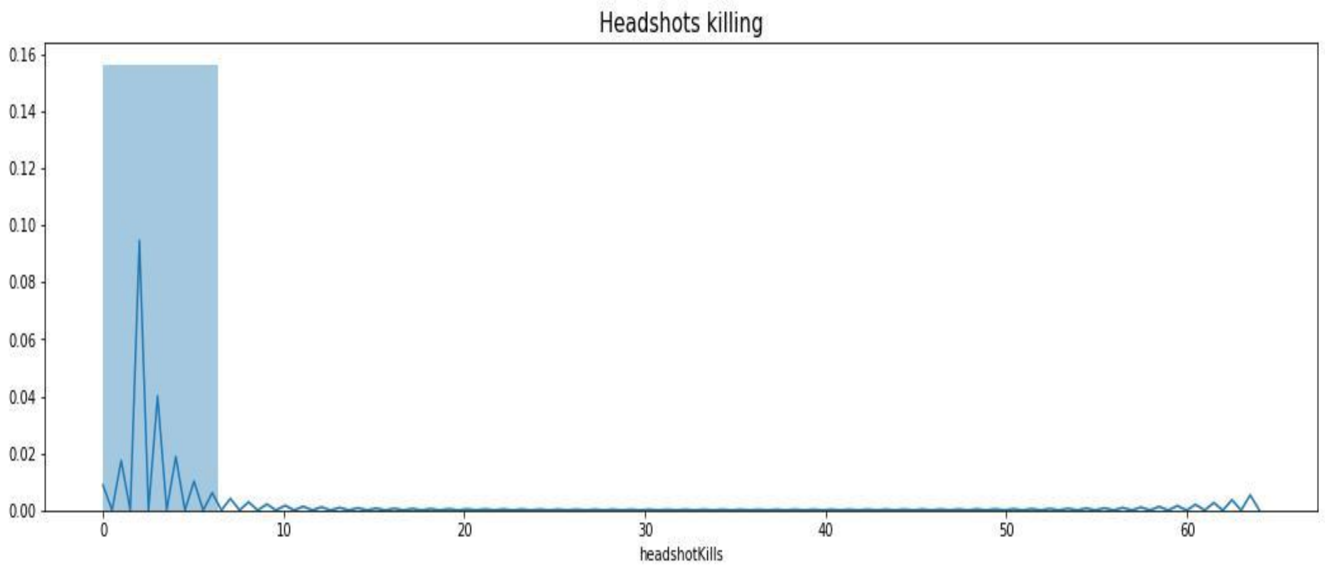
```
The average person kills 0.9248 players
  50% of people had 0.0 kills
  75% of people had 1.0 kills
  99% of people had 7.0 kills
  while the most kills recorded in the training data is 72
```

The above figure6 shows that 50% of people has zero kills while highest kills record is 72.

```
normal-solo-fpp     8
normal-squad-fpp    2
normal-duo-fpp      2
normal-squad        1
```

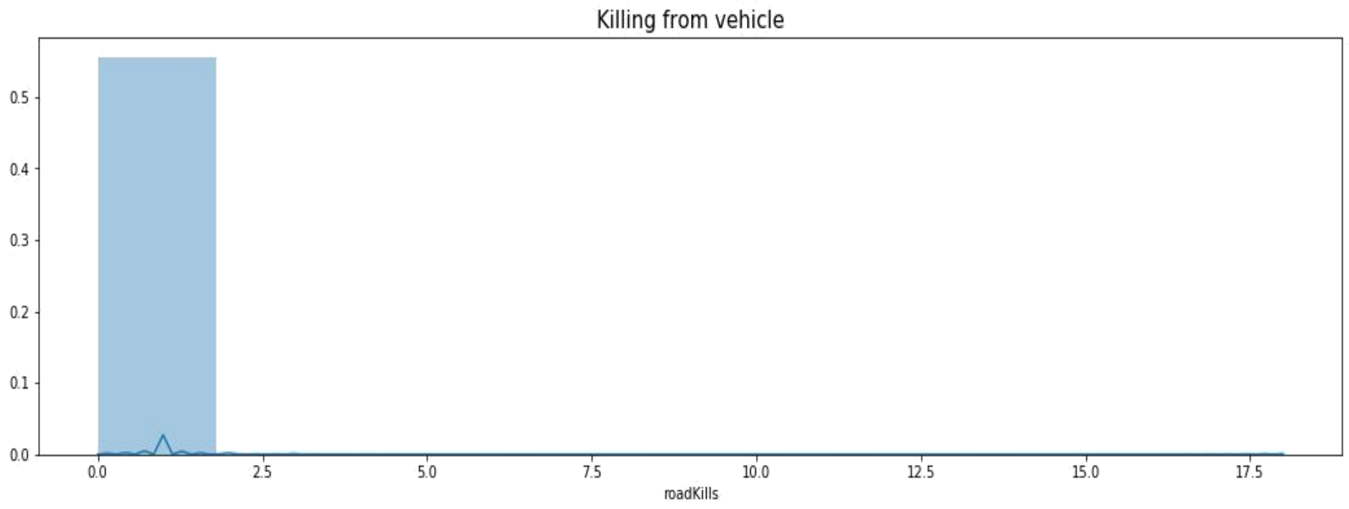
We can see that players in normal solo/duo/squad arcade mode kills more then 50 enemy compare to classic matches .. killing more then 50 may be killing zombies.

Headshots kills



There are 254066 players having headshots equal to total kills. Having headshot kills equal to total kills are not seem tome as outliers..but i can say them as acurate shoot players doing maximum damage to enemy.

Road kills



There are 5516 players having roadkills equal to total kills.

```
correlation of road kills with wining game: 0.03454387477694507
```

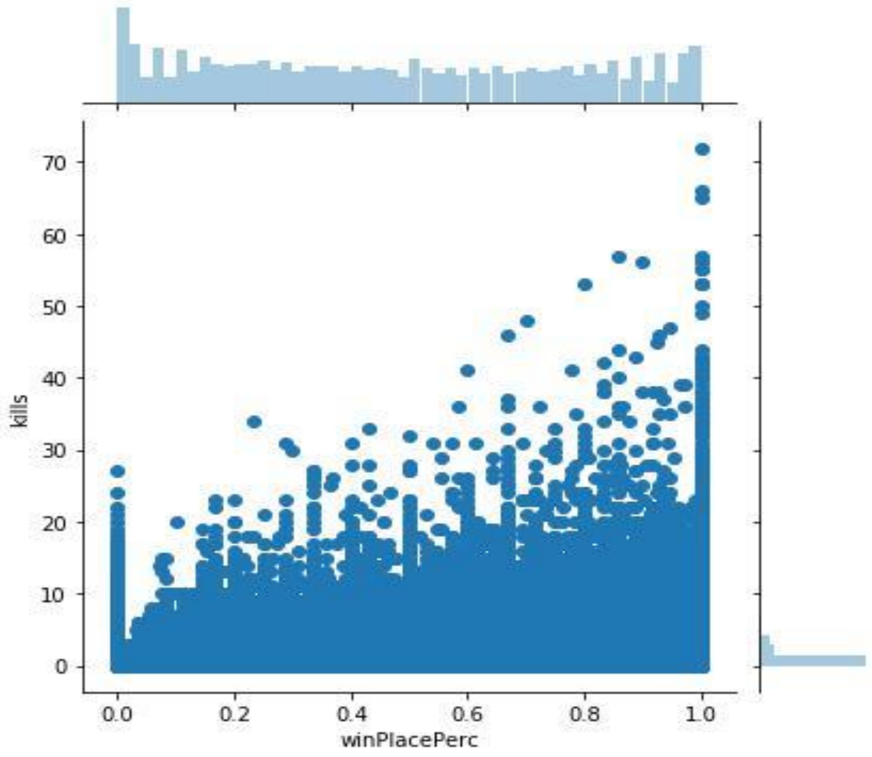
Roadkill doesnot appear to be much effecting winning of game as number of players done roadkills are very less compare to whole dataset due to inaccurate aim from vehicle..

But Driving over opponent is fun .Somethimes in zombies match its fun to kill zombies by vehicles..

Longest kill

The correlation with longest kills with headshot kills is 48%. Which proved that in long range players tries to make headshot as due to high sensitivity of scope its hard to kill enemy with multiple shot but in case of headshot a enemy is down by one shot on head which is more damaging to him.

Correlation between total kills and winPlacePerc-



```
killing impact on wining the game = 0.41995535428253616
```

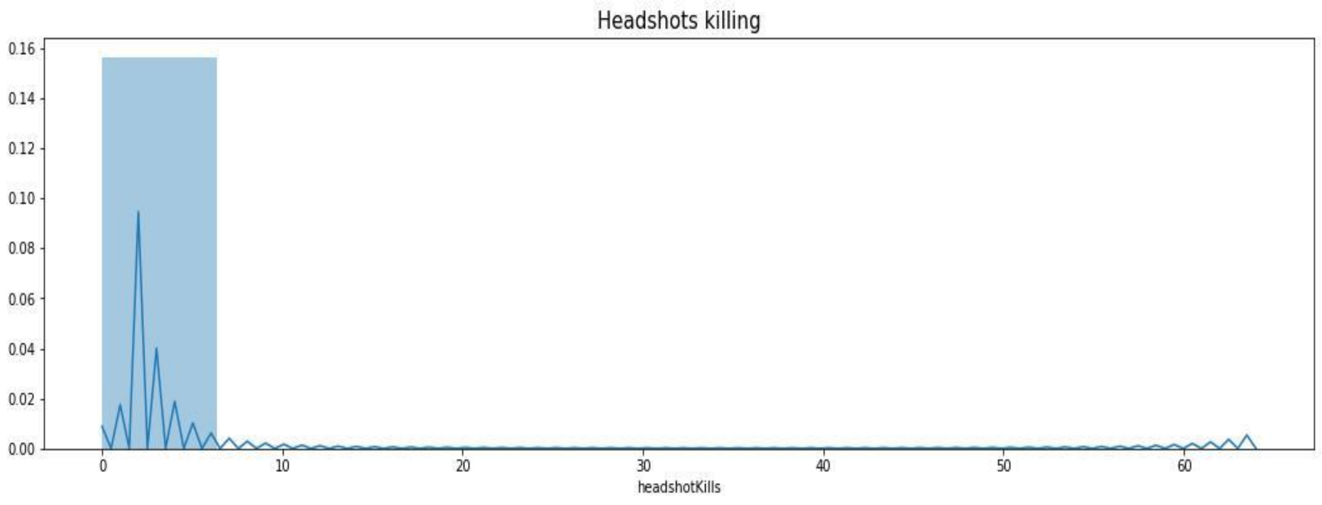
We can see that impact of kills on wining a game is great by 41.99% percentage.

Heals and Boosts

In Playerunknowns battlegrounds, every player has a maximum health of 100%, which can be seen at the bottom of the screen. As the player takes damage, the health bar will decrease and gradually turn red in colour. There are two ways to heal after taking damage/or boost your stamina for walk/run.

You can either use boost items which heal over time or instant heal items like first aid kits

To live longer till top 5 alive, its important apply heals and boost.



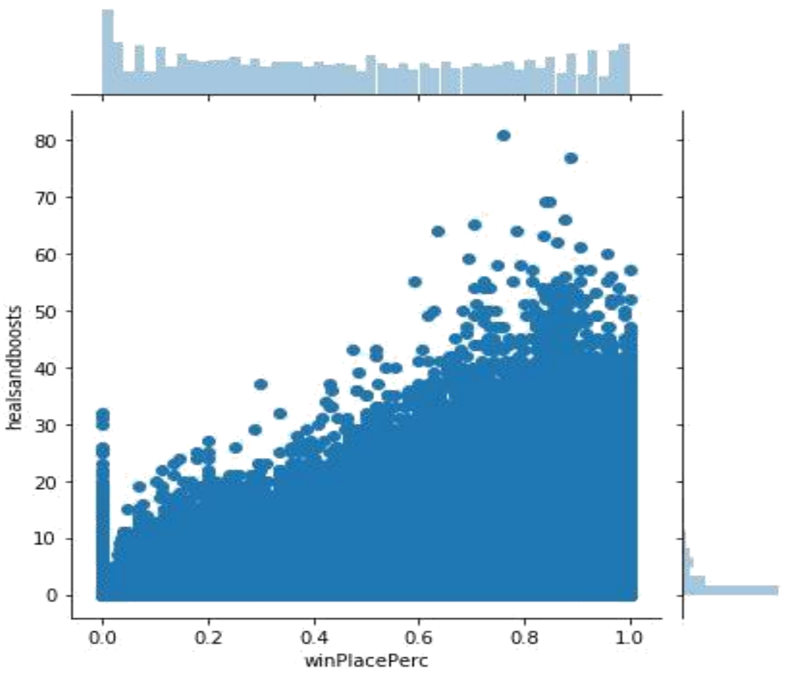
```
On average players used boosts: 1.11 times
99% players used boosts: 7.00 times
Maximum boost applied: 33

On average players used heals: 1.37 times
99% players used heals:12.00 times
Maximum heals applied: 80
```

Can't you think applying that much time of boost and heals(max boost 33 times and heals 80 times) is a player who kills or knocks enimies from point blank range or without any safety elements liks helmet, vest or without hiding.. look at there damageDealt its show how casually he played. I can't treat (id=5b0d0f8b2d5aab and 3be1ded892f443), who uses max boost and heals as Anomaly as except this two things everything looking fine and they contribute well towards winplaceperc.

Although according to me its not a safe to play like this ..

Correlation between health and winplaceperc-



```
The impact of appling heals and boost to win the game: 0.5759995165874834
```

As by correlation, it is proved that to live longer to win the game, its very important to apply Medkit/bandages and boost drinks. Therefore, who survies tills last wins and inorder to survie u must use heals and boost.

Weapons Acquired

According to PUBG wiki ...

There are different types of weapons like Sniper rifles,

- LMGs,
- SMGs,
- Shotguns,
- Bows,
- Pistols,
- Grenades and
- Melee weapons

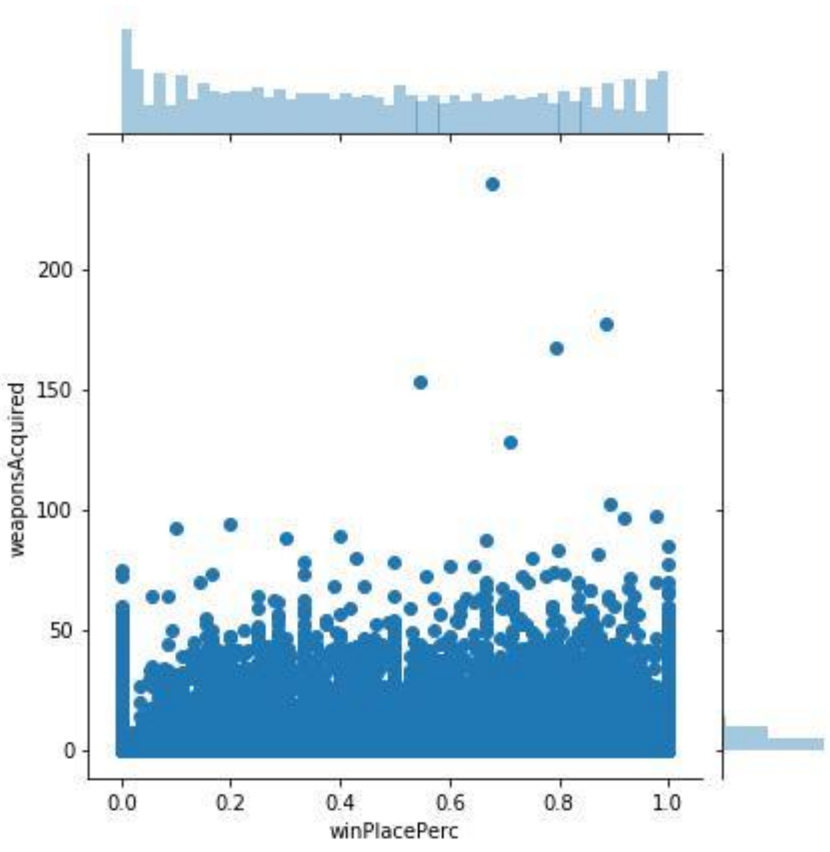
How many weapons can the player hold?

Depending on the players gear (vest 1 - 3 and backpack 1 - 3) you can hold as much as you can carry in your slots. A player can hold 2 primary weapons, 1 pistol, 1 melee weapon and grenades.

```
On average players aquired : 3.66 weapons
Maximum weapon acquired: 236
```

As in average most of players hold 3 weapons..it can be 2 primary weapons and 1pistol.. or if 3 to 20 weapons acquired then it may be 2 rifles ,1 pistol and others 10 will smoke grenades or frag grenades.

Correlation between weapon acquired and winplaceperc-



```
The impact of carrying more weapons to win the game: 0.584576097122193
```

This shows that without weapons you can't win the game..If you carry more weapons like 2 rifles 1 pistol and many smoke grenades to escape,many Stun grenades to blind enemy,many Molotov cocktail to stop enemies and many Frag grenades to kill hidden enemies, will be helpfull in winning the game.

Knocking and reviving

PUBG game play in squad and duo is all about helping teammates(duo and squad) and with the help of them you can win the game..in short its tells about teamwork

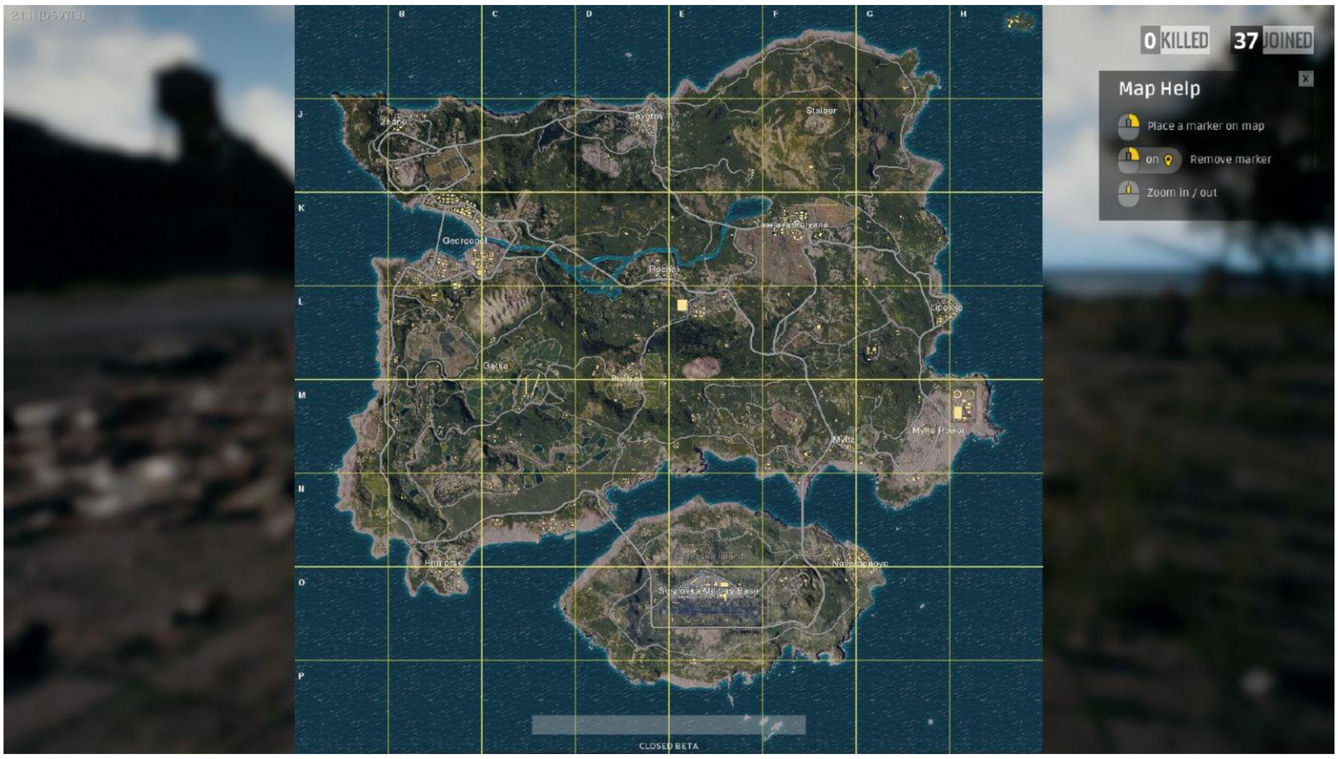
```
How well teamwork corelate to winning the game: 0.3916119278851811
```

Well its only 39% ..its great ..but we have solos also so this cannot be added as column in training set.

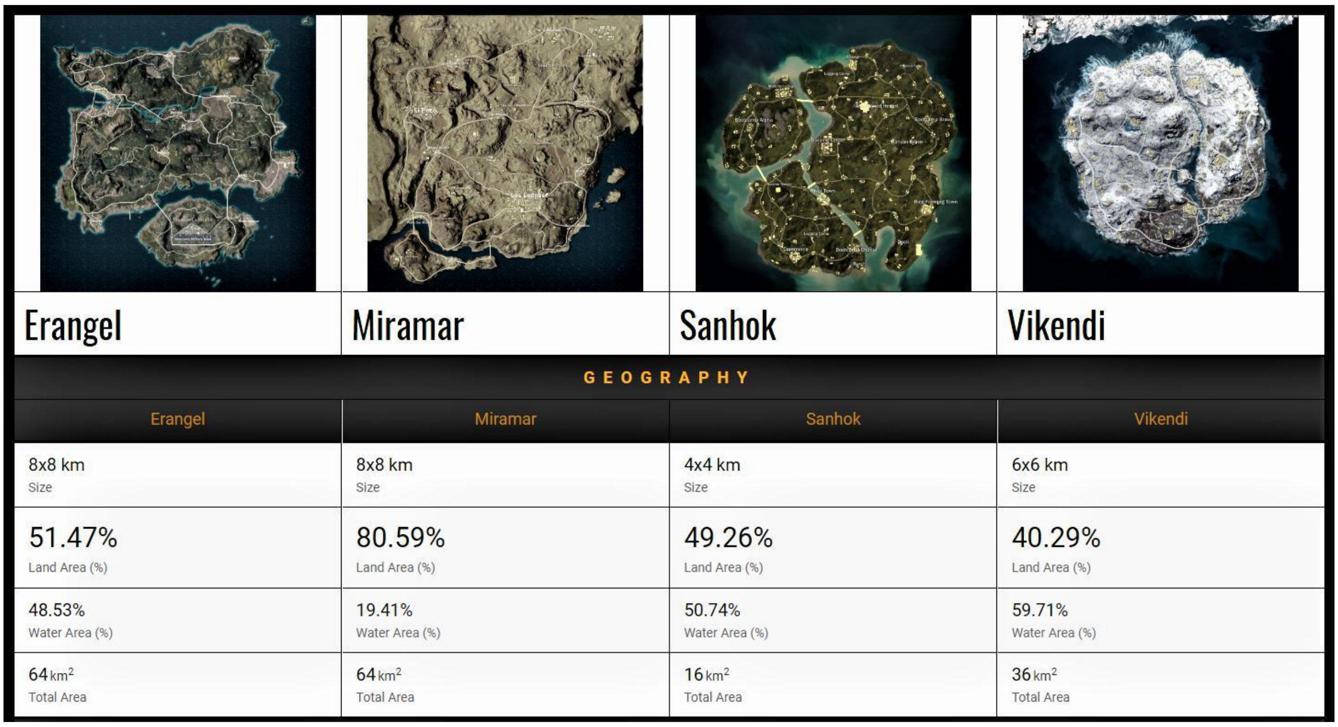
The Travellers

In the Discussion fourm the author says The dataset which we have is from any of the map..

Generally people plays 8x8km map like Erangel and 6x6km vikendi..



The above is erangel map having 8 x 8 km dimensions.. each yellow box is 1 km



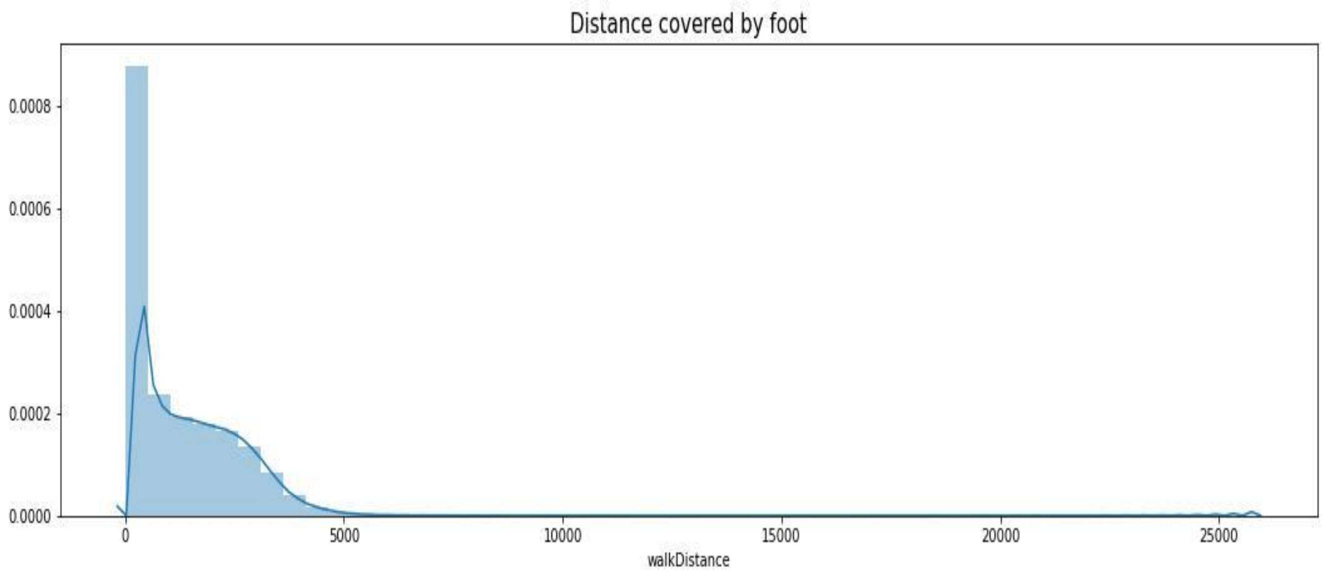
Erangel	Miramar	Sanhok	Vikendi
GEOGRAPHY			
Erangel	Miramar	Sanhok	Vikendi
8x8 km Size	8x8 km Size	4x4 km Size	6x6 km Size
51.47% Land Area (%)	80.59% Land Area (%)	49.26% Land Area (%)	40.29% Land Area (%)
48.53% Water Area (%)	19.41% Water Area (%)	50.74% Water Area (%)	59.71% Water Area (%)
64 km^2 Total Area	64 km^2 Total Area	16 km^2 Total Area	36 km^2 Total Area

Lets get detail on distance travelled-

- Maximum Walking Distance covered : 16KM 250M
- Maximum Swiming Distance covered: 3KM 823M
- Maximum Ride Distance covered: 37KM 670M

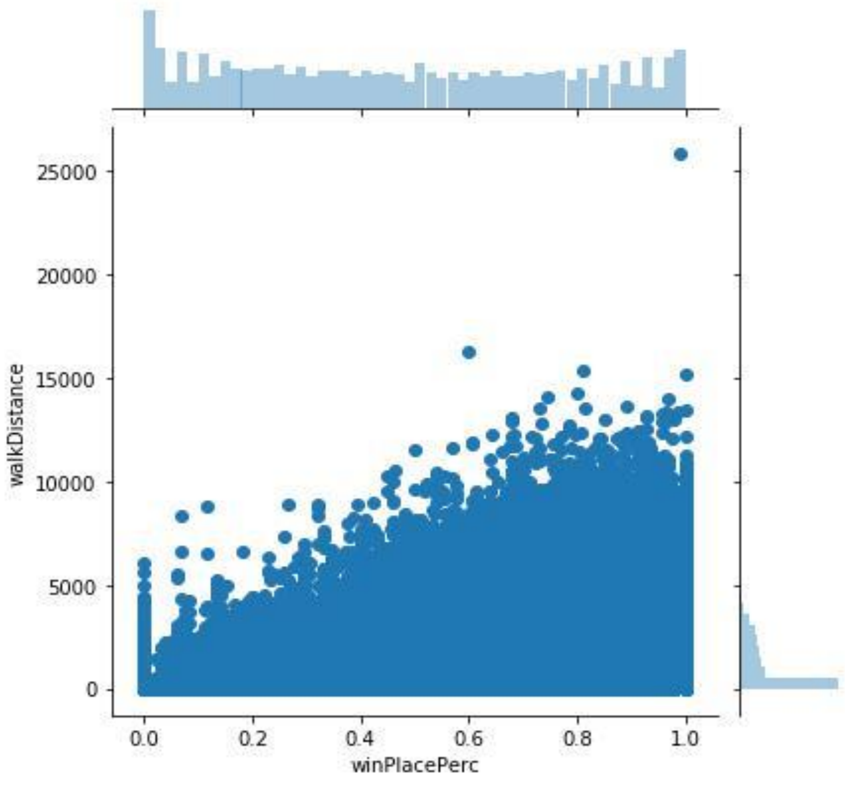
There are certain players who has walk distance=0km , ride distance==0km and swimdistance ==0km but still they have weapons. This can only be happen in arcade mode like war as their players already had weapons. But there are 4 non arcade solo match players who got 1-16 weapons with moving which is not possible. Hence I treated them as outlier and deleted them.

Travelling by foot



This shows that Players travels maximum till 5km. While, most travel less then 2km may be they do camping at one place and not move much till last.

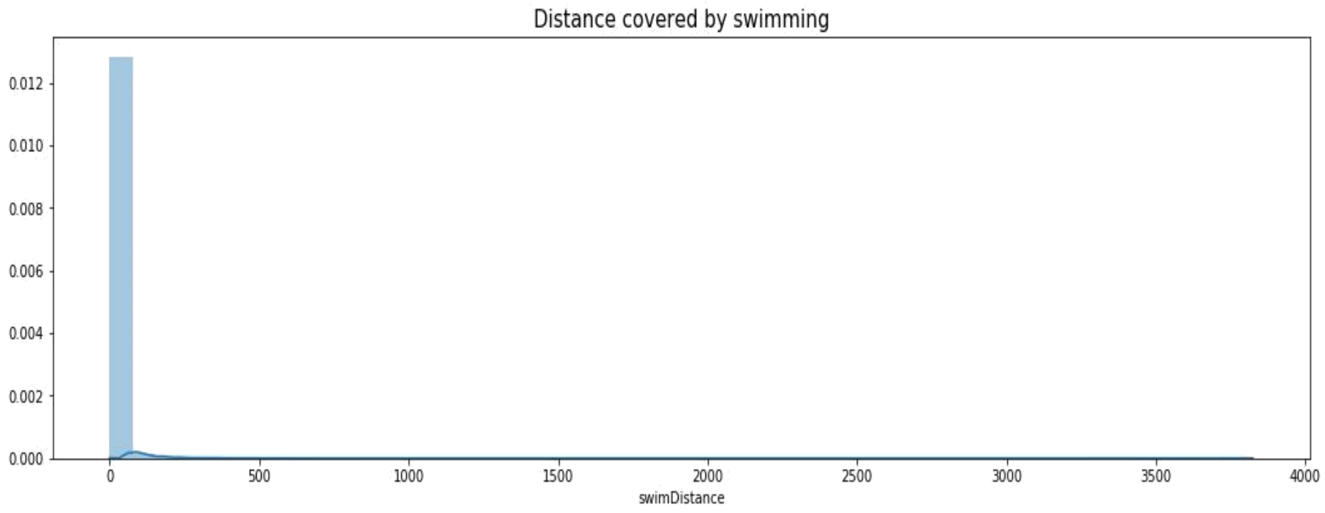
Correlation between traveling by foot and winplaceperc-



```
association between walkingdistance and winning of game: 0.811122123600819
```

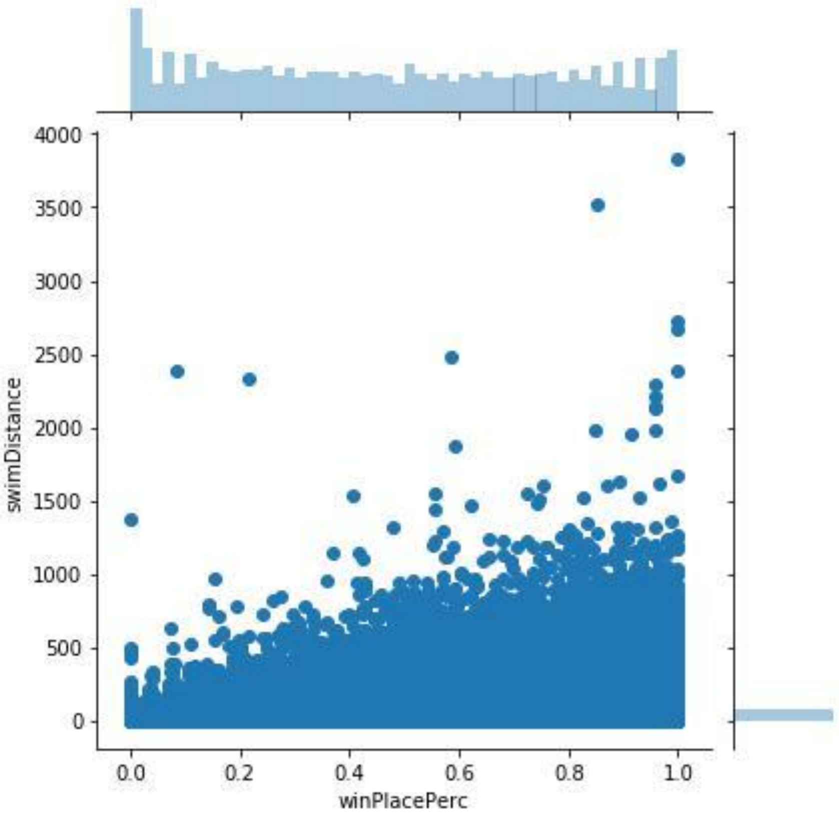
This is really fantastic to see how walking distance has influence on winning the game..

Travelling by swimming



This shows that players does not much interested in swimming except zombie match. In zombie match players like me go for swimming as zombies didn't know swimming so by applying this strategy I alive till last. Hence, wins most of time in zombie match.

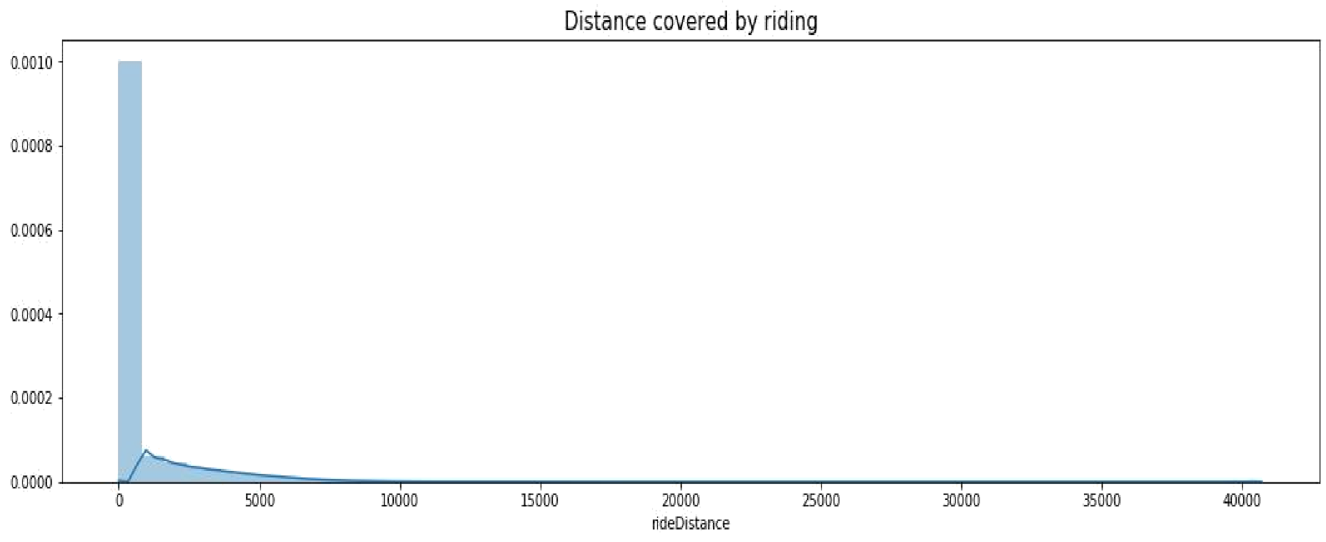
Correlation between swimming and winplaceperc-



```
Association between swimming and winning of game: 0.14963058599988016
```

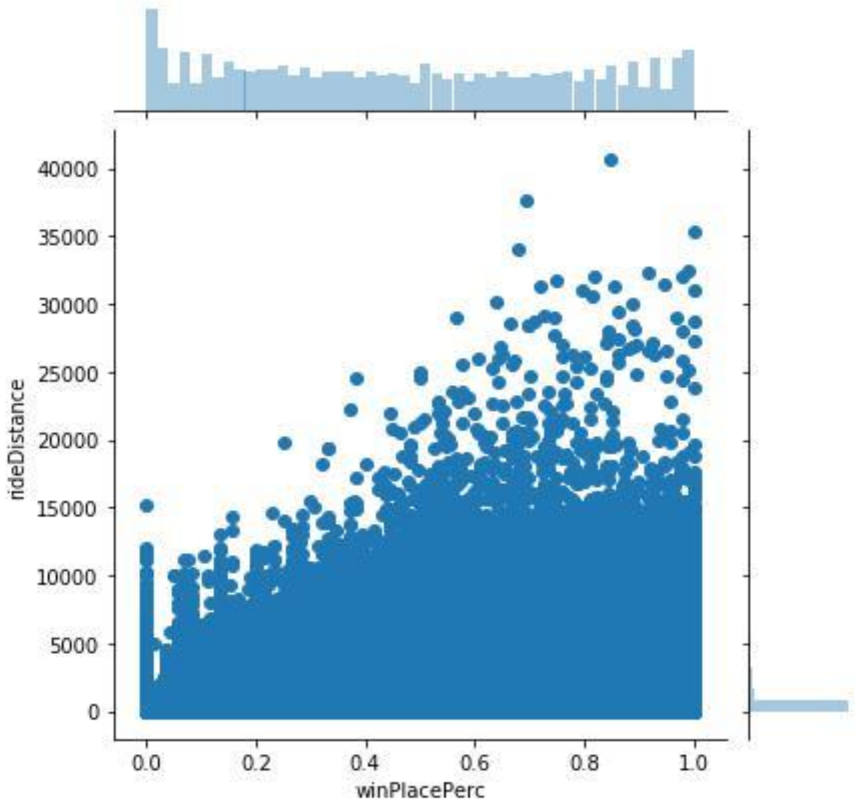
This is low for two reason- (1) didn't have much data on zombie match and (2) mostly last circle does not come on water so player interested to be on land.

Travelling by vehicle/boat



For reaching to drop which is far away from you, its necessary to reach drop location as soon as possible before anyone. So to reach their fast, its require vehicle. Also to safe from dying outside of converging blue circle. This shows that Riding long distances is not much acceptable in wining the game except crash carnage matches.

Correlation between riding and winplaceperc-



```
Association between riding and winning of game: 0.3429627134433459
```

The correlation is little lower due to not having much data on crash carnage match where you only need to ride to reach finals.

Algorithms and Techniques

Starting from data exploration, I used several techniques like covariance to know how much two random feature vary together and correlation to know important features. Along with them I used ploting library like matplotlib and seaborn for density plot which overcome the drawbacks of histograms. The histogram will work badly if we compare the distributions of one variable across multiple categories, histograms have issues with readability. Hence using seaborn density plot gives smooth estimate of the distribution using a kernel density estimation. I used density plot for various features like matchduration, headshot kills, roadkills, heals and boosts, walkdistance, swimdistance and ridedistance to density of people falling under their respective categories.

I also used jointplot for plotting covarince of various features like players in team, kills, heals and boosts, weapon acquired, walkdistance, swimdistance and ridedistance with target feature winplacperc.

Coming to training a model as this project is based on regression. So, I perform regression-based algorithms on this project.

like-

- Random forest(benchmark)
- Light GBM
- AdaBoost

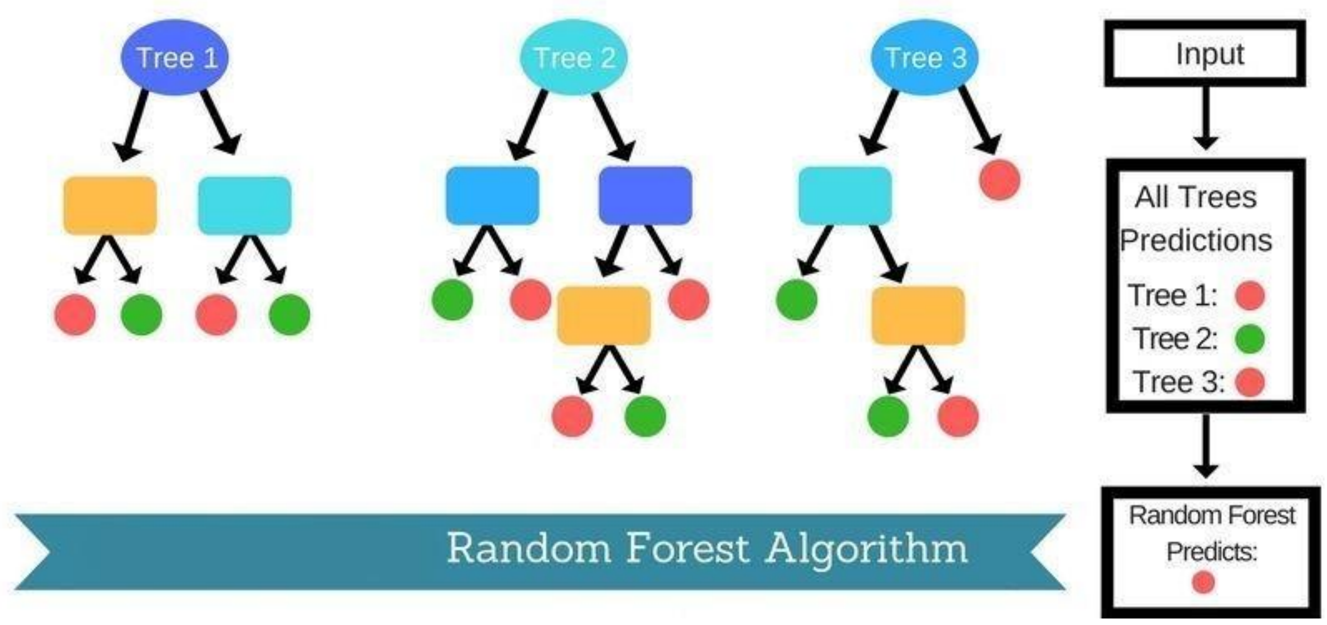
Random forest

Random forest is a bagging ensemble method type of model which tries to build multiple CART model with different sample and different initial variables.

The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

g(x)=f0(x)+f1(x)+f2(x)+...

where the final model g is the sum of simple base models fi. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called **model ensembling**. In random forests, all the base models are constructed independently using a **different subsample** of the data.



Real World Application-

Random forest can be used for Diabetic Retinopathy Classification Analyses.

Strength -

- It does not overfit on data.
- Random Forest performs well with large datasets.
- Random forest is like bootstrapping algorithm with Decision tree (CART) model for solving both classification and regression tree.

Weakness-

- Due to selecting random features to make result then interpreting all result to final result is hard.
- Difficult to judge complex relationships between features.

Resaons for being a good Candidate-

RandomForest are always a safe bet as they generally have good average accuracy rate in training data and overfit less.

Adaboost

It is an adaptive boosting algorithm used with decision tree as a weak learner to make strong learner.

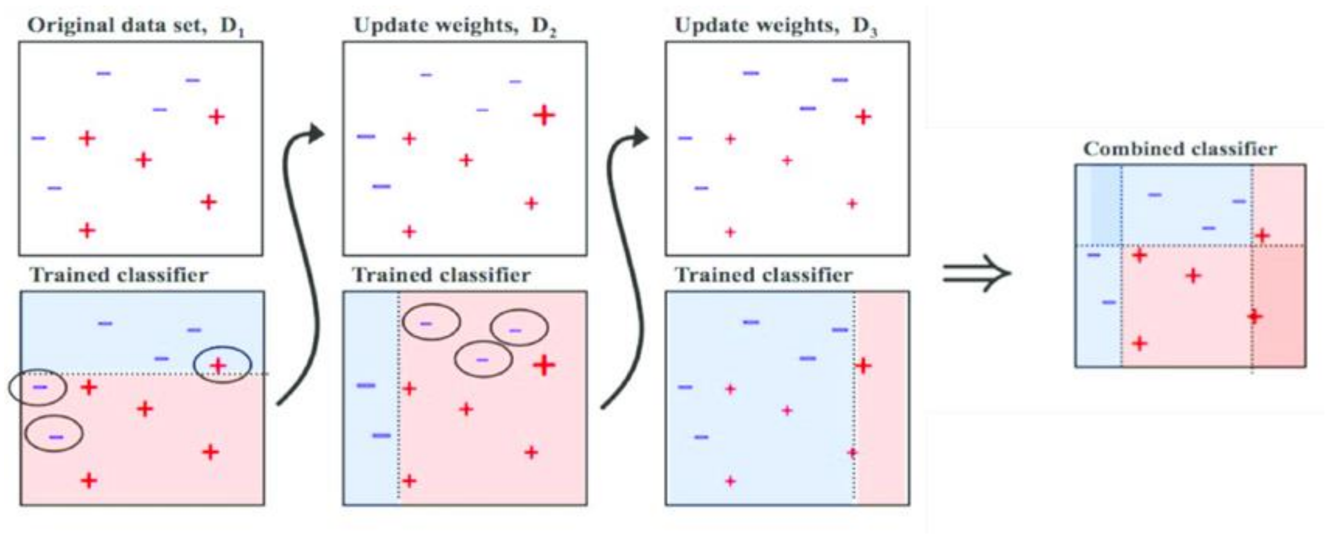
AdaBoost refers to a particular method of training a boosted classifier. A boost classifier is a classifier in the form

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

where each *f{t}* is a weak learner that takes an object *x* as input and returns a value indicating the class of the object. For example, in the two class problem, the sign of the weak learner output identifies the predicted object class and the absolute value gives the confidence in that classification. Similarly, the *Tth* classifier is positive if the sample is in the positive class and negative otherwise. Each weak learner produces an output hypothesis, *h(x_i),* for each sample in the training set. At each iteration *t,* a weak learner is selected and assigned a coefficient *alpha{t}* such that the sum training error *Et* of the resulting *t*-stage boost classifier is minimized.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)]$$

Here *F{t-1}(x)* is the boosted classifier that has been built up to the previous stage of training, *E(F)* is some error function and *f{t}(x)=alpha{t}h(x)* is the weak learner that is being considered for addition to the final classifier.



Real world application-

Used more commonly in Face Detection. As using adaboost,we detect a boundary of face and classify if the image is face or not. In refernce it is mentioned that first real-time face detector was proposed by viola and jones where they used adaboost algorithm and got a impressive result -high detection rate (about 90%) and very low false positive rates.In order to detect a face in an image we need to examine all possible sub-windows and determine whether they contain a face or not.

Strength-

Very little modification of parameter required.

Weakness-

- Sensitive to outliers and noise.
- Slow for large dataset with increase in n_estimators.

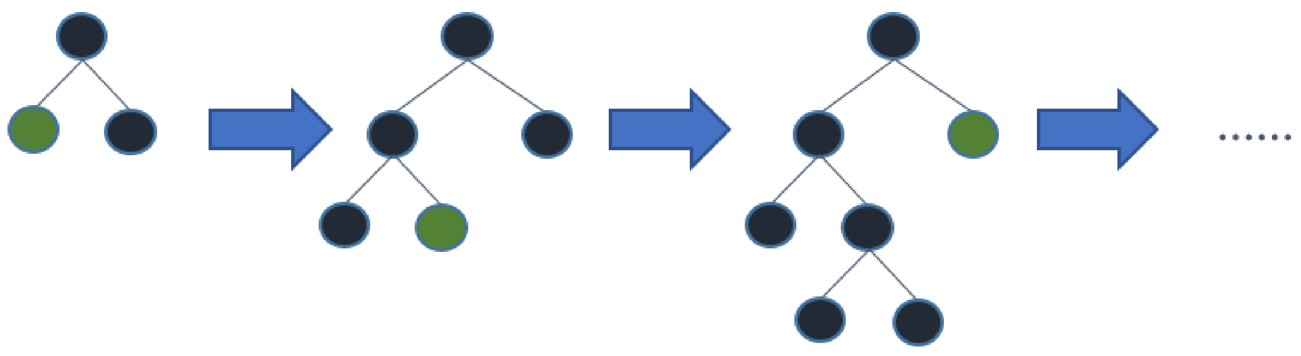
Reasons for being a good candidate-

Simple ensemble method which is quite effective and easy to use.

Light GBM

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.

Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word 'Light'.



Strength

- Faster training speed and higher efficiency: Light GBM use histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
- Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage. Better accuracy than any other boosting algorithm: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
- Compatibility with Large Datasets: It is capable of performing good with large datasets with a significant reduction in training time as compared to XGBOOST and Adaboost.
- Parallel learning supported.

Weakness-

Leaf wise splits lead to increase in complexity and may lead to overfitting. This weakness can be easily overcome by setting parameter like num_leaves exactly half of 2^maxdepth, min_data_in_leaf to between 100 to 1000 for large dataset, max_depth between 3-7 and lowering learning rate with large number of estimators.

Reasons for being a good candidate-

For my project, which has large dataset and high number of 172 features after feature engineering requires less training time then Adaboost and good memory allocation algorithm which xgboost failed to do. So, I used light GBM which respectfully completed my project requirement with great accuracy.

Technique used to know how well our model is performed-

As kaggle specifies that the mean absolute error is used to evaluate this regression project. So, I will going to perform regression model

Benchmark

Let's understand the term *benchmark* without taking into account the context. Benchmark is standard against which you compare the solutions, to get a feel if the solutions are better or worse. Now let's put it in context of machine learning. Benchmarking here means, a standard solution which already performs well. What would be the factors on which our solution will be tested? It's mostly going to be on given the amount of training / test data, what is the accuracy with which our solution is performing, as opposed to the benchmarked solution. What the hosts are seeking is a solution which works better than their existing solution. Of course they are not going to give it away, so that I can analyse and come up with something better. I just have to come up with a good solution and hope it works better than benchmarked solution. This could be achieved by implementing more sophisticated algorithms, or view things more holistically.

Reason for choosing Random forest with Default parameters as a benchmark-

Due to presence of large dataset a CART(classification and regression Tree) model like Random forest perhaps the easiest to train and often one of the most accurate algorithms. In fact, they are regularly used as a **starting point** in kaggle competitions. The **random forest performs implicit feature selection** because it splits nodes on the most **important** variables, but other machine learning models do not. One approach to improve other models is therefore to use the **random forest** feature importances to reduce the number of variables in the problem. This is the reason that random forest will performs well on this large dataset and can form as a competitor for other most advance ensemble methods like adaboost and Lightgbm to beat it, with parameter tunning. Making easiness for us to compare with fast innovations in parameter tunning.The results of this found in Justification section.

III. Methodology

Data Preprocessing

- Imported the dependencies needed for handling data, visualization and training our model. Important dependencies are: Pandas for their data frame structures and easy visualization. Matplotlib for visualization. Scikit-learn for machine learning.
- As shown in data exploration section the training data was explored to understand various types of features.
- While doing data exploration I came up with various outliers which need to removed before training.
- Following are outliers-
-
  - There is a NULL value present in winPlacePerc column at 2744605th row.
  - In longest kill

    There are 21 players who done sniping more then 1km away.. In pubg most zoomed 8x scope is only up to 1km.

    In weapon acquiring

    But there are 9 players acquired more then 60 weapons without moving and 3 players acquired more then 90 weapons in less then 1km walk which is not possible

    There are certain players who has walk distance=0km , ride distance==0km and swimdistance ==0km but still they have weapons. This can only be happen in arcade mode like war as their players already had weapons. But there are 4 non arcade solo match players who got 1-16 weapons with moving which is not possible.

  - Hence I treated them as outlier and deleted them.

- I applied feature engineering to get group mean feature, group max feature, group min feature, group size feature, match mean feature and match size feature on both training and testing dataset.
- Some more features are generated from combining two features like headshotrate=kills/headshotKills, killStreakrate=killStreaks/kills, healthitems= heals+ boosts and then deleted heals features
- Then appended this all features to training and testing dataset respectivelly. By this columns are increased to 172.
- I removed matchType feature due not having equal amount of data for each matchtype.
- For Random forest regressor which does not accept inf and nan, I replaced inf and nan with smallest value 0.01 .
- For Adaboost regressor I used Label encoder to encode labels with value between 0 and n_classes-1.
- Nothing like one hot encoding or label encoding requires for lightgbm as it ca use categorical features directly (without one-hot encoding).
- After this the training data is splited into train data and validation data using KFold cross validation.

Implementation

By inputing the training and testing dataset it is seen that that they take over 1GB and 413MB on RAM. I implemented reduce_mem_usage().

```
def reduce_mem_usage(dataset):
    df=dataset.copy()
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))
    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df
```

Which reduce the memory usage of training and testing dataset by 70.7% and 70.5% respectively.

```
Memory usage of dataframe is 983.90 MB
Memory usage after optimization is: 288.39 MB
Decreased by 70.7%
Memory usage of dataframe is 413.18 MB
Memory usage after optimization is: 121.74 MB
Decreased by 70.5%
```

The next Data exploration, Data cleaning are explained in above sections.

Next, Training classifier with tuning parameters-

- Random forest(benchmark)
- Adaboost
- Light GBM

<u>Random forest</u>

with default parameters is used as Benchmark-

```
from sklearn.ensemble import RandomForestRegressor

regr = RandomForestRegressor(random_state=42,verbose=2)
```

As it was the basic ensemble method which provide good accuracy by minimizing overfitting to some extent on default parameter which can be used as comparison model with advance ensemble methods.

<u>Adaboost-</u>

known for its adaptive boosting algorithm used decision tree as a weak learner to make strong model.

```
Class                 sklearn.ensemble.AdaBoostRegressor(base_estimator=None,
n_estimators=50, learning_rate=1.0, loss='linear', random_state=None)
```

<u>Light GBM</u>

known for giving best accuracy by taking less time to train. Grows tree leaf wise making num_leaves parameter the most important to control overfitting.

```
import lightgbm as lgb

class   lightgbm.Dataset(data,   label=None,   reference=None,   weight=None,
group=None,      init_score=None,      silent=False,      feature_name='auto',
categorical_feature='auto', params=None, free_raw_data=True)
```

The scores are discussed in upcoming sections.

Refinement

Since, I am doing project based on regression and evaluation metric as Mean absolute error. By keeping this in mind, this two parameters should be in Lightgbm.

`objective` : "regression",

`metric` : "mae"

Basic parameters

1. `num_leaves(LightGBM)` - This is the main parameter to control the complexity of the tree model. Theoretically, we can set `num_leaves = 2^(max_depth)` to obtain the same number of leaves as depth-wise tree. However, this simple conversion is not good in practice. The reason is that a leaf-wise tree is typically much deeper than a depth-wise tree for a fixed number of leaves. Unconstrained depth can induce over-fitting. Thus, when trying to tune the `num_leaves` , we should let it be smaller than `2^(max_depth)` . For example, when the `max_depth=7` the depth-wise tree can get good accuracy, but setting `num_leaves` to `127` may cause over-fitting, and setting it to `70` or `80` may get better accuracy than depth-wise. *Tested: [30,31,80]*
2. `min_data_in_leaf(LightGBM)` - This is a very important parameter to prevent over-fitting in a leaf-wise tree. Its optimal value depends on the number of training samples and `num_leaves` . Setting it to a large value can avoid growing too deep a tree, but may cause under-fitting. In practice, setting it to hundreds or thousands is enough for a large dataset. *Tested: [20, 200].*
3. `max_depth(LightGBM)` -You also can use `max_depth` to limit the tree depth explicitly. *Tested:[3, 6].*
4. Tree method used-

   For Light GBM `` `boosting `='gbdt' ``

   they both are traditional Gradient Boosting Decision Tree.
5. Tree learners used-

   For Light GBM, `tree_learner` ='serial', single machine tree learner.

   Control parameters- *Used for speed up training and control overfitting.*
6. For LightGBM, `bagging_fraction` -this will randomly select part of data without resampling. *Tested: [0.6, 0.7, 0.9].*
7. `bagging_freq:k` -frequency of bagging. `k` means perform bagging at every `k` iteration.*Tested: [10, 20].*
8. `early_stopping_rounds` -will stop training if one metric of one validation data doesn't improve in last. *Tested:[100, 200].*
9. For Adaboost and LightGBM , Learning_rate : tested between (0, 5].
10. `n_estimators` - number of weak learners. *Tested: [2000,4000,15000].*

IV. Results

Model Evaluation and Validation

In this section, I will discuss how I derived final solution along with sensitivity analysis.

To do sensitivity analysis for validating the robustness of data. I manipulate the input data by training my models on keeping the outliers present already in input data and also after removing those outliers.

The two boosting algoritms- Adaboost and lightgbm are compared on mean absolute error. This two models are trained in respective files-

Training model keeping outliers as it is-

Adaboost-

```
Adaboost-with-outliers.html
```

with final best parameters as

$$nestimators = 50, learning_rate = 0.05, loss =' linear'$$

By this I got mean absolute error= 0.09993

LightGBM-

```
Lightgbm-with-outliers.html
```

with final best parameters as

$$objective =" regression ", metric =" mae ", nestimators = 15000, earlystoppingrounds = 300,$$

$$numleaves = 31, learningrate = 0.05, baggingfraction = 0.7,$$

$$baggingseed = 0, num_threads = 4, colsamplebytree = 0.7$$

By this I got mean absolute error= 0.02446

Training model by removing outliers-

Adaboost-

```
Adaboost-without-any-outliers.html
```

with final best parameters as

$$nestimators = 55, learning_rate = 0.05, loss =' linear'$$

By this I got mean absolute error= 0.09957

LightGBM-

```
Lightgbm-without-outliers.html
```

with final best parameters as

$$objective = regression, metric = mae, nestimators = 20000, earlystopping_rounds = 200,$$

$$numleaves = 31, learningrate = 0.05, baggingfraction = 0.7,$$

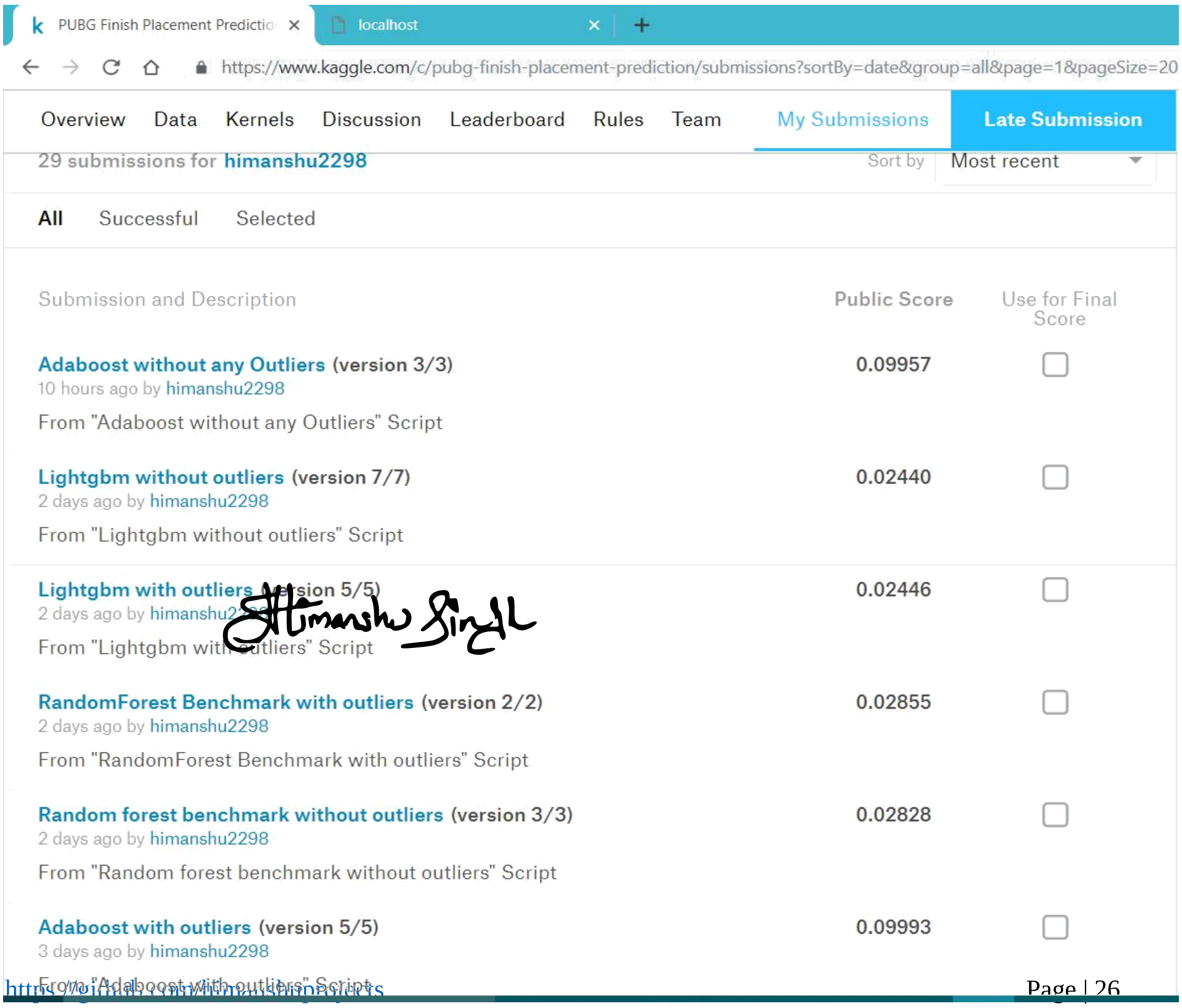
$$baggingseed = 0, numthreads = 4, colsamplebytree = 0.7$$

By this I got mean absolute error= 0.02440

Now, as I showed above models with sensitivity analysis and best fitted parameters. It is clearly observerd that, the adaboost performance is very poor with or without outliers and it also taking almost 7hrs to train.

The final model LightGBM without any outliers is performing better then training LightGBM with outliers as there is 0.0006 decrease in mean absolute error. Sensitivity analysis also proofs that for any model we train , the scores are always better on training model without outliers. So, our intuition on understanding outliers by data exploration is correct. This leads to the derivation of final model- LightGBM without any outliers.

Justification


PUBG Finish Placement Predictio
localhost
https://www.kaggle.com/c/pubg-finish-placement-prediction/submissions?sortBy=date&group=all&page=1&pageSize=20
Overview
Data
Kernels
Discussion
Leaderboard
Rules
Team
My Submissions
Late Submission
Sort by
Most recent
All
Successful
Selected

29 submissions for himanshu2298

Submission and Description	Public Score	Use for Final Score
Adaboost without any Outliers (version 3/3) 10 hours ago by himanshu2298 From "Adaboost without any Outliers" Script	0.09957	☐
Lightgbm without outliers (version 7/7) 2 days ago by himanshu2298 From "Lightgbm without outliers" Script	0.02440	☐
Lightgbm with outliers (version 5/5) 2 days ago by himanshu2298 From "Lightgbm with outliers" Script	0.02446	☐
RandomForest Benchmark with outliers (version 2/2) 2 days ago by himanshu2298 From "RandomForest Benchmark with outliers" Script	0.02855	☐
Random forest benchmark without outliers (version 3/3) 2 days ago by himanshu2298 From "Random forest benchmark without outliers" Script	0.02828	☐
Adaboost with outliers (version 5/5) 3 days ago by himanshu2298 From "Adaboost with outliers" Script	0.09993	☐

Himanshu Singh

The benchmark model I used is Random forest with default parameters. Trained on both in presence of outliers and not presence of outliers.

The training part is present in following files-

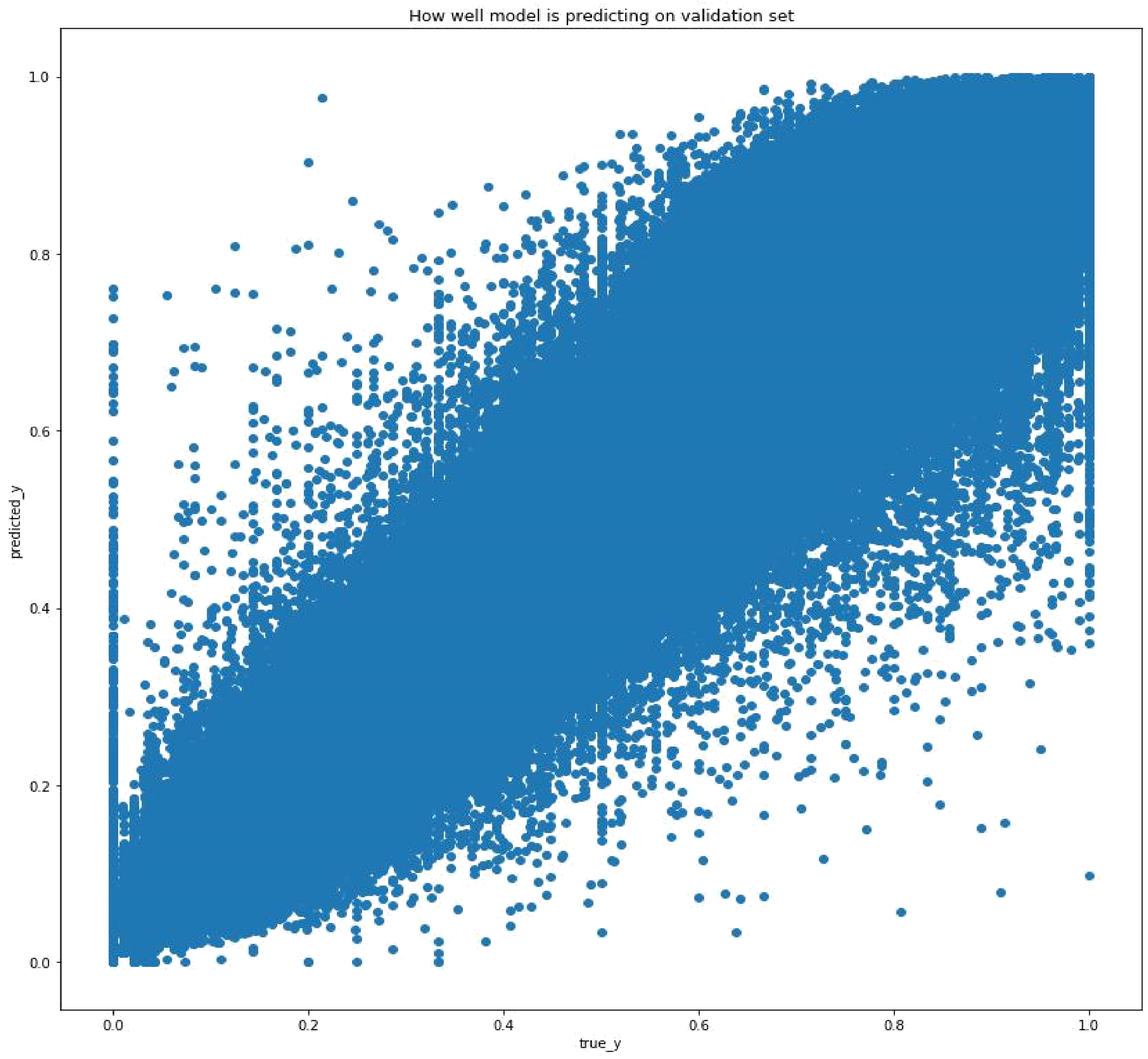
```
Randomforest-benchmark-with-outliers.html
```

```
Random-forest-benchmark-without-outliers.html
```

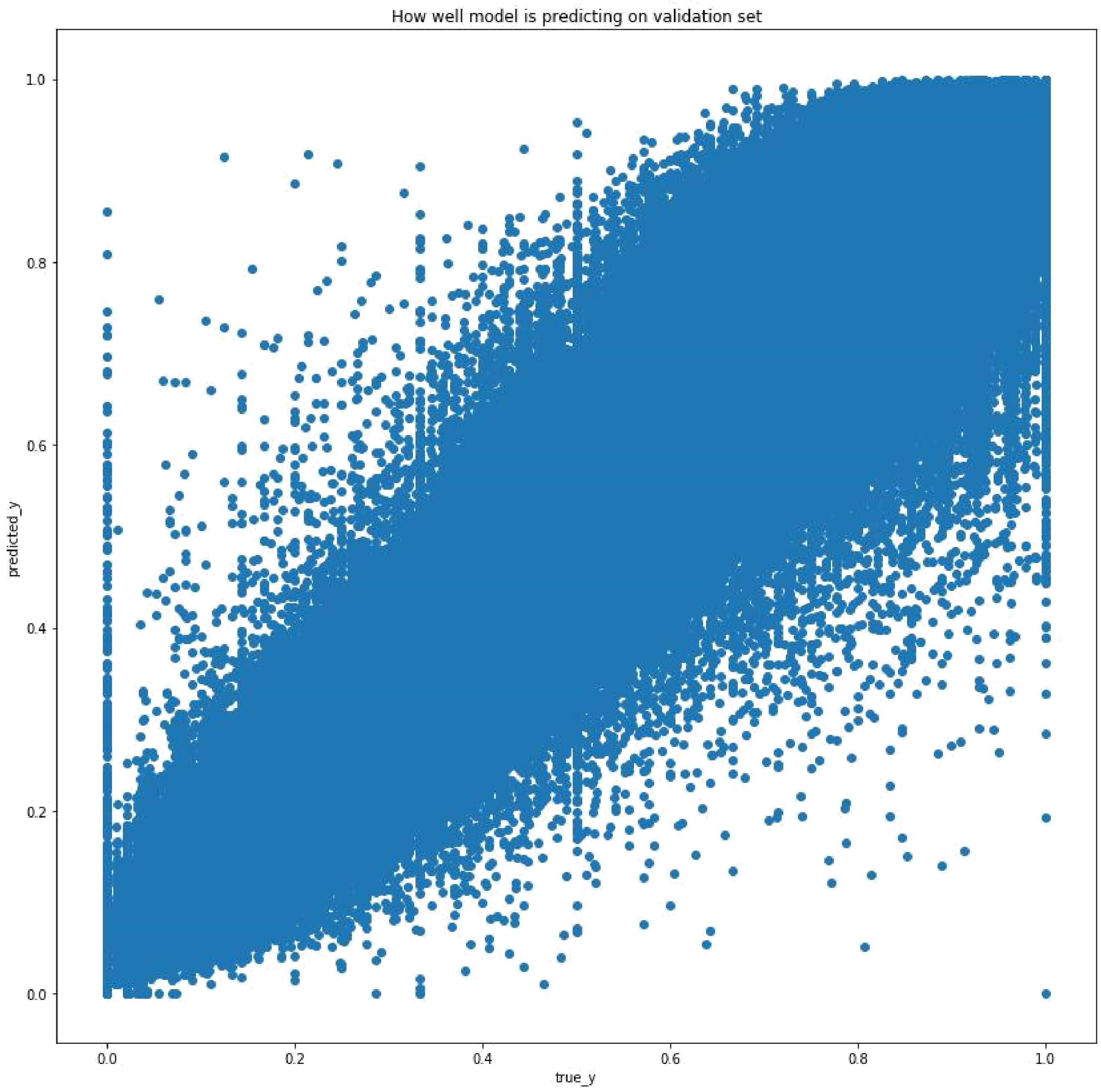
How Final LightGBM is better then benchmark?

Let see this by how well the model is predicting on validation set.

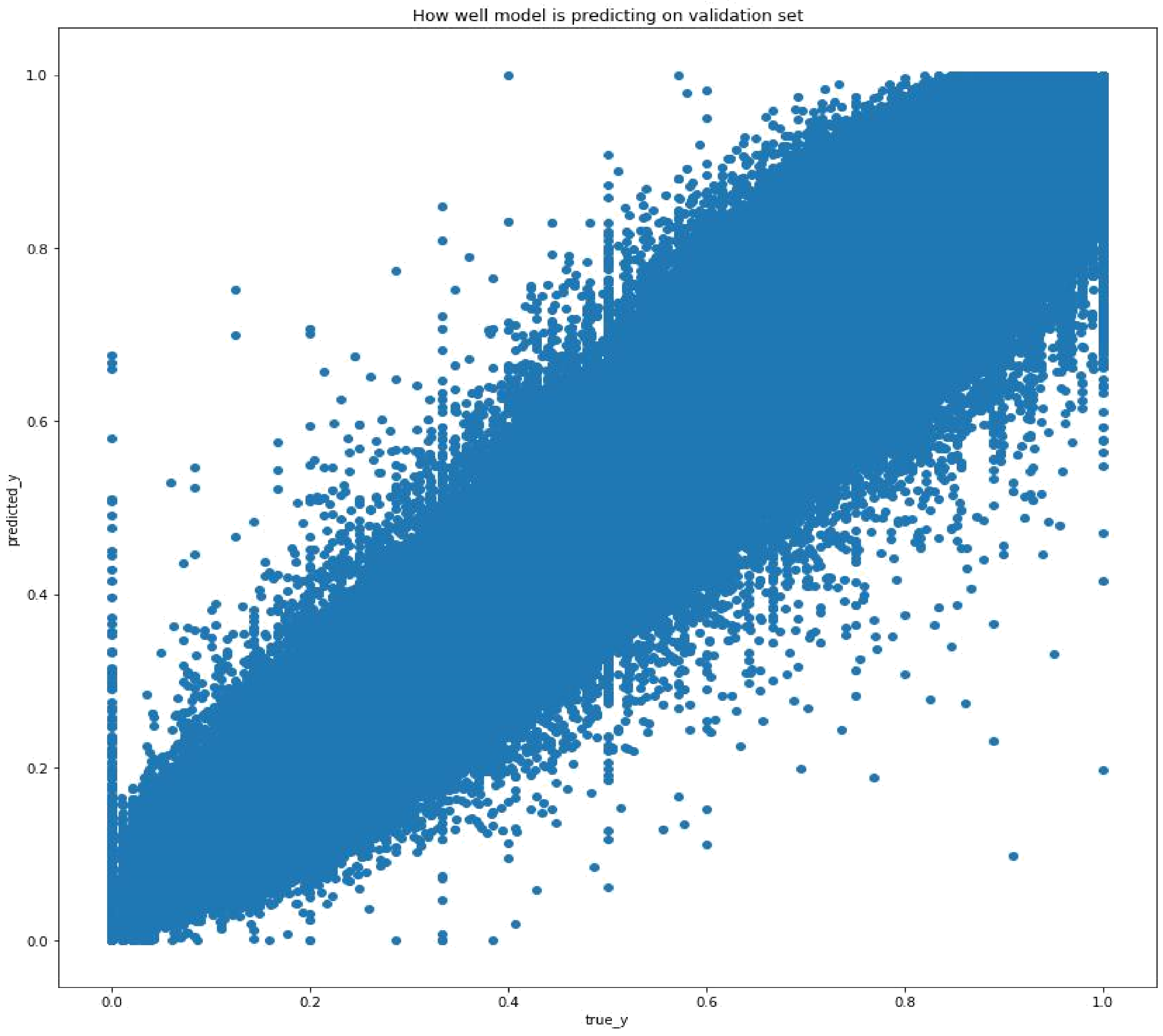
Benchmark Random forest trained in presence of outliers-



Benchmark Random forest trained in presence of NO outliers-

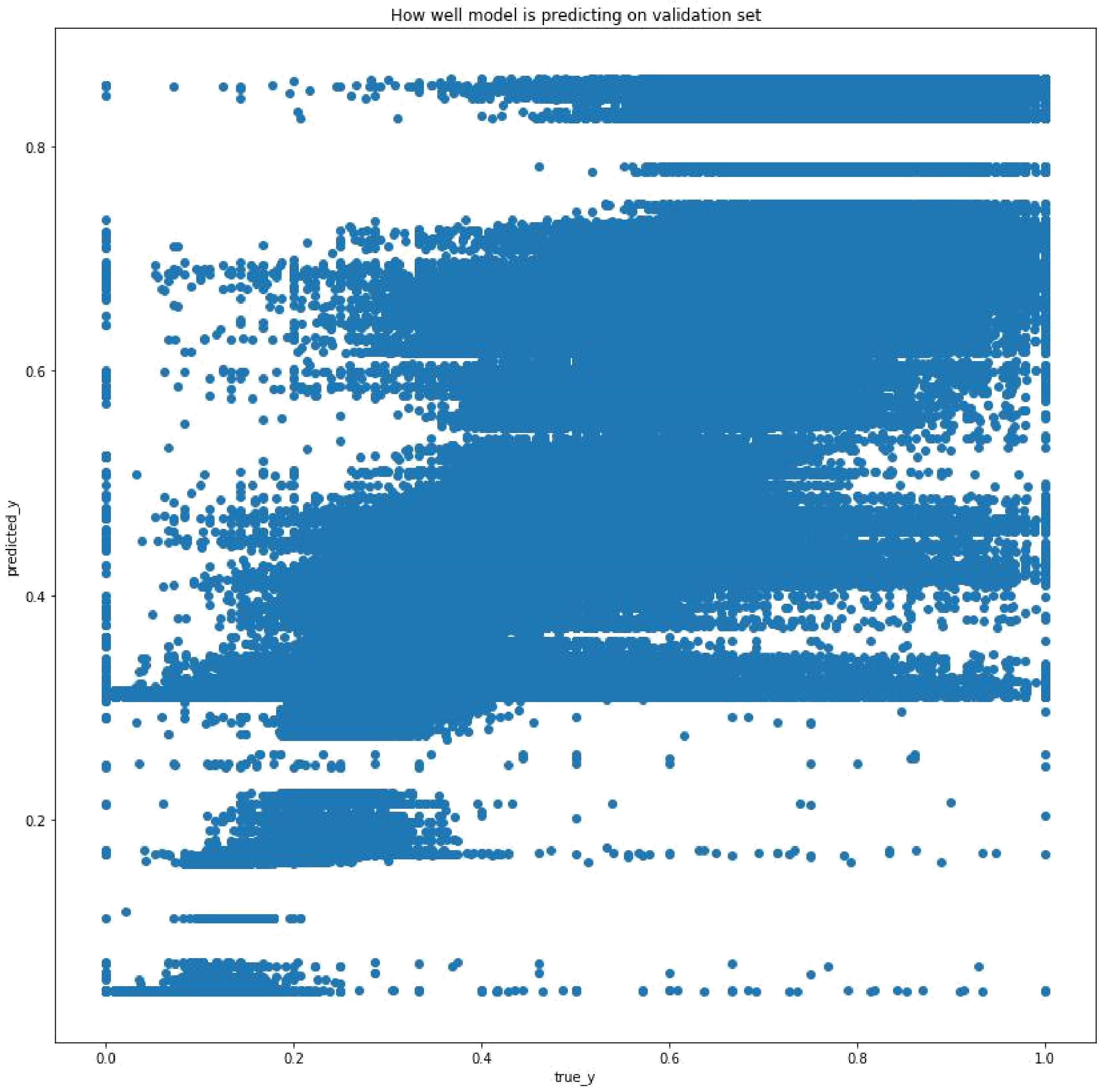


Light GBM training on no outliers-



As we can see that in benchmark -random forest models their is wide spread between true y (x-axis) and predicted y(y-axis) compare to final light gbm model. The wide spread means there is huge difference between correct value and predicted value on validation set. This means that our means absolute error on high spread will be higher compare to low spread. Hence my final model has a score of 0.02440 which is lower then benchmark scores of 0.02855 and 0.02828 on with and without outliers pressence respectively.

In addition to this I would like to show the prediction on validation set by adaboost-

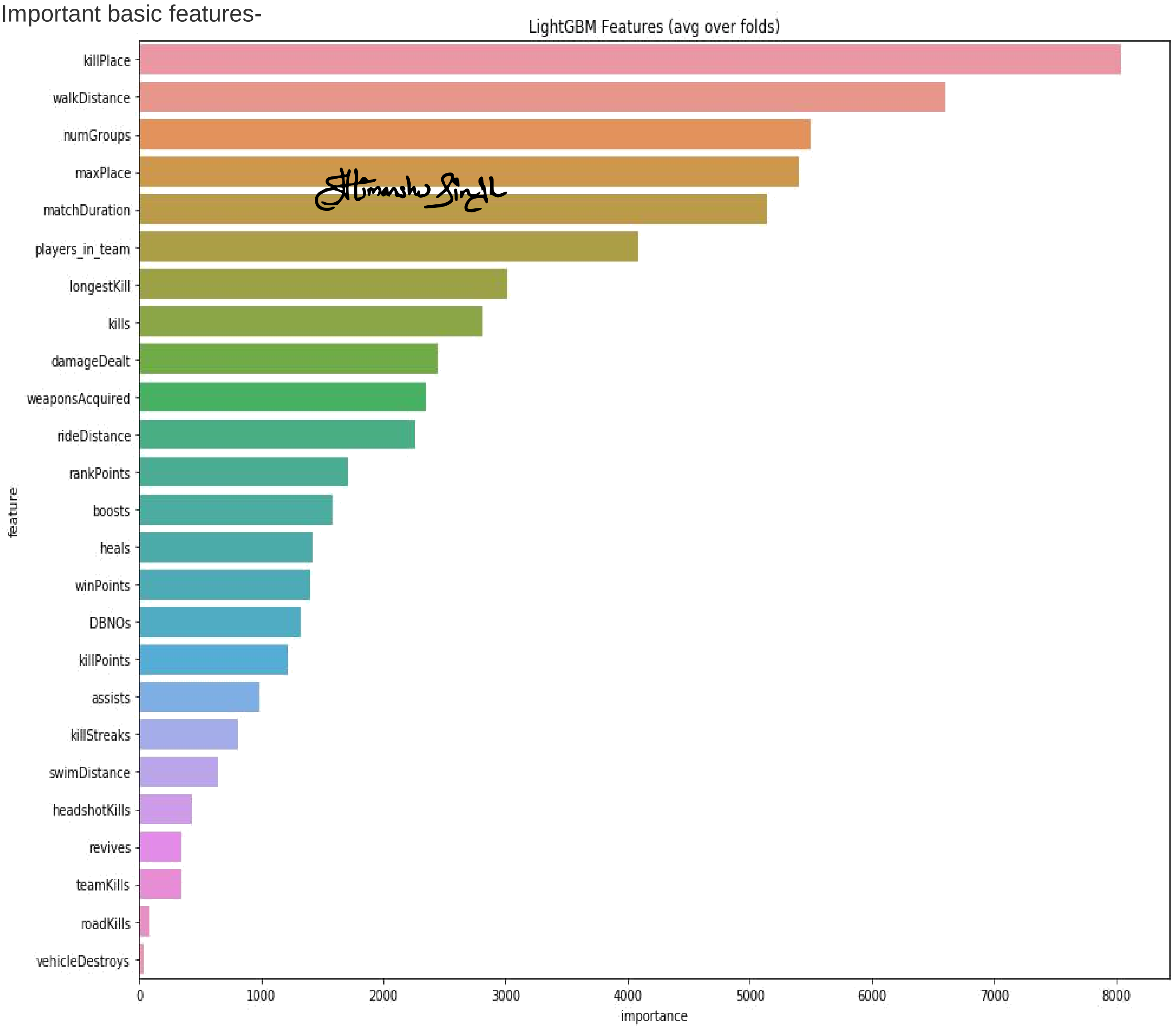


As we can see the adaboost predicting on validation set is nowhere performing better then benchmark model and final model. This is the reason it has poor score ,for predicting on test set, by 0.09993 and 0.09957 trained on presence of outliers and no presence of outliers respectively.

V. Conclusion

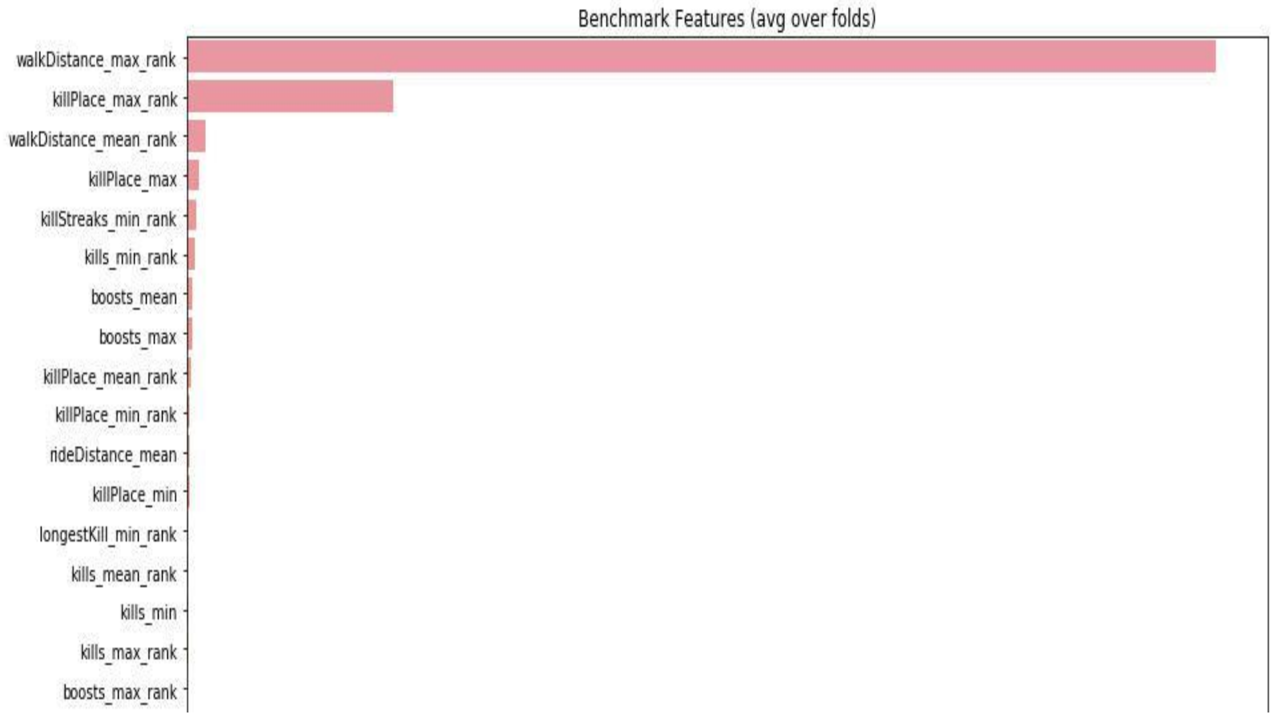
Free-Form Visualization

Important basic features-

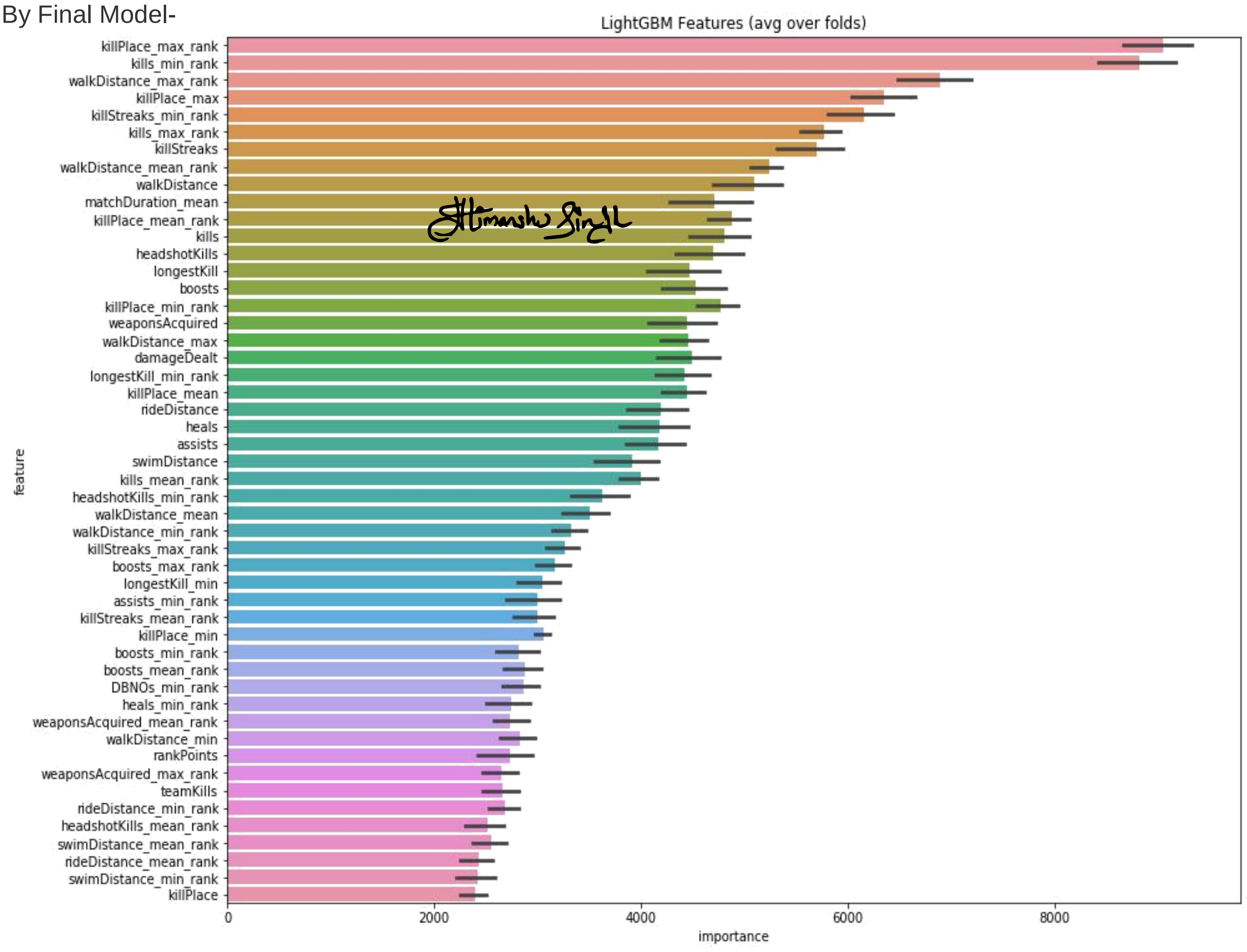


Important features after feature engineering-

By Benchmark model-



By Final Model-



From the Averaging above three figures of feature importance, its now clear that we founded the strategy behind winning the game. The strategies are the most important features of games like killplace, walk distance, weapon acquired, killstreaks, kills, boosts, heals, assist and ride distance. This important features are also similar to our correlation result from data exploratory section.

Why this features are important-

The killplace is a ranking of a player based on how many enemies he killed in that match. There are basically two types of players- proplayer and other is camping player. A proplayer has a kill strike rate maximum as he does not wait for a enemy to come, he just search for enemy to kill. This makes his walk distance and ride distance more. This also increases his killstrike rate. Any player who kills most has to deal with the damages comes from enemy. So, this increases the use of heals like med kits. The use of boost also increases with walk distance. This is due the fact that a proplayer needs to search, home to home by foot, for bullets and other items like scope and sniper to be in great power then enemy. The assist feature is very important in winning as this shows teamwork, if you helped your teammate in killing that one enemy, then its hard for enemy to give damage to you.

Reflection

This project turned out to be one that emphasizes on handling big data and utilizing computational resources; the given data was clean and thoroughly processed, the target concept is clearly defined, the features gave lots of room for engineering and research. All that was left was working around the sheer gigantic size of the data and optimizing all computing power at hand to fit it in, so much so that the data itself became the main problem instead of the target concept throughout the project.
Therefore, my problem solution not just includes data processing and training but also hardware optimization:

1. Loading the training and testing data
2. Reduce memory usage of them
3. Data exploration- On different features
4. Make notes of outliers found while exploring data.
5. Data preprocessing done to make data suitable for acceptence by different Machine learning algorithms.
6. Done feature engineering.
7. Keeping sensitivity analysis in mind- Train the models on both -without deleting outliers and with outliers.
8. Then Analysis the resultant scores of all model to get final best model.
9. From that understand the important features which solve our aim of finding winning strategy.

Improvement

- I also experiment with XGBoost which popularly known for scalability and performance. Grows tree depth wise making max_depth parameter the most important to control overfitting. But found that there is weakness in it. Due to large number of additional features after feature engineering and large training dataset, its crash in middle of training and shows badalloc error which is not in case with other algorithm. After some research on this problem I found on github fourm that internally it uses std::vector for allocation, which have a tendency to double the memory request when there is not enough of them. So, XGBoost will require additional memory for its column-based storage format. It is normal for it to crash at its half of the memory. So, I didn’t include in it, instead taken adaboost which also has good performance.

- The given pubg dataset by PUBG API tool, does not have equal number of data for each type of match. Like out of 4446967 players, basic mode match players data is more then Event mode match players. This is the reason I drop matchType feature. The strategy of basic mode and event mode is little different due to time durations and way to play this matches. If I got equal amount of data for each mode then I can train two different model for each which can give more better accuracy.

Reference

1. https://www.kaggle.com/c/pubg-finish-placement-prediction
   (Main dataset)
2. https://www.kaggle.com/lazyjustin/pubgplayerstats
3. https://www.kaggle.com/skihikingkevin/pubg-match-deaths/kernels
4. https://www.kaggle.com/arjanso/reducing-dataframe-memory-size-by-65
5. https://www.oreilly.com/ideas/setting-benchmarks-in-machine-learning
6. https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html
7. https://stackoverflow.com/questions/32210569/using-gridsearchcv-with-adaboost-and-decisiontreeclassifier
8. https://www.dbltap.com/posts/6176131-new-pubg-event-mode-flare-gun-to-last-10-days%5D
9. http://pubg.gamepedia.com/Game_Modes
10. https://comicbook.com/gaming/2018/10/04/pubg-event-mode-crash-carnage-/
11. https://www.kaggle.com/c/pubg-finish-placement-prediction/discussion/67742