PUBG Finish Placement Prediction

Group 4:

Monica Kumar

Nihit Save

Sabyasachi Choudhury

Shruti Rajani

Sidheswar Venkatachalapathi

Table of Contents

1. Background

PlayerUnknown's BattleGrounds(PUBG), launched in March 2017, has acquired immense fame in a very short span of time. It is the fifth best selling game with over 8 million copies sold and over a million active monthly players. The game, developed by PUBG Corporations, is currently available in Window, Xbox, iOS, Android and PlayStation platforms.

HOW THE GAME WORKS

Upto 100 players, either as individuals or teams, are dropped into a battle map unarmed and the objective is to win the game by being the last-man/team left alive in the island. Players start with no gear except for their clothing. Once the game begins, players can search buildings, they can scavenge weapons and armors to kill opponents, move from place to place via running/swimming/riding vehicles and collect healing items to boost health. Every few minutes, a circular perimeter called the PlayerZone begins to converge towards a random point in the map. Any player outside this region begins to take incremental damage and if they do not reach inside the area within a particular time, they would risk being eliminated from the game. When players complete a round, their in-game statistics are recorded and points are given based on their skills and how well they finished the game.

ABOUT OUR DATA

Since players from all across the world compete in a single game and at any given point, numerous channels of game rooms are being played across the servers. Hence, every second, petabytes of data are collected on a player's game statistics. As Business Analytics students, our team felt that this data could potentially be analysed from a gamer's perspective and results could be drawn on how one can place their game better than others for a strategic advantage. We will also explore how each variable in the data does/doesn't play a role in the final placement of a player. For our analysis, we took over 65000 games worth of anonymized player data, split into training and test sets, made available by Kaggle through PUBG Developer API.

PROBLEM STATEMENT

"Predict the final position of a player from final in-game stats and initial player ratings."

2. Objectives

Deriving from our key problem statement, we wish to address the following objectives in our report:

- Explore with the PUBG dataset and understand how the various variables are distributed across.
- Given the complexity of the data, we want to find the correlations between numeric variables and our target variable to establish any relationships and patterns that will help identify possibilities for dimension reduction.
- Once we understand our data clearly and optimize it, we wish to model our data to predict the target variable i.e final placement of a player with a good accuracy.
- Using the results of our model and our overall analysis, recommend strategies and techniques that can help a user stay ahead in the game.

3. Data Exploration

For any successful analysis, it is very crucial to understand what our data looks like, identify and be able to explain the meanings of the variables, clean/pre-process the data before it can be modeled and visualize how the data is spread. In this section, we attempt to explore our 65000 matches worth of PUBG dataset.

3.1. DATA DESCRIPTION

The input file type of our dataset is a simple file format stored in a tabular format of Comma Separated Values(.csv) We have two different datasets:

- *Training Dataset:*
  - Contains 4446965 observations with a total of 29 columns.
  - This dataset will be used to train our models and evaluate the accuracy of our prediction.
  - Size - 668.2 MB

- *Testing Dataset:*
  - Contains 1934174 observations with a total of 28 columns(excludes the target column).
  - This dataset is given by Kaggle so as to use our model to calculate the target variable.
  - Size - 276.7 MB

```r
```{r}
library(corrplot)
library(gplots)
library(FNN)
library(caret)
library(dummies)
library(rpart)
library(gains)
train <- read.csv('train_V2.csv',stringsAsFactors = F)
test <- read.csv('test_V2.csv',stringsAsFactors = F)
```
```

Let's summarize and find out how our dataset looks like:

```{r}
str(train)
summary(train)
sapply(train,class)
```

```
'data.frame':   4446966 obs. of  29 variables:
 $ Id             : chr  "7f96b2f878858a" "eef90569b9d03c" "1eaf90ac73de72" "4616d365dd2853" ...
 $ groupId        : chr  "4d4b580de459be" "684d5656442f9e" "6a4a42c3245a74" "a930a9c79cd721" ...
 $ matchId        : chr  "a10357fd1a4a91" "aeb375fc57110c" "110163d8bb94ae" "f1f1f4ef412d7e" ...
 $ assists        : int  0 0 1 0 0 0 0 0 0 0 ...
 $ boosts         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ damageDealt    : num  0 91.5 68 32.9 100 ...
 $ DBNOs          : int  0 0 0 0 0 1 0 0 0 0 ...
 $ headshotKills  : int  0 0 0 0 0 1 0 0 0 0 ...
 $ heals          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ killPlace      : int  60 57 47 75 45 44 96 48 64 74 ...
 $ killPoints     : int  1241 0 0 0 0 0 1262 1000 0 0 ...
 $ kills          : int  0 0 0 0 1 1 0 0 0 0 ...
 $ killStreaks    : int  0 0 0 0 1 1 0 0 0 0 ...
 $ longestKill    : num  0 0 0 0 58.5 ...
 $ matchDuration  : int  1306 1777 1318 1436 1424 1395 1316 1967 1375 1930 ...
 $ matchType      : chr  "squad-fpp" "squad-fpp" "duo" "squad-fpp" ...
 $ maxPlace       : int  28 26 50 31 97 28 28 96 28 29 ...
 $ numGroups      : int  26 25 47 30 95 28 28 92 27 27 ...
 $ rankPoints     : int  -1 1484 1491 1408 1560 1418 -1 -1 1493 1349 ...
 $ revives        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ rideDistance   : num  0 0.0045 0 0 0 ...
 $ roadKills      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ swimDistance   : num  0 11 0 0 0 ...
 $ teamKills      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ vehicleDestroys: int  0 0 0 0 0 0 0 0 0 0 ...
 $ walkDistance   : num  244.8 1434 161.8 202.7 49.8 ...
 $ weaponsAcquired: int  1 5 2 3 2 1 1 6 4 1 ...
 $ winPoints      : int  1466 0 0 0 0 0 1497 1500 0 0 ...
 $ winPlacePerc   : num  0.444 0.64 0.775 0.167 0.188 ...
```

```
            Id         groupId         matchId         assists           boosts     damageDealt
   "character"     "character"     "character"       "integer"       "integer"       "numeric"
         DBNOs   headshotKills           heals       killPlace      killPoints           kills
     "integer"       "integer"       "integer"       "integer"       "integer"       "integer"
   killStreaks     longestKill   matchDuration       matchType       maxPlace       numGroups
     "integer"       "numeric"       "integer"     "character"       "integer"       "integer"
    rankPoints         revives    rideDistance       roadKills    swimDistance       teamKills
     "integer"       "integer"       "numeric"       "integer"       "numeric"       "integer"
vehicleDestroys    walkDistance weaponsAcquired       winPoints    winPlacePerc
     "integer"       "numeric"       "integer"       "integer"       "numeric"
```

FIG-1 DATA DESCRIPTION

From the output in FIG-1, we infer the following information:

- Id, groupId and matchId represent identifier variables and hence they will not play any role in our model. They can be ignored in any analysis.

- We have one categorical variable : matchType. It will not contribute in any significance to the prediction model.

- We have 25 numerical variables, out of which, one is our target variable : winPlacePerc. The value of the target variable ranges from 1(highest) to 0(lowest).

```
     Id                groupId           matchId            assists           boosts
Length:4446966     Length:4446966     Length:4446966     Min.   : 0.0000   Min.   : 0.000
Class :character   Class :character   Class :character   1st Qu.: 0.0000   1st Qu.: 0.000
Mode  :character   Mode  :character   Mode  :character   Median : 0.0000   Median : 0.000
                                                         Mean   : 0.2338   Mean   : 1.107
                                                         3rd Qu.: 0.0000   3rd Qu.: 2.000
                                                         Max.   :22.0000   Max.   :33.000

 damageDealt          DBNOs         headshotKills        heals         killPlace       killPoints
Min.   :   0.00   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.00   Min.   :  1.0   Min.   :   0
1st Qu.:   0.00   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.00   1st Qu.: 24.0   1st Qu.:   0
Median :  84.24   Median : 0.0000   Median : 0.0000   Median : 0.00   Median : 47.0   Median :   0
Mean   : 130.72   Mean   : 0.6579   Mean   : 0.2268   Mean   : 1.37   Mean   : 47.6   Mean   : 505
3rd Qu.: 186.00   3rd Qu.: 1.0000   3rd Qu.: 0.0000   3rd Qu.: 2.00   3rd Qu.: 71.0   3rd Qu.:1172
Max.   :6616.00   Max.   :53.0000   Max.   :64.0000   Max.   :80.00   Max.   :101.0   Max.   :2170

    kills          killStreaks      longestKill      matchDuration   matchType           maxPlace
Min.   : 0.0000   Min.   : 0.000   Min.   :   0.00   Min.   :   9   Length:4446966     Min.   :  1.0
1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.:   0.00   1st Qu.:1367   Class :character   1st Qu.: 28.0
Median : 0.0000   Median : 0.000   Median :   0.00   Median :1438   Mode  :character   Median : 30.0
Mean   : 0.9248   Mean   : 0.544   Mean   :  23.00   Mean   :1580                      Mean   : 44.5
3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.:  21.32   3rd Qu.:1851                      3rd Qu.: 49.0
Max.   :72.0000   Max.   :20.000   Max.   :1094.00   Max.   :2237                      Max.   :100.0

  numGroups        rankPoints       revives         rideDistance        roadKills
Min.   :  1.00   Min.   :  -1   Min.   : 0.0000   Min.   :    0.00   Min.   : 0.000000
1st Qu.: 27.00   1st Qu.:  -1   1st Qu.: 0.0000   1st Qu.:    0.00   1st Qu.: 0.000000
Median : 30.00   Median :1443   Median : 0.0000   Median :    0.00   Median : 0.000000
Mean   : 43.01   Mean   : 892   Mean   : 0.1647   Mean   :  606.12   Mean   : 0.003496
3rd Qu.: 47.00   3rd Qu.:1500   3rd Qu.: 0.0000   3rd Qu.:    0.19   3rd Qu.: 0.000000
Max.   :100.00   Max.   :5910   Max.   :39.0000   Max.   :40710.00   Max.   :18.000000

 swimDistance        teamKills        vehicleDestroys     walkDistance     weaponsAcquired
Min.   :   0.000   Min.   : 0.00000   Min.   :0.000000   Min.   :    0.0   Min.   :  0.00
1st Qu.:   0.000   1st Qu.: 0.00000   1st Qu.:0.000000   1st Qu.:  155.1   1st Qu.:  2.00
Median :   0.000   Median : 0.00000   Median :0.000000   Median :  685.6   Median :  3.00
Mean   :   4.509   Mean   : 0.02387   Mean   :0.007918   Mean   : 1154.2   Mean   :  3.66
3rd Qu.:   0.000   3rd Qu.: 0.00000   3rd Qu.:0.000000   3rd Qu.: 1976.0   3rd Qu.:  5.00
Max.   :3823.000   Max.   :12.00000   Max.   :5.000000   Max.   :25780.0   Max.   :236.00

  winPoints       winPlacePerc
Min.   :   0.0   Min.   :0.0000
1st Qu.:   0.0   1st Qu.:0.2000
Median :   0.0   Median :0.4583
Mean   : 606.5   Mean   :0.4728
3rd Qu.:1495.0   3rd Qu.:0.7407
Max.   :2013.0   Max.   :1.0000
                 NA's   :1
```

FIG-2 SUMMARY STATISTICS

From the descriptive statistics seen in FIG-2, we can observe how each variable is distributed by it's 5-point summary. For example, we see that the target variable winPlacePerc has a mean value of 0.4728 and a median value of 0.4583. Since mean and median are almost close to each other, we can say that the entire data is evenly distributed around the mean, meaning there is not much skewness in the data. Also, we observe that winPlacePerc has a missing value in one row, which we will have to handle.

Here's the description of the data fields:

- **DBNOs** - Number of enemy players knocked.
- **assists** - Number of enemy players this player damaged that were killed by teammates.
- **boosts** - Number of boost items used.
- **damageDealt** - Total damage dealt.
- **headshotKills** - Number of enemy players killed with headshots.
- **heals** - Number of healing items used.
- **Id** - Player's Id.
- **killPlace** - Ranking in match of number of enemy players killed.
- **killPoints** - Kills-based external ranking of player.
- **killStreaks** - Max number of enemy players killed in a short amount of time.
- **kills** - Number of enemy players killed.
- **longestKill** - Longest distance between player and player killed at time of death.
- **matchDuration** - Duration of match in seconds.
- **matchId** - ID to identify match.
- **matchType** - String identifying the game mode that the data comes from. The standard modes are "solo", "duo", "squad", "solo-fpp", "duo-fpp", and "squad-fpp"; other modes are from events or custom matches.
- **rankPoints** - Elo-like ranking of player.
- **revives** - Number of times this player revived teammates.
- **rideDistance** - Total distance traveled in vehicles measured in meters.
- **roadKills** - Number of kills while in a vehicle.
- **swimDistance** - Total distance traveled by swimming measured in meters.
- **teamKills** - Number of times this player killed a teammate.
- **vehicleDestroys** - Number of vehicles destroyed.
- **walkDistance** - Total distance traveled on foot measured in meters.
- **weaponsAcquired** - Number of weapons picked up.
- **winPoints** - Win-based external ranking of player.
- **groupId** - ID to identify a group within a match. If the same group of players plays in different matches, they will have a different groupId each time.
- **numGroups** - Number of groups we have data for in the match.
- **maxPlace** - Worst placement we have data for in the match.

- **winPlacePerc** - The target of prediction. This is a percentile winning placement, where 1 corresponds to 1st place, and 0 corresponds to last place in the match.

3.2. MISSING VALUES AND UNIQUENESS

3.2.1. Missing Values

```{r}
library(mice)
input <- c("groupId", "matchId", "assists", "boosts",
           "damageDealt", "DBNOs", "headshotKills", "heals",
           "killPlace", "killPoints", "kills", "killStreaks",
           "longestKill", "matchDuration", "matchType",
           "maxPlace", "numGroups", "rankPoints", "revives",
           "rideDistance", "roadKills", "swimDistance",
           "teamKills", "vehicleDestroys", "walkDistance",
           "weaponsAcquired", "winPoints")
target <- "winPlacePerc"
# Generate a summary of the missing values in the dataset.
md.pattern(train[,c(input, target)], plot = FALSE)
```

```
        groupId matchId assists boosts damageDealt DBNOs headshotKills heals killPlace
4446965       1       1       1      1           1     1             1     1         1
1             1       1       1      1           1     1             1     1         1
              0       0       0      0           0     0             0     0         0
        killPoints kills killStreaks longestKill matchDuration matchType maxPlace numGroups
4446965          1     1           1           1             1         1        1         1
1                1     1           1           1             1         1        1         1
                 0     0           0           0             0         0        0         0
        rankPoints revives rideDistance roadKills swimDistance teamKills vehicleDestroys
4446965          1       1            1         1            1         1                1
1                1       1            1         1            1         1                1
                 0       0            0         0            0         0                0
        walkDistance weaponsAcquired winPoints winPlacePerc
4446965            1               1         1            1 0
1                  1               1         1            0 1
                   0               0         0            1 1
```

FIG-3 IDENTIFYING MISSING VALUES

We will have to omit the row which has an NA value for winPlacePerc, as seen in output of FIG-3. This we will handle in our preprocess step.

3.2.2. Uniqueness

```r
library(magrittr)
unq_val <- sapply(train, function(x) length(unique(x)))
unq_val[unq_val == 1] %>% sort(decreasing = T)
```

```
named integer(0)
```

FIG-4 VERIFYING UNIQUENESS

From the output in FIG-4, we see that there are no columns with constant variables. If there were, we would have to remove them in the pre-process.

3.3. PREPROCESSING

```r
test$winPlacePerc = NA
train <- na.omit(train)
# Remove IDs
train <- train[,-c(1:3)]
train$matchType <- as.factor(train$matchType)
```

We perform three steps here:

- Remove rows having missing values.
- Remove the columns which are identifier variables.
- Factor the categorical variable ‘matchType’ so that it isn’t used up in the modeling part.

3.4. CORRELATION ANALYSIS

FIG-5 shows the correlation matrix for 24 variables. The colour of the box indicates how strongly the variables are correlated by using colours. White sky blue to dark blue indicates positive relationship between the variables. A high positive correlation is indicated by dark blue colour whereas white sky blue indicates a weak positive correlation. Similarly, white to red indicates negative correlation between variable. Red indicating strong negative correlation and white peach light indicating weak negative relationship.

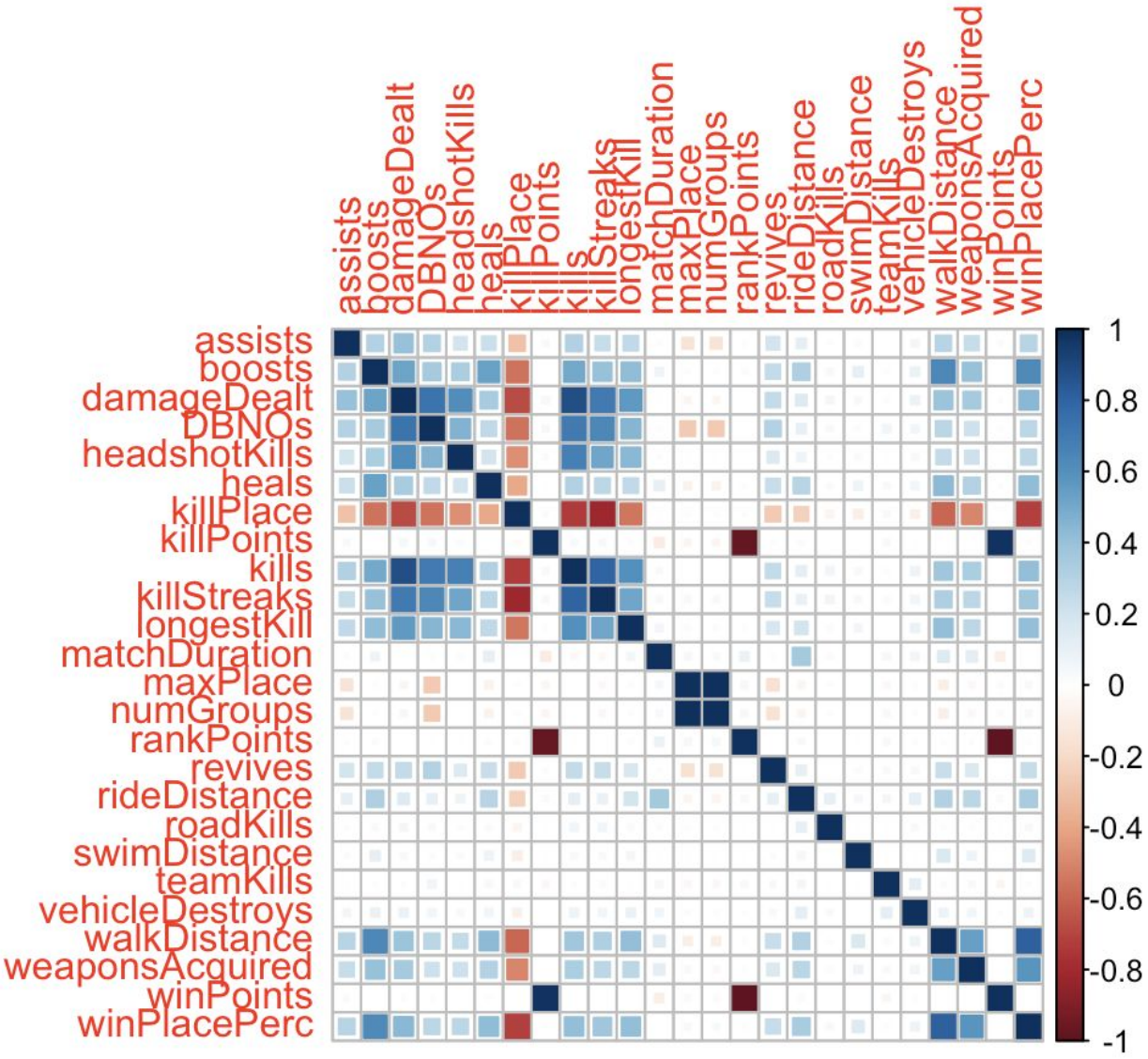


FIG-5 CORRELATION MATRIX

From the correlation plot, we observe high dependence between the following:

Variable 1	**Variable 2**	**Correlation Value**	**Correlation Type**
numGroups	maxPlace	0.9979	Very strong, Positive
kills	killStreaks	0.8025	Strong, Positive
killStreaks	killPlace	-0.8105	Strong, Negative
kills	damageDealt	0.8887	Strong, Positive

On plot these strong relationships, we can infer their dependence:

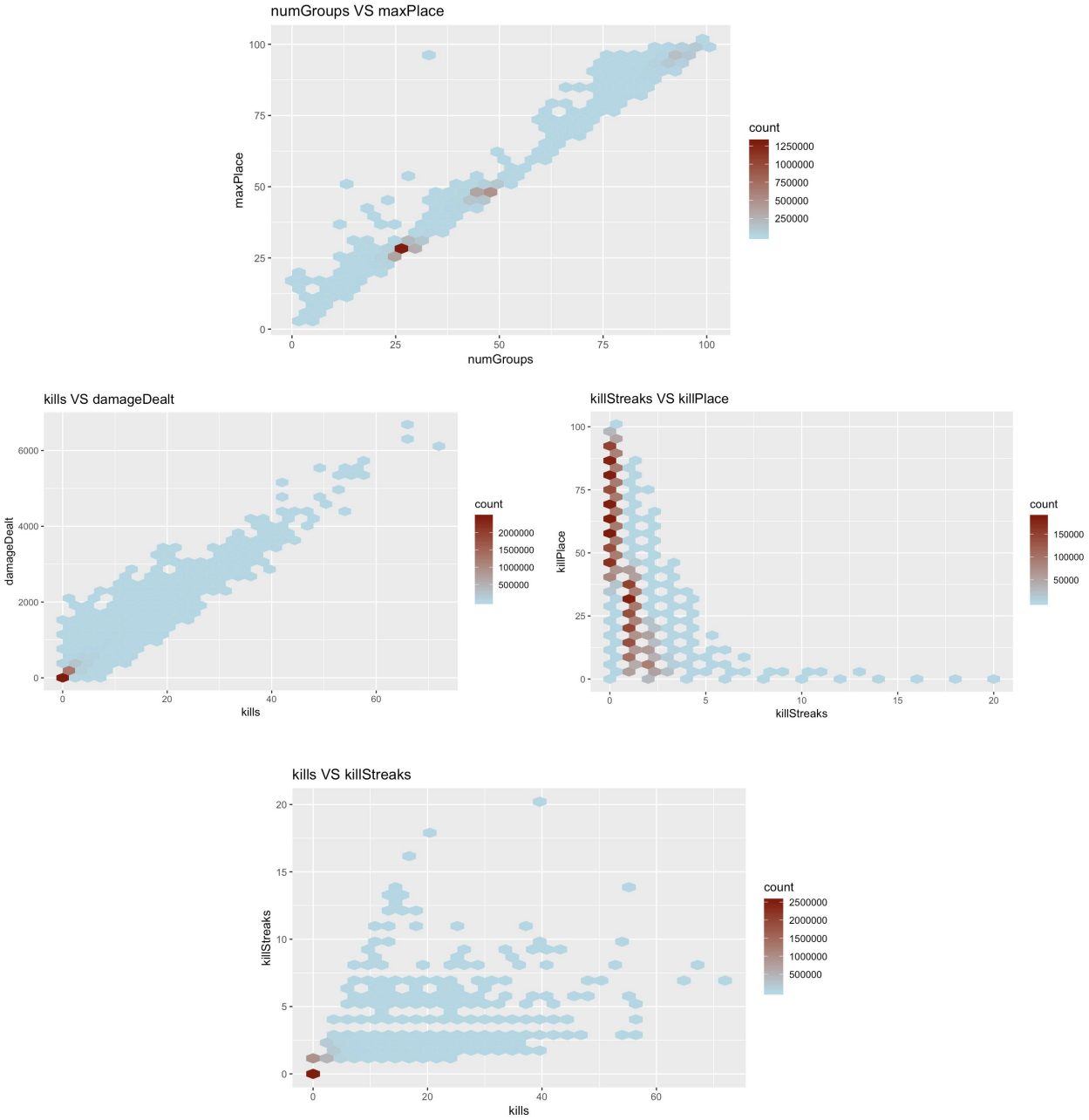


FIG-6 VARIABLE TO VARIABLE DEPENDENCE

1. The number of groups in any game always matches with the worst placement in a match. That is why we have the strongest positive correlation and hence in developing models, any one of the variables can be dropped.

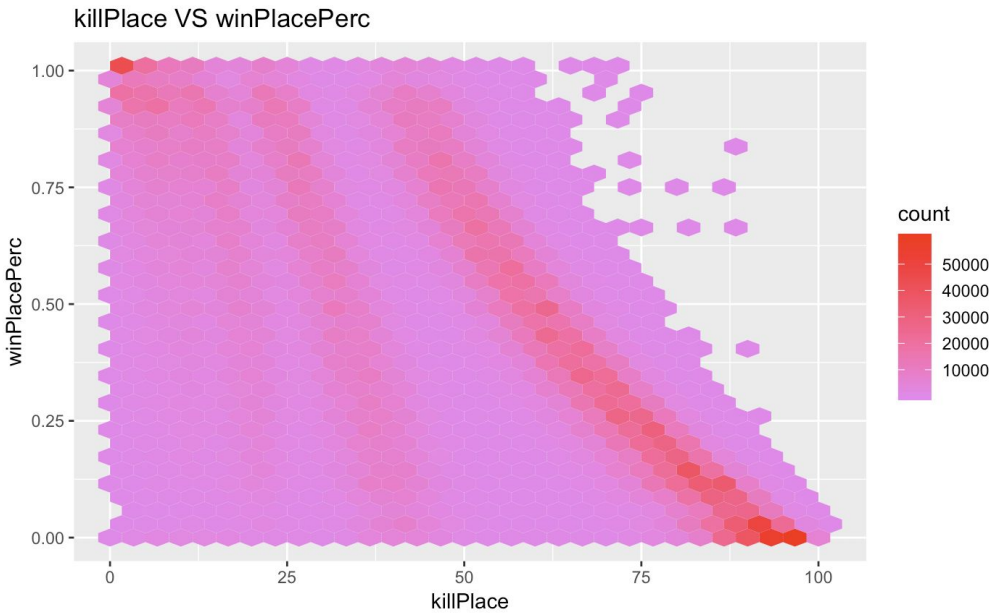
2. As the player kills more enemies, the total damage he has done to opponents increases. Hence, it is a direct inference that the more opponents you kill, the more damage you inflict. Anything above 2000 is considered a good score of damageDealt and we can infer that the corresponding x-value is 20. So, a player should aim to make at least 20 kills in a game.

3. As a player kills more enemies, his ranking increases. For zero kills, it's ideally Rank 100 and as the killStreaks increase, the player moves to up the ladder and hence the negative correlation. We see that it is common to have killStreaks ranging from 0-4 corresponding to a killPlace of the bottom 75 percentile. A Player with >4 killStreaks has a good chance to get better ranking.

4. When a player's killStreaks improve, obviously the kill count improves as well. Majority of the data is placed in the lower end indicating that they died very early in the game without making any kills and with no kills, there's obviously no kill streaks.

In specific to our target variable, we find some interesting strong correlations :

Target Variable	**Variable**	**Correlation Value**	**Correlation Type**
winPlacePerc	killPlace	-0.7191	Strong, Negative
winPlacePerc	walkDistance	0.8025	Strong, Positive
winPlacePerc	boosts	0.6342	Good, Positive



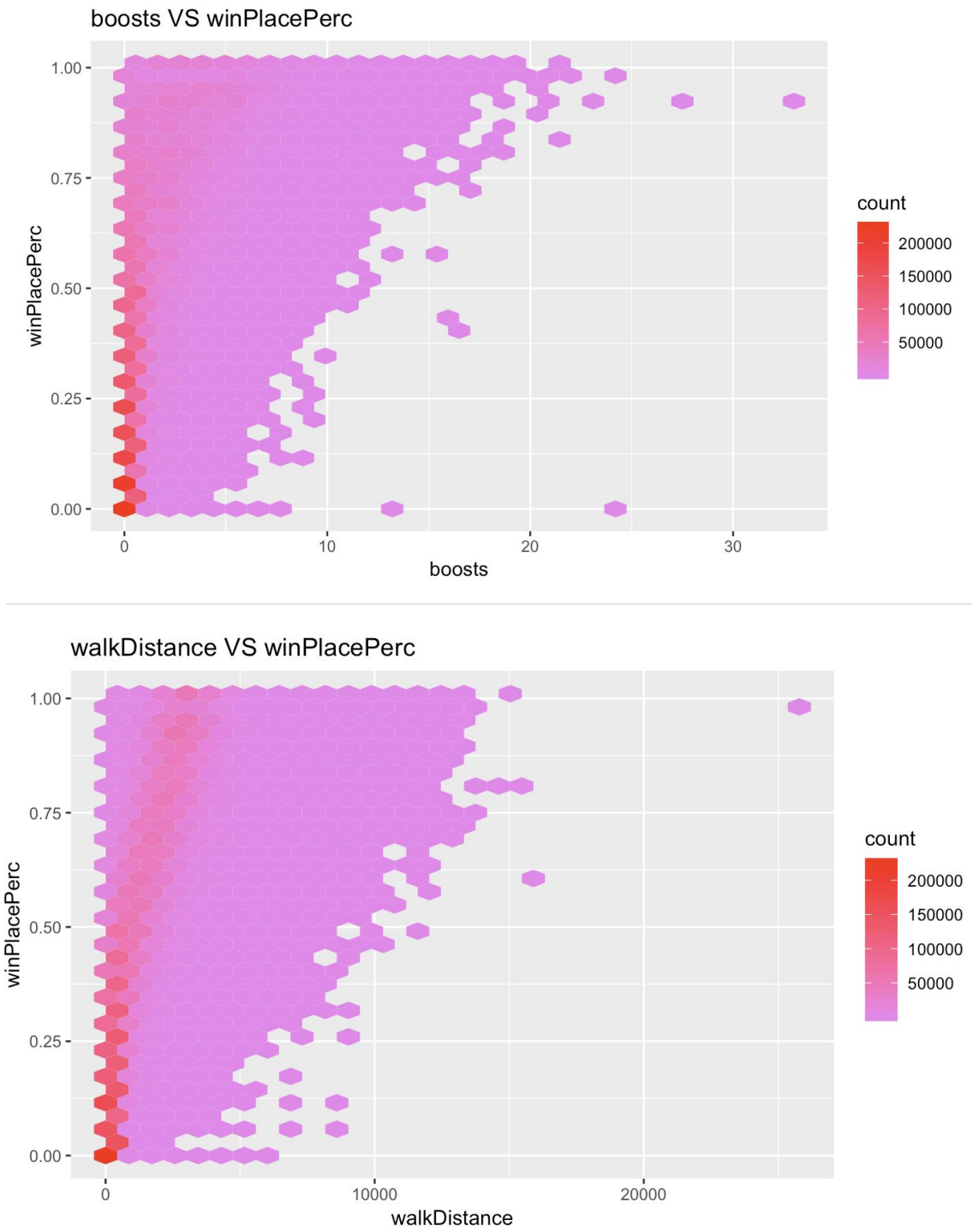


FIG-7 CORRELATION WITH TARGET VARIABLE

1. We can easily see that the ranking in a match by number of enemy players killed is a basic way to assess a player's skill level. If a player is new, he will not have a high rank and hence his

winPlacePerc will be low. For example, for the top 25 players(x<25), winPlacePerc will always be >0.8, whereas the bottom 25 players can never have a winPlacePerc >0.5

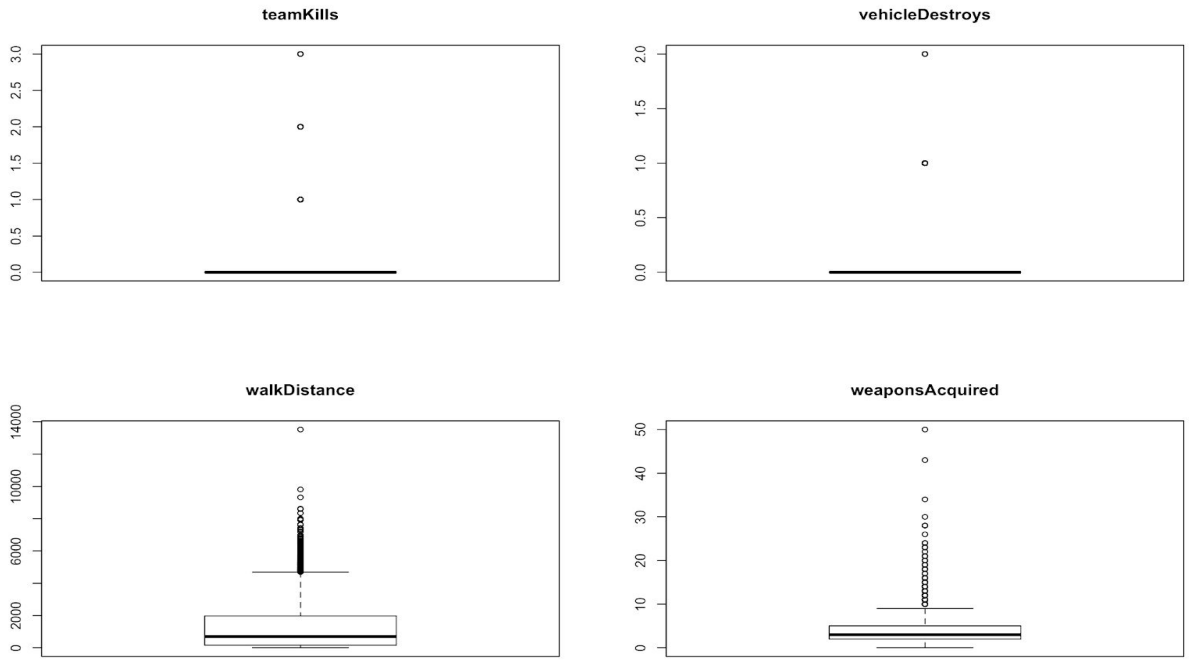
2. To increase the winPlacePerc, a person should consume as many boosts as possible. More the number of boosts used in the match, more is the winPlacePerc value. According to the graph, players who have taken at least 20 boosts have almost reached winPlacePerc value of 1.

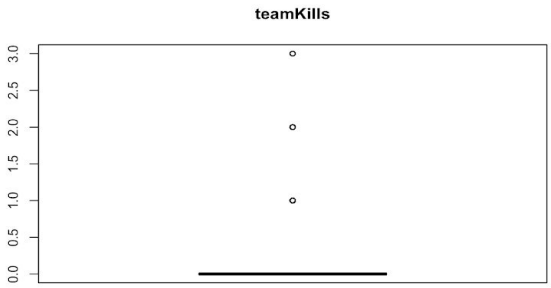
3. The most important observation in our correlation analysis is the WalkDistance. The graph clearly shows how walking more distance can easily improve your chances of getting a higher winPlacePerc (>0.8)

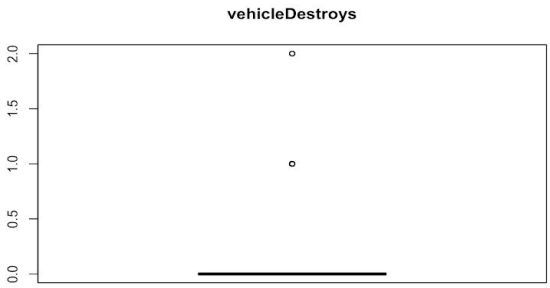
3.5. DATA VISUALIZATIONS

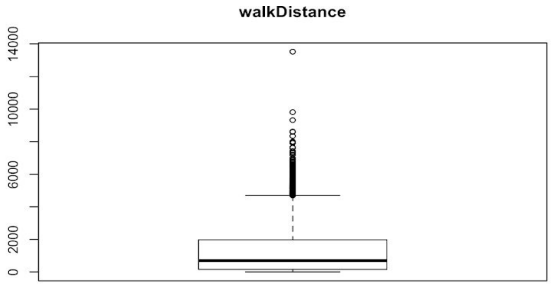
3.5.1. BOX PLOTS

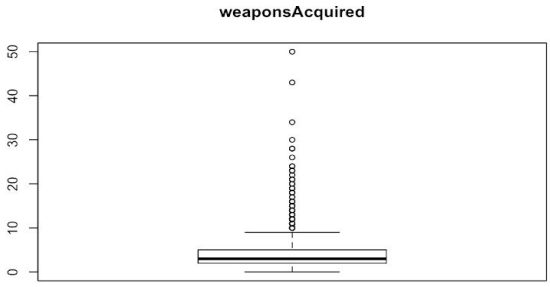
Boxplots helped us to measure how well the data is distributed in our PUBG data set since it divides the data set into three quartiles. Below attached are graphs created using boxplot function representing the minimum, maximum, median, first quartile and third quartile and any outliers in the data set. It was useful in comparing the distribution of data across by drawing boxplots for each of them. We were able to judge that that variable has high level of agreement by seeing that the boxplot is comparatively small. A taller box plot indicated higher difference between groups.

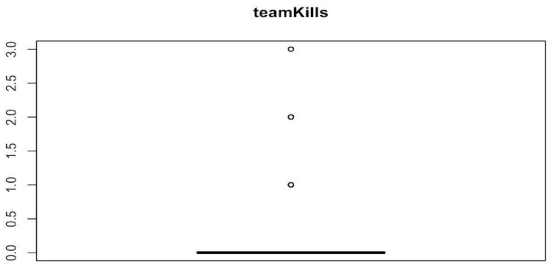


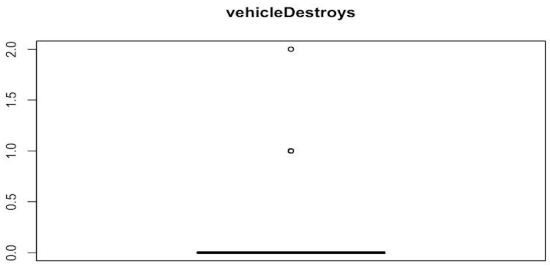

teamKills
0.0
0.5
1.0
1.5
2.0
2.5
3.0

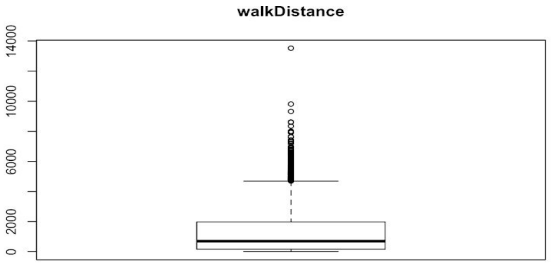

vehicleDestroys
0.0
0.5
1.0
1.5
2.0

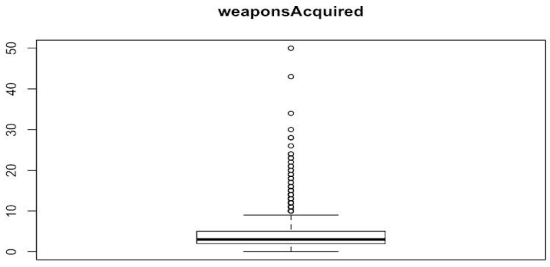

walkDistance
0
2000
6000
10000
14000


weaponsAcquired
0
10
20
30
40
50


teamKills
0.0
0.5
1.0
1.5
2.0
2.5
3.0


vehicleDestroys
0.0
0.5
1.0
1.5
2.0


walkDistance
0
2000
6000
10000
14000


weaponsAcquired
0
10
20
30
40
50

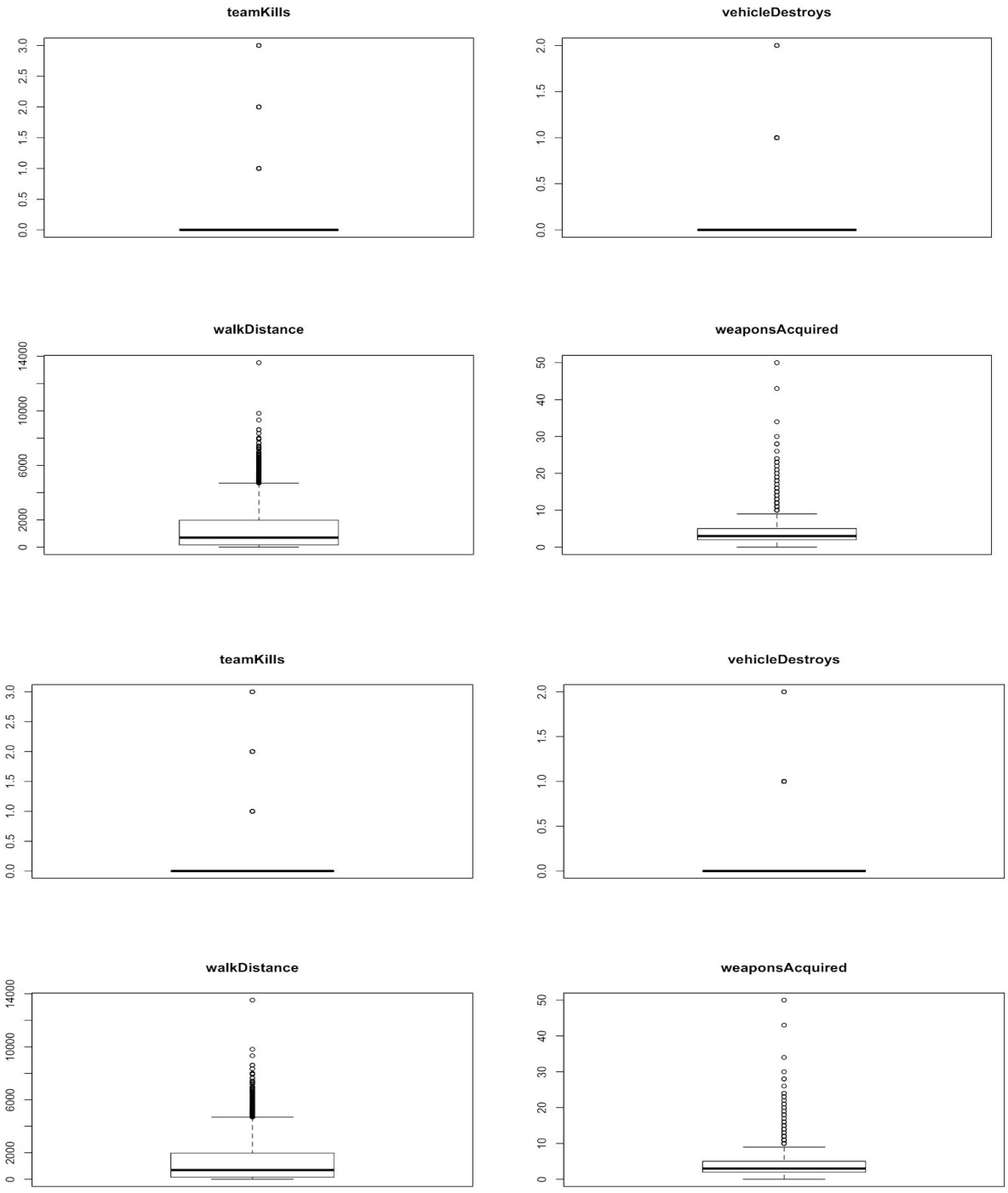


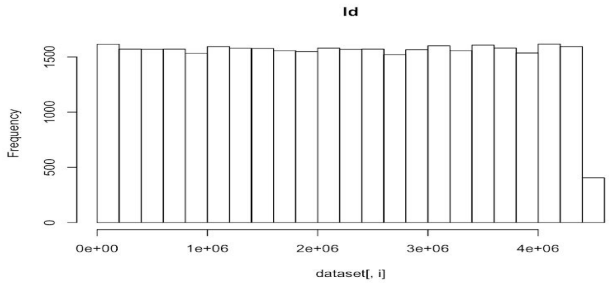
FIG-8 BOX PLOTS DISTRIBUTIONS

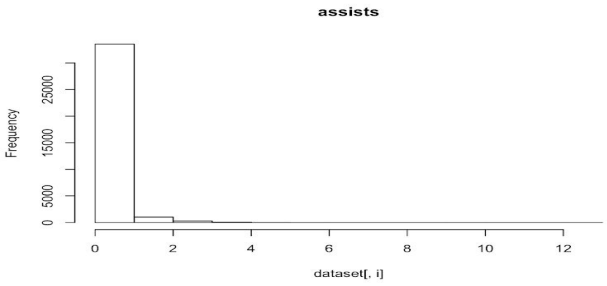
3.5.2. HISTOGRAMS

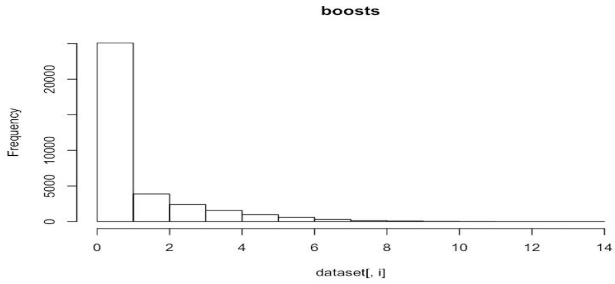
Plotting histogram helped us to visually represent and get us familiar with the dataset. As such, the shape of a histogram is the most obvious and informative characteristic: it allowed us to easily see where a relatively large amount of the data is situated and where there is very little data to be found. In other words, we were able to see where the middle is in your data distribution, how close the data lie around this middle and where possible outliers are to be found. The y-axis shows how frequently the variables in x-axis occur in the data, while the bars group ranges of values on the x-axis.

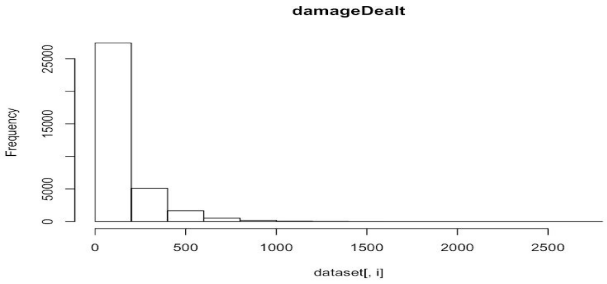
Important observations :

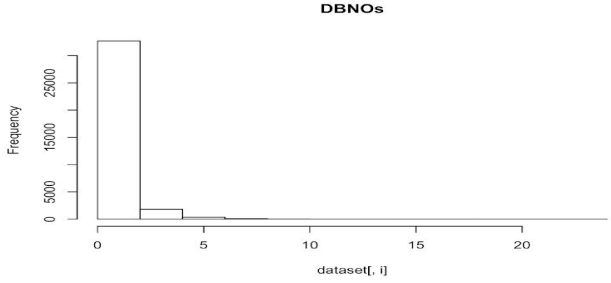
- Since this game is played in teams of 2 or 4 we observe that the variable numgroup mainly lies between 20 to 50 indicating the mode where 2 player team is involved and after a significant gap between 75-100 indicating the mode where game is played in 4 player team.
- From the distributions we see not every game was full. We think Walk Distance should be an important indicator of overall Win Place. The walking/running speed is relatively slow, so you would need to be alive for longer to achieve a higher value here. In PUBG in the early game <2-3 minutes there will be a lot of deaths as players in the more heavily populated areas such as School and Bootcamp battle it out. The distribution of walk distance also seems to back this up as there a lot of lowe values here. There does seem to be some major outliers here, e.g. Is it reasonable to expect a player to run 26km in around 30 minutes? We will need to tidy these values up as they are obviously fouled somehow.
- We also see the expected clumps in the number of groups denoting the game mode, e.g. Squad, Duo and Solo modes. The values less than 10 probably warrant further investigation as it is most likely these are custom games or disconnect errors.
- We should look at the spikes in killPlace as these look odd and seem evenly spaced.
- Boosts are also an important factor in winning a game, from experience it is unlikely you place highly without consuming some kind of boost.
- Judging from the distribution of number of players, the game mode seems to be ‘classic’ as it has 100 initial players. It is unlikely to be Arcade mode because the number of players in Arcade mode is 28 and a miserly number of games with 28 players are present in the dataset.
- Most games are either played solo or in teams of two.
- Number of groups tend to be around 25, 50, or 100. That is, groups of 4, 2, and 1 respectively.
- Number of distinct groupIds cannot exceed:100 incase of solo mode,50 incase of duo mode,25 incase of 4 players in a team,100/n in case of n players in team.

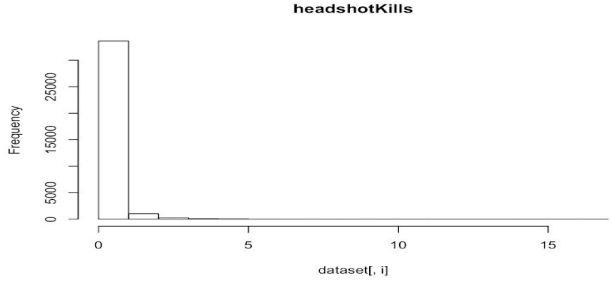

Id
Frequency
0
500
1000
1500
0e+00
1e+06
2e+06
3e+06
4e+06
dataset[, i]

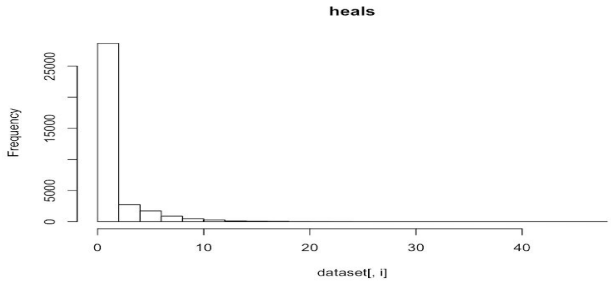

assists
Frequency
0
5000
15000
25000
0
2
4
6
8
10
12
dataset[, i]

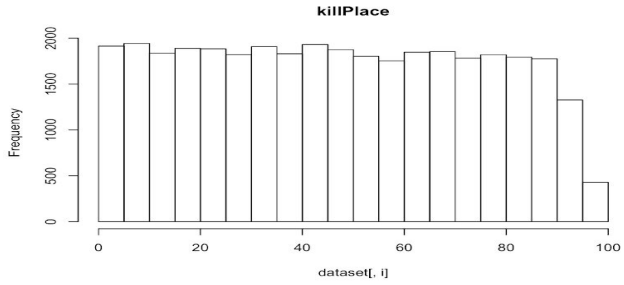

boosts
Frequency
0
5000
10000
20000
0
2
4
6
8
10
12
14
dataset[, i]

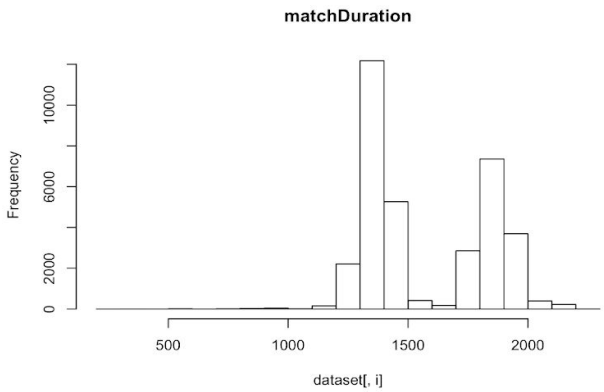

damageDealt
Frequency
0
5000
15000
25000
0
500
1000
1500
2000
2500
dataset[, i]

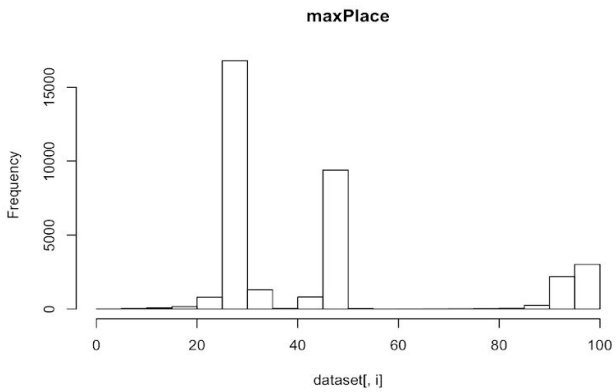

DBNOs
Frequency
0
5000
15000
25000
0
5
10
15
20
dataset[, i]

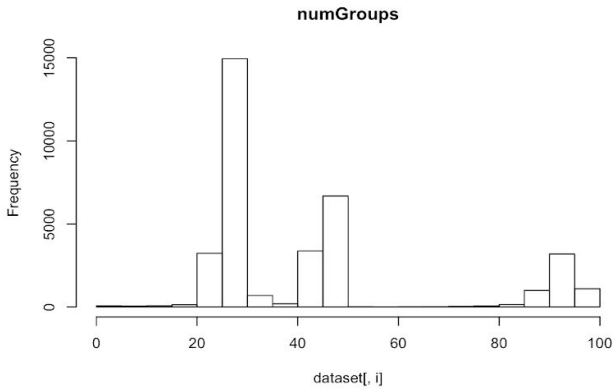

headshotKills
Frequency
0
5000
15000
25000
0
5
10
15
dataset[, i]

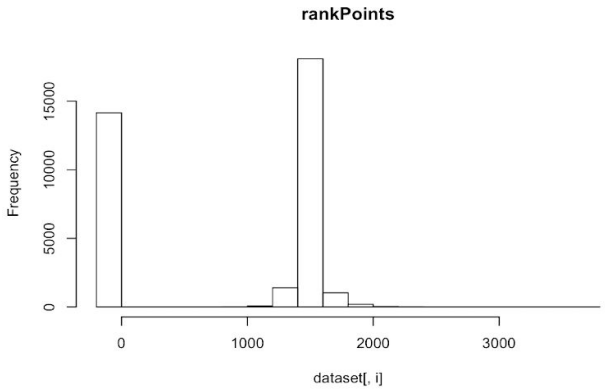

heals
Frequency
0
5000
15000
25000
0
10
20
30
40
dataset[, i]

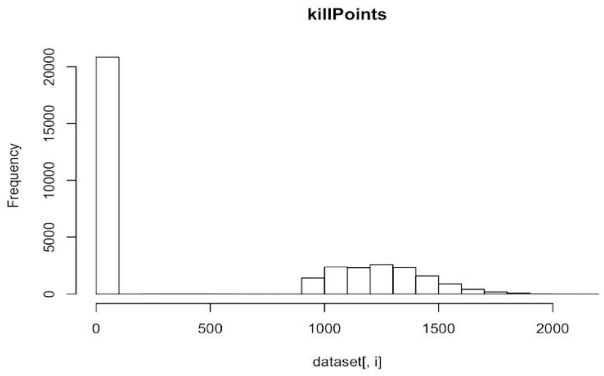

killPlace
Frequency
0
500
1000
1500
2000
0
20
40
60
80
100
dataset[, i]

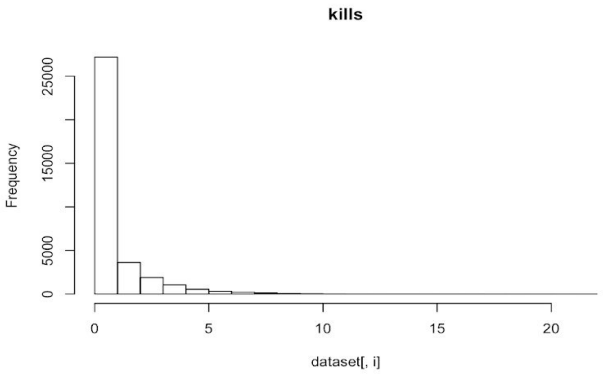

matchDuration
Frequency
0
2000
6000
10000
500
1000
1500
2000
dataset[, i]

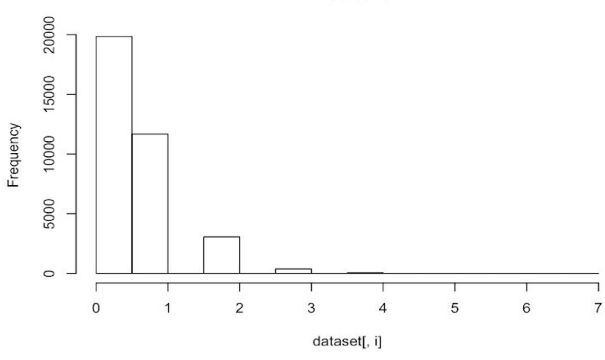

maxPlace
Frequency
0
5000
10000
15000
0
20
40
60
80
100
dataset[, i]

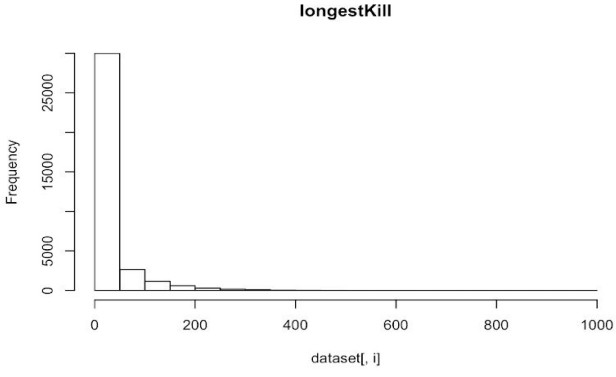

numGroups
Frequency
0
5000
10000
15000
0
20
40
60
80
100
dataset[, i]


rankPoints
Frequency
0
5000
10000
15000
0
1000
2000
3000
dataset[, i]


killPoints
Frequency
0
5000
10000
15000
20000
0
500
1000
1500
2000
dataset[, i]


kills
Frequency
0
5000
15000
25000
0
5
10
15
20
dataset[, i]


killStreaks
Frequency
0
5000
10000
15000
20000
0
1
2
3
4
5
6
7
dataset[, i]


longestKill
Frequency
0
5000
15000
25000
0
200
400
600
800
1000
dataset[, i]

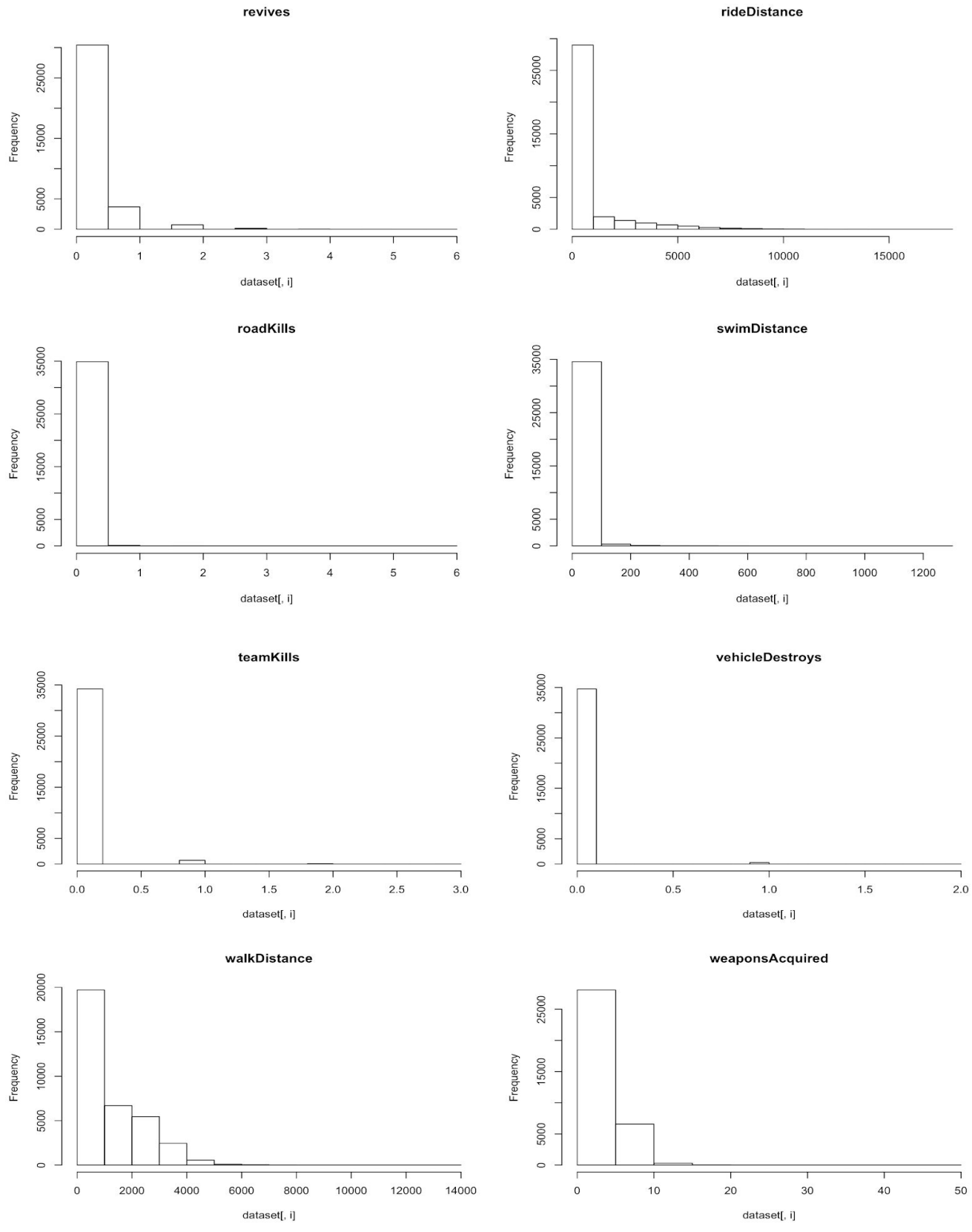


FIG-8 HISTOGRAM DISTRIBUTIONS

3.6. DIMENSIONALITY REDUCTION ANALYSIS

In any analysis, it is important to consider the order of complexity in which an algorithm works. Our correlation analysis gave us insights on the dependent variables, which essentially mean that few variables can be expressed as a linear combination of their dependents while modeling our data. In this analysis, we aim to reduce the number of random variables that we choose for analysis by exploring Principal Component Analysis Method(PCA) to select our features.

PRINCIPAL COMPONENT ANALYSIS

Here, we perform a linear mapping of our PUBG data to a lower-dimensional space where the variance of our data is maximized. This will create individual vectors and components with each containing the linear combination of all variables and their contributions.

```r
```{r}
library(tidyverse)
library(factoextra)

pca_model <- prcomp(train[,-c(13,26)], scale = TRUE)
summary(pca_model)

fviz_pca_var(pca_model,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
)

pubg.result.var <- get_pca_var(pca_model)
pubg_contrib <- round(pubg.result.var$contrib,2)
```
```

On running PCA, we get the summary to be

```
Importance of components:
                          PC1    PC2     PC3     PC4    PC5     PC6    PC7     PC8     PC9
PC10    PC11
Standard deviation     2.5097 1.7339 1.45096 1.35910 1.0777 1.01577 0.9968 0.97534 0.94724
0.92233 0.88875
Proportion of Variance 0.2624 0.1253 0.08772 0.07696 0.0484 0.04299 0.0414 0.03964 0.03739
0.03545 0.03291
Cumulative Proportion  0.2624 0.3877 0.47542 0.55238 0.6008 0.64377 0.6852 0.72481 0.76219
0.79764 0.83055
                          PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19    PC20
PC21
Standard deviation     0.85677 0.76544 0.75443 0.75174 0.67738 0.63322 0.55086 0.45856 0.35456
0.2857
Proportion of Variance 0.03059 0.02441 0.02371 0.02355 0.01912 0.01671 0.01264 0.00876 0.00524
0.0034
Cumulative Proportion  0.86114 0.88555 0.90926 0.93281 0.95193 0.96864 0.98128 0.99004 0.99528
0.9987
                          PC22    PC23    PC24
Standard deviation     0.15754 0.06931 0.04506
Proportion of Variance 0.00103 0.00020 0.00008
Cumulative Proportion  0.99972 0.99992 1.00000
```

FIG-9 PRINCIPAL COMPONENT ANALYSIS SUMMARY

We see the cumulative proportion to exactly determine how much accuracy we want in expressing our data to fully explain the variance. We observe that PC1 explains about 26.24% of the variations and PC2 explain 12.53% of the variations. To explain at least 90% of the variations, we will have to consider components upto PC14 which from FIG-9 explains about 90.93% of the variations. Hence in building our model, we will choose components upto PC14 and drop the rest to reduce dimensions.

Now that we have explained the PCs, it is useful to understand which features contribute more to the analysis. For this purpose, let's consider PC1 and PC2. Consider the plot below,

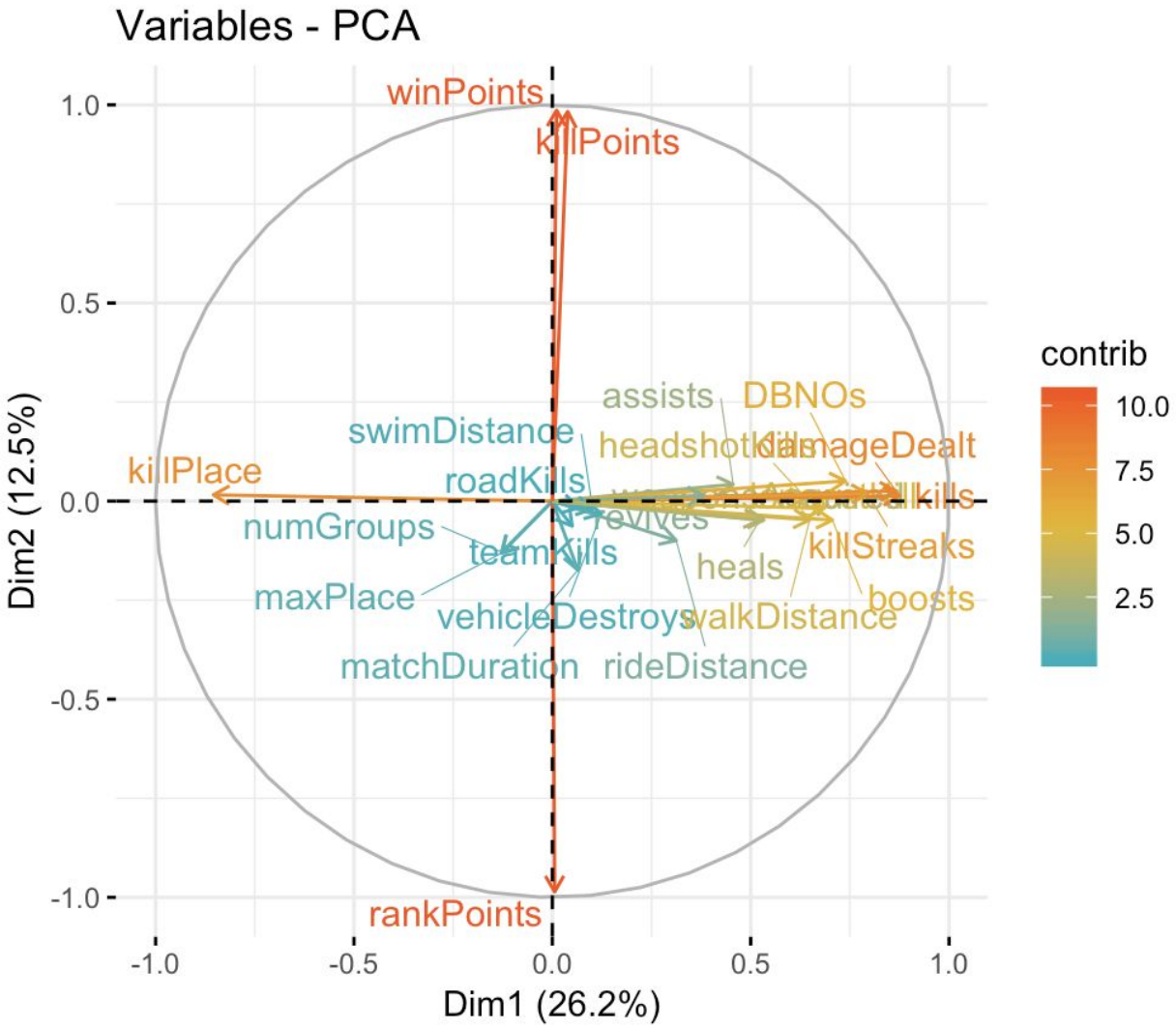


FIG-10 PLOT OF FEATURE CONTRIBUTION TO PC1 & PC2

From FIG-10, we observe that whichever feature has the longest arrow towards the perimeter of the circle, it will contribute to the PCs more significantly. For PC1, we observe that kills, damageDealt, killPlace and killStreaks have a lot of importance. All the 4 statistic measures deal with a player's ability to kill. So, we can note that kills in the game will be a feature that will contribute a lot in our final analysis. Their contribution is explained by their Dim.1 values as shown in FIG-11

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10	Dim.11
kills	12.15	0.01	1.09	5.36	0.57	0.13	0.00	0.42	0.01	0.03	0.00
damageDealt	11.91	0.02	0.40	3.44	0.27	0.12	0.05	0.02	0.10	0.03	1.64
killPlace	11.59	0.01	0.62	0.01	0.15	0.03	0.23	0.07	0.27	1.21	3.74
killStreaks	10.02	0.01	0.58	5.25	0.71	0.10	0.20	0.60	0.31	0.03	2.05

FIG-11 TOP 4 CONTRIBUTORS TO PC1

Similarly, we can see from the plot and in FIG-11 that all the in-game stats like winPoints, rankPoints and killPoints contribute to a great extent to explain the variations in Dim.2. This is a good inference because in any game, player points are of utmost importance and that's exactly what we've shown!

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10	Dim.11
winPoints	0.00	32.35	0.45	0.63	0.10	0.00	0.01	0.02	0.02	0.00	0.00
rankPoints	0.00	32.34	0.28	0.57	0.11	0.00	0.01	0.03	0.02	0.00	0.00
killPoints	0.02	32.14	0.35	0.39	0.10	0.00	0.01	0.02	0.02	0.00	0.00

FIG-11 TOP 3 CONTRIBUTORS TO PC2

4. Predictive/Classification Algorithm

Upon understanding our data, we identified that we had a prediction problem in hand. Our goal was to fit a model to predict final placement winPlacePerc. To do this, we have considered three algorithms:

- Linear Regression.
- K-nearest neighbours(KNN).
- Classification and Regression Trees(CART).

4.1. LINEAR REGRESSION

Linear regression is a linear approach to modelling the relationship between the response and one or more predictor variables. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data.

Why did we chose this model?

In our project, we have used the linear regression model to predict the response variable - winPlacePerc. It is one of the best ways to predict target variables that are continuous and have correlations associated to it. The following code was used to fit the model.

```
111 ```{r}
112 # Fit Linear Reg Model
113 linear.reg <- lm(winPlacePerc ~.,data = new_train)
114
115
116 summary(linear.reg)
117 ```
```

The summary of the model tells us how the model has been fitted.

```
# Fit Linear Reg Model
linear.reg <- lm(winPlacePerc ~.,data = new_train)


summary(linear.reg)
```

```
Call:
lm(formula = winPlacePerc ~ ., data = new_train)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7336 -0.0685 -0.0022  0.0648  3.3162

Coefficients:
                          Estimate Std. Error   t value Pr(>|t|)
(Intercept)               4.131e-01  2.327e-03   177.524  < 2e-16 ***
assists                   1.536e-02  1.262e-04   121.704  < 2e-16 ***
boosts                    1.384e-02  5.814e-05   238.048  < 2e-16 ***
damageDealt               6.982e-05  9.471e-07    73.721  < 2e-16 ***
DBNOs                    -7.593e-03  9.645e-05   -78.716  < 2e-16 ***
headshotKills             2.082e-03  1.478e-04    14.082  < 2e-16 ***
heals                     5.804e-04  2.977e-05    19.497  < 2e-16 ***
killPlace                -7.444e-03  5.504e-06 -1352.427  < 2e-16 ***
killPoints               -5.139e-05  6.007e-07   -85.552  < 2e-16 ***
kills                    -1.289e-02  1.197e-04  -107.639  < 2e-16 ***
killStreaks              -1.482e-01  2.057e-04  -720.391  < 2e-16 ***
longestKill              -3.492e-07  1.684e-06    -0.207  0.83574
matchDuration            -1.605e-04  2.841e-07  -564.917  < 2e-16 ***
matchTypecrashtpp         1.962e-02  7.260e-03     2.702  0.00689 **
matchTypeduo              9.015e-02  1.798e-03    50.134  < 2e-16 ***
matchTypeduo-fpp          9.399e-02  1.786e-03    52.631  < 2e-16 ***
matchTypeflarefpp         1.519e-01  5.484e-03    27.702  < 2e-16 ***
matchTypeflaretpp         1.364e-01  3.301e-03    41.336  < 2e-16 ***
matchTypenormal-duo       2.284e-03  1.026e-02     0.223  0.82382
matchTypenormal-duo-fpp  -2.749e-04  2.613e-03    -0.105  0.91622
matchTypenormal-solo      1.585e-02  8.007e-03     1.980  0.04768 *
matchTypenormal-solo-fpp  4.080e-03  3.921e-03     1.041  0.29810
matchTypenormal-squad     1.568e-01  6.236e-03    25.140  < 2e-16 ***
matchTypenormal-squad-fpp 1.184e-01  2.166e-03    54.677  < 2e-16 ***
matchTypesolo            -2.129e-01  1.961e-03  -108.593  < 2e-16 ***
matchTypesolo-fpp        -2.114e-01  1.957e-03  -108.039  < 2e-16 ***
matchTypesquad            1.726e-01  1.823e-03    94.653  < 2e-16 ***
matchTypesquad-fpp        1.877e-01  1.815e-03   103.373  < 2e-16 ***
maxPlace                  2.121e-03  5.337e-05    39.748  < 2e-16 ***
numGroups                 4.740e-03  4.723e-05   100.358  < 2e-16 ***
rankPoints                1.083e-04  8.358e-07   129.634  < 2e-16 ***
revives                   1.241e-02  1.493e-04    83.069  < 2e-16 ***
rideDistance              1.787e-05  5.172e-08   345.479  < 2e-16 ***
roadKills                 2.122e-02  9.116e-04    23.282  < 2e-16 ***
swimDistance              1.121e-04  2.165e-06    51.764  < 2e-16 ***
teamKills                -1.392e-02  3.948e-04   -35.246  < 2e-16 ***
vehicleDestroys           1.163e-02  7.158e-04    16.253  < 2e-16 ***
walkDistance              1.098e-04  8.852e-08  1240.445  < 2e-16 ***
weaponsAcquired           1.208e-02  3.432e-05   352.071  < 2e-16 ***
winPoints                 1.492e-04  1.008e-06   147.981  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1231 on 3557532 degrees of freedom
Multiple R-squared:  0.8397,    Adjusted R-squared:  0.8397
F-statistic: 4.78e+05 on 39 and 3557532 DF,  p-value: < 2.2e-16
```

FIG-12 LINEAR REGRESSION RESULTS

From the above summary we can conclude that except the predictor longest kill and few categories of match type (duo, duo-fpp and solo-fpp), rest all variables are significant.

Explanation

The R-squared as well as the Adjusted R-squared is 0.84 which suggests that model has fit the data well. The F-statistic is also has a very high value and is thus far away from 1, indicating that we can reject the null hypothesis that intercept only model is equal to the model with predictors in terms of performance. We used the following code to predict the linear regression model on the validation dataset

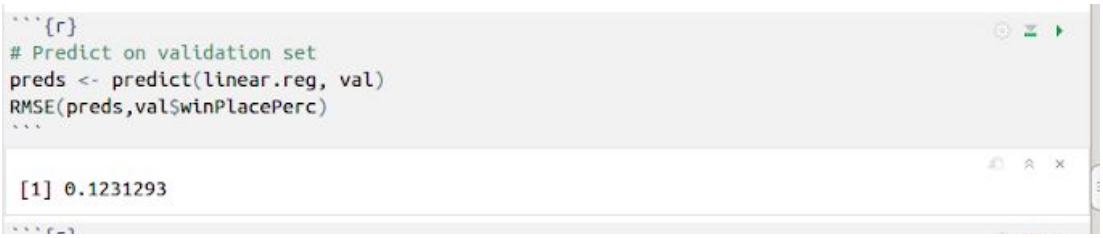


FIG-13 RMSE FOR LINEAR REGRESSION

Root Mean Squared Error was used as a metric to measure how our model performed on the validation set. Here we achieve an RMSE of 0.12 and we shall use this to compare with other models.

Advantages

Linear regression is known to work well when relationship between covariates and target variable is linear. In our case, it is very easy as we already know how our target variable correlates to few of the variables and makes the implementation simple too. Also R-squared and adjusted R-squared are equal, which means it's a good fit.

Disadvantages

Linear regression is sensitive to outliers. There is a lot of outlier in our data and hence we will have to try other algorithms and check first to conclude anything.

4.2. KNN

The k Nearest Neighbor is a non parametric algorithm, whose predictions for certain observation depends upon the distance from the nearest observation in the dataset.

Why did we chose this model?

KNN assumes that data is in feature space. Since data can be scalar with any metric, we have chosen to

try this algorithm to see how it works. Most of the variables are on different scale than each other, we need to normalize our dataset as KNN is distance based algorithm.

```r
# Normalize
norm.values <- preProcess(new_train, method=c("center", "scale"))
new_train.norm <- predict(norm.values, new_train)
val.norm <- predict(norm.values, val)

# Dummy variables for matchType because it's factor
new_train.norm <- dummy.data.frame(new_train.norm)
val.norm <- dummy.data.frame(val.norm)
```

Explanation

To determine k, we basically ran the model on 14 different values of k - 1 to 14 and chose the k with lowest RMSE.

```r
# Find K based on RMSE
accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))

### compute knn for different k on validation

for(i in 1:14) {
  knn.pred <-knn(train = new_train.norm,test = val.norm,cl = new_train.norm$winPlacePerc,k = i)
  knn.preds_num <- as.numeric(as.character(knn.pred))
  accuracy.df[i, 2] <- RMSE(val.norm$winPlacePerc,knn.preds_num)
  print(accuracy.df[i, 2])
}
write.csv(accuracy.df,'knn_acc.csv')
```

```
[1] 0.2247576
[1] 0.2389144
[1] 0.2663402
[1] 0.287035
[1] 0.3005109
[1] 0.3081035
[1] 0.31132
[1] 0.3115513
[1] 0.3093813
[1] 0.3058248
[1] 0.3011658
[1] 0.2964418
[1] 0.2918802
[1] 0.2873936
```

As we can see from the above figure, for 1st iteration the RMSE is lowest. Indicating that k = 1 is ideal.

Advantage

The cost of learning process is zero.

Disadvantage

It achieves RMSE of 0.22 which is worse than the linear regression model and hence we cannot use this model for final prediction. On a large dataset such as this, calculating the value of k is computationally expensive.

4.3. CART

We used Decision Trees for data mining with the objective of creating a model that predicts the value of winplaceprec based on the values of several input (or independent variables) which are available in our dataset. The algorithm of decision tree models works by repeatedly partitioning the data into multiple sub-spaces, so that the outcomes in each final sub-space is as homogeneous as possible. This approach is technically called recursive partitioning.
The produced result consists of a set of rules used for predicting the outcome variable, which can be either:

- a continuous variable, for regression trees
- a categorical variable, for classification trees

The decision rules generated by the CART predictive model are generally visualized as a binary tree.

Why did we chose this model?

In our project, we have used the Cart model to predict the response variable - winPlacePerc. We used the predict function on pca_val and type=raw to get the matrix of values returned by the network.

• We choose RSME as metric for the CART Model.
• We used method=”cv” that breaks the set into k-folds (parameter number which we putted as 3)
• The minimum number of observations that must exist in a node in order for a split to be attempted is 100 in our case.
• The minimum number of observations in any terminal leaf node is 100 in our case. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket*3 or minbucket to minsplit/3, as appropriate but as we know that minsplit is 100 in our case we have specified minbucket= round(100/3).
• We then created a grid using expand.grid function to Create a data frame from all combinations of the supplied vectors
• The next task was to fit the cart model for which we used the train function on the following arguments-

Data- The data on which we applied PCA
Method- A string specifying rpart model to be used
Metric- RSME
Tunegrid- To use the code used before by using exapnd.grid function
Trcontrol- To use method as cv, folds as 3
Control- To use minsplit as 100

```
> print(fit.cart)
CART

3557572 samples
     25 predictor

Pre-processing: centered (39), scaled (39)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 2371714, 2371716, 2371714
Resampling results across tuning parameters:

  cp    RMSE        Rsquared   MAE
  0.00  0.08720245  0.9195821  0.06146144
  0.05  0.15433752  0.7479576  0.11887546
  0.10  0.19258570  0.6075553  0.15736085

RMSE was used to select the optimal model using the
 smallest value.
The final value used for the model was cp = 0.
>
```

```
dt_preds <- predict(fit.cart,newdata = val,type = 'raw')
RMSE(val$winPlacePerc,dt_preds)
```

```
[1] 0.08654272
```

RSME is used to analyse how fairly our model worked on validation dataset. RSME of 0.08 is evident that this model is good.

The lift chart shows us that our model is doing good as there is clear separation from baseline.
We can conclude from the above representations that CART Model is doing the job well.

Advantages of Cart model

1. Decision trees implicitly perform variable screening or feature selection
2. Decision trees require relatively little effort from users for data preparation
3. Nonlinear relationships between parameters do not affect tree performance

Disadvantages of Cart Model

1. Data fragmentation : Each split in a tree leads to a reduced dataset under consideration. And, hence the model created at the split will potentially introduce bias.

2. High variance and unstable : As a result of the greedy strategy applied by decision tree's variance in finding the right starting point of the tree can greatly impact the final result. i.e small changes early on can have big impacts later. So- if for example you draw two different samples from your universe , the starting points for both the samples could be very different (and may even be different variables) this can lead to totally different results.

5. Results

	Model	RMSE
3	Decision Tree	0.08
5	PCA - KNN	0.09
6	PCA – Decision Tree	0.11
1	Linear Regression	0.12
4	PCA – Linear Regression	0.15
2	KNN	0.22

From the above table we can see how the models performed on the validation set.
The top 3 models based on RMSE are:

1. Decision Tree
2. KNN on the 14 Principal Components
3. Decision Tree on the 14 Principal Components

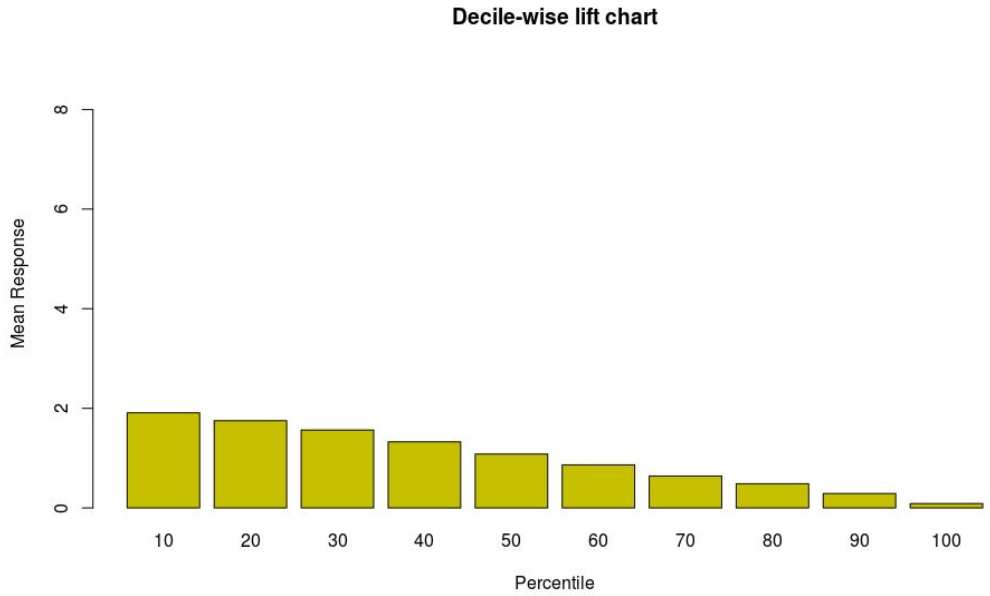
We can see a pattern that top nonlinear models have outperformed the linear regression model.
Thus we can conclude that the dataset had nonlinear features and the linear regression although gave a good accuracy wasn't able to capture those features.

Since we ran 2 sets of algorithms - Without PCA and with PCA, we can choose any algorithm sets according to our need.

Best Accuracy

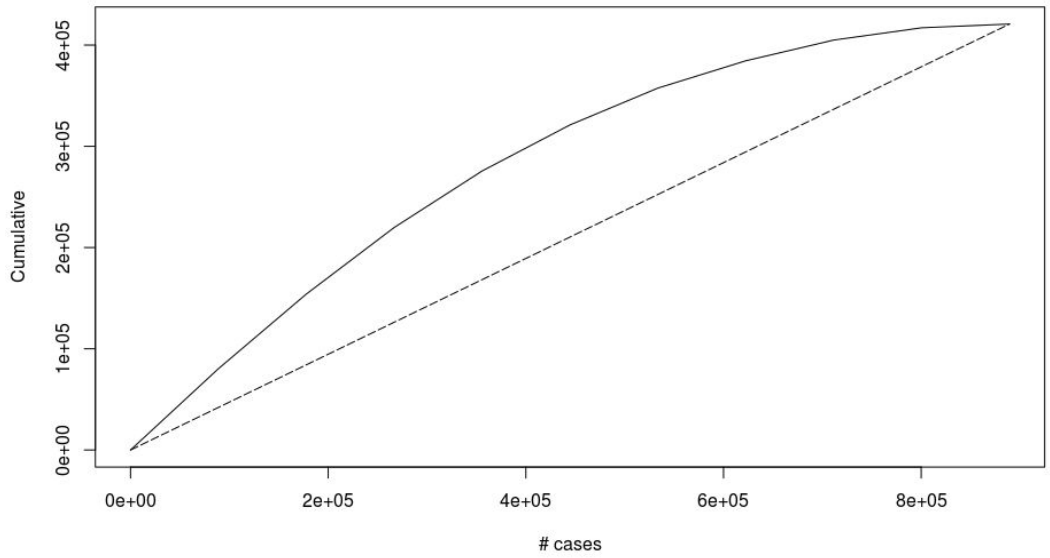
The most accurate model is the Decision Tree model and would give the best performance amongst others.

Decile Chart for Decision Tree Model



From the decile chart we can see a staircase effect, the bars are descending in order from left to right. This shows that our population correctly from most likely to win to least likely to win. This tells us that our model is performing well.

Lift Chart for Decision Tree Model



From the lift chart we can see that we have a good lift from base line. The lift curve and baseline are well separated. This shows that our model is performing well when compared to the random naive model.

High Speed and Low Computation Resources

Since predicting the player rank in PUBG relies on data we may want to choose an algorithm which is fast as well as computationally less expensive. For this particular application we can go for PCA - KNN or PCA - Decision Tree. Although KNN performs computations at runtime and hence using PCA - Decision Tree is advisable. This would mean sacrificing little accuracy but we would get better speed and use less computation resources which is favorable for real time applications.

Recommendations

According to our analysis we conclude that following actions result in better final ranking:

1. **Consume more boost items**
   Boost items like Energy Drink, Painkiller and Adrenaline Syringe fill up the boost bar of the player as well as heal certain percent of health. As the boost bar fills up it grants players health regeneration over time and movement speed. These traits can help players eliminate other players and hence it seem to lead to better ranking.
2. **Roam more around the map**
   Players who roam more often encounter more weapons and heal items and can eliminate other players as well. Thus increased walk distance leads to better final rank.
3. **Heal often**
   As evident from strong correlation of heal items and win place, collecting healing items and using them when health is low is good strategy to survive in the game.
4. **Finally kill, assist and knock down more enemies**
   As players kill more enemies they reduce their competition for the first place. In team games assisting fellow teammates to kill other players also helps the team to succeed.

6. References

[1] Trevor Hastie, Robert Tibshirani, Jerome Friedman 2001 Elements of Statistical Learning: Data Mining, Inference, and Prediction (Springer)

[2] Efron, Brian and Trevor Hastie. 2016. Computer Age Statistical Inference: Algorithms, Algorithms, Evidence, and Data Science (Institute of Mathematical Statistics Monographs) Cambridge University Press.

[3] R. Caruana, "Multitask learning," Mach. Learn., vol. 28, no. 1, pp. 41–75, 1997.

[4] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng, "Boosted multi-task learning," Mach. Learn., vol. 85, no. 1–2, pp. 149–173, 2011.

[5] McAloon, Alissa (July 31, 2017). "Playerunknown's Battlegrounds claims record for most Peak player in non-Value game" August 1, 2017

[6] Rad, Chloi (August 14, 2017). "PlayerUnknown's Battlegrounds: Total Player "Killcount and Other Crazy Stats" August 14, 2017

[7] Williams, Mike (December 22, 2017). "PlayerUnknown's Battlegrounds (PUBG) Review: "First Man Standing" US Gamer January 7, 2018

[8] Tony Cai, Linjun Zhang, Harrison H. Zhou (Forthcoming), "Adaptive Functional Linear Regression via Functional Principal Component Analysis and Block Thresholding" 2012

[9] Toutenburg, H. and Shalabh (1996): "Predictive Performance of the Methods of Restricted and Mixed Regression Estimators", Biometrical Journal, 38, 8, pp. 951-959.

[10] Shalabh (1997): "On Efficient Forecasting in Linear Regression Models", Journal of Quantitative Economics, Vol. 36, No. 2, pp. 133-140.

[11] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination:

consistency properties. Technical report, DTIC Document, 1951.

[12] DA Adeniyi, Z Wei, and Y Yongquan. Automated web usage data mining and recomendation system using k-nearest neighbor (knn) classification method. Applied Computing and Informatics, 12(1):90–108, 2016.

[13] Sadegh Bafandeh Imandoust and Mohammad Bolandrtar. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. International Journal of Engineering Research and Applications, 3(5):605–610, 2013.

Individual Report - Rajani, Shruti

Working in this project was a unique experience for me. Initially, choosing the dataset for the project also seemed to be a challenge since all the members were from different domain and had different perspective but we all collectively decided to enroll for the ongoing competition at Kaggle and started working on that dataset. The dataset was both fun to work with as well as challenging. I learned how different perspective on the same dataset can easily show pros and cons of working on a dataset which an individual would not have easily thought. After everyone's concern we decided to work on PUBG dataset from Kaggle.

I was lucky to be a part of a supportive, flexible and hardworking team. The smooth functioning of the project was because of the joint effort to learn collectively. We would meet to discuss further actions,discuss any challenges faced, plan of action for further working on project which gave us clear image of what is expected from us in the forthcoming week. It helped me gain knowledge from my peer group and helped learning the concepts more clearly.It was a great experience by working on this project since it helped me implement the various algorithms learned in class on a single problem dataset. Moreover, during meeting you are not only confined to project related talks but also something which helps us in functioning better in our practices involved.

Successes and challenges :

We were successful in achieving the task of predicting the chances of wining and predicting various factor which could be the best strategy to win for the player. We were happy with the accuracy we achieved for our various model. Talking about the challenges faced, it was difficult to implement various algorithms relevant for out dataset. We initially worked on 500,000 rows instead of actual number of rows to have unhindered progress while working on the project. Another challenge we faced was due to different class schedule of team members which made us discuss remotely at times about the updates. Another difficulty was to compile the whole code and run it since it was unsupportive for few of our machines we were working with. Overall, we collectively changed all the mirrors into windows by finding the right solution for the problem.

We were a team of 5 - Sabyasachi Choudhury, Monica Kumar, Shruti Rajani, Nihit Save, Sidheswar Venkatachalapathi. I worked on Exploratory Data Analysis using Correlation Matrix, Box Plot and other data visualization. Nihit Save played a major role in the completion of project by working on various algorithms - Linear Regression, KNN and compiled all the code, explaining result section and even helped others with the errors that came during the project working and any understanding of the arguments that are used. Sidheswar Venkatachalapathi came up with the idea of using this dataset which we all agreed since it was also an ongoing competition on Kaggle and was also involved in streamlining the flow of meetings. He actively worked on dimension reduction using PCA, professional quality report preparation and was also helpful in explaining what the PUBG game is all about. Sabyasachi and Monica assisted with the data understanding, report creation by expelling the objectives, introduction, result explanation and making it complete in right quality. Sabyasachi was always eager to help other team members in code and concepts.All the members contributed towards the success of this project by sharing the ideas, developments required and completing the project report in professional quality by completing any section that needs any further explanation.