Assignment - 1

PDV

Karthik. C

241058016

Big Data Analytics

1. what is web Scrapping. Describe the steps to perform web scrapping.

→ web scrapping: It is an automatic method to obtain large amounts of data from websites. most of this data is unstructured data in a HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications. There are many different ways to perform web scrapping to obtain data from websites. these include using online services, particular API's or Even creating your code for web scraping from scratch. Many large websites like Google, Twitter, Facebook, StackOverflow, Etc. have API's that allow you to access large amounts of data in a structured ~~data~~ format. This is the best option, but there are other sites that don't allow users to access large amounts of data in a structured from or they are simply not that technologically advanced. In that situation, it's best to use web scrapping to scrape the website for data.

Steps to perform data scrapping

i) Identify the target website :-

Choose a website from which data needs to be Extracted. Analyze the structure and format of the website, Ensuring that the data is publicly available and that scraping the site compiles with its terms of service.

ii) Inspect the web page:- use the browser developer tools (Inspect) to Examine the HTML structure of the web page.

Identify the specific tags such as <div> <table> <span> etc
that contains data you want to scrape.

for Ex in a Ecommerce site, product names and prices may be stored in specific HTML Elements

iii) choose the appropriate tools or libraries

Depending on the programming languages, choose appropriate libraries or frameworks for web scraping. In python popular libraries include :

i) Beautiful Soup :- for parsing HTML and Extracting data.

ii) Selenium :- for simulating browser interaction & scraping dynamic content.

iii) Scrapy :- A comprehensive framework for large scale scraping projects.

iv) Requests :- for sending HTTP requests to fetch

iv) Send HTTP requests :- use tools like requests to send HTTP GET requests to like server and retrieve the HTML content of the web page. the response from the server contains the raw HTML data

```
import requests
response = requests. get ('HTTPs: // Example. com)
html_content = response. content
```

v) Parse the HTML content :- one the HTML content is fetched, parse ~~the HTML content.~~ ~~on~~ it using a library like Beautiful Soup the allows for Easy navigation of the document tree & Extraction of data from specific tags & attributes from bs4 Import beautiful Soup.

```
soup = BeautifulSoup (html - content . 'HTML . parser')
titles = soup. find_all ('h1')
```

6) Extract required data:- use the HTML parsed to locate & Extract the relevant data, this might involve searching for specific tags classes (or) IDs that contain the info

```
prices = Soup. find_all ('span', class = 'price')
```

7) Handle Pagination:- many website split data across multiple pages. If the target website uses pagination Identify the URL patterns or navigations buttons & write logic to scrape all relevant pages iteratively.

viii) Store the Data: After Extraction, store the data in a structured format, such as csv file, JSON file (or) database import csv.

```
    with open ('data.csv', 'w') as file:
        writer = csv. writer (file)
        writer.writerow(['Title', 'Price'])
        for title, Price in zip (tittles, prices):
            writer. writerrow (titles. text, price text)
```

ix) Handle Javascript content!- Some websites load content dynamicany using Javascript. tools like selenium can be used to simulate a browser Environment and Extract data ~~from~~ after Javascript Execution.

x) Respect Ethical & legal Boundaries:- Always check the website's robots. txt file to understand the permissions for crawling & scraping Ensure compliance with legal regulations, including copyright laws & terms of service & avoid overloading servers

2. Discuss the challenges in performing web scraping

→ web scraping, while an effective method of extracting data from websites, comes with several challenges & limitations. these challenges can arise from legal technical or ethical considerations, making it curical for developers to address them when performing scrapping tasks.

1. IP Ban's :- An Ip address can be banned or rate-limited if a website determines that it's being used to make malicious or excessive critical if the bot uses a single IP address for a large number of requests.

2. CAPTCHA's:. (Completely Automated Public Turing Tests to Tell computers & Humans) are a popular security measure, making it difficult for scrapers to gain access & extract data from websites.

3. Dynamic content:- web scrapping techniques traditionally rely heavily on analyzing HTML source code, which frequently only contains immutable data.

4. Honeybot Traps:- These are all one of the web scraping challenges that bots mostly fall for.

5. Ethical & legal issues:- ~~the~~ web scraping is not an illegal act itself if the extracted data is not used for unethical purposes. In many legal cases where businesses were using web crawlers to extract competition's public data, judges did not find a legimate reason to rule against the crawlers, even though crawling was frowned upon by the data's owners. for Ex in the case of eBay vs. Bidder's Edge, an auction data aggregator who used a proxy to craw eBay's data.

3) Demonstrate a python web-scrapping script using beautifulsoup library for navigating any html file of your choice.

→ Beautiful soup is a python library that ~~ac~~ allows Easy navigation, searching and modification of HTML and

```
from bs4 Import BeautifulSoup
import requests
import time
~~[illegible]~~

def find_jobs():
    html_text = requests.get('https://www.timesjobs.com/candidate').text
    soup = BeautifulSoup(html_text, 'html.parser')
    jobs = soup.find_all('li', class_='clearfix job-bx wht').text

for index, job in Enumerate(jobs):
        Published_date = job.find('span', class_='sim-posted').span.text
        if 'few' in published-date:
            company_name = job.find('h3', class_='joblist').text
            skills = job.find('span', class_='srp-skills').text
            more_info = job.header.h2.a['href']
if __name__ == '__main__':
    while True:
        find_jobs()
        time.sleep(5)
Print (f''' company Name : {company name}
        Required Skills : {skills}
        more_info :{more-info} ''')
```

4) Illustrate the difference b/w loc and iloc function in python Program with a code snippet.

→ In python, loc & iloc are functions provided by the Pandas library to access data from a DataFrame. they serve similar purposes but differ in the way they select data:

- loc is used for label-based indexing. it allows you to select rows & columns by labels (row/column names or indices)
- iloc is used for position-based-indexing. it selects rows & columns by integer positions (row/column index numbers).

Code

```
import pandas as pd
 data = { 'Name' : ['Alice', 'Bob', 'charlie', 'Harsha'],
          'Age' : [25, 23, 26, 27],
          'Score' : [85, 90, 95, 87]
        }

 df = Pd. Dataframe (data, index = ['a', 'b', 'c', 'd']

 Print ("original DataFrame :\n", df)

 Print ("\using loc to select row 'b' & columns 'Name' and 'Age'.")
 Print (df. loc ['b', ['Name', 'Age']])

 Print ("\Using iloc to select rows 'b' to 'd' & columns 'Name' &
         'Score'. ")
 Print (df . loc ['b': 'd', ['Name', 'Score']])

 Print ("Using iloc to select the second row (index 1) & first two
         columns")
 Print (df. iloc [1, [0, 1]])
Print (" Using iloc to select rows 2 to 3 & columns 0 and 2.")
Print (df.iloc [2:4. [0,2]])
```

Output:-

```
Original Data Frame.                      Using loc to select row 'b' & columns

     Name      Age    Score                    Name     Bob
a    Alice     25     85                       Age      23
b    Bob       23     90
c    Charlie   26     95
d    Harsha    27     87

Using loc to select rows 'b' to 'd'      Using iloc to select the second row

     Name     Score                                 Name    Bob
b    Bob      90                                    Age     30
c    Charlie  95
d    Harsha   87

Using iloc to select rows 2 to 3

       Name     Score
   c   charlie  95
   d   Harsha   87
```

- loc uses label to select data
- iloc uses integer position to select data

5. Write a python program to create 3x4 array, delete 3rd column & insert new column in the 3rd column, print the intermediate results.

→
```
import numpy as np

array = np. array ([[1,2, 3. 4],
                    [5. 6, 7. 8],
                    [9. 10, 11, 12]])
Print (" Original Array (3x4): ")
Print (array)
array-without-3rd_col = np.delete (array, 2, axis=1)
Print (" Array after deleting 3rd column:")
Print (array_without-3rd col)

new_column = np.array ([[13,14,15]])

array-with-new-col = np.insert (array_without_3rd_col, 2,
        new-column, axis=1)      // axis is w.r.to column is 1
Print ("Array after inserting new column at 3rd column positio")
Print (array-with_new -col)
```

Output
=

```
Original Array (3x4)              Array after deleting 3rd column
[[1, 2, 3, 4]                         [[1  2  4]
 [5, 6, 7, 8]                          [5  6  8]
 [9, 10, 11, 12]]                      [9  10  12]]

Array after inserting new column at 3rd column position
 [[1  2  13  4]
  [5  6  14  8]
  [9  10  15  12]]
```