NBA Technology Data Analysis Group 3:

Kapil Tare, Shreyas Kashyap, Jovita Andrews, Kunal Jain

IST 652: Scripting for Data Analysis

Project Goal

- Our overall goal would be to provide team GMs with information to decide which players would be key to the success of their team based on the following -
  - Predicting Points per game is a factor in offensive success.
    - Using the historical data on players' ability to score a point such as no. of the 3-pointers scored, 2-pointers scored, passes made throughout the game, time spent on the court, and impact made in the previous games along with various other factors we will be predicting the points a player would potentially score in a game per season thereby providing team GMs ideas on how to allocate players towards a championship-winning team.
  - Understanding Players' potential with the help of Impact Score.
    - The impact score (+/-) does not only consider a player's ability to score but also considers their selfless impact on and off the basketball court. Through this, we will make calculated decisions to produce championship-winning lineups.

Project Approach

The team will be utilizing Python's built in model for NBA data.

There are static data sets and endpoints in the nba_api.

Data Cleaning and Integration

Exploratory Data Analysis

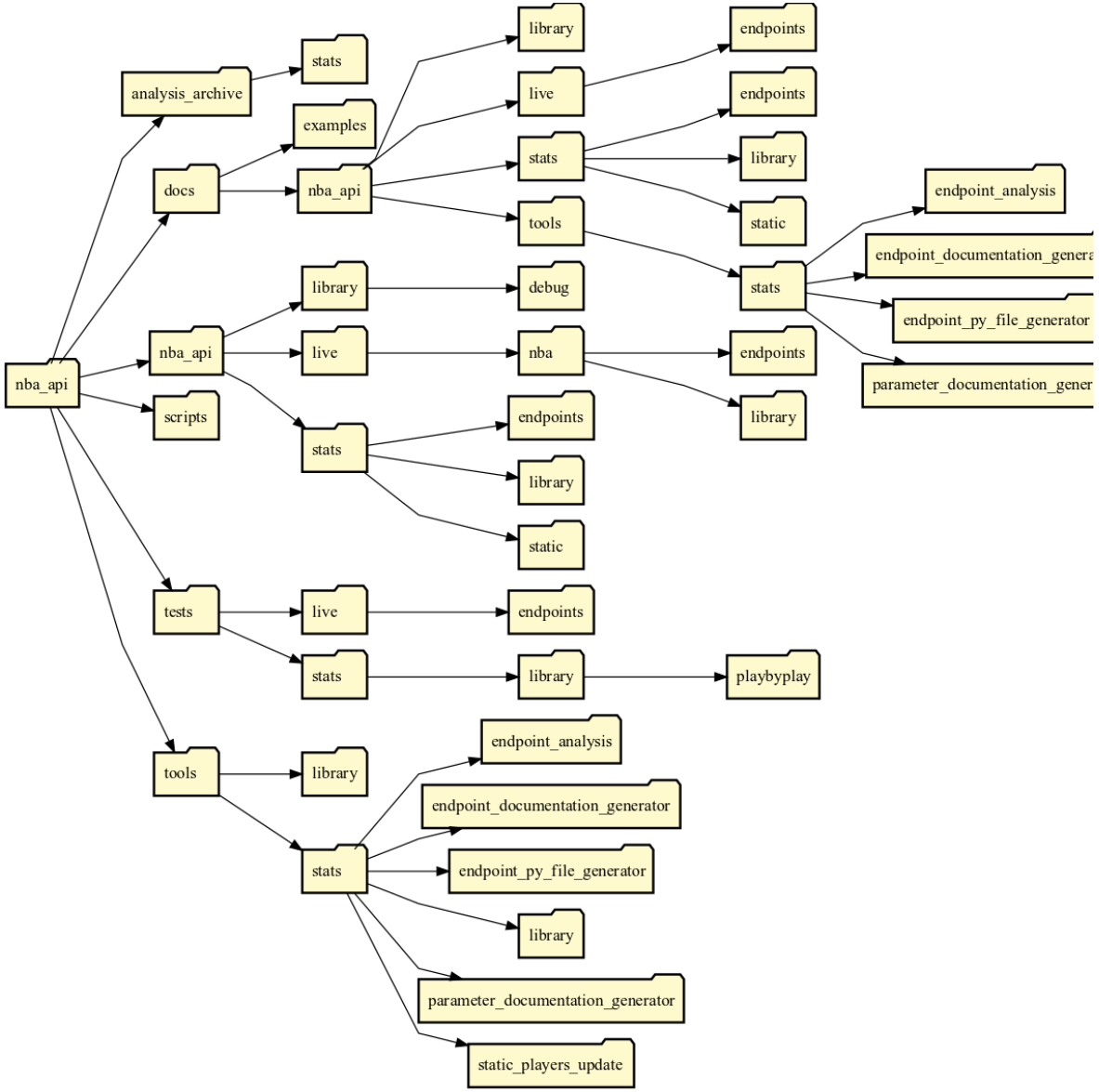
Feature Engineering

ML Modeling

Evaluating model results

Data Collection and Integration

- *The library is the whole package that you use to access the data.*
- *live refers to real-time data.*
- *stats is the historical and current statistical data.*
- *tools are the helper functions utilities that support data retrieval and manipulation.*
- *endpoints are the specific URLs defined by NBA API that you call to get certain pieces of information.*



Players and Teams Information

```
#dataframe that contains the teams information
teams_df = pd.DataFrame(teams.get_teams())
teams_df.head()
```

	id	full_name	abbreviation	nickname	city	state	year_founded
0	1610612737	Atlanta Hawks	ATL	Hawks	Atlanta	Georgia	1949
1	1610612738	Boston Celtics	BOS	Celtics	Boston	Massachusetts	1946
2	1610612739	Cleveland Cavaliers	CLE	Cavaliers	Cleveland	Ohio	1970
3	1610612740	New Orleans Pelicans	NOP	Pelicans	New Orleans	Louisiana	2002
4	1610612741	Chicago Bulls	CHI	Bulls	Chicago	Illinois	1966

Teams_df shape: 30, 7

```
#dataframe that contains the players information
players_df = pd.DataFrame(players.get_players())
players_df.head()
```

	id	full_name	first_name	last_name	is_active
0	76001	Alaa Abdelnaby	Alaa	Abdelnaby	False
1	76002	Zaid Abdul-Aziz	Zaid	Abdul-Aziz	False
2	76003	Kareem Abdul-Jabbar	Kareem	Abdul-Jabbar	False
3	51	Mahmoud Abdul-Rauf	Mahmoud	Abdul-Rauf	False
4	1505	Tariq Abdul-Wahad	Tariq	Abdul-Wahad	False

Players_df shape: 4900, 5

Stats information

```
#dataframe that contains player stats
player_stats = playercareerstats.PlayerCareerStats(player_id='2544')
# For now we have extracted the data for LeBron James's performance stats but, going ahead we'll be creating a modular function for pulling the sample data
# for random 15-20 players to avoid biases while performing modeling activities
stats_df = player_stats.get_data_frames()[0]
stats_df.head()
```

	PLAYER_ID	SEASON_ID	LEAGUE_ID	TEAM_ID	TEAM_ABBREVIATION	PLAYER_AGE	GP	GS	MIN	FGM	...	FT_PCT	OREB	DREB	REB	AST	STL	BLK	TOV	PF	PTS
0	2544	2003-04	00	1610612739	CLE	19.0	79	79	3120.0	622	...	0.754	99	333	432	465	130	58	273	149	1654
1	2544	2004-05	00	1610612739	CLE	20.0	80	80	3388.0	795	...	0.750	111	477	588	577	177	52	262	146	2175
2	2544	2005-06	00	1610612739	CLE	21.0	79	79	3361.0	875	...	0.738	75	481	556	521	123	66	260	181	2478
3	2544	2006-07	00	1610612739	CLE	22.0	78	78	3190.0	772	...	0.698	83	443	526	470	125	55	250	171	2132
4	2544	2007-08	00	1610612739	CLE	23.0	75	74	3027.0	794	...	0.712	133	459	592	539	138	81	255	165	2250

5 rows × 27 columns

- Stats dataframe: Index(['PLAYER_ID', 'SEASON_ID', 'LEAGUE_ID', 'TEAM_ID', 'TEAM_ABBREVIATION', 'PLAYER_AGE', 'GP', 'GS', 'MIN', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA', 'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS'], dtype='object')

Games information

```
#dataframe that contains game stats per team
games = leaguegamefinder.LeagueGameFinder()
games_df = games.get_data_frames()[0]
games_df.head()
```

	SEASON_ID	TEAM_ID	TEAM_ABBREVIATION	TEAM_NAME	GAME_ID	GAME_DATE	MATCHUP	WL	MIN	PTS	...	FT_PCT	OREB	DREB	REB	AST	STL	BLK	TOV	PF	PLUS_MINUS
0	22023	1610612760	OKC	Oklahoma City Thunder	0022300234	2023-11-22	OKC vs. CHI	W	241	116	...	0.816	11	34	45	24	4	12	12	17	14.0
1	22023	1610612738	BOS	Boston Celtics	0022300228	2023-11-22	BOS vs. MIL	W	240	119	...	0.824	4	39	43	27	3	5	15	15	3.0
2	52023	1612709925	OSC	Osceola Magic	2052300081	2023-11-22	OSC @ RGV	W	240	143	...	0.792	10	30	40	31	21	2	21	21	16.0
3	22023	1610612751	BKN	Brooklyn Nets	0022300227	2023-11-22	BKN @ ATL	L	265	145	...	0.833	22	37	59	28	8	6	16	24	-2.0
4	52023	1612709910	IMA	Indiana Mad Ants	2052300076	2023-11-22	IMA vs. MCC	W	240	117	...	0.667	22	46	68	26	8	3	17	21	5.8

5 rows x 28 columns

- Games columns: Index(['SEASON_ID', 'TEAM_ID', 'TEAM_ABBREVIATION', 'TEAM_NAME', 'GAME_ID', 'GAME_DATE', 'MATCHUP', 'WL', 'MIN', 'PTS', 'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA', 'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PLUS_MINUS'], dtype='object')
- 30000, 28

Standings information

```
#dataframe that contains players standings
standings = leaguestandings.LeagueStandings()
standings_df = standings.get_data_frames()[0]
standings_df.head()
```

	LeagueID	SeasonID	TeamID	TeamCity	TeamName	Conference	ConferenceRecord	PlayoffRank	ClinchIndicator	Division	...	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	PreAS	PostAS
0	00	22023	1610612738	Boston	Celtics	East	11-2	1		Atlantic	...	None	None	None	None	None	3-0	9-3	None	12-3	None
1	00	22023	1610612750	Minnesota	Timberwolves	West	7-1	1		Northwest	...	None	None	None	None	None	1-2	10-1	None	11-3	None
2	00	22023	1610612749	Milwaukee	Bucks	East	9-5	2		Central	...	None	None	None	None	None	2-1	8-4	None	10-5	None
3	00	22023	1610612760	Oklahoma City	Thunder	West	5-4	2		Northwest	...	None	None	None	None	None	3-1	8-3	None	11-4	None
4	00	22023	1610612743	Denver	Nuggets	West	8-3	3		Northwest	...	None	None	None	None	None	4-0	6-5	None	10-5	None

5 rows × 81 columns

- Standings columns: Index(['LeagueID', 'SeasonID', 'TeamID', 'TeamCity', 'TeamName', 'Conference', 'ConferenceRecord', 'PlayoffRank', 'ClinchIndicator', 'Division', 'DivisionRecord', 'DivisionRank', 'WINS', 'LOSSES', 'WinPCT', 'LeagueRank', 'Record', 'HOME', 'ROAD', 'L10', 'Last10Home', 'Last10Road', 'OT', 'ThreePTSOrLess', 'TenPTSOrMore', 'LongHomeStreak', 'strLongHomeStreak', 'LongRoadStreak', 'strLongRoadStreak', 'LongWinStreak', 'LongLossStreak', 'CurrentHomeStreak', 'strCurrentHomeStreak', 'CurrentRoadStreak', 'strCurrentRoadStreak', 'CurrentStreak', 'strCurrentStreak', 'ConferenceGamesBack', 'DivisionGamesBack', 'ClinchedConferenceTitle', 'ClinchedDivisionTitle', 'ClinchedPlayoffBirth', 'EliminatedConference', 'EliminatedDivision', 'AheadAtHalf', 'BehindAtHalf', 'TiedAtHalf', 'AheadAtThird', 'BehindAtThird', 'TiedAtThird', 'Score100PTS', 'OppScore100PTS', 'OppOver500', 'LeadInFGPCT', 'LeadInReb', 'FewerTurnovers', 'PointsPG', 'OppPointsPG', 'DiffPointsPG', 'vsEast', 'vsAtlantic', 'vsCentral', 'vsSoutheast', 'vsWest', 'vsNorthwest', 'vsPacific', 'vsSouthwest', 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec', 'PreAS', 'PostAS'], dtype='object')

Null Values

Null Values Present in the Game Dataframe:

- Win Loss : 15.773333
- Three Point Field Goal Percentage : 0.176667
- Free Throw Percentage : 18.54

These null values symbolizes that if the player is injured; the team may be performing better in the season or worse.

Columns with Null Values were dropped since mean imputation would create a bias in the games data set.

```
Null Value Percentages in Players DataFrame:
 PlayerID     0.0
FullName      0.0
FirstName     0.0
LastName      0.0
IsActive      0.0
dtype: float64

Null Value Percentages in Teams DataFrame:
 TeamID          0.0
TeamFullName     0.0
Abbreviation     0.0
Nickname         0.0
City             0.0
State            0.0
YearFounded      0.0
dtype: float64

Null Value Percentages in Games DataFrame:
 SeasonID                          0.000000
TeamID                             0.000000
TeamAbbreviation                   0.000000
TeamName                           0.000000
GameID                             0.000000
GameDate                           0.000000
Matchup                            0.000000
WinLoss                           15.773333
MinutesPlayed                      0.000000
Points                             0.000000
FieldGoalsMade                     0.000000
FieldGoalsAttempted                0.000000
FieldGoalPercentage                0.070000
ThreePointFieldGoalsMade           0.000000
ThreePointFieldGoalsAttempted      0.000000
ThreePointFieldGoalPercentage      0.176667
FreeThrowsMade                     0.000000
FreeThrowsAttempted                0.000000
FreeThrowPercentage               18.540000
OffensiveRebounds                  0.000000
DefensiveRebounds                  0.000000
TotalRebounds                      0.000000
Assists                            0.000000
Steals                             0.000000
Blocks                             0.000000
Turnovers                          0.000000
PersonalFouls                      0.000000
PlusMinusScore                     0.000000
dtype: float64

Null Value Percentages in Stats DataFrame:
 PlayerID                          0.0
SeasonID                           0.0
LeagueID                           0.0
TeamID                             0.0
TeamAbbreviation                   0.0
PlayerAge                          0.0
GamesPlayed                        0.0
GamesStarted                       0.0
```

Data Collection

There are a total of 135 endpoints in the NBA API module. But, For our Project, below are the data endpoints that we utilized to create a master dataset –

- Playercareerstats

  Provides comprehensive career statistics for a specific player. These statistics cover various aspects of a player's performance throughout their entire NBA career.

- Playergamelog

  Provides a detailed log of a player's performance in individual games. It offers a comprehensive record of statistics and metrics for each game a player has participated in during a specified season. Also provides granular insights into a player's performance on a game-by-game basis thereby helping understand a player's contributions, strengths, and areas for improvement throughout a season.

- Leaguedashptstats

  Provides detailed player performance metrics related to player movement, distance traveled, and other tracking-related statistics which essentially offers us comprehensive player tracking statistics for the NBA. It also provides detailed insights into player movement and performance metrics gathered from various tracking technologies used during games.

- Leaguedashplayerstats

  Provides detailed player statistics for a given season. This endpoint allows users to retrieve a wide range of player performance metrics, covering various aspects of their gameplay during a specified season thereby offering a detailed breakdown of a player's performance in various categories.

- Teamgamelog

  Provides detailed game-by-game statistics for a specific team over the course of a season. This endpoint is particularly useful for gaining insights into a team's performance, analyzing trends, and extracting various statistical metrics for each game.

- Commonplayerinfo

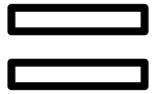
  provides comprehensive information about a specific NBA player, offering details that include personal information, draft history, player statistics, and current team affiliations

Data Integration

Player Career stats data

Team data

Common Player information



NBA Master data table

Game log data

Team data

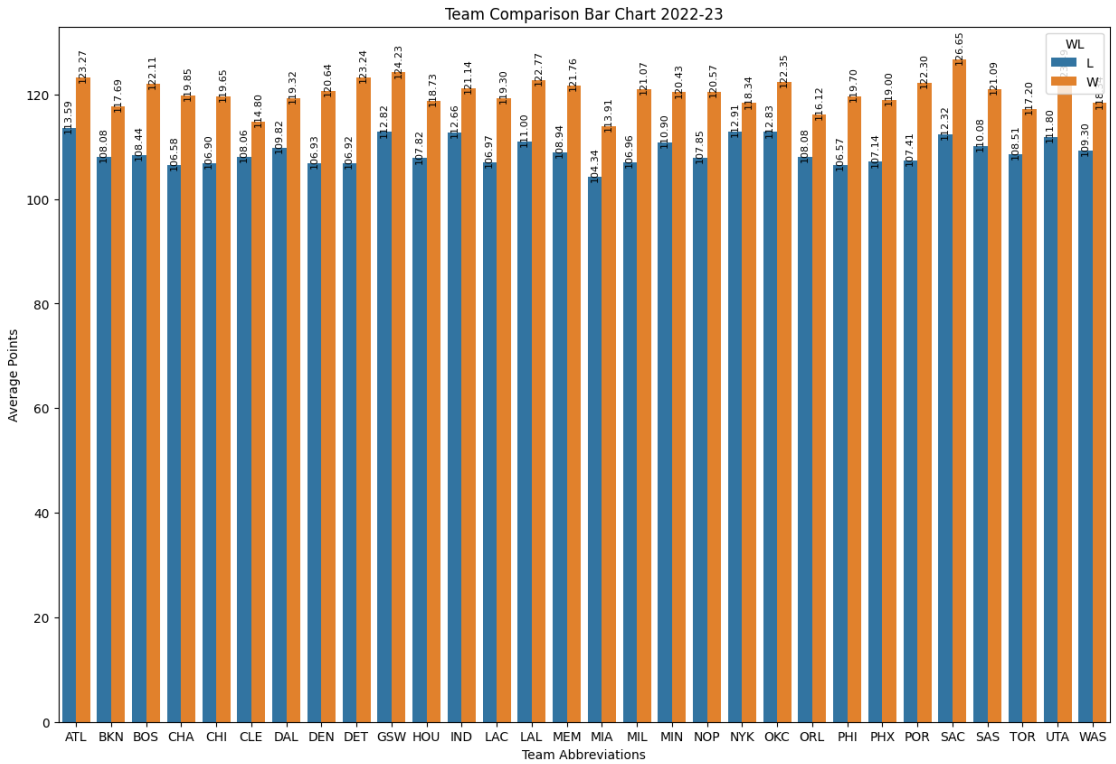
Exploratory Data Analysis

- To further understand the data available in the module and to create a plan towards executing our objective, there were 3 different avenues that were targeted during this step:
  - **Team Comparison Analysis with respect to average points scored throughout the 2022-23 season.**
    - The endpoints utilized here were teamgamelog and teams
    - This was achieved by merging dataframes from teamgamelog and teams on column 'team_ID'
  - **Average Plus/ Minus Score Comparison of NBA championship finalists for 2022-23 season.**
    - The endpoints utilized here were playergamelog, commonallplayers and players.
    - This gave us an understanding on how difference in plus/minus scores affects a team's overall success.
  - **Points Trend Over Time for Stephen Curry and LeBron James from 2016-2023**
    - The endpoints utilizedhere were playergamelog, and players
    - Merging of dataframes from players and playergamelogs for each respective players achieved by common column 'Player_ID'.

Team Comparison Analysis 2022-23

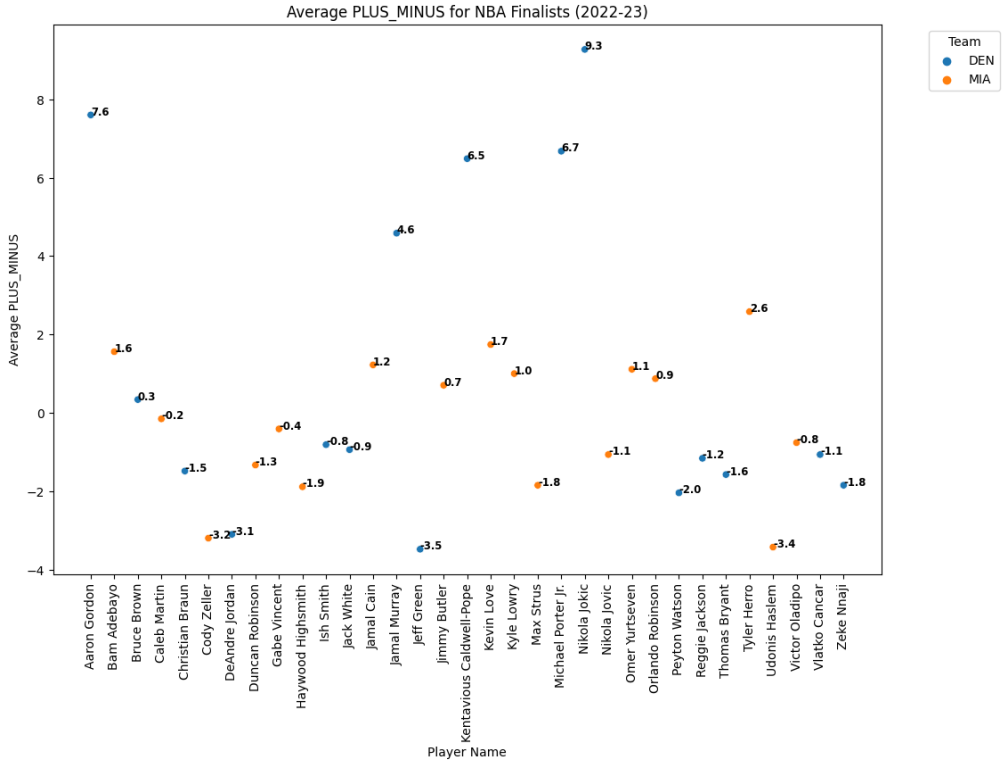
- The chart shows how teams perform with respect to average points scored on a win versus average points scored on a loss.
- This was to understand the importance of offensive success in a team's ability to win.

```python
## Team Comparison Bar Chart

from nba_api.stats.endpoints import teamgamelog
from nba_api.stats.static import teams
import matplotlib.pyplot as plt


teams_df = pd.DataFrame(teams.get_teams())
teams_list = teams_df['id'].astype(str)
seasons = ['2015-16', '2016-17', '2018-19', '2019-20', '2020-21']

# Create empty dataframe empty_tgl_df
tgl_df = pd.DataFrame()
# Get team game log data for current season.
for team in teams_list:
      team_game_log = teamgamelog.TeamGameLog(team_id=team, season='2022-23').get_data_frames()[0]
    # Append the team game log to the DataFrame
      tgl_df = tgl_df.append(team_game_log, ignore_index=True)

# merge tgl_df dataframe with teams_df
complete_tgl_data = teams_df.merge(tgl_df, left_on = 'id', right_on = 'Team_ID')

# Replace 'stats' with the actual statistics you want to include in the chart
stats = ['Team_ID', 'PTS']
wl_record = tgl_df['WL'].value_counts()

# Create a bar chart
fig, ax = plt.subplots(figsize=(15, 10))
barplot=sns.barplot(x='abbreviation', y= 'PTS', hue='WL',
            data = complete_tgl_data.groupby(['abbreviation','WL'])['PTS'].mean().reset_index())

```



Average Plus/Minus Score Comparison

- The chart on the left shows the average +/- scores of each player that made it to the NBA finals.
- DEN was the winning team in 2022-23 season. A majority of the player impact score is higher than that of players from MIA.
- This was to understand how player impact score could be a factor for a championship contending team.

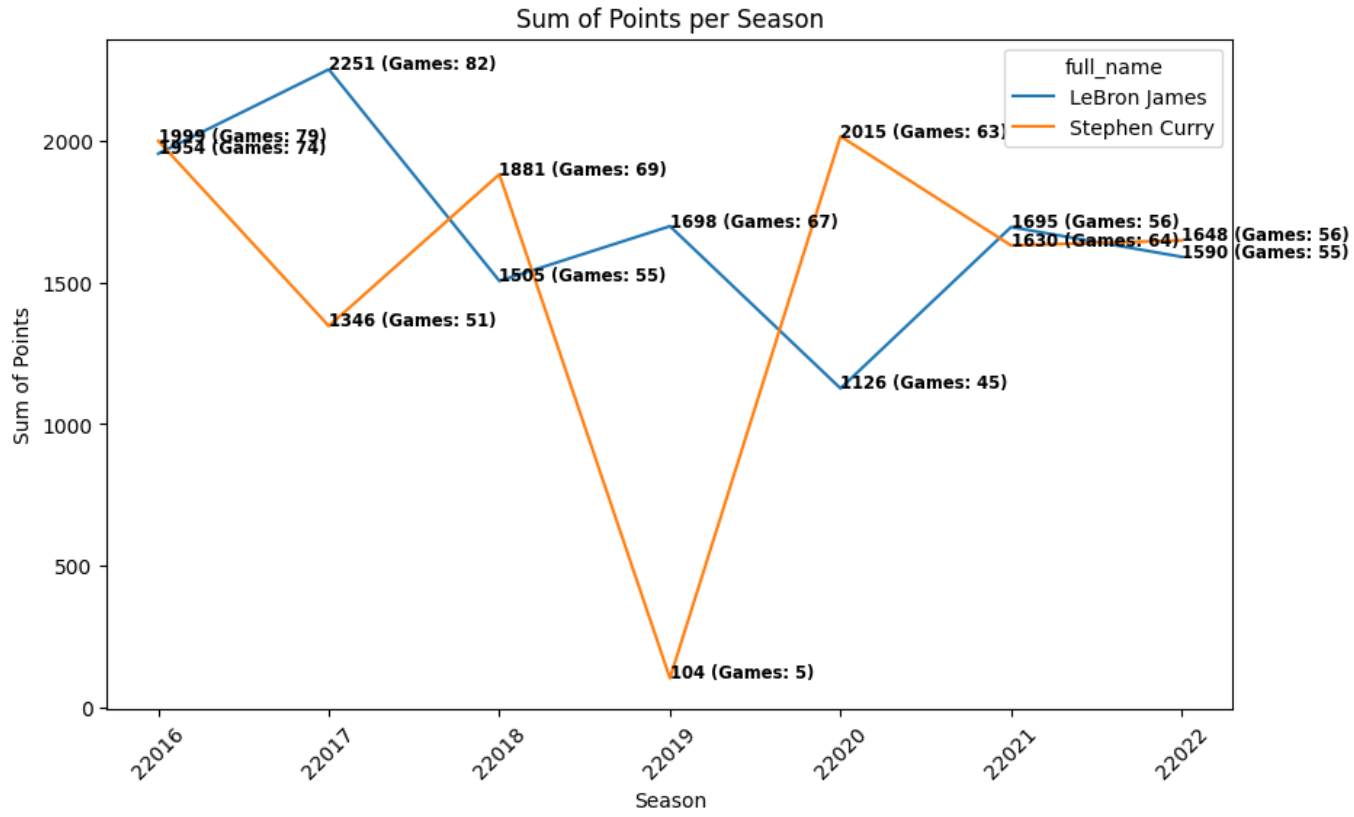



```python
player_info = pd.DataFrame(players.get_active_players())
player_id = player_info['id']

teams_df = pd.DataFrame(teams.get_teams())

# Getting team_IDs for teams that competed in the finals from 2022-23.
finals_team_list = teams_df.loc[teams_df['abbreviation'].isin(['MIA','DEN'])]['id'].astype(str)
# Specify the season
season = '2022-23'

# Get nba finalists for 2022-23 season
all_players = commonallplayers.CommonAllPlayers(season=season)
all_players_df = all_players.get_data_frames()[0]

# Filter players for the specified teams
nba_finalists_22023 = all_players_df[all_players_df['TEAM_ABBREVIATION'].isin(['MIA','DEN'])]


nba_finalists_22023.columns

player_efficiency_df = pd.DataFrame()

for id in nba_finalists_22023['PERSON_ID']:
  # Call the Player Game log function for each player ID
    game_log = playergamelog.PlayerGameLog(player_id=id, season = season)
    df_game_log = game_log.get_data_frames()[0]
    player_efficiency_df = player_efficiency_df.append(df_game_log, ignore_index=True)

complete_efficiency_df = nba_finalists_22023.merge(player_efficiency_df, left_on = 'PERSON_ID', right_on = 'Player_ID')

# Calculate average PLUS_MINUS for each player
avg_plus_minus = complete_efficiency_df.groupby('DISPLAY_FIRST_LAST')['PLUS_MINUS'].mean().reset_index()

# Merge with original data to get 'TEAM_ABBREVIATION'
avg_plus_minus = avg_plus_minus.merge(complete_efficiency_df[['DISPLAY_FIRST_LAST', 'TEAM_ABBREVIATION']].drop_duplicates(), on='DISPLAY_FIRST_LAST')
```

Points Trend Over Time

- With regards to points scored by players over time for each season, the team realized there were certain factors that needed to be considered for credible results.
- Total number of games played by each player would be necessary to better understand anomalies as seen in the chart.
- As seen for season 22020, there is an anomaly for Stephen Curry's total points scored in a season and that was factored by the number of games he played and not his offensive skills.
- Hence, deciding that Points Per Game per season would be a better feature for our prediction model than total points per season.

```python
# Finding Stephen Curry's Player_ID
sc_player_id = players_df.loc[players_df['DISPLAY_FIRST_LAST']=='Stephen Curry']['PERSON_ID'] #201939

lbj_player_id = ['2544']

# Define the list of seasons you're interested in
seasons = ['2016-17','2017-18','2018-19','2019-20','2020-21', '2021-22', '2022-23']

# Initialize an empty DataFrame to store the data
lbj_game_logs = pd.DataFrame()

# Loop through each season, fetch game log data, and append to the DataFrame
for season in seasons:
      game_log = playergamelog.PlayerGameLog(player_id=lbj_player_id, season=season)
      game_log_df = game_log.get_data_frames()[0]
      lbj_game_logs = lbj_game_logs.append(game_log_df, ignore_index=True)

sc_game_logs = pd.DataFrame()

# Loop through each season, fetch game log data, and append to the DataFrame
for season in seasons:
      game_log = playergamelog.PlayerGameLog(player_id=sc_player_id, season=season)
      game_log_df = game_log.get_data_frames()[0]
      sc_game_logs = sc_game_logs.append(game_log_df, ignore_index=True)

# Group by season and calculate the sum of points per season
active_players=pd.DataFrame(players.get_active_players())

sc_bio = active_players.loc[active_players['first_name'].isin(['Stephen'])]
sc_game_data = sc_game_logs.merge(sc_bio, left_on = 'Player_ID', right_on = 'id')
sc_games_played = sc_game_data.groupby(['SEASON_ID', 'full_name', 'Player_ID'])['Game_ID'].size().reset_index()

lbj_bio = active_players.loc[active_players['first_name'].isin(['LeBron'])]
lbj_game_data = lbj_game_logs.merge(lbj_bio, left_on = 'Player_ID', right_on = 'id')
lbj_games_played = lbj_game_data.groupby(['SEASON_ID', 'full_name','Player_ID'])['Game_ID'].size().reset_index()

# Calculate number of games played by each player for each season.
games_played = pd.concat([lbj_games_played,sc_games_played], ignore_index=True)
games_played

sc_lbj_game_logs = pd.concat([lbj_game_logs,sc_game_logs], ignore_index=True)
sc_lbj_bio = active_players.loc[active_players['first_name'].isin(['LeBron', 'Stephen'])]
sc_lbj_game_logs = sc_lbj_game_logs.merge(sc_lbj_bio, left_on = 'Player_ID', right_on = 'id')

sum_pts_per_season = sc_lbj_game_logs.groupby(['SEASON_ID', 'Player_ID'])['PTS'].sum().reset_index()
games_played_sorted = games_played.sort_values(by='SEASON_ID', ascending=True)
sum_pts_per_season = sum_pts_per_season.merge(games_played_sorted)
```



Model Building Process

Our target variable is PTS which is basically points scored per game by each player which is continuous in nature. Hence our go-to algorithm for this project is a simple Linear Regression model.

NBA Master data table

Performing data cleaning activities

Creating new features from the existing features

Creating a Train and test dataset basis the seasons. (*in our case we considered the seasons from 2015-16 to 2021-22 for training and 2022-23 for testing)*

Fitting the training dataset on our linear regression model for training & then subsequently testing the model on our test dataset

Evaluating our model performance using metrics like R-square and mean square error

Model Results

For Evaluating our model, we have utilized two metrics -

1. **R-square**

   This helps us understand how well a regression model predicts or explains the variability of the dependent variable. In simple terms, R-squared tells us the proportion of the variance in the dependent variable that is predictable from the independent variables. Basically, it indicates that more of the changes in the dependent variable can be attributed to the changes in the independent variables

2. **Mean Square Error**

   This is a measure that is used to evaluate how well a linear regression model predicts the values of a dependent variable. Basically, lower Mean Square Error indicates that the model's predictions are closer to the actual values.

Below mentioned are our model results which we obtained -

```
# Step 11: Evaluate Model Performance
mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
r2_test = r2_score(y_test, predictions)


print(f'Mean Squared Error: {mse}')
print("R^2 Score (Test Set):", r2_test)
```

```
Mean Squared Error: 1.1611467926887047
R^2 Score (Test Set): 0.9826991288745461
```

Appendix

PlayerGameLog	
Column Name	**Column Description**
SEASON_ID	The season in which the game occurred.
Player_ID	The unique identifier for the player.
Game_ID	The unique identifier for the game.
GAME_DATE	The date of the game.
MATCHUP	The matchup information, including opponent and game location.
WL	Win or loss for the player's team in the specific game.
MIN	Minutes played in the game.
FGM	Field goals made.
FGA	Field goals attempted.
FG_PCT	Field goal percentage.
FG3M	Three-point field goals made.
FG3A	Three-point field goals attempted.
FG3_PCT	Three-point field goal percentage.
FTM	Free throws made.
FTA	Free throws attempted.
FT_PCT	Free throw percentage.
OREB	Offensive rebounds.
DREB	Defensive rebounds.
REB	Total rebounds.
AST	Assists.
STL	Steals.
BLK	Blocks.
TO	Turnovers.
PF	Personal fouls.
PTS	Points scored.
PLUS_MINUS	Players impact on the court.

PlayerCareerStats	
Column Name	**Column Description**
PLAYER_ID	Unique identifier for the player.
SEASON_ID	Season identifier.
LEAGUE_ID	League identifier.
TEAM_ID	Team identifier.
TEAM_ABBREVIATION	Abbreviation for the team.
PLAYER_AGE	Age of the player.
GP	Games played.
GS	Games started.
MIN	Minutes per game.
FGM	Field goals made per game.
FGA	Field goals attempted per game.
FG_PCT	Field goal percentage.
FG3M	Three-point field goals made per game.
FG3A	Three-point field goals attempted per game.
FG3_PCT	Three-point field goal percentage.
FTM	Free throws made per game.
FTA	Free throws attempted per game.
FT_PCT	Free throw percentage.
OREB	Offensive rebounds per game.
DREB	Defensive rebounds per game.
REB	Total rebounds per game.
AST	Assists per game.
TOV	Turnovers per game.
STL	Steals per game.
BLK	Blocks per game.
PF	Personal fouls per game.
PTS	Points per game.

TeamGameLog	
Column Name	**Column Description**
Team_ID	Unique identifier for the team.
Game_ID	Unique identifier for the game.
GAME_DATE	Date of the game.
MATCHUP	Matchup information, including opponent and game location.
WL	Win or loss for the team in the specific game.
MIN	Minutes played in the game.
PTS	Points scored by the team.
FGM	Field goals made by the team.
FGA	Field goals attempted by the team.
FG_PCT	Field goal percentage for the team.
FG3M	Three-point field goals made by the team.
FG3A	Three-point field goals attempted by the team.
FG3_PCT	Three-point field goal percentage for the team.
FTM	Free throws made by the team.
FTA	Free throws attempted by the team.
FT_PCT	Free throw percentage for the team.
OREB	Offensive rebounds by the team.
DREB	Defensive rebounds by the team.
REB	Total rebounds by the team.
AST	Assists by the team.
STL	Steals by the team.
BLK	Blocks by the team.
TO	Turnovers by the team.
PF	Personal fouls by the team.

CommonPlayerInformation	
Column Name	**Column Description**
PERSON_ID	Unique identifier for the player.
FIRST_NAME	First name of the player.
LAST_NAME	Last name of the player.
DISPLAY_FIRST_LAST	Display name in the format "First Last."
DISPLAY_LAST_COMMA_FIRST	Display name in the format "Last, First."
DISPLAY_FI_LAST	Display name in the format "F. Last."
BIRTHDATE	Date of birth of the player.
SCHOOL	School attended by the player.
COUNTRY	Country of birth of the player.
HEIGHT	Height of the player in feet and inches.
WEIGHT	Weight of the player in pounds.
POSITION	Playing position of the player.
DRAFT_YEAR	Year in which the player was drafted.
DRAFT_ROUND	Draft round in which the player was selected.
DRAFT_NUMBER	Draft pick number for the player.
ROSTERSTATUS	Current roster status of the player.
TEAM_ID	Team identifier for the player's current team.
TEAM_NAME	Current team name of the player.
TEAM_ABBREVIATION	Current team abbreviation.
TEAM_CODE	Team code for the player's current team.

CommonAllPlayers	
Column Name	**Column Description**
PERSON_ID	Unique identifier for the player.
DISPLAY_LAST_COMMA_FIRST	Display name in the format "Last, First."
DISPLAY_FIRST_LAST	Display name in the format "First Last."
ROSTER_STATUS	Current roster status of the player.
FROM_YEAR	Represents first year of player in NBA
TO_YEAR	Represents last or current year of player in NBA
TEAM_ID	Team identifier for the player's current team.
TEAM_CITY	Location of current team
TEAM_NAME	Current team name of the player.
TEAM_ABBREVIATION	Current team abbreviation.
TEAM_CODE	Team code for the player's current team.
GAMES_PLAYED_FLAG	Number of games played

Players	
Column Name	**Column Description**
PERSON_ID	Unique identifier for the player.
full_name	Player's full name
first_name	Player's first name
last_name	Player's last name
is_active	Boolean values on player's active status

Resources

- https://github.com/swar/nba_api
- https://youtu.be/YzZGKhBdBWA?si=glAJbX_hR0WgoZ-v
- https://youtu.be/nHtlRlWmTV4?si=rYp6jKFaC1E7Yg6h
- https://pypi.org/project/nba_api/
- https://medium.com/@ben.g.ballard/how-to-analyze-nba-data-using-python-and-the-nba-api-429b0e65454d