Double-click (or enter) to edit

PLAYSIMPLE GAMES ASSIGNMENT FOR DATA SCIENCE ROLE

-by Divya Disha

PROBLEM STATEMENT 2:-

```
#import the libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn


#loading the dataset as a panda dataframe
data=pd.read_csv('data.csv')
```

```
data.head()
```

	Unnamed: 0	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	...
0	2093518	a3e3cea9f2e314	c9f6eaa81245b5	a3d8693390584c	0	0	25.93	0.0	0.0	0.0	...
1	3701672	43e4be7ad72cc7	521836de101ee8	b65de9055327e0	0	0	36.71	0.0	0.0	0.0	...
2	2059784	9a68690e31fdff	4a72ffa2cebd90	d6aad3f9830e60	0	1	47.73	0.0	0.0	0.0	...
3	1302036	b147e1bd448fc4	9a8991656b3fea	d931c0932d8aca	0	0	0.00	0.0	0.0	0.0	...
4	297180	d818b4edd59612	eece87c8b846b3	ec2b5ed94baae3	0	2	100.00	0.0	0.0	3.0	...

5 rows × 30 columns

```
data.tail()
```

	Unnamed: 0	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals
14716	3850976	a2506a09d487c3	ed4b8c8879fb67	d5332000f4b96c	0	5	436.8	3.0	0.0	5.0
14717	72065	c70ba32de82766	5756966f67fe90	6cb8f61945b52e	0	0	134.3	1.0	0.0	0.0
14718	548903	18a72324140db7	ffead4bd54f2db	1dfa8603656924	0	0	0.0	0.0	0.0	0.0
14719	3705920	3b77e339fb5118	6f7799ed5bacb6	df9f3c2289a200	0	0	0.0	0.0	0.0	0.0
14720	454566	1505e43fade34c	a7a8db94ade615	185a15a75c0431	0	7	164.0	NaN	NaN	NaN

5 rows × 30 columns

DATA EXPLORATION

```
data.shape
```

```
(14721, 30)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14721 entries, 0 to 14720
Data columns (total 30 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Unnamed: 0       14721 non-null  int64  
 1   Id               14721 non-null  object 
 2   groupId          14721 non-null  object 
 3   matchId          14721 non-null  object 
 4   assists          14721 non-null  int64  
 5   boosts           14721 non-null  int64  
 6   damageDealt      14721 non-null  float64
 7   DBNOs            14720 non-null  float64
 8   headshotKills    14720 non-null  float64
 9   heals            14720 non-null  float64
 10  killPlace        14720 non-null  float64
 11  killPoints       14720 non-null  float64
 12  kills            14720 non-null  float64
 13  killStreaks      14720 non-null  float64
 14  longestKill      14720 non-null  float64
 15  matchDuration    14720 non-null  float64
 16  matchType        14720 non-null  object 
 17  maxPlace         14720 non-null  float64
 18  numGroups        14720 non-null  float64
 19  rankPoints       14720 non-null  float64
 20  revives          14720 non-null  float64
 21  rideDistance     14720 non-null  float64
 22  roadKills        14720 non-null  float64
 23  swimDistance     14720 non-null  float64
 24  teamKills        14720 non-null  float64
 25  vehicleDestroys  14720 non-null  float64
 26  walkDistance     14720 non-null  float64
 27  weaponsAcquired  14720 non-null  float64
 28  winPoints        14720 non-null  float64
 29  winPlacePerc     14720 non-null  float64
dtypes: float64(23), int64(3), object(4)
memory usage: 3.4+ MB
```

```
data.describe()
```

	Unnamed: 0	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	killP
count	1.472100e+04	14721.000000	14721.000000	14721.000000	14720.000000	14720.000000	14720.000000	14720.000000	14720.0
mean	2.229761e+06	0.231370	1.112832	130.456614	0.657541	0.223981	1.368274	47.688043	499.5
std	1.285314e+06	0.588612	1.722249	168.397297	1.148900	0.593837	2.689172	27.542206	628.9
min	2.530000e+02	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.0
25%	1.115923e+06	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	24.000000	0.0
50%	2.227927e+06	0.000000	0.000000	84.830000	0.000000	0.000000	0.000000	47.000000	0.0
75%	3.345983e+06	0.000000	2.000000	186.000000	1.000000	0.000000	2.000000	72.000000	1170.2
max	4.446806e+06	12.000000	15.000000	2498.000000	23.000000	12.000000	42.000000	100.000000	2023.0

8 rows × 26 columns

```
data.isnull()
```

	Unnamed: 0	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	...	revives	rideDistanc
0	False	False	False	False	False	False	False	False	False	False	...	False	Fals
1	False	False	False	False	False	False	False	False	False	False	...	False	Fals
2	False	False	False	False	False	False	False	False	False	False	...	False	Fals
3	False	False	False	False	False	False	False	False	False	False	...	False	Fals
4	False	False	False	False	False	False	False	False	False	False	...	False	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	
14716	False	False	False	False	False	False	False	False	False	False	...	False	Fals
14717	False	False	False	False	False	False	False	False	False	False	...	False	Fals
14718	False	False	False	False	False	False	False	False	False	False	...	False	Fals
14719	False	False	False	False	False	False	False	False	False	False		False	Fals

```
data.isnull().sum()
```

```
Unnamed: 0         0
Id                 0
groupId            0
matchId            0
assists            0
boosts             0
damageDealt        0
DBNOs              1
headshotKills      1
heals              1
killPlace          1
killPoints         1
kills              1
killStreaks        1
longestKill        1
matchDuration      1
matchType          1
maxPlace           1
numGroups          1
rankPoints         1
revives            1
rideDistance       1
roadKills          1
swimDistance       1
teamKills          1
vehicleDestroys    1
walkDistance       1
weaponsAcquired    1
winPoints          1
winPlacePerc       1
dtype: int64
```

This since this a PubG player's analytics ,some data can be null as it means the player did not score at all for these features-hence we remove the null values by 0.

```
new_data=data.fillna(0)
```

```
new_data.isnull()
```

	Unnamed: 0	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	...	revives	rideDistanc
0	False	False	False	False	False	False	False	False	False	False	...	False	Fals
1	False	False	False	False	False	False	False	False	False	False	...	False	Fals
2	False	False	False	False	False	False	False	False	False	False	...	False	Fals
3	False	False	False	False	False	False	False	False	False	False	...	False	Fals
4	False	False	False	False	False	False	False	False	False	False	...	False	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	
14716	False	False	False	False	False	False	False	False	False	False	...	False	Fals
14717	False	False	False	False	False	False	False	False	False	False	...	False	Fals

```
new_data.isnull().sum()
```

```
Unnamed: 0         0
Id                 0
groupId            0
matchId            0
assists            0
boosts             0
damageDealt        0
DBNOs              0
headshotKills      0
heals              0
killPlace          0
killPoints         0
kills              0
killStreaks        0
longestKill        0
matchDuration      0
matchType          0
maxPlace           0
numGroups          0
rankPoints         0
revives            0
rideDistance       0
roadKills          0
swimDistance       0
teamKills          0
vehicleDestroys    0
walkDistance       0
weaponsAcquired    0
winPoints          0
winPlacePerc       0
dtype: int64
```

```
#dropping the unnecesarry columns
new_data= new_data.drop(['Unnamed: 0'], axis=1)
```

```
new_data.head()
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs	headshotKills	heals	killPlace	...
0	a3e3cea9f2e314	c9f6eaa81245b5	a3d8693390584c	0	0	25.93	0.0	0.0	0.0	84.0	...
1	43e4be7ad72cc7	521836de101ee8	b65de9055327e0	0	0	36.71	0.0	0.0	0.0	57.0	...
2	9a68690e31fdff	4a72ffa2cebd90	d6aad3f9830e60	0	1	47.73	0.0	0.0	0.0	50.0	...
3	b147e1bd448fc4	9a8991656b3fea	d931c0932d8aca	0	0	0.00	0.0	0.0	0.0	56.0	...
4	d818b4edd59612	eece87c8b846b3	ec2b5ed94baae3	0	2	100.00	0.0	0.0	3.0	31.0	...

5 rows × 29 columns

```
# Function to detect outliers in every feature
def detect_outliers(new_data):
    cols = list(new_data)
    outliers = pd.DataFrame(columns = ['Feature', 'Number of Outliers'])
    for column in cols:
        if column in new_data.select_dtypes(include=np.number).columns:
            q1 = new_data[column].quantile(0.25)
            q3 = new_data[column].quantile(0.75)
            iqr = q3 - q1
            fence_low = q1 - (1.5*iqr)
            fence_high = q3 + (1.5*iqr)
            outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_
    return outliers

detect_outliers(new_data)
```

```
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
<ipython-input-15-2cc6f31f0d65>:12: FutureWarning: The frame.append method is deprecated and will be removed from pandas in
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
```

As we can see there are majority of columns having outliers.

```
  outliers = outliers.append({'Feature':column, 'Number of Outliers':new_data.loc[(new_data[column] < fence_low) | (new_data
```

```
#Features selection
new_data=new_data[['Id','rankPoints','winPlacePerc','killPlace','kills','assists','matchDuration','revives','DBNOs','damageDealt',
new_data.head()
```

	Id	rankPoints	winPlacePerc	killPlace	kills	assists	matchDuration	revives	DBNOs	damageDealt	headshotK
0	a3e3cea9f2e314	-1.0	0.0667	84.0	0.0	0	1403.0	0.0	0.0	25.93	
1	43e4be7ad72cc7	-1.0	0.5862	57.0	0.0	0	1971.0	0.0	0.0	36.71	
2	9a68690e31fdff	1516.0	0.8105	50.0	0.0	0	1741.0	0.0	0.0	47.73	
3	b147e1bd448fc4	-1.0	0.5556	56.0	0.0	0	1738.0	1.0	0.0	0.00	
4	d818b4edd59612	1482.0	0.6429	31.0	1.0	0	2193.0	0.0	0.0	100.00	

5 heals 1065

rankPoints feature:-**Elo-like ranking of players. This ranking is inconsistent and is being deprecated in the API's next version, so use with caution. Value of -1 takes the place of "None".**

```
#replacing the rankPoints having values of '-1' by '0'
new_data['rankPoints'] = new_data['rankPoints'].replace(-1, 0)
new_data.head()
```

```
<ipython-input-17-a68b0c5c0bb4>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
  new_data['rankPoints'] = new_data['rankPoints'].replace(-1, 0)
```

	Id	rankPoints	winPlacePerc	killPlace	kills	assists	matchDuration	revives	DBNOs	damageDealt	headshotK
0	a3e3cea9f2e314	0.0	0.0667	84.0	0.0	0	1403.0	0.0	0.0	25.93	
1	43e4be7ad72cc7	0.0	0.5862	57.0	0.0	0	1971.0	0.0	0.0	36.71	
2	9a68690e31fdff	1516.0	0.8105	50.0	0.0	0	1741.0	0.0	0.0	47.73	
3	b147e1bd448fc4	0.0	0.5556	56.0	0.0	0	1738.0	1.0	0.0	0.00	
4	d818b4edd59612	1482.0	0.6429	31.0	1.0	0	2193.0	0.0	0.0	100.00	

```
new_data.tail()
```

	Id	rankPoints	winPlacePerc	killPlace	kills	assists	matchDuration	revives	DBNOs	damageDealt	heads
14716	a2506a09d487c3	1452.0	1.0000	2.0	4.0	0	1872.0	2.0	3.0	436.8	
14717	c70ba32de82766	1421.0	0.1071	39.0	1.0	0	1377.0	0.0	1.0	134.3	
14718	18a72324140db7	1485.0	0.1915	75.0	0.0	0	1389.0	0.0	0.0	0.0	
14719	3b77e339fb5118	1519.0	0.5833	58.0	0.0	0	1377.0	0.0	0.0	0.0	
14720	1505e43fade34c	0.0	0.0000	0.0	0.0	0	0.0	0.0	0.0	164.0	

Now , we take into consideration ony some features from dataset and calculate KPIs - Key Performance Indicators , We assign a score to each KPI based on their significance on the PubG game .

```
#assigning a score(1-5) to each feature based on rank
rankPoints =5
winPlacePerc =5
killPlace= 4
kills= 4
assists= 3
matchDuration= 3
revives= 3
DBNOs= 3
damageDealt= 2
headshotKills= 2
killStreaks= 2
rideDistance= 1
walkDistance= 1
```

Scaling/Standardization of Data

Since our features have outliers, we will use Z-Score scaling to standardize/scale our data as Min-Max scaling could be influenced heavily by extreme values.

```
# Calculate min and max values of the column
min_value = new_data['rankPoints'].min()
max_value = new_data['rankPoints'].max()

# Apply Min-Max scaling to the column
new_data['rankPoints_scaled'] = (new_data['rankPoints'] - min_value) / (max_value - min_value)
```

```
new_data['rankPoints_scaled']
```

```
0        0.000000
1        0.000000
2        0.365389
3        0.000000
4        0.357195
           ...
14716    0.349964
14717    0.342492
14718    0.357918
14719    0.366112
14720    0.000000
Name: rankPoints_scaled, Length: 14721, dtype: float64
```

```
new_data['rankPoints_scaled'].describe()
```

```
count    14721.000000
mean         0.217486
std          0.177442
min          0.000000
25%          0.000000
50%          0.348759
75%          0.361533
max          1.000000
Name: rankPoints_scaled, dtype: float64
```

```
mean_value_1 = new_data['rankPoints'].mean()
std_value_1 = new_data['rankPoints'].std()

# Apply Z-score scaling and convert the values to a range of 0 to 100
new_data['rankPoints_zscore'] = ((new_data['rankPoints'] - mean_value_1) / std_value_1) * 50 + 50
```

```
new_data['rankPoints_zscore']
```

```
0       -11.283751
1       -11.283751
2        91.676702
3       -11.283751
4        89.367563
           ...
14716    87.330087
14717    85.224695
14718    89.571310
14719    91.880450
14720   -11.283751
Name: rankPoints_zscore, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value2 = new_data['winPlacePerc'].min()
max_value2 = new_data['winPlacePerc'].max()

# Apply Min-Max scaling to the column
new_data['winPlacePerc_scaled'] = (new_data['winPlacePerc'] - min_value2) / (max_value - min_value2)
```

```
new_data['winPlacePerc']
```

```
0        0.0667
1        0.5862
2        0.8105
3        0.5556
4        0.6429
          ...
14716    1.0000
14717    0.1071
14718    0.1915
14719    0.5833
14720    0.0000
Name: winPlacePerc, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value3 = new_data['killPlace'].min()
max_value3 = new_data['killPlace'].max()

# Apply Min-Max scaling to the column
new_data['killPlace_scaled'] = (new_data['killPlace'] - min_value3) / (max_value3 - min_value3)

new_data['killPlace_scaled']
```

```
0        0.84
1        0.57
2        0.50
3        0.56
4        0.31
           ...
14716    0.02
14717    0.39
14718    0.75
14719    0.58
14720    0.00
Name: killPlace_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value4 = new_data['kills'].min()
max_value4 = new_data['kills'].max()

# Apply Min-Max scaling to the column
new_data['kills_scaled'] = (new_data['kills'] - min_value4) / (max_value4 - min_value4)

new_data['kills_scaled']
```

```
0        0.000000
1        0.000000
2        0.000000
3        0.000000
4        0.038462
             ...
14716    0.153846
14717    0.038462
14718    0.000000
14719    0.000000
14720    0.000000
Name: kills_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value5 = new_data['assists'].min()
max_value5= new_data['assists'].max()

# Apply Min-Max scaling to the column
new_data['assists_scaled'] = (new_data['assists'] - min_value5) / (max_value5 - min_value5)

new_data['assists_scaled']
```

```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
          ...
14716    0.0
14717    0.0
14718    0.0
14719    0.0
14720    0.0
Name: assists_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value6 = new_data['matchDuration'].min()
max_value6= new_data['matchDuration'].max()

# Apply Min-Max scaling to the column
new_data['matchDuration_scaled'] = (new_data['matchDuration'] - min_value6) / (max_value6 - min_value6)
```

```
new_data['matchDuration_scaled']
```

```
0        0.632837
1        0.889039
2        0.785295
3        0.783942
4        0.989175
           ...
14716    0.844384
14717    0.621110
14718    0.626522
14719    0.621110
14720    0.000000
Name: matchDuration_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value7 = new_data['revives'].min()
max_value7= new_data['revives'].max()

# Apply Min-Max scaling to the column
new_data['revives_scaled'] = (new_data['revives'] - min_value7) / (max_value7 - min_value7)
```

```
new_data['revives_scaled']
```

```
0        0.000000
1        0.000000
2        0.000000
3        0.142857
4        0.000000
           ...
14716    0.285714
14717    0.000000
14718    0.000000
14719    0.000000
14720    0.000000
Name: revives_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value8 = new_data['DBNOs'].min()
max_value8= new_data['DBNOs'].max()

# Apply Min-Max scaling to the column
new_data['DBNOs_scaled'] = (new_data['DBNOs'] - min_value8) / (max_value8 - min_value8)
```

```
new_data['DBNOs_scaled']
```

```
0        0.000000
1        0.000000
2        0.000000
3        0.000000
4        0.000000
           ...
14716    0.130435
14717    0.043478
14718    0.000000
14719    0.000000
14720    0.000000
Name: DBNOs_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value9 = new_data['damageDealt'].min()
max_value9 = new_data['damageDealt'].max()

# Apply Min-Max scaling to the column
new_data['damageDealt_scaled'] = (new_data['damageDealt'] - min_value9) / (max_value9 - min_value9)
```

```
new_data['damageDealt_scaled']
```

```
0        0.010380
1        0.014696
2        0.019107
```

```
3        0.000000
4        0.040032
           ...
14716    0.174860
14717    0.053763
14718    0.000000
14719    0.000000
14720    0.065653
Name: damageDealt_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_valuex = new_data['headshotKills'].min()
max_valuex = new_data['headshotKills'].max()

# Apply Min-Max scaling to the column
new_data['headshotKills_scaled'] = (new_data['headshotKills'] - min_valuex) / (max_valuex - min_valuex)

new_data['headshotKills_scaled']
```

```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
         ...
14716    0.0
14717    0.0
14718    0.0
14719    0.0
14720    0.0
Name: headshotKills_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_valuey = new_data['killStreaks'].min()
max_valuey = new_data['killStreaks'].max()

# Apply Min-Max scaling to the column
new_data['killStreaks_scaled'] = (new_data['killStreaks'] - min_valuey) / (max_valuey - min_valuey)

new_data['killStreaks_scaled']
```

```
0        0.0
1        0.0
2        0.0
3        0.0
4        0.2
         ...
14716    0.4
14717    0.2
14718    0.0
14719    0.0
14720    0.0
Name: killStreaks_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_valuez = new_data['rideDistance'].min()
max_valuez = new_data['rideDistance'].max()

# Apply Min-Max scaling to the column
new_data['rideDistance_scaled'] = (new_data['rideDistance'] - min_valuez) / (max_valuez - min_valuez)

new_data['rideDistance_scaled']
```

```
0        0.000000
1        0.000000
2        0.290754
3        0.000000
4        0.065320
           ...
```

```
14716    0.147795
14717    0.000000
14718    0.000000
14719    0.011422
14720    0.000000
Name: rideDistance_scaled, Length: 14721, dtype: float64
```

```
# Calculate min and max values of the column
min_value = new_data['walkDistance'].min()
max_value = new_data['walkDistance'].max()

# Apply Min-Max scaling to the column
new_data['walkDistance_scaled'] = (new_data['walkDistance'] - min_value) / (max_value - min_value)

new_data['walkDistance_scaled']
```

```
0        0.016316
1        0.206837
2        0.088429
3        0.004185
4        0.185593
           ...
14716    0.525073
14717    0.012768
14718    0.008138
14719    0.069783
14720    0.000000
Name: walkDistance_scaled, Length: 14721, dtype: float64
```

```
new_df = pd.DataFrame()

# New dataframe computed by multiplying scores of each KPI with their value
new_df['Id']=new_data['Id']
new_df['rankPoints'] = new_data['rankPoints_scaled'] * 5
new_df['winPlacePerc'] = new_data['winPlacePerc_scaled']*5
new_df['killPlace'] = new_data['killPlace_scaled'] * 4
new_df['kills']= new_data['kills_scaled'] * 4
new_df['assists']= new_data['assists_scaled'] * 3
new_df['matchDuration']= new_data['matchDuration_scaled'] * 3
new_df['revives']= new_data['revives_scaled'] * 3
new_df['DBNOs']= new_data['DBNOs_scaled'] * 3
new_df['damageDealt']= new_data['damageDealt_scaled'] * 2
new_df['headshotKills']= new_data['headshotKills_scaled'] * 2
new_df['killStreaks']= new_data['killStreaks'] * 2
new_df['rideDistance']= new_data['rideDistance_scaled'] * 1
new_df['walkDistance']= new_data['walkDistance_scaled'] * 1

new_df.head()
```

	Id	rankPoints	winPlacePerc	killPlace	kills	assists	matchDuration	revives	DBNOs	damageDealt	headsh
0	a3e3cea9f2e314	0.000000	0.000080	3.36	0.000000	0.0	1.898512	0.000000	0.0	0.020761	
1	43e4be7ad72cc7	0.000000	0.000706	2.28	0.000000	0.0	2.667118	0.000000	0.0	0.029392	
2	9a68690e31fdff	1.826946	0.000977	2.00	0.000000	0.0	2.355886	0.000000	0.0	0.038215	
3	b147e1bd448fc4	0.000000	0.000670	2.24	0.000000	0.0	2.351827	0.428571	0.0	0.000000	
4	d818b4edd59612	1.785973	0.000775	1.24	0.153846	0.0	2.967524	0.000000	0.0	0.080064	

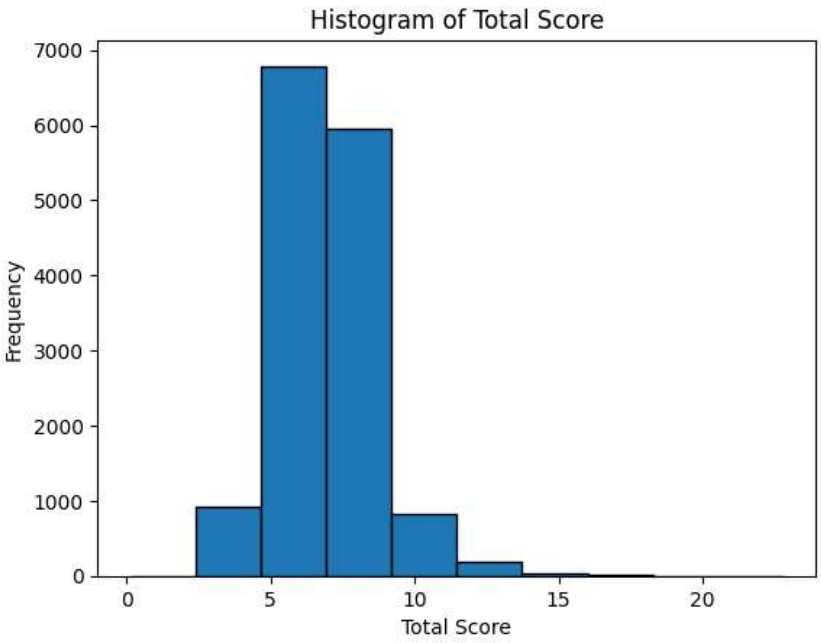
```
new_df['Total_score'] = new_df['rankPoints']+new_df['winPlacePerc']+new_df['killPlace']+new_df['kills']+new_df['assists']+new_df[
new_df['Total_score']
```

```
0         5.295669
1         5.184053
2         6.601207
3         5.025253
4         8.479094
           ...
14716    11.250597
14717     7.540494
14718     6.677523
14719     6.095799
14720     0.131305
Name: Total_score, Length: 14721, dtype: float64
```

```
new_df.head()
```

ce	kills	assists	matchDuration	revives	DBNOs	damageDealt	headshotKills	killStreaks	rideDistance	walkDistance	Tota
36	0.000000	0.0	1.898512	0.000000	0.0	0.020761	0.0	0.0	0.000000	0.016316	
28	0.000000	0.0	2.667118	0.000000	0.0	0.029392	0.0	0.0	0.000000	0.206837	
00	0.000000	0.0	2.355886	0.000000	0.0	0.038215	0.0	0.0	0.290754	0.088429	
24	0.000000	0.0	2.351827	0.428571	0.0	0.000000	0.0	0.0	0.000000	0.004185	
24	0.153846	0.0	2.967524	0.000000	0.0	0.080064	0.0	2.0	0.065320	0.185593	

```
plt.hist(new_df['Total_score'], bins=10, edgecolor='black')

# Add labels and title
plt.xlabel('Total Score')
plt.ylabel('Frequency')
plt.title('Histogram of Total Score')

# Display the plot
plt.show()
```




```
plt.scatter(new_df['Total_score'],)

# Add labels and title
plt.xlabel('Total Score')
plt.ylabel('Frequency')
```

```
plt.title('Histogram of Total Score')

# Display the plot
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-74-2507a486d007> in <cell line: 1>()
----> 1 plt.scatter(new_df['Total_score'])
      2
      3 # Add labels and title
      4 plt.xlabel('Total Score')
      5 plt.ylabel('Frequency')

TypeError: scatter() missing 1 required positional argument: 'y'
```

SEARCH STACK OVERFLOW

```
new_df['Total_score'].min()
```

```
0.1313050440352282
```

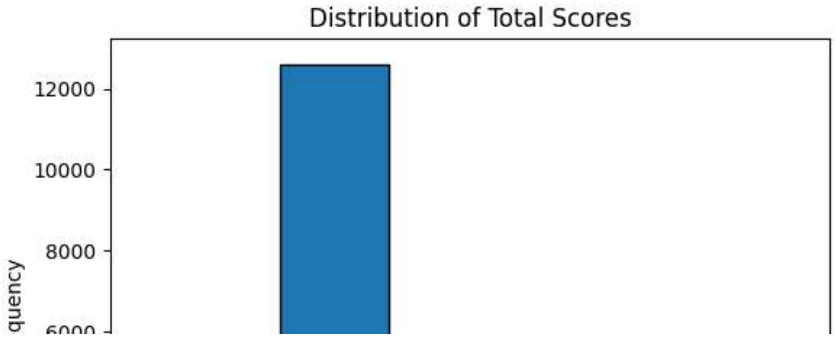
```
new_df['Total_score'].max()
```

```
22.812400049689696
```

```
new_df['Total_score'].describe()
```

```
count    14721.000000
mean         6.879924
std          1.609104
min          0.131305
25%          5.748796
50%          6.838185
75%          7.723998
max         22.812400
Name: Total_score, dtype: float64
```

```
bins = [0, 5, 10, 15, 20, 25]

# Use the 'cut' function to assign total scores to bins
score_bins = pd.cut(new_df['Total_score'], bins=bins, right=False)

# Get the frequency count of scores in each bin
bin_counts = score_bins.value_counts().sort_index()

# Plot the bar graph for the distribution of total scores
plt.bar(bin_counts.index.astype(str), bin_counts, edgecolor='black')

# Add labels and title
plt.xlabel('Total Score Range')
plt.ylabel('Frequency')
plt.title('Distribution of Total Scores')

# Display the plot
plt.show()
```



So we see from the graphs that the 'Total score' is distributed between the range 0 to 25.

```
total_score_range = new_df['Total_score'].max() - new_df['Total_score'].min()
print(total_score_range)
```

```
22.681095005654466
```

```
new_df['Total_score'].max()
```

```
22.812400049689696
```

```
new_df['Total_score'].min()
```

```
0.1313050440352282
```

Divide the 'Total score' column into 3 categories - Noob, Mid-level,Advanced based on range of Total Score(s)

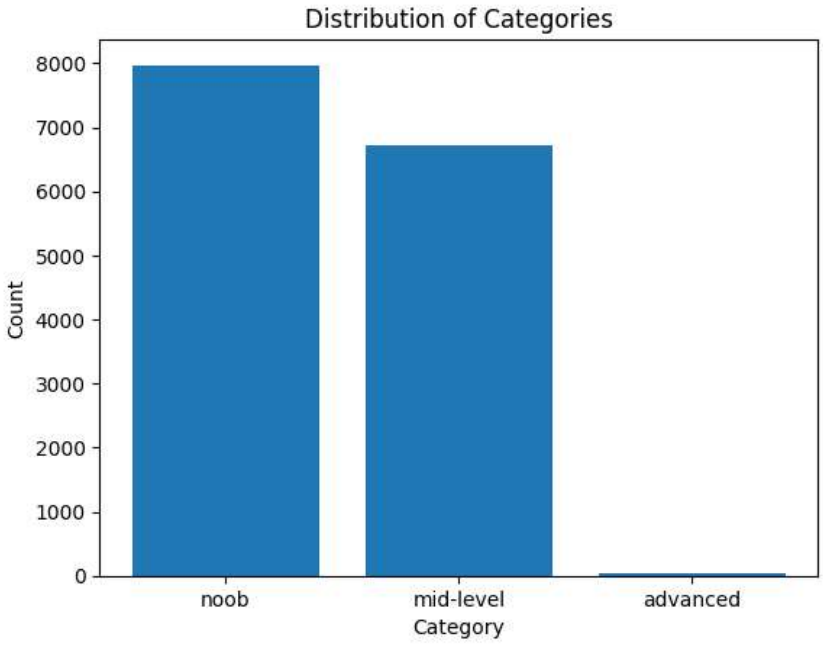
```
bins = [0.0, 7.0, 14.0,22.812400049689696 ]  # Adjust the bin ranges based on your desired cutoffs
labels = ['noob', 'mid-level', 'advanced']

# Create a new column to hold the categories
new_df['score_category'] = pd.cut(new_df['Total_score'], bins=bins, labels=labels)

new_df.head()

#new_df = new_df.drop('player_category', axis=1)

#new_df.head()
```

ists	matchDuration	revives	DBNOs	damageDealt	headshotKills	killStreaks	rideDistance	walkDistance	Total_score	score_c
0.0	1.898512	0.000000	0.0	0.020761	0.0	0.0	0.000000	0.016316	5.295669	
0.0	2.667118	0.000000	0.0	0.029392	0.0	0.0	0.000000	0.206837	5.184053	
0.0	2.355886	0.000000	0.0	0.038215	0.0	0.0	0.290754	0.088429	6.601207	
0.0	2.351827	0.428571	0.0	0.000000	0.0	0.0	0.000000	0.004185	5.025253	
0.0	2.967524	0.000000	0.0	0.080064	0.0	2.0	0.065320	0.185593	8.479094	

```
new_df.tail()
```

ts	matchDuration	revives	DBNOs	damageDealt	headshotKills	killStreaks	rideDistance	walkDistance	Total_score	score_c
.0	2.533153	0.857143	0.391304	0.349720	0.0	4.0	0.147795	0.525073	11.250597	
.0	1.863329	0.000000	0.130435	0.107526	0.0	2.0	0.000000	0.012768	7.540494	
.0	1.879567	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.008138	6.677523	
.0	1.863329	0.000000	0.000000	0.000000	0.0	0.0	0.011422	0.069783	6.095799	
.0	0.000000	0.000000	0.000000	0.131305	0.0	0.0	0.000000	0.000000	0.131305	

```
# Count the occurrences of each category
category_counts =new_df['score_category'].value_counts()

# Plot the bar graph for the distribution of categories
plt.bar(category_counts.index, category_counts)

# Add labels and title
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Distribution of Categories')

# Display the plot
plt.show()
```




```
count_advanced = new_df['score_category'].value_counts()['advanced']

print("Occurrences of 'advanced':", count_advanced)
```

```
Occurrences of 'advanced': 36
```

```
pair_df = pd.DataFrame()

pair_df['Id']=new_df['Id']
pair_df['Total_score']=new_df['Total_score']
pair_df['score_category']=new_df['score_category']
```

```
pair_df.head()
```

	Id	Total_score	score_category
0	a3e3cea9f2e314	5.295669	noob
1	43e4be7ad72cc7	5.184053	noob
2	9a68690e31fdff	6.601207	noob
3	b147e1bd448fc4	5.025253	noob
4	d818b4edd59612	8.479094	mid-level

```
# Create separate DataFrames for each category
noob_df = pair_df[pair_df['score_category'] == 'noob']
mid_level_df = pair_df[pair_df['score_category'] == 'mid-level']
advanced_df = pair_df[pair_df['score_category'] == 'advanced']



#Randomly shuffle the rows within each category DataFrame
def random_pairing(df):
    df_shuffled = df.sample(frac=1, random_state=42).reset_index(drop=True)
    return df_shuffled


shuffled_noob_df = random_pairing(noob_df)
shuffled_mid_level_df = random_pairing(mid_level_df)
shuffled_advanced_df = random_pairing(advanced_df)



#Ensure each player is paired only once within their category
shuffled_noob_df['paired_with'] = shuffled_noob_df['Id'].shift(-1)
shuffled_mid_level_df['paired_with'] = shuffled_mid_level_df['Id'].shift(-1)
shuffled_advanced_df['paired_with'] = shuffled_advanced_df['Id'].shift(-1)


# The last player in each category should be paired with the first player to complete the loop
shuffled_noob_df.loc[shuffled_noob_df.index[-1], 'paired_with'] = shuffled_noob_df.loc[0, 'Id']
shuffled_mid_level_df.loc[shuffled_mid_level_df.index[-1], 'paired_with'] = shuffled_mid_level_df.loc[0, 'Id']
shuffled_advanced_df.loc[shuffled_advanced_df.index[-1], 'paired_with'] = shuffled_advanced_df.loc[0, 'Id']


#Combine the pairings for all three categories into a single DataFrame
paired_data = pd.concat([shuffled_noob_df, shuffled_mid_level_df, shuffled_advanced_df], ignore_index=True)



print(paired_data[['Id', 'paired_with','score_category']])
```

```
                    Id     paired_with score_category
0      5b353a57dfbbf4  d791f20abcad10           noob
1      d791f20abcad10  9d242f3b5c351b           noob
2      9d242f3b5c351b  8d3000a4eb891f           noob
3      8d3000a4eb891f  dfb0a811073f5a           noob
4      dfb0a811073f5a  cfac06e40f08ef           noob
...               ...             ...            ...
14716  233b2aea921360  bc01d3c29c6a02       advanced
14717  bc01d3c29c6a02  78991f8112e0f3       advanced
14718  78991f8112e0f3  7dac1dc4eda3f7       advanced
14719  7dac1dc4eda3f7  a8f79ac6b80188       advanced
14720  a8f79ac6b80188  27f69212f3c2a0       advanced

[14721 rows x 3 columns]
```

```
#Input a player's player_id for whom you want to find the paired player
input_Id = str('5b353a57dfbbf4')

#Look up the row in paired_data that corresponds to the given player_id
```

```
paired_row = paired_data[paired_data['Id'] == input_Id]

if len(paired_row) == 0:
    print("Player  not found.")
else:
    #  Retrieve the paired_with value, which represents the player_id of the paired player within the same category
    paired_with_player_id = paired_row['paired_with'].iloc[0]
    print("The paired player  is player", paired_with_player_id)
```

```
 The paired player  is player d791f20abcad10
```

✓ 0s completed at 11:34 PM