ML7 Assignment - KNN algorithm to predict "how many points NBA players scored in the 2013-2014 season".

In [1]:

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

```
# Importing the dataset
nba = pd.read_csv('nba_2013.csv')
```

In [3]:

nba.head(5)

Out[3]:

player	pos	age	bref_team_id	g	gs	mp	fg	fga	fg.		drb	trb	ast	stl	bll
Quincy Acy	SF	23	ТОТ	63	0	847	66	141	0.468		144	216	28	23	26
Steven Adams	С	20	ОКС	81	20	1197	93	185	0.503		190	332	43	40	57
Jeff Adrien	PF	27	ТОТ	53	12	961	143	275	0.520		204	306	38	24	36
Arron Afflalo	SG	28	ORL	73	73	2552	464	1011	0.459		230	262	248	35	;
Alexis Ajinca	С	25	NOP	56	30	951	136	249	0.546		183	277	40	23	46
	Quincy Acy Steven Adams Jeff Adrien Arron Afflalo	Quincy Acy SF Steven Adams C Jeff Adrien PF Arron Afflalo SG Alexis C	Quincy Acy SF 23 Steven C 20 Jeff Adrien PF 27 Arron Afflalo SG 28 Alexis C 25	Quincy Acy SF 23 TOT Steven Adams C 20 OKC Jeff Adrien PF 27 TOT Arron Afflalo SG 28 ORL Alexis C 25 NOR	Quincy Acy SF 23 TOT 63 Steven Adams C 20 OKC 81 Jeff Adrien PF 27 TOT 53 Arron Afflalo SG 28 ORL 73 Alexis C 25 NOR 56	Quincy Acy SF 23 TOT 63 0 Steven Adams C 20 OKC 81 20 Jeff Adrien PF 27 TOT 53 12 Arron Afflalo SG 28 ORL 73 73 Alexis C 25 NOR 56 30	Quincy Acy SF 23 TOT 63 0 847 Steven Adams C 20 OKC 81 20 1197 Jeff Adrien PF 27 TOT 53 12 961 Arron Afflalo SG 28 ORL 73 73 2552 Alexis C 25 NOR 56 30 951	Quincy Acy SF 23 TOT 63 0 847 66 Steven Adams C 20 OKC 81 20 1197 93 Jeff Adrien PF 27 TOT 53 12 961 143 Arron Afflalo SG 28 ORL 73 73 2552 464 Alexis C 25 NOR 56 30 951 136	Quincy Acy SF 23 TOT 63 0 847 66 141 Steven Adams C 20 OKC 81 20 1197 93 185 Jeff Adrien PF 27 TOT 53 12 961 143 275 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 Alexis C 25 NOR 56 30 951 136 249	Quincy Acy SF 23 TOT 63 0 847 66 141 0.468 Steven Adams C 20 OKC 81 20 1197 93 185 0.503 Jeff Adrien PF 27 TOT 53 12 961 143 275 0.520 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 0.459 Alexis C 25 NOR 56 30 951 136 249 0.546	Quincy Acy SF 23 TOT 63 0 847 66 141 0.468 Steven Adams C 20 OKC 81 20 1197 93 185 0.503 Jeff Adrien PF 27 TOT 53 12 961 143 275 0.520 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 0.459 Alexis C 25 NOR 56 30 951 136 249 0.546	Quincy Acy SF 23 TOT 63 0 847 66 141 0.468 144 Steven Adams C 20 OKC 81 20 1197 93 185 0.503 190 Jeff Adrien PF 27 TOT 53 12 961 143 275 0.520 204 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 0.459 230 Alexis C 25 NOP 56 30 951 136 249 0.546 183	Quincy Acy SF 23 TOT 63 0 847 66 141 0.468 144 216 Steven Adams C 20 OKC 81 20 1197 93 185 0.503 190 332 Jeff Adrien PF 27 TOT 53 12 961 143 275 0.520 204 306 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 0.459 230 262 Alexis C 25 NOR 56 30 951 136 249 0.546 183 277	Quincy Acy SF 23 TOT 63 0 847 66 141 0.468 144 216 28 Steven Adams C 20 OKC 81 20 1197 93 185 0.503 190 332 43 Jeff Adrien PF 27 TOT 53 12 961 143 275 0.520 204 306 38 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 0.459 230 262 248 Alexis C 25 NOR 56 30 951 136 249 0.546 183 277 40	Quincy Acy SF 23 TOT 63 0 847 66 141 0.468 144 216 28 23 Steven Adams C 20 OKC 81 20 1197 93 185 0.503 190 332 43 40 Jeff Adrien PF 27 TOT 53 12 961 143 275 0.520 204 306 38 24 Arron Afflalo SG 28 ORL 73 73 2552 464 1011 0.459 230 262 248 35 Alexis C 25 NOR 56 30 951 136 249 0.546 183 277 40 23

5 rows × 31 columns

In [4]:

nba.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 481 entries, 0 to 480
Data columns (total 31 columns):
                481 non-null object
player
pos
                481 non-null object
                481 non-null int64
age
                481 non-null object
bref_team_id
                481 non-null int64
g
                481 non-null int64
gs
mp
                481 non-null int64
                481 non-null int64
fg
fga
                481 non-null int64
                479 non-null float64
fg.
                481 non-null int64
х3р
                481 non-null int64
x3pa
                414 non-null float64
x3p.
                481 non-null int64
x2p
                481 non-null int64
x2pa
                478 non-null float64
x2p.
efg.
                479 non-null float64
                481 non-null int64
ft
fta
                481 non-null int64
ft.
                461 non-null float64
                481 non-null int64
orb
drb
                481 non-null int64
                481 non-null int64
trb
ast
                481 non-null int64
                481 non-null int64
stl
blk
                481 non-null int64
                481 non-null int64
tov
pf
                481 non-null int64
pts
                481 non-null int64
                481 non-null object
season
season_end
                481 non-null int64
dtypes: float64(5), int64(22), object(4)
memory usage: 116.6+ KB
```

In [5]:

```
nba.describe()
```

Out[5]:

	age	g	gs	mp	fg	fga	fg.
count	481.000000	481.000000	481.000000	481.000000	481.000000	481.000000	479.000000
mean	26.509356	53.253638	25.571726	1237.386694	192.881497	424.463617	0.436436
std	4.198265	25.322711	29.658465	897.258840	171.832793	368.850833	0.098672
min	19.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	23.000000	32.000000	0.000000	388.000000	47.000000	110.000000	0.400500
50%	26.000000	61.000000	10.000000	1141.000000	146.000000	332.000000	0.438000
75%	29.000000	76.000000	54.000000	2016.000000	307.000000	672.000000	0.479500
max	39.000000	83.000000	82.000000	3122.000000	849.000000	1688.000000	1.000000

8 rows × 27 columns

EDA & Data Preprocessing

In [6]:

```
#Finding missing values
total = nba.isnull().sum().sort_values(ascending=False)
percent_1 = nba.isnull().sum()/nba.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(5)
```

Out[6]:

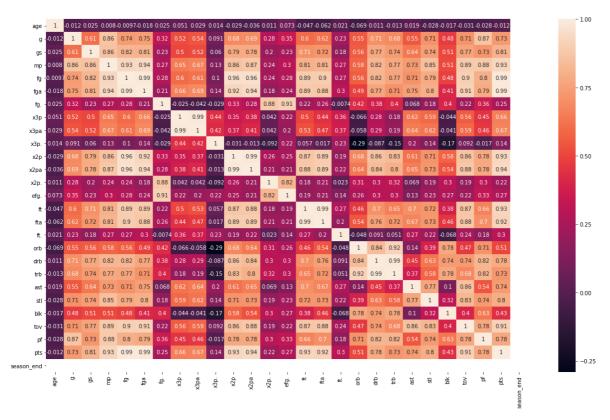
	Total	%
х3р.	67	13.9
ft.	20	4.2
x2p.	3	0.6
fg.	2	0.4
efg.	2	0.4

In [7]:

```
#Replacing the missing values with mean
nba = nba.fillna(nba.mean())
```

In [9]:

```
import seaborn as sns
hmap = nba.corr()
plt.subplots(figsize=(20, 12))
sns.heatmap(hmap,annot=True);
```



The target column pts has high correlation with maximum of the numerical features

In [10]:

In [11]:

```
X = nba[X_columns]
y = nba[y_column]
```

In [12]:

```
#Split the data in train & test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0
```

KNN model

```
In [13]:
```

```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
In [14]:
```

```
mse = (((y_pred - y_test) ** 2).sum()) / len(y_pred)
```

In [15]:

```
print(mse)
print("predictions[:5]:\n", y_pred[:5])
print("actual[:5]:\n", y_test[:5])
```

```
5434.724223602485
predictions[:5]:
  [ 316.6 693.6 1069.6 11.8 903.4]
actual[:5]:
  15 436
124 717
141 1096
263 19
170 988
Name: pts, dtype: int64
```