In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('C:\\Users\kants\Downloads/nba_2013.csv')
pd.set_option('display.max_columns',None)
df.head(3)
```

Out[2]:

	player	pos	age	bref_team_id	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	fta	ft.	orb	drb
0	Quincy Acy	SF	23	TOT	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	53	0.660	72	144
1	Steven Adams	C	20	OKC	81	20	1197	93	185	0.503	0	0	NaN	93	185	0.502703	0.503	79	136	0.581	142	190
2	Jeff Adrien	PF	27	TOT	53	12	961	143	275	0.520	0	0	NaN	143	275	0.520000	0.520	76	119	0.639	102	204

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 481 entries, 0 to 480
Data columns (total 31 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   player        481 non-null    object
 1   pos           481 non-null    object
 2   age           481 non-null    int64
 3   bref_team_id  481 non-null    object
 4   g             481 non-null    int64
 5   gs            481 non-null    int64
 6   mp            481 non-null    int64
 7   fg            481 non-null    int64
 8   fga           481 non-null    int64
 9   fg.           479 non-null    float64
 10  x3p           481 non-null    int64
 11  x3pa          481 non-null    int64
 12  x3p.          414 non-null    float64
 13  x2p           481 non-null    int64
 14  x2pa          481 non-null    int64
 15  x2p.          478 non-null    float64
 16  efg.          479 non-null    float64
 17  ft            481 non-null    int64
 18  fta           481 non-null    int64
 19  ft.           461 non-null    float64
 20  orb           481 non-null    int64
 21  drb           481 non-null    int64
 22  trb           481 non-null    int64
 23  ast           481 non-null    int64
 24  stl           481 non-null    int64
 25  blk           481 non-null    int64
 26  tov           481 non-null    int64
 27  pf            481 non-null    int64
 28  pts           481 non-null    int64
 29  season        481 non-null    object
 30  season_end    481 non-null    int64
dtypes: float64(5), int64(22), object(4)
memory usage: 116.6+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	
count	481.000000	481.000000	481.000000	481.000000	481.000000	481.000000	479.000000	481.000000	481.000000	414.000000	481.000000	4
mean	26.509356	53.253638	25.571726	1237.386694	192.881497	424.463617	0.436436	39.613306	110.130977	0.285111	153.268191	3
std	4.198265	25.322711	29.658465	897.258840	171.832793	368.850833	0.098672	50.855639	132.751732	0.157633	147.223161	2
min	19.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	23.000000	32.000000	0.000000	388.000000	47.000000	110.000000	0.400500	0.000000	3.000000	0.234355	31.000000	
50%	26.000000	61.000000	10.000000	1141.000000	146.000000	332.000000	0.438000	16.000000	48.000000	0.330976	110.000000	2
75%	29.000000	76.000000	54.000000	2016.000000	307.000000	672.000000	0.479500	68.000000	193.000000	0.375000	230.000000	4
max	39.000000	83.000000	82.000000	3122.000000	849.000000	1688.000000	1.000000	261.000000	615.000000	1.000000	706.000000	14

In [5]:
```
df.isna().sum()
```

Out[5]:
```
player           0
pos              0
age              0
bref_team_id     0
g                0
gs               0
mp               0
fg               0
fga              0
fg.              2
x3p              0
x3pa             0
x3p.            67
x2p              0
x2pa             0
x2p.             3
efg.             2
ft               0
fta              0
ft.             20
orb              0
drb              0
trb              0
ast              0
stl              0
blk              0
tov              0
pf               0
pts              0
season           0
season_end       0
dtype: int64
```

In [6]:
```
df['fg.'].fillna(df['fg.'].median(),inplace=True)
df['x3p.'].fillna(df['x3p.'].mean(),inplace=True)
df['x2p.'].fillna(df['x2p.'].median(),inplace=True)
df['efg.'].fillna(df['efg.'].mean(),inplace=True)
df['ft.'].fillna(df['ft.'].median(),inplace=True)
```

In [7]:
```
df.drop('player',axis=1,inplace=True)
df.drop('bref_team_id',axis=1,inplace=True)
df.drop('season',axis=1,inplace=True)
df.drop('season_end', axis=1, inplace=True)
```

In [8]:
```
df.head(3)
```

Out[8]:

	pos	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	fta	ft.	orb	drb	trb	ast	stl	blk	tov
0	SF	23	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	53	0.660	72	144	216	28	23	26	30
1	C	20	81	20	1197	93	185	0.503	0	0	0.285111	93	185	0.502703	0.503	79	136	0.581	142	190	332	43	40	57	71
2	PF	27	53	12	961	143	275	0.520	0	0	0.285111	143	275	0.520000	0.520	76	119	0.639	102	204	306	38	24	36	39

In [11]:
```
dummy = pd.get_dummies(df['pos'],prefix= 'pos',drop_first=True)
dummy.head(3)
```

Out[11]:

	pos_F	pos_G	pos_PF	pos_PG	pos_SF	pos_SG
0	0	0	0	0	1	0
1	0	0	0	0	0	0
2	0	0	1	0	0	0

In [12]:
```
df.drop('pos',axis=1,inplace=True)
df.head(3)
```

Out[12]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	fta	ft.	orb	drb	trb	ast	stl	blk	tov	pf
0	23	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	53	0.660	72	144	216	28	23	26	30	122
1	20	81	20	1197	93	185	0.503	0	0	0.285111	93	185	0.502703	0.503	79	136	0.581	142	190	332	43	40	57	71	203
2	27	53	12	961	143	275	0.520	0	0	0.285111	143	275	0.520000	0.520	76	119	0.639	102	204	306	38	24	36	39	108

In [13]:
```
df2 = pd.concat([df,dummy],axis=1)
df2.head(2)
```

Out[13]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	fta	ft.	orb	drb	trb	ast	stl	blk	tov	pf
0	23	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	53	0.660	72	144	216	28	23	26	30	122
1	20	81	20	1197	93	185	0.503	0	0	0.285111	93	185	0.502703	0.503	79	136	0.581	142	190	332	43	40	57	71	203

In [14]:
```
feature = df2.drop('pts',axis=1)
feature.head(2)
```

Out[14]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	fta	ft.	orb	drb	trb	ast	stl	blk	tov	pf
0	23	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	53	0.660	72	144	216	28	23	26	30	122
1	20	81	20	1197	93	185	0.503	0	0	0.285111	93	185	0.502703	0.503	79	136	0.581	142	190	332	43	40	57	71	203

In [16]:
```
target = df2['pts']
target.head(2)
```

Out[16]:
```
0    171
1    265
Name: pts, dtype: int64
```

In [17]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(feature,target,test_size=0.3,random_state=42)
```

In [18]:
```
from sklearn.neighbors import KNeighborsRegressor
knr = KNeighborsRegressor()
knr.fit(x_train,y_train)
```

Out[18]:
```
KNeighborsRegressor()
```

In [19]:
```
knr.score(x_test,y_test)
```

Out[19]:
```
0.9795107093730114
```

In [21]:
```
from sklearn.metrics import mean_squared_error
print(f'MSE:{mean_squared_error(y_test,y_pred)}')
print(f'RMSE:{np.sqrt(mean_squared_error(y_test,y_pred))}')
```

```
MSE:4080.937103448275
RMSE:63.88221273130945
```

In [23]:
```
data = pd.DataFrame({'Actual Points': y_test.tolist(), 'Predicted Points': y_pred.tolist()})
data.head()
```

Out[23]:

	Actual Points	Predicted Points
0	587	623.2
1	89	98.8
2	350	325.2
3	1417	1350.6
4	1071	1081.4

In [24]:
```
from sklearn.preprocessing import Normalizer
norm = Normalizer()
X_norm = norm.fit_transform(feature)
```

In [25]:
```
x_train,x_test,y_train,y_test = train_test_split(X_norm,target,test_size=0.3,random_state=42)
```

In [26]:
```
knn_norm = KNeighborsRegressor()
knn_norm.fit(x_train,y_train)
```

Out[26]:
```
KNeighborsRegressor()
```

In [27]:
```
knn_norm.score(x_test,y_test)
```

Out[27]:
```
0.6820088360944019
```

In [29]:
```
y_new_pred = knn_norm.predict(x_test)
```

In [30]:
```
print(f'MSE:{mean_squared_error(y_test,y_new_pred)}')
print(f'RMSE:{np.sqrt(mean_squared_error(y_test,y_new_pred))}')
```

```
MSE:63335.620689655174
RMSE:251.66569231751708
```

In [31]:
```
data = pd.DataFrame({'Actual Points': y_test.tolist(), 'Predicted Points': y_new_pred.tolist()})
data.head(10)
```

Out[31]:

	Actual Points	Predicted Points
0	587	643.8
1	89	213.8
2	350	499.6
3	1417	875.4
4	1071	708.6
5	338	562.2
6	54	118.6
7	879	592.4
8	225	1019.2
9	298	382.8

In [33]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled = scaler.fit_transform(feature)
```

In [34]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,target,test_size=0.3,random_state=42)
```

In [35]:
```
knn_scaled = KNeighborsRegressor()
knn_scaled.fit(x_train,y_train)
```

Out[35]:
```
KNeighborsRegressor()
```

In [36]:
```
knn_scaled.score(x_test,y_test)
```

Out[36]:
```
0.9617776592228647
```

In [38]:
```
y_pred_2 = knn_scaled.predict(x_test)
```

In [39]:
```
print(f'MSE:{mean_squared_error(y_test,y_pred_2)}')
print(f'RMSE:{np.sqrt(mean_squared_error(y_test,y_pred_2))}')
```

```
MSE:7612.902344827587
RMSE:87.25194751309328
```

In [40]:
```
data = pd.DataFrame({'Actual Points': y_test.tolist(), 'Predicted Points': y_pred_2.tolist()})
data.head()
```

Out[40]:

	Actual Points	Predicted Points
0	587	475.2
1	89	172.6
2	350	393.6
3	1417	1512.6
4	1071	922.6

In [ ]: