Practical No. 01

Roll No: 03 **Date:** 07/04/2022

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.

Code:

```
!pip install kora -q
```

```
[ ] !pip install kora -q

    WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', por
    WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ReadTimeoutError("HTTPSConnectionPool(host='files.pythonhosted.org', por
       |████████████████████████████████| 57 kB 2.1 MB/s
       |████████████████████████████████| 56 kB 3.9 MB/s
```

```
import csv from bs4 import
BeautifulSoup from kora.selenium
import wd
wd.get('https://www.amazon.in/')
wd.page_source
```

```
'<html lang="en-in" class=" a-js a-audio a-video a-canvas a-svg a-drag-drop a-geolocation a-history a-webworker a-autofocus a-input-placeholder a-textarea-placeholder a-lo
cal-storage a-gradients a-transform3d a-touch-scrolling a-text-shadow a-text-stroke a-box-shadow a-border-radius a-border-image a-opacity a-transform a-transition null" da
ta-19ax5a9jf="dingo" data-aui-build-date="3.21.9-2022-04-02" data-useragent="Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/99.0.484
4.84 Safari/537.36" data-platform="Linux x86_64"><!-- sp:feature:head-start --><head><script async="" src="https://c.amazon-adsystem.com/bao-csm/forensics/a9-tq-forensics.
min.js" crossorigin="anonymous"></script><script async="" src="https://images-eu.ssl-images-amazon.com/images/I/31YXrY93hfL.js" crossorigin="anonymous"></script><script>va
r aPageStart = (new Date()).getTime();</script><meta charset="utf-8">\n<!-- sp:end-feature:head-start -->\n\n<script type="text/javascript">var ue…'
```

```
from selenium import webdriver options
= webdriver.ChromeOptions()
options.add_argument('-headless')
options.add_argument('-no-sandbox')
options.add_argument('-disable-dev-shm-usage')

wd = webdriver.Chrome('chromedriver',options=options)
wd.get("https://www.amazon.in/")
print(wd.page_source) # results
```

```
<html lang="en-in" class=" a-js a-audio a-video a-canvas a-svg a-drag-drop a-geolocation a-history a-webworker a-autofocus a-input-placeholder a-textarea-placeholder a-l
<!-- sp:end-feature:head-start -->

<script type="text/javascript">var ue_t0=ue_t0||+new Date();</script>
<!-- sp:feature:cs-optimization -->
<meta http-equiv="x-dns-prefetch-control" content="on">
<link rel="dns-prefetch" href="https://images-eu.ssl-images-amazon.com">
<link rel="dns-prefetch" href="https://m.media-amazon.com">
<link rel="dns-prefetch" href="https://completion.amazon.com">
<!-- sp:end-feature:cs-optimization -->
<script type="text/javascript">
window.ue_ihb = (window.ue_ihb || window.ueinit || 0) + 1;
if (window.ue_ihb === 1) {

var ue_csm = window,
    ue_hob = +new Date();
(function(d){var e=d.ue=d.ue||{},f=Date.now||function(){return+new Date};e.d=function(b){return f()-(b?0:d.ue_t0)};e.stub=function(b,a){if(!b[a]){var c=[];b[a]=function(


    var ue_err_chan = 'jserr-rw';
(function(d,e){function h(f,b){if(!(a.ec>a.mxe)&&f){a.ter.push(f);b=b||{};var c=f.logLevel||b.logLevel;c&&c!==k&&c!==m&&c!==n&&c!==p||a.ec++;c&&c!=k||a.ecf++;b.pageURL="
pec:0,ts:0,erl:[],ter:[],mxe:50,startTimer:function(){a.ts++;setInterval(function(){d.ue&&a.pec<a.ec&&d.uex("at");a.pec=a.ec},1E4)}};l.skipTrace=1;h.skipTrace=1;h.isStub


var ue_id = '8K4YEM9SF0PGRFH1VEV5',
    ue_url = '/rd/uedata',
    ue_navtiming = 1,
    ue_mid = 'A21TJRUUN4KGV',
    ue_sid = '260-3776493-5410318',
    ue_sn = 'www.amazon.in',
    ue_furl = 'fls-eu.amazon.in',
    ue_surl = 'https://unagi-eu.amazon.com/1/events/com.amazon.csm.nexusclient.prod',
    ue_int = 0.
```

```
def get_url(search_term):
  template = "https://www.amazon.in/s?k={}&rh=n%3A1389401031&ref=nb_sb_nos
s"   search_term = search_term.replace(' ','+')
  return template.format(search_term)

url = get_url('mobiles') print(url)
```

```
https://www.amazon.in/s?k=mobiles&rh=n%3A1389401031&ref=nb_sb_noss
```

```
wd.get(url)
soup = BeautifulSoup(wd.page_source, 'html.parser')

result     =     soup.find_all('div',{'data-component-type':'s-search-result'})
len(result)
```

```
30
```

```
item = result[1]

atag = item.h2.a

atag.text
```

```
'Vivo Y73 (Roman Black, 8GB RAM, 128GB Storage) with No Cost EMI/Additional Exchange Offers
'
```

```
price_parent = item.find('span','a-price') price_parent.find('span','a-
offscreen').text
```

```
'₹19,990'
```

```
 rating = item.i.text
print(rating)
```

```
4.4 out of 5 stars
```

```
def extract_record(item1):
  atag = item1.h2.a   description
= atag.text.strip()
 # url = "https://www.amazon.in/" + atag.get('href')
   price_parent = item1.find('span','a-price')
price_parent.find('span','a-offscreen').text
   rating = ""   result = (description,
price_parent, rating)   return result

records = []
results = soup.find_all('div',{'data-component-type':'s-search-result'})

for item in results:
  records.append(extract_record(item))
 records[0]
```

```
('Samsung Galaxy M12 (Blue,4GB RAM, 64GB Storage) 6000 mAh with 8nm Processor | True 48 MP
 <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹11,4
 '')
```

```
for x in range(len(records)):
print(records[x])
```

```
('Samsung Galaxy M12 (Blue,4GB RAM, 64GB Storage) 6000 mAh with 8nm Processor | True 48 MP Quad Camera | 90Hz Refresh Rate', <span class="a-price" data-a-color="price" dat
('Vivo Y73 (Roman Black, 8GB RAM, 128GB Storage) with No Cost EMI/Additional Exchange Offers', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-of
('Redmi 9 (Sky Blue, 4GB RAM, 64GB Storage) | 2.3GHz Mediatek Helio G35 Octa core Processor', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-off
('OPPO A31 (Fantasy White, 6GB RAM, 128GB Storage) with No Cost EMI/Additional Exchange Offers', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-
('Samsung Galaxy M12 (Blue,4GB RAM, 64GB Storage) 6000 mAh with 8nm Processor | True 48 MP Quad Camera | 90Hz Refresh Rate', <span class="a-price" data-a-color="price" dat
('OPPO A31 (Mystery Black, 6GB RAM, 128GB Storage) with No Cost EMI/Additional Exchange Offers', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-
('Redmi 9 Activ (Metallic Purple, 4GB RAM, 64GB Storage)', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹9,499</span><span aria-hid
('Redmi Note 10T 5G (Metallic Blue, 4GB RAM, 64GB Storage) | Dual 5G | 90Hz Adaptive Refresh Rate | MediaTek Dimensity 700 7nm Processor', <span class="a-price" data-a-col
('Redmi Note 10S (Frost White, 6GB RAM, 64GB Storage) - Super Amoled Display | 64 MP Quad Camera | Alexa Built in', <span class="a-price" data-a-color="price" data-a-size=
('realme narzo 50i (Carbon Black, 2GB RAM+32GB Storage) - with No Cost EMI/Additional Exchange Offers', <span class="a-price" data-a-color="price" data-a-size="l"><span cl
('Redmi 9A(Midnight Black 3GB RAM 32GB Storage) | 2GHz Octa-core Helio G25 Processor | 5000 mAh Battery', <span class="a-price" data-a-color="price" data-a-size="l"><span
('Samsung Galaxy M32 5G (Sky Blue, 6GB RAM, 128GB Storage)', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹16,999</span><span aria-
('Samsung Galaxy M12 (White,4GB RAM, 64GB Storage) 6000 mAh with 8nm Processor | True 48 MP Quad Camera | 90Hz Refresh Rate', <span class="a-price" data-a-color="price" da
('Redmi 10 Prime (Phantom Black 4GB RAM 64GB | Helio G88 with extendable RAM Upto 2GB | FHD+ 90Hz Adaptive Sync Display)', <span class="a-price" data-a-color="price" data-
('Redmi Note 11 (Horizon Blue, 6GB RAM, 128GB Storage)|90Hz FHD+ AMOLED Display | Qualcomm® Snapdragon™ 680-6nm | Alexa Built-in', <span class="a-price" data-a-color="pric
('Samsung Galaxy M32 (Light Blue, 4GB RAM, 64GB Storage) 6 Months Free Screen Replacement for Prime', <span class="a-price" data-a-color="price" data-a-size="l"><span clas
('Samsung Galaxy M32 5G (Slate Black, 6GB RAM, 128GB Storage)', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹16,999</span><span ar
('Samsung Galaxy M12 (Black,4GB RAM, 64GB Storage) 6000 mAh with 8nm Processor | True 48 MP Quad Camera | 90Hz Refresh Rate', <span class="a-price" data-a-color="price" da
('Redmi Note 11T 5G (Stardust White 8GB RAM 128GB ROM) | Dimensity 810 5G | 33W Pro Fast Charging', <span class="a-price" data-a-color="price" data-a-size="l"><span class=
('Samsung Galaxy M12 (Blue,6GB RAM, 128GB Storage) 6 Months Free Screen Replacement for Prime', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-o
('Redmi Note 10S (Deep Sea Blue, 6GB RAM, 64GB Storage) -Super Amoled Display', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹14,99
('realme narzo 50A (Oxygen Green , 4GB RAM + 128 GB Storage) Helio G85 Processor | 50MP AI Triple Camera | 6000 mAh Battery', <span class="a-price" data-a-color="price" da
('Redmi 9A Sport (Coral Green, 2GB RAM, 32GB Storage) | 2GHz Octa-core Helio G25 Processor | 5000 mAh Battery', <span class="a-price" data-a-color="price" data-a-size="l">
('Samsung Galaxy M32 (Black, 6GB RAM, 128GB Storage) 6 Months Free Screen Replacement for Prime', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a
('OnePlus Nord CE 2 5G (Bahamas Blue, 6GB RAM, 128GB Storage)', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹23,999</span><span ar
('realme narzo 50A (Oxygen Green , 4GB RAM + 64 GB Storage) Helio G85 Processor | 50MP AI Triple Camera | 6000 mAh Battery', <span class="a-price" data-a-color="price" dat
('realme narzo 50 (Speed Blue, 6GB RAM+128GB Storage) Helio G96 Processor | 50MP AI Triple Camera | 120Hz Ultra Smooth Display', <span class="a-price" data-a-color="price"
('realme narzo 50A (Oxygen Blue , 4GB RAM + 128 GB Storage) Helio G85 Processor | 50MP AI Triple Camera | 6000 mAh Battery', <span class="a-price" data-a-color="price" dat
('I KALL Z1 Smartphone (Grey, 5.5 Inch, 4GB, 32G) | 4G Volte | Android 8.1', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹5,399</s
('I KALL Z1 Smartphone (4GB, 32G) (Grey)', <span class="a-price" data-a-color="price" data-a-size="l"><span class="a-offscreen">₹5,399</span><span aria-hidden="true"><span
```

Practical No. 02

Roll No :03 **Date:** 16/04 /2022

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.

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 = 15

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
[ 5 14  4  2  1  8  0  9 11  6 13  3 12 10  7]
```

p_dict is dictionary of tuples p_dict

= nx.pagerank(g)

p_sort = sorted(p_dict.items(), key=lambda x: x[1], reverse=True)

print("PageRank using inbuilt pagerank method") for i in p_sort:

print(i[0], end=", ")

Output:-

```
PageRank using inbuilt pagerank method
5, 14, 2, 1, 4, 8, 0, 9, 11, 13, 6, 3, 12, 10, 7,
```

Practical No.03

Roll No :03 **Date:** 16/04/2022

Aim: Perform Spam Classifier.

Code:-

```
# Commented out IPython magic to ensure Python compatibility.
#Importing the colab drive library to get the dataset
import numpy as np import pandas as pd import
matplotlib.pyplot as plt import seaborn as sns
import scipy as sp from google.colab import drive

from sklearn import feature_extraction, model_selection, naive_bayes, metrics, svm
from sklearn.ensemble import RandomForestClassifier from
sklearn.model_selection import train_test_split

from sklearn.metrics import precision_recall_fscore_support as score
# %matplotlib inline drive.mount('/content/drive')
```

OUTPUT:-

```
Mounted at /content/drive
```

```
dataset = pd.read_csv("/content/drive/My Drive/spam/spam.csv", encoding='latin-1') dataset.head()
```

OUTPUT:-

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

```
#removing unnamed columns dataset =
dataset.drop('Unnamed: 2', 1) dataset =
dataset.drop('Unnamed: 3', 1) dataset =
dataset.drop('Unnamed: 4', 1)
```

OUTPUT:-

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argume

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argume
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argume
  after removing the cwd from sys.path.
```

dataset.head()

OUTPUT:-

	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...

dataset = dataset.rename(columns = {'v1':'label','v2':'message'}) dataset.groupby('label').describe()

OUTPUT:-

	message			
	count	unique	top	freq
label				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

dataset.head(4) **OUTPUT:-**

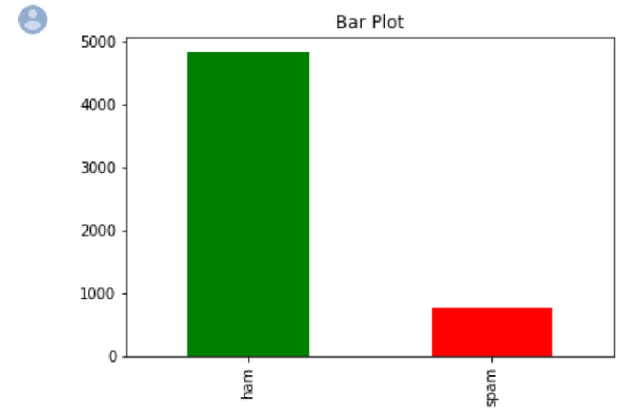
	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...

count_Class=pd.value_counts(dataset["label"], sort= True)

count_Class.plot(kind = 'bar',color = ["green","red"])

plt.title('Bar Plot') plt.show();

OUTPUT:-



f = feature_extraction.text.CountVectorizer(stop_words = 'english')

X = f.fit_transform(dataset["message"]) np.shape(X)

OUTPUT:-

```
(5572, 8404)
```

Classifying spam and not spam msgs as 1 and 0

dataset["label"]=dataset["label"].map({'spam':1,'ham':0})

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, dataset['label'], test_size=0.70, random_state=42) list_alpha = np.arange(1/100000, 20, 0.11) score_train = np.zeros(len(list_alpha)) score_test = np.zeros(len(list_alpha)) recall_test = np.zeros(len(list_alpha)) precision_test= np.zeros(len(list_alpha)) count = 0 for alpha in list_alpha:

bayes = naive_bayes.MultinomialNB(alpha=alpha) bayes.fit(X_train, y_train)

score_train[count] = bayes.score(X_train, y_train) score_test[count]= bayes.score(X_test, y_test)

recall_test[count] = metrics.recall_score(y_test, bayes.predict(X_test)) precision_test[count] = metrics.precision_score(y_test, bayes.predict(X_test)) count = count + 1

matrix = np.matrix(np.c_[list_alpha, score_train, score_test, recall_test, precision_test]) models = pd.DataFrame(data = matrix, columns =

['alpha', 'Train Accuracy', 'Test Accuracy', 'Test Recall', 'Test Precision']) models.head(n=10)

OUTPUT:-

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	0.998803	0.961805	0.913793	0.820998
1	0.11001	0.998803	0.966163	0.946360	0.826087
2	0.22001	0.999402	0.967444	0.938697	0.837607
3	0.33001	0.999402	0.968726	0.938697	0.844828
4	0.44001	0.999402	0.971546	0.929119	0.867621
5	0.55001	0.998803	0.976160	0.925287	0.899441
6	0.66001	0.998803	0.976160	0.919540	0.903955
7	0.77001	0.997606	0.977698	0.917625	0.915870
8	0.88001	0.997606	0.977954	0.909962	0.924125
9	0.99001	0.997606	0.978980	0.902299	0.938247

best_index = models['Test Precision'].idxmax()

models.iloc[best_index, :] **OUTPUT:-**

```
alpha             10.670010
Train Accuracy     0.977259
Test Accuracy      0.962574
Test Recall        0.720307
Test Precision     1.000000
Name: 97, dtype: float64
```

rf = RandomForestClassifier(n_estimators=100,max_depth=None,n_jobs=-1)

rf_model = rf.fit(X_train,y_train) y_pred=rf_model.predict(X_test)

precision,recall,fscore,support =score(y_test,y_pred,pos_label=1, average ='binary') print('Precision

: {} / Recall : {} / fscore : {} / Accuracy:

{} '.format(round(precision,3),round(recall,3),round(fscore,3),round((y_pred==y_test).sum()/len(y _test),3)))

OUTPUT:-

```
Precision : 0.995 / Recall : 0.722 / fscore : 0.837 / Accuracy: 0.962
```

!pip install keras.utils

OUTPUT:-

```
Collecting keras.utils
  Downloading keras-utils-1.0.13.tar.gz (2.4 kB)
Requirement already satisfied: Keras>=2.1.5 in /usr/local/lib/python3.7/dist-packages (from keras.utils) (2.8.0)
Building wheels for collected packages: keras.utils
  Building wheel for keras.utils (setup.py) ... done
  Created wheel for keras.utils: filename=keras_utils-1.0.13-py3-none-any.whl size=2656 sha256=efae3711676742ecb3fd1e94c61ee820a4c29cae87caae061075c59d7c36f0df
  Stored in directory: /root/.cache/pip/wheels/d0/dd/3b/493952a5240d486a83805d65360dedadbadeae71d25e2c877f
Successfully built keras.utils
Installing collected packages: keras.utils
Successfully installed keras.utils-1.0.13
```

import tensorflow as tf

from keras.preprocessing.text import Tokenizer from

keras.layers import Embedding, LSTM, Dropout, Dense from

keras.models import Sequential from tensorflow.keras.utils

import to_categorical

#from keras.utils import to_categorical

from keras.preprocessing.sequence import pad_sequences
import tensorflow as tf

vocab_size = 400 oov_tok

= "<OOV>" max_length =

250 embedding_dim = 16

encode = ({'ham': 0, 'spam': 1}) #new

dataset with replaced values dataset

= dataset.replace(encode)

X = dataset['message'] Y

= dataset['label']

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok) tokenizer.fit_on_texts(X)

convert to sequence of integers

X = tokenizer.texts_to_sequences(X)

X = np.array(X) y = np.array(Y)

OUTPUT:-

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tu
  """Entry point for launching an IPython kernel.
```

X = pad_sequences(X, maxlen=max_length) model

= tf.keras.Sequential([

tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length), tf.keras.layers.GlobalAveragePooling1D(), tf.keras.layers.Dense(24, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')

])

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.summary() **OUTPUT:-**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 250, 16)           6400

 global_average_pooling1d (G  (None, 16)               0
 lobalAveragePooling1D)

 dense (Dense)               (None, 24)                408

 dense_1 (Dense)             (None, 1)                 25

=================================================================
Total params: 6,833
Trainable params: 6,833
Non-trainable params: 0
_________________________________________________________________
```

num_epochs = 50

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.20, random_state=7) history = model.fit(X_train, y_train, epochs=num_epochs, validation_data=(X_test,y_test), verbose=2)

OUTPUT:-

```
Epoch 1/50
140/140 - 4s - loss: 0.5501 - accuracy: 0.8344 - val_loss: 0.4011 - val_accuracy: 0.8700 - 4s/epoch - 25ms/step
Epoch 2/50
140/140 - 1s - loss: 0.3870 - accuracy: 0.8649 - val_loss: 0.3697 - val_accuracy: 0.8700 - 916ms/epoch - 7ms/step
Epoch 3/50
140/140 - 1s - loss: 0.3734 - accuracy: 0.8649 - val_loss: 0.3594 - val_accuracy: 0.8700 - 1s/epoch - 7ms/step
Epoch 4/50
140/140 - 1s - loss: 0.3619 - accuracy: 0.8649 - val_loss: 0.3462 - val_accuracy: 0.8700 - 806ms/epoch - 6ms/step
Epoch 5/50
140/140 - 1s - loss: 0.3452 - accuracy: 0.8649 - val_loss: 0.3265 - val_accuracy: 0.8700 - 764ms/epoch - 5ms/step
Epoch 6/50
140/140 - 1s - loss: 0.3192 - accuracy: 0.8649 - val_loss: 0.2942 - val_accuracy: 0.8700 - 826ms/epoch - 6ms/step
Epoch 7/50
140/140 - 1s - loss: 0.2794 - accuracy: 0.8649 - val_loss: 0.2496 - val_accuracy: 0.8700 - 768ms/epoch - 5ms/step
Epoch 8/50
140/140 - 1s - loss: 0.2333 - accuracy: 0.8649 - val_loss: 0.2072 - val_accuracy: 0.8700 - 736ms/epoch - 5ms/step
Epoch 9/50
140/140 - 0s - loss: 0.2023 - accuracy: 0.8649 - val_loss: 0.1846 - val_accuracy: 0.8700 - 436ms/epoch - 3ms/step
Epoch 10/50
140/140 - 0s - loss: 0.1802 - accuracy: 0.9266 - val_loss: 0.1593 - val_accuracy: 0.9498 - 464ms/epoch - 3ms/step
Epoch 11/50
140/140 - 0s - loss: 0.1535 - accuracy: 0.9524 - val_loss: 0.1344 - val_accuracy: 0.9578 - 439ms/epoch - 3ms/step
Epoch 12/50
140/140 - 0s - loss: 0.1323 - accuracy: 0.9603 - val_loss: 0.1168 - val_accuracy: 0.9623 - 459ms/epoch - 3ms/step
Epoch 13/50
140/140 - 0s - loss: 0.1168 - accuracy: 0.9634 - val_loss: 0.1040 - val_accuracy: 0.9641 - 428ms/epoch - 3ms/step
Epoch 14/50
140/140 - 0s - loss: 0.1045 - accuracy: 0.9668 - val_loss: 0.0924 - val_accuracy: 0.9650 - 417ms/epoch - 3ms/step
Epoch 15/50
```

```
140/140 - 0s - loss: 0.0430 - accuracy: 0.9861 - val_loss: 0.0448 - val_accuracy: 0.9874 - 418ms/epoch - 3ms/step
Epoch 37/50
140/140 - 0s - loss: 0.0427 - accuracy: 0.9854 - val_loss: 0.0427 - val_accuracy: 0.9892 - 430ms/epoch - 3ms/step
Epoch 38/50
140/140 - 0s - loss: 0.0423 - accuracy: 0.9856 - val_loss: 0.0423 - val_accuracy: 0.9901 - 457ms/epoch - 3ms/step
Epoch 39/50
140/140 - 0s - loss: 0.0409 - accuracy: 0.9863 - val_loss: 0.0424 - val_accuracy: 0.9883 - 443ms/epoch - 3ms/step
Epoch 40/50
140/140 - 0s - loss: 0.0403 - accuracy: 0.9854 - val_loss: 0.0418 - val_accuracy: 0.9901 - 406ms/epoch - 3ms/step
Epoch 41/50
140/140 - 0s - loss: 0.0392 - accuracy: 0.9859 - val_loss: 0.0417 - val_accuracy: 0.9901 - 438ms/epoch - 3ms/step
Epoch 42/50
140/140 - 0s - loss: 0.0385 - accuracy: 0.9863 - val_loss: 0.0418 - val_accuracy: 0.9874 - 419ms/epoch - 3ms/step
Epoch 43/50
140/140 - 0s - loss: 0.0387 - accuracy: 0.9868 - val_loss: 0.0412 - val_accuracy: 0.9901 - 443ms/epoch - 3ms/step
Epoch 44/50
140/140 - 0s - loss: 0.0378 - accuracy: 0.9870 - val_loss: 0.0411 - val_accuracy: 0.9901 - 443ms/epoch - 3ms/step
Epoch 45/50
140/140 - 0s - loss: 0.0368 - accuracy: 0.9881 - val_loss: 0.0417 - val_accuracy: 0.9901 - 422ms/epoch - 3ms/step
Epoch 46/50
140/140 - 0s - loss: 0.0365 - accuracy: 0.9877 - val_loss: 0.0416 - val_accuracy: 0.9865 - 410ms/epoch - 3ms/step
Epoch 47/50
140/140 - 0s - loss: 0.0355 - accuracy: 0.9877 - val_loss: 0.0413 - val_accuracy: 0.9901 - 400ms/epoch - 3ms/step
Epoch 48/50
140/140 - 0s - loss: 0.0360 - accuracy: 0.9890 - val_loss: 0.0407 - val_accuracy: 0.9892 - 380ms/epoch - 3ms/step
Epoch 49/50
140/140 - 0s - loss: 0.0351 - accuracy: 0.9877 - val_loss: 0.0407 - val_accuracy: 0.9901 - 392ms/epoch - 3ms/step
Epoch 50/50
140/140 - 0s - loss: 0.0338 - accuracy: 0.9888 - val_loss: 0.0418 - val_accuracy: 0.9910 - 421ms/epoch - 3ms/step
```

results = model.evaluate(X_test, y_test)

loss = results[0] accuracy

= results[1]

OUTPUT:-

```
35/35 [==============================] - 0s 2ms/step - loss: 0.0418 - accuracy: 0.9910
```

print(f"[+] Accuracy: {accuracy*100:.2f}%")

OUTPUT:-

```
[+] Accuracy: 99.10%
```

from keras.preprocessing import sequence

#Defining the function def

get_predictions(txts):

    txts = tokenizer.texts_to_sequences(txts)

    txts = sequence.pad_sequences(txts, maxlen=max_length)

preds = model.predict(txts) if(preds[0] > 0.5):

        print("SPAM MESSAGE")

else:

        print('NOT SPAM')

txts=["You have won a free ticket to las vegas. Contact now"] get_predictions(txts)

OUTPUT:-

```
SPAM MESSAGE
```

txts=["Hey there call me asap!!"] get_predictions(txts)

OUTPUT:-

```
NOT SPAM
```

Practical No . 04

Roll No :03 **Date 18/04/2022**

Aim: Demonstrate Text Mining And Webpage Pre-Processing Using Meta Information From The Web Pages (Local/Online).

Code:-

```
# Commented out IPython magic to ensure Python compatibility.
# Imports import requests
import numpy as np import
pandas as pd from bs4 import
BeautifulSoup import
matplotlib.pyplot as plt

# %matplotlib inline # IMDB's
homepage imdb_url =
'https://www.imdb.com'

# Use requests to retrieve data from a given URL imdb_response
= requests.get(imdb_url)

# Parse the whole HTML page using BeautifulSoup imdb_soup
= BeautifulSoup(imdb_response.text, 'html.parser')

# Title of the parsed page
imdb_soup.title
```

Output:-

```
<title>IMDb: Ratings, Reviews, and Where to Watch the Best Movies &amp; TV Shows</title>
```

```
imdb_soup.title.string
```

Output:-

```
'IMDb: Ratings, Reviews, and Where to Watch the Best Movies & TV Shows'
```

```
trailers = imdb_soup.find('div', {'class': 'ab_hero'}) for widget in
imdb_soup.find_all('div', {'class': 'aux-content-widget-2'}):

   # Check that the widget has a heading
if widget.h3:

      # Print the widget's heading along with the movie titles.
      print(widget.h3.string)        for title in
widget.find_all('div', {'class': 'title'}):

         print(title.text)        print() for article in
imdb_soup.find_all('div', {'class': 'article'}):

   if article.h3:
```

```
    # Title of the article
    print(article.h3.string)

    # Text

    print(article.p.text)
print()

# Find all links

links = [link.get('href') for link in imdb_soup.find_all('a')]

# Add homepage and keep the unique links

fixed_links = set([''.join([imdb_url, link]) for link in links if link])

# Box Office Mojo - UK Weekend box office

boxofficemojo_url = 'https://www.boxofficemojo.com/intl/uk/?yr=2019&wk=33&currency=local'

# Use requests to retrieve data from a given URL

bom_response = requests.get(boxofficemojo_url) # Parse the

whole HTML page using BeautifulSoup bom_soup =

BeautifulSoup(bom_response.text, 'html.parser')

print(f"NUMBER OF TABLES IN THE PAGE: {len(bom_soup.find_all('table'))}")
```

Output:-

```
NUMBER OF TABLES IN THE PAGE: 1
```

table = bom_soup.find_all('table')[0] table

Output:-

```
<table class="a-bordered a-horizontal-stripes a-size-base a-span12 mojo-body-table mojo-table-annotated"><tr><th class="a-text-left mojo-field-type-date_interval
</th><th class="a-text-left mojo-field-type-genre hidden mojo-sortable-column hidden a-nowrap"><span title="Genre">Genre</span>
</th><th class="a-text-right mojo-field-type-money hidden mojo-sortable-column hidden a-nowrap"><span title="Budget">Budget</span>
</th><th class="a-text-right mojo-field-type-duration hidden mojo-sortable-column hidden a-nowrap"><span title="Running Time">Running Time</span>
</th><th class="a-text-right mojo-field-type-date_interval mojo-sortable-column a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=weekNum&amp;re
</th></tr><tr><td class="a-text-left mojo-header-column mojo-truncate mojo-field-type-date_interval mojo-sort-column"><a class="a-link-normal" href="/weekend/2019
```

table.find_all('tr')[0].contents

Output:-

```
[<th class="a-text-left mojo-field-type-date_interval mojo-sort-column mojo-sortable-column a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=st
 <th class="a-text-right mojo-field-type-money mojo-sortable-column a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=top10Gross&amp;ref_=bo_wey
 <th class="a-text-right mojo-field-type-percent_delta mojo-sortable-column a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=top10GrossDelta&an
 <th class="a-text-right mojo-field-type-money mojo-sortable-column mojo-estimatable a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=gross&amp
 <th class="a-text-right mojo-field-type-percent_delta mojo-sortable-column mojo-estimatable a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=g
 <th class="a-text-right mojo-field-type-positive_integer mojo-sortable-column a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=numReleases&amp
 <th class="a-text-left mojo-field-type-release mojo-cell-wide mojo-sortable-column a-nowrap"><span title="#1 Release">#1 Release</span>
 </th>,
 <th class="a-text-left mojo-field-type-genre hidden mojo-sortable-column hidden a-nowrap"><span title="Genre">Genre</span>
 </th>,
 <th class="a-text-right mojo-field-type-money hidden mojo-sortable-column hidden a-nowrap"><span title="Budget">Budget</span>
 </th>,
 <th class="a-text-right mojo-field-type-duration hidden mojo-sortable-column hidden a-nowrap"><span title="Running Time">Running Time</span>
 </th>,
 <th class="a-text-right mojo-field-type-date_interval mojo-sortable-column a-nowrap"><a class="a-link-normal a-nowrap" href="?area=GB&amp;sort=weekNum&amp;ref_=b
 <th class="a-text-right mojo-field-type-boolean hidden mojo-sortable-column hidden a-nowrap"><span title="Long Weekend">Long Weekend</span>
 </th>]
```

table.find_all('tr')[0].text **Output:-**

```
'DatesTop 10 Gross%± LWOverall Gross%± LWReleases#1 Release\nGenre\nBudget\nRunning Time\nWeekLong Weekend\n'
```

Print text "consumes" the newline characters print(table.find_all('tr')[0].text)

Output:-

```
DatesTop 10 Gross%± LWOverall Gross%± LWReleases#1 Release
Genre
Budget
Running Time
WeekLong Weekend
```

table.find_all('tr')[0].text.split('\n')

lst = [] for row in table.find_all('tr')[1:-1]:

s = pd.Series([data.text for data in row.find_all('td')])

lst.append(s)

data = pd.concat(lst, axis=1).T data.head(2)

Output:-

	0	1	2	3	4	5	6	7	8	9	10	11
0	Dec 27-29	$29,979,260	-24.7%	$30,516,920	-24.1%	51	Star Wars: Episode IX - The Rise of Skywalker	-	-	-	52	false
1	Dec 20-22	$39,831,375	+150.4%	$40,218,275	+138.4%	53	Star Wars: Episode IX - The Rise of Skywalker	-	-	-	51	false

print(f'(MOVIES, COLUMNS) -> {data.shape}')

Output:-

```
(MOVIES, COLUMNS) -> (51, 12)
```

print(f'% OF MISSING VALUES PER COLUMN\n{(data.isnull().sum() / data.shape[0]) * 100}')

Output:-

```
% OF MISSING VALUES PER COLUMN
0     0.0
1     0.0
2     0.0
3     0.0
4     0.0
5     0.0
6     0.0
7     0.0
8     0.0
9     0.0
10    0.0
11    0.0
dtype: float64
```

Practical No. 05

Roll No: 03 **Date:** 23/04/2022

Aim: Apriori algorithm implementation in case study.

Itemset ={ Bread, Chicken, Butter, Milk, Toast}

Transaction ID	**Items**
100	{Bread, Butter, Milk}
200	{Chicken, Butter, Toast}
300	{Bread, Chicken, Butter, Toast}
400	{Chicken, Toast}

Item	**Support**
Bread	**2 /4 = 0.5 = 50%**
Chicken	**3 /4 = 0.5 = 75%**
Butter	**3 /4 = 0.75 = 75%**
Milk	**1/4 = 0.25 = 25%**
Toast	**3/4 = 0.75 = 75%**

Itemset = { Bread, Chicken , Butter, Toast}

Item	**Support**
{Bread, Chicken}	**¼ = 0.25 =25%**
{Bread, Butter}	**2/4 =0.50 = 50%**
{Bread, Toast}	**¼ = 0.25 = 25%**
{Chicken, Butter }	**2/4 = 0.50 = 50 %**
{Chicken, Toast}	**¾ = 0.75 = 75%**
{Butter, Toast}	**2/4 = 0.50 = 50%**

Itemset = ({Bread, Butter}, {Chicken, Butter} , {Chicken, Toast}, {Butter, Toast})

Item	**Support**
{Bread, Butter, Toast}	1/ 4 = 0.25 = 25%
{Chicken, Butter, Toast}	2/4 =0.50 = 50 %
{Bread, Butter, Chicken}	¼ = 0.25 = 25%

Final Resultant Set based on Support = {Chicken, Butter, Toast}

Rules

1 . (Chicken & Butter) - > Toast 2 (50%)

2. (Butter & Toast) -> Chicken 2 (50%)

3. (Chicken & Toast) -> Butter 2 (50%)

4. Chicken - > (Butter & Toast) 2 (50%)

5. Toast -> (Chicken & Butter) 2 (50%)

6. Butter -> (Chicken & Toast) 2 (50%)

Confidence = S(A U B).count / S(A).count

1 . (Chicken & Butter) - > Toast 2 (50%)

S((Chicken &Butter) U (Toast))/ S(Chicken & Butter)

=2 / 2 = 1
= **100%**

2. . (Butter & Toast) -> Chicken

Confidence = S(A U B).count / S(A).count
S((Butter & Toast) U Chicken)) /S(Butter & Toast)

=2 / 2 = 1 = **100%**

3. (Chicken & Toast) -> Butter 2 (50%)

Confidence = S(A U B).count / S(A).count

S((Chicken & Toast) U (Butter))/S(Chicken & Toast)

=2/3 = 0.666 **= 67%**

4. Chicken - > (Butter & Toast) 2 (50%)

Confidence = S(A U B).count / S(A).count

S((Chicken) U (Butter & Toast))/S(Chicken)

=2/3 = 0.666 = **67%**

5. Toast -> (Chicken & Butter) 2 (50%) Confidence = S(A U B).count / S(A).count S((Toast) U (Chicken & Butter))/S(Toast)

=2/3 = 0.666 = **67%**

6. Butter -> (Chicken & Toast) 2 (50%) Confidence

= S(A U B).count / S(A).count

S((Butter) U (Chicken & Toast))/S(Butter)

=2/3 = 0.666 = **67%**

Final Associated Items rules are

1 . (Chicken & Butter) - > Toast 2 (50%)

2. (Butter & Toast) -> Chicken 2 (50%)

Practical No. 06

Roll No: 03 **Date:** 18/04 /2022

Aim: Develop A Basic Crawler For The Web Search For User Defined Keywords.

Code:- import

requests

url = 'https://en.wikipedia.org/wiki/Stock_market'

Connect to the url using requests.get response

= requests.get(url) response.status_code

Output:-

```
200
```

Add a timeout to prevent hanging response

= requests.get(url, timeout=3)

response.status_code

Output:-

```
200
```

import requests

url = 'https://en.wikipedia.org/wiki/Stock_market' import

csv

from bs4 import BeautifulSoup response =

requests.get(url, timeout=3) print('Status

code: ',response.status_code) if

response.status_code==200:

print('Connection successfull.\n\n') else:

print('Error. Check status code table.\n\n')

Output:-

```
Status code:  200
Connection successfull.
```

Print out the contents of a request's response print(f"{'---

'*20}\n\tContents of Response.items():\n{'---'*20}") **Output:-**

```
--------------------------------------------------------------
        Contents of Response.items():
--------------------------------------------------------------
```

for k,v in response.headers.items():

print(f"{k:{25}}: {v:{40}}") # Note: add :{number} inside of a

Output:-

```
date: Sun, 17 Apr 2022 14:51:17 GMT
server: mw1325.eqiad.wmnet
x-content-type-options: nosniff
p3p: CP="See https://en.wikipedia.org/wiki/Special:CentralAutoLogin/P3P for more info."
content-language: en
vary: Accept-Encoding,Cookie,Authorization
last-modified: Sun, 17 Apr 2022 14:45:56 GMT
content-type: text/html; charset=UTF-8
content-encoding: gzip
age: 54765
x-cache: cp3052 hit, cp3058 hit/16
x-cache-status: hit-front
server-timing: cache;desc="hit-front", host;desc="cp3058"
strict-transport-security: max-age=106384710; includeSubDomains; preload
report-to: { "group": "wm_nel", "max_age": 86400, "endpoints": [{ "url": "https://intake-logging.wikimedia.org/v1/events?stream=w3c.reportingapi.network_error&sch
nel: { "report_to": "wm_nel", "max_age": 86400, "failure_fraction": 0.05, "success_fraction": 0.0}
set-cookie: WMF-Last-Access=18-Apr-2022;Path=/;HttpOnly;secure;Expires=Fri, 20 May 2022 00:00:00 GMT, WMF-Last-Access-Global=18-Apr-2022;Path=/;Domain=.wikipedia.
accept-ch: Sec-CH-UA-Arch,Sec-CH-UA-Bitness,Sec-CH-UA-Full-Version-List,Sec-CH-UA-Model,Sec-CH-UA-Platform-Version
permissions-policy: interest-cohort=(),ch-ua-arch=(self "intake-analytics.wikimedia.org"),ch-ua-bitness=(self "intake-analytics.wikimedia.org"),ch-ua-full-versior
x-client-ip: 34.150.137.179
cache-control: private, s-maxage=0, max-age=0, must-revalidate
accept-ranges: bytes
content-length: 65700
```

for k,v in response.headers.items():

print(f"{k}: {v}") # Note: add :{number} inside of a **Output:-**


```
date: Sun, 17 Apr 2022 14:51:17 GMT
server: mw1325.eqiad.wmnet
x-content-type-options: nosniff
p3p: CP="See https://en.wikipedia.org/wiki/Special:CentralAutoLogin/P3P for more info."
content-language: en
vary: Accept-Encoding,Cookie,Authorization
last-modified: Sun, 17 Apr 2022 14:45:56 GMT
content-type: text/html; charset=UTF-8
content-encoding: gzip
age: 54765
x-cache: cp3052 hit, cp3058 hit/16
x-cache-status: hit-front
server-timing: cache;desc="hit-front", host;desc="cp3058"
strict-transport-security: max-age=106384710; includeSubDomains; preload
report-to: { "group": "wm_nel", "max_age": 86400, "endpoints": [{ "url": "https://intake-logging.wikimedia.org/v1/events?stream=w3c.reportingapi.network_error&sch
nel: { "report_to": "wm_nel", "max_age": 86400, "failure_fraction": 0.05, "success_fraction": 0.0}
set-cookie: WMF-Last-Access=18-Apr-2022;Path=/;HttpOnly;secure;Expires=Fri, 20 May 2022 00:00:00 GMT, WMF-Last-Access-Global=18-Apr-2022;Path=/;Domain=.wikipedia.
accept-ch: Sec-CH-UA-Arch,Sec-CH-UA-Bitness,Sec-CH-UA-Full-Version-List,Sec-CH-UA-Model,Sec-CH-UA-Platform-Version
permissions-policy: interest-cohort=(),ch-ua-arch=(self "intake-analytics.wikimedia.org"),ch-ua-bitness=(self "intake-analytics.wikimedia.org"),ch-ua-full-versior
x-client-ip: 34.150.137.179
cache-control: private, s-maxage=0, max-age=0, must-revalidate
accept-ranges: bytes
content-length: 65700
```


print(f"Status code: {response.status_code}") print(f"Status code: {response.status_code:>{20}}") print(f"Status code: {response.status_code:->{20}}") **Output:-**

```
Status code: 200
Status code:                  200
Status code: -----------------200
```

Define Url and establish connection url = 'https://en.wikipedia.org/wiki/Stock_market'

response = requests.get(url, timeout=3)

Feed the response's .content into BeauitfulSoup page_content = response.content

soup = BeautifulSoup(page_content,'lxml') #'html.parser')

Preview soup contents using .prettify() print(soup.prettify()[:2000])

Output:-

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Stock market - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDate
"All articles containing potentially dated statements","Articles with limited geographic scope from November 2020","United States-centric","All accuracy disputes"
```

body = soup.body for child in body.children: # print child if its not empty

print(child if child is not None else ' ', '\n\n') # '\n\n' for visual separation

Output:-

```
<div class="noprint" id="mw-page-base"></div>

<div class="noprint" id="mw-head-base"></div>

<div class="mw-body" id="content" role="main">
<a id="top"></a>
<div id="siteNotice">
<div class="mw-indicators">
</div>
<h1 class="firstHeading mw-first-heading" id="firstHeading">Stock market</h1>
<div class="vector-body" id="bodyContent">
<div class="noprint" id="siteSub">From Wikipedia, the free encyclopedia</div>
<div id="contentSub"></div>
<div id="contentSub2"></div>
<div id="jump-to-nav"></div>
<a class="mw-jump-link" href="#mw-head">Jump to navigation</a>
```

```
</p>
<style data-mw-deduplicate="TemplateStyles:r1045330069">.mw-parser-output .sidebar{width:22em;float:right;clear:right;margin:0.5em 0 1em 1em;background:#f8f9fa;
<div class="sidebar-list mw-collapsible mw-collapsed"><div class="sidebar-list-title" style="background:#efefef;"><a href="/wiki/Financial_market" title="Finan
<a href="/wiki/Financial_asset" title="Financial asset">Assets</a></th></tr><tr><td class="sidebar-content hlist" style="padding-left:0.5em;padding-right:0.5em;
<ul><li><a href="/wiki/Bond_market" title="Bond market">Bond</a></li>
<li><a href="/wiki/Commodity_market" title="Commodity market">Commodity</a></li>
<li><a href="/wiki/Derivatives_market" title="Derivatives market">Derivatives</a></li>
<li><a href="/wiki/Foreign_exchange_market" title="Foreign exchange market">Foreign exchange</a></li>
<li><a href="/wiki/Money_market" title="Money market">Money</a></li>
<li><span class="nowrap"><a href="/wiki/Over-the-counter_(finance)" title="Over-the-counter (finance)">Over-the-counter</a></span></li>
<li><a href="/wiki/Private_equity" title="Private equity">Private equity</a></li>
<li><a href="/wiki/Real_estate" title="Real estate">Real estate</a></li>
<li><a href="/wiki/Spot_market" title="Spot market">Spot</a></li>
<li><a class="mw-selflink selflink">Stock</a></li></ul></td>
</tr><tr><th class="sidebar-heading" style="background:#e9e9ff;font-weight:normal;font-style:italic;">
<a href="/wiki/Financial_market_participants" title="Financial market participants">Participants</a></th></tr><tr><td class="sidebar-content hlist" style="padd
<ul><li><a href="/wiki/Investor" title="Investor">Investor</a>
<ul><li><a href="/wiki/Institutional_investor" title="Institutional investor">institutional</a></li></ul></li>
<li><a href="/wiki/Retail" title="Retail">Retail</a></li>
<li><a href="/wiki/Speculation" title="Speculation">Speculator</a></li></ul></td>
</tr><tr><th class="sidebar-heading" style="background:#e9e9ff;font-weight:normal;font-style:italic;">
<a href="/wiki/Financial_centre" title="Financial centre">Locations</a></th></tr><tr><td class="sidebar-content hlist" style="padding-top:0.15em;">
<ul><li><a href="/wiki/Financial_centre" title="Financial centre">Financial centres</a></li>
<li><a href="/wiki/Offshore_financial_centre" title="Offshore financial centre">Offshore financial centres</a></li>
<li><a href="/wiki/Conduit_and_sink_OFCs" title="Conduit and sink OFCs">Conduit and sink OFCs</a></li></ul></td>
```

✓ 2s completed at 4:57 PM

```
<script>(RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgPageParseReport":{"limitreport":{"cputime":"1.702","walltime":"2.049","ppvisitednodes":{"value":

<script type="application/ld+json">{"@context":"https:\/\/schema.org","@type":"Article","name":"Stock market","url":"https:\/\/en.wikipedia.org\/wiki\/Stock_mar

<script>(RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponseTime":182,"wgHostname":"mw1325"});});</script>
```

title = soup.head.title

print(title.parent.name)

Output:-

```
head
```

results = soup.find_all() results

Output:-

```
[<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>Stock market - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDe
"All articles containing potentially dated statements","Articles with limited geographic scope from November 2020","United States-centric","All accuracy disput
"wgWMEPageLength":60000,"wgNoticeProject":"wikipedia","wgMediaViewerOnClick":true,"wgMediaViewerEnabledByDefault":true,"wgPopupsFlags":10,"wgULSCurrentAutonym"
"mediawiki.toc","skins.vector.legacy.js","ext.gadget.ReferenceTooltips","ext.gadget.charinsert","ext.gadget.extra-toolbar-buttons","ext.gadget.refToolbar","ext
<script>(RLQ=window.RLQ||[]).push(function(){mw.loader.implement("user.options@1i9g4",function($,jQuery,require,module){mw.user.tokens.set({"patrolToken":"+\\
<link href="/w/load.php?lang=en&amp;modules=ext.cite.styles%7Cext.uls.interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.wikimediaBadges%7Cj
<script async="" src="/w/load.php?lang=en&amp;modules=startup&amp;only=scripts&amp;raw=1&amp;skin=vector"></script>
<meta content="" name="ResourceLoaderDynamicStyles"/>
<link href="/w/load.php?lang=en&amp;modules=site.styles&amp;only=styles&amp;skin=vector" rel="stylesheet"/>
<meta content="MediaWiki 1.39.0-wmf.7" name="generator"/>
<meta content="origin" name="referrer"/>
<meta content="origin-when-crossorigin" name="referrer"/>
<meta content="origin-when-cross-origin" name="referrer"/>
```

✓ 2s completed at 4:57 PM

```
<meta content="telephone=no" name="format-detection"/>
<meta content="https://upload.wikimedia.org/wikipedia/commons/d/d7/Philippine-stock-market-board.jpg" property="og:image"/>
<meta content="1200" property="og:image:width"/>
<meta content="900" property="og:image:height"/>
<meta content="https://upload.wikimedia.org/wikipedia/commons/d/d7/Philippine-stock-market-board.jpg" property="og:image"/>
<meta content="800" property="og:image:width"/>
<meta content="600" property="og:image:height"/>
<meta content="640" property="og:image:width"/>
<meta content="480" property="og:image:height"/>
<meta content="Stock market - Wikipedia" property="og:title"/>
<meta content="website" property="og:type"/>
<link href="//upload.wikimedia.org" rel="preconnect"/>
<link href="//en.m.wikipedia.org/wiki/Stock_market" media="only screen and (max-width: 720px)" rel="alternate"/>
<link href="/w/index.php?title=Stock_market&amp;action=edit" rel="alternate" title="Edit this page" type="application/x-wiki"/>
<link href="/static/apple-touch/wikipedia.png" rel="apple-touch-icon"/>
<link href="/static/favicon/wikipedia.ico" rel="shortcut icon"/>
<link href="/w/opensearch_desc.php" rel="search" title="Wikipedia (en)" type="application/opensearchdescription+xml"/>
<link href="//en.wikipedia.org/w/api.php?action=rsd" rel="EditURI" type="application/rsd+xml"/>
```

✓ 2s completed at 4:57 PM

```
<li><a href="/wiki/Cash" title="Cash">Cash</a></li>
<li><a class="mw-redirect" href="/wiki/Collateralised_debt_obligation" title="Collateralised debt obligation">Collateralised debt obligation</a></li>
<li><a href="/wiki/Credit_default_swap" title="Credit default swap">Credit default swap</a></li>
<li><a href="/wiki/Time_deposit" title="Time deposit">Time deposit</a> (<a href="/wiki/Certificate_of_deposit" title="Certificate of deposit">certificate of de
<li><a href="/wiki/Line_of_credit" title="Line of credit">Credit line</a></li>
<li><a href="/wiki/Deposit_account" title="Deposit account">Deposit</a></li>
<li><a href="/wiki/Derivative_(finance)" title="Derivative (finance)">Derivative</a></li>
<li><a href="/wiki/Futures_contract" title="Futures contract">Futures contract</a></li>
<li><a href="/wiki/Indemnity" title="Indemnity">Indemnity</a></li>
<li><a href="/wiki/Insurance" title="Insurance">Insurance</a></li>
<li><a href="/wiki/Letter_of_credit" title="Letter of credit">Letter of credit</a></li>
<li><a href="/wiki/Loan" title="Loan">Loan</a></li>
<li><a class="mw-redirect" href="/wiki/Mortgage" title="Mortgage">Mortgage</a></li></ul>
<div class="hlist hlist-separated"><ul><li><a href="/wiki/Option_(finance)" title="Option (finance)">Option</a> (<a href="/wiki/Call_option" title="Call optio
<ul><li><a class="mw-redirect" href="/wiki/Performance_bonds" title="Performance bonds">Performance bonds</a></li>
<li><a href="/wiki/Repurchase_agreement" title="Repurchase agreement">Repurchase agreement</a></li>
<li><a href="/wiki/Stock" title="Stock">Stock</a></li>
<li><a href="/wiki/Security_(finance)" title="Security (finance)">Security</a></li>
<li><a href="/wiki/Syndicated_loan" title="Syndicated loan">Syndicated loan</a></li>
<li><a href="/wiki/Synthetic_CDO" title="Synthetic CDO">Synthetic CDO</a></li></ul>
</div></div></div></td>
</tr><tr><td class="sidebar-content">
<div class="sidebar-list mw-collapsible mw-collapsed"><div class="sidebar-list-title" style="background:#efefef;"><a href="/wiki/Corporate_finance" title="Cor
General</th></tr><tr><td class="sidebar-content hlist" style="padding-left:0.5em;padding-right:0.5em;">
<ul><li><a href="/wiki/Accounting" title="Accounting">Accounting</a></li>
<li><a href="/wiki/Audit" title="Audit">Audit</a></li>
<li><a href="/wiki/Capital_budgeting" title="Capital budgeting">Capital budgeting</a></li>
```

✓ 2s completed at 4:57 PM

```
<li><a class="mw-redirect" href="/wiki/Leon_Walras" title="Leon Walras">Leon Walras</a></li>
<li><a href="/wiki/Ludwig_von_Mises" title="Ludwig von Mises">Ludwig von Mises</a></li>
<li><a href="/wiki/Ayn_Rand" title="Ayn Rand">Ayn Rand</a></li>
<li><a href="/wiki/Murray_Rothbard" title="Murray Rothbard">Murray Rothbard</a></li>
<li><a href="/wiki/Joseph_Schumpeter" title="Joseph Schumpeter">Joseph Schumpeter</a></li>
<li><a href="/wiki/Thorstein_Veblen" title="Thorstein Veblen">Thorstein Veblen</a></li>
<li><a href="/wiki/Max_Weber" title="Max Weber">Max Weber</a></li>
<li><a href="/wiki/Ronald_Coase" title="Ronald Coase">Ronald Coase</a></li></ul></div>, <ul><li><a href="/wiki/Adam_Smith" title="Adam Smith">Adam Smith</a></
<li><a href="/wiki/John_Stuart_Mill" title="John Stuart Mill">John Stuart Mill</a></li>
<li><a href="/wiki/David_Ricardo" title="David Ricardo">David Ricardo</a></li>
<li><a href="/wiki/Thomas_Robert_Malthus" title="Thomas Robert Malthus">Thomas Robert Malthus</a></li>
<li><a href="/wiki/Jean-Baptiste_Say" title="Jean-Baptiste Say">Jean-Baptiste Say</a></li>
<li><a href="/wiki/Karl_Marx" title="Karl Marx">Karl Marx</a></li>
<li><a href="/wiki/Milton_Friedman" title="Milton Friedman">Milton Friedman</a></li>
<li><a href="/wiki/Friedrich_Hayek" title="Friedrich Hayek">Friedrich Hayek</a></li>
<li><a href="/wiki/John_Maynard_Keynes" title="John Maynard Keynes">John Maynard Keynes</a></li>
<li><a href="/wiki/Alfred_Marshall" title="Alfred Marshall">Alfred Marshall</a></li>
<li><a href="/wiki/Vilfredo_Pareto" title="Vilfredo Pareto">Vilfredo Pareto</a></li>
<li><a class="mw-redirect" href="/wiki/Leon_Walras" title="Leon Walras">Leon Walras</a></li>
<li><a href="/wiki/Ludwig_von_Mises" title="Ludwig von Mises">Ludwig von Mises</a></li>
<li><a href="/wiki/Ayn_Rand" title="Ayn Rand">Ayn Rand</a></li>
<li><a href="/wiki/Murray_Rothbard" title="Murray Rothbard">Murray Rothbard</a></li>
<li><a href="/wiki/Joseph_Schumpeter" title="Joseph Schumpeter">Joseph Schumpeter</a></li>
<li><a href="/wiki/Thorstein_Veblen" title="Thorstein Veblen">Thorstein Veblen</a></li>
<li><a href="/wiki/Max_Weber" title="Max Weber">Max Weber</a></li>
<li><a href="/wiki/Ronald_Coase" title="Ronald Coase">Ronald Coase</a></li></ul>, <li><a href="/wiki/Adam_Smith" title="Adam Smith">Adam Smith</a></li>, <a hre
```

✓ 2s completed at 4:57 PM

Practical No. 07

Roll No: 03 **Date: 20/04 /2022**

Aim: Develop a focused crawler for local search.

Code:- import

requests import

lxml

from bs4 import BeautifulSoup

url = "https://www.rottentomatoes.com/top/bestofrt/" f

= requests.get(url)

url = "https://www.rottentomatoes.com/top/bestofrt/" headers

= {

'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QIHU 360SE'

}

f = requests.get(url, headers = headers)

soup = BeautifulSoup(f.content,'lxml')

movies = soup.find('table',{'class':'table'}).find_all('a')

movies_lst = [] num =

0 for anchor in

movies:

urls = 'https://www.rottentomatoes.com' + anchor['href']

movies_lst.append(urls)

num += 1 movie_url =

urls

movie_f = requests.get(movie_url, headers = headers) movie_soup

= BeautifulSoup(movie_f.content, 'lxml') movie_content =

movie_soup.find('div', {

'class': 'movie_synopsis clamp clamp-6 js-clamp'

})

print(num, urls, '\n', 'Movie:' + anchor.string.strip()) print('Movie info:' + movie_content.string.strip())

Output:-

```
1 https://www.rottentomatoes.com/m/the_battle_of_algiers
 Movie:The Battle of Algiers (La Battaglia di Algeri) (1967)
Movie info:Paratrooper commander Colonel Mathieu (Jean Martin), a former French Resistance fighter during World War II, is sent to 1950s Algeria to reinforce effo
```

Practical No.08

Roll No: 03 **Date:** 20/04 /2022

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

Code:

!pip install matplotlib pandas nltk textblob

Output:-

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (3.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.3.5)
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.2.5)
Requirement already satisfied: textblob in /usr/local/lib/python3.7/dist-packages (0.15.3)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.21.6)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (3.0.8)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.4.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib) (4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2022.1)
```

import nltk

nltk.download('vader_lexicon')

nltk.download('movie_reviews') nltk.download('punkt')

Output:-

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA **Output:-**

```
/usr/local/lib/python3.7/dist-packages/nltk/twitter/__init__.py:20: UserWarning: The twython library has not been installed. Some functionality from the twitter p
  warnings.warn("The twython library has not been installed. "
```

sia

= SIA()

sia.polarity_scores("This restaurant was great, but I'm not sure if I'll go there again.") **Output:-**

```
{'compound': 0.0276, 'neg': 0.153, 'neu': 0.688, 'pos': 0.159}
```

text

= "I just got a call from my boss - does he realise it's Saturday?"

sia.polarity_scores(text)

Output:-

```
{'compound': 0.0, 'neg': 0.0, 'neu': 1.0, 'pos': 0.0}
```

text = "I just got a call from my boss - does he realise it's Saturday? :)"

sia.polarity_scores(text)

Output:-

```
{'compound': 0.4588, 'neg': 0.0, 'neu': 0.786, 'pos': 0.214}
```

text = "I just got a call from my boss - does he realise it's Saturday? " sia.polarity_scores(text)

Output:-

```
{'compound': 0.0, 'neg': 0.0, 'neu': 1.0, 'pos': 0.0}
```

from textblob import TextBlob from

textblob import Blobber

from textblob.sentiments import NaiveBayesAnalyzer

blob = TextBlob("This restaurant was great, but I'm not sure if I'll go there again.") blob.sentiment

Output:-

```
Sentiment(polarity=0.275, subjectivity=0.8194444444444444)
```

blobber = Blobber(analyzer=NaiveBayesAnalyzer())

blob = blobber("This restaurant was great, but I'm not sure if I'll go there again.") blob.sentiment

Output:-

```
Sentiment(classification='pos', p_pos=0.5879425317005774, p_neg=0.41205746829942275)
```

import pandas as pd

pd.set_option("display.max_colwidth", 200)

```
df = pd.DataFrame({'content': [
    "I love love love love this kitten",
    "I hate hate hate hate this keyboard",
    "I'm not sure how I feel about toast",
    "Did you see the baseball game yesterday?",
    "The package was delivered late and the contents were broken",
    "Trashy television shows are some of my favorites",
    "I'm seeing a Kubrick film tomorrow, I hear not so great things about it.",
    "I find chirping birds irritating, but I know I'm not the only one",
]})
```

Df

Output:-

	content
0	I love love love love this kitten
1	I hate hate hate hate this keyboard
2	I'm not sure how I feel about toast
3	Did you see the baseball game yesterday?
4	The package was delivered late and the contents were broken
5	Trashy television shows are some of my favorites
6	I'm seeing a Kubrick film tomorrow, I hear not so great things about it.
7	I find chirping birds irritating, but I know I'm not the only one

```
def get_scores(content):     blob = TextBlob(content)     nb_blob = blobber(content)

    sia_scores = sia.polarity_scores(content)


    return pd.Series({
        'content': content,
        'textblob': blob.sentiment.polarity,
        'textblob_bayes': nb_blob.sentiment.p_pos - nb_blob.sentiment.p_neg,
'nltk': sia_scores['compound'],

    })


scores = df.content.apply(get_scores)
scores.style.background_gradient(cmap='RdYlGn', axis=None, low=0.4, high=0.4)
```

Output:-

	content	textblob	textblob_bayes	nltk
0	I love love love love this kitten	0.500000	-0.087933	0.957100
1	I hate hate hate hate this keyboard	-0.800000	-0.214151	-0.941300
2	I'm not sure how I feel about toast	-0.250000	0.394659	-0.241100
3	Did you see the baseball game yesterday?	-0.400000	0.613050	0.000000
4	The package was delivered late and the contents were broken	-0.350000	-0.574270	-0.476700
5	Trashy television shows are some of my favorites	0.000000	0.040076	0.421500
6	I'm seeing a Kubrick film tomorrow, I hear not so great things about it.	0.800000	0.717875	-0.629600
7	I find chirping birds irritating, but I know I'm not the only one	-0.200000	0.257148	-0.250000