# IS5126 : Hands on with Business Analytics

Group 10

15-Feb-2016

# Part 1 - Web-Scraping

1. The starting URLs for web-scraping are:

For players related information:

http://www.basketball-reference.com/players/

For teams related information:

http://www.basketball-reference.com/teams/

2. Players are grouped by their last names in A to Z categories. We need to open each web page of the categories to get the URL of every active individual player. All categories for players are in http://www.basketball-reference.com/players/[a-z]/. So for page in http://www.basketball-reference.com/players/a/, we need to grab all the URLs of active players which are listed in **bold**. The syntax of their URLs is like /players/[a-z]/.\*\.html (e.g. /players/a/abdelal01.html). For teams, we need to grab all the URLs of teams in the **Active Franchises** table of the page in http://www.basketball-reference.com/teams/. All active player URLs are stored in ***player\_urls.csv*** and all active team URLs are stored in ***team\_urls.csv***.

a. Regex to get player index A-Z URLs:

|  |
| --- |
| urls = re.findall('href="(/players/[a-z]/)"',webpage)  hrefs = urls[0:25] if len(urls) > 0 else []  return hrefs; |

Regex to get all active player URLs:

|  |
| --- |
| webpage = urlopen(playerurl).read()  strongtags = re.findall('<strong><a(.\*?)</strong>',webpage)  playerurls = []  if strongtags:  for i in strongtags:  href = re.findall('/players/[a-z]/.\*\.html', i)  if href:  playerurls.append(''.join(href).strip())  return playerurls; |

Regex to get all active team URLs:

|  |
| --- |
| urls = re.findall('<td align="left" ><a href="(.\*?)">', webpage)  #only active teams  hrefs = urls[0:30]  return hrefs; |

b. BS4 to get player index A-Z URLs:

|  |
| --- |
| soup = BeautifulSoup(webpage, "html.parser")  index = soup.find("div",{"id":"page\_content"}).find("p")  hrefs = []  for i in index.findAll('a'):  href = i.get('href')  hrefs.append(href)  return hrefs; |

BS4 to get all active player URLs:

|  |
| --- |
| webpage = urlopen(playerurl).read()  soup = BeautifulSoup(webpage, "html.parser")  players = soup.find("table",{"id":"players"}).findAll("strong")  playerurls = []  for i in players:  href = i.find("a").get("href")  playerurls.append(href)  return playerurls; |

BS4 to get all active team URLs:

|  |
| --- |
| soup = BeautifulSoup(webpage, "html.parser")  table= soup.find("table",{"id":"active"}).findAll("tr",{"class":"full\_table"})  hrefs = []  for i in table:  for k in i.find\_all('a'):  team = k.get('href')  hrefs.append(team)  return hrefs; |

**c. Comparing Regual Expression (Regex) and BeautifulSoup (BS4) to grab data**

We use the Python **timeit** module to calculate the running time for each method to grab all the necessary data. Reading one web page delays 1 second. Source code is in CompareMethods.py. Running results are as follows:

|  |  |  |
| --- | --- | --- |
| Method | Running Time (s) | Remarks |
| A. Regex to get player index A-Z URLs | 1.79 | Include time to read a single web page |
| B. BS4 to get player index A-Z URLs | 1.86 | Include time to read a single web page |
| C. Regex to get all active player URLs | 89.95 | Include method A and read many web pages |
| D. BS4 to get all active player URLs | 123.44 | Include method B and read many web pages |
| E. Regex to get all active team URLs | 2.21 | Include time to read a single web page |
| F. BS4 to get all active team URLs | 3.89 | Include time to read a single web page |

We chose Regex to process all active player and team URLs data. It's relatively easier to write the code using Regex method to grab these data. And using Regex is much faster to process the data.

**3. Player's page**

a. Basic player's profile information is stored in ***players\_profile.csv*** file.

b. Player statistics

c. Player salaries

d.

4. Team's page

a. Basic team information

b. Team statistics y season

5. Other information