Task Assigned use Knn Algorithm on Nba 2013 Dataset

(Parth Madan)

In [1]:

```
#Import libraries
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
from sklearn import metrics
```

In [2]:

```
data = pd.read_csv(r'C:\Users\Parth Madan\Downloads\nba_2013.csv')
data.head()
```

Out[2]:

	player	pos	age	bref_team_id	g	gs	mp	fg	fga	fg.	...	drb	trb	ast	stl	blk	tov	pf	pts	season	season_end
0	Quincy Acy	SF	23	TOT	63	0	847	66	141	0.468	...	144	216	28	23	26	30	122	171	2013-2014	2013
1	Steven Adams	C	20	OKC	81	20	1197	93	185	0.503	...	190	332	43	40	57	71	203	265	2013-2014	2013
2	Jeff Adrien	PF	27	TOT	53	12	961	143	275	0.520	...	204	306	38	24	36	39	108	362	2013-2014	2013
3	Arron Afflalo	SG	28	ORL	73	73	2552	464	1011	0.459	...	230	262	248	35	3	146	136	1330	2013-2014	2013
4	Alexis Ajinca	C	25	NOP	56	30	951	136	249	0.546	...	183	277	40	23	46	63	187	328	2013-2014	2013

5 rows × 31 columns

Data Pre Processing

In [3]:

```
data.isnull().any()
```

Out[3]:

```
player          False
pos             False
age             False
bref_team_id    False
g               False
gs              False
mp              False
fg              False
fga             False
fg.              True
x3p             False
x3pa            False
x3p.             True
x2p             False
x2pa            False
x2p.             True
efg.             True
ft              False
fta             False
ft.              True
orb             False
drb             False
trb             False
ast             False
stl             False
blk             False
tov             False
pf              False
pts             False
season          False
season_end      False
dtype: bool
```

Fill Null Values with its mean Value

In [4]:

```
data["fg."].fillna(data["fg."].mean(),inplace=True)
data["x2p."].fillna(data["x2p."].mean(),inplace=True)
data["efg."].fillna(data["efg."].mean(),inplace=True)
data["x3p."].fillna(data["x3p."].mean(),inplace=True)
data["ft."].fillna(data["ft."].mean(),inplace=True)
```

Select Valid Numeric Columns from respective dataset

In [5]:

```
distance_columns = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'ef
g.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf', 'pts']
data_numeric = data[distance_columns]
```

In [6]:

```
data_numeric
```

Out[6]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	...	ft.	orb	drb	trb	ast	stl	blk	tov	pf	pts
0	23	63	0	847	66	141	0.468	4	15	0.266667	...	0.660	72	144	216	28	23	26	30	122	171
1	20	81	20	1197	93	185	0.503	0	0	0.285111	...	0.581	142	190	332	43	40	57	71	203	265
2	27	53	12	961	143	275	0.520	0	0	0.285111	...	0.639	102	204	306	38	24	36	39	108	362
3	28	73	73	2552	464	1011	0.459	128	300	0.426667	...	0.815	32	230	262	248	35	3	146	136	1330
4	25	56	30	951	136	249	0.546	0	1	0.000000	...	0.836	94	183	277	40	23	46	63	187	328
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
476	20	72	16	1765	345	808	0.427	40	188	0.212766	...	0.641	69	159	228	217	78	16	204	151	939
477	28	64	9	1810	387	889	0.435	135	350	0.385714	...	0.825	29	137	166	95	46	12	95	156	1144
478	25	79	78	2718	582	1283	0.454	90	292	0.308219	...	0.712	166	310	476	182	167	36	165	213	1417
479	21	82	3	1416	172	404	0.426	0	1	0.000000	...	0.730	118	235	353	92	40	41	87	170	490
480	24	70	9	1049	156	290	0.538	0	1	0.000000	...	0.719	103	179	282	36	18	38	60	137	399

481 rows × 26 columns

Apply Normalization

In [7]:

```
data_normalized = data_numeric.apply(lambda x: (x - x.min()) / (x.max() - x.min()))
```

In [8]:

```
#categorical Columns
data_category = data[['player', 'bref_team_id', 'season']]
```

Train Test Split

In [9]:

```
data = pd.concat([data_category, data_normalized], axis=1)

from sklearn.model_selection import train_test_split

# The columns that we will be making predictions with.
x_columns = data[['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']]

# The column that we want to predict.
y_column = data["pts"]

x_train, x_test, y_train, y_test = train_test_split(x_columns, y_column, test_size=0.3, random_state=0) #70%train
and 30% Test
```

Create Knn Model

In [10]:

```
for k in range(10):
    k_value = k + 1
    knn = KNeighborsRegressor(n_neighbors = k_value)
    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)
    print ("Regression score is:",format(metrics.r2_score(y_test, y_pred),'.4f'), "for k_value:", k_value)
```

```
Regression score is: 0.9145 for k_value: 1
Regression score is: 0.9464 for k_value: 2
Regression score is: 0.9548 for k_value: 3
Regression score is: 0.9594 for k_value: 4
Regression score is: 0.9583 for k_value: 5
Regression score is: 0.9579 for k_value: 6
Regression score is: 0.9579 for k_value: 7
Regression score is: 0.9609 for k_value: 8
Regression score is: 0.9576 for k_value: 9
Regression score is: 0.9557 for k_value: 10
```

In [11]:

```
#As we got Highest value at k=8
knn = KNeighborsRegressor(n_neighbors = 8)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
print ("Mean Squared Error is:", format(metrics.mean_squared_error(y_test, y_pred), '.7f'))
print ("Regression score is:", format(metrics.r2_score(y_test, y_pred),'.4f'))
```

```
Mean Squared Error is: 0.0011143
Regression score is: 0.9609
```

In [12]:

```
Predicted_value = pd.DataFrame({'Actual Points': y_test.tolist(), 'Predicted Points': y_pred.tolist()})

Predicted_value
```

Out[12]:

	Actual Points	Predicted Points
0	0.168145	0.125723
1	0.276514	0.297243
2	0.422676	0.363189
3	0.007327	0.011088
4	0.381026	0.373939
...	...	...
140	0.426919	0.421664
141	0.013498	0.019379
142	0.312379	0.303943
143	0.306980	0.273766
144	0.036251	0.029985

145 rows × 2 columns

END

In [ ]: