UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

M.Sc. Computer Science – Semester II (NEP 2020)

Web Data Analysis

JOURNAL

2023-2024

Seat No. ____________________

मुंबई विद्यापीठ
University of Mumbai
Re-accredited with A++ Grade
(CGPA 3.65) by NAAC (3rd Cycle 2021)

UNIVERSITY OF MUMBAI

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the work entered in this journal was done in the University Department of Computer Science laboratory by Mr./Ms.______________________________ Seat No. ____________ for the course of M.Sc. (Computer Science) - Semester II (NEP 2020) during the academic year 2023- 2024 in a satisfactory manner.

____________________ ____________________

Subject In-charge **Head of Department**

____

External Examiner

INDEX

Practical No: 1

Aim: - Scrape an online E-Commerce Site for Data.

1. Extract product data from Amazon - be it any product and put these details in the MySQL database. One can use pipeline. Like 1 pipeline to process the scraped data and other to put data in the database and since Amazon has some restrictions on scraping of data, ask them to work on small set of requests otherwise proxies and all would have to be used.

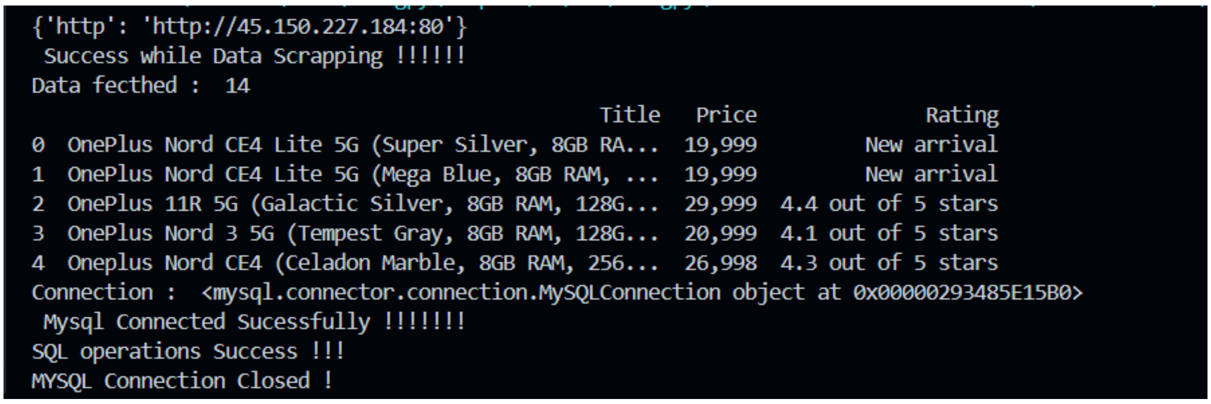
2. Scrape the details like color, dimensions, material etc. Or customer ratings by features

Theory: Scrapy is a framework for extracting data from websites. It does so by crawling websites and extracting data from the HTML of the web pages. It provides a systematic approach to scraping, which involves sending requests to websites, parsing the responses, and extracting the desired data.

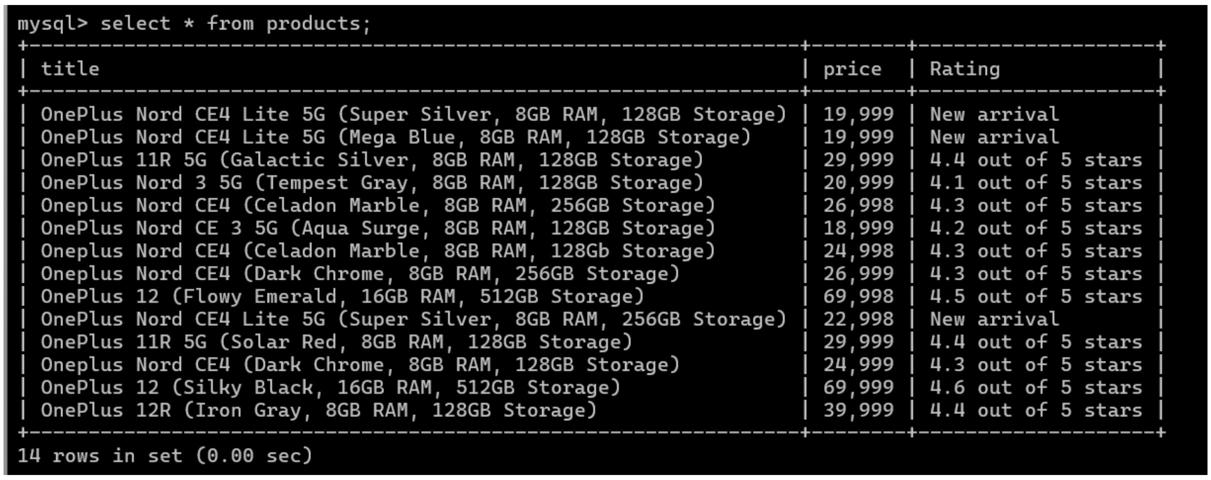
Code:

```
import requests
from fp.fp import FreeProxy
from bs4 import BeautifulSoup as bs
import mysql.connector
from mysql.connector import Error
import pandas  as pd

proxy = FreeProxy(country_id=['US']).get()
pro = { "http"  : proxy}
print(pro)

head = {
 "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.
8,application/signed-exchange;v=b3;q=0.7",
    "Accept-Encoding": "gzip, deflate, br, zstd",
    "Accept-Language": "en-GB,en-US;q=0.9,en;q=0.8",
    "Cache-Control": "max-age=0",
    "Device-Memory": "8",
    "Downlink": "10",
    "Dpr": "1.25",
    "Ect": "4g",
    "Priority": "u=0, i",
    "Rtt": "0",
    "Sec-Ch-Device-Memory": "8",
    "Sec-Ch-Dpr": "1.25",
    "Sec-Ch-Ua": "\"Not/A)Brand\";v=\"8\", \"Chromium\";v=\"126\", \"Google Chrome\";v=\"126\"",
    "Sec-Ch-Ua-Mobile": "?1",
    "Sec-Ch-Ua-Platform": "\"Android\"",
    "Sec-Ch-Ua-Platform-Version": "\"6.0\"",
    "Sec-Ch-Viewport-Width": "777",
    "Sec-Fetch-Dest": "document",
    "Sec-Fetch-Mode": "navigate",
    "Sec-Fetch-Site": "none",
    "Sec-Fetch-User": "?1",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/126.0.0.0 Mobile Safari/537.36",
    "Viewport-Width": "777"
}

response=
requests.get('https://www.amazon.in/s?bbn=1389401031&rh=n%3A1389401031%2Cp_89%3AOnePl
us',headers=head,proxies=pro)

if response.status_code == 200:
    print(' Success while Data Scrapping !!!!!! ')
    soup = bs(response.content,features="lxml")
    divs = soup.find_all('div')
     cards = soup.find_all('div',{'class':"s-result-item s-asin sg-col sg-col-12-of-12 s-widget-spacing-
small"})
    print("Data fecthed : ",len(cards))

    t,p,r = [],[],[]
    for card in cards:
        price = card.find('span',{'class':'a-price-whole'})
        if price is None:
            continue

        title = card.find('span',{'class':'a-size-small a-color-base a-text-normal'})
        rating = card.find('span',{'class':'a-icon-alt'})

        if rating:
            rating = rating.text
        else:
            rating = "New arrival"

        t.append(title.text)
        p.append(price.text)
        r.append(rating)

    df = pd.DataFrame({"Title":t,"Price":p,"Rating":r})
    print(df.head(5))

    username = 'root'
    password = 'Mohit'
    host = 'localhost'
    port = '3306'
    database = 'bank'

    try:
        connection = mysql.connector.connect(
            host=host,
            user=username,
            password=password,
            database=database,
            port=port
        )

        print("Connection : ",connection)

        if connection.is_connected():
            print(" Mysql Connected Sucessfully !!!!!!!")
            cursor = connection.cursor()

            sql = "DROP TABLE IF EXISTS products"
            cursor.execute(sql)
            connection.commit()

            cursor.execute("CREATE TABLE products (title TEXT, price TEXT, Rating TEXT)")

            for data in zip(df.values):
                sql = "INSERT INTO products (title, price, rating) VALUES (%s, %s,%s)"
                val = (data[0][0],data[0][1],data[0][2])
                cursor.execute(sql, val)

                connection.commit()

            print("SQL operations Success !!!")

    except Error as e:
        print(f"Error: {e}")

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
        print("MYSQL Connection Closed !")

else:
    print(response.status_code )
```

Output: -



```
{'http': 'http://45.150.227.184:80'}
 Success while Data Scrapping !!!!!!
Data fecthed :  14
                                               Title   Price              Rating
0  OnePlus Nord CE4 Lite 5G (Super Silver, 8GB RA...  19,999         New arrival
1  OnePlus Nord CE4 Lite 5G (Mega Blue, 8GB RAM, ...  19,999         New arrival
2  OnePlus 11R 5G (Galactic Silver, 8GB RAM, 128G...  29,999  4.4 out of 5 stars
3  OnePlus Nord 3 5G (Tempest Gray, 8GB RAM, 128G...  20,999  4.1 out of 5 stars
4  Oneplus Nord CE4 (Celadon Marble, 8GB RAM, 256...  26,998  4.3 out of 5 stars
Connection :  <mysql.connector.connection.MySQLConnection object at 0x00000293485E15B0>
 Mysql Connected Sucessfully !!!!!!!
SQL operations Success !!!
MYSQL Connection Closed !
```



```
mysql> select * from products;
+------------------------------------------------------------------+--------+--------------------+
| title                                                            | price  | Rating             |
+------------------------------------------------------------------+--------+--------------------+
| OnePlus Nord CE4 Lite 5G (Super Silver, 8GB RAM, 128GB Storage)  | 19,999 | New arrival        |
| OnePlus Nord CE4 Lite 5G (Mega Blue, 8GB RAM, 128GB Storage)     | 19,999 | New arrival        |
| OnePlus 11R 5G (Galactic Silver, 8GB RAM, 128GB Storage)         | 29,999 | 4.4 out of 5 stars |
| OnePlus Nord 3 5G (Tempest Gray, 8GB RAM, 128GB Storage)         | 20,999 | 4.1 out of 5 stars |
| Oneplus Nord CE4 (Celadon Marble, 8GB RAM, 256GB Storage)        | 26,998 | 4.3 out of 5 stars |
| OnePlus Nord CE 3 5G (Aqua Surge, 8GB RAM, 128GB Storage)        | 18,999 | 4.2 out of 5 stars |
| Oneplus Nord CE4 (Celadon Marble, 8GB RAM, 128Gb Storage)        | 24,998 | 4.3 out of 5 stars |
| Oneplus Nord CE4 (Dark Chrome, 8GB RAM, 256GB Storage)           | 26,999 | 4.3 out of 5 stars |
| OnePlus 12 (Flowy Emerald, 16GB RAM, 512GB Storage)              | 69,998 | 4.5 out of 5 stars |
| OnePlus Nord CE4 Lite 5G (Super Silver, 8GB RAM, 256GB Storage)  | 22,998 | New arrival        |
| OnePlus 11R 5G (Solar Red, 8GB RAM, 128GB Storage)               | 29,999 | 4.4 out of 5 stars |
| Oneplus Nord CE4 (Dark Chrome, 8GB RAM, 128GB Storage)           | 24,999 | 4.3 out of 5 stars |
| OnePlus 12 (Silky Black, 16GB RAM, 512GB Storage)                | 69,999 | 4.6 out of 5 stars |
| OnePlus 12R (Iron Gray, 8GB RAM, 128GB Storage)                  | 39,999 | 4.4 out of 5 stars |
+------------------------------------------------------------------+--------+--------------------+
14 rows in set (0.00 sec)
```

Practical No: 2

Aim; - Scrape an online Social Media Site for Data. Use python to scrape information from Time of India.

Theory: -

Web scraping involves extracting data from websites. This can be accomplished using various tools and libraries available in Python. In this example, we'll focus on scraping information from the Times of India website using Python and the requests and BeautifulSoup libraries. Here's a step-by-step guide and the theory behind each step:

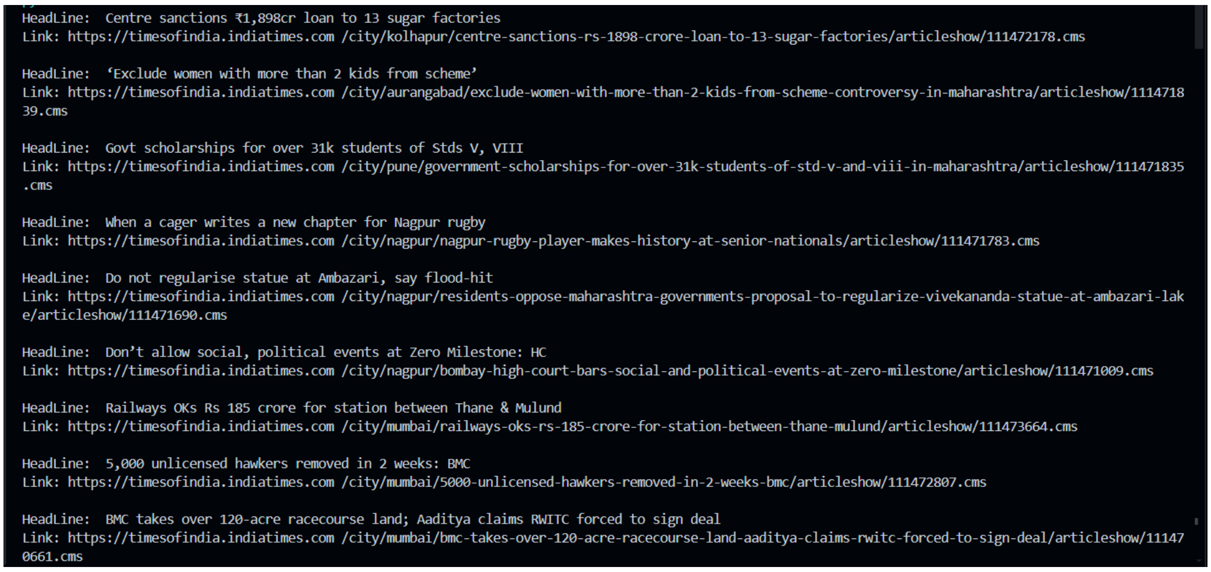
Understanding Web Scraping

Web scraping is the automated process of extracting information from websites. It involves making HTTP requests to web servers, retrieving the HTML content, and then parsing and extracting the desired data from that content.

Code: -

```
import requests
from bs4 import BeautifulSoup

#url = 'https://timesofindia.indiatimes.com/india'
url = 'https://timesofindia.indiatimes.com/india/maharashtra'
#headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
try:
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    headlines = soup.find_all('span', {'class': 'w_tle'})
    for headline in headlines:
        text = headline.text.strip()
        link = headline.find('a')['href']
        print("HeadLine: ",text)
        print("Link: https://timesofindia.indiatimes.com",link)
        print()
except Exception as e:
    print("Error:", e)
```

Output: -



```
HeadLine:  Centre sanctions ₹1,898cr loan to 13 sugar factories
Link: https://timesofindia.indiatimes.com /city/kolhapur/centre-sanctions-rs-1898-crore-loan-to-13-sugar-factories/articleshow/111472178.cms

HeadLine:  ‘Exclude women with more than 2 kids from scheme’
Link: https://timesofindia.indiatimes.com /city/aurangabad/exclude-women-with-more-than-2-kids-from-scheme-controversy-in-maharashtra/articleshow/1114718
39.cms

HeadLine:  Govt scholarships for over 31k students of Stds V, VIII
Link: https://timesofindia.indiatimes.com /city/pune/government-scholarships-for-over-31k-students-of-std-v-and-viii-in-maharashtra/articleshow/111471835
.cms

HeadLine:  When a cager writes a new chapter for Nagpur rugby
Link: https://timesofindia.indiatimes.com /city/nagpur/nagpur-rugby-player-makes-history-at-senior-nationals/articleshow/111471783.cms

HeadLine:  Do not regularise statue at Ambazari, say flood-hit
Link: https://timesofindia.indiatimes.com /city/nagpur/residents-oppose-maharashtra-governments-proposal-to-regularize-vivekananda-statue-at-ambazari-lak
e/articleshow/111471690.cms

HeadLine:  Don’t allow social, political events at Zero Milestone: HC
Link: https://timesofindia.indiatimes.com /city/nagpur/bombay-high-court-bars-social-and-political-events-at-zero-milestone/articleshow/111471009.cms

HeadLine:  Railways OKs Rs 185 crore for station between Thane & Mulund
Link: https://timesofindia.indiatimes.com /city/mumbai/railways-oks-rs-185-crore-for-station-between-thane-mulund/articleshow/111473664.cms

HeadLine:  5,000 unlicensed hawkers removed in 2 weeks: BMC
Link: https://timesofindia.indiatimes.com /city/mumbai/5000-unlicensed-hawkers-removed-in-2-weeks-bmc/articleshow/111472807.cms

HeadLine:  BMC takes over 120-acre racecourse land; Aaditya claims RWITC forced to sign deal
Link: https://timesofindia.indiatimes.com /city/mumbai/bmc-takes-over-120-acre-racecourse-land-aaditya-claims-rwitc-forced-to-sign-deal/articleshow/11147
0661.cms
```

Practical No: 3

Aim: Page Rank for link analysis using python Create a small set of pages namely page1, page2, page3 and page4 apply random walk on the same

Theory:

PageRank is an algorithm used by Google Search to rank web pages in their search engine results. It was developed by Larry Page and Sergey Brin, the founders of Google. The basic idea behind PageRank is that the importance of a webpage is determined by the number and quality of links pointing to it. Essentially, it views links to a page as votes of confidence.

Here are some key points about PageRank:

1. Link Voting: PageRank interprets a link from page A to page B as a vote by page A for page B. The more votes (links) a page receives, the higher its PageRank.
2. Quality Matters: Not all votes are equal. A link from a highly ranked page carries more weight than a link from a low-ranked page.
3. Algorithm Complexity: PageRank uses a complex algorithm that considers both the number of links and their quality. It also takes into account the PageRank of the pages that link to a particular page.
4. Iterative Process: The calculation of PageRank is iterative, meaning it's done repeatedly until the values converge to stable numbers. This ensures accuracy and consistency in ranking.
5. Not the Sole Factor: While PageRank was a significant factor in Google's early algorithm, today, Google uses a multitude of factors to rank search results, including relevance, freshness, and user context.

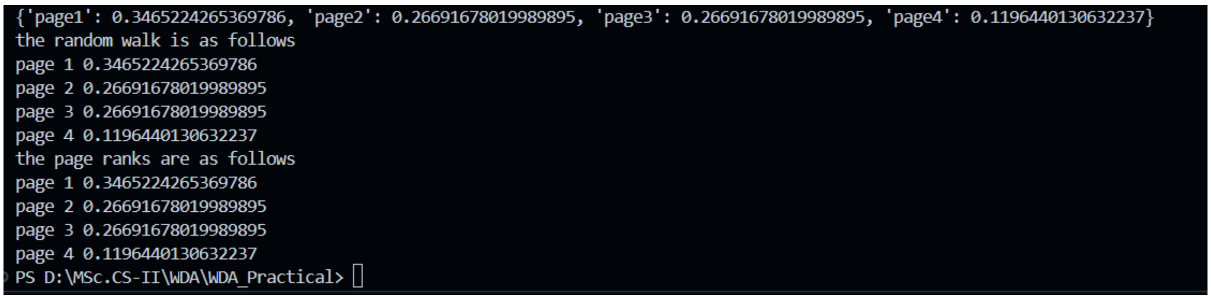
PageRank has had a profound impact on the way search engines rank web pages, influencing the development of SEO (Search Engine Optimization) strategies aimed at improving a website's visibility in search results.

Code: -

```
# pip install networkx

import networkx as nx
# create the graph
G = nx.DiGraph()
G.add_nodes_from(['page1', 'page2', 'page3', 'page4'])
G.add_edges_from([('page1', 'page2'), ('page1', 'page2'), ('page1', 'page3'), ('page3', 'page1')])

# calculate PageRank scores
pr = nx.pagerank(G, alpha=0.85)
rw = nx.pagerank(G)
```

```
# Print the random walk probability of each page
print(rw)
print("the random walk is as follows")
print("page 1",rw['page1'])
print("page 2",rw['page2'])
print("page 3",rw['page3'])
print("page 4",rw['page4'])

# print the results
print("the page ranks are as follows")
print("page 1",pr['page1'])
print("page 2",pr['page2'])
print("page 3",pr['page3'])
print("page 4",pr['page4'])
```

Output: -



```
{'page1': 0.3465224265369786, 'page2': 0.26691678019989895, 'page3': 0.26691678019989895, 'page4': 0.1196440130632237}
the random walk is as follows
page 1 0.3465224265369786
page 2 0.26691678019989895
page 3 0.26691678019989895
page 4 0.1196440130632237
the page ranks are as follows
page 1 0.3465224265369786
page 2 0.26691678019989895
page 3 0.26691678019989895
page 4 0.1196440130632237
PS D:\MSc.CS-II\WDA\WDA_Practical>
```

Practical No: 4

Aim: - Perform Spam Classifier

Theory: - A classifier, in the context of machine learning, is a computational model that assigns input data points to one of several predefined categories or classes. Its primary goal is to learn patterns from labelled training data and then use that knowledge to predict the class labels of new, unseen data points.

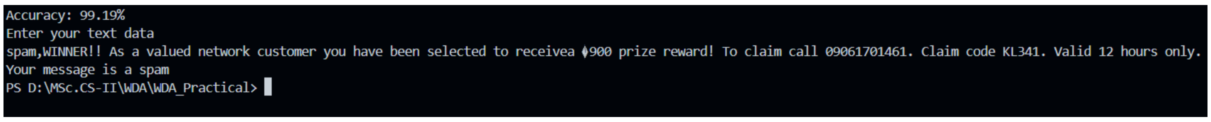
Classifiers are typically used in supervised learning scenarios where the training data is labelled with known outcomes (class labels). The classifier learns from these labelled examples to make predictions on new data.

Code: -

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

# Load the spam dataset
spam_df = pd.read_csv('spam_data.csv', encoding='latin-1')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(spam_df['text'], spam_df['label'], test_size=0.2,
random_state=42)

# Vectorize the text data using a Bag of Words model
vectorizer = CountVectorizer()
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)

# Train a Multinomial Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_vect, y_train)

# Evaluate the accuracy of the model on the test set
accuracy = clf.score(X_test_vect, y_test)
print("Accuracy: {:.2f}%".format(accuracy*100))

# Test the model on a new message
#You have won a prize! Click here to claim it now!
new_message = [input("Enter your text data\n")]
new_message_vect = vectorizer.transform(new_message)
prediction = clf.predict(new_message_vect)
```

```
if(prediction[0]=="spam"):
    print("Your message is a spam")
else:
    print("Your message is genuine")
```

Output: -



```
Accuracy: 99.19%
Enter your text data
spam,WINNER!! As a valued network customer you have been selected to receivea ⧫900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
Your message is a spam
PS D:\MSc.CS-II\WDA\WDA_Practical> █
```

Practical No: 5

Aim: - Demonstrate Text Mining and Webpage Pre-processing using meta information from the web pages (Local/Online).

Theory: -

Text mining involves extracting valuable information from text data, typically sourced from web pages, documents, or social media. This process includes steps like text extraction, preprocessing (e.g., tokenization, normalization), feature extraction (e.g., Bag of Words, TF-IDF), and analysis using techniques such as classification or clustering.

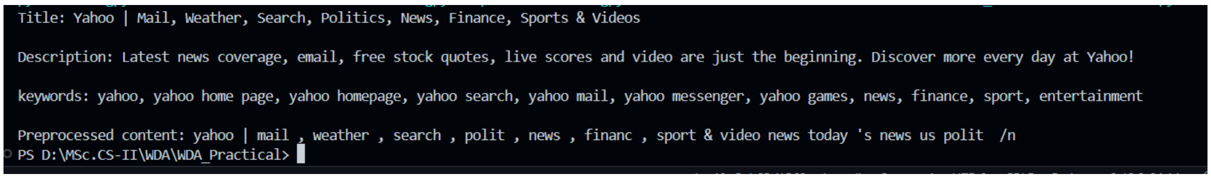
Webpage preprocessing involves extracting structured metadata (e.g., title, description, keywords) from web pages to enhance text mining efforts. This metadata provides context and enriches text data, aiding in better understanding and analysis. Techniques like web scraping, API integration, and browser automation are used to retrieve meta information efficiently and ethically.

Code: -

```
#pip install nltk---->has to be done in command prompt
#import nltk---->has to be done in python shell
#nltk.download('stopwords')---->has to be done in python shell
from bs4 import BeautifulSoup
import requests
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

# Download stopwords and initialize stemmer
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

# Retrieve webpage content
url = 'https://www.yahoo.com/?guccounter=1'
response = requests.get(url)

# Parse HTML and extract meta tags
soup = BeautifulSoup(response.content, 'html.parser')
meta_tags = soup.find_all('meta')

# Extract meta information
```

```
title = ''
description = ''
keywords = []

for tag in meta_tags:
        if tag.get('property')=='og:title':
           title=tag.get('content')

        if tag.get('property')=='og:description':
           discription=tag.get('content')

        if tag.get('name')=='keywords':
           keywords=tag.get('content')

# Preprocess content
content = soup.get_text()
tokens = word_tokenize(content)
filtered_tokens = [token.lower()for token in tokens if token.lower() not in stop_words]
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
preprocessed_content = ' '.join(stemmed_tokens)

# Print results
print('Title:', title,'\n')
print('Description:', discription,'\n')
print('keywords:', keywords,'\n')
print('Preprocessed content:', preprocessed_content[0:100],'/n')
```

Output: -



```
Title: Yahoo | Mail, Weather, Search, Politics, News, Finance, Sports & Videos

Description: Latest news coverage, email, free stock quotes, live scores and video are just the beginning. Discover more every day at Yahoo!

keywords: yahoo, yahoo home page, yahoo homepage, yahoo search, yahoo mail, yahoo messenger, yahoo games, news, finance, sport, entertainment

Preprocessed content: yahoo | mail , weather , search , polit , news , financ , sport & video news today 's news us polit  /n
PS D:\MSc.CS-II\WDA\WDA_Practical>
```

Practical No: 6

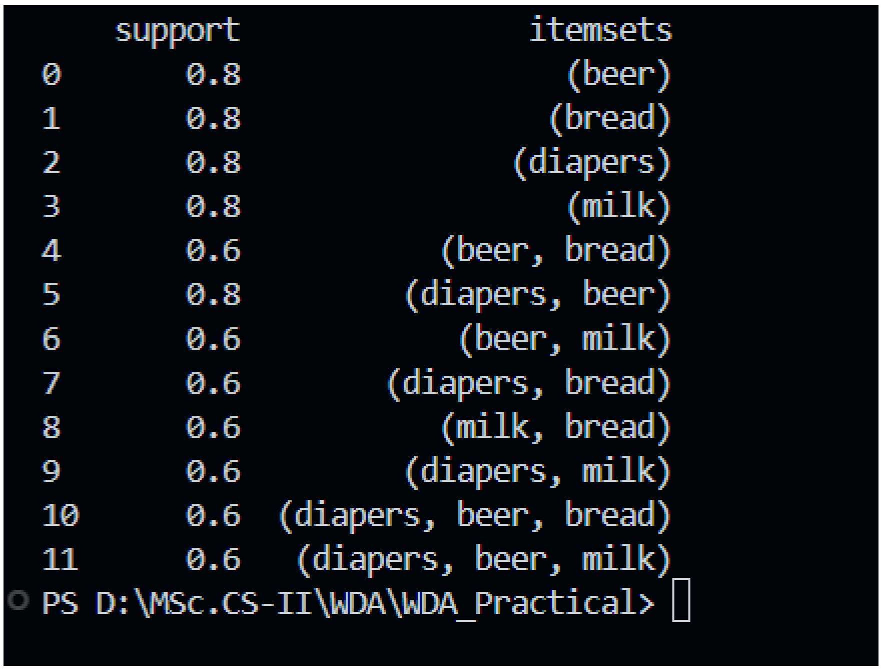
Aim: - Apriori Algorithm implementation in case study.

Theory: - Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

Code: -

```
import pandas as pd
import mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

# Example dataset
dataset = [['bread', 'milk', 'eggs'],
        ['bread', 'diapers', 'beer', 'eggs'],
        ['milk', 'diapers', 'beer', 'cola'],
        ['bread', 'milk', 'diapers', 'beer', 'cola'],
        ['bread', 'milk', 'diapers', 'beer']]

# Convert the dataset to a one-hot encoded matrix
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Apply the Apriori algorithm to generate frequent itemsets
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

# Print the frequent itemsets
print(frequent_itemsets)
```

Output: -



```
    support                  itemsets
0       0.8                    (beer)
1       0.8                   (bread)
2       0.8                 (diapers)
3       0.8                    (milk)
4       0.6             (beer, bread)
5       0.8           (diapers, beer)
6       0.6              (beer, milk)
7       0.6          (diapers, bread)
8       0.6             (milk, bread)
9       0.6           (diapers, milk)
10      0.6   (diapers, beer, bread)
11      0.6    (diapers, beer, milk)
PS D:\MSc.CS-II\WDA\WDA_Practical>
```

Practical No: 7

Aim: - Develop a basic crawler for the web search for user defined keywords.

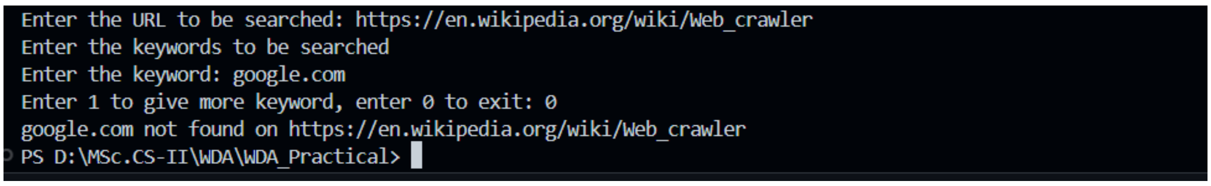
Theory: - A Web crawler, sometimes called a spider or spiderbot and often shortened to crawler, is an Internet bot that systematically browses the World Wide Web and that is typically operated by search engines for the purpose of Web indexing.

Code: -

```
import requests
from bs4 import BeautifulSoup

def web_crawler(url, keywords):
    # Make a request to the URL
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code == 200:
        # Parse the HTML content using BeautifulSoup
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find all the text on the page
        page_text = soup.get_text()

        # Check if any of the keywords are present on the page
        for keyword in keywords:
            if keyword.lower() in page_text.lower():
                print(f'{keyword} found on {url}')
            else:
                print(f'{keyword} not found on {url}')

    else:
        print(f'Request failed with status code {response.status_code}')

# Example usage
url = input("Enter the URL to be searched: ") #https://en.wikipedia.org/wiki/Web_crawler
keywords = []
print("Enter the keywords to be searched")
while True:
    k = input("Enter the keyword: ")
    keywords.append(k)
    x = int(input("Enter 1 to give more keyword, enter 0 to exit: "))
    if x == 0:
        break
```

```
web_crawler(url, keywords)
```

Output: -



```
Enter the URL to be searched: https://en.wikipedia.org/wiki/Web_crawler
Enter the keywords to be searched
Enter the keyword: google.com
Enter 1 to give more keyword, enter 0 to exit: 0
google.com not found on https://en.wikipedia.org/wiki/Web_crawler
PS D:\MSc.CS-II\WDA\WDA_Practical>
```

Practical No: 8

Aim: - Develop a programme for deep search implementation to detect plagiarism in documents online.

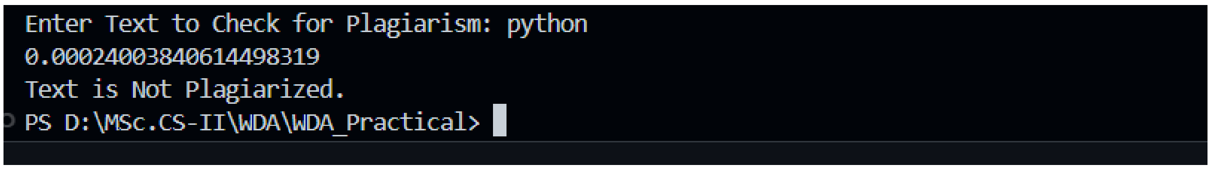
Theory: - Plagiarism checkers work by using advanced database software to scan for matches between your text and existing texts. Their accuracy is determined by two factors: the algorithm (which recognizes the plagiarism) and the size of the database (with which your document is compared).

Code: -

```
# Import Required Libraries
import requests
from bs4 import BeautifulSoup
import difflib
import nltk
#nltk.download('stopwords')
#nltk.download('punkt')
# Collect Data from User
text = input("Enter Text to Check for Plagiarism: ")

# Text Preprocessing

stop_words = set(nltk.corpus.stopwords.words('english'))
tokens = nltk.word_tokenize(text)
tokens = [token.lower() for token in tokens if token.isalpha()]
tokens = [token for token in tokens if token not in stop_words]

# Scrape Web for Similar Text
url = 'https://en.wikipedia.org/wiki/Python_(programming_language)'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
soup_text = soup.get_text()
soup_tokens = nltk.word_tokenize(soup_text)
soup_tokens = [token.lower() for token in soup_tokens if token.isalpha()]
soup_tokens = [token for token in soup_tokens if token not in stop_words]
# Calculate Similarity
similarity = difflib.SequenceMatcher(None, tokens, soup_tokens).ratio()

# Set Threshold for Plagiarism
threshold = 0.002
print(similarity)
# Plagiarism Detection
```

```
if similarity >= threshold:
    print("Text is Plagiarized.")
else:
    print("Text is Not Plagiarized.")
```

Output: -



```
Enter Text to Check for Plagiarism: python
0.00024003840614498319
Text is Not Plagiarized.
PS D:\MSc.CS-II\WDA\WDA_Practical>
```

Practical No: 9

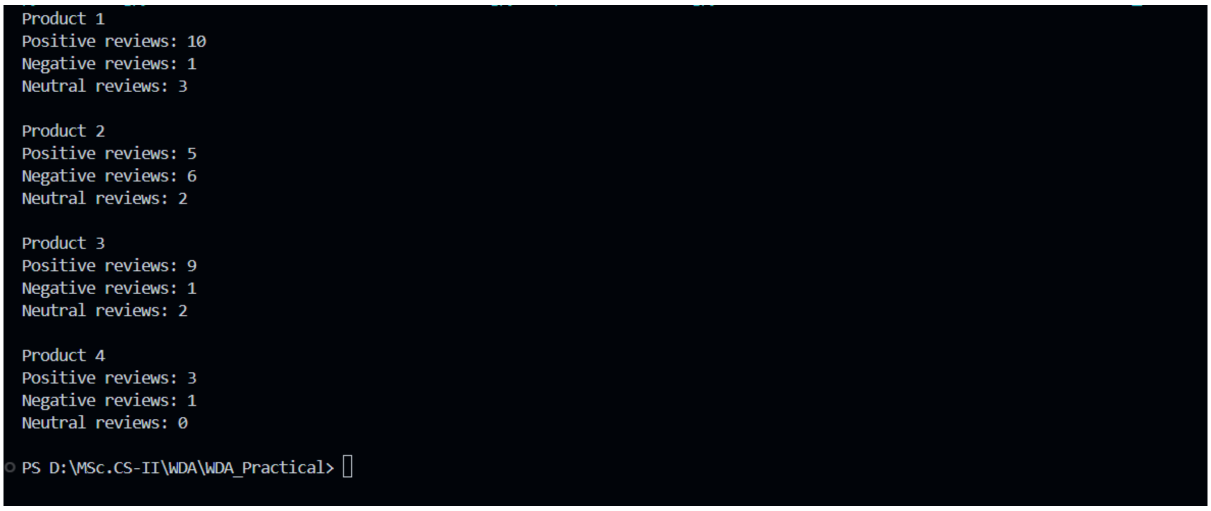
Aim: - Sentiment analysis for reviews by customers and visualize the same.

Theory: - Sentiment analysis is a natural language processing (NLP) technique used to determine the sentiment expressed in a piece of text. The goal of sentiment analysis is to identify and extract subjective information from the text, such as opinions, emotions, attitudes, and feelings expressed by the author or speaker.

Code: -

```
import pandas as pd
from textblob import TextBlob

# read in the customer_reviews.csv file as a Pandas dataframe
reviews_df = pd.read_csv('customer_reviews.csv')

# create a new column in the dataframe to hold the sentiment polarity score for each review
reviews_df['sentiment_score'] = reviews_df['review_text'].apply(lambda x: TextBlob(x).sentiment.polarity)

# categorize the sentiment scores into positive, negative, and neutral
reviews_df['sentiment_category'] = reviews_df['sentiment_score'].apply(lambda x: 'positive' if x > 0 else 'negative' if x < 0 else 'neutral')

# print out the count of reviews in each sentiment category for each product
for product_id in reviews_df['product_id'].unique():
    product_reviews_df = reviews_df[reviews_df['product_id'] == product_id]
    print('Product', product_id)
    print('Positive reviews:', len(product_reviews_df[product_reviews_df['sentiment_category'] == 'positive']))
    print('Negative reviews:', len(product_reviews_df[product_reviews_df['sentiment_category'] == 'negative']))
    print('Neutral reviews:', len(product_reviews_df[product_reviews_df['sentiment_category'] == 'neutral']))
    print()
```

Output: -



```
Product 1
Positive reviews: 10
Negative reviews: 1
Neutral reviews: 3

Product 2
Positive reviews: 5
Negative reviews: 6
Neutral reviews: 2

Product 3
Positive reviews: 9
Negative reviews: 1
Neutral reviews: 2

Product 4
Positive reviews: 3
Negative reviews: 1
Neutral reviews: 0

PS D:\MSc.CS-II\WDA\WDA_Practical>
```