## Part 1 : Webscraping along with Pipeline

Introduction:

Web scraping is an automated process to extract information from the internet. It is a powerful tool that can help businesses and individuals gather useful information from various sources on the internet. The data can be used for various purposes such as market research, sentiment analysis, trend analysis, etc. In this blog, we will discuss why web scraping is necessary and how it can be done using Python.

Motivation:

With the proliferation of the internet and the increasing availability of information online, there is a need for an efficient way to collect and analyze this information. Web scraping offers a solution to this problem by allowing users to extract large amounts of data from the internet quickly and easily. This data can then be used for various purposes such as market research, trend analysis, and competitor analysis, to name a few.

Why we need webscraping?

Web scraping is essential because it allows us to collect and analyze data that is not readily available in a structured format. It helps us in understanding market trends, product reviews, and pricing strategies. It also enables us to gather data that can help us gain a competitive advantage in the market. Web scraping can be used for a wide range of applications such as e-commerce, data analysis, and sentiment analysis.

For example, imagine you own an online clothing store and want to monitor the prices of a popular competitor’s clothing items. You could use a web scraper to extract the prices of those items from their website and store the data in a spreadsheet. Then, you could set up an automated notification to alert you when a price drops below a certain threshold. This would give you a competitive advantage by allowing you to adjust your prices accordingly.

In this way, web scraping can save you time, help you make more informed decisions, and give you a leg up on the competition.

Example of the webscraping using below code:

The following code is an example of how web scraping can be done using Python. The code scrapes a webpage and extracts all the tables from the webpage using BeautifulSoup. It then extracts the country sex ratios from the tables and cleans the data to retrieve ISO country codes from a GitHub repository.

import math  
import re  
import urllib3  
from typing import List, Tuple  
import pandas as pd  
from bs4 import BeautifulSoup  
  
  
def get\_webpage\_tables(url: str) -> List[BeautifulSoup]:  
 """  
 Downloads a webpage from the given url and extracts all the tables using BeautifulSoup.  
  
 Parameters:  
 url (str): The url of the webpage to download.  
  
 Returns:  
 List[BeautifulSoup]: A list of BeautifulSoup objects, one for each table on the webpage.  
 """  
 http = urllib3.PoolManager()  
 response = http.request("GET", url)  
 soup = BeautifulSoup(response.data, features="html.parser")  
 tables = soup.find\_all('table')  
 return tables  
  
  
def process\_num(num: str) :  
 """  
 Processes a string representing a number to a float.  
  
 Parameters:  
 num (str): A string representing a number.  
  
 Returns:  
 Union[float, str]: The float representation of the input string, or the original string if it cannot be converted.  
 """  
 try:  
 return float(re.sub(r'[^\d.-]', '', num))  
 except ValueError:  
 return num  
  
  
  
def is\_valid\_cell(cells: List[BeautifulSoup]) -> bool:  
 """  
 Determines if a list of BeautifulSoup objects representing table cells is valid.  
  
 A cell is considered invalid if its text content is 'N/A'.  
  
 Parameters:  
 cells (List[BeautifulSoup]): A list of BeautifulSoup objects representing table cells.  
  
 Returns:  
 bool: True if all the cells are valid, False otherwise.  
 """  
 return all(cell.text.strip() != 'N/A' for cell in cells)  
  
  
  
def make\_str\_valid(s: str) -> str:  
 """  
 Makes a string valid by removing any text in parentheses.  
  
 Parameters:  
 s (str): The string to make valid.  
  
 Returns:  
 str: The input string with any text in parentheses removed.  
 """  
 ind\_delim = s.find('(')  
 if ind\_delim != -1:  
 wrd = s[:ind\_delim-1]  
 else:  
 wrd = s  
 return wrd  
  
  
def extract\_country\_sex\_ratio(tables: List[BeautifulSoup]) -> pd.DataFrame:  
 """  
 Extracts country sex ratios from a list of BeautifulSoup tables.  
  
 Parameters:  
 tables (List[BeautifulSoup]): A list of BeautifulSoup tables.  
  
 Returns:  
 pd.DataFrame: A pandas DataFrame containing the extracted country sex ratio data.  
 """  
 data = []  
 for table in tables:  
 rows = table.find\_all('tr')  
 for row in rows:  
 cells = row.find\_all('td')  
 if len(cells) > 1 and is\_valid\_cell(cells):  
 country = make\_str\_valid(cells[0].text.strip())  
 sex\_ratio = process\_num(cells[-1].text.strip())  
 data.append([country, sex\_ratio])  
  
 df = pd.DataFrame(data, columns=['Country', 'Sex-Ratio'])  
 return df  
  
  
  
def get\_country\_codes(df) -> pd.DataFrame:  
 """  
 Retrieves ISO country codes from a GitHub repository and returns a cleaned DataFrame  
 """  
 country\_raw = pd.read\_csv('https://raw.githubusercontent.com/lukes/ISO-3166-Countries-with-Regional-Codes/master/all/all.csv')  
 df\_c = country\_raw.iloc[:, [0,2]]  
 df\_c = df\_c.rename(columns={'name':'Country', 'alpha-3':'ISO-code'})  
 df\_c = df\_c.drop\_duplicates(subset=['Country'], keep='last').dropna()  
  
 # Manually modify some of the country names to match the names in the main DataFrame  
 df\_c.at[df\_c[df\_c['Country']=='Viet Nam'].index.values[0], 'Country'] = 'Vietnam'  
 df\_c.at[df\_c[df\_c['Country']=='United States of America'].index.values[0], 'Country'] = 'United States'  
 df\_c.at[df\_c[df\_c['Country']=='Iran (Islamic Republic of)'].index.values[0], 'Country'] = 'Iran'  
 df\_c.at[df\_c[df\_c['Country']=='Russian Federation'].index.values[0], 'Country'] = 'Russia'  
 df\_c.at[df\_c[df\_c['Country']=='United Kingdom of Great Britain and Northern Ireland'].index.values[0], 'Country'] = 'United Kingdom'  
 df\_c.at[df\_c[df\_c['Country']=='Venezuela (Bolivarian Republic of)'].index.values[0], 'Country'] = 'Venezuela'  
 df\_c.at[df\_c[df\_c['Country']=="Korea (Democratic People's Republic of)"].index.values[0], 'Country'] = 'Korea, North'  
 df\_c.at[ df\_c[df\_c['Country']=='Korea, Republic of'].index.values[0], 'Country' ] = 'Korea, South'  
 df\_c.at[ df\_c[df\_c['Country']=='Bolivia (Plurinational State of)'].index.values[0], 'Country' ] = 'Bolivia'  
 df\_c.at[ df\_c[df\_c['Country']=='Côte d\'Ivoire'].index.values[0], 'Country' ] = 'Ivory Coast'  
 df\_c.at[ df\_c[df\_c['Country']=='Congo'].index.values[0], 'Country' ] = 'Congo, Republic of the'  
 df\_c.at[ df\_c[df\_c['Country']=='Tanzania, United Republic of'].index.values[0], 'Country' ] = 'Tanzania'  
 # Using the ISO-3166 coding standard to map countries  
 df['ISO-code'] = df['Country'].map(df\_c.set\_index('Country')['ISO-code'])  
 # Clean data-frame ( Duplicates & NaNs )  
 df.isna().sum()   
 df = df.dropna()  
 return df

import pandas as pd  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import FunctionTransformer  
  
  
url = 'https://en.wikipedia.org/wiki/List\_of\_countries\_by\_sex\_ratio'  
  
# define pipeline  
pipeline = Pipeline([  
 ('get\_tables', FunctionTransformer(get\_webpage\_tables)),  
 ('extract\_ratio', FunctionTransformer(extract\_country\_sex\_ratio)),  
 ('get\_codes', FunctionTransformer(get\_country\_codes))  
])  
  
# apply pipeline to input URL  
df = pipeline.transform(url)  
  
df

/usr/local/lib/python3.8/dist-packages/urllib3/connectionpool.py:842: InsecureRequestWarning:  
  
Unverified HTTPS request is being made. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings

|  | Country | Sex-Ratio | ISO-code |
| --- | --- | --- | --- |
| 1 | Afghanistan | 1.03 | AFG |
| 2 | Albania | 0.98 | ALB |
| 3 | Algeria | 1.03 | DZA |
| 4 | American Samoa | 1.0 | ASM |
| 5 | Andorra | 1.06 | AND |
| ... | ... | ... | ... |
| 225 | Wallis and Futuna | 1.04 | WLF |
| 227 | Western Sahara | 0.99 | ESH |
| 228 | Yemen | 1.02 | YEM |
| 229 | Zambia | 1.0 | ZMB |
| 230 | Zimbabwe | 0.96 | ZWE |

import plotly.express as px  
  
thres = 1.3  
df\_th = df.drop(df[ df['Sex-Ratio'] > thres ].index)  
  
# color pallete @ https://plotly.com/python/builtin-colorscales/  
fig = px.choropleth(df\_th, locations='ISO-code',  
 color="Sex-Ratio", hover\_name="Country",  
 color\_continuous\_scale=px.colors.sequential.Sunset, projection="natural earth")  
fig.update\_layout(title={'text':'Sex-Ratio per country', 'y':0.95, 'x':0.5, 'xanchor':'center', 'yanchor':'top'})  
fig.show()

Unable to display output for mime type(s): text/html

import plotly.express as px  
  
thres = 1.3  
df\_th = df.drop(df[ df['Sex-Ratio'] > thres ].index)  
  
# color pallete @ https://plotly.com/python/builtin-colorscales/  
fig = px.choropleth(df\_th, locations='ISO-code',  
 color="Sex-Ratio", hover\_name="Country",  
 color\_continuous\_scale=px.colors.sequential.Sunset, projection="orthographic")  
fig.update\_layout(title={'text':'Sex-Ratio per country', 'y':0.95, 'x':0.5, 'xanchor':'center', 'yanchor':'top'})  
fig.show()

Unable to display output for mime type(s): text/html