In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import os
import random
import shutil
import plotly
os.chdir('C:\\Users\\Travis\\OneDrive\\Data Science\\Personal_Projects\\Sports\\NBA
```

Initial Features

These initial features include:

- Rolling season averages for each of the team metrics per game (around 60 metrics)
- The difference between the two team's rolling averages.

Analyze the Boxscore Dataframe that we will be using to solve

In [ ]:
```
df = pd.read_csv('data/team/aggregates/Both_Team_Boxscores_ALL_with_Trad_Difs.csv')
df.head()
```

Out[ ]:

	Unnamed: 0	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_
0	1513	1514	LAL	LAL vs. SAC	2010-01-01	W	48.0	10
1	1512	1513	ATL	ATL vs. NYK	2010-01-01	L	53.0	10
2	1511	1512	ORL	ORL @ MIN	2010-01-01	W	48.0	10
3	1510	1511	NYK	NYK @ ATL	2010-01-01	W	53.0	11
4	1509	1510	MIN	MIN vs. ORL	2010-01-01	L	48.0	9

5 rows × 230 columns

In [ ]:
```
colz= list(df.columns)
```

In [ ]:
```
df.trad_season.unique()
```

Out[ ]:
```
array([2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
       2020, 2021, 2022], dtype=int64)
```

In [ ]:
```
df['my_gameid'] = df['trad_matchup'] + '_' + df['trad_gamedate'].replace('/','_')
```

Make trad_gamedates work

In [ ]:
```
# fix trad_gamedates
df['trad_gamedate'] = pd.to_datetime(df['trad_gamedate'])
df['trad_gamedate'] = df['trad_gamedate'].dt.strftime('%Y-%m-%d')
df.sort_values('trad_gamedate', ascending = False, inplace=True)
df
```

Out[ ]:

	Unnamed: 0	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	t
33065	29928	19	CHI	CHI @ NYK	2022-12-23	W	48.0	
33050	29926	17	NYK	NYK vs. CHI	2022-12-23	L	48.0	
33038	29924	15	TOR	TOR @ CLE	2022-12-23	W	48.0	
33039	29937	28	PHI	PHI vs. LAC	2022-12-23	W	48.0	
33040	29936	27	MIA	MIA vs. IND	2022-12-23	L	48.0	
...	...	...	...	...	...	...	...	
2	1511	1512	ORL	ORL @ MIN	2010-01-01	W	48.0	
1	1512	1513	ATL	ATL vs. NYK	2010-01-01	L	53.0	
4	1509	1510	MIN	MIN vs. ORL	2010-01-01	L	48.0	
5	1508	1509	SAC	SAC @ LAL	2010-01-01	L	48.0	
0	1513	1514	LAL	LAL vs. SAC	2010-01-01	W	48.0	

33066 rows × 231 columns

From here, df is 2016 on

In [ ]:
```
df = df[df.trad_season >= 2016]
df
```

Out[ ]:

	Unnamed: 0	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	t
33065	29928	19	CHI	CHI @ NYK	2022-12-23	W	48.0	
33050	29926	17	NYK	NYK vs. CHI	2022-12-23	L	48.0	
33038	29924	15	TOR	TOR @ CLE	2022-12-23	W	48.0	
33039	29937	28	PHI	PHI vs. LAC	2022-12-23	W	48.0	
33040	29936	27	MIA	MIA vs. IND	2022-12-23	L	48.0	
...	...	...	...	...	...	...	...	...
16968	18251	2460	NYK	NYK @ CLE	2016-10-25	L	48.0	
16969	18250	2459	GSW	GSW vs. SAS	2016-10-25	L	48.0	
16973	18246	2455	UTA	UTA @ POR	2016-10-25	L	48.0	
16971	18248	2457	CLE	CLE vs. NYK	2016-10-25	W	48.0	
16972	18247	2456	POR	POR vs. UTA	2016-10-25	W	48.0	

16098 rows × 231 columns

In [ ]:
```
df_reg = df[df.trad_season_type == 'Regular']
```

In [ ]:
```
#team_games_2016
games = df['my_gameid'].unique()
games
```

Out[ ]:
```
array(['CHI @ NYK_2022-12-23', 'NYK vs. CHI_2022-12-23',
       'TOR @ CLE_2022-12-23', ..., 'UTA @ POR_2016-10-25',
       'CLE vs. NYK_2016-10-25', 'POR vs. UTA_2016-10-25'], dtype=object)
```

In [ ]:
```
def get_avgs_by_game(my_gameid, team, season, game_date, metric):
    data = df[df['trad_team'] == team]
    data = data[data['trad_season'] == season]
    data = data[data['trad_season_type'] == 'Regular']
    # filter by date
    data = data[data['trad_gamedate'] < game_date]
    data = data.dropna(subset = [metric])
    metric = data[metric].mean()
    return metric
```

In [ ]:
```
# test on a random row from df
random_row = df.sample(1)
get_avgs_by_game(random_row['my_gameid'].values[0], random_row['trad_team'].values[
```

Out[ ]:
```
97.83333333333333
```

Add Team 1 running averages

Question: What are the most important features to keep? Should we just make as many as possible and analyze from there? It does take a bit of processing power to calculate these metrics.

In [ ]:
```
colz = pd.DataFrame(df.columns)
colz
```

Out[ ]:

	0
0	Unnamed: 0
1	trad_unnamed: 0
2	trad_team
3	trad_matchup
4	trad_gamedate
...	...
226	t1_t2_blk
227	t1_t2_tov
228	t1_t2_pf
229	who_wins
230	my_gameid

231 rows × 1 columns

In [ ]:
```
#get possible features from non-object columns
# identify columns that are numeric
numeric_cols = [col for col in df.columns if df[col].dtype in ['int64', 'float64']]
numeric_cols
```

Out[ ]:
```
['Unnamed: 0',
 'trad_unnamed: 0',
 'trad_min',
 'trad_pts',
 'trad_fgm',
 'trad_fga',
 'trad_fg%',
 'trad_3pm',
 'trad_3pa',
 'trad_3p%',
 'trad_ftm',
 'trad_fta',
 'trad_ft%',
 'trad_oreb',
 'trad_dreb',
 'trad_reb',
 'trad_ast',
 'trad_tov',
 'trad_stl',
 'trad_blk',
 'trad_pf',
 'trad_+/-',
 'trad_season',
 'adv_unnamed: 0',
 'adv_min',
 'adv_offrtg',
 'adv_defrtg',
 'adv_netrtg',
 'adv_ast%',
 'adv_ast/to',
 'adv_astratio',
 'adv_oreb%',
 'adv_dreb%',
 'adv_reb%',
 'adv_tov%',
 'adv_efg%',
 'adv_ts%',
 'adv_pace',
 'adv_pie',
 'adv_season',
 'four_unnamed: 0',
 'four_min',
 'four_efg%',
 'four_ftarate',
 'four_tov%',
 'four_oreb%',
 'four_oppefg%',
 'four_oppfta\xa0rate',
 'four_opptov%',
 'four_opporeb%',
 'four_season',
 'misc_unnamed: 0',
 'misc_min',
 'misc_ptsoff\xa0to',
 'misc_2ndpts',
 'misc_fbps',
```

```
'misc_pitp',
'misc_opp\xa0ptsoff\xa0to',
'misc_opp2nd\xa0pts',
'misc_oppfbps',
'misc_opppitp',
'misc_season',
'score_unnamed: 0',
'score_min',
'score_%fga2pt',
'score_%fga3pt',
'score_%pts2pt',
'score_%pts2pt\xa0mr',
'score_%pts3pt',
'score_%ptsfbps',
'score_%ptsft',
'score_%ptsoff\xa0to',
'score_%ptspitp',
'score_2fgm%ast',
'score_2fgm%uast',
'score_3fgm%ast',
'score_3fgm%uast',
'score_fgm%ast',
'score_fgm%uast',
'score_season',
'tm2__trad_unnamed: 0',
'tm2__trad_min',
'tm2__trad_pts',
'tm2__trad_fgm',
'tm2__trad_fga',
'tm2__trad_fg%',
'tm2__trad_3pm',
'tm2__trad_3pa',
'tm2__trad_3p%',
'tm2__trad_ftm',
'tm2__trad_fta',
'tm2__trad_ft%',
'tm2__trad_oreb',
'tm2__trad_dreb',
'tm2__trad_reb',
'tm2__trad_ast',
'tm2__trad_tov',
'tm2__trad_stl',
'tm2__trad_blk',
'tm2__trad_pf',
'tm2__trad_+/-',
'tm2__trad_season',
'tm2__adv_unnamed: 0',
'tm2__adv_min',
'tm2__adv_offrtg',
'tm2__adv_defrtg',
'tm2__adv_netrtg',
'tm2__adv_ast%',
'tm2__adv_ast/to',
'tm2__adv_astratio',
'tm2__adv_oreb%',
'tm2__adv_dreb%',
```

```
'tm2__adv_reb%',
'tm2__adv_tov%',
'tm2__adv_efg%',
'tm2__adv_ts%',
'tm2__adv_pace',
'tm2__adv_pie',
'tm2__adv_season',
'tm2__four_unnamed: 0',
'tm2__four_min',
'tm2__four_efg%',
'tm2__four_ftarate',
'tm2__four_tov%',
'tm2__four_oreb%',
'tm2__four_oppefg%',
'tm2__four_oppfta\xa0rate',
'tm2__four_opptov%',
'tm2__four_opporeb%',
'tm2__four_season',
'tm2__misc_unnamed: 0',
'tm2__misc_min',
'tm2__misc_ptsoff\xa0to',
'tm2__misc_2ndpts',
'tm2__misc_fbps',
'tm2__misc_pitp',
'tm2__misc_opp\xa0ptsoff\xa0to',
'tm2__misc_opp2nd\xa0pts',
'tm2__misc_oppfbps',
'tm2__misc_opppitp',
'tm2__misc_season',
'tm2__score_unnamed: 0',
'tm2__score_min',
'tm2__score_%fga2pt',
'tm2__score_%fga3pt',
'tm2__score_%pts2pt',
'tm2__score_%pts2pt\xa0mr',
'tm2__score_%pts3pt',
'tm2__score_%ptsfbps',
'tm2__score_%ptsft',
'tm2__score_%ptsoff\xa0to',
'tm2__score_%ptspitp',
'tm2__score_2fgm%ast',
'tm2__score_2fgm%uast',
'tm2__score_3fgm%ast',
'tm2__score_3fgm%uast',
'tm2__score_fgm%ast',
'tm2__score_fgm%uast',
'tm2__score_season',
't1_t2_pts',
't1_t2_fgm',
't1_t2_fga',
't1_t2_fg_percent',
't1_t2_3pm',
't1_t2_3pa',
't1_t2_3p_percent',
't1_t2_ftm',
't1_t2_fta',
```

```
 't1_t2_ft_percent',
 't1_t2_oreb',
 't1_t2_dreb',
 't1_t2_reb',
 't1_t2_ast',
 't1_t2_stl',
 't1_t2_blk',
 't1_t2_tov',
 't1_t2_pf']
```

In [ ]:
```
for col in numeric_cols:
    if 'season' in col:
        numeric_cols.remove(col)
```

In [ ]:
```
for col in numeric_cols:
    if 'season' in col:
        numeric_cols.remove(col)
```

In [ ]:
```
features = numeric_cols[2:67]
features
```

Out[ ]:
```
['trad_min',
 'trad_pts',
 'trad_fgm',
 'trad_fga',
 'trad_fg%',
 'trad_3pm',
 'trad_3pa',
 'trad_3p%',
 'trad_ftm',
 'trad_fta',
 'trad_ft%',
 'trad_oreb',
 'trad_dreb',
 'trad_reb',
 'trad_ast',
 'trad_tov',
 'trad_stl',
 'trad_blk',
 'trad_pf',
 'trad_+/-',
 'adv_unnamed: 0',
 'adv_min',
 'adv_offrtg',
 'adv_defrtg',
 'adv_netrtg',
 'adv_ast%',
 'adv_ast/to',
 'adv_astratio',
 'adv_oreb%',
 'adv_dreb%',
 'adv_reb%',
 'adv_tov%',
 'adv_efg%',
 'adv_ts%',
 'adv_pace',
 'adv_pie',
 'four_unnamed: 0',
 'four_min',
 'four_efg%',
 'four_ftarate',
 'four_tov%',
 'four_oreb%',
 'four_oppefg%',
 'four_oppfta\xa0rate',
 'four_opptov%',
 'four_opporeb%',
 'misc_unnamed: 0',
 'misc_min',
 'misc_ptsoff\xa0to',
 'misc_2ndpts',
 'misc_fbps',
 'misc_pitp',
 'misc_opp\xa0ptsoff\xa0to',
 'misc_opp2nd\xa0pts',
 'misc_oppfbps',
 'misc_opppitp',
```

```
 'score_unnamed: 0',
 'score_min',
 'score_%fga2pt',
 'score_%fga3pt',
 'score_%pts2pt',
 'score_%pts2pt\xa0mr',
 'score_%pts3pt',
 'score_%ptsfbps',
 'score_%ptsft']
```

Create average features for each of these for each team

In [ ]:
```
# ignore warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
# Note: 72 min runtime

for f in features:
    df['t1_running_' + f] = df.apply(lambda row: get_avgs_by_game(row['my_gameid'],
```

Add Team 2

In [ ]:
```
# Note: 66 min runtime

for f in features:
    df['t2_running_' + f] = df.apply(lambda row: get_avgs_by_game(
                                                 row['my_gameid'],
                                                 row['team_2'],
                                                 row['trad_season'],
                                                 row['trad_gamedate'],
                                                 f),
                                                 axis=1)
```

Add dif between teams

In [ ]:
```
for f in features:
    df['running_t1-t2_' + f] = df['t1_running_' + f] - df['t2_running_' + f]
```

In [ ]:
```
df.to_csv('data/team/aggregates/Boxes_GameDifferences_and_RunningSeasonAverages.csv
```

Load from here on pickup

In [ ]:
```
df = pd.read_csv('data/team/aggregates/Boxes_GameDifferences_and_RunningSeasonAvera
```

Add All Teams Running (Mean of all Teams) avgs (to compare)

For this we need to change the function to all teams

In [ ]:
```
def get_league_avgs_by_game(my_gameid, season, game_date, metric):
    data = df[df['trad_season'] == season]
    data = data[data['trad_season_type'] == 'Regular']
    # filter by date
    data = data[data['trad_gamedate'] < game_date]
    data = data.dropna(subset = [metric])
    metric = data[metric].mean()
    return metric
```

In [ ]:
```
# Takes 140 minutes to run

for f in features:
    df['league_running_' + f] = df.apply(lambda row: get_league_avgs_by_game(row['m
```

In [ ]:
```
df.to_csv('data/team/aggregates/Boxes_GameDifferences_and_RunningSeasonAverages_and
```

Add Both Teams Delta from the Mean of All Teams

For each metric

In [ ]:
```
df = pd.read_csv('data/team/aggregates/Boxes_GameDifferences_and_RunningSeasonAvera
```

In [ ]:
```
colz = df.columns
colz = [c for c in colz if 'league_running' in c]
list(colz)
```

Out[ ]:
```
['league_running_trad_min',
 'league_running_trad_pts',
 'league_running_trad_fgm',
 'league_running_trad_fga',
 'league_running_trad_fg%',
 'league_running_trad_3pm',
 'league_running_trad_3pa',
 'league_running_trad_3p%',
 'league_running_trad_ftm',
 'league_running_trad_fta',
 'league_running_trad_ft%',
 'league_running_trad_oreb',
 'league_running_trad_dreb',
 'league_running_trad_reb',
 'league_running_trad_ast',
 'league_running_trad_tov',
 'league_running_trad_stl',
 'league_running_trad_blk',
 'league_running_trad_pf',
 'league_running_trad_+/-',
 'league_running_adv_unnamed: 0',
 'league_running_adv_min',
 'league_running_adv_offrtg',
 'league_running_adv_defrtg',
 'league_running_adv_netrtg',
 'league_running_adv_ast%',
 'league_running_adv_ast/to',
 'league_running_adv_astratio',
 'league_running_adv_oreb%',
 'league_running_adv_dreb%',
 'league_running_adv_reb%',
 'league_running_adv_tov%',
 'league_running_adv_efg%',
 'league_running_adv_ts%',
 'league_running_adv_pace',
 'league_running_adv_pie',
 'league_running_four_unnamed: 0',
 'league_running_four_min',
 'league_running_four_efg%',
 'league_running_four_ftarate',
 'league_running_four_tov%',
 'league_running_four_oreb%',
 'league_running_four_oppefg%',
 'league_running_four_oppfta\xa0rate',
 'league_running_four_opptov%',
 'league_running_four_opporeb%',
 'league_running_misc_unnamed: 0',
 'league_running_misc_min',
 'league_running_misc_ptsoff\xa0to',
 'league_running_misc_2ndpts',
 'league_running_misc_fbps',
 'league_running_misc_pitp',
 'league_running_misc_opp\xa0ptsoff\xa0to',
 'league_running_misc_opp2nd\xa0pts',
 'league_running_misc_oppfbps',
 'league_running_misc_opppitp',
```

```
 'league_running_score_unnamed: 0',
 'league_running_score_min',
 'league_running_score_%fga2pt',
 'league_running_score_%fga3pt',
 'league_running_score_%pts2pt',
 'league_running_score_%pts2pt\xa0mr',
 'league_running_score_%pts3pt',
 'league_running_score_%ptsfbps',
 'league_running_score_%ptsft']
```

In [ ]:
```
features = features[:-1]
```

I want to add:

- t1_running_f - league_running_f,
- T2-avg, and
- (T1-avg) - (T2-avg)

In [ ]:
```
#t1_running - league_running
for f in features:
    df['t1_league_delta_' + f] = df['t1_running_' + f] - df['league_running_' + f]
```

In [ ]:
```
for f in features:
    df['t2_league_delta_' + f] = df['t2_running_' + f] - df['league_running_' + f]
```

In [ ]:
```
for f in features:
    df['t1_delta_minus_t2_delta_' + f] = df['t1_league_delta_' + f] - df['t2_league
```

In [ ]:
```
df.to_csv('data/team/aggregates/Boxes_GameDifferences_and_RunningSeasonAverages_and
```