Rachana Patil

AGENDA

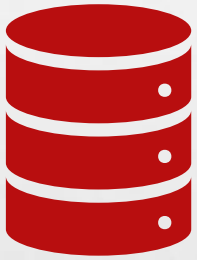
- Overview of the dataset
- Problem Statement
- Literature Review
- Data cleaning and preprocessing
- Method implementations – MLR, CART, kNN
- Analysis
- Conclusion

INTRODUCTION

- GAME BACKGROUND

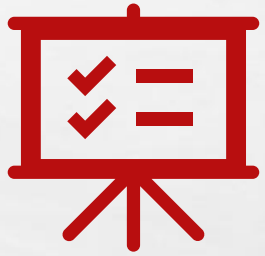
- SOURCE: HTTPS://WWW.KAGGLE.COM/C/PUBG-FINISH-PLACEMENT-PREDICTION

OVERVIEW OF THE DATASET



variables and records in the data

29 Variables : 4.4 million records



Types of variables

28 continuous, 1 categorical

OVERVIEW OF THE DATASET (VARIABLES)

DBNOs	killPoints
assists	killStreaks
boosts	kills
damageDealt	longestKill
headshotKills	matchDuration
heals	matchId
id	matchType
killPlace	rankPoints

OVERVIEW OF THE DATASET (VARIABLES)

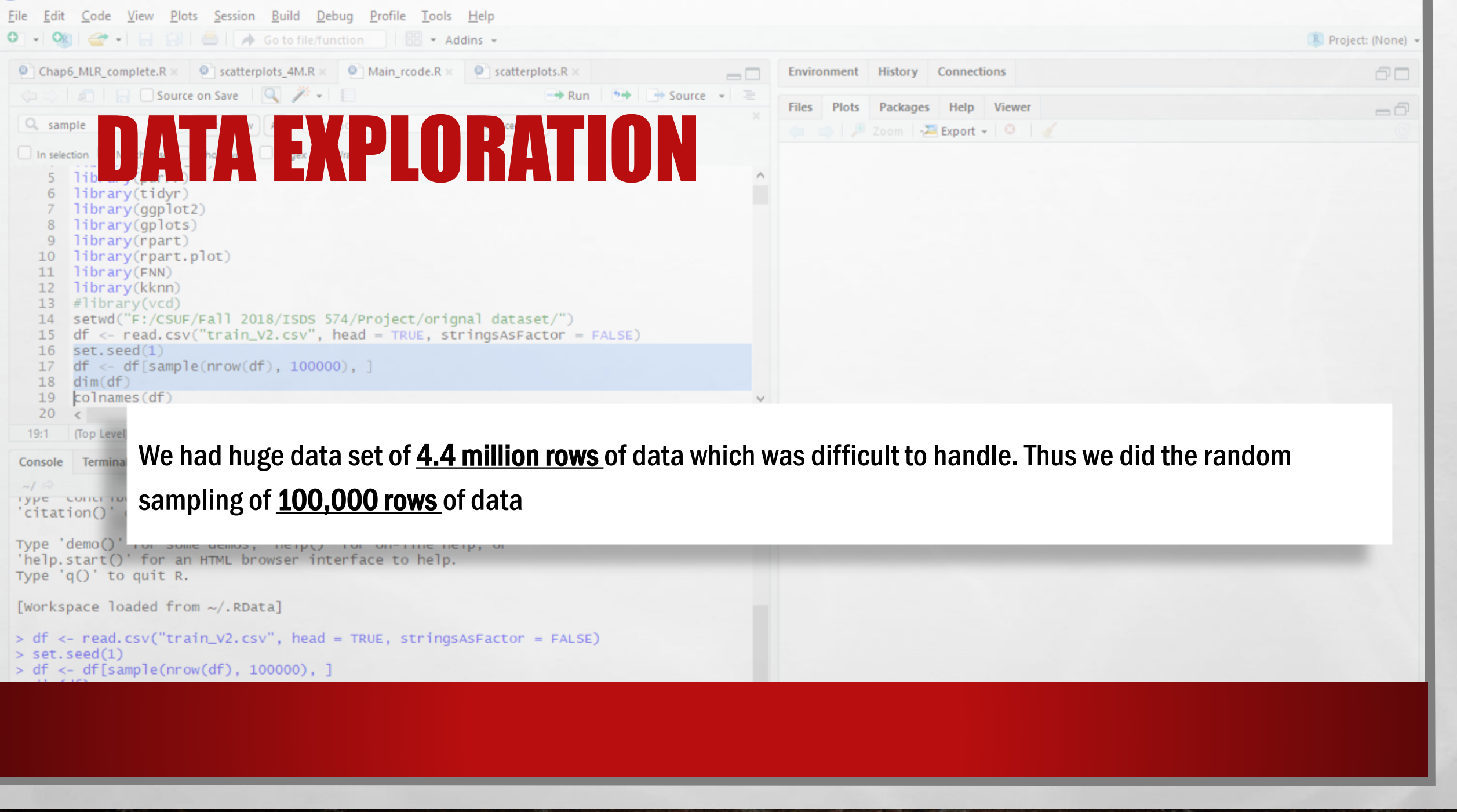
revives	winPoints
rideDistance	groupId
roadKills	numGroups
swimDistance	maxPlace
teamKills	winPlacPerc
vehicleDestroys	
walkDistance	
weaponsAcquired	

PROBLEM STATEMENT

The objective of this data mining task is to predict a player's finishing placement based on the final stats generated at the end of the match.

Output variable: Winplaceperc

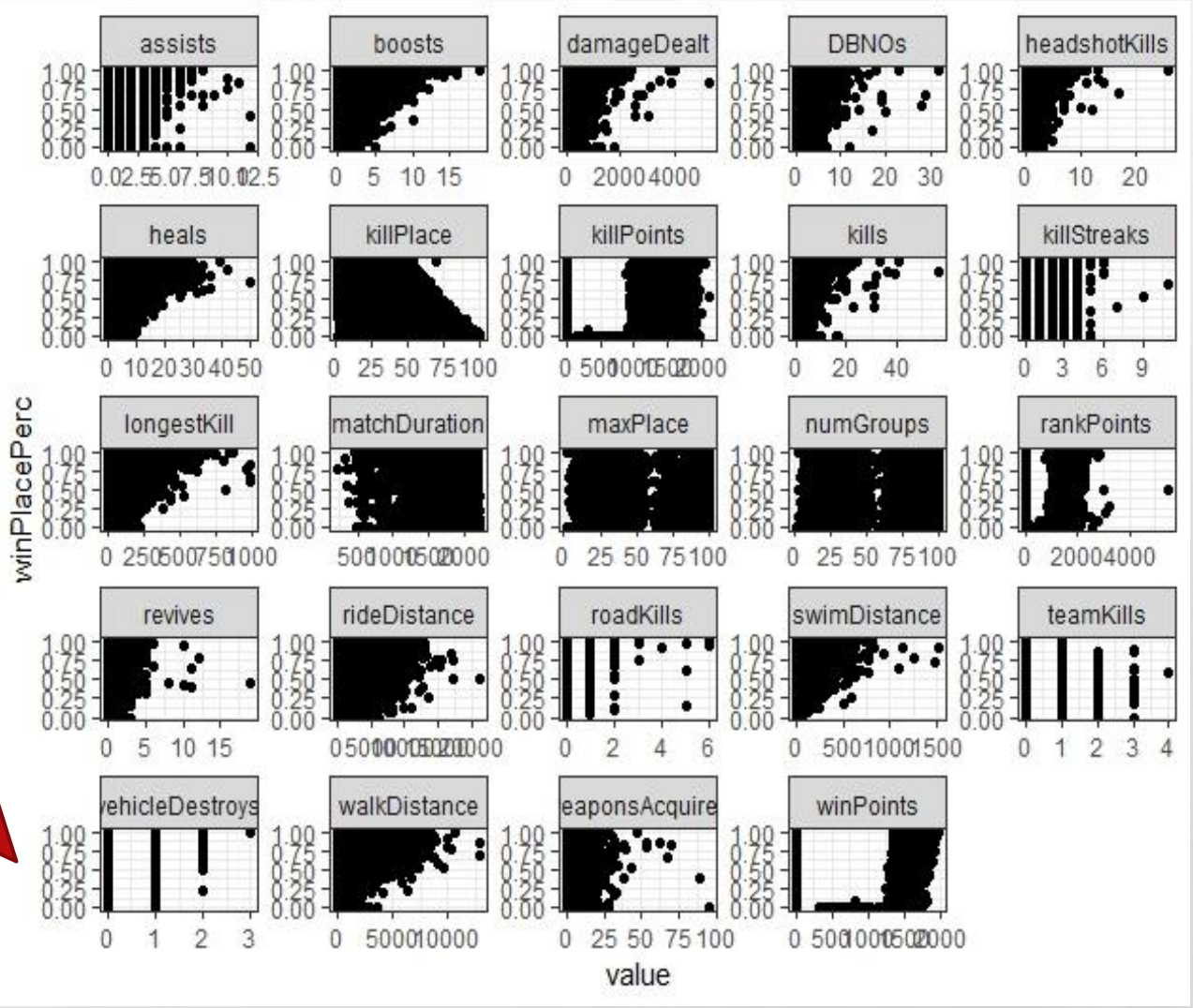
DATA EXPLORATION



We had huge data set of **4.4 million rows** of data which was difficult to handle. Thus we did the random sampling of **100,000 rows** of data

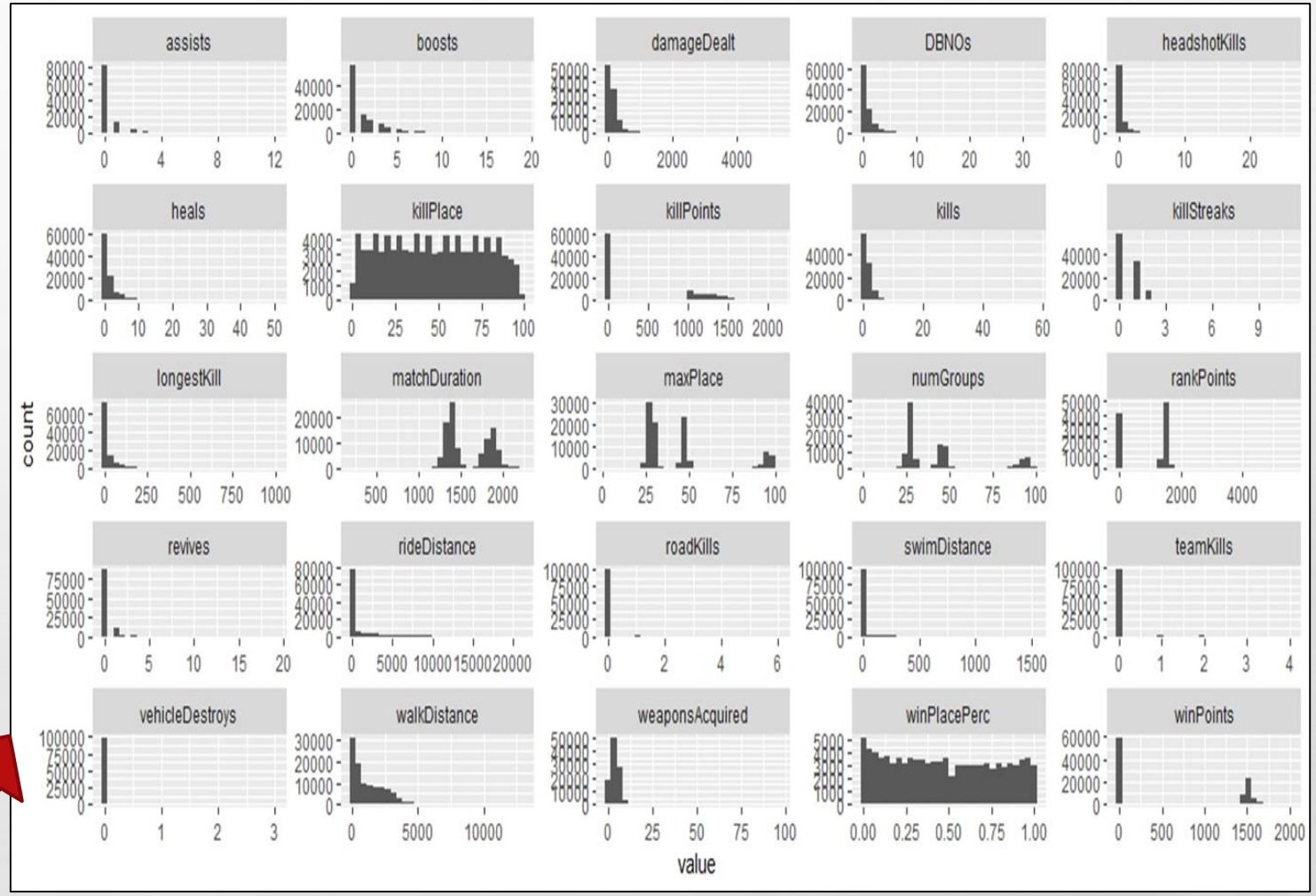
SCATTERPLOTS

```
35
36 ###### Consolidated Scatter Plot################
37 df %>%
38   gather(-winPlacePerc, key = "var", value = "value") %>%
39   ggplot(aes(x = value, y = winPlacePerc)) +
40   geom_point() +
41   facet_wrap(~ var, scales = "free") +
42   theme_bw()
43
```



HISTOGRAMS

```
43
44 ###### Consolidated Histogram #################
45 df %>%
46   keep(is.numeric) %>%
47   gather() %>%
48   ggplot(aes(value)) +
49   facet_wrap(~ key, scales = "free") +
50   geom_histogram()
51
```



DATA CLEANING AND PREPROCESSING

Variable Reduction

Data Set	Rows, Columns
After Random Sampling	[100,000, 29]
Based On Domain Knowledge	[100,000, 24]

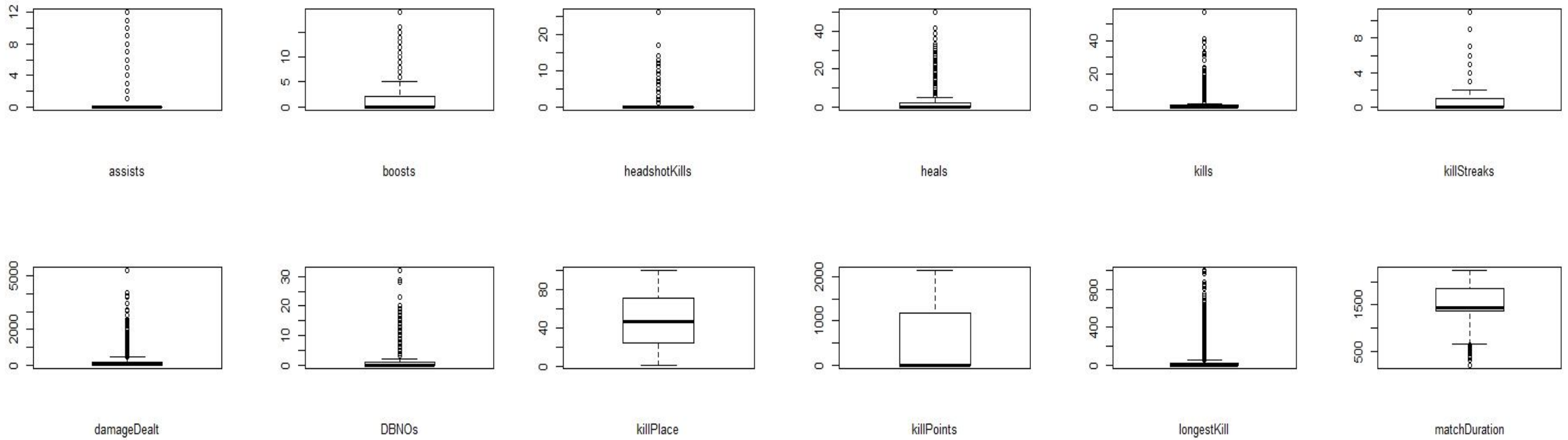
1

1) Group ID
2) Match ID
3) ID
4) Matchtype
5) RoadKills

DATA CLEANING AND PREPROCESSING

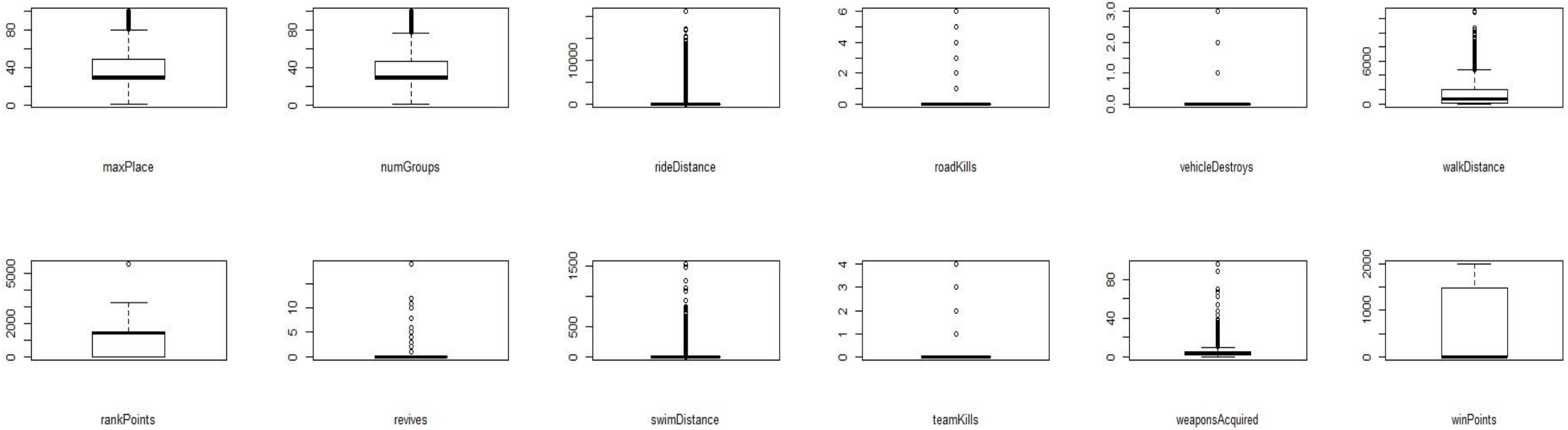
Outlier Detection: Box Plot

2



DATA CLEANING AND PREPROCESSING

Outlier Detection: Box Plot



DATA CLEANING AND PREPROCESSING

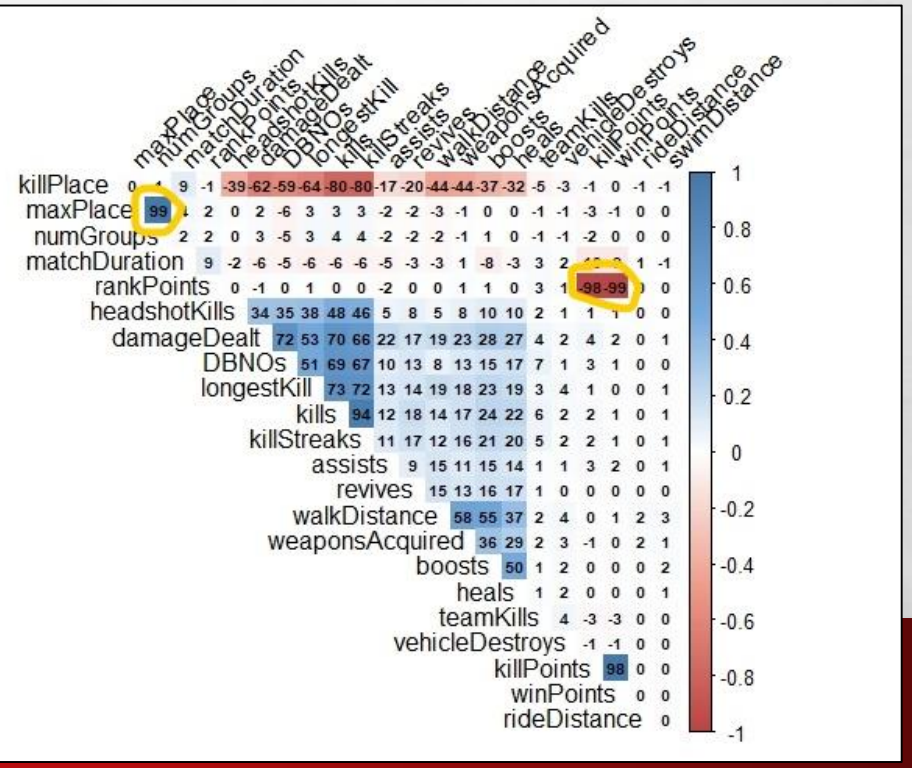
3 Outliers Handled

```
########################## Data Cleaning #############

as <- which(df$assists > 1)
bo <- which(df$boosts >= 6)
Dma <- which(df$damageDealt >= 500)
DBno <- which(df$DBNOs >= 3)
hskill <- which(df$headshotKills >= 2)
```

[47044,24]

4 Correlation Matrix

```
230  cormat<- cor(df3) #/// to create corelation matrix
231
232  corrplot(cormat,method = 'number', title = 'Corealation Matrix',
233           addCoefasPercent = TRUE, number.cex = 0.65)
234
```



Cutoff Value: 0.90

5

MaxPlace, Numgroup	99
Killpoints, Rankpoints	-98
Winpoints, Rankpoints	-99

[47044,21]

DATA CLEANING AND PREPROCESSING

6 Normalization

```
273
274  df3 <- as.data.frame(scale(df3[,c(1:20)]))
275
276  class(df3)
277  summary(df3)
```

On all variables except Output variable

```
> summary(df3)
    assists             boosts          damageDealt
 Min.   :-0.3387   Min.   :-0.4252   Min.   :-0.8609
 1st Qu.:-0.3387   1st Qu.:-0.4252   1st Qu.:-0.8609
 Median :-0.3387   Median :-0.4252   Median :-0.2882
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.:-0.3387   3rd Qu.:-0.4252   3rd Qu.: 0.3873
 Max.   : 2.9522   Max.   : 5.9531   Max.   : 5.3053
```

7 Partitioning Data Set

70% Training Dataset

```
trainind <- sample(1:nrow(df3), nrow(df3)*.7)
train_df <- df3[trainind, ]
length(trainind)
```

```
> dim(train_df)
[1] 32930    21
```

30% Test Dataset

```
testind <- setdiff(1:nrow(df3), trainind)
test_df <- df3[testind, ]
length(testind)
```

```
> dim(test_df)
[1] 14114    21
```

METHOD IMPLEMENTATIONS

MLR – Multiple Linear Regression

CART – Classification and Regression Tree

kNN – k Nearest Neighbours

MULTIPLE LINEAR REGRESSION

- Predict the value of the dependent variable winPlacePerc based on predictors kills, DBNOs, assists, boosts etc.
- Regression coefficients B_1, B_2,..., B_P In the equation and they are calculated using ordinary least squares (OLS).

$$Y = B_0 + B_1X_1 + B_2X_2 + \ldots.. + B_PX_P + \text{ERROR}$$

MULTIPLE LINEAR REGRESSION

VARIABLE SELECTION METHODS

- FORWARD SELECTION

```
obj.null <- lm(winPlacePerc ~ 1, dat = train_df)
obj.full <- lm(winPlacePerc ~ ., dat = train_df)
obj_fwd = step(obj.null, scope=list(lower=obj.null, upper=obj.full), direction='forward')
summary(obj_fwd)
```

LM(FORMULA = WINPLACEPERC ~ WALKDISTANCE + KILLPLACE + KILLSTREAKS + NUMGROUPS + MATCHDURATION + KILLS + BOOSTS + WEAPONSACQUIRED + ASSISTS + DBNOS + HEALS + REVIVES + KILLPOINTS + LONGESTKILL + HEADSHOTKILLS + DAMAGEDEALT + SWIMDISTANCE, DATA = TRAIN_DF)

```
Residual standard error: 0.1165 on 32912 degrees of freedom
Multiple R-squared:  0.7891,    Adjusted R-squared:  0.7889
F-statistic:  7242 on 17 and 32912 DF,  p-value: < 2.2e-16
```

MULTIPLE LINEAR REGRESSION

VARIABLE SELECTION METHODS

- BACKWARD SELECTION

```
obj.null <- lm(winPlacePerc ~ 1, dat = train_df)
obj.full <- lm(winPlacePerc ~ ., dat = train_df)
obj_bkwd = step(obj.null, scope=list(lower=obj.null, upper=obj.full), direction='backward')
summary(obj_bkwd)
```

LM(FORMULA = WINPLACEPERC ~ ASSISTS + BOOSTS + DAMAGEDEALT + DBNOS + HEADSHOTKILLS + HEALS + KILLPLACE + KILLPOINTS + KILLS + KILLSTREAKS + LONGESTKILL + MATCHDURATION + NUMGROUPS + REVIVES + SWIMDISTANCE + WALKDISTANCE + WEAPONSACQUIRED, DATA = TRAIN_DF)

```
Residual standard error: 0.1165 on 32912 degrees of freedom
Multiple R-squared:  0.7891,    Adjusted R-squared:  0.7889
F-statistic:  7242 on 17 and 32912 DF,  p-value: < 2.2e-16
```

MULTIPLE LINEAR REGRESSION

VARIABLE SELECTION METHODS

- STEPWISE SELECTION

```
obj.null <- lm(winPlacePerc ~ 1, dat = train_df)
obj.full <- lm(winPlacePerc ~ ., dat = train_df)
obj_step = step(obj.null, scope=list(lower=obj.null, upper=obj.full), direction='both')
summary(obj_step)
```

LM(FORMULA = WINPLACEPERC ~ WALKDISTANCE + KILLPLACE + KILLSTREAKS + NUMGROUPS + MATCHDURATION + KILLS + BOOSTS + WEAPONSACQUIRED + ASSISTS + DBNOS + HEALS + REVIVES + KILLPOINTS + LONGESTKILL + HEADSHOTKILLS + DAMAGEDEALT + SWIMDISTANCE, DATA = TRAIN_DF)

```
Residual standard error: 0.1165 on 32912 degrees of freedom
Multiple R-squared:  0.7891,    Adjusted R-squared:  0.7889
F-statistic:  7242 on 17 and 32912 DF,  p-value: < 2.2e-16
```

MULTIPLE LINEAR REGRESSION

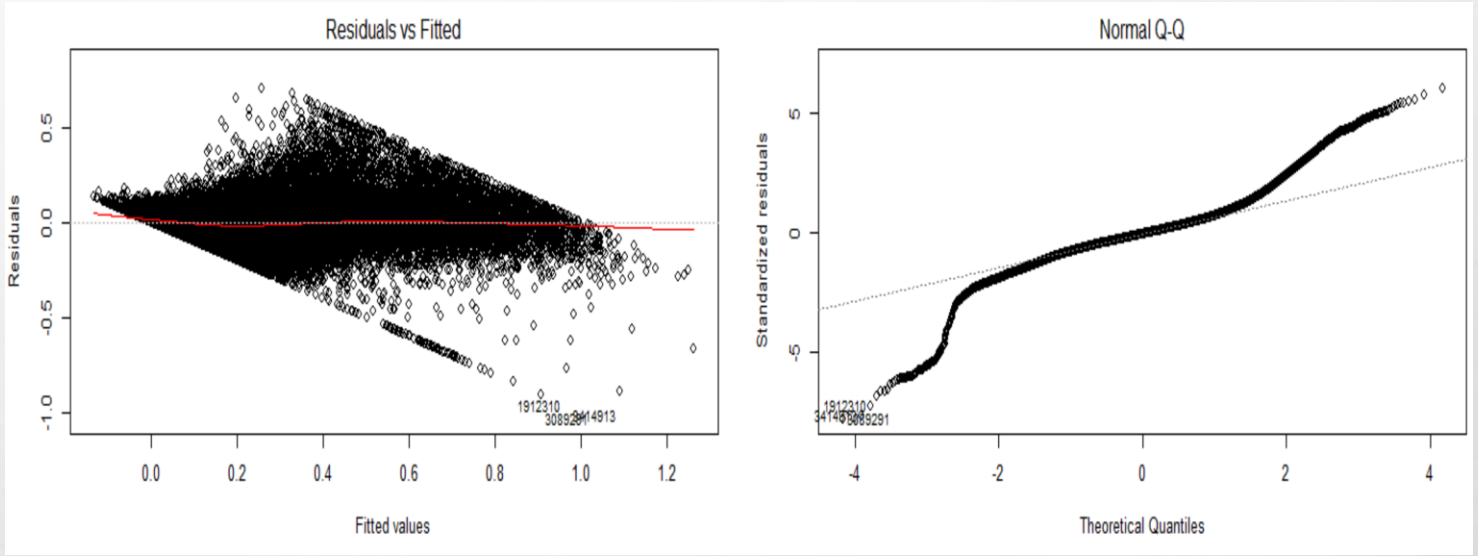
ASSUMPTIONS

- **NORMALITY**

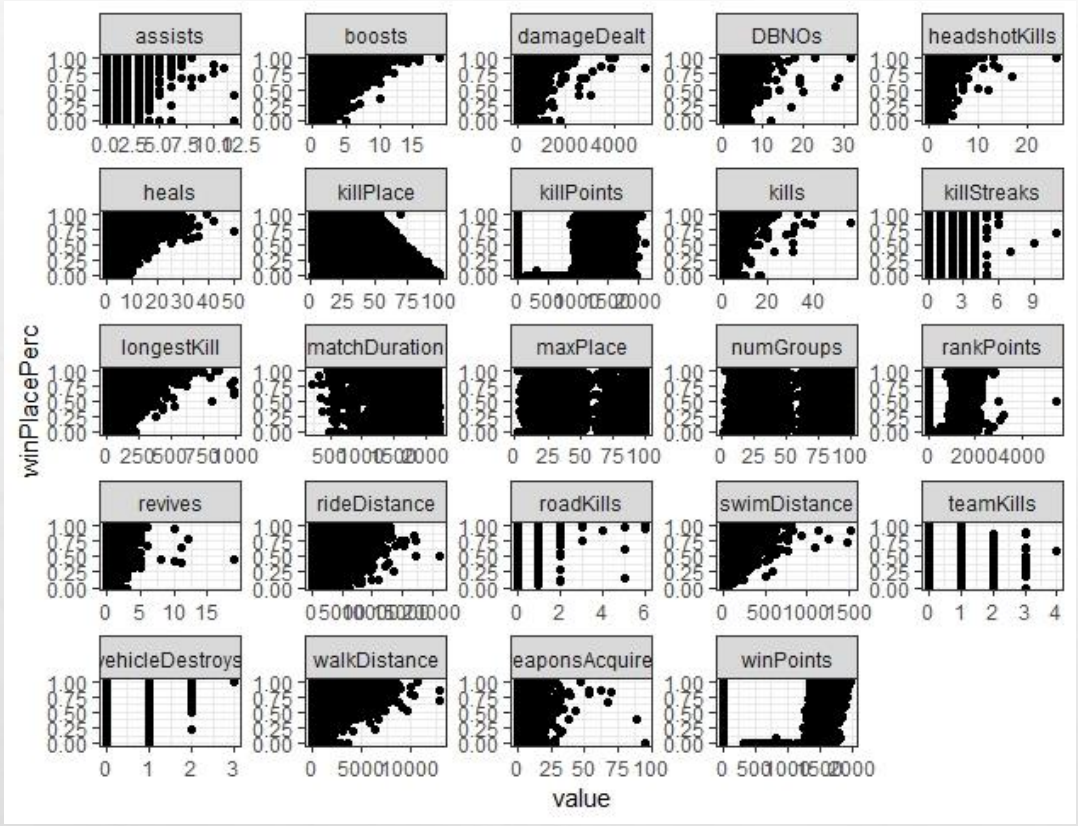
The residuals (errors) should be normally

Distributed about the predicted

Responses.



Two common methods to check this assumption include using: (a) a histogram (with a superimposed normal curve) and a Normal P-P Plot; or (b) a Normal Q-Q Plot of the studentized residuals.



MULTIPLE LINEAR REGRESSION

ASSUMPTIONS

- **LINEARITY**

Linear relationship between the outcome variable and the independent variables.

Scatterplots can show whether there is a linear or curvilinear relationship

MULTIPLE LINEAR REGRESSION

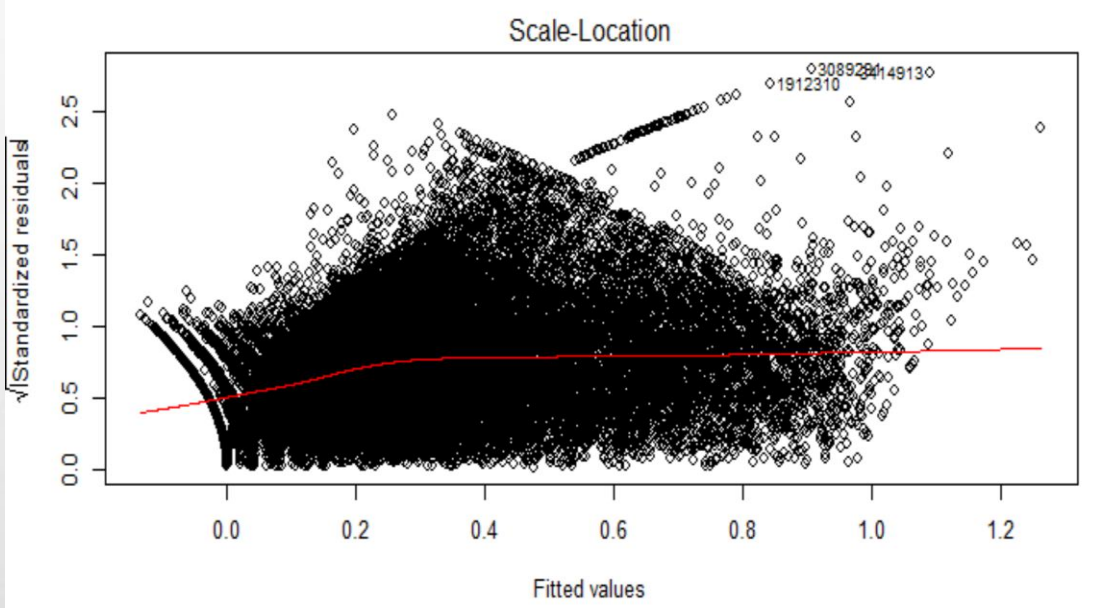
ASSUMPTIONS

- **INDEPENDENCE**

All predictors must be independent of each other.

- **HOMOSCEDASTICITY**

Plot of standardized residuals versus predicted values can show whether points are equally distributed across all values of the independent variables.



MULTIPLE LINEAR REGRESSION

RESULTS

YHAT = PREDICT(OBJ_FWD, NEWDATA=TEST_DF[,])

YTEST = TEST_DF$WINPLACEPERC

RMSE(YTEST, YHAT)

[1] 0.1163719 → RMSE

```
Coefficients:
                 Estimate Std. Error  t value Pr(>|t|)
(Intercept)     0.3040876  0.0006420  473.631  < 2e-16 ***
walkDistance    0.0971712  0.0009572  101.516  < 2e-16 ***
killPlace      -0.2096509  0.0014117 -148.513  < 2e-16 ***
killStreaks    -0.0807176  0.0019531  -41.328  < 2e-16 ***
numGroups       0.0302902  0.0006503   46.580  < 2e-16 ***
matchDuration  -0.0291164  0.0006517  -44.677  < 2e-16 ***
kills          -0.0631869  0.0020713  -30.506  < 2e-16 ***
boosts          0.0122526  0.0008523   14.376  < 2e-16 ***
weaponsAcquired 0.0150136  0.0008272   18.150  < 2e-16 ***
assists         0.0060512  0.0006700    9.032  < 2e-16 ***
DBNOs          -0.0105053  0.0010303  -10.196  < 2e-16 ***
heals           0.0066036  0.0007602    8.687  < 2e-16 ***
revives         0.0055498  0.0006629    8.372  < 2e-16 ***
killPoints      0.0034375  0.0006465    5.317 1.06e-07 ***
longestKill     0.0052059  0.0009684    5.376 7.68e-08 ***
headshotKills  -0.0036143  0.0007365   -4.907 9.28e-07 ***
damageDealt     0.0031530  0.0010672    2.954  0.00313 **
swimDistance    0.0015185  0.0006192    2.452  0.01420 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1165 on 32912 degrees of freedom
Multiple R-squared:  0.7891,    Adjusted R-squared:  0.7889
F-statistic:  7242 on 17 and 32912 DF,  p-value: < 2.2e-16
```

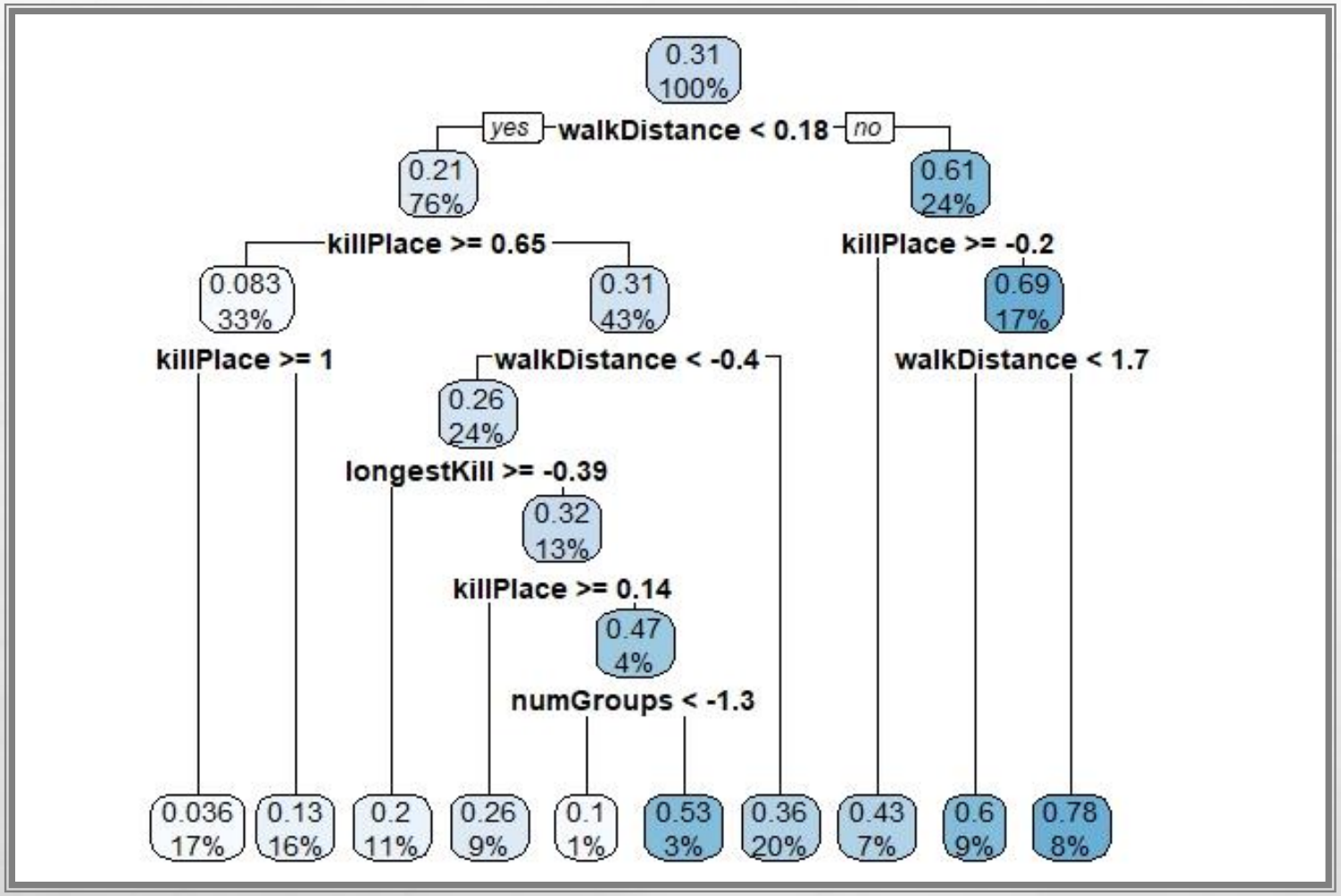
REGRESSION TREE

Predicted Rating

If killPlace value lies between 0.66 and 0.99, then the winning percentage will be 0.13

Decision nodes with variable importance

- Walkdistance: 43
- Killplace: 20
- Longestkill: 4
- Numgroups: 1



REGRESSION TREE

Best Pruned Tree

RMSE: 0.1278694

REGRESSION TREE

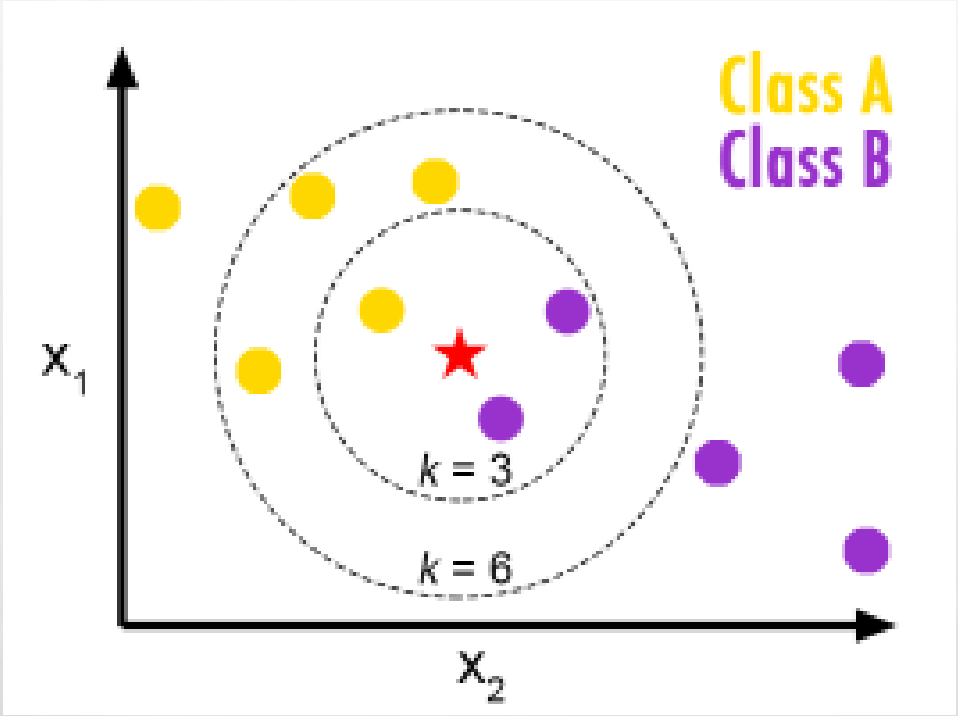
```
> rpart.plot(cart_fit)
> pfit = prune(cart_fit, cp = cart_fit$cptable[which.min(cart_fit$cptable[,"xerror"]),"CP"])
> rpart.plot(pfit)
```

Output:

```
> rmse(yhat.test, ytest)
[1] 0.1278694
> rmse(yhat.test_prune, ytest)
[1] 0.1278694
```

	CP	nsplit	rel error	xerror	xstd
1	0.458263	0	1	1.000063	0.007357
2	0.145269	1	0.541737	0.5439	0.005379
3	0.052109	2	0.396468	0.398624	0.004279
4	0.021327	3	0.344359	0.346266	0.004052
5	0.016359	4	0.323032	0.32504	0.003942
6	0.012975	7	0.273955	0.276986	0.003498
7	0.012213	8	0.260981	0.272975	0.003467
8	0.01	9	0.248768	0.25355	0.003518

KNN – K NEAREST NEIGHBORS



KNN – K NEAREST NEIGHBORS

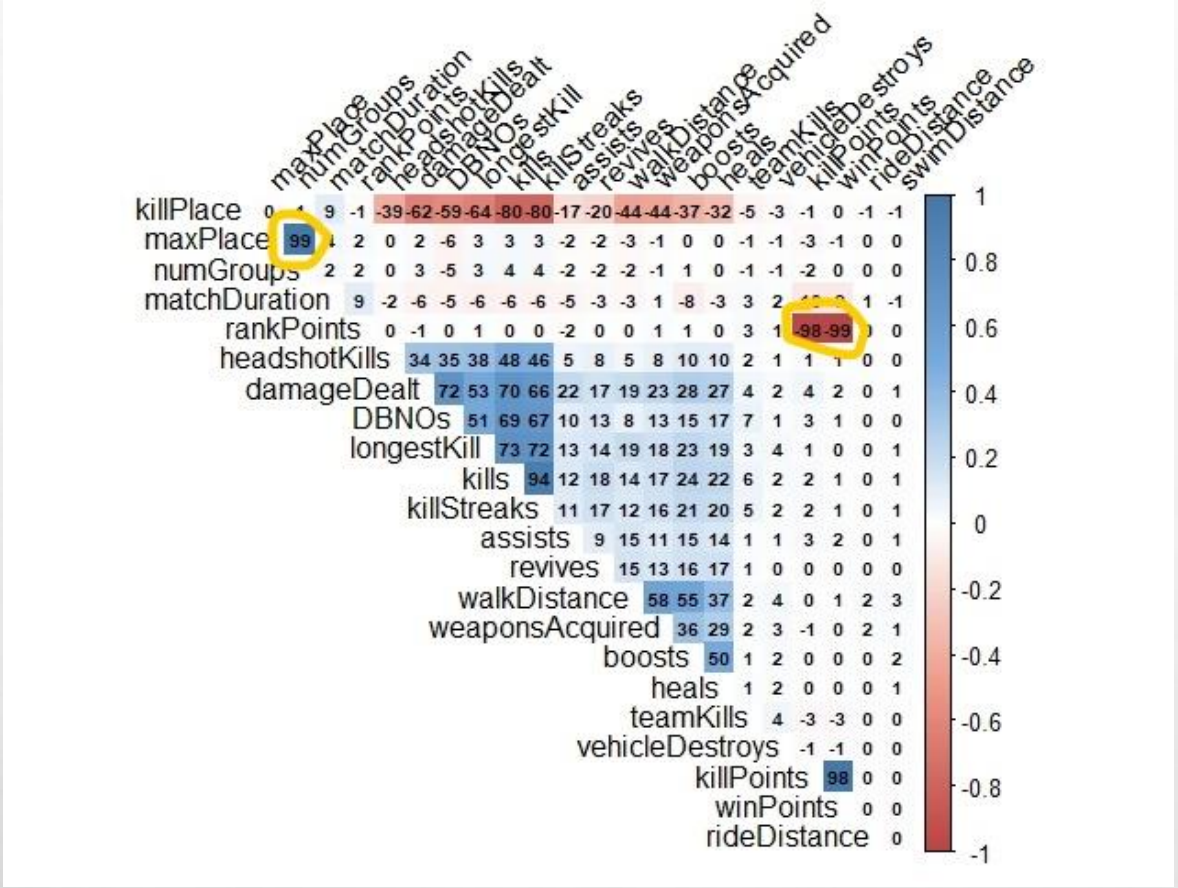
Optimum K

```
> knn.reg.bestK = function(Xtrain, Xtest, ytrain, ytest, kmax=NULL) {
+   if (is.null(kmax)) kmax = length(ytrain)^.5
+   vec.rmse = rep(NA, kmax)
+   for (k in 1:kmax) {
+     yhat.test = knn.reg(Xtrain, Xtest, ytrain, k)$pred
+     vec.rmse[k] = rmse(yhat.test, ytest)
+   }
+   list(k.opt = which.min(vec.rmse), rmse.min = min(vec.rmse), vec.rmse)
+ }
> knn.reg.bestK(train_df[,1:20], test_df[,1:20], train_df[,21], test_df[,21])
$`k.opt`
[1] 11

$rmse.min
[1] 0.1164329

[[3]]
  [1] 0.1489087 0.1292303 0.1235195 0.1208125 0.1189639 0.1178877 0.1173630 0.1167980 0.1164932
 [10] 0.1165901 0.1164329 0.1165463 0.1166584 0.1166734 0.1167291 0.1169729 0.1169507 0.1171207
 [19] 0.1172004 0.1173454 0.1174112 0.1175510 0.1175958 0.1176799 0.1178767 0.1180416 0.1181332
 [28] 0.1183772 0.1185622 0.1187579 0.1189284 0.1191680 0.1193679 0.1194682 0.1196381 0.1197152
 [37] 0.1199071 0.1200618 0.1202195 0.1203314 0.1204788 0.1206817 0.1208332 0.1209504 0.1210391
```

KNN – K NEAREST NEIGHBORS

kNN objects

```
> knn_obj <- kknn(winPlacePerc~., train_df, test_df, k=11)
> names(knn_obj)
 [1] "fitted.values" "CL"            "W"             "D"             "C"
 [6] "prob"          "response"      "distance"      "call"          "terms"
> head(knn_obj$D[1:2,1:11])
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]
[1,] 1.7706986 1.7731392 1.8984154 1.9963801 2.0224436 2.0795305 2.0921001 2.1677385 2.1833689
[2,] 0.4299748 0.4889075 0.5464724 0.5548825 0.5832166 0.6005727 0.6326097 0.6489151 0.6865022
         [,10]     [,11]
[1,] 2.2006689 2.2373768
[2,] 0.7151994 0.7310168
> head(knn_obj$fitted.values)
[1] 0.287107873 0.151477529 0.149663675 0.000000000 0.008124091 0.216216771
> head(knn_obj$D[1:2,1:11])
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9]
[1,] 1.7706986 1.7731392 1.8984154 1.9963801 2.0224436 2.0795305 2.0921001 2.1677385 2.1833689
[2,] 0.4299748 0.4889075 0.5464724 0.5548825 0.5832166 0.6005727 0.6326097 0.6489151 0.6865022
         [,10]     [,11]
[1,] 2.2006689 2.2373768
[2,] 0.7151994 0.7310168
> head(knn_obj$response)
[1] "continuous"
> head(knn_obj$call)
kknn(formula = winPlacePerc ~ ., train = train_df, test = test_df,
    k = 11)
> knn_obj$terms
winPlacePerc ~ assists + boosts + damageDealt + DBNOs + headshotKills +
    heals + killPlace + killPoints + kills + killStreaks + longestKill +
    matchDuration + numGroups + revives + rideDistance + swimDistance +
    teamKills + vehicleDestroys + walkDistance + weaponsAcquired
```



ANALYSIS

- Applied PCA on kill related variables.

```
pca_obj <- prcomp(train_df[ ,c(5,7,9,10,11)])
```

RESULTS

MLR OUTPUT WITH PCA

```
Residual standard error: 0.1485 on 32917 degrees of freedom
Multiple R-squared:  0.6574,    Adjusted R-squared:  0.6573
F-statistic:  5264 on 12 and 32917 DF,  p-value: < 2.2e-16
```

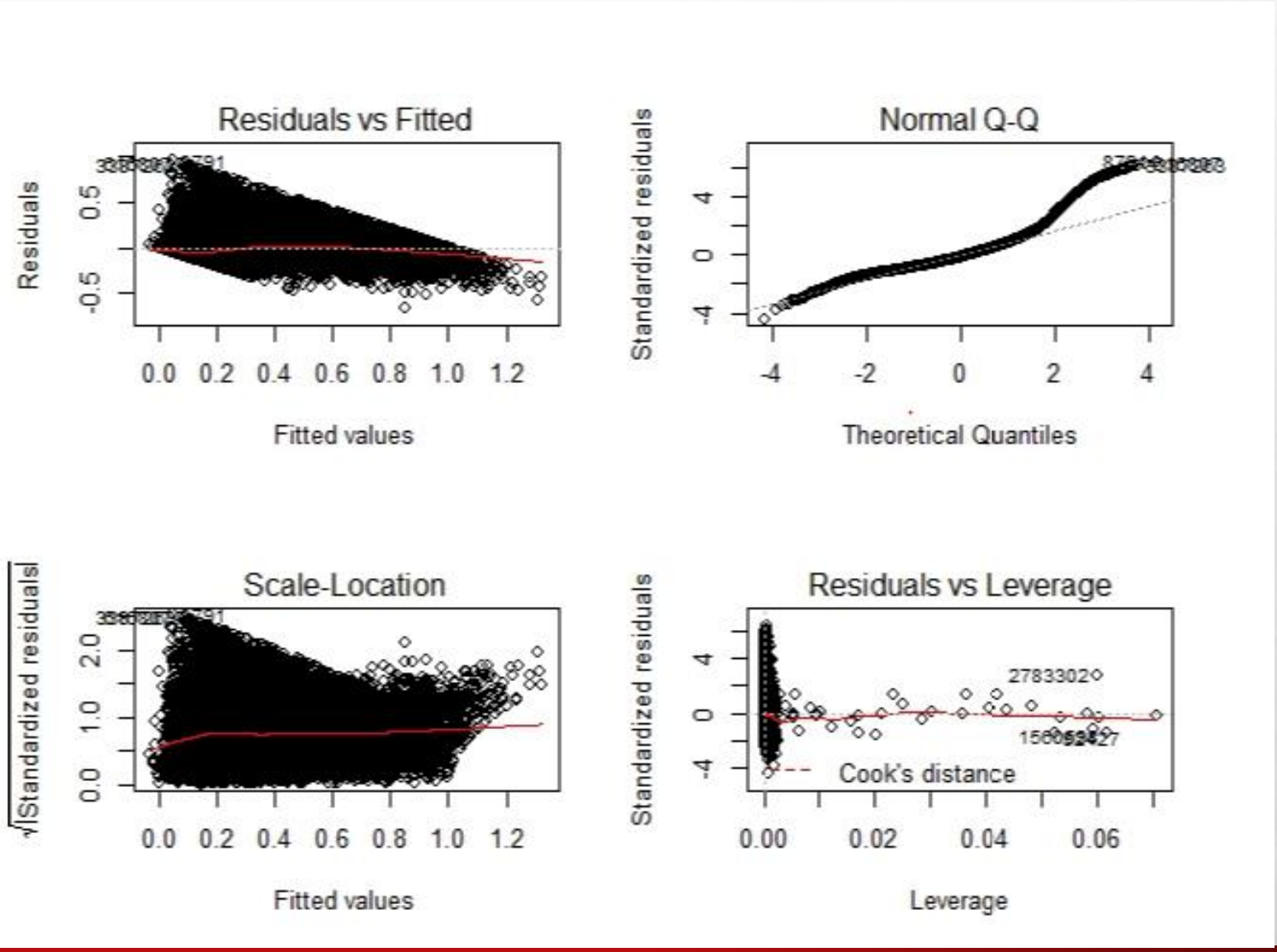
MLR OUTPUT WITHOUT PCA

```
Residual standard error: 0.1178 on 33183 degrees of freedom
Multiple R-squared:  0.7849,    Adjusted R-squared:  0.7848
F-statistic:  7121 on 17 and 33183 DF,  p-value: < 2.2e-16
```

RESULTS

MLR assumptions with PCA

```
par(mfrow=c(2,2))
plot(obj_fwd)
```



RESULTS

RMSE		
	Without PCA	With PCA
MLR	0.1162	0.1479
CART	0.1256	0.1273
k-NN	0.1160	0.1319

CONCLUSION

Analysis without applying PCA provides the best results for all the three algorithms.

MLR is not an optimal method to predict win place percentile as the assumptions of MLR does not meet in either of the case.

Based on the RMSE value k-NN with k = 11 is the best method to predict the win place percentile variable

PREDICTED OUTPUT

```
> head(df_output,34)
                Id         winPlacePerc
1  4125d337a9bc41 0.0210994094581172
2  8a35b912403f72  0.118658789158246
3  209943df673589  0.452947508854676
4  011ce2c63edda6   0.47935400822543
5  e865b7c0755bbe 0.0865101575176968
6  31faec823f564b  0.760342888276353
7  d1aed2cd37df1f  0.712297257821613
8  e9c9d620252c77  0.106515898862378
9  4fa2e0712efa15  0.282700405389399
10 93234b66565eef  0.241654342100448
11 c72dba44deeae7  0.580401924676827
12 2f30b13bf09bba  0.148551242971799
13 33fab8d8035f58 0.0700019494104264
14 0b2276e56128ef   0.33655708136644
15 34990f1d2378bd  0.239701090601804
16 3aa86db23adf23  0.077701295739267
17 4f22ab218e3ebd  0.234635477001815
18 51324ce6157f61  0.258170785297725
19 ad8e72c55070b1  0.469025704317451
20 5bd89dd6b03179  0.201024376983084
21 a8f0e6dd80cdac  0.477613045647615
22 fe2a790031dd15  0.385390098095539
23 31adbd1645e520  0.126201990626131
24 bc4d5b63a0019e   0.0640082342752
25 4a5b45bc5c0324   0.57950009519022
26 db4289b05a135f  0.489947786657775
27 40a90775dc0c0b  0.298227135327823
28 e3057bbf900782  0.320867812608204
29 295641fbbe6bfd 0.0231543226945885
30 f302dde0e3e487   0.39531421355708
31 386ef41cba8f09  0.267115585869294
32 f1f080e1c2f3e9   0.48324077641656
33 22ff8b3fc32fdc  0.197296916803907
34 c128d433b0e6c8  0.396629780646545
```

BEST STRATEGY