Predict Basketball MVP

- Web Scraper

In [2]:
```
!pip install requests
```

```
Requirement already satisfied: requests in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (2.28.
0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/mac/opt/anaconda3/lib/python3.9/site-pa
ckages (from requests) (1.26.7)
Requirement already satisfied: idna<4,>=2.5 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (f
rom requests) (3.2)
Requirement already satisfied: certifi>=2017.4.17 in /Users/mac/opt/anaconda3/lib/python3.9/site-packa
ges (from requests) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/mac/opt/anaconda3/lib/python3.9/sit
e-packages (from requests) (2.0.4)
```

In [3]:
```
import os
os.getcwd()
```

Out[3]:
```
'/Users/mac/Dataquest Project Walkthrough'
```

In [4]:
```
years = list(range(1991,2022)) # non inclusive the last year
```

In [5]:
```
url_start = "https://www.basketball-reference.com/awards/awards_{}.html"
```

- save file that we do not need to access the table again

In [6]:
```
# request and save
import requests
for year in years:
    url = url_start.format(year)
    data = requests.get(url)

    with open("/Users/mac/Dataquest Project Walkthrough/{}.html".format(year),"w+") as f:
        f.write(data.text)
```

- extract table from html

- what is BeautifulSoup?
- Beautiful Soup is a Python library for pulling data out of HTML and XML files.

In [7]:
```
!pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages
(4.10.0)
Requirement already satisfied: soupsieve>1.2 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages
(from beautifulsoup4) (2.2.1)
```

In [8]:
```
from bs4 import BeautifulSoup
```

In [9]:
```
with open("/Users/mac/Dataquest Project Walkthrough/1991.html") as f:
    page = f.read()
type(page)
```

Out[9]:
```
str
```

- Parse Page

In [10]:
```
soup = BeautifulSoup(page,'html.parser')
```

- Remove elements using decompose

```
In [11]: soup.find('tr', class_='over_header').decompose()
```

```
In [12]: mvp_table = soup.find_all(id='mvp')
```

```
In [13]: type(mvp_table)
```

```
Out[13]: bs4.element.ResultSet
```

```
In [14]: import pandas as pd
```

In [15]:
```
pd.read_html(str(mvp_table))[0]
```

Out[15]:

	Rank	Player	Age	Tm	First	Pts Won	Pts Max	Share	G	MP	PTS	TRB	AST	STL	BLK	FG%	3P%	FT%	WS	WS/48
0	1	Michael Jordan	27	CHI	77.0	891.0	960	0.928	82	37.0	31.5	6.0	5.5	2.7	1.0	0.539	0.312	0.851	20.3	0.321
1	2	Magic Johnson	31	LAL	10.0	497.0	960	0.518	79	37.1	19.4	7.0	12.5	1.3	0.2	0.477	0.320	0.906	15.4	0.251
2	3	David Robinson	25	SAS	6.0	476.0	960	0.496	82	37.7	25.6	13.0	2.5	1.5	3.9	0.552	0.143	0.762	17.0	0.264
3	4	Charles Barkley	27	PHI	2.0	222.0	960	0.231	67	37.3	27.6	10.1	4.2	1.6	0.5	0.570	0.284	0.722	13.4	0.258
4	5	Karl Malone	27	UTA	0.0	142.0	960	0.148	82	40.3	29.0	11.8	3.3	1.1	1.0	0.527	0.286	0.770	15.5	0.225
5	6	Clyde Drexler	28	POR	1.0	75.0	960	0.078	82	34.8	21.5	6.7	6.0	1.8	0.7	0.482	0.319	0.794	12.4	0.209
6	7	Kevin Johnson	24	PHO	0.0	32.0	960	0.033	77	36.0	22.2	3.5	10.1	2.1	0.1	0.516	0.205	0.843	12.7	0.220
7	8	Dominique Wilkins	31	ATL	0.0	29.0	960	0.030	81	38.0	25.9	9.0	3.3	1.5	0.8	0.470	0.341	0.829	11.4	0.177
8	9T	Larry Bird	34	BOS	0.0	25.0	960	0.026	60	38.0	19.4	8.5	7.2	1.8	1.0	0.454	0.389	0.891	6.6	0.140
9	9T	Terry Porter	27	POR	0.0	25.0	960	0.026	81	32.9	17.0	3.5	8.0	2.0	0.1	0.515	0.415	0.823	13.0	0.235
10	11	Patrick Ewing	28	NYK	0.0	20.0	960	0.021	81	38.3	26.6	11.2	3.0	1.0	3.2	0.514	0.000	0.745	10.0	0.155
11	12	John Stockton	28	UTA	0.0	15.0	960	0.016	82	37.8	17.2	2.9	14.2	2.9	0.2	0.507	0.345	0.836	14.0	0.217
12	13	Isiah Thomas	29	DET	0.0	11.0	960	0.011	48	34.5	16.2	3.3	9.3	1.6	0.2	0.435	0.292	0.782	3.4	0.098
13	14	Robert Parish	37	BOS	0.0	10.0	960	0.010	81	30.1	14.9	10.6	0.8	0.8	1.3	0.598	0.000	0.767	10.0	0.198
14	15	Joe Dumars	27	DET	0.0	8.0	960	0.008	80	38.1	20.4	2.3	5.5	1.1	0.1	0.481	0.311	0.890	9.9	0.155
15	16	Bernard King	34	WSB	0.0	7.0	960	0.007	64	37.5	28.4	5.0	4.6	0.9	0.3	0.472	0.216	0.790	3.5	0.070
16	17	Kenny Smith	25	HOU	0.0	5.0	960	0.005	78	34.6	17.7	2.1	7.1	1.4	0.1	0.520	0.363	0.844	9.0	0.161
17	18	Hakeem Olajuwon	28	HOU	0.0	4.0	960	0.004	56	36.8	21.2	13.8	2.3	2.2	3.9	0.508	0.000	0.769	8.6	0.201

	Rank	Player	Age	Tm	First	Pts Won	Pts Max	Share	G	MP	PTS	TRB	AST	STL	BLK	FG%	3P%	FT%	WS	WS/48
18	19T	Tim Hardaway	24	GSW	0.0	1.0	960	0.001	82	39.2	22.9	4.0	9.7	2.6	0.1	0.476	0.385	0.803	9.9	0.148
19	19T	Kevin McHale	33	BOS	0.0	1.0	960	0.001	68	30.4	18.4	7.1	1.9	0.4	2.1	0.553	0.405	0.829	7.9	0.182

In [16]:
```
dfs = pd.DataFrame()

for year in years:
    with open("/Users/mac/Dataquest Project Walkthrough/{}.html".format(year)) as f:
        page = f.read()
    soup = BeautifulSoup(page,'html.parser')
    mvp_table = soup.find_all(id='mvp')
    mvp = pd.read_html(str(mvp_table))[0]
    mvp['Year'] = year
    dfs = pd.concat([dfs,mvp])
```

- store the dataframe into csv format

In [17]:
```
dfs.to_csv('mvps.csv')
```

- predict who is going to be MVP

- get player_data

In [18]:
```
import requests
for year in years:
    url = url_start.format(year)
    data = requests.get(url)

    with open("/Users/mac/Dataquest Project Walkthrough/{}.html".format(year),"w+") as f:
        f.write(data.text)
```

In [19]:
```
url_player = 'https://www.basketball-reference.com/leagues/NBA_{}_per_game.html'
year2 = years.copy()
for i in year2:
    url_players = url_player.format(year)
    stats = requests.get(url_players)
    with open("/Users/mac/Dataquest Project Walkthrough/player_stats/{}.html".format(i),"w+") as f:
        f.write(stats.text)
```

- but the problem is it is not full dataset since webpage has JS or other components
- that need to run on the client to get the full data
- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else

- Selenium
- https://www.aneasystone.com/archives/2018/02/python-selenium-spider.html (https://www.aneasystone.com/archives/2018/02/python-selenium-spider.html)

In [ ]:
```
```

In [20]:
```
!pip install selenium
```

```
Requirement already satisfied: selenium in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (4.4.
3)
Requirement already satisfied: urllib3[socks]~=1.26 in /Users/mac/opt/anaconda3/lib/python3.9/site-pac
kages (from selenium) (1.26.7)
Requirement already satisfied: trio~=0.17 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (fro
m selenium) (0.21.0)
Requirement already satisfied: certifi>=2021.10.8 in /Users/mac/opt/anaconda3/lib/python3.9/site-packa
ges (from selenium) (2021.10.8)
Requirement already satisfied: trio-websocket~=0.9 in /Users/mac/opt/anaconda3/lib/python3.9/site-pack
ages (from selenium) (0.9.2)
Requirement already satisfied: sortedcontainers in /Users/mac/opt/anaconda3/lib/python3.9/site-package
s (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: outcome in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (from t
rio~=0.17->selenium) (1.2.0)
Requirement already satisfied: sniffio in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (from t
rio~=0.17->selenium) (1.2.0)
Requirement already satisfied: attrs>=19.2.0 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages
(from trio~=0.17->selenium) (21.2.0)
Requirement already satisfied: idna in /Users/mac/opt/anaconda3/lib/python3.9/site-packages (from trio
~=0.17->selenium) (3.2)
Requirement already satisfied: async-generator>=1.9 in /Users/mac/opt/anaconda3/lib/python3.9/site-pac
kages (from trio~=0.17->selenium) (1.10)
Requirement already satisfied: wsproto>=0.14 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages
(from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied: PySocks!=1.5.7,<2.0,>=1.5.6 in /Users/mac/opt/anaconda3/lib/python3.9/s
ite-packages (from urllib3[socks]~=1.26->selenium) (1.7.1)
Requirement already satisfied: h11<1,>=0.9.0 in /Users/mac/opt/anaconda3/lib/python3.9/site-packages
(from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.13.0)
```

In [21]:
```
from selenium import webdriver
```

In [22]:
```
driver = webdriver.Chrome(executable_path="/Users/mac/Downloads/chromedriver")
```

```
/var/folders/cf/s1wshv2j2bz4cbfxfg5qgrgm0000gn/T/ipykernel_14365/3566386855.py:1: DeprecationWarning:
executable_path has been deprecated, please pass in a Service object
  driver = webdriver.Chrome(executable_path="/Users/mac/Downloads/chromedriver")
```

In [23]:
```
url_1991 = url_player.format(1991)
```

In [24]:
```
url_1991
```

Out[24]:
```
'https://www.basketball-reference.com/leagues/NBA_1991_per_game.html'
```

In [25]:
```
url_players
```

Out[25]:
```
'https://www.basketball-reference.com/leagues/NBA_2021_per_game.html'
```

In [26]:
```
import time

driver.get(url_player)
driver.execute_script("window.scrollTo(1,10000)")
time.sleep(2)

html = driver.page_source
```

- write html into file

In [27]:
```
with open("player_stats/{}.html".format(1991),"w+") as f:
    f.write(html)
```

- use for loop to read all html into files

In [28]:

```
for i in year2:
    url_full = url_player.format(i)
    driver.get(url_full)
    driver.execute_script("window.scrollTo(1,10000)")
    time.sleep(2)
    html = driver.page_source
    with open("player_stats/{}.html".format(i),"w+") as f:
        f.write(html)
    print('{}_is done'.format(i))
```

```
1991_is done
1992_is done
1993_is done
1994_is done
1995_is done
1996_is done
1997_is done
```

- clean html and full table data from players

In [ ]:
```
player_stats_full = pd.DataFrame()

for year in year2:
    with open("/Users/mac/Dataquest Project Walkthrough/player_stats/{}.html".format(year)) as f:
        page = f.read()
    soup = BeautifulSoup(page,'html.parser')
    soup.find('tr',class_='thead').decompose()
    player_table = soup.find_all(id='per_game_stats')
    player = pd.read_html(str(player_table))[0]
    player['Year'] = year
    player_stats_full = pd.concat([player_stats_full,player])
    print('{} is done'.format(year))
```

In [ ]:
```
player_stats_full.to_csv("Player_stats_file")
```

In [ ]:
```
player_stats_full
```

- find team record for this

In [30]:
```
teams_status_url = "https://www.basketball-reference.com/leagues/NBA_{}_standings.html"
```

In [31]:
```
for year in year2:
    url_team = teams_status_url.format(year)
    data = requests.get(url_team)
    with open("team_stats/{}.html".format(year),"w+") as f:
        f.write(data.text)
```

In [39]:
```
dfs = []
for year in year2:
    with open("team_stats/{}.html".format(year)) as f:
        page = f.read()

    soup = BeautifulSoup(page,'html.parser')
    soup.find('tr',class_='thead').decompose()
    team_table = soup.find_all(id='divs_standings_E')
    team = pd.read_html(str(team_table))[0]
    team['Year'] = year
    team['Team'] = team['Eastern Conference']
    del team['Eastern Conference']
    dfs.append(team)

    soup = BeautifulSoup(page,'html.parser')
    soup.find('tr',class_='thead').decompose()
    team_table = soup.find_all(id='divs_standings_W')
    team = pd.read_html(str(team_table))[0]
    team['Year'] = year
    team['Team'] = team['Western Conference']
    del team['Western Conference']

    dfs.append(team)
```

In [40]:
```
team_ = pd.concat(dfs)
team_.to_csv('team_')
```

In [ ]: