Phase3

March 18, 2020

1 Data Analysis on Successful Teams in the NBA

1.1 Introduction

In this phase of the project, we will be making some predictions, and using data analysis methods to prove or disprove the predictions. In this portion specifically, we will be analysizing successful teams in past NBA seasons and try to find some patterns and/or similarities between the teams to see if we can find out what is necessary to be an NBA champsionship team.

Next we will build a model to find how likely current NBA teams are to win the championships this season should they make the finals.

1.2 Finding Successful Traits

[556]:
```
#Reading in datasets
import pandas as pd

df_st = pd.read_csv("successful_teams.csv")
df_potw = pd.read_csv("NBA_player_of_the_week2.csv")
df_finals = pd.read_csv("finals_games.csv")
df_curr_teams = pd.read_csv("teams.csv")
df_opponents = pd.read_csv("opponents.csv")
df_champs = pd.read_csv("nba_champs.csv")
df_runnerups = pd.read_csv("nba_runner_up.csv")
df_nba_rating = pd.read_csv('nba_rating.txt')
df_nba_rating.drop(df_nba_rating.columns[df_nba_rating.columns.str.
 ↪contains('unnamed',case = False)],axis = 1, inplace = True)
```

[557]:
```
print("Successful Teams")
df_st.head(2)
```

```
Successful Teams
```

[557]:

	Unnamed: 0	Year	League	Champion	RunnerUp	FinalsMVP	Points	Rebounds	Assists
0	0	2019	NBA	Toronto Raptors	Golden State Warriors	K. Leonard	K. Leonard (732)	D. Green (223)	D. Green (187)
1	1	2018	NBA	Golden State Warriors	Cleveland Cavaliers				

```
1   K. Durant    L. James (748)  D. Green (222)  L. James (198)
```

[558]:
```
print("Player Of The Week")
df_potw.head(2)
```

```
Player Of The Week
```

[558]:

	Unnamed: 0	Player	Team	Conference	Date	Position	\
0	0	Jayson Tatum	Boston Celtics	East	Feb 10, 2020	SF	
1	1	Nikola Jokic	Denver Nuggets	West	Feb 10, 2020	C	

	Height	Weight	Age	Draft Year	Seasons in league	Season	Season short	\
0	6'8	208	21	2017	2	2019-2020	2020	
1	7'0	250	25	2014	4	2019-2020	2020	

	Pre-draft Team	Real_value	Height CM	Weight KG	Last Season
0	Duke	0.5	203	94	1
1	KK Mega Bemax (Serbia)	0.5	213	113	1

[559]:
```
print("Championship And Runner Up Finals Stats")
df_finals = df_finals.drop(columns=["Unnamed: 0"])
df_finals.head(2)
```

```
Championship And Runner Up Finals Stats
```

[559]:

	Year	Status	Team	PTS	FG	FGA	FGP	TP	\
0	1980	Champion	Lakers	109.5	45.000000	92.000000	48.913043	0.0	
1	1981	Champion	Celtics	96.5	40.166667	85.333333	47.070312	0.5	

	TPA	TPP	…	FTP	ORB	DRB	TRB	\
0	0.666667	0.000000	…	81.250000	17.166667	34.166667	51.333333	
1	2.833333	17.647059	…	72.868217	16.666667	30.666667	47.333333	

	AST	STL	BLK	TOV	PF	Finals_Games
0	26.666667	9.166667	6.166667	20.000000	24.500000	4
1	22.833333	6.666667	5.333333	16.833333	23.166667	4

[2 rows x 22 columns]

[560]:
```
print("Current(2019-2020) Team Stats")
df_curr_teams = df_curr_teams.drop(columns=["Unnamed: 0"])
df_curr_teams.head(2)
```

```
Current(2019-2020) Team Stats
```

[560]:

	Team	FG	FGA	FGP	TP	TPA	TPP	FT	FTA	FTP	\
0	Dallas Mavericks	41.6	90.0	0.462	15.3	41.5	0.369	17.9	23.1	0.773	
1	Milwaukee Bucks	43.5	91.2	0.477	13.7	38.6	0.356	17.8	24.0	0.742	

```
   …   TRB   AST  STL  BLK   TOV    PF    PTS  Year  Status  Finals_Games
0  …  47.0  24.5  6.3  5.0  12.8  19.0  116.4  2020     TBD             0
1  …  51.7  25.9  7.4  6.0  14.9  19.2  118.6  2020     TBD             0

[2 rows x 22 columns]
```

[561]:
```
print("Current(2019-2020) Opposing Team Stats")
df_opponents = df_opponents.drop(columns=["Unnamed: 0"])
df_opponents.head(2)
```

```
Current(2019-2020) Opposing Team Stats
```

[561]:
```
                 Team    FG   FGA    FGP    TP   TPA    TPP    FT   FTA  \
0  Los Angeles Lakers  39.0  87.7  0.444  11.3  33.0  0.342  17.7  22.8
1     Toronto Raptors  37.9  88.5  0.429  13.0  38.5  0.337  17.7  23.0

     FTP  …   TRB   AST  STL  BLK   TOV    PF    PTS  Year  Status  \
0  0.774  …  42.3  23.2  8.2  3.7  15.7  21.4  106.9  2020     TBD
1  0.768  …  46.3  25.6  7.0  5.3  16.8  20.0  106.5  2020     TBD

   Finals_Games
0             0
1             0

[2 rows x 22 columns]
```

[562]:
```
print("Past NBA Champs")
df_champs = df_champs.drop(columns=["Unnamed: 0"])
df_champs.head(2)
```

```
Past NBA Champs
```

[562]:
```
   Year    Status     Team    PTS         FG        FGA        FGP   TP  \
0  1980  Champion   Lakers  109.5  45.000000  92.000000  48.913043  0.0
1  1981  Champion  Celtics   96.5  40.166667  85.333333  47.070312  0.5

        TPA        TPP  …        FTP        ORB        DRB        TRB  \
0  0.666667   0.000000  …  81.250000  17.166667  34.166667  51.333333
1  2.833333  17.647059  …  72.868217  16.666667  30.666667  47.333333

         AST       STL       BLK        TOV        PF  Finals_Games
0  26.666667  9.166667  6.166667  20.000000  24.500000             4
1  22.833333  6.666667  5.333333  16.833333  23.166667             4

[2 rows x 22 columns]
```

[563]:
```
print("Past NBA Runner-Ups")
df_runnerups = df_runnerups.drop(columns=["Unnamed: 0"])
df_runnerups.head(2)
```

```
Past NBA Runner-Ups
```

[563]:

	Year	Status	Team	PTS	FG	FGA	FGP	\
0	1980	Runner Up	Sixers	104.166667	43.000000	88.333333	48.679245	
1	1981	Runner Up	Rockets	86.666667	33.833333	89.333333	37.873134	

	TP	TPA	TPP	…	FTP	ORB	DRB	\
0	0.166667	2.666667	6.250000	…	75.000000	9.500000	27.666667	
1	0.500000	1.833333	27.272727	…	71.612903	18.666667	24.000000	

	TRB	AST	STL	BLK	TOV	PF	Finals_Games
0	37.166667	31.0	9.333333	10.000000	15.166667	22.500000	2
1	42.666667	18.0	7.833333	5.666667	13.166667	20.166667	2

```
[2 rows x 22 columns]
```

1.3 Comparing Champions to Runner-Ups

Before we begin building a model that will try and predict how current NBA teams would do if they were in the finals, I want to try and find what stats have the biggest effect on finals outcome

[564]:
```
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np

plt.rcParams["figure.figsize"] = [20, 10]
```

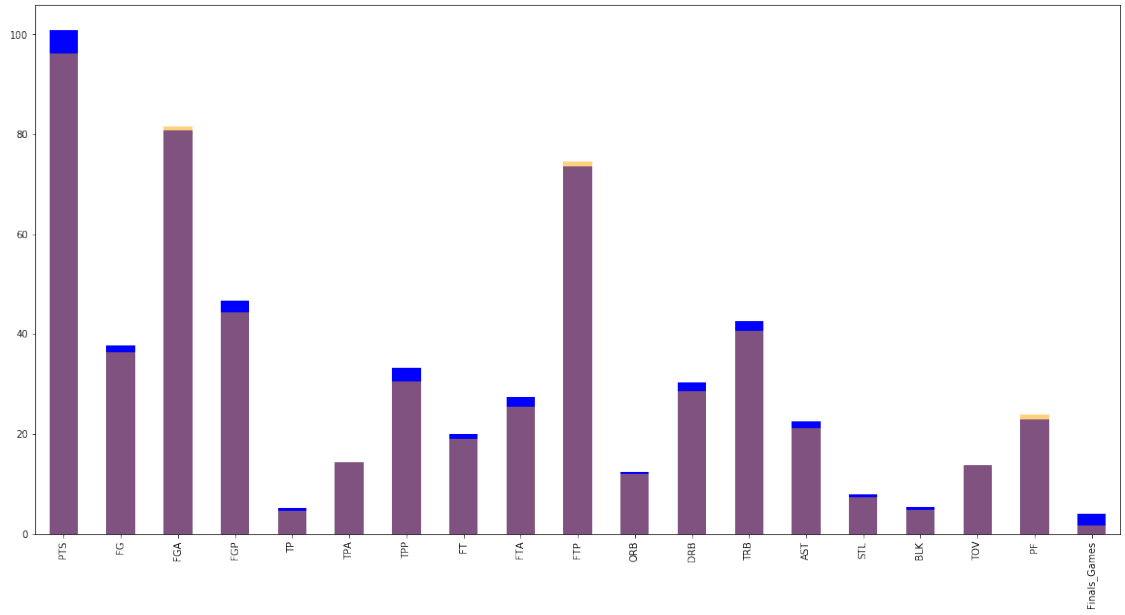
[565]:
```
avg_champ = df_finals[df_finals["Status"] == "Champion"]
avg_ru = df_finals[df_finals["Status"] == "Runner Up"]

plot_avg_champ = avg_champ.drop(columns=["Year", "Status", "Team"])
plot_avg_champ = plot_avg_champ.mean()

plot_avg_ru = avg_ru.drop(columns=["Year", "Status", "Team"])
plot_avg_ru = plot_avg_ru.mean()

fig = plot_avg_champ.plot(kind = 'bar', color = 'blue')
plot_avg_ru.plot(kind = 'bar', color = 'orange', fig = fig, alpha = .5)
```

[565]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd6bc26d5f8>
```



Before we use linear regression lets just digest what the above graph is telling us. Nearly every stat is in favor or the winning team besides 3 (Field Goal Attempts, Free Throw Percentage, Personal Fouls). Well for one FGA and PF is not a good stat to win so we'll ignore that. However surprisingly, Free throw percentage is SLIGHTLY in favor of the losing side which is an interesting stat however the overwhelmingly majority of other stats proves that we can use statistics as a measure of how likely a team is to win the finals this year.

1.4 Predict Successful Teams of 19/20 Season

1.4.1 Model 1 (Team Stats)

First we will need to make training and testing dataframes.

[566]:
```
df_train = df_finals
df_train.head()
```

[566]:

	Year	Status	Team	PTS	FG	FGA	FGP	\
0	1980	Champion	Lakers	109.500000	45.000000	92.000000	48.913043	
1	1981	Champion	Celtics	96.500000	40.166667	85.333333	47.070312	
2	1982	Champion	Lakers	112.333333	45.500000	91.833333	49.546279	
3	1983	Champion	Sixers	110.500000	43.000000	88.500000	48.587571	
4	1984	Champion	Celtics	116.000000	42.000000	92.857143	45.230769	

	TP	TPA	TPP	…	FTP	ORB	DRB	\
0	0.000000	0.666667	0.000000	…	81.250000	17.166667	34.166667	
1	0.500000	2.833333	17.647059	…	72.868217	16.666667	30.666667	
2	0.166667	1.500000	11.111111	…	67.195767	18.333333	29.000000	
3	0.000000	0.750000	0.000000	…	71.014493	18.000000	30.000000	

```
4  1.285714  3.285714  39.130435  …  79.629630  17.428571  30.000000

         TRB        AST        STL       BLK        TOV         PF  \
0  51.333333  26.666667   9.166667  6.166667  20.000000  24.500000
1  47.333333  22.833333   6.666667  5.333333  16.833333  23.166667
2  47.333333  31.333333  10.666667  7.000000  19.333333  25.500000
3  48.000000  26.000000  11.000000  8.000000  17.000000  22.250000
4  47.428571  24.142857   9.857143  4.571429  15.857143  28.142857

   Finals_Games
0             4
1             4
2             4
3             4
4             4

[5 rows x 22 columns]
```

[567]:
```
df_test = df_curr_teams
df_test.head()
```

[567]:
```
                    Team    FG   FGA    FGP    TP   TPA    TPP    FT   FTA  \
0        Dallas Mavericks  41.6  90.0  0.462  15.3  41.5  0.369  17.9  23.1
1         Milwaukee Bucks  43.5  91.2  0.477  13.7  38.6  0.356  17.8  24.0
2         Houston Rockets  41.1  90.7  0.454  15.4  44.3  0.348  20.5  26.0
3  Portland Trail Blazers  41.9  90.9  0.461  12.6  33.8  0.372  17.3  21.7
4           Atlanta Hawks  40.6  90.6  0.449  12.0  36.1  0.333  18.5  23.4

     FTP  …   TRB   AST  STL  BLK   TOV    PF    PTS  Year  Status  \
0  0.773  …  47.0  24.5  6.3  5.0  12.8  19.0  116.4  2020     TBD
1  0.742  …  51.7  25.9  7.4  6.0  14.9  19.2  118.6  2020     TBD
2  0.787  …  44.9  21.5  8.5  5.1  14.7  21.6  118.1  2020     TBD
3  0.798  …  45.5  20.2  6.1  6.2  13.0  21.4  113.6  2020     TBD
4  0.790  …  43.3  24.0  7.8  5.1  16.2  23.1  111.8  2020     TBD

   Finals_Games
0             0
1             0
2             0
3             0
4             0

[5 rows x 22 columns]
```

Note: Thankfully, in phase 2 we spent cleaned our data effectively so that both the training and test datasets have the same columns, and they all of float type.

[568]:
```
df_train.describe()
```

[568]:

	Year	PTS	FG	FGA	FGP	TP
count	76.00000	76.000000	76.000000	76.000000	76.000000	76.000000
mean	1998.50000	98.564223	37.023559	81.179605	45.522578	4.986623
std	11.03872	10.318278	4.831042	6.861020	3.446519	3.440890
min	1980.00000	79.800000	30.000000	67.400000	36.991870	0.000000
25%	1989.00000	90.742857	32.958333	75.425000	43.199529	2.166667
50%	1998.50000	99.190476	36.083333	80.600000	45.402275	4.857143
75%	2008.00000	106.666667	40.812500	87.000000	47.509158	7.258929
max	2017.00000	121.600000	46.714286	92.857143	52.762431	14.200000

	TPA	TPP	FT	FTA	FTP	ORB
count	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000
mean	14.352318	31.894275	19.530482	26.405263	74.031161	12.157832
std	8.687152	9.669931	3.694309	4.745111	5.213839	2.716640
min	0.666667	0.000000	13.000000	16.833333	56.983240	5.600000
25%	6.958333	28.195652	17.107143	22.928571	70.939956	10.371429
50%	14.708333	32.470588	19.500000	25.833333	74.079023	11.732143
75%	20.250000	38.291855	21.833333	29.744048	77.383193	13.714286
max	37.200000	48.000000	30.714286	38.571429	85.164835	18.666667

	DRB	TRB	AST	STL	BLK	TOV
count	76.000000	76.000000	76.000000	76.000000	76.000000	76.000000
mean	29.453885	41.611717	21.813189	7.618421	5.100063	13.728321
std	2.558105	3.389569	4.488547	1.419080	1.612898	2.242452
min	23.800000	32.200000	14.000000	4.000000	2.200000	8.714286
25%	27.833333	38.916667	18.333333	6.833333	4.000000	12.421429
50%	29.309524	41.690476	21.333333	7.633333	5.000000	13.309524
75%	31.297619	43.628571	24.725000	8.488095	5.808333	15.041667
max	35.000000	51.333333	32.000000	11.000000	10.000000	20.000000

	PF	Finals_Games
count	76.000000	76.000000
mean	23.388315	2.842105
std	2.895576	1.327179
min	16.857143	0.000000
25%	21.321429	2.000000
50%	23.083333	3.500000
75%	25.375000	4.000000
max	30.000000	4.000000

[569]:
```
df_test.describe()
```

[569]:

	FG	FGA	FGP	TP	TPA	TPP
count	31.000000	31.000000	31.000000	31.000000	31.000000	31.000000
mean	40.832258	88.806452	0.459871	12.138710	33.938710	0.357548

```
std      1.350404   2.037962   0.012505   1.431008   3.840024   0.013718
min     37.300000  84.400000   0.434000   9.600000  27.500000   0.333000
25%     40.050000  87.850000   0.451500  11.050000  31.500000   0.349500
50%     40.800000  88.800000   0.461000  12.100000  33.800000   0.357000
75%     41.850000  90.300000   0.470000  13.050000  35.600000   0.368000
max     43.500000  92.200000   0.485000  15.400000  44.300000   0.383000

              FT         FTA         FTP         ORB         DRB         TRB  \
count  31.000000  31.000000  31.000000  31.000000  31.000000  31.000000
mean   17.664516  22.900000   0.770774  10.119355  34.745161  44.858065
std     1.568555   1.762574   0.028136   0.857679   2.115946   2.216720
min    15.100000  19.100000   0.694000   8.100000  31.400000  41.700000
25%    16.450000  21.900000   0.752500   9.750000  33.400000  42.950000
50%    17.700000  22.900000   0.772000  10.400000  34.600000  44.800000
75%    18.600000  24.000000   0.791000  10.650000  35.700000  46.050000
max    20.800000  26.200000   0.826000  12.000000  42.200000  51.700000

              AST         STL         BLK         TOV         PF          PTS  \
count  31.000000  31.000000  31.000000  31.000000  31.000000   31.000000
mean   24.348387   7.664516   4.938710  14.522581  20.580645  111.461290
std     1.744491   0.873899   0.748188   1.056633   1.343483    3.843581
min    20.200000   5.900000   3.200000  12.300000  17.600000  102.900000
25%    23.300000   7.200000   4.500000  13.900000  19.650000  109.150000
50%    24.100000   7.600000   5.000000  14.700000  20.600000  111.800000
75%    25.900000   8.200000   5.450000  15.200000  21.550000  113.450000
max    27.200000  10.000000   6.800000  16.500000  23.100000  118.600000

         Year  Finals_Games
count    31.0          31.0
mean   2020.0           0.0
std       0.0           0.0
min    2020.0           0.0
25%    2020.0           0.0
50%    2020.0           0.0
75%    2020.0           0.0
max    2020.0           0.0
```

[570]:
```
X_train = df_train.copy(deep=True) # copy the dataframe
X_train = X_train[["FG", "FGA", "FGP", "TP", "TPA", "TPP", "FT", "FTA", "FTP",␣
 ↪"ORB", "DRB", "TRB", "AST", "STL", "BLK", "BLK", "TOV", "PF", "PTS",␣
 ↪"Finals_Games"]] # use all features for start to see how accurate
#X_train = X_train[["FG", "FGP", "TPP", "FT", "TRB", "AST", "STL", "BLK",␣
 ↪"TOV", "PF", "Finals_Games"]] # use all features for start to see how␣
 ↪accurate
#X_train.head(2)
```

[571]:
```
X_test = df_test.copy(deep=True) # copy the dataframe
X_test = X_test[["FG", "FGA", "FGP", "TP", "TPA", "TPP", "FT", "FTA", "FTP",␣
 ↪"ORB", "DRB", "TRB", "AST", "STL", "BLK", "BLK", "TOV", "PF", "PTS",␣
 ↪"Finals_Games"]]
#X_test = X_test[["FG", "FGP", "TPP", "FT", "TRB", "AST", "STL", "BLK", "TOV",␣
 ↪"PF", "Finals_Games"]]
#X_test.head(2)
```

[579]:
```
from sklearn.linear_model import LinearRegression

y_train = df_train["Finals_Games"]

model = LinearRegression()
model.fit(X=X_train, y=y_train)

y_predict = model.predict(X=X_test)
```

[580]:
```
output=pd.DataFrame(data={"id":df_test.index,"Final_Games":y_predict}) # first␣
 ↪need to create df_test

#output.to_csv(path_or_buf="results.csv",index=False,quoting=3,sep=',')
```

[581]:
```
output
```

[581]:

	id	Final_Games
0	0	1.570966e-14
1	1	3.443285e-05
2	2	1.571263e-04
3	3	-3.443285e-05
4	4	3.443285e-05
5	5	3.443285e-05
6	6	1.713213e-14
7	7	1.684763e-14
8	8	-3.443285e-05
9	9	1.828399e-14
10	10	3.443285e-05
11	11	3.443285e-05
12	12	-3.443285e-05
13	13	-3.443285e-05
14	14	1.462025e-14
15	15	1.610517e-14
16	16	3.443285e-05
17	17	1.544598e-14
18	18	-6.886569e-05
19	19	1.517536e-14
20	20	1.667416e-14
21	21	-1.915592e-04

```
22  22  1.571263e-04
23  23 -3.443285e-05
24  24  1.609823e-14
25  25  1.520312e-14
26  26 -3.443285e-05
27  27  1.915592e-04
28  28  1.571263e-04
29  29  1.521006e-14
30  30 -1.571263e-04
```

[582]:
```
output = output.drop(columns=["id"])
df_final1 = pd.concat([df_nba_rating.reset_index(drop=True), output.
 ↪reset_index(drop=True)], axis=1)
df_final1 = df_final1.drop([0])
df_final1.sort_values("Final_Games", ascending=False)
```

[582]:

	Rk	Team	W	L	ORtg	DRtg	NRtg	Final_Games
27	27.0	Charlotte Hornets	23.0	42.0	106.3	113.3	-7.0	1.915592e-04
28	28.0	Atlanta Hawks	20.0	47.0	107.2	114.8	-7.6	1.571263e-04
2	2.0	Los Angeles Lakers*	49.0	14.0	113.0	105.6	7.4	1.571263e-04
22	22.0	Minnesota Timberwolves	19.0	45.0	108.1	112.2	-4.1	1.571263e-04
4	4.0	Toronto Raptors*	46.0	18.0	111.6	105.2	6.4	3.443285e-05
16	16.0	Phoenix Suns	26.0	39.0	110.5	111.8	-1.3	3.443285e-05
11	11.0	Miami Heat	41.0	24.0	112.7	109.4	3.3	3.443285e-05
10	10.0	Oklahoma City Thunder	40.0	24.0	111.6	109.1	2.5	3.443285e-05
1	1.0	Milwaukee Bucks*	53.0	12.0	112.6	101.9	10.7	3.443285e-05
5	5.0	Dallas Mavericks	40.0	27.0	116.7	110.6	6.1	3.443285e-05
9	9.0	Denver Nuggets	43.0	22.0	112.5	109.5	3.0	1.828399e-14
6	6.0	Boston Celtics*	43.0	21.0	112.9	106.8	6.1	1.713213e-14
7	7.0	Houston Rockets	40.0	24.0	113.8	110.2	3.6	1.684763e-14
20	20.0	San Antonio Spurs	27.0	36.0	111.9	113.7	-1.8	1.667416e-14
15	15.0	Orlando Magic	30.0	35.0	108.0	109.0	-1.0	1.610517e-14
24	24.0	Detroit Pistons	20.0	46.0	109.0	112.7	-3.7	1.609823e-14
17	17.0	Memphis Grizzlies	32.0	33.0	109.4	110.4	-1.0	1.544598e-14
29	29.0	Cleveland Cavaliers	19.0	46.0	107.5	115.4	-7.9	1.521006e-14
25	25.0	Washington Wizards	24.0	40.0	111.9	115.8	-3.9	1.520312e-14
19	19.0	Brooklyn Nets	30.0	34.0	108.1	108.7	-0.6	1.517536e-14
14	14.0	New Orleans Pelicans	28.0	36.0	110.8	111.6	-0.8	1.462025e-14
12	12.0	Philadelphia 76ers	39.0	26.0	110.4	108.2	2.2	-3.443285e-05
13	13.0	Indiana Pacers	39.0	26.0	110.3	108.3	2.0	-3.443285e-05
3	3.0	Los Angeles Clippers	44.0	20.0	113.6	107.2	6.4	-3.443285e-05
26	26.0	New York Knicks	21.0	45.0	106.5	113.0	-6.5	-3.443285e-05
8	8.0	Utah Jazz	41.0	23.0	112.6	109.4	3.2	-3.443285e-05
23	23.0	Chicago Bulls	22.0	43.0	106.7	109.8	-3.1	-3.443285e-05
18	18.0	Portland Trail Blazers	29.0	37.0	112.5	114.1	-1.6	-6.886569e-05
30	30.0	Golden State Warriors	15.0	50.0	105.2	113.8	-8.6	-1.571263e-04
21	21.0	Sacramento Kings	28.0	36.0	109.7	111.6	-1.9	-1.915592e-04

```
31   NaN          League Average   NaN   NaN  110.4  110.4   NaN           NaN
```

As we can see this prediction is not very accurate at all. This is most likely due to the NBA game style changing so much throughout the 1980's to today. Lets try to find a better model.

1.4.2 Model 2 (Opposing Team Average Stats)

Now lets try taking team stats and opposing team stats in order to get differences between the averages to build our 2nd model.

[583]:
```
df_champs.head()
```

[583]:
```
   Year    Status     Team         PTS         FG        FGA        FGP  \
0  1980  Champion   Lakers  109.500000  45.000000  92.000000  48.913043
1  1981  Champion  Celtics   96.500000  40.166667  85.333333  47.070312
2  1982  Champion   Lakers  112.333333  45.500000  91.833333  49.546279
3  1983  Champion   Sixers  110.500000  43.000000  88.500000  48.587571
4  1984  Champion  Celtics  116.000000  42.000000  92.857143  45.230769

         TP       TPA        TPP  …        FTP        ORB        DRB  \
0  0.000000  0.666667   0.000000  …  81.250000  17.166667  34.166667
1  0.500000  2.833333  17.647059  …  72.868217  16.666667  30.666667
2  0.166667  1.500000  11.111111  …  67.195767  18.333333  29.000000
3  0.000000  0.750000   0.000000  …  71.014493  18.000000  30.000000
4  1.285714  3.285714  39.130435  …  79.629630  17.428571  30.000000

         TRB        AST        STL       BLK        TOV         PF  \
0  51.333333  26.666667   9.166667  6.166667  20.000000  24.500000
1  47.333333  22.833333   6.666667  5.333333  16.833333  23.166667
2  47.333333  31.333333  10.666667  7.000000  19.333333  25.500000
3  48.000000  26.000000  11.000000  8.000000  17.000000  22.250000
4  47.428571  24.142857   9.857143  4.571429  15.857143  28.142857

   Finals_Games
0             4
1             4
2             4
3             4
4             4

[5 rows x 22 columns]
```

[584]:
```
df_runnerups.head()
```

[584]:
```
   Year     Status     Team         PTS         FG        FGA        FGP  \
0  1980  Runner Up   Sixers  104.166667  43.000000  88.333333  48.679245
1  1981  Runner Up  Rockets   86.666667  33.833333  89.333333  37.873134
2  1982  Runner Up   Sixers  112.500000  46.166667  92.500000  49.909910
```

```
3  1983  Runner Up   Lakers  100.500000  40.750000  90.250000  45.152355
4  1984  Runner Up   Lakers  117.428571  46.714286  90.714286  51.496063

         TP       TPA        TPP  …        FTP        ORB        DRB  \
0  0.166667  2.666667   6.250000  …  75.000000   9.500000  27.666667
1  0.500000  1.833333  27.272727  …  71.612903  18.666667  24.000000
2  0.666667  2.333333  28.571429  …  70.059880  15.000000  26.666667
3  0.750000  3.750000  20.000000  …  78.494624  15.250000  27.500000
4  0.857143  2.571429  33.333333  …  70.434783  13.714286  30.000000

         TRB        AST       STL        BLK        TOV         PF  \
0  37.166667  31.000000  9.333333  10.000000  15.166667  22.500000
1  42.666667  18.000000  7.833333   5.666667  13.166667  20.166667
2  41.666667  28.666667  8.833333   8.500000  16.666667  26.500000
3  42.750000  24.500000  9.000000   7.250000  20.000000  30.000000
4  43.714286  28.285714  8.428571   5.857143  16.571429  29.428571

   Finals_Games
0             2
1             2
2             2
3             0
4             3

[5 rows x 22 columns]
```

[585]:
```
import numpy as np
df_diff = df_champs
for i, row in df_diff.iterrows():
    temp_val = df_champs.at[i, 'PTS'] - df_runnerups.at[i, 'PTS']
    df_diff.at[i, 'PTS'] = temp_val
    temp_val = df_champs.at[i, 'FG'] - df_runnerups.at[i, 'FG']
    df_diff.at[i, 'FG'] = temp_val
    temp_val = df_champs.at[i, 'FGA'] - df_runnerups.at[i, 'FGA']
    df_diff.at[i, 'FGA'] = temp_val
    temp_val = df_champs.at[i, 'FGP'] - df_runnerups.at[i, 'FGP']
    df_diff.at[i, 'FGP'] = temp_val
    temp_val = df_champs.at[i, 'TP'] - df_runnerups.at[i, 'TP']
    df_diff.at[i, 'TP'] = temp_val
    temp_val = df_champs.at[i, 'TPA'] - df_runnerups.at[i, 'TPA']
    df_diff.at[i, 'TPA'] = temp_val
    temp_val = df_champs.at[i, 'TPP'] - df_runnerups.at[i, 'TPP']
    df_diff.at[i, 'TPP'] = temp_val
    temp_val = df_champs.at[i, 'FT'] - df_runnerups.at[i, 'FT']
    df_diff.at[i, 'FT'] = temp_val
    temp_val = df_champs.at[i, 'FTA'] - df_runnerups.at[i, 'FTA']
    df_diff.at[i, 'FTA'] = temp_val
```

```
    temp_val = df_champs.at[i, 'FTP'] - df_runnerups.at[i, 'FTP']
    df_diff.at[i, 'FTP'] = temp_val
    temp_val = df_champs.at[i, 'ORB'] - df_runnerups.at[i, 'ORB']
    df_diff.at[i, 'ORB'] = temp_val
    temp_val = df_champs.at[i, 'DRB'] - df_runnerups.at[i, 'DRB']
    df_diff.at[i, 'DRB'] = temp_val
    temp_val = df_champs.at[i, 'TRB'] - df_runnerups.at[i, 'TRB']
    df_diff.at[i, 'TRB'] = temp_val
    temp_val = df_champs.at[i, 'AST'] - df_runnerups.at[i, 'AST']
    df_diff.at[i, 'AST'] = temp_val
    temp_val = df_champs.at[i, 'STL'] - df_runnerups.at[i, 'STL']
    df_diff.at[i, 'STL'] = temp_val
    temp_val = df_champs.at[i, 'BLK'] - df_runnerups.at[i, 'BLK']
    df_diff.at[i, 'BLK'] = temp_val
    temp_val = df_champs.at[i, 'TOV'] - df_runnerups.at[i, 'TOV']
    df_diff.at[i, 'TOV'] = temp_val
    temp_val = df_champs.at[i, 'PF'] - df_runnerups.at[i, 'PF']
    df_diff.at[i, 'PF'] = temp_val
    temp_val = df_champs.at[i, 'Finals_Games'] - df_runnerups.at[i,␣
↪'Finals_Games']
    df_diff.at[i, 'Finals_Games'] = temp_val

df_diff.head()
```

[585]:

	Year	Status	Team	PTS	FG	FGA	FGP	TP	\
0	1980	Champion	Lakers	5.333333	2.000000	3.666667	0.233798	-0.166667	
1	1981	Champion	Celtics	9.833333	6.333333	-4.000000	9.197178	0.000000	
2	1982	Champion	Lakers	-0.166667	-0.666667	-0.666667	-0.363630	-0.500000	
3	1983	Champion	Sixers	10.000000	2.250000	-1.750000	3.435216	-0.750000	
4	1984	Champion	Celtics	-1.428571	-4.714286	2.142857	-6.265294	0.428571	

	TPA	TPP	…	FTP	ORB	DRB	TRB	\
0	-2.000000	-6.250000	…	6.250000	7.666667	6.500000	14.166667	
1	1.000000	-9.625668	…	1.255314	-2.000000	6.666667	4.666667	
2	-0.833333	-17.460317	…	-2.864113	3.333333	2.333333	5.666667	
3	-3.000000	-20.000000	…	-7.480131	2.750000	2.500000	5.250000	
4	0.714286	5.797101	…	9.194847	3.714286	0.000000	3.714286	

	AST	STL	BLK	TOV	PF	Finals_Games
0	-4.333333	-0.166667	-3.833333	4.833333	2.000000	2
1	4.833333	-1.166667	-0.333333	3.666667	3.000000	2
2	2.666667	1.833333	-1.500000	2.666667	-1.000000	2
3	1.500000	2.000000	0.750000	-3.000000	-7.750000	4
4	-4.142857	1.428571	-1.285714	-0.714286	-1.285714	1

[5 rows x 22 columns]

[586]:
```
df_curr_diff = df_curr_teams
i = 0

for i, row in df_curr_teams.iterrows():
    temp_val = df_curr_teams.at[i, 'PTS'] - df_opponents.at[i, 'PTS']
    df_curr_diff.at[i, 'PTS'] = temp_val
    temp_val = df_curr_teams.at[i, 'FG'] - df_opponents.at[i, 'FG']
    df_curr_diff.at[i, 'FG'] = temp_val
    temp_val = df_curr_teams.at[i, 'FGA'] - df_opponents.at[i, 'FGA']
    df_curr_diff.at[i, 'FGA'] = temp_val
    temp_val = df_curr_teams.at[i, 'FGP'] - df_opponents.at[i, 'FGP']
    df_curr_diff.at[i, 'FGP'] = temp_val
    temp_val = df_curr_teams.at[i, 'TP'] - df_opponents.at[i, 'TP']
    df_curr_diff.at[i, 'TP'] = temp_val
    temp_val = df_curr_teams.at[i, 'TPA'] - df_opponents.at[i, 'TPA']
    df_curr_diff.at[i, 'TPA'] = temp_val
    temp_val = df_curr_teams.at[i, 'TPP'] - df_opponents.at[i, 'TPP']
    df_curr_diff.at[i, 'TPP'] = temp_val
    temp_val = df_curr_teams.at[i, 'FT'] - df_opponents.at[i, 'FT']
    df_curr_diff.at[i, 'FT'] = temp_val
    temp_val = df_curr_teams.at[i, 'FTA'] - df_opponents.at[i, 'FTA']
    df_curr_diff.at[i, 'FTA'] = temp_val
    temp_val = df_curr_teams.at[i, 'FTP'] - df_opponents.at[i, 'FTP']
    df_curr_diff.at[i, 'FTP'] = temp_val
    temp_val = df_curr_teams.at[i, 'ORB'] - df_opponents.at[i, 'ORB']
    df_curr_diff.at[i, 'ORB'] = temp_val
    temp_val = df_curr_teams.at[i, 'DRB'] - df_opponents.at[i, 'DRB']
    df_curr_diff.at[i, 'DRB'] = temp_val
    temp_val = df_curr_teams.at[i, 'TRB'] - df_opponents.at[i, 'TRB']
    df_curr_diff.at[i, 'TRB'] = temp_val
    temp_val = df_curr_teams.at[i, 'AST'] - df_opponents.at[i, 'AST']
    df_curr_diff.at[i, 'AST'] = temp_val
    temp_val = df_curr_teams.at[i, 'STL'] - df_opponents.at[i, 'STL']
    df_curr_diff.at[i, 'STL'] = temp_val
    temp_val = df_curr_teams.at[i, 'BLK'] - df_opponents.at[i, 'BLK']
    df_curr_diff.at[i, 'BLK'] = temp_val
    temp_val = df_curr_teams.at[i, 'TOV'] - df_opponents.at[i, 'TOV']
    df_curr_diff.at[i, 'TOV'] = temp_val
    temp_val = df_curr_teams.at[i, 'PF'] - df_opponents.at[i, 'PF']
    df_curr_diff.at[i, 'PF'] = temp_val
    temp_val = df_curr_teams.at[i, 'Finals_Games'] - df_opponents.at[i,␣
 ↪'Finals_Games']
    df_curr_diff.at[i, 'Finals_Games'] = temp_val

df_curr_diff.head()
```

[586]:
```
                    Team   FG  FGA    FGP   TP  TPA    TPP   FT  FTA    FTP  \
0        Dallas Mavericks  2.6  2.3  0.018  4.0  8.5  0.027  0.2  0.3 -0.001
1         Milwaukee Bucks  5.6  2.7  0.048  0.7  0.1  0.019  0.1  1.0 -0.026
2         Houston Rockets  2.8  3.9  0.012  3.6  9.8  0.006  2.1  1.8  0.026
3  Portland Trail Blazers  1.6  1.6  0.009  1.5  2.2  0.021  1.2  0.8  0.029
4           Atlanta Hawks -0.5  0.6 -0.007  0.6  2.9 -0.011  3.7  4.8 -0.005

   …  TRB  AST  STL  BLK  TOV   PF   PTS  Year  Status  Finals_Games
0  …  4.7  1.3 -1.9  1.3 -2.9 -2.4   9.5  2020     TBD             0
1  …  5.4  0.3  0.4  0.7 -1.9 -0.8  12.1  2020     TBD             0
2  …  0.6 -0.7  1.5 -0.5 -0.6  1.0  11.3  2020     TBD             0
3  …  2.9 -1.0 -1.7  1.6  1.0  0.8   5.7  2020     TBD             0
4  … -1.1  0.9  1.1  1.0  2.2  0.3   3.5  2020     TBD             0

[5 rows x 22 columns]
```

[587]:
```
df_train2 = df_diff
df_test2 = df_curr_diff
```

[588]:
```
X_train2 = df_train2.copy(deep=True) # copy the dataframe
#X_train2 = X_train2[["FG", "FGA", "FGP", "TP", "TPA", "TPP", "FT", "FTA",␣
 ↪"FTP", "ORB", "DRB", "TRB", "AST", "STL", "BLK", "BLK", "TOV", "PF", "PTS",␣
 ↪"Finals_Games"]] # use all features for start to see how accurate
X_train2 = X_train2[["PTS", "TOV", "TRB", "TPP", "FGP", "Finals_Games"]] # use␣
 ↪all features for start to see how accurate
X_train2.head(2)
```

[588]:
```
        PTS       TOV        TRB       TPP       FGP  Finals_Games
0  5.333333  4.833333  14.166667 -6.250000  0.233798             2
1  9.833333  3.666667   4.666667 -9.625668  9.197178             2
```

[589]:
```
X_test2 = df_test2.copy(deep=True) # copy the dataframe
#X_test2 = X_test2[["FG", "FGA", "FGP", "TP", "TPA", "TPP", "FT", "FTA", "FTP",␣
 ↪"ORB", "DRB", "TRB", "AST", "STL", "BLK", "BLK", "TOV", "PF", "PTS",␣
 ↪"Finals_Games"]]
X_test2 = X_test2[["PTS", "TOV", "TRB", "TPP", "FGP", "Finals_Games"]]
X_test2.head(2)
```

[589]:
```
    PTS  TOV  TRB    TPP    FGP  Finals_Games
0   9.5 -2.9  4.7  0.027  0.018             0
1  12.1 -1.9  5.4  0.019  0.048             0
```

[590]:
```
y_train2 = df_train2["Finals_Games"]

model2 = LinearRegression()
model2.fit(X=X_train2, y=y_train2)
```

```
y_predict2 = model2.predict(X=X_test2)
```

[591]:
```
model2.intercept_
```

[591]: 4.440892098500626e-16

[592]:
```
output2=pd.DataFrame(data={"id":df_test.index,"Final_Games":y_predict2}) #␣
↪first need to create df_test
```

[593]:
```
output2["Final_Games"] = output2["Final_Games"] / 4.440892098500626e-16
```

[594]:
```
output2
```

[594]:

	id	Final_Games
0	0	3.020106
1	1	3.515590
2	2	3.698705
3	3	2.032638
4	4	1.811065
5	5	2.876489
6	6	2.863168
7	7	2.983114
8	8	2.003906
9	9	2.171656
10	10	1.639663
11	11	1.718021
12	12	1.364502
13	13	1.892119
14	14	1.526136
15	15	1.520420
16	16	0.757222
17	17	0.512829
18	18	0.330041
19	19	0.315069
20	20	1.278608
21	21	-0.165217
22	22	-0.090147
23	23	-1.572561
24	24	0.012522
25	25	-1.470222
26	26	-1.062839
27	27	-0.822918
28	28	-1.795207
29	29	-2.617701
30	30	1.000000

[595]:
```
output2 = output2.drop(columns=["id"])
df_final = pd.concat([df_nba_rating.reset_index(drop=True), output2.
 ↪reset_index(drop=True)], axis=1)
df_final = df_final.drop([0])
df_final.sort_values("Final_Games", ascending=False)
```

[595]:

	Rk	Team	W	L	ORtg	DRtg	NRtg	Final_Games
2	2.0	Los Angeles Lakers*	49.0	14.0	113.0	105.6	7.4	3.698705
1	1.0	Milwaukee Bucks*	53.0	12.0	112.6	101.9	10.7	3.515590
7	7.0	Houston Rockets	40.0	24.0	113.8	110.2	3.6	2.983114
5	5.0	Dallas Mavericks	40.0	27.0	116.7	110.6	6.1	2.876489
6	6.0	Boston Celtics*	43.0	21.0	112.9	106.8	6.1	2.863168
9	9.0	Denver Nuggets	43.0	22.0	112.5	109.5	3.0	2.171656
3	3.0	Los Angeles Clippers	44.0	20.0	113.6	107.2	6.4	2.032638
8	8.0	Utah Jazz	41.0	23.0	112.6	109.4	3.2	2.003906
13	13.0	Indiana Pacers	39.0	26.0	110.3	108.3	2.0	1.892119
4	4.0	Toronto Raptors*	46.0	18.0	111.6	105.2	6.4	1.811065
11	11.0	Miami Heat	41.0	24.0	112.7	109.4	3.3	1.718021
10	10.0	Oklahoma City Thunder	40.0	24.0	111.6	109.1	2.5	1.639663
14	14.0	New Orleans Pelicans	28.0	36.0	110.8	111.6	-0.8	1.526136
15	15.0	Orlando Magic	30.0	35.0	108.0	109.0	-1.0	1.520420
12	12.0	Philadelphia 76ers	39.0	26.0	110.4	108.2	2.2	1.364502
20	20.0	San Antonio Spurs	27.0	36.0	111.9	113.7	-1.8	1.278608
30	30.0	Golden State Warriors	15.0	50.0	105.2	113.8	-8.6	1.000000
16	16.0	Phoenix Suns	26.0	39.0	110.5	111.8	-1.3	0.757222
17	17.0	Memphis Grizzlies	32.0	33.0	109.4	110.4	-1.0	0.512829
18	18.0	Portland Trail Blazers	29.0	37.0	112.5	114.1	-1.6	0.330041
19	19.0	Brooklyn Nets	30.0	34.0	108.1	108.7	-0.6	0.315069
24	24.0	Detroit Pistons	20.0	46.0	109.0	112.7	-3.7	0.012522
22	22.0	Minnesota Timberwolves	19.0	45.0	108.1	112.2	-4.1	-0.090147
21	21.0	Sacramento Kings	28.0	36.0	109.7	111.6	-1.9	-0.165217
27	27.0	Charlotte Hornets	23.0	42.0	106.3	113.3	-7.0	-0.822918
26	26.0	New York Knicks	21.0	45.0	106.5	113.0	-6.5	-1.062839
25	25.0	Washington Wizards	24.0	40.0	111.9	115.8	-3.9	-1.470222
23	23.0	Chicago Bulls	22.0	43.0	106.7	109.8	-3.1	-1.572561
28	28.0	Atlanta Hawks	20.0	47.0	107.2	114.8	-7.6	-1.795207
29	29.0	Cleveland Cavaliers	19.0	46.0	107.5	115.4	-7.9	-2.617701
31	NaN	League Average	NaN	NaN	110.4	110.4	NaN	NaN

1.5 Conclusion:

As we take a look above, we have fairly accurate representation of who is most likely to win the NBA finals. We utilized our knowledge on the differential stats between championship teams and runner-ups in order to train our machine to predict who will win the NBA finals this year.

On the table above, the first column, "Rk", is the current standings of NBA teams. The order of the table is based on who is most likely to win the NBA finals.

Of course our prediction is based on simply stats so we shouldn't bet money on this prediction, however, considering our prediction is based on the stats of previous winners, we can say our prediction wouldn't be too off.