B. K. Birla College of Arts, Science & Commerce (Autonomous), Kalyan
(Department of Computer Science)

SEMESTER: II

SUBJECT: WEB MINING

NAME:

CLASS: M.SC. COMPUTER SCIENCE PART-1

ROLL NO.:

B. K. Birla College of Arts, Science & Commerce (Autonomous), Kalyan
(Department of Computer Science)

CERTIFICATE

This is to certify that
Mr./Miss. ______________________________

*Roll No._____ Exam Seat No. ____________ has satisfactorily completed the Practical in **WEB MINING** as laid down in the regulation of University of Mumbai for the purpose of MSc Computer Science* Semester-II (Practical) *Examination* 2023-2024.

Date:

Place: Kalyan

____
Head
Department of Computer Science

Signature of Examiners

1) ____________________

2) ____________________

____
Professor In-Charge
Computer Science

INDEX

Practical 01

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

Background information:

Web scraping is the process of extracting data from websites. This can be done manually or through automated means, such as software programs or scripts. Scraping is a great way to collect large amounts of data quickly and easily. When projects require extracted data from hundreds or even thousands of web pages, automated web scraping tools can do the job more quickly and efficiently.

Code:

1st install following command in code or cmd:

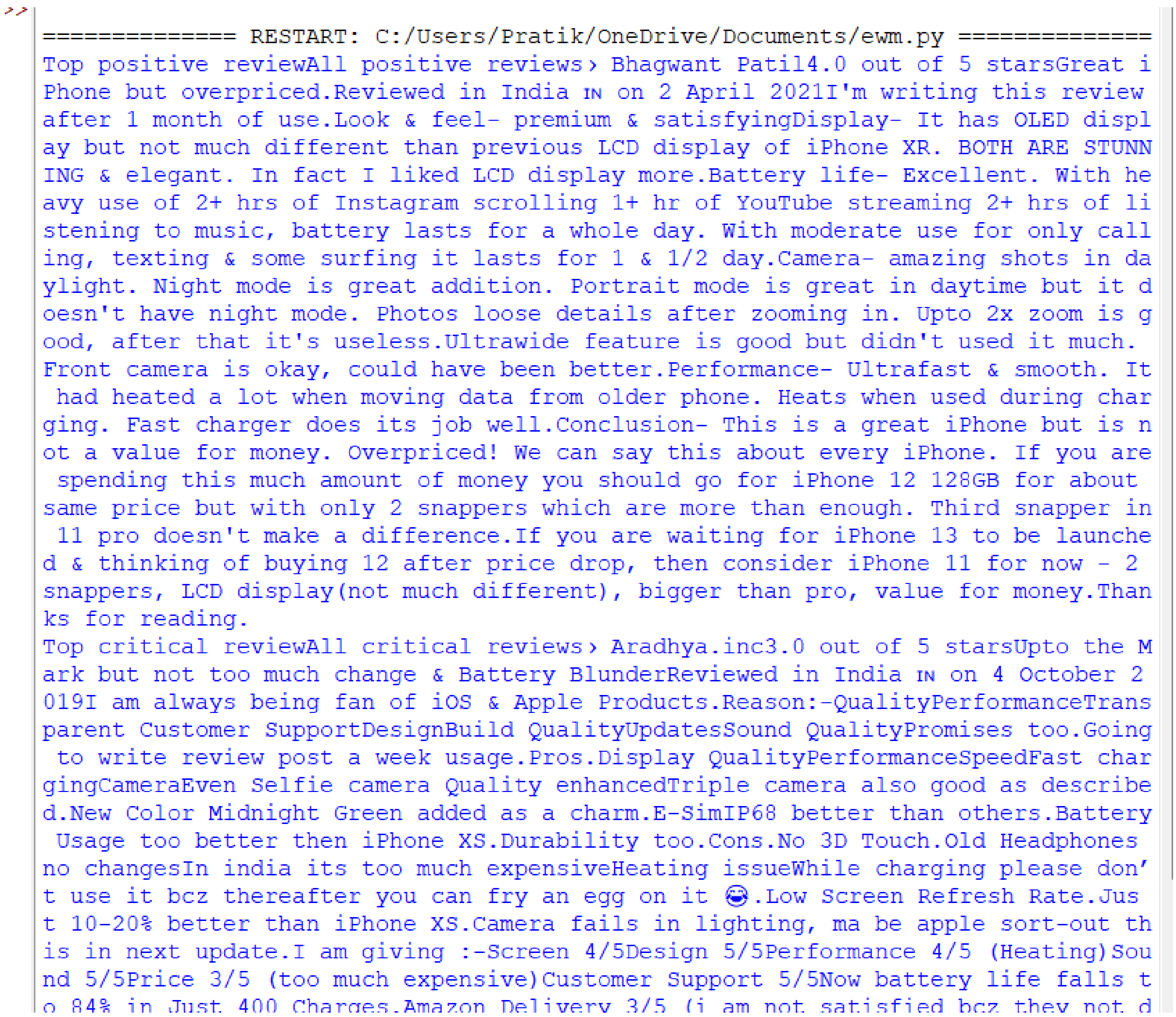
```
pip install beautifulsoup4
pip install requests
pip install pandas
```

Then implement following code:

```
# In[ ]:
import sys
import time
from bs4 import BeautifulSoup
import requests
import pandas as pd
# In[ ]:
try:
#use the browser to get the url. This is suspicious command that might blow
up.
    url = "https://www.amazon.in//Apple-iPhone-11-Pro-256GB//product-
reviews//B07XVMJF2D//ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_revi
ews"
    page=requests.get(url)
except Exception as e:
    error_type, error_obj, error_info = sys. exc_info()
    print ('ERROR FOR LINK:',url)
    print (error_type, 'Line:', error_info.tb_lineno)
time.sleep(2)
```

```
soup=BeautifulSoup (page.text,'html.parser')
links=soup.find_all('div',attrs={'class':'a-expander-content a-expander-
partial-collapse-content'})
# In[ ]:
soup
# In[ ]:
links
# In[ ]:
for i in links :
    print ( i.text )
print ( "\n" )
```

Output:



```
============== RESTART: C:/Users/Pratik/OneDrive/Documents/ewm.py ==============
Top positive reviewAll positive reviews› Bhagwant Patil4.0 out of 5 starsGreat i
Phone but overpriced.Reviewed in India IN on 2 April 2021I'm writing this review
after 1 month of use.Look & feel- premium & satisfyingDisplay- It has OLED displ
ay but not much different than previous LCD display of iPhone XR. BOTH ARE STUNN
ING & elegant. In fact I liked LCD display more.Battery life- Excellent. With he
avy use of 2+ hrs of Instagram scrolling 1+ hr of YouTube streaming 2+ hrs of li
stening to music, battery lasts for a whole day. With moderate use for only call
ing, texting & some surfing it lasts for 1 & 1/2 day.Camera- amazing shots in da
ylight. Night mode is great addition. Portrait mode is great in daytime but it d
oesn't have night mode. Photos loose details after zooming in. Upto 2x zoom is g
ood, after that it's useless.Ultrawide feature is good but didn't used it much.
Front camera is okay, could have been better.Performance- Ultrafast & smooth. It
 had heated a lot when moving data from older phone. Heats when used during char
ging. Fast charger does its job well.Conclusion- This is a great iPhone but is n
ot a value for money. Overpriced! We can say this about every iPhone. If you are
 spending this much amount of money you should go for iPhone 12 128GB for about
same price but with only 2 snappers which are more than enough. Third snapper in
 11 pro doesn't make a difference.If you are waiting for iPhone 13 to be launche
d & thinking of buying 12 after price drop, then consider iPhone 11 for now - 2
snappers, LCD display(not much different), bigger than pro, value for money.Than
ks for reading.
Top critical reviewAll critical reviews› Aradhya.inc3.0 out of 5 starsUpto the M
ark but not too much change & Battery BlunderReviewed in India IN on 4 October 2
019I am always being fan of iOS & Apple Products.Reason:-QualityPerformanceTrans
parent Customer SupportDesignBuild QualityUpdatesSound QualityPromises too.Going
 to write review post a week usage.Pros.Display QualityPerformanceSpeedFast char
gingCameraEven Selfie camera Quality enhancedTriple camera also good as describe
d.New Color Midnight Green added as a charm.E-SimIP68 better than others.Battery
 Usage too better then iPhone XS.Durability too.Cons.No 3D Touch.Old Headphones
no changesIn india its too much expensiveHeating issueWhile charging please don’
t use it bcz thereafter you can fry an egg on it 😂.Low Screen Refresh Rate.Jus
t 10-20% better than iPhone XS.Camera fails in lighting, ma be apple sort-out th
is in next update.I am giving :-Screen 4/5Design 5/5Performance 4/5 (Heating)Sou
nd 5/5Price 3/5 (too much expensive)Customer Support 5/5Now battery life falls t
o 84% in Just 400 Charges.Amazon Delivery 3/5 (i am not satisfied bcz they not d
```

Practical 02

Aim: Scrape an online Social Media Site for Data. Use python to scrape information from twitter.

Background information:

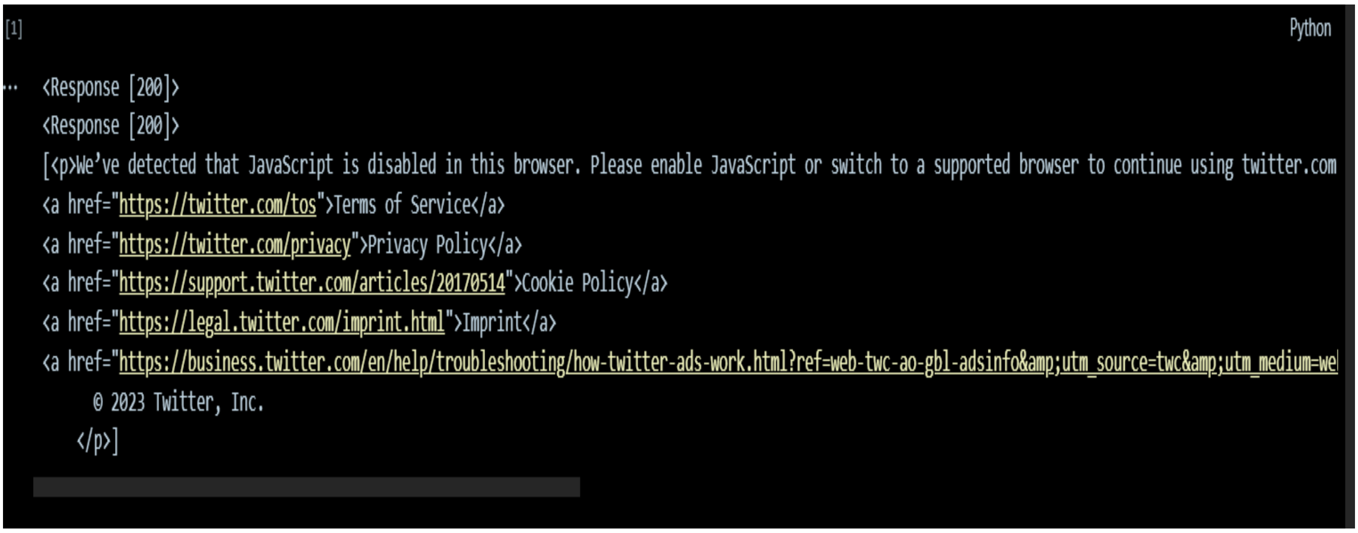
User-Generated content contains valuable information that is important to any business. It is not an easy task to extract data from social media websites such as Facebook, Twitter, Instagram, LinkedIn, Google Plus, etc. for any business user. The scraping of social media websites is the process of gathering, analyzing, and presenting actionable patterns and insights from social media data. The scraping of social media sites introduces basic ideas and important algorithms that are suitable for analyzing large amounts of social media data.

Code:

```
import pandas as pd
from bs4 import BeautifulSoup
import requests

url =
'https://twitter.com/BillGates?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor'
response = requests.get(url)
print(response)

if response.status_code == 200:
    print(response)

# This will store the HTML content as a stream of bytes:
html_content = response.content
# This will store the HTML content as a string:
html_content_string = response.text

soup = BeautifulSoup(html_content, 'html.parser')

soup

appPromoBanner = soup.find('div', {'class':'css-1dbjc4n'})
appPromoBanner
```

```
all_paragraphs = soup.findAll('p')
print(all_paragraphs[0:3]
```

Output:



```
··· <Response [200]>
    <Response [200]>
    [<p>We’ve detected that JavaScript is disabled in this browser. Please enable JavaScript or switch to a supported browser to continue using twitter.com
    <a href="https://twitter.com/tos">Terms of Service</a>
    <a href="https://twitter.com/privacy">Privacy Policy</a>
    <a href="https://support.twitter.com/articles/20170514">Cookie Policy</a>
    <a href="https://legal.twitter.com/imprint.html">Imprint</a>
    <a href="https://business.twitter.com/en/help/troubleshooting/how-twitter-ads-work.html?ref=web-twc-ao-gbl-adsinfo&amp;utm_source=twc&amp;utm_medium=we
        © 2023 Twitter, Inc.
      </p>]
```

Practical 03

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.

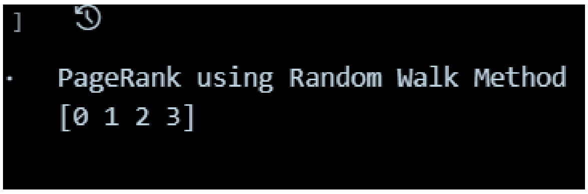
Background information:

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites. PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set.

Code:

```
import networkx as nx
import random
import numpy as np

# Add directed edges in graph
def add_edges(g, pr):
    for each in g.nodes():
        for each1 in g.nodes():
            if (each != each1):
                ra = random.random()
                if (ra < pr):
                    g.add_edge(each, each1)
                else:
                    continue
    return g

# Sort the nodes
def nodes_sorted(g, points):
    t = np.array(points)
    t = np.argsort(-t)
    return t

# Distribute points randomly in a graph
def random_Walk(g):
    rwp = [0 for i in range(g.number_of_nodes())]
    nodes = list(g.nodes())
    r = random.choice(nodes)
    rwp[r] += 1
    neigh = list(g.out_edges(r))
    z = 0

    while (z != 10000):
        if (len(neigh) == 0):
            focus = random.choice(nodes)
        else:
            r1 = random.choice(neigh)
            focus = r1[1]
        rwp[focus] += 1
        neigh = list(g.out_edges(focus))
        z += 1
    return rwp


# Main
# 1. Create a directed graph with N nodes
g = nx.DiGraph()
N = 4
g.add_nodes_from(range(N))

# 2. Add directed edges in graph
g = add_edges(g, 0.4)

# 3. perform a random walk
points = random_Walk(g)

# 4. Get nodes rank according to their random walk points
sorted_by_points = nodes_sorted(g, points)
print("PageRank using Random Walk Method")
print(sorted_by_points)
```

Output:



```
PageRank using Random Walk Method
[0 1 2 3]
```

Practical 04

Aim: Perform Spam Classifier.

Background information:

A spam message classification is a step towards building a tool for scam message identification and early scam detection. Many email services today provide spam filters that are able to classify emails into spam and non-spam email with high accuracy.

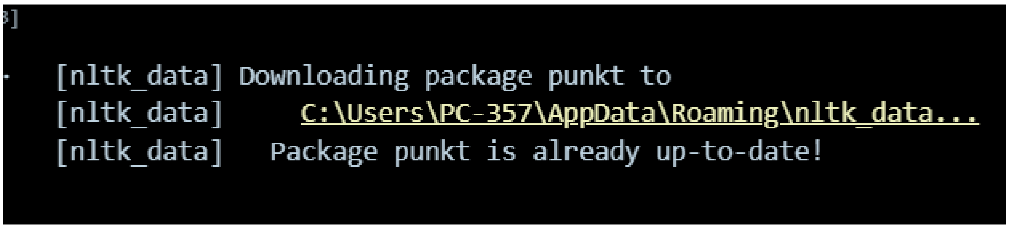
Code:

1st install following command in code or cmd:

```
pip install wordcloud
pip install pandas
pip install numpy
```

Run following code in vs code:

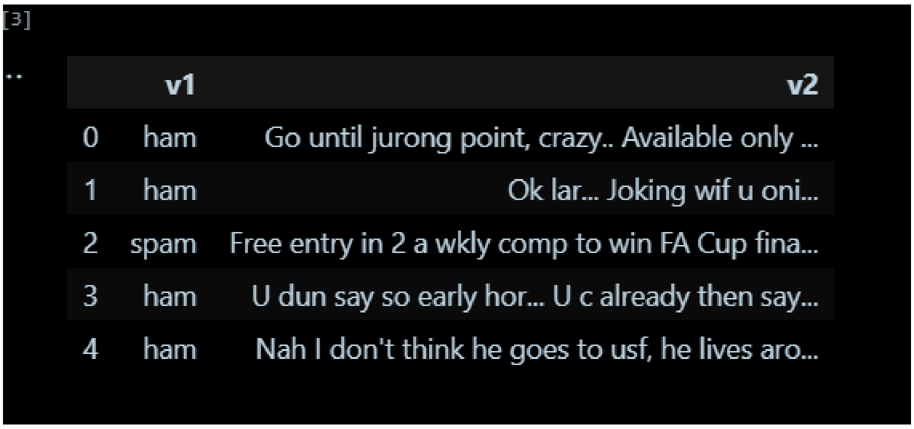
```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```



```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\PC-357\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from math import log, sqrt
import pandas as pd
import numpy as np
import re
```

```
from IPython import get_ipython

get_ipython().run_line_magic('matplotlib', 'inline')


mails = pd.read_csv(r"D:\MSc study material\SEM 2\WM\Pracs\spam.csv", 
encoding = 'latin-1')
mails.head()
```

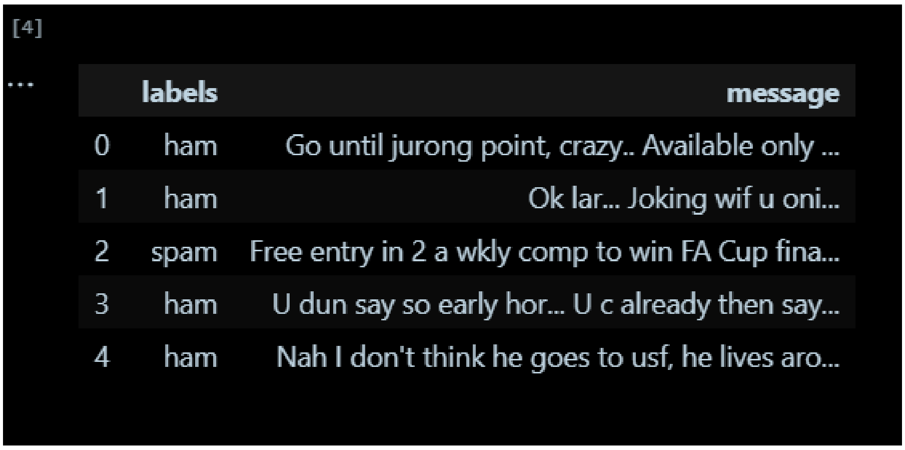


[3]

..

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
mails.rename(columns = {'v1': 'labels', 'v2': 'message'}, inplace = True)
mails.head()
```

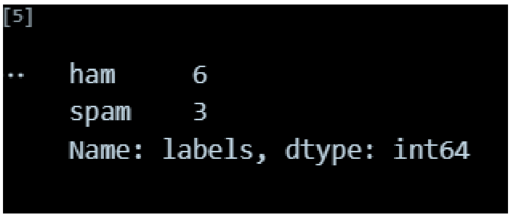


[4]

...

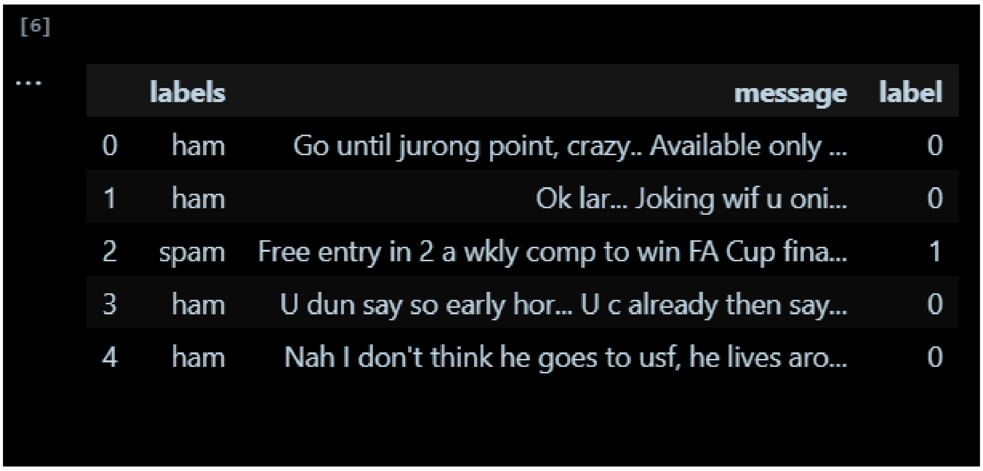
	labels	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
mails['labels'].value_counts()
```



```
[5]

..   ham     6
     spam    3
     Name: labels, dtype: int64
```

```
mails['label'] = mails['labels'].map({'ham': 0, 'spam': 1})
mails.head()
```



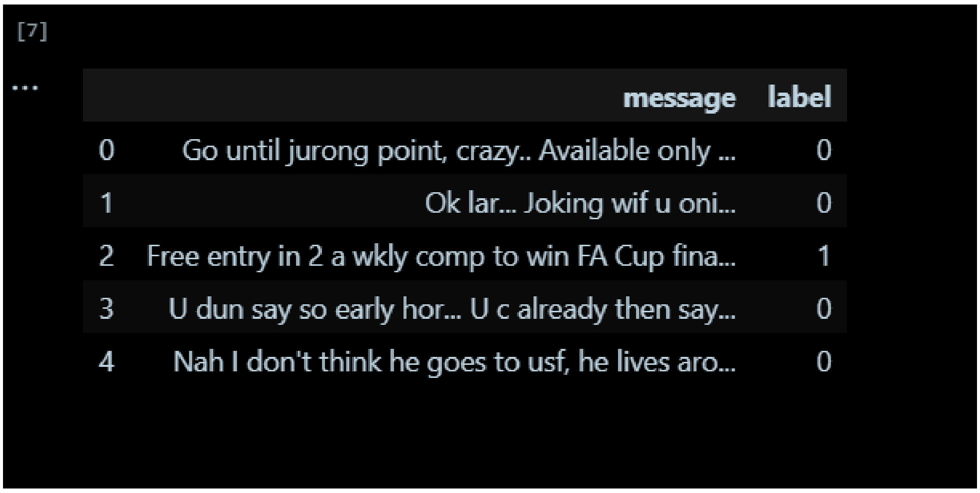
[6]

...

	labels	message	label
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
mails.drop(['labels'], axis = 1, inplace = True)
mails.head()
```

Output:



[7]

...

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	Ok lar... Joking wif u oni...	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	U dun say so early hor... U c already then say...	0
4	Nah I don't think he goes to usf, he lives aro...	0

Practical 05

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

Background information:

Text data mining is another name for text mining. The goal is to extract useful numerical indices from the text from the unstructured material. Make the text's information accessible to the different algorithms as a result. The documents' information can be extracted to create summaries. As a result, you can examine individual words and word groups in texts. Text mining, to put it simply, "turns text into numbers." such involves the use of unsupervised learning techniques in predictive data mining initiatives.

Code:

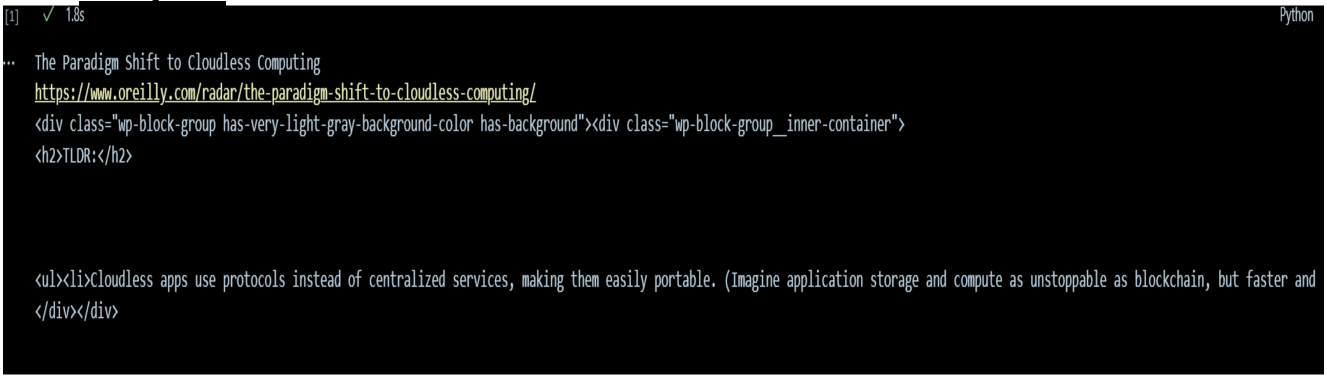
1st install following command in code or cmd:

```
pip install feedparser
```

Run following code in vs code:

```
import feedparser

FEED_URL='http://feeds.feedburner.com/oreilly/radar/atom'

fp = feedparser.parse(FEED_URL)

for e in fp.entries:
    print (e.title)
    print (e.links[0].href)
    print (e.content[0].value)
```

Output:



```
The Paradigm Shift to Cloudless Computing
https://www.oreilly.com/radar/the-paradigm-shift-to-cloudless-computing/
<div class="wp-block-group has-very-light-gray-background-color has-background"><div class="wp-block-group__inner-container">
<h2>TLDR:</h2>



<ul><li>Cloudless apps use protocols instead of centralized services, making them easily portable. (Imagine application storage and compute as unstoppable as blockchain, but faster and
</div></div>
```

Practical 06

Aim: Apriori Algorithm implementation in case study.

Background information:

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal and R Srikant and came to be known as Apriori. This algorithm uses two steps "join" and "prune" to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

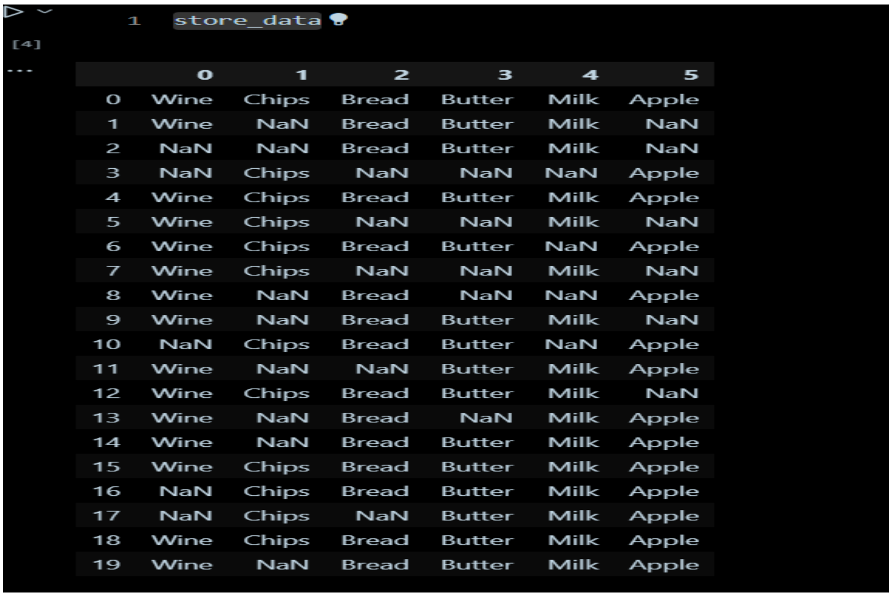
Code:

1st install following command in code or cmd:

```
pip install apyori
```

Run following code in vs code: import numpy as np

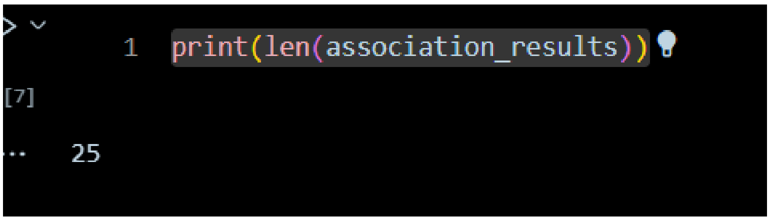
```
import numpy as np
import pandas as pd
from apyori import apriori

store_data = pd.read_csv('TransactionData.csv',header=None)

store_data
```



```
store_data
[4]
        0       1       2       3       4       5
 0   Wine   Chips   Bread  Butter    Milk   Apple
 1   Wine     NaN   Bread  Butter    Milk     NaN
 2    NaN     NaN   Bread  Butter    Milk     NaN
 3    NaN   Chips     NaN     NaN     NaN   Apple
 4   Wine   Chips   Bread  Butter    Milk   Apple
 5   Wine   Chips     NaN     NaN    Milk     NaN
 6   Wine   Chips   Bread  Butter     NaN   Apple
 7   Wine   Chips     NaN     NaN    Milk     NaN
 8   Wine     NaN   Bread     NaN     NaN   Apple
 9   Wine     NaN   Bread  Butter    Milk     NaN
10    NaN   Chips   Bread  Butter     NaN   Apple
11   Wine     NaN     NaN  Butter    Milk   Apple
12   Wine   Chips   Bread  Butter    Milk     NaN
13   Wine     NaN   Bread     NaN    Milk   Apple
14   Wine     NaN   Bread  Butter    Milk   Apple
15   Wine   Chips   Bread  Butter    Milk   Apple
16    NaN   Chips   Bread  Butter    Milk   Apple
17    NaN   Chips     NaN  Butter    Milk   Apple
18   Wine   Chips   Bread  Butter    Milk   Apple
19   Wine     NaN   Bread  Butter    Milk   Apple
```

```
records = []
for i in range(0,20):
    records.append([str(store_data.values[i,j]) for j in range(0,6)])

association_rules = apriori(records,min_support=0.50,min_confidence=0.7)
association_results = list(association_rules)
```

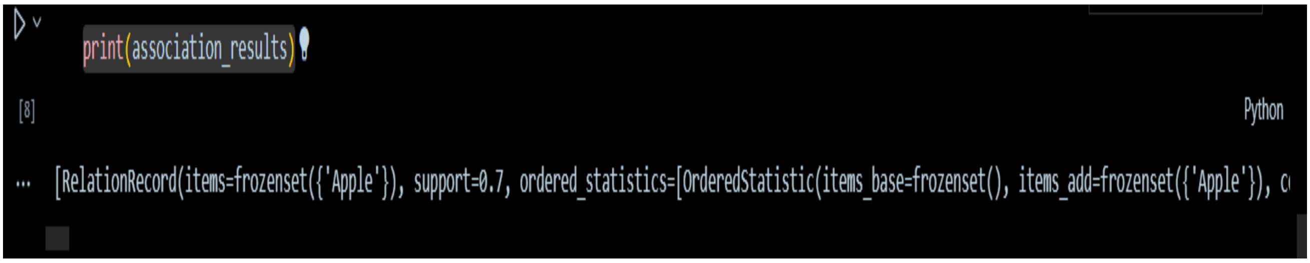
```
print(len(association_results))
```



```
print(len(association_results))
[7]
...   25
```

```
print(association_results)
```

Output:



```
print(association_results)
[8]                                                                          Python
... [RelationRecord(items=frozenset({'Apple'}), support=0.7, ordered_statistics=[OrderedStatistic(items_base=frozenset(), items_add=frozenset({'Apple'}), c
```

Practical 07

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

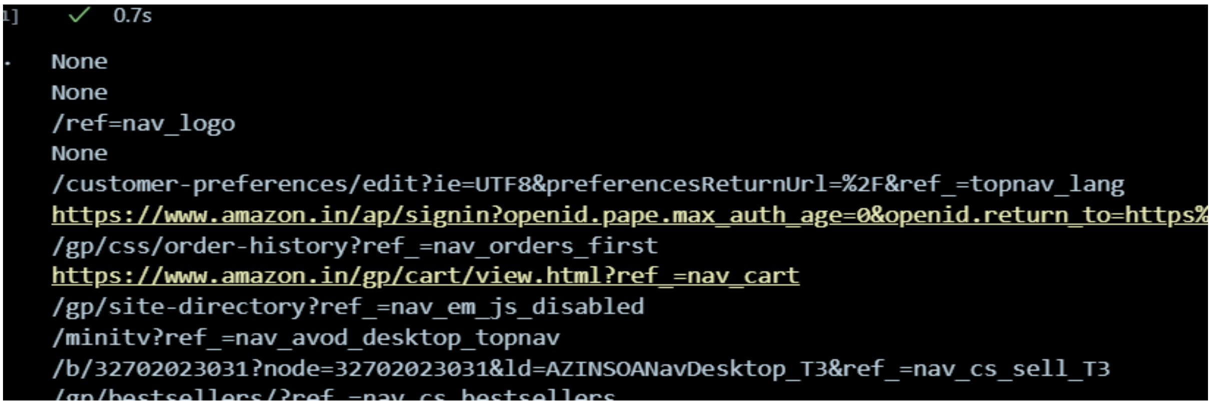
Background information:

A web crawler, or spider, is a type of bot that is typically operated by search engines like Google and Bing. Their purpose is to index the content of websites all across the Internet so that those websites can appear in search engine results.
Web crawling is a component of web scraping, the crawler logic finds URLs to be processed by the scraper code.A web crawler starts with a list of URLs to visit, called the seed. For each URL, the crawler finds links in the HTML, filters those links based on some criteria and adds the new links to a queue.

Code:

```
import requests
from bs4 import BeautifulSoup
url=("www.amazon.in")
code=requests.get("https://"+url)
plain=code.text
soup=BeautifulSoup(plain,"html.parser")
for link in soup.find_all('a'):
    print(link.get('href'))
```

Output:



```
1]    ✓  0.7s

·   None
    None
    /ref=nav_logo
    None
    /customer-preferences/edit?ie=UTF8&preferencesReturnUrl=%2F&ref_=topnav_lang
    https://www.amazon.in/ap/signin?openid.pape.max_auth_age=0&openid.return_to=https%
    /gp/css/order-history?ref_=nav_orders_first
    https://www.amazon.in/gp/cart/view.html?ref_=nav_cart
    /gp/site-directory?ref_=nav_em_js_disabled
    /minitv?ref_=nav_avod_desktop_topnav
    /b/32702023031?node=32702023031&ld=AZINSOANavDesktop_T3&ref_=nav_cs_sell_T3
    /gp/bestsellers/?ref_=nav_cs_bestsellers
```

Practical 08

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

Background information:

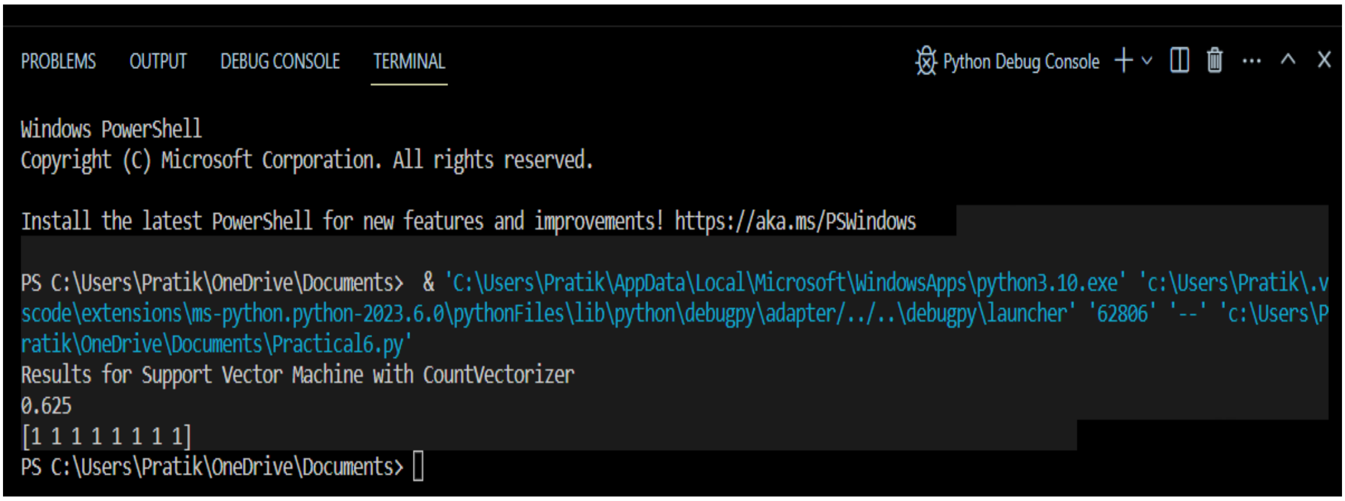
Sentiment analysis is the process of classifying whether a block of text is positive, negative, or, neutral. The goal which Sentiment analysis tries to gain is to be analyzed people’s opinions in a way that can help businesses expand. It focuses not only on polarity (positive, negative & neutral) but also on emotions (happy, sad, angry, etc.)

Code:

```
# Sentiment analysis for reviews by customers and visualize the same.

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import svm
import numpy as np

df = pd.read_csv('AmazonReview.csv')
#getting rid of null values
df = df.dropna()

#Taking a 30% representative sample
np.random.seed(34)
df1 =df.sample(frac = 0.3)

#Adding the sentiments column
df1['sentiments'] = df1.rating.apply(lambda x: 0 if x in [1, 2] else 1)

X = df1['review']
y = df1['sentiments']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05, random_state=0)

#Vectorizing the text data
```

```
cv = CountVectorizer()
ctmTr = cv.fit_transform(X_train)
X_test_dtm = cv.transform(X_test)

#Training the model
svcl = svm.SVC()
svcl.fit(ctmTr, y_train)
svcl_score = svcl.score(X_test_dtm, y_test)
print("Results for Support Vector Machine with CountVectorizer")
print(svcl_score)

y_pred_sv = svcl.predict(X_test_dtm)

#Conclusion matrix
print(y_pred_sv)
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Pratik\OneDrive\Documents>  & 'C:\Users\Pratik\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\Pratik\.v
scode\extensions\ms-python.python-2023.6.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '62806' '--' 'c:\Users\P
ratik\OneDrive\Documents\Practical6.py'
Results for Support Vector Machine with CountVectorizer
0.625
[1 1 1 1 1 1 1 1]
PS C:\Users\Pratik\OneDrive\Documents> 
```