Team Boxscore Scraping

FULLY WORKING 12.29

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.ticker as mtick
import sqlite3
import seaborn as sns
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from bs4 import BeautifulSoup
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import requests
import shutil
import datetime
from scipy.stats import norm
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder
import os
import winsound
os.chdir('C:\\Users\\Travis\\OneDrive\\Data Science\\Personal_Projects\\Sports\\NBA
import warnings
warnings.filterwarnings('ignore')

from selenium.common.exceptions import WebDriverException
```

In [ ]:
```
# check latest date in boxscores file
check_df = pd.read_csv('data/team/aggregates/Both_Team_Boxscores_ALL_with_Trad_Difs
check_df['trad_gamedate'] = pd.to_datetime(check_df['trad_gamedate'])
check_df['trad_gamedate'] = check_df['trad_gamedate'].dt.date
latest_date = check_df['trad_gamedate'].max()
latest_date
```

Out[ ]:
```
datetime.date(2023, 1, 1)
```

In [ ]:
```
# drop duplicates
check_df = check_df.drop_duplicates()
```

In [ ]:
```
# if no driver is open, open one

driver = None

def check_and_open_driver(driver):
  # if driver is not open
    if driver == None:
        driver = webdriver.Chrome()
    return driver
```

In [ ]:
```
driver = check_and_open_driver(driver)
```

In [ ]:
```
def replace_name_values(filename):
        # replace values with dashes for compatibility
    filename = filename.replace('%','_')
    filename = filename.replace('=','_')
    filename = filename.replace('?','_')
    filename = filename.replace('&','_')
    filename = filename.replace('20Season_','')
    filename = filename.replace('_20Season','')
    filename = filename.replace('SeasonType_','')
    filename = filename.replace('sort_gdate_dir_-1_','')
    filename = filename.replace('SeasonYear_','')
    return filename
```

In [ ]:
```
def grab_team_data(url_list, file_folder):
        i = 0
        for u in url_list:
                try:
                        driver.get(u)
                        src = driver.page_source
                        parser = BeautifulSoup(src, "lxml")
                        table = parser.find("table", attrs = {"class":"Crom_table__
                except:
                        print(f'{u} Failed to load')
                        continue
                try:
                        headers = table.findAll('th')
                        headerlist = [h.text.strip() for h in headers[0:]]
                        row_names = table.findAll('a')
                        row_list = [b.text.strip() for b in row_names[0:]]
                        rows = table.findAll('tr')[0:]
                except:
                        print(f'{u} Failed to load')
                        continue

                # get the data
                player_stats = [[td.getText().strip() for td in rows[i].findAll('td
                tot_cols = len(player_stats[1])                          #set the
                headerlist = headerlist[:tot_cols]
                stats = pd.DataFrame(player_stats, columns = headerlist)

                # name file
                filename = file_folder + str(u[31:]).replace('/', '_') + '.csv'
                filename = replace_name_values(filename)

                # save to csv
                pd.DataFrame.to_csv(stats, filename)
                i += 1
                lu = len(url_list)
                print(f'{filename} Completed Successfully! {i} / {lu} Complete!')
        winsound.Beep(523, 500)
```

In [ ]:
```
def team_data_filename_transformer(url):
    filename = str(url[31:]).replace('/', '_') + '.csv'
    filename = replace_name_values(filename)
    return filename
```

In [ ]:

```
def grab_box_scores(url_list, file_folder):

        # Function to scrape box scores of individual games

        i = 0
        for u in url_list:
                driver.get(u)

                # click "all pages"
                xpath_all = '//*[@id="__next"]/div[2]/div[2]/div[3]/section[2]/div/
                elem = WebDriverWait(driver, 30).until(EC.presence_of_element_locat
                driver.find_element(by=By.XPATH, value=xpath_all).click()
                src = driver.page_source
                parser = BeautifulSoup(src, "lxml")

                # Find Table and scrape data
                table = parser.find("table", attrs = {"class":"Crom_table__p1iZz"})
                headers = table.findAll('th')
                headerlist = [h.text.strip() for h in headers[0:]]
                row_names = table.findAll('a')
                row_list = [b.text.strip() for b in row_names[0:]]
                rows = table.findAll('tr')[0:]
                player_stats = [[td.getText().strip() for td in rows[i].findAll('td
                tot_cols = len(player_stats[2])
                headerlist = headerlist[:tot_cols]
                stats = pd.DataFrame(player_stats, columns = headerlist)

                # File Name
                filename = file_folder + str(u[31:]).replace('/', '_') + '.csv'
                filename = replace_name_values(filename)
                # Save to CSV
                pd.DataFrame.to_csv(stats, filename)
                i += 1
                lu = len(url_list)
                print(f'{filename} Completed Successfully! {i} / {lu} Complete!')
```

In [ ]:

```
def append_the_data(folder, data_prefix, filename_selector):

    # Appending data together via folder and/or file name

    path = folder
    p = os.listdir(path)
    pf = pd.DataFrame(p)

    # first filter
    pf_reg = pf.loc[pf[0].astype(str).str.contains(filename_selector)]

    appended_data = pd.DataFrame()

    for file in pf_reg[0]:
        data = pd.read_csv(folder + '/' + file)
        # if Season is in the data, drop it
        if 'Season' in data.columns:
            data = data.drop(columns = ['Season'])

        # for some reason, this was the only way I could get the data to append cor
        if '1996' in file:
            data['Season'] = 1996
        elif '1997' in file:
            data['Season'] = 1997
        elif '1998' in file:
            data['Season'] = 1998
        elif '1999' in file:
            data['Season'] = 1999
        elif '2000' in file:
            data['Season'] = 2000
        elif '2001' in file:
            data['Season'] = 2001
        elif '2002' in file:
            data['Season'] = 2002
        elif '2003' in file:
            data['Season'] = 2003
        elif '2004' in file:
            data['Season'] = 2004
        elif '2005' in file:
            data['Season'] = 2005
        elif '2006' in file:
            data['Season'] = 2006
        elif '2007' in file:
            data['Season'] = 2007
        elif '2008' in file:
            data['Season'] = 2008
        elif '2009' in file:
            data['Season'] = 2009
        elif '2010' in file:
            data['Season'] = 2010
        elif '2011' in file:
            data['Season'] = 2011
        elif '2012' in file:
            data['Season'] = 2012
        elif '2013' in file:
            data['Season'] = 2013
```

```
        elif '2014' in file:
            data['Season'] = 2014
        elif '2015' in file:
            data['Season'] = 2015
        elif '2016' in file:
            data['Season'] = 2016
        elif '2017' in file:
            data['Season'] = 2017
        elif '2018' in file:
            data['Season'] = 2018
        elif '2019' in file:
            data['Season'] = 2019
        elif '2020' in file:
            data['Season'] = 2020
        elif '2021' in file:
            data['Season'] = 2021
        elif '2022' in file:
            data['Season'] = 2022

        data['season_type'] = np.where('Reg' in file, 'Regular', 'Playoffs')
        # add prefix to columns
        data = data.add_prefix(data_prefix)
        data.columns = data.columns.str.lower()

        # save data to appended data
        appended_data = pd.DataFrame.append(appended_data, data)


    appended_data = pd.concat(appended_data)

    return appended_data
```

In [ ]:

```
def quintuple_merge(df1, df2, df3, df4, df5, prefix1, prefix2, prefix3, prefix4, pr

    # Merge 5 dataframes together


    merge_cols1 = [(prefix1 + 'team'), (prefix1 + 'matchup'), (prefix1 + 'gamedate'
    merge_cols2 = [(prefix2 + 'team'), (prefix2 + 'matchup'), (prefix2 + 'gamedate'
    merge_cols3 = [(prefix3 + 'team'), (prefix3 + 'matchup'), (prefix3 + 'gamedate'
    merge_cols4 = [(prefix4 + 'team'), (prefix4 + 'matchup'), (prefix4 + 'gamedate'
    merge_cols5 = [(prefix5 + 'team'), (prefix5 + 'matchup'), (prefix5 + 'gamedate'

    df = pd.merge(df1, df2, left_on= merge_cols1, right_on = merge_cols2, how='left
    df = pd.merge(df,df3, left_on= merge_cols1, right_on = merge_cols3, how='left')
    df = pd.merge(df,df4, left_on= merge_cols1, right_on = merge_cols4, how='left')
    df = pd.merge(df,df5, left_on= merge_cols1, right_on = merge_cols5, how='left')

    return df
```

In [ ]:
```
def fix_box_url(url):
    name = team_data_filename_transformer(url)
    new_name = name.replace('Reg_Season_Season', 'Reg')
    new_name = new_name.replace('Playoffs_Season_Season', 'Playoffs')
    new_name = new_name.replace('SeasonYear', 'Yr')
    new_name = new_name.replace('SeasonType', '')
    return new_name
```

In [ ]:
```
def post_dl_fix(filename):
    new_name = filename.replace('Reg_Season_Season', 'Reg')
    new_name = new_name.replace('Playoffs_Season_Season', 'Playoffs')
    new_name = new_name.replace('SeasonYear', 'Yr')
    new_name = new_name.replace('SeasonType', '')
    return new_name
```

Add 2022 Season Update

In [ ]:
```
override = True # set to True to override existing files
```

In [ ]:
```
new_boxes = ['https://www.nba.com/stats/teams/boxscores-traditional?SeasonType=Regu
             'https://www.nba.com/stats/teams/boxscores-advanced?SeasonType=Regular%
             'https://www.nba.com/stats/teams/boxscores-four-factors?SeasonType=Regu
             'https://www.nba.com/stats/teams/boxscores-misc?SeasonType=Regular%20Se
             'https://www.nba.com/stats/teams/boxscores-scoring?SeasonType=Regular%2


today = datetime.date.today()
yesterday = today - datetime.timedelta(days=1)

if latest_date < yesterday or override == True:
    # get the data
    driver = webdriver.Chrome()
    grab_box_scores(new_boxes, 'data/team/team_boxscores/')
else:
    print('yesterday is not later than latest date, so do not run the code')
```

```
data/team/team_boxscores/_boxscores-traditional_Regular_Season_2022-23_2022-23.csv
Completed Successfully! 1 / 5 Complete!
data/team/team_boxscores/_boxscores-advanced_Regular_Season_2022-23_2022-23.csv Com
pleted Successfully! 2 / 5 Complete!
data/team/team_boxscores/_boxscores-four-factors_Regular_Season_2022-23_2022-23.csv
Completed Successfully! 3 / 5 Complete!
data/team/team_boxscores/_boxscores-misc_Regular_Season_2022-23_2022-23.csv Complet
ed Successfully! 4 / 5 Complete!
data/team/team_boxscores/_boxscores-scoring_Regular_Season_2022-23_2022-23.csv Comp
leted Successfully! 5 / 5 Complete!
```

Append

In [ ]:
```
def append_the_data(folder, data_prefix, filename_selector):
    # Appending data together via folder and/or file name

    path = folder
    p = os.listdir(path)
    pf = pd.DataFrame(p)


    # filter for files that contain the filename_selector
    pf_reg = pf.loc[pf[0].astype(str).str.contains(filename_selector)]

    appended_data = []
    for file in pf_reg[0]:
        data = pd.read_csv(folder + '/' + file)
        # if "Season" a column, drop it
        if 'Season' in data.columns:
            data = data.drop(columns = ['Season'])


        data['season'] = file[(file.find('20')):(file.find('20'))+4]
        data['season_type'] = np.where('Regular' in file, 'Regular', 'Playoffs')
        # add prefix to columns
        data = data.add_prefix(data_prefix)
        data.columns = data.columns.str.lower()


        if 'trad_gamedate' in data.columns:
            data['trad_gamedate'] = pd.to_datetime(data['trad_gamedate'])
        if 'adv_gamedate' in data.columns:
            data['adv_gamedate'] = pd.to_datetime(data['adv_gamedate'])
        if 'four_gamedate' in data.columns:
            data['four_gamedate'] = pd.to_datetime(data['four_gamedate'])
        if 'misc_gamedate' in data.columns:
            data['misc_gamedate'] = pd.to_datetime(data['misc_gamedate'])
        if 'scor_gamedate' in data.columns:
            data['scor_gamedate'] = pd.to_datetime(data['scor_gamedate'])

        appended_data.append(data)

    appended_data = pd.concat(appended_data)
    return appended_data
```

In [ ]:
```
trad_files_reg = append_the_data('data/team/team_boxscores/', 'trad_', 'traditional
trad_files_play = append_the_data('data/team/team_boxscores/', 'trad_', 'traditiona
```

In [ ]:
```
adv_files_reg = append_the_data('data/team/team_boxscores/', 'adv_', 'advanced_Regu
adv_files_play = append_the_data('data/team/team_boxscores/', 'adv_', 'advanced_Pla

four_files_reg = append_the_data('data/team/team_boxscores/', 'four_', 'four-factor
four_files_play = append_the_data('data/team/team_boxscores/', 'four_', 'four-facto

misc_files_reg = append_the_data('data/team/team_boxscores/', 'misc_', 'misc_Regula
misc_files_play = append_the_data('data/team/team_boxscores/', 'misc_', 'misc_Playo

score_files_reg = append_the_data('data/team/team_boxscores/', 'score_', 'scoring_R
score_files_play = append_the_data('data/team/team_boxscores/', 'score_', 'scoring_
```

In [ ]:
```
trad_files_total = pd.concat([trad_files_reg, trad_files_play])
adv_files_total = pd.concat([adv_files_reg, adv_files_play])
four_files_total = pd.concat([four_files_reg, four_files_play])
misc_files_total = pd.concat([misc_files_reg, misc_files_play])
score_files_total = pd.concat([score_files_reg, score_files_play])
```

In [ ]:
```
# Fix Trad Dtypes

# get any columns with % in them
percent_cols = trad_files_total.columns[trad_files_total.columns.str.contains('%')]

# fix dtypes

for col in percent_cols:
    trad_files_total[col] = trad_files_total[col].astype(str).str.replace('%', '')
    trad_files_total[col] = trad_files_total[col].astype(float)
    trad_files_total[col] = trad_files_total[col] / 100

# change matchup and gamedate columns to one word
trad_files_total = trad_files_total.rename(columns={'trad_match up': 'trad_matchup'
                                                    'trad_match up': 'trad_matchup',
```

In [ ]:
```
# Fix Adv Dtypes

# get any columns with % in them
percent_cols = adv_files_total.columns[adv_files_total.columns.str.contains('%')]

# fix dtypes

for col in percent_cols:
    adv_files_total[col] = adv_files_total[col].astype(str).str.replace('%', '')
    adv_files_total[col] = adv_files_total[col].astype(float)
    adv_files_total[col] = adv_files_total[col] / 100

# change matchup and gamedate columns to one word
adv_files_total = adv_files_total.rename(columns={'adv_match\xa0up': 'adv_matchup',
                                                  'adv_match up': 'adv_matchup', 'adv
```

In [ ]:
```
# Fix Four Dtypes

# get any columns with % in them
percent_cols = four_files_total.columns[four_files_total.columns.str.contains('%')]

# fix dtypes

for col in percent_cols:
    four_files_total[col] = four_files_total[col].astype(str).str.replace('%', '')
    four_files_total[col] = four_files_total[col].astype(float)
    four_files_total[col] = four_files_total[col] / 100

# change matchup and gamedate columns to one word
four_files_total = four_files_total.rename(columns={'four_match up': 'four_matchup'
                                                    'four_match\xa0up': 'four_matchu
```

In [ ]:
```
# Fix Misc Dtypes

# get any columns with % in them
percent_cols = misc_files_total.columns[misc_files_total.columns.str.contains('%')]

# fix dtypes

for col in percent_cols:
    misc_files_total[col] = misc_files_total[col].astype(str).str.replace('%', '')
    misc_files_total[col] = misc_files_total[col].astype(float)
    misc_files_total[col] = misc_files_total[col] / 100

# change matchup and gamedate columns to one word
misc_files_total = misc_files_total.rename(columns={'misc_match up': 'misc_matchup'
                                                    'misc_match\xa0up': 'misc_match
```

In [ ]:
```
# Fix Score Dtypes

# get any columns with % in them
percent_cols = score_files_total.columns[score_files_total.columns.str.contains('%'

# fix dtypes

for col in percent_cols:
    score_files_total[col] = score_files_total[col].astype(str).str.replace('%', ''
    score_files_total[col] = score_files_total[col].astype(float)
    score_files_total[col] = score_files_total[col] / 100

# change matchup and gamedate columns to one word
score_files_total = score_files_total.rename(columns={'score_match up': 'score_matc
                                                      'score_match\xa0up': 'score_m
```

In [ ]:
```
# print df shapes
print(trad_files_total.shape)
print(adv_files_total.shape)
print(four_files_total.shape)
print(misc_files_total.shape)
print(score_files_total.shape)
```

```
(66664, 27)
(66664, 22)
(66664, 16)
(66664, 16)
(66664, 23)
```

In [ ]:
```
# filter out all years before 2010
# drop nas in gamedate column
trad_files_total = trad_files_total.dropna(subset=['trad_gamedate'])
adv_files_total = adv_files_total.dropna(subset=['adv_gamedate'])
four_files_total = four_files_total.dropna(subset=['four_gamedate'])
misc_files_total = misc_files_total.dropna(subset=['misc_gamedate'])
score_files_total = score_files_total.dropna(subset=['score_gamedate'])

trad_files_total = trad_files_total[trad_files_total['trad_gamedate'].str.contains(
adv_files_total = adv_files_total[adv_files_total['adv_gamedate'].str.contains('201
four_files_total = four_files_total[four_files_total['four_gamedate'].str.contains(
misc_files_total = misc_files_total[misc_files_total['misc_gamedate'].str.contains(
score_files_total = score_files_total[score_files_total['score_gamedate'].str.conta
```

In [ ]:
```
team_boxes_df1 = pd.merge(trad_files_total, adv_files_total,
                          left_on= ['trad_matchup', 'trad_gamedate', 'trad_team']
                          right_on= ['adv_matchup', 'adv_gamedate', 'adv_team'],
                          how='left')

team_boxes_df2 = pd.merge(team_boxes_df1, four_files_total,
                          left_on= ['trad_matchup', 'trad_gamedate', 'trad_team']
                          right_on= ['four_matchup', 'four_gamedate', 'four_team'
                          how='left')

team_boxes_df3 = pd.merge(team_boxes_df2, misc_files_total,
                          left_on= ['trad_matchup', 'trad_gamedate', 'trad_team']
                          right_on= ['misc_matchup', 'misc_gamedate', 'misc_team'
                          how='left')

team_boxes_df4 = pd.merge(team_boxes_df3, score_files_total,
                          left_on= ['trad_matchup', 'trad_gamedate', 'trad_team']
                          right_on= ['score_matchup', 'score_gamedate', 'score_te
                          how='left')

team_boxes_df4.to_csv('data/team/aggregates/All_Boxes.csv', index=False)
```

In [ ]:
```
team_boxes_df4.shape
```

Out[ ]:
```
(34680, 104)
```

In [ ]:
```
all_boxes = team_boxes_df4
```

In [ ]:
```
all_boxes_copy = all_boxes.copy()
all_boxes_copy = all_boxes_copy.add_prefix('tm2__')
```

In [ ]:
```
print(f' all_boxes shape: {all_boxes.shape}, all_boxes_copy shape: {all_boxes_copy.
```

```
 all_boxes shape: (34680, 104), all_boxes_copy shape: (34680, 104)
```

In [ ]:
```
# add game id and second team to all_boxes
all_boxes['trad_matchup'] = all_boxes['trad_matchup'].astype(str)
all_boxes['team_2'] = all_boxes['trad_matchup'].str[-3:]
all_boxes.head(3)
```

Out[ ]:

	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_pts	trad_fgı
0	1	CHA	CHA vs. CHI	04/14/2010	L	48.0	89.0	32
1	2	MEM	MEM @ OKC	04/14/2010	L	48.0	105.0	39
2	3	LAC	LAC vs. LAL	04/14/2010	W	48.0	107.0	42

3 rows × 105 columns

In [ ]:
```
all_boxes['game_id'] = np.where(all_boxes['trad_matchup'].str.contains('vs'),
                                all_boxes['trad_matchup'].str[-3:] + ' @ ' +
                                all_boxes['trad_matchup'].str[0:3] + '_' +
                                all_boxes['trad_gamedate'].astype(str),
                                all_boxes['trad_matchup'].astype(str) +  '_' + all_
```

In [ ]:
```
# create Game_id
all_boxes_copy['game_id'] = np.where(all_boxes_copy['tm2__trad_matchup'].str.contai
                                all_boxes_copy['tm2__trad_matchup'].str[-3:] + ' @
                                all_boxes_copy['tm2__trad_matchup'].str[0:3] + '_'
                                all_boxes_copy['tm2__trad_gamedate'].astype(str),
                                all_boxes_copy['tm2__trad_matchup'].astype(str) +
all_boxes_copy.head(3)
```

Out[ ]:

	tm2__trad_unnamed: 0	tm2__trad_team	tm2__trad_matchup	tm2__trad_gamedate	tm2__trad_w/l
0	1	CHA	CHA vs. CHI	04/14/2010	L
1	2	MEM	MEM @ OKC	04/14/2010	L
2	3	LAC	LAC vs. LAL	04/14/2010	W

3 rows × 105 columns

In [ ]:
```
check1 = all_boxes[['game_id', 'team_2']]
check2 = all_boxes_copy[['game_id', 'tm2__trad_team']]
```

In [ ]:
```
all_boxes.game_id = all_boxes.game_id.astype(str)
all_boxes_copy.game_id = all_boxes_copy.game_id.astype(str)
all_boxes_copy.tm2__=team = all_boxes_copy.tm2__trad_team.astype(str)
all_boxes.team_2 = all_boxes.team_2.astype(str)
```

In [ ]:
```
# merge the dfs
boxes_both_teams = pd.merge(all_boxes, all_boxes_copy,
                    left_on = ['game_id', 'team_2'],
                    right_on = ['game_id', 'tm2__trad_team'],
                    how = 'left')
boxes_both_teams
```

Out[ ]:

	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_pts	tra
0	1	CHA	CHA vs. CHI	04/14/2010	L	48.0	89.0	
1	2	MEM	MEM @ OKC	04/14/2010	L	48.0	105.0	
2	3	LAC	LAC vs. LAL	04/14/2010	W	48.0	107.0	
3	4	PHI	PHI @ ORL	04/14/2010	L	48.0	111.0	
4	5	SAS	SAS @ DAL	04/14/2010	L	48.0	89.0	
...	...	...	...	...	...	...	...	
82311	170	DEN	DEN @ GSW	04/16/2022	L	48.0	107.0	
82312	171	MIN	MIN @ MEM	04/16/2022	W	48.0	130.0	
82313	172	UTA	UTA @ DAL	04/16/2022	W	48.0	99.0	
82314	173	MEM	MEM vs. MIN	04/16/2022	L	48.0	117.0	
82315	174	PHI	PHI vs. TOR	04/16/2022	W	48.0	131.0	

82316 rows × 210 columns

In [ ]:
```
boxes_both_teams['trad_gamedate'] = pd.to_datetime(boxes_both_teams['trad_gamedate'
boxes_both_teams.sort_values(by=['trad_gamedate'], inplace=True)
boxes_both_teams
```

Out[ ]:

	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_pts	tra
1513	1514	LAL	LAL vs. SAC	2010-01-01	W	48.0	109.0	
1512	1513	ATL	ATL vs. NYK	2010-01-01	L	53.0	108.0	
1511	1512	ORL	ORL @ MIN	2010-01-01	W	48.0	106.0	
1510	1511	NYK	NYK @ ATL	2010-01-01	W	53.0	112.0	
1509	1510	MIN	MIN vs. ORL	2010-01-01	L	48.0	94.0	
...	...	...	...	...	...	...	...	
54496	5	BOS	BOS @ DEN	2023-01-01	L	48.0	111.0	
54495	4	WAS	WAS @ MIL	2023-01-01	W	48.0	118.0	
54494	3	DEN	DEN vs. BOS	2023-01-01	W	48.0	123.0	
54493	2	SAC	SAC @ MEM	2023-01-01	L	48.0	108.0	
54492	1	MIL	MIL vs. WAS	2023-01-01	L	48.0	95.0	

82316 rows × 210 columns

In [ ]:
```
boxes_both_teams = boxes_both_teams.drop_duplicates()
boxes_both_teams2 = boxes_both_teams.dropna(subset = ['tm2__score_season'])
```

In [ ]:
```
boxes_both_teams2
```

Out[ ]:

	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_pts	tra
1513	1514	LAL	LAL vs. SAC	2010-01-01	W	48.0	109.0	
1512	1513	ATL	ATL vs. NYK	2010-01-01	L	53.0	108.0	
1511	1512	ORL	ORL @ MIN	2010-01-01	W	48.0	106.0	
1510	1511	NYK	NYK @ ATL	2010-01-01	W	53.0	112.0	
1509	1510	MIN	MIN vs. ORL	2010-01-01	L	48.0	94.0	
...	...	...	...	...	...	...	...	
54496	5	BOS	BOS @ DEN	2023-01-01	L	48.0	111.0	
54495	4	WAS	WAS @ MIL	2023-01-01	W	48.0	118.0	
54494	3	DEN	DEN vs. BOS	2023-01-01	W	48.0	123.0	
54493	2	SAC	SAC @ MEM	2023-01-01	L	48.0	108.0	
54492	1	MIL	MIL vs. WAS	2023-01-01	L	48.0	95.0	

82316 rows × 210 columns

In [ ]:
```
rbbt = boxes_both_teams2
rbbt = rbbt.drop_duplicates()
```

In [ ]:
```
rbbt.to_csv('data/team/aggregates/both_team_boxscores_ALL.csv')
```

Add Initial Features

In [ ]:
```
rbbt = rbbt.assign(t1_t2_pts = rbbt['trad_pts'] - rbbt['tm2__trad_pts'])
rbbt = rbbt.assign(t1_t2_fgm = rbbt['trad_fgm'] - rbbt['tm2__trad_fgm'])
rbbt = rbbt.assign(t1_t2_fga = rbbt['trad_fga'] - rbbt['tm2__trad_fga'])
rbbt = rbbt.assign(t1_t2_fg_percent = rbbt['trad_fg%'] - rbbt['tm2__trad_fg%'])
rbbt = rbbt.assign(t1_t2_3pm = rbbt['trad_3pm'] - rbbt['tm2__trad_3pm'])
rbbt = rbbt.assign(t1_t2_3pa = rbbt['trad_3pa'] - rbbt['tm2__trad_3pa'])
rbbt = rbbt.assign(t1_t2_3p_percent = rbbt['trad_3p%'] - rbbt['tm2__trad_3p%'])
rbbt = rbbt.assign(t1_t2_ftm = rbbt['trad_ftm'] - rbbt['tm2__trad_ftm'])
rbbt = rbbt.assign(t1_t2_fta = rbbt['trad_fta'] - rbbt['tm2__trad_fta'])
rbbt = rbbt.assign(t1_t2_ft_percent = rbbt['trad_ft%'] - rbbt['tm2__trad_ft%'])
rbbt = rbbt.assign(t1_t2_oreb = rbbt['trad_oreb'] - rbbt['tm2__trad_oreb'])
rbbt = rbbt.assign(t1_t2_dreb = rbbt['trad_dreb'] - rbbt['tm2__trad_dreb'])
rbbt = rbbt.assign(t1_t2_reb = rbbt['trad_reb'] - rbbt['tm2__trad_reb'])
rbbt = rbbt.assign(t1_t2_ast = rbbt['trad_ast'] - rbbt['tm2__trad_ast'])
rbbt = rbbt.assign(t1_t2_stl = rbbt['trad_stl'] - rbbt['tm2__trad_stl'])
rbbt = rbbt.assign(t1_t2_blk = rbbt['trad_blk'] - rbbt['tm2__trad_blk'])
rbbt = rbbt.assign(t1_t2_tov = rbbt['trad_tov'] - rbbt['tm2__trad_tov'])
rbbt = rbbt.assign(t1_t2_pf = rbbt['trad_pf'] - rbbt['tm2__trad_pf'])
```

In [ ]:
```
rbbt
```

Out[ ]:

	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_pts	tra
1513	1514	LAL	LAL vs. SAC	2010-01-01	W	48.0	109.0	
1512	1513	ATL	ATL vs. NYK	2010-01-01	L	53.0	108.0	
1511	1512	ORL	ORL @ MIN	2010-01-01	W	48.0	106.0	
1510	1511	NYK	NYK @ ATL	2010-01-01	W	53.0	112.0	
1509	1510	MIN	MIN vs. ORL	2010-01-01	L	48.0	94.0	
...	...	...	...	...	...	...	...	
54496	5	BOS	BOS @ DEN	2023-01-01	L	48.0	111.0	
54495	4	WAS	WAS @ MIL	2023-01-01	W	48.0	118.0	
54494	3	DEN	DEN vs. BOS	2023-01-01	W	48.0	123.0	
54493	2	SAC	SAC @ MEM	2023-01-01	L	48.0	108.0	
54492	1	MIL	MIL vs. WAS	2023-01-01	L	48.0	95.0	

82316 rows × 228 columns

In [ ]:
```
new_cols = ['t1_t2_pts', 't1_t2_fgm', 't1_t2_fga', 't1_t2_fg_percent', 't1_t2_3pm',
```

In [ ]:
```
rbbt['who_wins'] = np.where(rbbt['trad_w/l'] == 'W', 't1', 't2')
```

In [ ]:
```
rbbt = rbbt.drop_duplicates()
```

In [ ]:
```
rbbt.to_csv('data/team/aggregates/Both_Team_Boxscores_ALL_with_Trad_Difs.csv')
```

In [ ]:
```
rbbt
```

Out[ ]:

	trad_unnamed: 0	trad_team	trad_matchup	trad_gamedate	trad_w/l	trad_min	trad_pts	tra
1513	1514	LAL	LAL vs. SAC	2010-01-01	W	48.0	109.0	
1512	1513	ATL	ATL vs. NYK	2010-01-01	L	53.0	108.0	
1511	1512	ORL	ORL @ MIN	2010-01-01	W	48.0	106.0	
1510	1511	NYK	NYK @ ATL	2010-01-01	W	53.0	112.0	
1509	1510	MIN	MIN vs. ORL	2010-01-01	L	48.0	94.0	
...	...	...	...	...	...	...	...	
54496	5	BOS	BOS @ DEN	2023-01-01	L	48.0	111.0	
54495	4	WAS	WAS @ MIL	2023-01-01	W	48.0	118.0	
54494	3	DEN	DEN vs. BOS	2023-01-01	W	48.0	123.0	
54493	2	SAC	SAC @ MEM	2023-01-01	L	48.0	108.0	
54492	1	MIL	MIL vs. WAS	2023-01-01	L	48.0	95.0	

82316 rows × 229 columns

In [ ]: