University of Warsaw
Faculty of Economic Sciences

Robert Kowalczyk	Szymon Socha	Jan Dudzik
397108	411462	396596

Webscraping and Social Media Scraping - information of basketball players

Final project for the Webscraping and
Social Media Scraping
class chaired by Przemysław Kurek
(Deparment of Political Economy)
WNE UW

Warsaw, May 2022

Short description of the topic and the web page

Our project is to use three available webscrap libraries:

- BeautifulSoup
- Scrapy
- Selenium

in the Python programming language to collect information on basketball players. This uses data available from the following website: *https://www.basketball-reference.com/*. On this website, it is also possible to view all data on existing teams in all leagues in the world, check the latest match results or delve into historical data. Our team has focused strictly on the basic statistics describing basketball players, such as number of games played, average points, assists or rebounds per game, 3-point shooting efficiency, and overall player rating.

The website itself is very clear, full of many interesting information. In addition, we have not noticed that this website has dynamic appearance generation through JavaScript.

Short description of scraper mechanics

BeautifulSoup & Selenium

We can treat the BeautifulSoup and Selenium libraries similarly, as the behaviour of these scrapers does not differ significantly.

- Importing the necessary libraries and preparation of the relevant parameters.
    a. At the very beginning of the program, we import the necessary libraries to handle web queries. For each scraper we import essential libraries like *BeautifulSoup* from *bs4*, *webdriver* from *selenium* and *scrapy*. We also import other libraries like *pandas* for data manipulation, *warnings* to handle future warnings, *tqdm* to implement progress bar.
    b. The algorithm uses two boolean parameters: `limit_to_100` and `save_output_to_csv`, limiting the number of pages searched to 100 and deciding whether the analysis's result should be saved to a csv file respectively.
- Collecting page addresses for each letter of the alphabet.
    a. The collection of information by the scraper starts with the address of the website: *https://www.basketball-reference.com/players/* where the players are sorted by alphabet. An example link for A players is as follows: *https://www.basketball-reference.com/players/a/*.
    b. For each letter of the alphabet, the scraper tries to collect links to the first 20 players in the table. When the `limit_to_100` parameter takes the value **True**, our scraper is

limited to only 100 players. It should collect information about players for the letters A, B, C, D and E and complete its operation. Otherwise, we will collect 20 basketball players for each letter of the alphabet (as long as that many players exist in the table). An example link for an Alaa Abdelnaby player is: *https://www.basketball-reference.com/players/a/abdelal01.html*

- Collecting basic information about the basketball player.
    a. For each player, the scraper collects information on the number of games played, average points, assists and rebounds per game, 3-point shooting efficiency and overall rating of the basketball player.
- Saving data to DataFrame and optionally exporting to csv file.
    a. After each iteration of scraping a basketball player's statistics, new data is added to the DataFrame.
    b. If the parameter `save_output_to_csv` takes the value **True** the export of the collected data to a csv file takes place.

Scrapy

Scrapy part of the project consists of three separate files called Spiders.

- nba_list_of_links.py

  First spider called *nba_list_of_links* is activated by the terminal command "scrapy crawl nba_list_of_links -o list_of_links.csv". After starting at the initial link https://www.basketball-reference.com/players/ first spider fetches href links to websites containing players with names starting with a particular alphabet letter between a and z. Suggested command gathers links in a csv file called list_of_links.csv.

- nba_links.py

  Second spider named *nba_players_links* uses output of the first spider as a starting point for scraping href links to particular players sites. For every line of the first output (https://www.basketball-reference.com/players/a; https://www.basketball-reference.com/players/b etc.) nba_players_links spider using specified xpath finds first 20 players from every site. Spider can be run by terminal command "scrapy crawl nba_players_links -o players_links.csv" saving output href links as a csv.

- nba_stats.py

  Third and last spider named *players* take results from nba_player_links as a start urls. For every provided player's page, the spider fetches basic statistics using specified xpaths. What is more, the columns' order is specified in custom settings of the LinksSpider class. With command "scrapy crawl players -o ballers_big.csv" spider can be run and statistics of every player are stored in a csv as a rows.

Short technical description of the output

Each of the scrappers returns two flat files - ballers_small.csv and ballers_big.csv, which differ only by number of rows. ballers_small.csv takes only 100 first records (limit_to_100 = True), while ballers_big.csv takes all of the records - 20 per letter.
The output files follow the column structure: Name and Surname, games, points, rebounds, assists, field Goal Percentage, three Field Goal, free Throw Percentage, effective Field Goal, player Efficiency, win Shares. Value of "-" means that the statistic provided by a website is blank, but it exists, while blanks are used for cells, which weren't mentioned at all on the website for this player.

Data analysis

We use scraped data to show the relationship between points and assists with division to good and bad players. First, we have to replace all non-numeric symbols with zeros. Then we calculate the mean of players' efficiency. If one's player efficiency is below the mean he's considered as bad, otherwise (if more than mean) he's considered as a good player. Next, we use a scatter plot with points on x-axis and assists on y-axis. Additionally, with a help of color argument we distinguish two data subsamples; good and bad players. We also add a trendline.

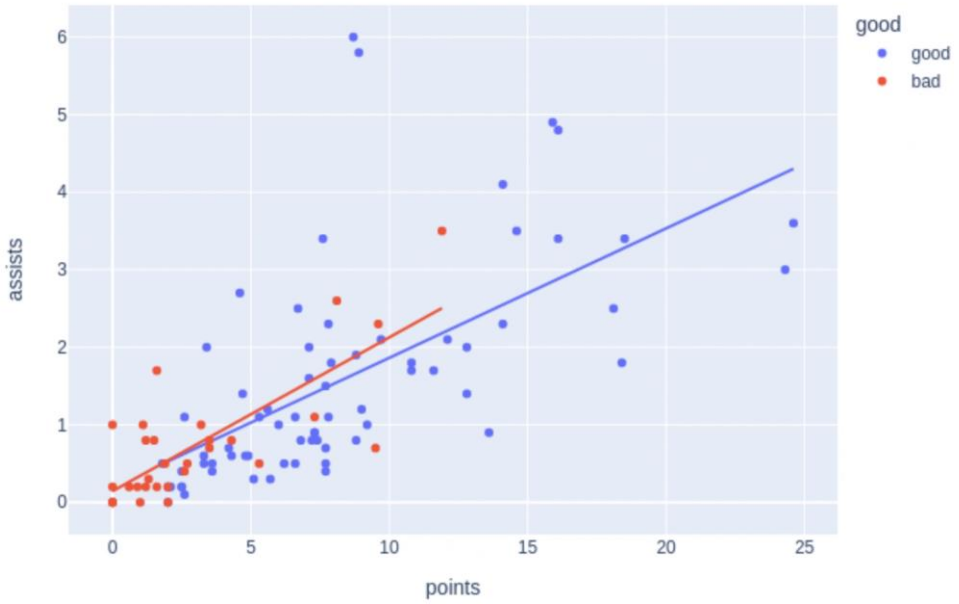

Figure 1 Scatter plot of dependency between points and assists by good and bad players

Code used to plot the figure above:

```
import pandas as pd
import numpy as np
import plotly. express as px
import statsmodels.api as sm
small = pd.read_csv ('ballers_small.csv')
small["playerEfficiency"] = (pd.to_numeric(small["playerEfficiency"],errors='coerce').fillna(0))
small["playerEfficiency"] = small["playerEfficiency"].astype(float).astype(int)
mean = small["playerEfficiency"].mean()
small["good"] = np.where(small["playerEfficiency"] > mean, "good", "bad")
fig = px.scatter(small, x="points", y="assists", trendline="ols", color = "good")
fig.show()
fig.write_image("points_assists.png")
```

As we could expect, bad players score fewer points and get fewer assists. The slope for bad players is slightly steeper than for good players. So, we can expect that scoring more points will result in a higher increase in the number of assists than we could observe for good ones.

The data collected could further help us create a machine learning model that could suggest to coaches how to play specific teams, which players we should cover the most. This could produce significant results and take the quality of basketball to even higher levels.

Participation in webscraping project

Task accomplished	Responsible person
Creation of a scraper for the BeautifulSoup library	Robert Kowalczyk
Creation of a scraper for the Scrapy library	Jan Dudzik
Creation of a scraper for the Selenium library	Szymon Socha
Creation of README.md file	Robert Kowalczyk Jan Dudzik Szymon Socha
Short description of the topic and the web page	Robert Kowalczyk
Short description of scraper mechanics	Robert Kowalczyk (BeautifulSoup) Jan Dudzik (Scrapy) Szymon Socha (Selenium)
Short technical description of the output	Jan Dudzik
Extremely elementary data analysis	Szymon Socha
Creation of description.pdf file	Robert Kowalczyk Jan Dudzik Szymon Socha