

```
In [ ]: # installs mermaid diagram library. Only important for the Fantasy  
Game Simulator diagrams.  
import sys  
!{sys.executable} -m pip install --user nb-mermaid
```

Introduction

DATA604 Project Report

Team : DATA_NERDS

Members: Gavin Lau, Prem Patel, Vardaan Bhatia, Mridul Gupta

Data and statistics are both ubiquitous in sports. Whether it is tracking "Power Play Goals" in Hockey, "First Serve Points Won" in Tennis, or "Height and Reach" in Boxing, they all speak to how integral we believe these metrics to be to sports performances and outcomes. Despite the universality of data in sports in this decade, sports data and the question of how to use this data to inform decisions in sports is a fairly recent development. The formation of the Society for American Baseball Research (SABR) in 1971 was the first major sports-related institution that recognized the value of statistics in sports (DSG, 2021), but the concept of leveraging these statistics to make decisions around baseball (aptly named SABR-metrics) did not have any public or commercial recognition until the publication of Moneyball (Lewis, 2004) and the release of the film with the same name in 2011.

Basketball's own Association for Professional Basketball Research was established in 1997, and subsequent APBR-metrics (a much less phonetically pleasing name) only made its way into the sport in 2004, with Dean Oliver being hired by the Seattle Supersonics, making him the first statistician hired in the NBA. The current "Statistical Player Value" is derived from "Minutes, Points, Rebounds, Assists, Steals, Blocks, and Turnovers" (Statistical Player Value, SPV - NBAstuffer, 2022). Other metrics such as "Offensive Rating", "Defensive Rating", and "Player Winning Percentage" (a combination of Offensive and Defensive Rating), have been developed to try to more accurately represent the game of basketball statistically.

Our goal for this project is to approach basketball with the same statistics and data-driven approach, to try to effectively model the game of basketball, bring out some insights, and to ultimately create an entertaining tool that will enable users to interact with the data and models in an intuitive and fun simulator-game.

A simple example of how even basic analytics can leave a mark on the game can be seen in the increasing adoption of the three-point shot. Teams asked themselves, "why would we continue to shoot a deep 2-point shot? Not only does it have a higher uncertainty than a layup or dunk, but it lacks the extra point gained from a three-pointer." In the simplest terms, a shooter who makes 50% of his two-point attempts only needs to make 33% of his

three-point attempts to score the same number of points. Layups, dunks, free throws, and threes became the standard following what is colloquially known as the "3 is greater than 2" mentality.

For us, being avid fans of the game, getting an opportunity to incorporate our love for the sport with this project made this work assignment engaging and a joy to challenge ourselves to work through problems and surpass our expectations. Within the scope of this engagement, each of us has explored basketball from an analytical point of view and tried to understand how certain statistics affect the general functioning of a team, the metrics required for winning games, and general player-related data sources.

From a broader point of view considering the NBA and the sport of basketball as a whole, our analysis gives minor insights into team and player behaviors and some important metrics to consider when building a NBA General Manager is building a Championship team.

Dataset and Web Scraping

Overview

As a team, our goal is to find a dataset that contains relevant information to answer our guiding questions. The sources we are utilizing for our datasets are:

[Basketball-Reference - Basketball Statistics and History.](#)

[NBA.com - Official NBA Stats](#)

Data on these websites are provided by 'SPORTS RADAR', (A Sports Technology Company), the official stats partner for the National Basketball Association (NBA). We are permitted to use the stats on the NBA website under its copyright terms which states that "the NBA Statistics may only be used, displayed or published for legitimate news reporting or private, non-commercial purposes" (National Basketball Association).

For player analysis, our dataset includes a player's information such as age, team, game_played, 2-point field goals, 3-point field goals, points scored, blocks, assists, injuries (body part)

For team analysis, our dataset includes information on a specific NBA team's Wins, Losses, make_Playoffs, Championships won, games_played, Points Per Game, Opponent Points Per Game

For Meta-Game Analysis our datasets include the following variables that will help us analyze how basketball has changed over the years: season, games, field goal, points_per_game, offensive/defensive rebounds

The data that is being used for the above three topics have been collected using web scraping. The web scraper in Python has been built using two main libraries that are `BeautifulSoup4` and `urllib`. The data available on these websites are mainly built with CSS and HTML, which allows us to parse through the web data in HTML format for each component required from the webpage.

Parsing Contracts data for Players playing in 2022-2023 Season

WEBPAGE: <https://www.basketball-reference.com/contracts/players.html>

The following example is one of the simpler examples of the scripts that we have built, the following code allows us to extract the subheader for contracts by finding all the "tr" attributes for the table-player contracts. One of our questions deals with player injuries and if there is a relationship between player injuries and the contracts that the said players are currently employed on.

For instance, Derrick Rose was one of the best young players when he was drafted in 2009, he won the MVP in 2011 (The youngest player to ever do so). Unfortunately, he suffered an ACL tear in 2012, and his career took a major dip from his peak. Since 2018, he has contracted with 22 lower body injuries. The goal here is to understand the kind of contract an injury-prone player like him commands right now and to correlate it with other players with similar situations.

In our SQL joining scripts, this data is being used similar to the above example, that is to join this data with injury-related data to analyze the trends of how injuries affect a player getting a contract, how relevant this high-level observation is, and whether there are any outliers to this.

```
In [ ]: #Importing required libraries
from urllib.request import urlopen
from bs4 import BeautifulSoup
import pandas as pd
import time
import plotly.express as px
import numpy as np
from plotly import graph_objs as go
import sqlalchemy as sq
from sqlalchemy import update, delete, insert
import mysql.connector
from mysql.connector import errorcode
```

In []:

```

years = []
#Creating list with nba years
for year in range(1980,2024):
    years.append(year)

#Looping through each year in years list
for i in years:
    #Loading Website to variable
    url = "https://www.basketball-
reference.com/leagues/NBA_"+str(i)+"_per_game.html"
    html = urlopen(url)
    soup = BeautifulSoup(html, features="html.parser")
    #Parsing the required HTML elements
    data = soup.find_all('table', id="per_game_stats")
[0].findAll('tr')
    pergame_stats = [[td.getText() for td in data[i].findAll('td')]
for i in range(0,len(data))]
    head = [[th.getText() for th in data[i].findAll('th')] for i in
range(0,len(data))]
    pergame_stats=pergame_stats[1:]
    header = head[1:]
    #Setting headers and data
    pergame_stat = pd.DataFrame(pergame_stats, columns = ['Player',
'Pos', 'Age', 'Tm', 'G', 'GS', 'MP', 'FG', 'FGA', 'FG%', '3P',
'3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%', 'ORB',
'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'])
    pergame_stat["Season"]= i
    pergame_stat.dropna(inplace = True)
    #Writing data to file
    pergame_stat.to_csv("pergamebasics.csv", mode='a', index=False)
    print(str(i)," year done...")
    time.sleep(5)

```

1980	year	done...
1981	year	done...
1982	year	done...
1983	year	done...
1984	year	done...
1985	year	done...
1986	year	done...
1987	year	done...
1988	year	done...
1989	year	done...
1990	year	done...
1991	year	done...
1992	year	done...
1993	year	done...
1994	year	done...
1995	year	done...
1996	year	done...
1997	year	done...
1998	year	done...
1999	year	done...
2000	year	done...
2001	year	done...
2002	year	done...

```

-----
KeyboardInterrupt                                Traceback (most recent call last)
Cell In [2], line 11
      9 url = "https://www.basketball-reference.com/leagues/NBA_"+str(i)+"_per
_game.html"
     10 html = urlopen(url)
--> 11 soup = BeautifulSoup(html, features="html.parser")
     12 #Parsing the required HTML elements
     13 data = soup.find_all('table', id="per_game_stats")[0].findAll('tr')

File ~/opt/anaconda3/lib/python3.9/site-packages/bs4/__init__.py:333, in Beaut
ifulSoup.__init__(self, markup, features, builder, parse_only, from_encoding,
exclude_encodings, element_classes, **kwargs)
     331 self.builder.initialize_soup(self)
     332 try:
--> 333     self._feed()
     334     success = True
     335     break

File ~/opt/anaconda3/lib/python3.9/site-packages/bs4/__init__.py:451, in Beaut
ifulSoup._feed(self)
     448 # Convert the document to Unicode.
     449 self.builder.reset()
--> 451 self.builder.feed(self.markup)
     452 # Close out any unfinished strings and close all the open tags.
     453 self.endData()

File ~/opt/anaconda3/lib/python3.9/site-packages/bs4/builder/_htmlparser.py:39
9, in HTMLParserTreeBuilder.feed(self, markup)
     397 parser.soup = self.soup
     398 try:
--> 399     parser.feed(markup)
     400     parser.close()
     401 except HTMLParseError as e:

File ~/opt/anaconda3/lib/python3.9/html/parser.py:110, in HTMLParser.feed(sel
f, data)
     104 r""""Feed data to the parser.
     105
     106 Call this as often as you want, with as little or as much text
     107 as you want (may include '\n').
     108 """
     109 self.rawdata = self.rawdata + data
--> 110 self.goahead(0)

File ~/opt/anaconda3/lib/python3.9/html/parser.py:170, in HTMLParser.goahead(s
elf, end)
     168 if startswith('<', i):
     169     if starttagopen.match(rawdata, i): # < + letter
--> 170         k = self.parse_starttag(i)
     171     elif startswith("</", i):
     172         k = self.parse_endtag(i)

```

```

File ~/opt/anaconda3/lib/python3.9/html/parser.py:302, in HTMLParser.parse_starttag(self, i)
    300 def parse_starttag(self, i):
    301     self.__starttag_text = None
--> 302     endpos = self.check_for_whole_start_tag(i)
    303     if endpos < 0:
    304         return endpos
KeyboardInterrupt:

```

Parsing through NBA team data from 1980 to 2023

WEBPAGE: https://www.basketball-reference.com/leagues/NBA_stats_per_game.html

This data source hosts overall team statistics from 1980 to 2023, this includes per-game, shooting, and advanced statistics from all teams per season for the said timespan. This data is essential to talk about the overview of how the NBA has changed over the years. This includes rule changes like the introduction of the 3-point line, its increase in rate, and the boom of its usage after Steph Curry joined the NBA back in 2008. This dataset will help my teammates in their meta-analysis of the game through graphical representation.

The use case of this specific data source can be seen when the "Meta" of Basketball has been discussed and especially talking about the rise of 3-point shooting in the NBA, this along with the meteoric rise of Steph Curry within the League.

```

In [ ]: #Loading Website to variable
url = "https://www.basketball-
reference.com/leagues/NBA_stats_per_game.html"
html = urlopen(url)
soup = BeautifulSoup(html, features="html.parser")

#Parsing the required HTML elements
data = soup.find_all('table', id="stats")[0].findAll('tr')
overall = soup.find_all('table', id="stats")[0].findAll('tr')
overall_stats = [[td.getText() for td in data[i].findAll('td')] for
i in range(0,len(data))]
overall_stats = overall_stats[2:]

#Setting headers and data
overall_header = [[th.getText() for th in data[i].findAll('th')] for
i in range(0,len(data))]
overall_stat = pd.DataFrame(overall_stats, columns = ['Season', 'Lg',
'Age', 'Ht', 'Wt', 'G', 'MP', 'FG', 'FGA', '3P', '3PA', 'FT', 'FTA',

```

```
'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS', 'FG%',
'3P%', 'FT%', 'Pace', 'eFG%', 'TOV%', 'ORB%', 'FT/FGA', 'ORTg'])
overall_stat.dropna(inplace = True)

#Writing data to file
overall_stat.to_csv("stat_overall.csv", index=False)
```

Parsing per game statistics per player for each team

WEBPAGES:

<https://www.basketball-reference.com/teams/BOS/2023.html>

<https://www.basketball-reference.com/players/t/tatumja01.html>

<https://www.basketball-reference.com/players/t/tatumja01/gamelog/2018>

Algorithm From a code point of view, the following is happening:- 1.For each team in the NBA, set abbreviation and build URL (a) 2.Open web page using urllib and beautifulsoup, find HREF for the roster div- Getting all players for a team 3.Loop through each player in each team 4.List all seasons the player has played in . For example Jayson Tatum has played in 6 NBA seasons. 5.Build the URL using each seasons code (b) 6.For each season, go to per game stats and pull data (c) 7.Store data in csv file in append mode 8.Exit loop

The data collected from the script below becomes the fundamental basis of the "Game Simulator" that the team has built. The data sources mentioned above parses through the per-game statistics of all currently active players from each team in the league. The data is then cleaned, type-corrected, and split into the most recent season which is 2022-2023. This data is used by us to predict the score per game per player for our simulations. These scores aggregate to a total value which when compared to an opposing team allows us to predict an outcome for a head-to-head matchup. Models similar to the ones we have built can be seen on betting sites like Fanduels, PointBet, and some Fantasy Leagues where each statistic that is each point, assist, rebound, steal or block a player gets amounts to a fantasy score. The player with the highest fantasy score ends up being the winner.

```
In [ ]: # Setting batches of teams since parsing through multiple players
per game stats takes some time to process

batch1 = ["BOS", "TOR", "PHI", "BRK", "NYK"]
batch2 = ["MIL", "CLE", "IND", "CHI", "DET"]
batch3 = ["ATL", "WAS", "MIA", "ORL", "CHO"]
batch4 = ["UTA", "DEN", "POR", "MIN", "OKC"]
batch5 = ["PHO", "SAC", "LAC", "GSW", "LAL"]
batch6 = ["MEM", "NOP", "DAL", "SAS", "HOU"]
```



```

# change your team selection here!
teams = batch1
# csv file path
csvFilePath =
f"../Cleaned_Datasets/cummalativePerGame"+"batch1"+" .csv"
#Looping though each team in batch
for team in teams:
    print(str(team)+" starting...")
    #Loading URL to variable
    url = "https://www.basketball-
reference.com/teams/"+str(team)+"/2023.html"
    html = urlopen(url)
    soup = BeautifulSoup(html, features="html.parser")
    #Parsing all players in each team
    data = soup.find_all('table', id="roster")[0].find_all("td")
    #Looping through each player in parsed data
    for player in data:
        for a in player.find_all("a", href=True):
            time.sleep(1)
            if "players" in a["href"]:
                #Loading URL to variable
                url1 = "https://www.basketball-
reference.com"+str(a["href"])
                html = urlopen(url1)
                soup = BeautifulSoup(html, features="html.parser")
                playername = soup.find_all('div', id="meta")[
                    0].find_all("span")[0].text.strip()
                data1 = soup.find_all('div',
id="bottom_nav_container")[0]
                print("Scraping "+str(playername)+" data")
                #Looping through per game stats for each player
                for data in data1.find_all("a"):
                    time.sleep(1)
                    if "gamelog-playoffs" in data.attrs["href"]:
                        continue
                    elif "gamelog" in data.attrs["href"]:
                        #Loading URL to variable
                        url2 = "https://www.basketball-

```

```

reference.com" + \
        str(data.attrs["href"])
    html = urlopen(url2)
    soup = BeautifulSoup(html,
features="html.parser")
    #Parsing the required HTML elements
    pergamestats = soup.find_all('table',
id="pgl_basic")[
        0].findAll("tr")
    pergamestatsdata = [[td.getText() for td in
pergamestats[i].findAll(
        'td')] for i in range(0,
len(pergamestats))]
    pergamestatsheader = [[th.getText() for th
in pergamestats[i].findAll(
        'th')] for i in range(0,
len(pergamestats))]
    pergamestatsdata = pergamestatsdata[1:]
    #Setting headers and data
    pergamestat = pd.DataFrame(pergamestatsdata,
columns=['G', 'Date', 'Age', 'Tm', '\xa0', 'Opp', '\xa0', 'GS',
'MP', 'FG', 'FGA',
        'FG%', '3P',
'3PA', '3P%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL',
'BLK', 'TOV', 'PF', 'PTS', 'GmSc', '+/-'])
    pergamestat["Player"] = playername
    #Writing data to file
    pergamestat.to_csv(
        csvFilePath, mode='a', index=False)
    else:
        continue
    else:
        continue
    print(str(team), " done...")

```

Parsing player continuity data since 1952 season

WEBPAGE: <https://www.basketball-reference.com/friv/continuity.html>

Team chemistry is an intangible which cannot be evaluated by simple statistics but a team's ability to retain a player becomes an important aspect of team building and the overall morale of the team. For instance, the San Antonio Spurs with their core of Tim Duncan, Tony Parker, and Manu Ginobili along with their coach Greg Popovich was one of the longest groups of people working together in the NBA in the 2000s and 2010s. Trying to quantify the team's bonds is not possible due to the availability of the stat but we can quantify if that team bond actually led to any success for the team or not. Joining and visualizing the continuity of a team with the overall team success is something that was in scope for our work.

```
In [ ]: #Loading URL to variable
url = "https://www.basketball-reference.com/friv/continuity.html"
html = urlopen(url)
soup = BeautifulSoup(html, features="html.parser")

#Parsing the required HTML elements
data = soup.find_all('table', id="continuity")[0].findAll('tr')
continuity = [[td.getText() for td in data[i].findAll('td')] for i
in range(0,len(data))]
head = [[th.getText() for th in data[i].findAll('th')] for i in
range(0,len(data))]
print(head[1:])
continuity=continuity[1:]
#Setting headers and data
continuity_stat = pd.DataFrame(continuity, columns = ['ATL', 'BOS',
'CHA', 'CHI', 'CLE', 'DAL', 'DEN', 'DET', 'GSW', 'HOU', 'IND',
'LAC', 'LAL', 'MEM',
'MIA', 'MIL',
'MIN', 'NJN', 'NOH', 'NYK', 'OKC', 'ORL', 'PHI', 'PHO', 'POR',
'SAC', 'SAS', 'TOR', 'UTA', 'WAS'])
continuity_stat["Season"] = head[1:]
#Writing data to file
continuity_stat.to_csv("C:\Code\continuity.csv", index=False)
```

Parsing individual accomplishment's data since the inception of awards

WEBPAGE: <https://www.basketball-reference.com/awards/mvp.html>

The common thought is that the top teams will have the top players playing for them.

Awards are one of the most sound metric to decide who these top players are. The purpose

of getting this data is to understand how often the teams with the so-called "Best Players" actually end up having championship success. For example, the Boston Celtics went up against the Golden State Warriors in the 2022 NBA Finals. The Celtics had the reigning Defensive Player of the Year. This award however became useless since Steph Curry from the Warriors averaged 31.1 points in the 6-game span and won the title. Looking into how relevant these award winners are can be good from a team-building point of view.

```
In [ ]: awards = ['mvp', 'roy', 'dpoy', 'smoy', 'mip']
        executive_awards=['coy', 'eoy']

#Looping through each award in the awards list
for award in awards:
    #Loading URL to variable
    url = "https://www.basketball-
reference.com/awards/"+str(award)+".html"
    html = urlopen(url)
    soup = BeautifulSoup(html, features="html.parser")
    id = str(award)+"_NBA"
    #Parsing the required HTML elements
    data = soup.find_all('table', id=id)[0].find_all('tr')
    award_stats = [[td.getText() for td in data[i].find_all('td')]
for i in range(0,len(data))]
    head = [[th.getText() for th in data[i].find_all('th')] for i in
range(0,len(data))]
    award_stats=award_stats[2:]
    h = head[1]
    header = h[1:]
    #Setting headers and data
    award_stats = pd.DataFrame(award_stats, columns = header)
    award_stats['Award'] = award
    #Writing data to file
    award_stats.to_csv("C:\Code\statsaward.csv", mode='a',
index=False)
    print(str(award), " award done...")
```

Parsing advanced statistics from 1980 to 2023

WEBPAGE:https://www.basketball-reference.com/leagues/NBA_2022_advanced.html#advanced_stats::per

Ever since sports analytics became a popular profession, multiple teams started hiring and tracking advanced statistics for the player and team metrics. This includes statistics like how often a player turns over the ball or how much of the team's possessions goes through the said player. Moreover, we can look into Player Efficiency Ratings, win share percentages, and true shooting percentages as a metric to evaluate the overall confidence in a player's skills.

```
In [ ]: years = []

#Setting NBA years into years list
for year in range(1980,2024):
    years.append(year)

#Looping through each year in years list
for season in years:
    time.sleep(2)
    #Loading URL to variable
    url = "https://www.basketball-
reference.com/leagues/NBA_"+str(season)+"_advanced.html#advanced_stats
    html = urlopen(url)
    soup = BeautifulSoup(html, features="html.parser")

    #Parsing the required HTML elements
    data = soup.find_all('table', id="advanced_stats")
    [0].findAll('tr')
    player_stats = [[td.getText() for td in data[i].findAll('td')]
for i in range(0,len(data))]
    player_header = [[th.getText() for th in data[i].findAll('th')]
for i in range(0,len(data))]
    player_stats = player_stats[1:]

    #Setting headers and data
    player_stat = pd.DataFrame(player_stats,columns = ['Player',
'Pos', 'Age', 'Tm', 'G', 'MP', 'PER', 'TS%', '3PAr', 'FTr', 'ORB%',
'DRB%', 'TRB%', 'AST%',
'STL%',
'BLK%', 'TOV%', 'USG%', '\xa0', 'OWS', 'DWS', 'WS', 'WS/48', '\xa0',
'OBPM', 'DBPM', 'BPM', 'VORP'])
    player_stat.dropna(inplace = True)
```

```

player_stat['Season'] = season
#Writing data to file

player_stat.to_csv("C:\Code\playerstat.csv", mode='a',
index=False)

print(str(season), " season done...")

```

Data Exploration

Guiding Question 1: How have rule changes or player/team behaviors affected team stats?

The NBA implemented a new rule in 1979 with the introduction of the 3-point line, and not much changed right away. It is statistically proven that scoring one-third of your shots from behind the 3-point line will yield a higher score than a player scoring half their shots from inside the line. In other words: Players Shooting as many 3-pointers as possible will lead to a higher score. But the question we would like to analyze is whether teams have adopted their shooting style to shoot more 3-pointers. What is the general trend we see in teams scoring 3-pointers? After analyzing should we suggest coaches to make sure that players are well-trained in taking 3-pointer shots?

```

In [ ]: #making a secure connection to my MariaDB Database
engine = sq.create_engine('mysql+mysqlconnector://L01-
7:2XW^s3H6@datasciencedb.ucalgary.ca/L01-7')

```

```

In [ ]: #Importing the dataset stat_overall into pandas dataframe
df = pd.read_csv("stat_overall.csv")

#Filling all the NaN in the dataframe with 0
df.fillna(0,inplace=True)

#Sorting the the seasons from lowest to highest
overallstat=df[::-1]
display(overallstat)

```

```

In [ ]: #Inserting the dataset into MariaDB using SQLalchemy

query = pd.read_sql_query('SELECT * FROM stat_overall', engine)
query.head()

```

Out []:

	Season	Lg	Age	Ht	Wt	G	MP	FG	FGA	3P	...	PTS	FG%%	3P%%	FT%%	Pac
0	1946-47	BAA	0.0	0	0	331	0.0	25.9	92.9	0.0	...	67.8	0.279	0.0	0.641	0
1	1947-48	BAA	0.0	0	0	192	0.0	27.2	96.0	0.0	...	72.7	0.284	0.0	0.675	0
2	1948-49	BAA	0.0	0	0	360	0.0	29.0	88.7	0.0	...	80.0	0.327	0.0	0.703	0
3	1949-50	NBA	0.0	0	0	561	0.0	28.2	83.1	0.0	...	80.0	0.340	0.0	0.714	0
4	1950-51	NBA	0.0	0	0	354	0.0	29.8	83.6	0.0	...	84.1	0.357	0.0	0.733	0

5 rows x 31 columns

In []:

```
#selecting only necessary columns for our analysis
query= '''
SELECT Season, 3PA FROM stat_overall
'''
overallstat = pd.read_sql_query(query, engine)
display(overallstat)
```

	Season	3PA
0	1946-47	0.0
1	1947-48	0.0
2	1948-49	0.0
3	1949-50	0.0
4	1950-51	0.0
...
72	2018-19	32.0
73	2019-20	34.1
74	2020-21	34.6
75	2021-22	35.2
76	2022-23	34.3

77 rows x 2 columns

Three-pointer trends over time

In []:

```
#Scatter Graph
fig = px.scatter(overallstat, x="Season", y="3PA", color="Season",
```

```

trendline="ols",
    )
fig.update_layout(
    title_text="3P Trend",
    showlegend=False,
    width=800, height=500
)

fig.add_vline(x='1978-79', line_width=3, line_dash="dash",
line_color="green", name='3 Pointer Started')
fig.add_vline(x='2009-10', line_width=3, line_dash="dash",
line_color="red", name='Steph Curry Drafted')

fig.show()

```

From the above graph, we can see that after the three-point line was introduced in 1979 (Green dashed line), there is a steady rise in the number of three-point attempts.

The red-dashed line at 2009 represents when Steph Curry was drafted. He later set the NBA seasonal record for most three-pointers in 2012-13 with 272. With his introduction to the NBA, the entire sport was forced to shift to focusing on three-point shots in a desperate bid to keep up.

We can see a steep increase after the year 2009 in three-pointer attempts demonstrating how one player has inspired other NBA teams to take more three-point shots.

Guiding Question 2: How does player continuity affect the overall success of the team?

Throughout the history of the NBA, one of the most frequently overlooked ingredients in the recipe for winning a championship has been team consistency. In order to secure a win it is important for NBA teams to stick to the same players so that the teams can evolve together and do better? As Bob Myers, the general manager of the Golden State Warriors said, "Playing together with the same group of people for a long time makes you better. It just does."

In order to test this theory we will utilize following datasets

team_continuity.csv : this dataset reflects rotation continuity on NBA teams from 1952-2022. Continuity is defined by the overlap in minutes played by the same players from one year to the next. Or 100 - % change in rotation minutes between years.

`teamstats.csv` : this dataset contains number of Wins and Loss for every team of every season

`team_mapping` : this contains Team names and its abbreviations

```
In [ ]: #Importing the datasets into pandas dataframe
df1 = pd.read_csv("team_continuity.csv")
df2 = pd.read_csv("teamstats.csv")
df3 = pd.read_csv("team_mapping.csv")
```

```
In [ ]: #Importing the team_continuity.csv dataset into pandas dataframe
df1 = pd.read_csv("team_continuity.csv")

#Filling all the team continuity NaN in the dataframe with 0
df1.fillna(0,inplace=True)
display(df1)

#Pivot the team_continuity data to match the format with other datasets to be used for joining
team_continuity = pd.melt(df1, id_vars = 'Season', value_vars =
['ATL','BOS','CHA','CHI','CLE','DAL','DEN','DET','GSW','HOU','IND','LA',
,var_name = 'Team', value_name = 'Coninuity%'])

display(team_continuity)
```

```
In [ ]: # Importing team continuity to SQL
# team_continuity.to_sql('team_continuity', engine)

query = '''
SELECT * FROM team_continuity;
'''

team_continuity = pd.read_sql_query(query, engine)

team_continuity.head()
```

```
Out [ ]:
```

	row_names	Season	ATL	BOS	CHA	CHI	CLE	DAL	DEN	DET	...	OKC	ORL	PHI	PH
0	1	2016-17	0.78	0.82	0.78	0.51	0.87	0.55	0.75	0.73	...	0.56	0.51	0.51	0.7
1	2	2015-16	0.87	0.85	0.51	0.9	0.84	0.55	0.7	0.50	...	0.93	0.76	0.63	0.5
2	3	2014-15	0.88	0.50	0.61	0.64	0.39	0.38	0.62	0.57	...	0.66	0.52	0.29	0.8
3	4	2013-14	0.53	0.59	0.64	0.69	0.6	0.48	0.62	0.59	...	0.89	0.81	0.37	0.4
4	5	2012-13	0.52	0.53	0.49	0.63	0.55	0.39	0.82	0.71	...	0.8	0.26	0.63	0.4

5 rows x 32 columns

```
In [ ]:
```

```
#Importing the team_continuity.csv dataset into pandas dataframe

df2 = pd.read_csv("teamstats.csv")
df2=df2.drop(df2.columns[[8,15]], axis=1) #delete unnamed and NAN
columns

display(df2)

df2.drop(['SRS', 'W/L%', 'Lg', 'Finish', 'Playoffs', 'Coaches', 'Top
WS'], axis=1, inplace=True)
df2=df2[df2.Season != 'Season'] # Cleaning row headers in middle of
every data

df2.fillna(0,inplace=True) # #Filling all the teamstat NaN in the
dataframe with 0

#Changing the datatypes to numeric
df2[['Pace', 'W', 'L', 'Rel Pace', 'ORTg', 'Rel ORtg', 'DRtg', 'Rel
DRtg']] = df2[['Pace', 'W', 'L', 'Rel Pace', 'ORTg', 'Rel ORtg', 'DRtg',
'Rel DRtg']].apply(pd.to_numeric)

#Adding Winning percentage column after calculating
df2['W%'] = df2['W']/(df2['W']+df2['L'])

print(df2.dtypes)
display(df2)
```

```
In [ ]: # Importing teamstats to SQL
# df2.to_sql('teamstats', engine)

query = '''

SELECT * FROM teamstats;
'''

teamstat = pd.read_sql_query(query, engine)

display(teamstat)
```

	index	Season	Team	W	L	Pace	Rel Pace	ORtg	Rel ORtg	DRtg	Rel DRtg	W%
0	0	2022-23	Brooklyn Nets	2	6	98.5	-1.2	112.2	0.1	119.0	6.9	0.250000
1	1	2021-22	Brooklyn Nets*	44	38	99.0	0.8	113.6	1.6	112.8	0.8	0.536585
2	2	2020-21	Brooklyn Nets*	48	24	99.5	0.3	118.3	6.0	113.8	1.5	0.666667
3	3	2019-20	Brooklyn Nets*	35	37	101.4	1.1	108.9	-1.7	109.5	-1.1	0.486111
4	4	2018-19	Brooklyn Nets*	42	40	100.8	0.8	109.6	-0.8	109.7	-0.7	0.512195
...
1628	1628	1950-51	New York Knicks*	36	30	94.8	9.3	88.0	2.9	87.6	2.5	0.545455
1629	1629	1949-50	New York Knicks*	40	28	0.0	0.0	0.0	0.0	0.0	0.0	0.588235
1630	1630	1948-49	New York Knicks*	32	28	0.0	0.0	0.0	0.0	0.0	0.0	0.533333
1631	1631	1947-48	New York Knicks*	26	22	0.0	0.0	0.0	0.0	0.0	0.0	0.541667
1632	1632	1946-47	New York Knicks*	33	27	0.0	0.0	0.0	0.0	0.0	0.0	0.550000

1633 rows x 12 columns

```
In [ ]: #Importing the team_mapping.csv dataset into pandas dataframe

#df3 = pd.read_csv("team_mapping.csv")

# Importing team map to SQL
#df3.to_sql('team_map', engine)
```

```

query = '''
SELECT * FROM team_mapping;
'''

teammmap = pd.read_sql_query(query, engine)

display(teammmap)

```

	Team_Name	Abbreviation
0	Brooklyn Nets	BKN
1	New Jersey Nets	NJN
2	New York Nets	NYN
3	New Jersey Americans	NJA
4	Cleveland Cavaliers	CLE
...
57	Milwaukee Hawks	MLH
58	Tri-Cities Blackhawks	TRI
59	Philadelphia 76ers	PHI
60	Syracuse Nationals	SYR
61	New York Knicks	NYK

62 rows × 2 columns

```

In [ ]: #Joining the three datasets using SQL queries

query = '''

SELECT subquery.Season,subquery.Team,
       subquery.Abbreviation as Team_Abrv,round(subquery.`W%`,2) as
`Win%`,tc.`Coninuity%` as `Continuity%`
FROM (
    SELECT ts.Season,
           REPLACE(ts.Team, '*', '') as Team,
           tm.Abbreviation,
           ts.`W%`
    FROM teamstats as ts
        JOIN team_mapping as tm
        on REPLACE(ts.Team, '*', '') = tm.Team_Name
    ) as subquery
JOIN team_conti_unpivot as tc

```

```

on subquery.Abbreviation = tc.Team and tc.Season =
subquery.Season;
'''
team_wc = pd.read_sql_query(query, engine)

display(team_wc)

```

	Season	Team	Team_Abrv	Win%	Continuity%
0	2022-23	Atlanta Hawks	ATL	0.62	0.64
1	2021-22	Atlanta Hawks	ATL	0.52	0.84
2	2020-21	Atlanta Hawks	ATL	0.57	0.48
3	2019-20	Atlanta Hawks	ATL	0.30	0.49
4	2018-19	Atlanta Hawks	ATL	0.35	0.45
...
1275	1978-79	Washington Bullets	WAS	0.66	0.97
1276	1977-78	Washington Bullets	WAS	0.54	0.75
1277	1976-77	Washington Bullets	WAS	0.59	0.69
1278	1975-76	Washington Bullets	WAS	0.59	0.82
1279	1974-75	Washington Bullets	WAS	0.73	0.81

1280 rows × 5 columns

Example: Chicago Bulls team continuity vs. Win Percentage

Lets check the data for the Chicago Bulls. The team won the NBA championships in 1991, 1992, 1993, 1996, 1997, and 1998. When we look at the above data, team continuity for the Chicago Bulls has stayed above 80% throughout those championship years. This is also one of the factors why the Chicago Bulls were able to win so many NBA Championships in a row.

We can see that in 1996-97 for Chicago Bulls (CHI) when their team continuity was 97% their Win percentage was as high as 84%.

```

In [ ]: team_wc.loc[(team_wc['Team_Abrv'] == 'CHI') &
(team_wc['Continuity%'] > .70)]

```

Out[]:

	Season	Team	Team_Abrv	Win%	Continuity%
149	2022-23	Chicago Bulls	CHI	0.56	0.87
156	2015-16	Chicago Bulls	CHI	0.51	0.90
160	2011-12	Chicago Bulls	CHI	0.76	0.93
162	2009-10	Chicago Bulls	CHI	0.50	0.72
163	2008-09	Chicago Bulls	CHI	0.50	0.73
164	2007-08	Chicago Bulls	CHI	0.40	0.76
166	2005-06	Chicago Bulls	CHI	0.50	0.82
174	1997-98	Chicago Bulls	CHI	0.76	0.92
175	1996-97	Chicago Bulls	CHI	0.84	0.97
176	1995-96	Chicago Bulls	CHI	0.88	0.81
177	1994-95	Chicago Bulls	CHI	0.57	0.80
179	1992-93	Chicago Bulls	CHI	0.70	0.86
180	1991-92	Chicago Bulls	CHI	0.82	0.96
181	1990-91	Chicago Bulls	CHI	0.74	0.89
182	1989-90	Chicago Bulls	CHI	0.67	0.83
183	1988-89	Chicago Bulls	CHI	0.57	0.77
184	1987-88	Chicago Bulls	CHI	0.61	0.71
187	1984-85	Chicago Bulls	CHI	0.46	0.72
188	1983-84	Chicago Bulls	CHI	0.33	0.71
190	1981-82	Chicago Bulls	CHI	0.41	0.89
194	1977-78	Chicago Bulls	CHI	0.49	0.79
196	1975-76	Chicago Bulls	CHI	0.29	0.72
198	1973-74	Chicago Bulls	CHI	0.66	0.96
199	1972-73	Chicago Bulls	CHI	0.62	0.82
200	1971-72	Chicago Bulls	CHI	0.70	0.73
201	1970-71	Chicago Bulls	CHI	0.62	0.75
203	1968-69	Chicago Bulls	CHI	0.40	0.73
204	1967-68	Chicago Bulls	CHI	0.35	0.77

Exploring Win Percentage vs. Team Continuity for the whole NBA

In []:

```
# Calculating Win%/Continuity% to find out which team benefited the most (Most wins) with Team Continuity being their factor
```

```
team_wc['Ratio'] = team_wc['Win%']/team_wc['Continuity%']
display(team_wc)
```

	Season	Team	Team_Abrv	Win%	Continuity%	Ratio
0	2022-23	Atlanta Hawks	ATL	0.62	0.64	0.968750
1	2021-22	Atlanta Hawks	ATL	0.52	0.84	0.619048
2	2020-21	Atlanta Hawks	ATL	0.57	0.48	1.187500
3	2019-20	Atlanta Hawks	ATL	0.30	0.49	0.612245
4	2018-19	Atlanta Hawks	ATL	0.35	0.45	0.777778
...
1275	1978-79	Washington Bullets	WAS	0.66	0.97	0.680412
1276	1977-78	Washington Bullets	WAS	0.54	0.75	0.720000
1277	1976-77	Washington Bullets	WAS	0.59	0.69	0.855072
1278	1975-76	Washington Bullets	WAS	0.59	0.82	0.719512
1279	1974-75	Washington Bullets	WAS	0.73	0.81	0.901235

1280 rows x 6 columns

```
In [ ]: #Importing Conf Mapping

query = '''
SELECT * FROM team_mapping_conf;
'''

confdata = pd.read_sql_query(query, engine)

# Merging with Conference data
conf_merge=team_wc.merge(confdata, on='Team_Abrv', how='left')
display(conf_merge)
```

	Season	Team	Team_Abrv	Win%	Continuity%	Ratio	Team_Name	Conference
0	2022-23	Atlanta Hawks	ATL	0.62	0.64	0.968750	Atlanta Hawks	East
1	2021-22	Atlanta Hawks	ATL	0.52	0.84	0.619048	Atlanta Hawks	East
2	2020-21	Atlanta Hawks	ATL	0.57	0.48	1.187500	Atlanta Hawks	East
3	2019-20	Atlanta Hawks	ATL	0.30	0.49	0.612245	Atlanta Hawks	East
4	2018-19	Atlanta Hawks	ATL	0.35	0.45	0.777778	Atlanta Hawks	East
...
1275	1978-79	Washington Bullets	WAS	0.66	0.97	0.680412	Washington Wizards	East
1276	1977-78	Washington Bullets	WAS	0.54	0.75	0.720000	Washington Wizards	East
1277	1976-77	Washington Bullets	WAS	0.59	0.69	0.855072	Washington Wizards	East
1278	1975-76	Washington Bullets	WAS	0.59	0.82	0.719512	Washington Wizards	East
1279	1974-75	Washington Bullets	WAS	0.73	0.81	0.901235	Washington Wizards	East

1280 rows × 8 columns

In []:

```
# Filtering by NBA Teams who are currently active

df = pd.DataFrame(conf_merge.groupby(['Team_Abrv']).agg(
    median=('Ratio', 'median')))

tempdf = conf_merge
#display(df)
result=tempdf.merge(df, on='Team_Abrv', how='left')
#del tempdf['Median']
team_wc_filtered=result.sort_values(by=['median'])
display(team_wc_filtered)
# tempdf['Median'] = df
# display(tempdf.head(70))
```


	Season	Team	Team_Abrv	Win%	Continuity%	Ratio	Team_Name	Conference	
622	1984-85	Los Angeles Clippers	LAC	0.38	0.55	0.690909	Los Angeles Clippers	West	0.
605	2001-02	Los Angeles Clippers	LAC	0.48	0.81	0.592593	Los Angeles Clippers	West	0.
606	2000-01	Los Angeles Clippers	LAC	0.38	0.54	0.703704	Los Angeles Clippers	West	0.
607	1999-00	Los Angeles Clippers	LAC	0.18	0.61	0.295082	Los Angeles Clippers	West	0.
608	1998-99	Los Angeles Clippers	LAC	0.18	0.65	0.276923	Los Angeles Clippers	West	0.
...
724	2006-07	Miami Heat	MIA	0.54	0.92	0.586957	Miami Heat	East	0
723	2007-08	Miami Heat	MIA	0.18	0.52	0.346154	Miami Heat	East	0
722	2008-09	Miami Heat	MIA	0.52	0.57	0.912281	Miami Heat	East	0
732	1998-99	Miami Heat	MIA	0.66	0.77	0.857143	Miami Heat	East	0
728	2002-03	Miami Heat	MIA	0.30	0.66	0.454545	Miami Heat	East	0

1280 rows × 9 columns

```
In [ ]: #Filtering by last 10 years of Season data

recent = team_wc_filtered.loc[(team_wc_filtered.Season > "2011-2012")] #& (df.carrier == "B6")
display(recent)
```

	Season	Team	Team_Abrv	Win%	Continuity%	Ratio	Team_Name	Conference	
584	2022-23	Los Angeles Clippers	LAC	0.50	0.89	0.561798	Los Angeles Clippers	West	0.
586	2020-21	Los Angeles Clippers	LAC	0.65	0.71	0.915493	Los Angeles Clippers	West	0.
587	2019-20	Los Angeles Clippers	LAC	0.68	0.55	1.236364	Los Angeles Clippers	West	0.
588	2018-19	Los Angeles Clippers	LAC	0.59	0.66	0.893939	Los Angeles Clippers	West	0.
589	2017-18	Los Angeles Clippers	LAC	0.51	0.36	1.416667	Los Angeles Clippers	West	0.
...
714	2016-17	Miami Heat	MIA	0.50	0.50	1.000000	Miami Heat	East	0.
715	2015-16	Miami Heat	MIA	0.59	0.65	0.907692	Miami Heat	East	0.
716	2014-15	Miami Heat	MIA	0.45	0.50	0.900000	Miami Heat	East	0.
717	2013-14	Miami Heat	MIA	0.66	0.91	0.725275	Miami Heat	East	0.
718	2012-13	Miami Heat	MIA	0.80	0.82	0.975610	Miami Heat	East	0.

294 rows × 9 columns

In []:

```
#Scatter Graph of all teams that with their respective continuity for all seasons
fig = px.scatter(team_wc_filtered, x="Continuity%", y="Win%",
color="Team_Abrv", trendline="ols",
)
fig.update_layout(
    title_text="Winning Percentage vs. Team Continuity",
    width=800, height=600
)
fig.update_yaxes(tickformat=".0%")
fig.update_xaxes(tickformat=".0%")
fig.show()
```

```
In [ ]: #Scatter Graph of all active teams divided into EAST and WEST from
Last 10 seasons
fig = px.scatter(recent, x="Continuity%", y="Win%",
color="Conference", trendline="ols",
)
fig.update_layout(
    title_text="Winning Percentage vs. Team Continuity Past 10 Years
By Conference",
    width=800, height=600
)
fig.update_yaxes(tickformat=".0%")
fig.update_xaxes(tickformat=".0%")
fig.show()
```

From the the graph above, we can conclude that there exists a postive correlation between Winning Percentage and a team's continuity percentage. This means that it is more likely for a team that sticks together to secure a win for in that season.

```
In [ ]: fig = px.scatter(team_wc_filtered, x="Ratio", y="Team_Abrv",
color="median", orientation='h',
labels={
    "Team_Abrv": "Teams",
    "Ratio": "Win/Continuity",
    "median": "Ratio"
})
fig.update_layout(
    title_text="Winning Percentage to Team Continuity Ratio",
    height=600
)
fig.show()
```

From our second graph, we can observe the ratio of Winning Percentage to Continuity over all the seasons, sorted from highest to lowest average ratio. This figure focuses on teams that have the least roster retention, and acts as a complementary metric for the previous scatterplots. The greater the ratio, on average, the more wins a team is able to produce with fewer returning players. You can see that this metric is highest for the Miami Heat and the L.A. Lakers

Finally we can agree with general manager Bob Myers' statement that Team Continuity plays a big role in Team Wins and it should not be overlooked by any General Manager when building the team or trading players to build a NBA championship winning Team.

Guiding Question 3: How does playing defensively vs offensively impact a teams overall season?

There is a common understanding in the NBA that defense has a bigger impact on the overall team success compared to offense. On multiple occasions commentators and NBA legends like Bill Russel have been quoted saying "Offense wins games, Defense wins Championships."

Due to the rise in three-point shot usage in the league and the development of high paced offense-focused teams, the reliability of the previous quote seems dubious. We want to see if there is still any weight in the late NBA legend's saying or has the three-point, pass-heavy and team-focused game style totally overpowered the modern NBA defense.

```
In [ ]: def_vs_off = pd.read_sql_query('''
SELECT
    Playstyle,
    (COUNT(*)/73)*100 AS Win_Percentage
FROM
    (
        SELECT
            *,
            case when ORtg >= 107.9 then "Offensive" when
DRtg <= 108.1 then "Defensive"
            ELSE      "Cannot Classify"
            END AS Playstyle
        FROM
            teamstats_mg
        WHERE
            playoffs = "Won Finals"
            AND      lg      = "NBA") a
GROUP BY
    1;''', engine)
print(def_vs_off)
```

	Playstyle	Win_Percentage
0	Defensive	52.05
1	Offensive	47.95

```
In [ ]: fig = px.bar(def_vs_off, x='Playstyle', y='Win_Percentage', range_y=[0,100], labels={
    "Playstyle ": "Playstyle ",
    "Win_Percentage": "Percentage of seasons won
    (%)"})
fig.update_traces(texttemplate="%{y}")
fig.update_layout(
    title_text="Distribution of playstyles among season winners",
    height=600
)
fig.show()
```

Within the scope of this analysis, we classified each season's winning team's playstyle into either "defensive" or "offensive" based on their defensive and offensive ratings. These ratings are based on the baskets scored in either a defensive or offensive context. Thus, lower the defensive ratings are, the better a team's defensive play is, whereas the higher the offensive ratings for the team are, the better their offensive play is.

Based on the above categorization, the plotted graph concludes that there are slightly more defensive teams winning the NBA Championships, but it is a really slight difference, suggesting the concept of "Defense wins Championships" is fairly unfounded.

Guiding Question 4: Do individual accomplishments accumulate towards winning the NBA season?

Continuing with the trend of team success and factors that affect that, we are looking into the individual accomplishments of players who have been awarded one or more of the 5 core NBA awards. These awards include Most Valuable Player (MVP), Defensive Player of the Year, Most Improved Player, Six Man of the Year, and Rookie of the Year.

This question will help us understand the importance of having award winning players on your team and whether these players can lead a team to a successful record or even championships. Intuitively, one might think having a top 5 player or players with many accolades on your team might lead you deep in the NBA playoffs.

This is certainly the case for Milwaukee Bucks star Giannis Antetoukoumpo, who has been to the all star game 6 times, has won three of 5 core awards - MVP(x2), MIP, DPOY, and was the star player of the Bucks when they won the championship in 2021.

Is this trend consistent throughout the NBA or was this an isolated incident?

```

In [ ]: individual_accomplishments = pd.read_sql_query(''
SELECT
    Awards_category,
    (COUNT(*)/73)*100 AS win_pct
FROM
    (
        SELECT
            *,
            case when No_of_Awards >= 1 then "High"
            ELSE    "Low"
            END AS Awards_category
        FROM
            (
                SELECT
                    a.*,
                    case when b.NO_of_Awards IS
NULL then 0
                    ELSE    b.NO_of_Awards
                    END AS No_of_Awards
                FROM
                    (
                        SELECT
                            season
                        ,
                            Replace(team,
"*", "") AS Cleaned_Team,
                            playoffs
                        ,
                            W
                        ,
                            SRS
                        ,
                            Pace
                        ,
                            `Rel.Pace`
                        ,
                            ORtg

```

```

                                `Rel.ORTg`
,
                                DRtg
,
                                `Rel.DRtg`
                                FROM
                                teamstats_mg a
                                WHERE
                                lg IN ( "nba" ))
a
                                LEFT JOIN
                                (
                                SELECT
                                season      ,
                                tm          ,
                                Team_Name,
                                Count(award) AS
NO_of_Awards
                                FROM
                                statsaward_mg
saw
                                LEFT JOIN
                                team_mapping map
                                ON
                                saw.Tm =
map.Abbreviation
                                WHERE
                                lg IN ( "nba" )
                                GROUP BY
                                1,
                                2) b
                                ON
                                a.Cleaned_Team=b.Team_Name
                                AND  a.Season      = b.Season) a
                                WHERE
                                playoffs = "Won Finals") a
GROUP BY
1''' ,engine)
print(individual_accomplishments)

```

	Awards_category	win_pct
0	High	36.99
1	Low	63.01

```
In [ ]: fig = px.bar(individual_accomplishments, x='Awards_category',
y='win_pct', range_y=[0,100], labels={
    "Awards_category": "No.of Awards (Category)",
    "win_pct": "Win Percentage (%)"})
fig.update_traces(texttemplate="%{y}")
fig.show()
```

As significant as individual accomplishments are, the game of basketball is not dictated by the award winners of the game. Being a team game, the winning percentage (based on season winners) is not higher for teams that have a high number of players receiving awards, but is rather a factor of the team's overall performance.

Based on the graph plotted, we can interpret that teams that have higher number of players receiving individual awards do not have a higher win percentage as compared to those teams that have lesser number of players with individual achievements.

Guiding Question 5: Does a players net statistics affect their in game performance? Does this performance translate to team success?

There is a consistent flow of good and bad teams and players each season. While it seems obvious that a player with better net statistics will contribute more to their team's success, this is not necessarily true.

One common trend is that good players in bad teams often "Stat Pad," that is, because of their team's poor overall performance, they are more readily able to score, pass, rebound, etc. thus inflating their individual statistics. Teams often get trapped into thinking a player on a bad team is a star based on these inflated numbers. The purpose of the following is to analyze whether their net statistics is a good determinant of the player's in game statistics.

```
In [ ]: points_vs_nrr = pd.read_sql_query('''
Select
    distinct a.Player,
    avg(a.PTS) as `Average Points`,
    avg(b.WS) as `Average Win Share`
from
    pergamebasics_mg a
    left join (
        Select
            distinct Player as player_b,
            season as season_b,
            WS
        from
            advancedstats
    ) b on a.Player = b.player_b
    and a.Season = b.season_b
group by
    1''', engine)

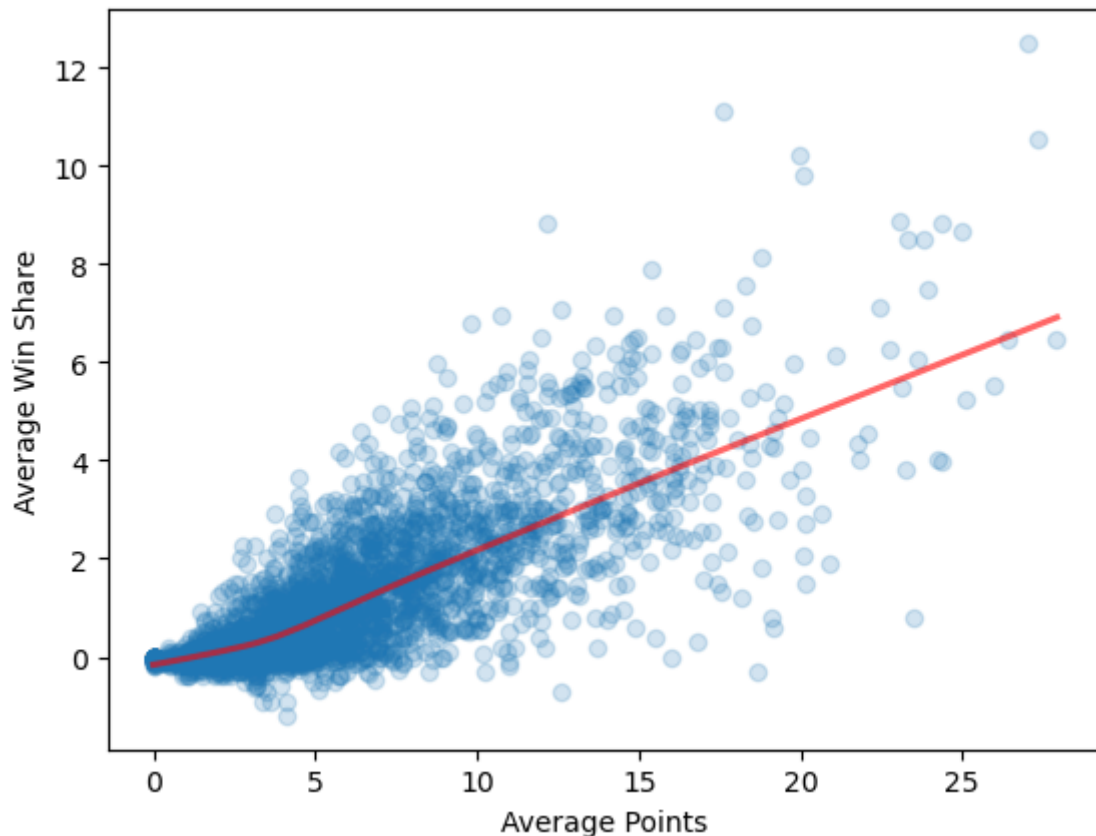
print(points_vs_nrr[["Average Points", "Average Win Share"]].corr())

import seaborn as sns
sns.regplot(data=points_vs_nrr, x='Average Points', y='Average Win
Share', lowess=True, scatter_kws={'alpha':0.2}, line_kws={'color':
'r', 'alpha':0.6})
print("\nAs we can observe from the data as well as the graph, there
```

is a positive correlation between points scored and win share of a player i.e. as the points increase, we also observe an increase in the win share")

	Average Points	Average Win Share
Average Points	1.000000	0.797059
Average Win Share	0.797059	1.000000

As we can observe from the data as well as the graph, there is a positive correlation between points scored and win share of a player i.e. as the points increase, we also observe an increase in the win share



From this figure, it seems that even though stat "padding" is a phenomenon in the NBA, individual performance does increase the chances of a team victory. As we can observe from the graph, there is a positive correlation between points and win share of a player. As the player's points increase, we also observe an increase in the win share, clearly indicating these players are contributing to their team's victories.

Guiding Question 6: How do injuries affect player contract prices?

Injuries and the notoriously harsh NBA schedule goes hand in hand. This has been the case since the sport first adopted a staggering 82 game season. Many players have been affected by the "injury bug" due to this high-paced schedule and have lost money and health over the years.

Players like Derrick Rose, Greg Oden, Brandon Roy and Gordon Hayward have become the posterchildren for injured players in the NBA. Coaches, players and fans have called for the NBA to shorten the season over the past 10 years but to no avail.

Here we are trying to analyze the average amount of money a player loses due to injuries and if the NBPA (National Basketball Player Association) has legitimate evidence to support their plea to shorten the season.

```
In [ ]: injuries = pd.read_sql_query('''
with injuries as (
Select
    distinct trim(Relinquished) as Player,
    count(date) as times_injured
from
    `injuries_2010-2020`
where
    Acquired is null
    and Relinquished is not null
    and Relinquished != ("76ers")
group by
    1
)
Select
distinct Injured_freq_category as Injured_Frequency,
avg(
    cast(
        replace(right(Guaranteed, length(Guaranteed) -1), ',', '')
    as UNSIGNED
    )
) as contract_value
from(
    Select
        *,
        case
            when times_injured >= 21 THEN "High"
            else "Low"
        END as Injured_freq_category
    From(
        Select
            distinct c.Player,
```

```

        c.Guaranteed,
        i.times_injured
    from
        contracts c
    left join injuries i on i.Player = c.Player
    where
        i.Player is not null
    ) a
) b
group by
1;''' , engine)
print(injuries)

```

	Injured_Frequency	contract_value
0	High	4.521015e+07
1	Low	3.418836e+07

```

In [ ]: fig = px.bar(injuries, x='Injured_Frequency', y='contract_value')
fig.show()

```

The data above represents the players split into two categories based on the frequency of injuries compared compared to the sum of the salaries. The current players who have been injured more than 20 times over the last 10 years were categorized as High frequency of injury, while all other players have been categorized as Low frequency. The average salary for the 2022-2023 season has been computed and compared for both the categories and the results can be seen above.

We conclude that players that have a high frequency of getting injured have ~25% of the contract value of low injury frequency players.

Fantasy Game Simulator

The culmination of the entire group's work from data collection to model building can be seen in this component of our project.

The game allows users to pick a team of ten from a selection of current players in the 2022-2023 with a predetermined salary cap. The teams built by the players can go then go head to head against one another in a simulated game with the click of a button.

The results of the simulated matchup are computed through models built using Multiple Linear Regression and random sampling, providing us with a breakdown of scores similar to those of an actual NBA game.

Access the [game simulator](#) here!

Overview

The diagrams below display the general layout of our web application

1. Server The server is the location where the user login credentials, the details to create a new user and database is hosted where querying is done on demand based on the specific player selected and requirement of the current player statistics.
2. Models The models used for the real time predictions of the game outcomes.
3. Web Application Client The web app client is built with **React** . The web app has components like player selection for a team roster, viewing the fantasy team statistics and game simulation.

Legend

The following diagrams use this general notation:

```

mermaid
graph LR
  subgraph containers
    process(with processes inside)
  end
  button(("button<br>(on front end)"))
  database[("tables<br>(databases)")]
  A--Function or method call-->B
  A.2-.Data transfer .->B.2

```

Server Overview

This is a condensed version of the primary processes happening on the server side.

```

mermaid
graph TD
  client["API calls from Client"]
  subgraph Virtual Server
    subgraph Server API
      server.findTeam("/findteam")
    end
  end

```

```

server.saveTeam("/saveteam")
server.signup("/signup")
server.findPlayer("/findPlayer")
server.login("/login")
end
subgraph Docker Container
  subgraph MariaDBserver
    subgraph NBA_APP Database
      nba_app.users[(App Users<br>--Email<br>--Password<br>--
team_id)]
      nba_app.players[(Current player stats<br>--General
Stats<br>--Contract Price)]
      nba_app.teams[(Teams<br>--roster<br>--'owner')]
    end
  end
end
end
end
client --> server.login
client -->server.signup
client--team_id-->server.findTeam

client-.team_id/ Roster.->server.saveTeam
server.findTeam -.team object.->client
server.login -->nba_app.users
server.signup -.new User data.->nba_app.users
server.findTeam-->nba_app.teams
server.saveTeam-.->nba_app.teams
server.findPlayer -->nba_app.players
nba_app.players-.player stats.->server.findPlayer
nba_app.users-.user info.->server.login
nba_app.teams-.team data.->server.findTeam

```

Models

The models make up the component of the web application responsible for simulating the basketball game outcomes. The data used in this model can be found under the header "Parsing per game statistics per player for each team".

By predicting the individual player scores of a particular roster (which consists of three-pointers scored, two-pointers scored, etc.), we are able to simulate semi-realistic game outcomes and determine a winner.

React application

The `React` application uses a component library called [Chakra UI](#) and is written in `Typescript`. The project's repo for both the server and client can be [browsed here](#) with additional documentation on both parts of the project found under the `docs` folder. The references for various parts of the project are also listed in the repo (there were too many tutorials and documentation pages consulted to record them all here).

The app is essentially rendered as a single page and handles API calls using the [mariaDB - javascript connector](#). The simulations occur on the clientside and only the results are sent to the server. This was done to by-pass any potential server load issues, but may change as we continue to build on this app.

Note: the repo is subject to continuous change as we build upon the app.

Discussion

Vardaan's Reflection

In my opinion, the use case of web scraping was really challenging and interesting for me. Working through multiple datasets and seeing real time results from scraping the web elements was very rewarding. Furthermore, to see the scrapped data being utilized by myself and my teams was really interesting due to the additional data slicing and type processing as well as understanding the needs of my teammates required data sources.

From a game simulation point of view, having the time and the understanding of intermediate machine learning concepts would have helped us refine the simulation code and provide us with more accurate results. Moreover the use case of linear regression caused minor issues due to the type and the magnitude of data that we are utilizing. Based on minor research on the topic, it was found that an algorithm like Random Forest would work exponentially better than a simple or multiple linear regression model.

After having additional understanding into some of the mistakes and learnings that were made in the duration of the project, building and refining the web app for simulation as well as building a real time ML algorithm that works accurately with our current data could be a scope of improvement.

-Vardaan Bhatia

Gavin's Reflection

We started this project as soon as we could, which meant the only database we had been exposed to up until that point was SQL. Naturally, we took it and ran, but a few downsides quickly became obvious. For starters, a team could have (theoretically) 15 or more players. To represent a team object in SQL that included other data as well such as a team's budget or coach or owner and the roster, meant that we effectively had to create 15 columns to represent up to 15 players on each team.

This was far from efficient.

Suppose we wanted to remove just one player. Then we would have a Null entry where that player used to be. Or we'd have to rewrite the entire entry (which is what the app does). Either way, it is clear that an object based NoSQL database such as mongo DB would have been much better served for this project, and it's something we are looking to adapt in future courses if we have the opportunity to build on this app.

Overall, this was an awesome opportunity to learn not only how to use each tool individually, but to integrate all the tools together in production.

- *Gavin*

Prem's Reflection

I learned a lot in this project which was a brand new hands-on experience for me. For example, I had never been introduced to web scraping techniques using beautiful soup in python to extract data from the webpage. But I learned how to do web scraping under the guidance of Vardaan, who had some experience in it.

Another core aspect I realized when analyzing the datasets is that it is extremely important to clean and filter your data, otherwise, your graph might not show the correct information you wanted. Moreover, after getting feedback from other teams, I learned that it is important to show only necessary information that is easily digestible for the audience. For example, my first graph that showed the win and continuity was very busy, with lots of teams being shown on 1 diagram. changing the graph to show only data from the past 10 seasons from teams grouped into east and west conferences was useful, making the data more easily digestible. This goes to show that when working on a project at a company, we should only show relevant information to the client, so they can easily grasp what analyses of the graph is presenting.

From my teammate Gavin, I learned how to design a web app workflow. Previously, I had never worked on a web app that connected to a database and retrieved relevant information according to the user's input. Though we had a short time frame to work on our web app, it

was harder to grasp some of the complicated aspects of the app such as the implementation of a statistical model when predicting the wins of two teams.

In context to the real-time game simulation model as a part of the project, we can also enhance the algorithm at the backend based on advanced modeling and machine learning algorithms. There is also scope to incorporate Fantasy League games as a next step for the simulator.

- Prem Patel

Mridul's Reflection

The project as a whole was very new of a concept for me as with no prior knowledge or NBA or basketball as a whole, performing the data wrangling process and then deriving meaning full insights was something that was challenging. Also, this had been my first iteration of cleaning, processing and summarizing i.e., basically doing everything from scratch to the final output in SQL which was very interesting and gave a deeper level of understanding and expertise.

To my limited knowledge, I would say the use of R (since I am more comfortable with R than python) might have made all this slightly easier and I would have preferred if given a free will to perform the analysis on. Also, for the scope of this project, the classification of teams/players into certain categories was more simple/restricted and could have been done using more complex algorithms and hence providing more accurate results

- Mridul

Conclusion

Sports analytics has been on the rise and as seen from the small exploration into the domain by our team. From parsing web scraping data to building a website application of real time model usage with a react web application. The team has had an opportunity to learn alot about the functioning of fundamental python, sql and application building experience. The field of analytis is growing daily and when sports analytics is concerned, many real life applications of the usecases have already been realized. From gambling websites, fantasy leagues, player improvment and team building are just scratching the surface of the possibilities of the data science in this field.

GitHub

All of our Datasets, WebApp, and Documentation for our project could be found in below GitHub Repository:

Link : <https://github.com/vdbee1/Sports-Analytics-NBA>

References

1. Association for Professional Basketball Research [WWW Document], n.d. URL <https://www.apbr.org/> (accessed 11.2.22).
2. Basketball Statistics & History of Every Team & NBA and WNBA Players [WWW Document], n.d. . Basketball-Reference.com. URL <https://www.basketball-reference.com> (accessed 11.5.22).
3. DSG, 2021. The Rise Of Sports Analytics [WWW Document]. URL <https://datasportsgroup.com/news-article/74282/the-rise-of-sports-analytics/> (accessed 11.2.22).
4. Lewis, M. (2004) Moneyball: The Art of Winning an Unfair Game. 1st edition. New York, NY: W. W. Norton & Company.
5. Official NBA Stats | Stats | NBA.com [WWW Document], n.d. URL <https://www.nba.com/stats> (accessed 11.5.22).
6. The Role of Data Science in Sports, 2020. . CORP-MIDS1 (MDS). URL <https://www.mastersindatascience.org/resources/big-data-in-sports/> (accessed 11.5.22).
7. Statistical Player Value, SPV - NBAstuffer (2022). Available at: <https://www.nbastuffer.com/analytics101/statistical-player-value-spv/> (accessed: 7 November 2022).
8. www.Researchgate.Net [Online]. Available at: https://www.researchgate.net/publication/332406802_A_systematic_review_of_sports_analytics (Accessed: 7 November 2022).
9. D'souza, G. (2020). Web Scraping — Python (Requests and BeautifulSoup). [online] The Startup. Available at: <https://medium.com/swlh/web-scraping-python-requests-and-beautifulsoup-45d5f48f5a1> [Accessed 30 Nov. 2022].
10. Stack Overflow. (n.d.). python - scrape through website with href references. [online] Available at: <https://stackoverflow.com/questions/19429126/scrape-through-website-with-href-references#:~:text=If%20you%20need%20to%20scrape%20data%20out%20a> [Accessed 30 Nov. 2022].
11. sports.sites.yale.edu. (n.d.). NBA Model Math | Yale Undergraduate Sports Analytics Group. [online] Available at: <https://sports.sites.yale.edu/nba-model-math#:~:text=Model%201%3A%20NBA%20Power%20Rankings%20%28Linear%20Regression> [Accessed 30 Nov. 2022].
12. Annieshie (2020). NBA Player Salaries Prediction with Linear Regression. [online] Medium. Available at: <https://medium.com/analytics-vidhya/nba-player-salaries-prediction->

[with-linear-regression-2b90280ff4e8.](#)

