Analysing and Predicting PUBG Player Performance

Author: U1856817

Abstract—This report performs statistical analysis on the PUBG player performance dataset released by Kaggle in 2018[15]. Key insights into the data will be identified before several regression and classification models are used to predict the match outcomes. Each player's ranking at the end of a match is forecasted to three degrees of specificity. The prediction models are compared and contrasted before results are discussed.

1 Introduction

To understand the significance of PlayerUnknown's Battlegrounds[1] (PUBG), one must be familiar with the idea of a 'battle royale' game. 'Battle royale' is a concept originating from a novel by Takami [2] published in 1999 that was adapted into a film in the following year[3]. It refers to combat between a large number of individuals or teams where the objective is to eliminate all others and become the only player or team left. The movie was controversial, but overall received well in Japan, inspiring other movies, writing and manga[4]. However it is the gaming community that has strongly embraced the concept due to the deep sense of immersion that video games allow working extremely well in a competitive battle royale setting.

Although components of 'battle royale' have previously been used in games such as 'Super Smash Bros.' [5], the large scale military shooter format used in PUBG was introduced by H1Z1[6] in 2015. H1Z1 was not successful in bringing the genre to the mainstream. When PUBG was initially released on 23rd March 2017, it quickly gained popularity; by October 2017 it had an average of 1,000,000 players online concurrently[7]. It soon became the most played game on the Steam platform[8], peaking at a staggering 3,236,027 concurrent player count[7]. A less realistic, free to play Fortnite[9] has become even more popular, surpassing PUBG in player count[10] and revenue[11]; however PUBG has continued to maintain it's popularity becoming the first game on Steam to have one million daily concurrent players for an entire year[12].

The immense success PUBG has brought the battle royale format into the mainstream. Countless games have copied and benefited from PUBG's success (Including H1Z1[13]), with the number of battle royale games increasing exponentially since it's popularity[6]. It will surely go down as one of the most influential games of all time.

With the release of an API to allow player and match statistics to be mined and analysed, the data science site Kaggle.com has provided a large anonymous dataset of player performance records from around 65000 matches[15]. This report aims to use this data to gain key insights into player performance across different game modes. We will then attempt to predict a given player's final rank to varied levels of specificity; i.e. exact position, placement range and the binary state of a being in the top ten or not. An important stage of the prediction process, in addition to selecting an appropriate classifier, will be feature selection as it is important to identify what attributes of a players performance actually effect their overall chances of winning; does one really need to engage enemies in combat or could one win by playing passively and hiding? The methodology, code, evaluation techniques and results of each stage will be outlined clearly in order to ensure the analysis can be reproduced and verified. Improvements, future work and the conclusions drawn will then be outlined.

2 The Game Mechanics

The key components of a game of PUBG are:

- **The Map:** Four unique maps are available as of writing this report. Each map has a mixture of terrains with key settlements/towns that tend to have a higher concentration of weapons, items and thus players.
- **Weapons & Items:** A wide array of weapons, healing items and boosts are scattered around the map in a semi-random distribution. Weapons are more concentrated around certain locations on each map.
- **Vehicles:** A wide array of drivable land and sea vehicles are available. Each player has a limited number of slots for weapons and items; although finding 'bag' items like backpacks and allow players to increase inventory size.
- **The Players:** Up to a 100 players are deployed per match, either solo or in teams of 2, 3 or 4.
- **The Playzone (The Circle):** The main driving force of a PUBG match. This is a random circular area of the map that players must remain in. If outside the circle, players take damage and eventually die. The playzone shrinks in 8 stages, each stage is faster and more damaging than the previous. Before the actual playzone (represented by a blue circle) shrinks, a white circle on the map shows players where the next playzone will be. Each circle representing the new playzone is randomly generated.

A standard PUBG match begins with up to a 100 players joining a lobby. The players are then transported in a plane that travels in a randomly generated straight line across an island. Players can choose to exit the plane and parachute down to the surface of the map at any point; if any player does not jump out, they are automatically ejected from the plane at the far boundary of the map.

Upon leaving the plane, players do not have any items or weapons; just their bare hands for combat. Thus, selecting where to land at the start of the match is a crucial. Some

players/teams tend to avoid certain settlements, as they tend to have a higher concentration of weapons and players; thus avoiding a highly competitive and dangerous initial few minutes. Some are more aggressive and prefer to race to these locations in search of better loot. They often have to engage opponents immediately; this is called 'hot dropping'.
Once on the ground, the initial priority is to scavenge for weapons and amour whilst getting an idea of where enemy players might be and where the playzone is shrinking to. The playzone begins with a very large circle; diameter of around 4500m. Players are shown the next stage; around 3000m with a white circle, before the actual zone begins to shrink. This repeats with smaller and smaller playzones, the smallest being just 46.4m, until just one player or team is left.
As the playzone gets smaller, players are forced to stay on the move. They are pushed into a more concentrated area, resulting in more encounters. Combat is mainly based around firearms, although hand-to-hand combat may also take place. Once a player takes sufficient damage they may be 'downed'; immobilised until a team-mate revives them. If no revival takes place, or if a player is playing alone, they are 'killed' and given their ranking according to how many players are left alive. They are not allowed to rejoin the match.
Evidently there are countless strategies players can adopt, due to the wide variety of items and vehicles available, the vast and diverse maps as well as the unpredictability that comes with 100 competitive human players. For more information, Chris Carter from Polygon has an insightful article on the basic concepts and strategies of PUBG[14].

3 THE DATA

The data is made up of 6.38million records in total, divided into 2 files, training (4.45 million) and testing (1.93 million). For the purpose of this report we will be separating our own training and testing sets. Each record is made up of 29 attributes that describe one player's statistics throughout one match.
The following is a short description of the attributes used in this report:

- **DBNOs** - # of players 'downed'
- **assists** - # of enemies damaged that were killed by team-mates
- **boosts** - # of boost items activated
- **damageDealt** - Total amount of damage dealt on other players
- **headshotKills** - # of kills by shooting to the head
- **heals** - # of healing items consumed
- **Id** - player ID
- **killPlace** - Where this player is ranked in the current match by kills
- **killStreaks** - # consecutive kills in a short time-frame
- **kills** - Total enemy player kills
- **longestKill** - Distance between current player and the farthest enemy killed. "This may be misleading, as downing a player and driving away may lead to a large longestKill stat."[15]
- **matchId** - The ID of the match the player was a part of.
- **matchType** - Game mode i.e. solo, duo, squad or custom
- **revives** - # team-mates revived
- **rideDistance** - Distance travelled by vehicle
- **roadKills** - # of kills while riding in a vehicle
- **swimDistance** - Distance travelled by swimming
- **teamKills** - # of team-mates killed
- **vehicleDestroys** - # of vehicles destroyed
- **walkDistance** - Distance travelled by walking
- **weaponsAcquired** - # of weapons picked up
- **groupId** - ID for each team in a match
- **numGroups** - # of teams data is available for in a given match
- **winPlacePerc** - The "percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match."[15]

The data is downloadable in *.csv* format and will be imported and analysed in Python 3.7 using the libraries *pandas*, *numpy*, *sklearn*, *matplotlib* and *seaborn*.

3.1 Preprocessing

3.1.1 Removing Null Values

The first steps taken involve preprocessing the data in order to clean and better organise the records. Any null values in essential attributes are identified using the numpy $'notnull()'$ function. Just one record with a *null* value for **winPlacePerc** was found, thus the decision was made to delete this record.

3.1.2 Standardising matchType

PUBG has 3 primary match types, *Solo*, *Duo* and *Squad*; in addition to custom matches. Each match type may allow both third and first person views or force players to play using a first person view, depending on the settings of the server. Therefore the classes for the **matchType** attribute have four classes per standard game type (*solo*, *normal-solo*, *solo-fpp*, *normal-solo-fpp*, *duo*, *normal-duo*, *duo-fpp*, *normal-duo-fpp*, *squad*, *normal-squad*, *squad-fpp*, *normal-squad-fpp*) in addition to the names of custom games.
For the purpose of this report we will not be considering the differences between matches with different gameplay view settings. Further, as custom games alter the game rules, these records may not reflect the occurrences of a standard game type.
First, a total of 9881 custom games are identified and removed. Then the match types are standardized by grouping them into three clear labels (Solo, Duo, Squad).
E.g.

```
data['matchType'][data['matchType'] == 'solo
    -fpp'] = 'Solo'
```

3.1.3 Generating New Attributes

New attributes are created using exiting features in order to gain further insight into, better organise, and capture the different aspects of each player's performance.

- **totalDistance** : The total distance walked, swum and travelled by vehicle. Players exist with a totalDistance of 0, which should not be possible. This is addressed when identifying anomalies and cheaters in subsection 3.3.

```
data['totalDistance'] = data['
    walkDistance'] + data['rideDistance']
     + data['swimDistance']
```

- **totalKills** : The total kills by weapons and vehicles.

```
data['totalKills'] = data['kills'] +
    data['roadKills']
```

- **totaltems** : The total number of boosts and heals used.

```
data['totalItems'] = data['boosts'] +
    data['heals']
```

- **playersInMatch** : The total number of players that have joined each match. This is calculated by grouping all the players by **matchId** and counting the how many belong to the current match.

```
data['playersInMatch'] = data.groupby('
    matchId')['matchId'].transform('count
    ')
```

- **placementRange** : Created using the winPlacePerc attribute. Denotes what 'range' each player is placed in at the the end of the game; 1st-10th, 10th-20th, 30th-40th etc. Creates a distinct set of values for prediction in addition to the continuous **winPlacePerc**.

```
data['placementRange'] = raw_data.apply(
    ranges, axis=1)
# ranges() is a function that returns
    the range given the winPlacePerc
    value
```

- **topTen** : A binary class to denote if a player comes in the top ten at the end of a match or not.

```
data['topTen'] = data.apply(top_ten,
    axis=1)
# top_ten() is a function that returns 1
     if a player is in the top ten, 0 if
    a player is not
```

3.1.4 Separating Files

Once the data is preprocessed, it is separated into two sets of *.csv* files; one set containing all the records and one with just 500,000 for quick and easy testing. Each set has 4 files, one with all the records in addition to three separate files separated by **matchType**. This is because depending on the match type, different features of the data must be analysed. For example, players cannot be 'downed' or get assists and team-kills in *Solo* games.
Processing and separating the data once and then generating new files removes the need to do this every time a classifier is run during the next stages of the report.

3.2 Insights

This section will outline insightful and interesting statistical characteristics uncovered during the initial analysis.

3.2.1 playersInMatch

We begin by looking at the frequency distribution of the **playersInMatch**, seen in Fig. 1. It can be seen that although relatively not many matches are full (100 players), most have 90+. It is possible that the amount of players in a match could relate to certain other attributes, e.g. with less players on the map, it is harder to find enemies to kill. However, intuitively we can deduce that most attributes may not be affected, such as **walkDistance**; no matter how many players are on the map, the playzone forces players to keep moving and they would probably be just as careful when picking their route.

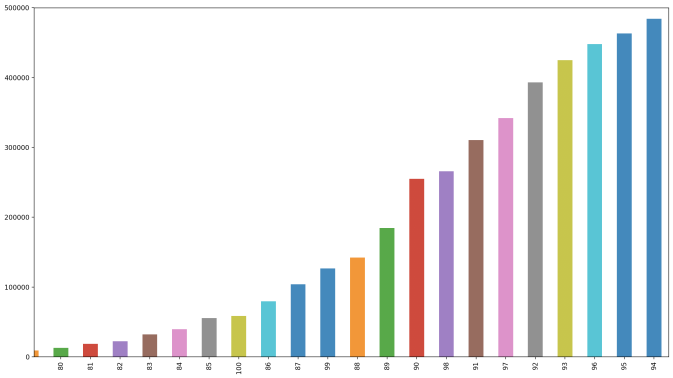

Fig. 1. Frequency distribution of playersInMatch

To get a better idea of the correlations between attributes, we then generate a heat-map of r values. The heat-map, seen in Fig. 2, is given a threshold of $|0.4|$. All correlations less than 0.4 are given the value of 0.

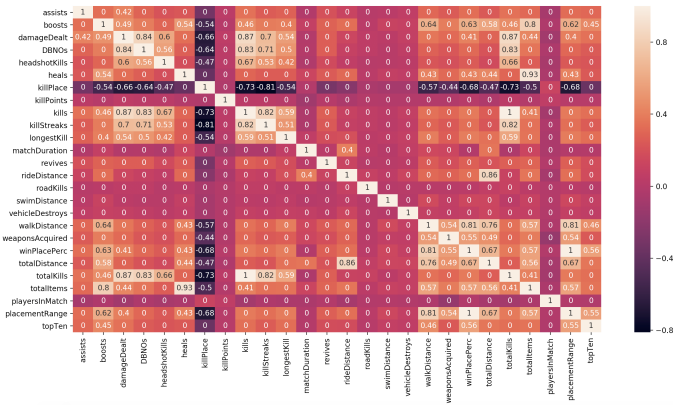

Fig. 2. Heat-map of r values

Judging by the r value alone, it seems the **playersInMatch** does not significantly correlate to the other attributes, which somewhat confirms the intuitive hypothesis.

3.2.2 Performance vs kills

Most competitive military style shooters, such as PUBG, have a huge focus on combat and thus have many objectives based around getting the most kills. With the emergence of battle-royale modes, it is debated whether getting kills is actually helpful when aiming to be the last one standing. Getting into combat should increase the chances of death,

hiding while other players eliminate each other should be a better strategy, right? Judging by just the r value, it seems the **totalKills** does not significantly correlate with the chances of doing better in the rankings($r = 0.41$). This is supported by the box plot in Fig. 3 due to the wide quartile ranges for lower kills and the high amount of anomalies at the range $> 10kills$. Looking at the averages, the mean kills by players is 0.95, while the 90% quantile gets 3.0; while the mean for players in the top 10% is 2.63 kills with their 90% quantile gets 6.00. Although there is a minor correlation with the amount of kills, it seems this is not significant.

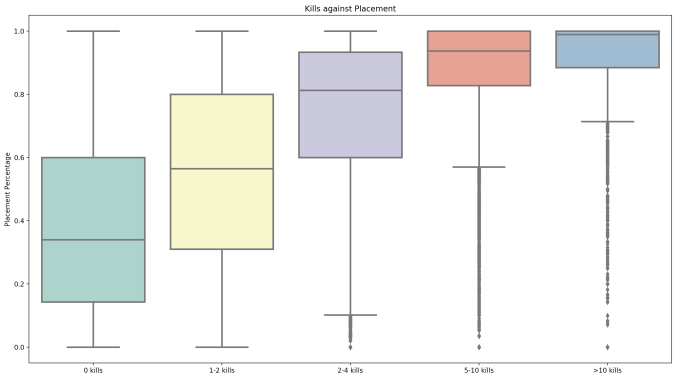

Fig. 3. Box plot of totalKills against winPlacePerc

On the other hand, **killPlace** has the second highest correlation to the rankings ($r = -0.71$). This could mean, by being relatively more aggressive than one's opponents, one has a better chance of winning. This could also be a result of the data being collected after the game has ended; as the player who has placed better would be involved in the fast paced elimination stage at the end when the playzone is much smaller. There is a clear correlation here but it is uncertain whether it causes better performance or if it is the result of better performance.

By looking at the probability distribution of the **placementRange** in Fig. 4, we see that a player is most likely to be ranked low (badly), then high (well) on the rankings but least likely to finish somewhere in the middle. This suggests a player is more likely to 'die' either at the early stages of a match, or towards the end. This intuitively makes sense as these are the two stages of a PUBG match that have the most conflicts; the start when all the players are rushing to settlements and loot, and the end when the playzone has shrunk.

3.2.3 walkDistance & boosts

As this report aims to predict **winPlacePerc**, **placementRange**, **topTen**, we will now focus on the two other attributes that correlate to these.

The attributes **walkDistance** and **boosts** are significantly correlated to player performance with r values of 0.81 and 0.63 respectively. We see that **weaponsAcquired** has a r of 0.57 overall, which rises to 0.62 when considering *Duo* matches.

The mean distance travelled by players is 1190.06m, while the 90% quantile travels 2937.00m. When considering only players in the top 10%, the mean value is 2819.16m and the

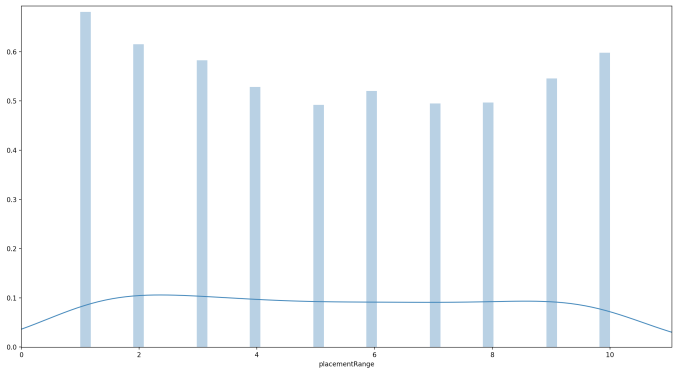

Fig. 4. P(placementRange) distribution

90% quantile is 3934.00m. There is a clear increase when considering players who rank better; however similar to **killPlace**, there is a chicken or the egg problem here where this could be a result of surviving longer rather than a causation of surviving longer. Intuitively we know that the longer a player stays 'alive' the more they have the chance to move around, but the difference in distances is large enough where it is unlikely that the increase is just by being alive; the winning players are likely to be more mobile in general.

The correlation of **boosts** brings similar concerns. The mean value for all players is 2.53 items, with a 90% quantile of 8.00. For players in the top 10%, the mean is 6.94 items, with a 90% quantile of 13.00. The longer a player is alive, the more boosts they have the opportunity to pick up. However this attribute measures the boosts *used* rather than *picked up*. The players who make the choice to pick up and use boosts in combat appear to have an advantage over players who do not. This applies to **weaponsAcquired** as well, a player does not automatically pick up weapons in PUBG, one must willingly choose to pick up weapons that then take up one of the limited slots in an inventory. A skilled player would be prioritising between picking up and dropping weapons throughout the game.

3.3 Identifying Anomalies and Cheaters

Players spend between 1-3 minutes [16] at the start of each match parachuting down to the surface. This dataset does not capture the distance covered during this time, and thus, players who somehow get eliminated before landing, or land and fail to move at all have a **totalDistance** of 0. As there is roughly, 97370 player records (2%) with a **totalDistance** of 0, it would be useful to identify how this is possible and thus whether to consider these records or not.

The game prevents players eliminating themselves by forcing the parachute to open at a certain height. There may be rare cases where players who land early are able to shoot down other players as they parachute. Further, players who are 'AFK' (away from keyboard) may land, but die without moving at all. The problem of AFK players is quite common, and may explain a lot of these records, however there is another issue that has plagued the PUBG servers that may contribute.

Hackers and cheaters have been an ongoing issue in PUBG[19], one common exploit allows a hacker to eliminate players who are yet to land by 'downing' them in mid air

[17]. This could very well explain some of these records. Further, we see that 1535/97370 of the players with **totalDistance** of 0 still have more than 0 kills; which could be hackers who are known to 'float' around the map and thus do not record a distance attribute[18].
Whether or not these records are AFK players, hackers or their victims, most of these cases do not reflect the occurrences of a standard PUBG match and therefore the decision was made to remove these records.
We now turn towards the **kills** and **headshotKills** attributes. Several players have a headshot accuracy of 100%, the best being 15/15 headshots. These seems like either hackers or extremely skilled players. Looking at the probability distribution of *headshotKills* accuracy in Fig. 5, the latter seems more likely. It is known that aiming for the head is a basic strategy for all players.

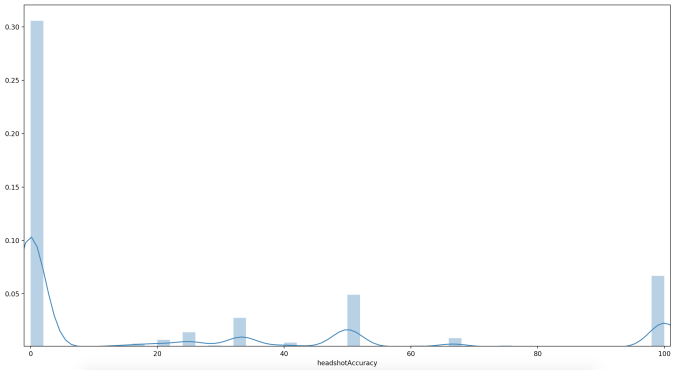

Fig. 5. P(headshotKills/kills) distribution

The player with the highest kills has 72, got 64 headshots and travelled a total of just 728.1m. These are rare and mostly likely are the product of 'hot dropping' into concentrated areas where a lot of conflicts arise in a small area. Extremely high kills are clear anomalies but are unlikely to be hackers. As we have already eliminated hackers by removing players who didn't register distance, and we see that during feature selection that the *kills* attribute is not considered, the decision was made to keep these records as they still capture legitimate PUBG behaviour.

4 Predicting Match Outcomes

We now move on to the predictive stage where we will apply and evaluate several regression and classification algorithms in order to predict **winPlacePerc**, **placementRange** and **topTen**. All models will be run on the same set of 10KFolds each time, generated using the *sklearn* library. This is to ensure the evaluation is fair and the models do not over-fit.
We will be testing a total of 8 algorithms. This is possible by creating individual functions that handle each model and then adding these functions into two lists, REGRESSIONS and CLASSIFIERS. We can then iterate through either list and apply the same 10 KFolds to each algorithm.

```
# Coding approach to evaluate multiple
    models using 10KFolds
# Predicting player placement range
for clf in CLASSIFIERS:
        # Parameters for kfold: data,
            classifier, number of folds,
            feature to predict
        kfold(processed_data, clf, 10, '
            winPlacePerc')
for reg in REGRESSIONS:
        # Parameters for kfold: data,
            classifier, number of folds,
            feature to predict
        kfold(processed_data, reg, 10, '
            winPlacePerc')
```

4.1 Feature Selection

We will carry out the feature selection mainly based on the Pearson Correlation Coefficient or *r* value:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}$$

For all three match types, the attributes **boots**, **killPlace** and **walkDistance** are the most correlated (See Fig. 2). We do not use **totalDistance** as although it has a high *r* value with performance, it is also highly correlated to **walkDistance** and thus we must avoid multicollinearity.
When considering just **matchType** Duo, the **weaponsAcquired** attribute also shows high correlation($r = 0.62$). We will evaluate the use of this feature as well for all match types to see if the accuracy increases significantly (¿1%).
By visualising the QQ plots for these attributes we see that they are all heavily skewed towards lower values, an example is shown in 6. The distribution of **killPlace** is more or less uniform (Figure 7). This must be noted for the classification stage as some classifiers assume normal distribution.

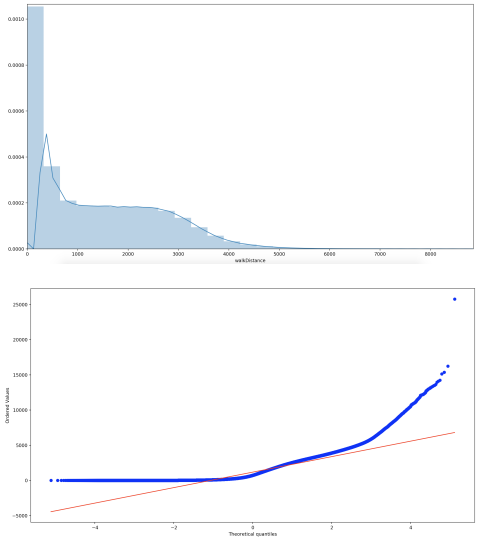

Fig. 6. Probability distribution and QQ plot of walkDistance

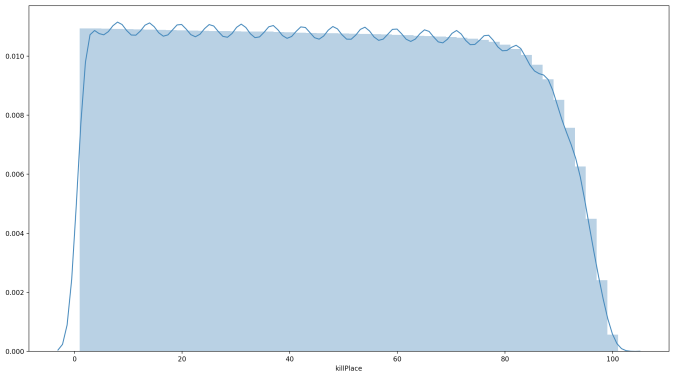

Fig. 7. P(killPlace) distribution

Model	Solo	Duo	Squad	Overall
MLR	0.800/0.099	0.788/0.106	0.743/0.119	0.756/0.115
RFR	0.898/0.066	0.862/0.0777	0.791/0.099	0.811/0.093

TABLE 1
winPlacePerc Regression Model Comparison r^2/MAE

4.2 Predicting the Exact Position

The exact positions of a player in the final ranking is represented by a continuous value between 1 and 0 (1 being first place, 0 being last) in the **winPlacePerc** attribute. Due to this value being continuous, we opt for regression models. The scatter plots (E.g. Fig. 8) of individual features do not

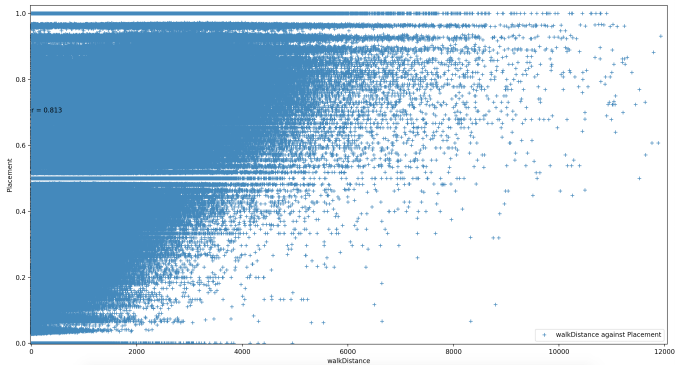

Fig. 8. walkDistance against winPlacePerc

clearly suggest a linear, quadratic or cubic model. We will therefore evaluate a multi-linear regression model (MLR) and a random forest regressor (RFR) model. These models will be evaluated using their r^2 values as well as the mean absolute error.

We evaluate the use of three features against four, and deduce that adding **weaponsAcquired** influences the accuracy significantly for all match types, especially with MLR where we see an increase from $r^2 = 0.78$ to for $r^2 = 0.80$ for *Solo*. From Table 1 it is clear that the decision tree method (random forest regressor) with 10 estimators and no depth limit is significantly more effective considering both accuracy measures throughout all match types. The highest accuracy for both models is for predicting *Solo* matches. By taking a closer look at the coefficients and weightings given to each attribute (Table 2), the decision tree is heavily depended on the **walkDistance** to reduce uncertainty. On the other hand, the multi linear regression method does not

Model	killPlace	boots	walkDistance	weaponsAcquired
MLR	-0.004	0.012	0.000	0.016
RFR	0.169	0.012	0.80	0.015

TABLE 2
winPlacePerc Feature Weights

Model	Solo	Duo	Squad	Overall
MLR	79.1%	78.0%	73.6%	74.8%
RFR (10 estimators)	88.8%	85.3%	78.5%	80.4%
RFC (10 estimators)	45.3%	41.4%	35.9%	37.2%
CNB	29.9%	21.3%	24.2%	22.6 %
GNB	35.3%	36.0%	33.6%	32.7%
KNN (k = 101)	52.3%	49.3%	43.6%	45.3%
DT	43.5%	40.0%	34.8%	36.2%

TABLE 3
placementRange Model Comparison % Classified Correctly

weight this attribute at all, it focuses on the **boosts** and **weaponsAcquired** which we know from the r values are the least correlated of the four attributes.

4.3 Predicting the Placement Range

We have created a new attribute, **placementRange** that captures what range in the overall rankings a player finishes in. It is made up of 10 distinct numerical values; from 1 (finishes in the last 10% of players) to 10 (finishes in the first 10% of players). This is approached as both a regression problem and a 10 class classification problem. Therefore we will evaluate Multi Linear Regression (MLR), Random Forest Regressor (RFR), Gaussian Naive Bayes (GNB), Complement Naive Bayes (CNB), K Nearest Neighbour Classifier (KNN), Decision Tree (DT) and a Random Forest Classifier (RFC). The ease and convenience of the *sklearn* library allows us to compare a large amount of models.

It should be noted that the features are not normally distributed, therefore a Gaussian Naive Bayes model should not be suitable. We also use a Complement Naive Bayes which should improve on the assumptions made by the GNB; using a power law distribution rather than a gaussian distribution[20].

These models will be assessed using the mean overall accuracy of misclassified cases across all classes. This is a very harsh metric for a 10 class classification, however it is the most suitable in this case as a 10x10 confusion matrix is too large to effectively analyse and false positive/negative values for each class have equal implications.

From the accuracies in Table 3 we see that the classifiers are clearly not as effective as the regression models. This may be due to the large amount of classes as well as the harsh accuracy measure.

The RFR model is the most effective over all match types while all models peaked accuracy on the *Solo* match type. This behaviour is identical to what was observed when predicting **winPlacePerc**. When looking at the weightings given to each feature, all models have higher values for **walkDistance** except for the two Naive Bayes models and the Multi Linear Regression which have the highest weighting for **weaponsAcquired**.

Model	killPlace	boots	walkDistance	weaponsAcquired
MLR	-0.040	0.108	0.001	0.160
RFR	0.163	0.013	0.808	0.016
RFC	0.283	0.035	0.598	0.084
GNB	3.051	1.667	1.759	5.093
CNB	3.010	7.143	0.050	5.748
DT ()	0.273	0.039	0.581	0.107

TABLE 4
placementRange Feature Weights

Model	Solo	Duo	Squad	Overall
LR	86.9%	86.5%	83.7%	84.9%
RFC (10 estimators)	86.6%	85.6%	82.5%	83.4%
CNB	85.5%	84.9%	82.6%	83.7%
GNB	86.7%	86.3%	83.5%	84.7%
KNN (k = 101)	87.1%	87.4%	85.8%	86.8%
DT	83.9%	82.9%	79.7%	80.9%

TABLE 5
topTen Model Comparison % Classified Correctly

4.4 Predicting Placement in the Top 10%

The third feature this report will attempt to predict is a binary value **topTen**, which is 1 when a player finishes in the top 10% and 0 when a player does not. The immediate concern when approaching this problem is the huge imbalance in the classes, a majority of players do not finish in the top 10% and therefore most models will have a bias towards this class (0). Therefore the decision was made to balance the classes in the data prior to the KFolds algorithm.

For this section we will be comparing Logistic Regression, Random Forest Classification, Gaussian Naive Bayes, Complement Naive Bayes and a Decision Tree classifier. All models will be compared using the mean overall accuracy of misclassified cases across all classes. Table 5 outlines the accuracies for each, and it is clear that all models are similarly effective; K Nearest Neighbour is marginally better than others. Similar to the other prediction attempts, overall best accuracy for all models is for the *Solo* match type with a mean of 84.6%.

The confusion matrices expressed in Table 6 are for all models predicting **topTen** across all match types. The matrices are very similar and false/true negatives/positives are more or less even. However despite balancing the data, some models such as CNB, GNB and LR that still have higher false positives for 0.

4.5 Results Analysis

Feature selection has concluded that some attributes that one may intuitively attribute to performance in a military shooter may not mathematically correlate. Using just four attributes we are able to effectively predict performance to three different levels of specificity. Through the three predictive cases we see a random forest regressor is ideal for two of them, **winPlacePerc** and **PlacementRange**. A random forest regressor is a type of decision tree that chooses the best splits out of several trees with random splits[21]. It is an extremely effective and powerful machine learning algorithm[22] and this is reflected in the results. However, a random forest gives very little control during design, allowing only to select the number of trees and the random start state; sometimes seems like a black box.When predicting **topTen**, all evaluated models have similar accuracies and confusion matrices. All models are very effective, however KNN is marginally more accurate.

A support vector classifier was also attempted as an additional classifier, however due to a complexity that is above n^4, it proved to be not feasible with such a large dataset.

A trend that was identified throughout all prediction attempts and classifiers was that *Solo* match accuracy was much better, with up to a 10% jump for **placementRange** using RFR. The next most accurate match type was *Duo*. This suggests that the features used **walkDistance**, **killPlace**, **boosts** and **weaponsAcquired** may not capture the winning characteristics of a player in a team. This intuitively makes sense as the boosts and weapons used do not benefit a team as a whole. In addition, the placement one player has in the kills ranking does not always reflect the entire team's performance.

RFC	1	0
1	39070	7893
0	7598	39068

GNB	1	0
1	38001	8802
0	5474	41351

CNB	1	0
1	34964	11881
0	3432	43351

LR	1	0
1	38499	8346
0	5911	40872

KNN	1	0
1	38102	8593
0	9314	37620

DT	1	0
1	38127	8718
0	9269	37514

TABLE 6
Overall **topTen** Confusion Matrices

5 Extension & Improvement

This section will suggest further work and improvements that could be made on top of the analysis and prediction models generated as a part of this report. The dataset used here is made up of data from the end of a match, the final rankings have already been decided. An ideal prediction model would be able to predict winners as a match goes on, based on a player's historical player data as well as the periodically updated statistics of the on-going match. This would require historical data of players as well as timestamped data from different points in a match. A system that can predict the ranking before a match is completed would have applications in e-sports game analysis and betting.

During the prediction stage of this report we uncovered reductions in accuracy for the team based match types. A better measure of team performance needs to be captured. The obvious features such as assists and revives do not seem to correlate and thus a different approach might be suitable, such as grouping the features of a squad to a statistic about the entire team rather than the individual player.

6 Conclusion

This report is a successful baseline for the analysis and prediction of player performance during PUBG matches. It begins by investigating the key characteristics and patterns in the data, before identifying anomalies and thus possible cheaters and AFK players. We then carry a feature selection stage to then predict where each player ranks at the end of the game to three degrees of specificity. Several different machine learning algorithms are evaluated and reasonable accuracy is achieved in all three cases. Finally, the results are discussed along with possible improvements are future work.

REFERENCES

[1] PLAYERUNKNOWN'S BATTLEGROUNDS, 2018. [Online]. Available: https://www.pubg.com/. [Accessed: 13- Dec- 2018]

[2] K. Takami and Y. Oniki, Battle royale. San Francisco: Haikasoru, 2012.

[3] K. Fukasaku, Battle Royale. Japan, 2000.

[4] "Manga", En.wikipedia.org, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Manga. [Accessed: 13- Dec- 2018]

[5] "Super Smash Bros. Ultimate for the Nintendo Switch system", 2018. [Online]. Available: https://www.smashbros.com/en_US/. [Accessed: 13- Dec- 2018]

[6] D. Green, "A Timeline Of The Battle Royale Game Mode", Dorkly, 2018. [Online]. Available: http://www.dorkly.com/post/86647/battle-royale-games. [Accessed: 13- Dec- 2018]

[7] "PLAYERUNKNOWN'S BATTLEGROUNDS - Steam Charts", Steamcharts.com, 2018. [Online]. Available: https://steamcharts.com/app/578080#All. [Accessed: 13- Dec- 2018]

[8] F. Brown, "PlayerUnknown's Battlegrounds beats Dota 2's highest concurrent player record", pcgamer, 2018. [Online]. Available: https://www.pcgamer.com/uk/playerunknowns-battlegrounds-beats-dota-2s-highest-concurrent-player-record/. [Accessed: 13- Dec- 2018]

[9] "Epic Games' Fortnite", Epic Games' Fortnite, 2018. [Online]. Available: https://www.epicgames.com/fortnite/en-US/buy-now/battle-royale. [Accessed: 13- Dec- 2018]

[10] A. Chalk, "Fortnite passes PUBG with 3.4 million concurrent players", pcgamer, 2018. [Online]. Available: https://www.pcgamer.com/uk/fortnite-passes-pubg-with-34-million-concurrent-players/. [Accessed: 13- Dec- 2018]

[11] "Fortnite surpasses PUBG in monthly revenue with $126 million in February sales", The Verge, 2018. [Online]. Available: https://www.theverge.com/2018/3/21/17148918/fortnite-battle-royale-pubg-monthly-revenue-sales-126-million-superdata-research. [Accessed: 13- Dec- 2018]

[12] "PUBG becomes first Steam game to have one million concurrent players 365 days running", GamesIndustry.biz, 2018. [Online]. Available: https://www.gamesindustry.biz/articles/2018-09-10-pubg-becomes-first-game-on-steam-to-have-one-million-concurrent-players-every-day-for-a-year. [Accessed: 13- Dec- 2018]

[13] "H1Z1 - Steam Charts", Steamcharts.com, 2018. [Online]. Available: https://steamcharts.com/app/433850#All. [Accessed: 13- Dec- 2018]

[14] "Understanding Playerunknown's Battlegrounds", Polygon, 2018. [Online]. Available: https://www.polygon.com/playerunknowns-battlegrounds-guide/2017/6/9/15721366/pubg-how-to-play-blue-wall-white-red-circle-map-weapon-vehicle-inventory-air-drop. [Accessed: 14- Dec- 2018]

[15] "PUBG Finish Placement Prediction (Kernels Only) — Kaggle", Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/c/pubg-finish-placement-prediction/leaderboard. [Accessed: 14- Dec- 2018]

[16] "PUBG - All About Parachuting", Gameplay.tips // Game Guides, Walkthroughs, Tips & Tricks, Cheat Codes and Easter Eggs, 2018. [Online]. Available: https://gameplay.tips/guides/658-playerunknowns-battlegrounds.html. [Accessed: 16- Dec- 2018]

[17] "Hacker Kills Several People from Parachute PUBG", YouTube, 2018. [Online]. Available: https://www.youtube.com/watch?v=75R-OoIy-LY. [Accessed: 16- Dec- 2018]

[18] "PUBG - floating hackers", YouTube, 2018. [Online]. Available: https://www.youtube.com/watch?v=mMViGfSCoCc. [Accessed: 16- Dec- 2018]

[19] "PUBG Corp. confirms the arrest of 15 hackers in China", Polygon, 2018. [Online]. Available: https://www.polygon.com/2018/4/30/17302174/pubg-cheating-hacking-arrests-china. [Accessed: 16- Dec- 2018]

[20] Rennie, J.D., Shih, L., Teevan, J. and Karger, D.R., 2003. Tackling the poor assumptions of naive bayes text classifiers. In Proceedings of the 20th international conference on machine learning (icml-03) (pp. 616-623).

[21] Liaw, A. and Wiener, M., 2002. Classification and regression by randomForest. R news, 2(3), pp.18-22.

[22] Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems?. The Journal of Machine Learning Research, 15(1), pp.3133-3181.