

# Data Scraping and Analysis : Build your NBA Dream Team

You want to create the perfect 5 Major ? You want to know which players will be the next NBA stars ? You want to predict what will be the american olympic team at the next Olympic Games ?

- In this project, I will use NBA players datas in order to create a function that may turn out to be very useful in your team making decision process.
- A Five Major is the five players of a team that start games, oftentimes the best players of the team.

In this notebook I will use :

- Web scraping for collecting data on a website.
- Object oriented programming.
- SQL commands.
- Data cleaning for making the datas standardized and usable.
- Data analysis for checking data quality and getting insights.

## Package and libraries

```
In [124]: from bs4 import BeautifulSoup
import requests
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.font_manager import FontProperties

import warnings;
warnings.filterwarnings('ignore')

%matplotlib inline
```

## 2) Web Scraping

We will scrape each season datas from 1980 to 2020. A season table contain the players average statistics per game.

```
In [125]: ## 1) Create the framework of the dataframe

page = requests.get('https://www.basketball-reference.com/leagues/NBA_2020_per_game.html')
soup = BeautifulSoup(page.content, 'html.parser')
tableau = soup.find_all(class_="full_table")

head = soup.find(class_='thead')
column_name = [head.text for item in head][0]
column_name2 = column_name.split('\n')
del(column_name2[2])
del(column_name2[1])
df2 = pd.DataFrame(players, columns=column_name2)
df2['Season'] = season
df = df.append(df2, ignore_index=True, sort=False)
```

```
In [3]: df

Out[3]:
```

	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0 rows × 23 columns																						

```
In [5]: ## 2) Fill the dataframe

for season in range(1980,2021):

    url_players = 'https://www.basketball-reference.com/leagues/NBA_{}_per_game.html'.format(season)

    page = requests.get(url_players)
    soup = BeautifulSoup(page.content, 'html.parser')
    tableau = soup.find_all(class_="full_table")

    players = []

    for i in range(len(tableau)):

        player = []
        for j in range(len(tableau[i].find_all('td'))):
            player.append(j.text)
            players.append(player)

    head2 = soup.find(class_='thead')
    column_name2 = [head2.text for item in head][0]
    column_name2 = column_name2.split('\n')
    del(column_name2[2])
    del(column_name2[1])
    df2 = pd.DataFrame(players, columns=column_name2)
    df2['Season'] = season
    df = df.append(df2, ignore_index=True, sort=False)
```

## 3) Data quality check & data cleaning

```
In [14]: df.head()
```

```
Out[14]:
```

	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0	Kareem Abdul-Jabbar	C	32	LAL	82	NaN	38.3	10.2	16.9	0.604	...	2.3	8.5	10.8	4.5	1.0	3.4	3.6	2.6	24.8	1980
1	Tom Abernethy	PF	25	GSW	67	18.2	2.3	4.7	0.481	...	0.9	1.9	2.9	1.3	0.5	0.2	0.6	1.8	5.4	1980	
2	Alvan Adams	C	25	PHO	75	28.9	6.2	11.7	0.531	...	2.1	6.0	8.1	4.3	1.4	0.7	2.9	3.2	14.9	1980	
3	Tiny Archibald	PG	31	BOS	80	80.0	35.8	4.8	9.9	0.482	...	0.7	1.7	2.5	8.4	1.3	0.1	3.0	2.7	14.1	1980
4	Dennis Awrey	C	31	CHI	26	21.5	1.0	2.3	0.450	...	1.1	3.3	4.4	1.5	0.5	0.6	1.0	2.5	3.3	1980	
5 rows × 30 columns																					

```
In [21]: for col in df.columns[4:38]:
df[col] = pd.to_numeric(df[col], errors='coerce')

df['Age'] = df['Age'].astype(int)
df['Season'] = df['Season'].astype(int)
```

```
In [22]: df.head()
```

```
Out[22]:
```

	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0	Kareem Abdul-Jabbar	C	32	LAL	82	NaN	38.3	10.2	16.9	0.604	...	2.3	8.5	10.8	4.5	1.0	3.4	3.6	2.6	24.8	1980
1	Tom Abernethy	PF	25	GSW	67	NaN	18.2	2.3	4.7	0.481	...	0.9	1.9	2.9	1.3	0.5	0.2	0.6	1.8	5.4	1980
2	Alvan Adams	C	25	PHO	75	NaN	28.9	6.2	11.7	0.531	...	2.1	6.0	8.1	4.3	1.4	0.7	2.9	3.2	14.9	1980
3	Tiny Archibald	PG	31	BOS	80	80.0	35.8	4.8	9.9	0.482	...	0.7	1.7	2.5	8.4	1.3	0.1	3.0	2.7	14.1	1980
4	Dennis Awrey	C	31	CHI	26	NaN	21.5	1.0	2.3	0.450	...	1.1	3.3	4.4	1.5	0.5	0.6	1.0	2.5	3.3	1980
5 rows × 30 columns																					

Check the NaN values percentage in each column. Fortunately columns with NaN will not be used in our case.

```
In [29]: df.isna().sum()/df.shape[0]
```

```
Out[29]:
```

Player	0.000000
Pos	0.000000
Age	0.000000
Tm	0.000000
G	0.000000
GS	0.961689
MP	0.000000
FG	0.000000
FGA	0.000000
FG%	0.002989
ORB	0.000000
DRB	0.000000
TRB	0.000000
AST	0.000000
STL	0.000000
BLK	0.000000
TOV	0.000000
PF	0.000000
PTS	0.000000
Season	0.000000
dtype:	float64

Visualize the composition of non numeric columns

- A ~ after a player's name means that the player is not in the NBA anymore
- A basketball team is composed of five players. Each of them has a position : 'C' means 'Center', 'PF' means 'Power Forward', 'PG' means 'Point Guard', 'SG' means 'Shooting Guard', 'SF' means 'Small Forward'. There are also players that get play at two different positions.
- NBA teams are often reported by their acronym, for instance 'LAL' means Los Angeles Lakers.

```
In [37]: for col in df.select_dtypes('object'):
print(f'{col} :<48') (df[col].unique()))

Player----- ['Kareem Abdul-Jabbar', 'Tom Abernethy', 'Alvan Adams', ...
'Kigell Williams-Goss', 'Zion Williamson', 'Justin Wright-Foreman']
'PG-SG', 'PF-SF', 'PG-SF', 'SG-PF', 'SF-C']
PF----- ['C', 'PF', 'PG', 'SG', 'SF', 'SG-PG', 'SG-SG', 'SG-SF', 'C-PF', 'PF-C', 'SF-PF',
'PHI', 'KCK', 'NJN', 'MIL', 'UTA', 'CLE', 'NYK', 'ATL', 'SAS', 'DET', 'DEN', 'DAL',
'LAC', 'SAC', 'CHI', 'MIA', 'ORL', 'MIN', 'VAN', 'TOR', 'WAS', 'MEM', 'NOH', 'CHA',
'IND', 'ORC', 'BKN', 'NOP', 'CHO']
```

```
In [31]: df.describe().transpose()
```

```
Out[31]:
```

	count	mean	std	min	25%	50%	75%	max
Age	17734.0	26.671479	4.049753	18.0	24.000	26.000	29.000	44.0
G	17734.0	54.691948	25.276153	1.0	36.000	63.000	77.000	85.0
GS	16640.0	26.503005	29.682342	0.0	1.000	11.000	53.000	83.0
MP	17734.0	20.583066	10.112226	0.0	12.100	20.100	29.100	43.7
FG	17734.0	3.229892	2.291933	0.0	1.400	2.700	4.600	13.4
FGA	17734.0	7.070971	4.712295	0.0	3.300	5.900	10.000	27.8
FG%	17681.0	0.443537	0.090097	0.0	0.407	0.448	0.489	1.0
3P	17734.0	0.418473	0.812239	0.0	0.000	0.100	0.700	5.1
3PA	17734.0	1.217159	1.622814	0.0	0.000	0.400	2.000	13.2
3P%	14926.0	0.251542	0.171284	0.0	0.125	0.296	0.364	1.0
2P	17734.0	2.810500	2.149860	0.0	1.125	2.200	4.000	13.2
2PA	17734.0	5.852650	4.223609	0.0	2.600	4.700	8.300	27.0
2P%	17649.0	0.464669	0.093403	0.0	0.430	0.472	0.508	1.0
eFG%	17681.0	0.471554	0.091737	0.0	0.440	0.480	0.515	1.5
FT	17734.0	1.615146	1.437898	0.0	0.600	1.200	2.200	10.3
FTA	17734.0	2.160697	1.816570	0.0	0.900	1.600	2.900	13.1
FT%	17215.0	0.725363	0.193653	0.0	0.667	0.750	0.813	1.0
ORB	17734.0	1.082310	0.875655	0.0	0.400	0.800	1.500	7.0
DRB	17734.0	2.567007	1.822295	0.0	1.200	2.100	3.400	12.3
TRB	17734.0	3.627963	2.576345	0.0	1.700	3.000	4.900	18.7
AST	17734.0	1.933771	1.873434	0.0	0.600	1.300	2.600	14.5
STL	17734.0	0.685491	0.482607	0.0	0.300	0.600	0.900	3.7
BLK	17734.0	0.427179	0.515082	0.0	0.100	0.300	0.500	5.6
TOV	17734.0	1.282162	0.832343	0.0	0.700	1.100	1.800	5.7
PF	17734.0	1.971050	0.863359	0.0	1.300	2.000	2.600	6.0
PTS	17734.0	8.492032	6.099345	0.0	3.700	7.000	12.000	37.1
Season	17734.0	2001.089207	12.005391	1980.0	1991.000	2002.000	2012.000	2020.0

## 4) Database Storage

```
In [103]: import sqlite3

conn = sqlite3.connect("projects.sqlite")
df.to_sql("nba_players_stats", conn, if_exists="replace")
conn.commit()
conn.close()
```

```
In [126]: import sqlite3

conn = sqlite3.connect("projects.sqlite")
df = pd.read_sql("SELECT * FROM nba_players_stats", conn)
conn.close()
```

```
In [64]: df.head()
```

```
Out[64]:
```

	index	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Season
0	0	Kareem Abdul-Jabbar	C	32	LAL	82	NaN	38.3	10.2	16.9	...	2.3	8.5	10.8	4.5	1.0	3.4	3.6	2.6	24.8	1980
1	1	Tom Abernethy	PF	25	GSW	67	NaN	18.2	2.3	4.7	...	0.9	1.9	2.9	1.3	0.5	0.2	0.6	1.8	5.4	1980
2	2	Alvan Adams	C	25	PHO	75	NaN	28.9	6.2	11.7	...	2.1	6.0	8.1	4.3	1.4	0.7	2.9	3.2	14.9	1980
3	3	Tiny Archibald	PG	31	BOS	80	80.0	35.8	4.8	9.9	...	0.7	1.7	2.5	8.4	1.3	0.1	3.0	2.7	14.1	1980
4	4	Dennis Awrey	C	31	CHI	26	NaN	21.5	1.0	2.3	...	1.1	3.3	4.4	1.5	0.5	0.6	1.0	2.5	3.3	1980
5 rows × 31 columns																					

## 5) Data analysis

Best scorer per season

```
In [127]: df_pts = pd.DataFrame(columns=column_name)

for season in range(1980,2021):
df2 = df.loc[df['Season']==season].nlargest(1, ['PTS'])
df_pts = df_pts.append(df2)

df_pts[['Player','PTS','Season']].sort_values(by='PTS',ascending=False)
```

```
Out[127]:
```

2909	Michael Jordan	37.1	1987.0
16880	James Harden	36.1	2019.0
10604	Kobe Bryant	35.4	2006.0
3249	Michael Jordan	35.0	1988.0
17402	James Harden	34.3	2020.0
3957	Michael Jordan	33.6	1990.0
97	George Gervin	33.1	1980.0
2272	Bernard King	32.9	1985.0
5119	Michael Jordan	32.6	1993.0
3598	Michael Jordan	32.5	1989.0
9279	George Gervin	32.3	1982.0
14459	Tracy McGrady	32.1	2003.0
14332	Kevin Durant	32.0	2014.0
11067	Kobe Bryant	31.6	2007.0
16106	Russell Westbrook	31.6	2017.0
4344	Michael Jordan	31.5	1991.0
8962	Allen Iverson	31.4	2002.0
8516	Allen Iverson	31.1	2001.0
10286	Allen Iverson	30.7	2005.0
1555	Adrian Dantley	30.7	1983.0
346	Adrian Dantley	30.7	1981.0
1969	Adrian Dantley	30.6	1984.0
6332	Michael Jordan	30.4	1996.0
16328	James Harden	30.4	2018.0
2748	Dominique Wilkins	30.3	1986.0
12318	Dwayne Wade	30.2	2009.0
15277	Stephen Curry	30.1	2016.0
4736	Michael Jordan	30.1	1992.0
12482	Kevin Durant	30.1	2010.0
11672	LeBron James	30.0	2008.0
5645	David Robinson	29.8	1994.0
8185	Shaquille O'Neal	29.7	2000.0
6783	Michael Jordan	29.6	1997.0
5999	Shaquille O'Neal	29.3	1995.0
13745	Carmelo Anthony	28.7	2013.0
7215	Michael Jordan	28.7	1990.0
15150	Russell Westbrook	28.1	2015.0
13377	Kevin Durant	28.0	2012.0
9901	Tracy McGrady	28.0	2004.0
12927	Kevin Durant	27.7	2011.0
7629	Allen Iverson	26.8	1999.0

```
In [128]: best_scorers = {}

for i in range(len(df_pts)):

    if df_pts.at[i,0] not in best_scorers:
        best_scorers.update({df_pts.at[i,0] : 1 })
    else:
        best_scorers[df_pts.at[i,0]] += 1

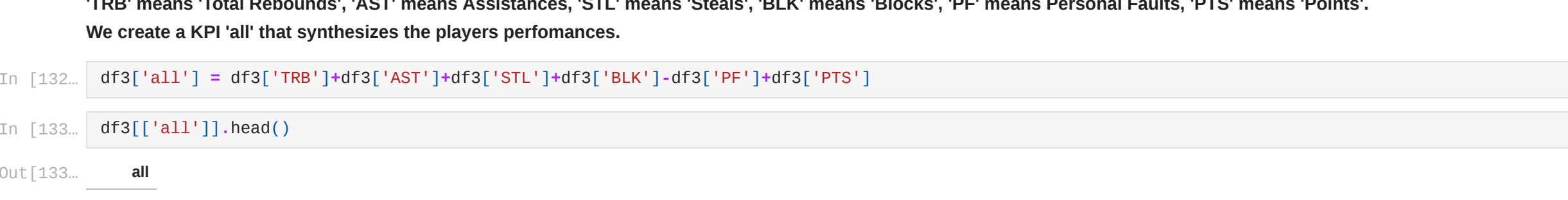
print(best_scorers)

{'George Gervin': 2, 'Adrian Dantley': 3, 'Bernard King': 1, 'Dominique Wilkins': 1, 'Michael Jordan': 10, 'David Robinson': 1, 'Shaquille O'Neal': 2, 'Allen Iverson': 4, 'Tracy McGrady': 2, 'Kobe Bryant': 2, 'LeBron James': 1, 'Dwayne Wade': 1, 'Kevin Durant': 4, 'Carmelo Anthony': 1, 'Russell Westbrook': 3, 'Stephen Curry': 1, 'James Harden': 3}
```

```
In [129]: plt.figure(figsize=(9,4.5))
plt.grid(axis='x', linestyle='--', linewidth=0.4)
plt.xticks(list(range(max(best_scorers.values())+1)))
plt.xlabel('Frequency')
plt.barh(*zip(*best_scorers.items()))
plt.title('Season Best Scorer Frequency')
```

```
Out[129]:
```

Text(0.5, 1.0, 'Season Best Scorer Frequency')



Check the evolution of these players


```
In [130]: df2 = df[['Player','Age','PTS']].loc[df['Player'].isin(['Michael Jordan','Kevin Durant','Kobe Bryant'])]
df2 = df2.groupby(['Age','Player']).mean()

FontP = FontProperties()
```

```
df2.unstack(1).plot(figsize=(10,6),linewidth=4, marker='o')
plt.xticks(list(range(18,40)))
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', prop=FontP)
plt.grid(linestyle=':')
plt.barh(*zip(*best_scorers.items()))
plt.title('All Times Best NBA Scorers Evolution')
```

```
Out[130]:
```

Text(0.5, 1.0, 'All Times Best NBA Scorers Evolution')



```
In [69]: df_pts.describe().transpose()
```

```
Out[69]:
```

	count	mean	std	min	25%	50%	75%	max
GS	39.0	74.025641	1.171030	22.000	71.5000	79.0000	81.0000	82.000
MP	41.0	38.812195	2.236313	34.200	37.5000	38.9000	40.3000	43.700
FG	41.0	11.014634	1.181220	9.000	10.2000	10.9000	11.9000	13.400
FGA	41.0	22.724390	2.345184	18.200	20.8000	22.6000	24.2000	27.800
FG%	41.0	0.486805	0.049087	0.398	0.4490	0.4860	0.5260	0.583
3P	41.0	1.370732	1.343362	0.000	0.2000	1.3000	2.0	