Use machine learning to predict All NBA Team, Data Collection

February 6, 2021

1 Scraping all nba team member

[221]:
```python
from bs4 import BeautifulSoup
import pandas as pd
import requests
import numpy as np

# Define a function to change the season name for a easier prediction
def change_season_name(row):
    if pd.notnull(row['Season']):
        return row['Season'][:4]
    return row['Season']

def change_tm(row):
    if row['Tm'] == '1st':
        return 1
    elif row['Tm'] == '2nd':
        return 2
    else:
        return 3

def change_name_C(row):
    return row['C'][:-2]

def change_name_PF(row):
    return row['PF'][:-2]

def change_name_SF(row):
    return row['SF'][:-2]

def change_name_SG(row):
    return row['SG'][:-2]

def change_name_PG(row):
    return row['PG'][:-2]
```

[806]:
```
# 1. Leave all non-null data
# 2. Select the data from 1988-1989 season, since it is the first season having␣
↪3 all nba team
# 3. Rename the column by position on the court
# 4. Change Season and Tm to int for future visualization
# 5. Delete the position of players after their names

def data_cleaning():
    html = requests.get("https://www.basketball-reference.com/awards/all_league.
↪html")
    soup = BeautifulSoup(html.content, 'html.parser')
    t = soup.findAll('table')

    df1 = pd.read_html(str(t))[0][:99]

    df1 = df1.rename(columns = {'Unnamed: 3': "C", "Unnamed: 4": "PF", "Unnamed:
↪ 5": "SF", "Unnamed: 6": "SG", "Unnamed: 7": "PG"})
    df1['Season'] = df1.apply(change_season_name, axis = 1)
    df1['Tm'] = df1.apply(change_tm, axis = 1)
    df1 = df1[pd.notnull(df1.Season)][['Season', 'Tm', 'C', 'PF', 'SF', 'SG',␣
↪'PG']]
    df1['C'] = df1.apply(change_name_C, axis = 1)
    df1['PF'] = df1.apply(change_name_PF, axis = 1)
    df1['SF'] = df1.apply(change_name_SF, axis = 1)
    df1['SG'] = df1.apply(change_name_SG, axis = 1)
    df1['PG'] = df1.apply(change_name_PG, axis = 1)

    return df1


def put_in_dict1(season_player_dict, season, player_list):
    season = int(season)
    if season not in season_player_dict:
        season_player_dict[season] = player_list
    else:
        season_player_dict[season] = season_player_dict[season] + player_list

def put_in_dict2(team_dict, name, season):
    season = int(season)
    if name not in team_dict:
        team_dict[name] = [season]
    else:
        team_dict[name].append(season)

def load_nba_team():
    data = data_cleaning()
    nba_team_year_dict = {}
    season_nba_team_dict = {}

    for i, row in data.iterrows():
        season = row['Season']
        team = row['Tm']
        C = row['C']
        PF = row['PF']
        SF = row['SF']
        SG = row['SG']
        PG = row['PG']
        season_player_list = [C, PF, SF, SG, PG]

        put_in_dict1(season_nba_team_dict, season, season_player_list)

        put_in_dict2(nba_team_year_dict, C, season)
        put_in_dict2(nba_team_year_dict, PF, season)
        put_in_dict2(nba_team_year_dict, SF, season)
        put_in_dict2(nba_team_year_dict, SG, season)
        put_in_dict2(nba_team_year_dict, PG, season)

    return nba_team_year_dict, season_nba_team_dict
```

[807]:
```
if __name__ == '__main__':
    nba_team_year_dict, season_nba_team_dict = load_nba_team()
```

2 Scraping statistics for these players

[ ]:
```
from selenium import webdriver
from selenium.webdriver.support.ui import Select
import pickle
import time

driver = webdriver.Chrome('/Users/yhschan/Desktop/Data Science Project/
 ↪chromedriver')

START_YEAR, END_YEAR = 1996, 2020

TIME_DELAY_TEAMS = 5
TIME_DELAY_PLAYERS = 12

team_mapper_dict = {
    'Atlanta Hawks' : 'ATL',
    'Boston Celtics' : 'BOS',
    'Charlotte Hornets Old' : 'CHH', # deprecated
    'Chicago Bulls' : 'CHI',
    'Cleveland Cavaliers' : 'CLE',
    'Dallas Mavericks' : 'DAL',
    'Denver Nuggets' : 'DEN',
    'Detroit Pistons' : 'DET',
    'Golden State Warriors' : 'GSW',
    'Houston Rockets' : 'HOU',
    'Indiana Pacers' : 'IND',
    'Los Angeles Clippers' : 'LAC', # deprecated
    'LA Clippers' : 'LAC',
    'Los Angeles Lakers' : 'LAL',
    'Miami Heat' : 'MIA',
    'Milwaukee Bucks' : 'MIL',
    'Minnesota Timberwolves' : 'MIN',
    'New Jersey Nets' : 'NJN', # deprecated
    'New York Knicks' : 'NYK',
    'Orlando Magic' : 'ORL',
    'Philadelphia 76ers' : 'PHI',
    'Phoenix Suns' : 'PHX',
    'Portland Trail Blazers' : 'POR',
    'Sacramento Kings' : 'SAC',
    'San Antonio Spurs' : 'SAS',
    'Seattle SuperSonics' : 'SEA', # deprecated
    'Toronto Raptors' : 'TOR',
    'Utah Jazz' : 'UTA',
    'Vancouver Grizzlies' : 'VAN', # deprecated
    'Washington Bullets' : 'WAS', # deprecated
    'Washington Wizards' : 'WAS',
    'Memphis Grizzlies' : 'MEM',
    'New Orleans Hornets' : 'NOH', # deprecated
    'Charlotte Bobcats' : 'CHA', # deprecated
    'New Orleans/Oklahoma City Hornets' : 'NOK', # deprecated
    'Oklahoma City Thunder' : 'OKC',
    'Brooklyn Nets' : 'BKN',
    'Charlotte Hornets New' : 'CHA',
    'New Orleans Pelicans' : 'NOP'
}

def change_team(row):
    row['TEAM'] = team_mapper_dict[row['TEAM']]
    return row['TEAM']

def adjust_hornets(row):
    if row['TEAM'] == 'Charlotte Hornets':
        return 'Charlotte Hornets Old' if row['Year'] <= 2001 else 'Charlotte␣
↪Hornets New'
    return row['TEAM']

def is_selected(row):
    year = row['Year']
    player =  row['PLAYER']
    if player in season_nba_team_dict[year]:
        return 1
    return 0

def is_selected_last_year(row):
    year = row['Year']-1
    player = row['PLAYER']
    if player in season_nba_team_dict[year]:
        return 1
    return 0


# maps year to average league pace
html = requests.get('https://www.basketball-reference.com/leagues/
 ↪NBA_stats_per_game.html')
s_pace = BeautifulSoup(html.content, 'html.parser')
t_pace = s_pace.find('table')
df_pace = pd.read_html(str(t_pace))[0]
df_pace.columns = df_pace.columns.droplevel()


pace_lookup = {}
for i, row in df_pace.iterrows():
    if pd.notnull(row['Season']) and row['Season'] != 'Season':
        year = int(row['Season'][:4])
        pace_lookup[year] = row['Pace']
    if year == START_YEAR:
        break


# collect training df
df_train_list = []

# Scraping url for each season
for year in range(START_YEAR, END_YEAR):

    season_label = str(year) + '-' + str(year+1)[2:]
    print('Scraping stats.nba.com for {} season...'.format(season_label))

    url_trad_stats = '''https://www.nba.com/stats/players/traditional/?
 ↪sort=PTS&dir=-1
                        &Season={}&SeasonType=Regular%20Season'''.
 ↪format(season_label)
```

```
    url_adv_stats = '''https://www.nba.com/stats/players/advanced/?
↪sort=PTS&dir=-1
                        &Season={}&SeasonType=Regular%20Season'''.
↪format(season_label)

    url_teams = '''https://www.nba.com/stats/teams/traditional/?
↪sort=W_PCT&dir=-1
                    &Season={}&SeasonType=Regular%20Season'''.
↪format(season_label)


    # Get Team rank and name
    driver.get(url_teams)

    time.sleep(TIME_DELAY_TEAMS)

    s_teams = BeautifulSoup(driver.page_source, 'html.parser').find('table')
    df_teams = pd.read_html(str(s_teams))[0]
    df_teams['Year'] = year
    df_teams.rename(columns = {'Unnamed: 0': 'Rank'}, inplace = True)

    df_teams['TEAM'] = df_teams[['TEAM', 'Year']].apply(adjust_hornets, axis =␣
↪1)
    df_teams['TEAM'] = df_teams[['TEAM']].apply(change_team, axis = 1)

    # Get Traditional Stats
    driver.get(url_trad_stats)

    time.sleep(TIME_DELAY_PLAYERS)

    # By js, only 50 players are displayed per page. We need to change this␣
↪using the dropdown select element
    select = Select(driver.find_elements_by_xpath('/html/body/main/div/div/
↪div[2]/div/div/nba-stat-table/div[1]/div/div/select')[0])
    select.select_by_visible_text('All')

    s_traditional = BeautifulSoup(driver.page_source, 'html.parser').
↪find('table')
    df_traditional = pd.read_html(str(s_traditional))[0].
↪dropna(subset=['PLAYER'])
    df_traditional = df_traditional.iloc[:, 1:30]

    # Get Advanced stats
    driver.get(url_adv_stats)

    time.sleep(TIME_DELAY_PLAYERS)
    select = Select(driver.find_elements_by_xpath('/html/body/main/div/div/
↪div[2]/div/div/nba-stat-table/div[1]/div/div/select')[0])
    select.select_by_visible_text('All')

    s_advanced = BeautifulSoup(driver.page_source, 'html.parser').find('table')
    df_advanced = pd.read_html(str(s_advanced))[0].dropna(subset=['PLAYER'])
    df_advanced = df_advanced.iloc[:, 1:24]

    # Merge
    df = df_traditional.merge(df_advanced[['PLAYER', 'OFFRTG', 'DEFRTG',␣
↪'NETRTG', 'TS%', 'USG%', 'PIE']], on = 'PLAYER')

    # stitching it all together
    df['Year'] = year
    df['Avg Pace'] = df['Year'].map(lambda x : pace_lookup[x])
    df = df.merge(df_teams[['TEAM', 'Rank']], on = 'TEAM')
    df['PLAYER'] = df['PLAYER'].map(lambda x : 'Ron Artest' if x == 'Metta␣
↪World Peace' else x)
    df['Selected?'] = df.apply(is_selected, axis = 1)
    df['Selected Last Year?'] = df.apply(is_selected_last_year, axis = 1)

    if year != END_YEAR:
        df_train_list.append(df)

driver.quit()

# Export to csv
pd.concat(df_train_list).to_csv('All_NBA_TEAM_train.csv', index = False)
df.to_csv('ALL_NBA_TEAM_test.csv', index = False)
```