KENPOM SCRAPER

November 11, 2017

1 Project DATA Gathering

The code provided immediately following this is what I used to scrape my data. It's mostly selenium based, and crawls through kenpom.com (which did have all the data for which I was looking) and downloads the elements of the table objects containing player stats. After loggin in, it goes through each team listed for a given year and checks to see if that team participated in the NCAA tournament. If it did, it then downloads the scouting report for the team that year (giving important team statistics), and then iterates through all players listed on the roster and goes to that players' individual page. From there, it grabs the table that pertains to that individual players' game-by-game statistics for the given year, and downloads it.

```
In [4]: import re
        import time
        import requests
        import os
        import pandas as pds
        import html5lib
        from bs4 import BeautifulSoup
        import numpy as np
        from selenium import webdriver
        from selenium.webdriver.common.keys import Keys
        from selenium.common.exceptions import NoSuchElementException
```

```
In [2]: # Ceiling function
        def ceil(n) :
            if int(n) == n :
                return n
            else :
                return int(n)+1

        # Returns a number mod 40, but returns 40 instead of 0
        # This is because of how the xpath for the tables on the homepage of kenpor
        def mod_fix(n,r=40) :
            if n%r == 0 :
                return r
            else :
                return n%r
```

```
# Path to data
path = "../DATA/"

# Homepage for kenpom.com
home_url = "https://kenpom.com/"

def login(browser) :
    # Go to webpage
    try :
        browser.get(home_url)
        # Enter user id
        try :
            usr = browser.find_element_by_name('email')
            usr.clear()
            email = str(input("EMAIL: "))
            usr.send_keys(email)
        except :
            print("Couldn't find email.")
            raise ValueError("Field not found.")
        # Enter password
        try :
            pwd = browser.find_element_by_name('password')
            pwd.clear()
            psswd = str(input("PASSWORD: "))
            pwd.send_keys(psswd)
        except :
            print("Couldn't find password bar.")
            raise ValueError("Field not found.")
        # Click login button
        try :
            login = browser.find_element_by_name('submit')
            login.click()
        except :
            print("Couldn't find login button.")
            raise ValueError("Button not found.")
        # Switch to new page
        browser.switch_to_window(browser.window_handles[-1])
    except :
        print("Couldn't find home page.")
        raise ValueError("Page not found.")
    return browser

def get_scouting_report(browser,team_path) :
    # Get scouting report table
    team_soup = BeautifulSoup(browser.page_source, 'html.parser')
    report = team_soup.find('div', attrs={'id':'report'})
    with open(team_path+'Scouting_Report','w') as outfile :
        outfile.write(str(report))
```

```
def get_players(browser,current_team,team_path,year) :
    # The numbers in this range include all players, but some are missing
    # This will skip the missing ones and grab the ones that do exist.
    for i in range(20) :
        try :
            # Player link by xpath
            player = browser.find_element_by_xpath('//*[@id="player-table"]
            player_name = player.text
            # Open link in new tab
            player.send_keys(Keys.COMMAND + Keys.RETURN)
            # Wait for tab to load
            time.sleep(2)
            # Switch view to new tab
            browser.find_element_by_tag_name('body').send_keys(Keys.CONTRO
            # Switch browser to new tab
            browser.switch_to_window(browser.window_handles[-1])
            # Get player table and save to player file
            player_soup = BeautifulSoup(browser.page_source, 'html.parser')
            player_name = player_soup.find_all('div',attrs={'id' : 'content
            player_path = team_path + player_name
            # Get schedule data
            table = player_soup.find_all('table', attrs={'id':'schedule-tab
            # Check all tables if they're the ones we want
            for i in range(len(table)) :
                # This element appears above the table pertaining to the de
                if browser.find_element_by_xpath('//*[@id="players"]/h3[{}]
                    with open(team_path+player_name+' {}'.format(i),'w') as
                        outfile.write(str(table[i]))
            browser.close()
            browser.switch_to_window(current_team)
        except NoSuchElementException :
            pass


def get_years(years) :
    # For each year of interest
    for year in years :
        time.sleep(1)
        # Get the link for given year
        try :
            link = browser.find_element_by_link_text(year)
            link.click()
            browser.switch_to_window(browser.window_handles[-1])
            # Save page, because we'll be returning to it a lot
            current_year = browser.current_window_handle
            # Create folder for year if not already in existence
            if not os.path.exists(path+year+'/') :
                os.makedirs(path+year+'/')
            # Initialize list of teams
            teams = []
            # Counter for specific xpath
            i = 1
            # 68 teams (round of 64 and first 4 gives 4 additional)
            while len(teams) < 68 and i < 350:
                time.sleep(1)
                try :
                    # Team link by xpath
                    team = browser.find_element_by_xpath('//*[@id="ratings-
                    team_name = team.text
                    # Open team link in new tab
                    team.send_keys(Keys.COMMAND + Keys.RETURN)
                    # Wait for tab to load
                    time.sleep(2)
                    # Switch to new tab
                    browser.find_element_by_tag_name('body').send_keys(Keys
                    browser.switch_to_window(browser.window_handles[-1])
                    current_team = browser.current_window_handle
                    in_tourn = True
                    try :
                        tournament_tag = browser.find_element_by_xpath("//
                    except :
                        in_tourn = False
                    # If in the Tournament, get data
                    if in_tourn :
                        teams.append(team_name)
                        # Create Team Folder
                        team_path = path+year+'/'+teams[-1]+'/'
                        if not os.path.exists(team_path) :
                            os.makedirs(team_path)
                        get_scouting_report(browser,team_path)
                        get_players(browser,current_team,team_path,year)
                    # Close tab
                    browser.close()
                    # Switch back to previous window
                    browser.switch_to_window(current_year)
                except NoSuchElementException :
                    pass
                i = i + 1
        except :
            print("Couldn't get data for {}".format(year))
```

```
In [3]: # Open chrome
        browser = webdriver.Chrome('./chromedriver')
        # Login
        login(browser)
        # Years of interest
        years = ['2013']#['2013', '2014','2015','2016','2017']
        # Get data
        get_years(years)
        # Pause
        time.sleep(2)
        # Close browser
        browser.close()
```

```
EMAIL: darrenlund59@gmail.com
PASSWORD: qwerty12345
2
```

The code below cleans the data. Code is commented for further information. All variables are fully described at https://kenpom.com/blog/help-with-team-page/ .

```
In [ ]: base_path = '../DATA/'
        # Possible table numbers from scraper
        table_numbers = ['0','1','2','3']
        # Walk through scraped data
        for dir_path , dir_name , file_names in os.walk(base_path) :
            for name in file_names :
                # Scouting Report formatting
                if name == 'Scouting_Report' :
                    with open(os.path.join(dir_path,name),'r') as infile :
                        report = infile.read()
                        # Remove extra information
                        report = report[332:358] + report[496:]
                    team = pds.read_html(report)[0]
                    # Rename columns
                    team.columns = ['Category','Offense','Defense','D-I Avg.']
                    # Drop columns without data
                    team = team.dropna(thresh=3)
                    # Fix first row
                    team.set_value(1,'D-I Avg.',team.loc[1]['Defense'])
                    team.set_value(1,'Defense',team.loc[1]['Offense'])
                    team = team.fillna('N/A')
                    # Remove ranking from offense and defense values
                    for ind in team.index :
                        team.set_value(ind,'Offense',(team.loc[ind]['Offense']).sp
                        if team.loc[ind]['Offense'][0] == '+' :
                            team.set_value(ind,'Offense',team.loc[ind]['Offense'][
                        elif team.loc[ind]['Offense'][-1] == r'%' :
                            team.set_value(ind,'Offense',team.loc[ind]['Offense'][
                        elif team.loc[ind]['Offense'][-1] == r'"' :
                            team.set_value(ind,'Offense',team.loc[ind]['Offense'][
                        team.set_value(ind,'Offense',float(team.loc[ind]['Offense']
                        team.set_value(ind,'Defense',(team.loc[ind]['Defense']).spl
                        if team.loc[ind]['Defense'][-1] == r'%' :
                            team.set_value(ind,'Defense',team.loc[ind]['Defense'][-
                        elif team.loc[ind]['Defense'][-1] == r'"' :
                            team.set_value(ind,'Defense',team.loc[ind]['Defense'][-
                    # Save
                    team.to_csv(os.path.join(dir_path,name) + '_csv')
                elif name[-1] in table_numbers :
                    with open(os.path.join(dir_path,name),'r') as infile :
                        table = infile.read()
                        # Remove empty superfluous <tbody> attribute
                        table = table[:28] + table[44:]
                    # Read in html
                    player = pds.read_html(table)[0]
                    # Replace unnamed columns and filter out unnecessary data
                    new_columns = list(player.columns)
                    new_columns[1] = 'Date'
                    new_columns[5] = 'OTs'
                    new_columns[7] = 'Conference'
                    player.columns = new_columns
                    player = player.filter(['Date','Opponent','Result','OTs','Site'
                    # Drop NaN rows
                    player = player.dropna(thresh=19)
                    # Switch NaN to '0OT' in the OTs column
                    player = player.fillna('0OT')
                    # Save
                    player.to_csv(os.path.join(dir_path,name) + '_csv')
```

Responses :

2.

(a) For the most part, the dat shouldn't have any bias. Most of it is collected directly from the NCAA records which are collected independent of the teams the numbers represent. The only potential problem I can see is that there are a few variables which are calculated by a very specific formula that I didn't create and can't currently verify. kenpom.com verifies them as accurate, but I don't currently have anything more than it's word. That being said, kenpom.com is also independent of all NCAA teams, and so has no reason to intentionally be misleading about these variables.

(b) All data seems complete, and like it is accurate. I've identified no odd, or abnormal values in my sampling.

3.

While the data seems to contain everything I need to evaluate the individual contribution of each player to NCAA tournament and season games as a whole, a few more questions have come to my attention. For instance, which players are likely to perform well together? Is it possible to identify team leaders from the performance data gathered? If those leaders are having an "off" day, does the entire team perform worse, or does another player step up? How does this effect the success or failure of the team in the NCAA tournament?

4.

In essence, focusing on what data from the season will indicate the performance of a given team in the tournament will be most important. The goal will be to see if the story a team plays out during the season will be able to determine the conclusion in the tournament.