Practical 1

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

Theory:

What is Web Scraping?

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

Web scraping requires two parts, namely the crawler and the scraper. The crawler is an artificial intelligence algorithm that browses the web to search for the particular data required by following the links across the internet. The scraper, on the other hand, is a specific tool created to extract data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data

Code:

In [23]:
```
import csv
from kora.selenium import wd
from bs4 import BeautifulSoup

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(executable_path='chromedriver.exe')
wd = webdriver.Chrome(r'C:\Users\LENOVO\chromedriver',options=options)
wd.get("https://www.amazon.in/")
print(wd.page_source) # result
```

In [24]:
```
def get_url(search_term):
 template = "https://www.amazon.in/s?k={}&rh=n%3A1389401031&ref=nb_sb_noss"
 search_term = search_term.replace(' ','+')
 return template.format(search_term)

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


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

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

print(len(result))

item = result[1]

atag = item.h2.a
print(atag.text)

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

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

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

Output:

```
<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-in
put-placeholder a-textarea-placeholder a-local-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" data-19ax5a9jf="dingo" data-aui-build-date
="3.23.1-2023-04-28" data-useragent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessC
hrome/112.0.5615.138 Safari/537.36" data-platform="Win32"><!-- sp:feature:head-start --><head><script async="" src="https://c.a
mazon-adsystem.com/bao-csm/forensics/a9-tq-forensics.min.js" crossorigin="anonymous"></script><script async="" src="https://ima
ges-eu.ssl-images-amazon.com/images/I/31bJewCvY-L.js" crossorigin="anonymous"></script><style class="vjs-styles-defaults">
      .video-js {
        width: 300px;
        height: 150px;
      }

      .vjs-fluid {
        padding-top: 56.25%
      }
    </style><style class="vjs-styles-dimensions">
      .vjs_video_3-dimensions {
        width: 300px;
        height: 168.75px;
      }

      .vjs_video_3-dimensions.vjs-fluid {
        padding-top: 56.25%;
      }
    </style><script>var aPageStart = (new Date()).getTime();</script><meta charset="utf-8">
<!-- sp:end-feature:head-start -->
<!-- sp:feature:csm:head-open-part1 -->

<script type="text/javascript">var ue_t0=ue_t0||+new Date();</script>
<!-- sp:end-feature:csm:head-open-part1 -->
<!-- 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 -->
```

```
https://www.amazon.in/s?k=mobiles&rh=n%3A1389401031&ref=nb_sb_noss
24
realme narzo N55 (Prime Black, 4GB+64GB) 33W Segment Fastest Charging | Super High-res 64MP Primary AI Camera
₹10,999
4.4 out of 5 stars
('Samsung Galaxy M04 Dark Blue, 4GB RAM, 64GB Storage | Upto 8GB RAM with RAM Plus | MediaTek Helio P35 Octa-core Processor | 5
000 mAh Battery | 13MP Dual Camera', <span class="a-price" data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹6,99
9</span><span aria-hidden="true"><span class="a-price-symbol">₹</span><span class="a-price-whole">6,999</span></span></span>,
'')
('realme narzo N55 (Prime Black, 4GB+64GB) 33W Segment Fastest Charging | Super High-res 64MP Primary AI Camera', <span class
="a-price" data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹10,999</span><span aria-hidden="true"><span class="a
-price-symbol">₹</span><span class="a-price-whole">10,999</span></span></span>, '')
('Samsung Galaxy M04 Light Green, 4GB RAM, 64GB Storage | Upto 8GB RAM with RAM Plus | MediaTek Helio P35 Octa-core Processor |
5000 mAh Battery | 13MP Dual Camera', <span class="a-price" data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹6,9
99</span><span aria-hidden="true"><span class="a-price-symbol">₹</span><span class="a-price-whole">6,999</span></span></span>,
'')
('OnePlus Nord CE 3 Lite 5G (Chromatic Gray, 8GB RAM, 128GB Storage)', <span class="a-price" data-a-color="base" data-a-size="x
l"><span class="a-offscreen">₹19,999</span><span aria-hidden="true"><span class="a-price-symbol">₹</span><span class="a-price-w
hole">19,999</span></span></span>, '')
('realme narzo 50i Prime (Dark Blue 4GB RAM+64GB Storage) Octa-core Processor | 5000 mAh Battery', <span class="a-price" data-a
-color="base" data-a-size="xl"><span class="a-offscreen">₹7,999</span><span aria-hidden="true"><span class="a-price-symbol">₹</
span><span class="a-price-whole">7,999</span></span></span>, '')
('OnePlus Nord CE 2 Lite 5G (Black Dusk, 6GB RAM, 128GB Storage)', <span class="a-price" data-a-color="base" data-a-size="xl"><
span class="a-offscreen">₹18,490</span><span aria-hidden="true"><span class="a-price-symbol">₹</span><span class="a-price-whol
e">18,490</span></span></span>, '')
('Nokia G21 Android Smartphone, Dual SIM, 3-Day Battery Life, 6GB RAM + 128GB Storage, 50MP Triple AI Camera | Nordic Blue', <s
pan class="a-price" data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹12,999</span><span aria-hidden="true"><span
class="a-price-symbol">₹</span><span class="a-price-whole">12,999</span></span></span>, '')
('Samsung Galaxy M13 (Midnight Blue, 4GB, 64GB Storage) | 6000mAh Battery | Upto 8GB RAM with RAM Plus', <span class="a-price"
data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹9,699</span><span aria-hidden="true"><span class="a-price-symbo
l">₹</span><span class="a-price-whole">9,699</span></span></span>, '')
('Redmi 10A (Slate Grey, 4GB RAM, 64GB Storage) | 2 Ghz Octa Core Helio G25 | 5000 mAh Battery | Finger Print Sensor | Upto 5GB
RAM with RAM Booster', <span class="a-price" data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹8,599</span><span
aria-hidden="true"><span class="a-price-symbol">₹</span><span class="a-price-whole">8,599</span></span></span>, '')
('Redmi A1 (Light Green, 2GB RAM 32GB ROM) | Segment Best AI Dual Cam | 5000mAh Battery | Leather Texture Design | Android 12',
<span class="a-price" data-a-color="base" data-a-size="xl"><span class="a-offscreen">₹5,699</span><span aria-hidden="true"><spa
n class="a-price-symbol">₹</span><span class="a-price-whole">5,699</span></span></span>, '')
```

Practical 2

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:

What is Page Rank?
PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. 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. The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called “iterations”, through the collection to adjust approximate PageRank values to reflect the theoretical true value more closely.

What is Random Walk?
A random walk is a mathematical object, known as a stochastic or random process, that describes a path that consists of a succession of random steps on some mathematical space such as the integers. An elementary example of a random walk is the random walk on the integer number line, which starts at 0 and at each step moves +1 or -1 with equal probability.

Code:

In [4]:
```
# 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
```

In [5]:
```
# Main
# 1. Create a directed graph with N nodes
g = nx.DiGraph()
N = 10
g.add_nodes_from(range(N))

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

In [6]:
```
# 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)

# 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:

In [6]:
```
# 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)

# 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=", ")
```

```
PageRank using Random Walk Method
[5 4 8 7 6 1 9 3 0 2]
PageRank using inbuilt pagerank method
5, 4, 8, 7, 6, 1, 9, 3, 0, 2,
```

Practical 3

Aim: Perform Spam Classifier.

Theory:

What is a Spam Classifier?
Spam messages are messages sent to a large group of recipients without their prior consent, typically advertising for goods and services or business opportunities. In the recent period, the percentage of scam messages amongst spam have increased sharply. Scam messages typically trick people into giving away money or personal details by offering an attractive or false deal. Based on the statistics from the Singapore Police Force, from January till June 2020, the amount cheated through scams have increased by more than S$8 million! A spam message classification is a step towards building a tool for scam message identification and early scam detection. Several research work have employed neural network to classify unwanted emails as spam by applying content-based filtering. These techniques decide the properties by either computing the rate of occurrence of keywords or patterns in the email messages.

Code & Output:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sp
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
```

In [2]:
```
dataset = pd.read_csv(r"C:\Users\LENOVO\Downloads\spam.csv", encoding='latin-1')
dataset.head()
```

Out[2]:

	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

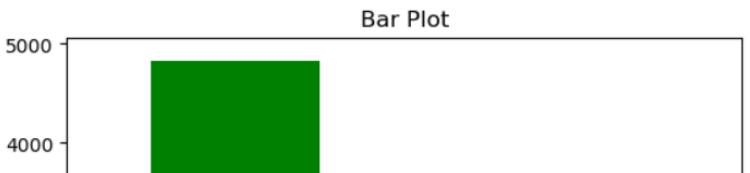
In [3]:
```
#removing unnamed columns
dataset = dataset.drop('Unnamed: 2', 1)
dataset = dataset.drop('Unnamed: 3', 1)
dataset = dataset.drop('Unnamed: 4', 1)

dataset.head()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_26524\1082831009.py:2: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-only.
  dataset = dataset.drop('Unnamed: 2', 1)
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_26524\1082831009.py:3: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-only.
  dataset = dataset.drop('Unnamed: 3', 1)
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_26524\1082831009.py:4: FutureWarning: In a future version of pandas all arguments
of DataFrame.drop except for the argument 'labels' will be keyword-only.
  dataset = dataset.drop('Unnamed: 4', 1)
```

Out[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...

In [5]:
```
count_Class=pd.value_counts(dataset["label"], sort= True)
count_Class.plot(kind = 'bar',color = ["green","red"])
plt.title('Bar Plot')
plt.show();
```



In [6]:
```python
f = feature_extraction.text.CountVectorizer(stop_words = 'english')
X = f.fit_transform(dataset["message"])
np.shape(X)
```

Out[6]:
```
(5572, 8404)
```

In [7]:
```python
# 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))
print(score_train)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [8]:
```python
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))
    print(precision_test[count])
    count = count + 1
```

```
0.8209982788296041
0.8260869565217391
0.8376068376068376
0.8448275862068966
0.8676207513416816
0.8994413407821229
0.903954802259887
0.9158699808795411
0.9241245136186771
0.9382470119521913
0.9435483870967742
0.9454545454545454
0.9490835030549898
0.9548254620123203
0.9566115702479339
0.9605809128630706
0.9746300211416491
0.974522292993630 6
0.9744680851063829
0.9764957264957265
0.9806034482758621
```

```
0.9974226804123711
0.9974160206718347
0.9973958333333334
0.9973958333333334
0.9973958333333334
0.9973821989528796
0.9973753280839895
0.9973684210526316
0.9973614775725593
0.9973474801061007
0.9973474801061007
0.9973474801061007
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
```

In [9]:
```
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)
```

Out[9]:

	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

In [10]:
```
best_index = models['Test Precision'].idxmax()
models.iloc[best_index, :]
```

Out[10]:
```
alpha             10.670010
Train Accuracy     0.977259
Test Accuracy      0.962574
Test Recall        0.720307
Test Precision     1.000000
Name: 97, dtype: float64
```

In [11]:
```
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)
preci
print
```

```
Preci
```

In [12]:
```
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.utils import pad_sequences
```

In [13]:
```
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)
```

In [14]:
```
# convert to sequence of integers
X = tokenizer.texts_to_sequences(X)
```

In [17]:
```
X = np.array(X)
y = np.array(Y)
```

In [18]:
```
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()
```

```
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
_________________________________________________________________
```

In [19]:
```
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)
```

```
Epoch 1/50
140/140 - 7s - loss: 0.5265 - accuracy: 0.8591 - val_loss: 0.3832 - val_accuracy: 0.8700 - 7s/epoch - 52ms/step
Epoch 2/50
140/140 - 1s - loss: 0.3819 - accuracy: 0.8649 - val_loss: 0.3676 - val_accuracy: 0.8700 - 952ms/epoch - 7ms/step
Epoch 3/50
140/140 - 1s - loss: 0.3705 - accuracy: 0.8649 - val_loss: 0.3548 - val_accuracy: 0.8700 - 1s/epoch - 10ms/step
Epoch 4/50
140/140 - 1s - loss: 0.3549 - accuracy: 0.8649 - val_loss: 0.3373 - val_accuracy: 0.8700 - 1s/epoch - 9ms/step
Epoch 5/50
140/140 - 1s - loss: 0.3254 - accuracy: 0.8647 - val_loss: 0.2909 - val_accuracy: 0.8700 - 859ms/epoch - 6ms/step
Epoch 6/50
140/140 - 1s - loss: 0.2597 - accuracy: 0.8708 - val_loss: 0.2118 - val_accuracy: 0.8996 - 913ms/epoch - 7ms/step
Epoch 7/50
140/140 - 1s - loss: 0.1867 - accuracy: 0.9307 - val_loss: 0.1528 - val_accuracy: 0.9498 - 871ms/epoch - 6ms/step
Epoch 8/50
140/140 - 1s - loss: 0.1425 - accuracy: 0.9549 - val_loss: 0.1260 - val_accuracy: 0.9623 - 1s/epoch - 9ms/step
Epoch 9/50
140/140 - 1s - loss: 0.1200 - accuracy: 0.9621 - val_loss: 0.1048 - val_accuracy: 0.9632 - 1s/epoch - 9ms/step
Epoch 10/50
140/140 - 1s - loss: 0.1064 - accuracy: 0.9670 - val_loss: 0.0920 - val_accuracy: 0.9677 - 915ms/epoch - 7ms/step
Epoch 11/50
140/140 - 1s - loss: 0.0962 - accuracy: 0.9688 - val_loss: 0.0848 - val_accuracy: 0.9695 - 955ms/epoch - 7ms/step
Epoch 12/50
140/140 - 1s - loss: 0.0884 - accuracy: 0.9711 - val_loss: 0.0778 - val_accuracy: 0.9722 - 1s/epoch - 8ms/step
Epoch 13/50
140/140 - 1s - loss: 0.0817 - accuracy: 0.9737 - val_loss: 0.0731 - val_accuracy: 0.9749 - 1s/epoch - 8ms/step
Epoch 14/50
140/140 - 1s - loss: 0.0764 - accuracy: 0.9744 - val_loss: 0.0653 - val_accuracy: 0.9767 - 1s/epoch - 8ms/step
Epoch 15/50
140/140 - 1s - loss: 0.0709 - accuracy: 0.9764 - val_loss: 0.0636 - val_accuracy: 0.9794 - 1s/epoch - 8ms/step
Epoch 16/50
140/140 - 1s - loss: 0.0674 - accuracy: 0.9778 - val_loss: 0.0612 - val_accuracy: 0.9794 - 1s/epoch - 7ms/step
```

```
Epoch 17/50
140/140 - 1s - loss: 0.0641 - accuracy: 0.9785 - val_loss: 0.0563 - val_accuracy: 0.9803 - 1s/epoch - 8ms/step
Epoch 18/50
140/140 - 1s - loss: 0.0613 - accuracy: 0.9780 - val_loss: 0.0546 - val_accuracy: 0.9830 - 844ms/epoch - 6ms/step
Epoch 19/50
140/140 - 1s - loss: 0.0600 - accuracy: 0.9809 - val_loss: 0.0564 - val_accuracy: 0.9839 - 1s/epoch - 7ms/step
Epoch 20/50
140/140 - 1s - loss: 0.0571 - accuracy: 0.9805 - val_loss: 0.0510 - val_accuracy: 0.9830 - 902ms/epoch - 6ms/step
Epoch 21/50
140/140 - 1s - loss: 0.0552 - accuracy: 0.9816 - val_loss: 0.0497 - val_accuracy: 0.9848 - 1s/epoch - 8ms/step
Epoch 22/50
140/140 - 1s - loss: 0.0537 - accuracy: 0.9825 - val_loss: 0.0500 - val_accuracy: 0.9839 - 929ms/epoch - 7ms/step
Epoch 23/50
140/140 - 1s - loss: 0.0527 - accuracy: 0.9814 - val_loss: 0.0479 - val_accuracy: 0.9848 - 1s/epoch - 7ms/step
Epoch 24/50
140/140 - 1s - loss: 0.0511 - accuracy: 0.9832 - val_loss: 0.0469 - val_accuracy: 0.9874 - 1s/epoch - 8ms/step
Epoch 25/50
140/140 - 1s - loss: 0.0491 - accuracy: 0.9834 - val_loss: 0.0490 - val_accuracy: 0.9857 - 995ms/epoch - 7ms/step
Epoch 26/50
140/140 - 1s - loss: 0.0486 - accuracy: 0.9845 - val_loss: 0.0457 - val_accuracy: 0.9883 - 1s/epoch - 8ms/step
Epoch 27/50
140/140 - 1s - loss: 0.0467 - accuracy: 0.9843 - val_loss: 0.0449 - val_accuracy: 0.9883 - 1s/epoch - 8ms/step
Epoch 28/50
140/140 - 1s - loss: 0.0453 - accuracy: 0.9847 - val_loss: 0.0446 - val_accuracy: 0.9883 - 1s/epoch - 9ms/step
Epoch 29/50
140/140 - 1s - loss: 0.0445 - accuracy: 0.9861 - val_loss: 0.0441 - val_accuracy: 0.9883 - 1s/epoch - 7ms/step
Epoch 30/50
140/140 - 1s - loss: 0.0448 - accuracy: 0.9859 - val_loss: 0.0445 - val_accuracy: 0.9865 - 651ms/epoch - 5ms/step
Epoch 31/50
140/140 - 1s - loss: 0.0435 - accuracy: 0.9850 - val_loss: 0.0447 - val_accuracy: 0.9865 - 900ms/epoch - 6ms/step
Epoch 32/50
140/140 - 1s - loss: 0.0420 - accuracy: 0.9856 - val_loss: 0.0478 - val_accuracy: 0.9839 - 885ms/epoch - 6ms/step
Epoch 33/50
140/140 - 1s - loss: 0.0424 - accuracy: 0.9861 - val_loss: 0.0424 - val_accuracy: 0.9901 - 1s/epoch - 8ms/step
Epoch 34/50
140/140 - 1s - loss: 0.0394 - accuracy: 0.9872 - val_loss: 0.0422 - val_accuracy: 0.9901 - 1s/epoch - 8ms/step
```

```
Epoch 35/50
140/140 - 1s - loss: 0.0393 - accuracy: 0.9870 - val_loss: 0.0433 - val_accuracy: 0.9848 - 846ms/epoch - 6ms/step
Epoch 36/50
140/140 - 1s - loss: 0.0378 - accuracy: 0.9874 - val_loss: 0.0425 - val_accuracy: 0.9892 - 982ms/epoch - 7ms/step
Epoch 37/50
140/140 - 1s - loss: 0.0369 - accuracy: 0.9877 - val_loss: 0.0417 - val_accuracy: 0.9901 - 1s/epoch - 7ms/step
Epoch 38/50
140/140 - 1s - loss: 0.0377 - accuracy: 0.9872 - val_loss: 0.0415 - val_accuracy: 0.9901 - 1s/epoch - 8ms/step
Epoch 39/50
140/140 - 1s - loss: 0.0378 - accuracy: 0.9883 - val_loss: 0.0422 - val_accuracy: 0.9892 - 991ms/epoch - 7ms/step
Epoch 40/50
140/140 - 1s - loss: 0.0360 - accuracy: 0.9872 - val_loss: 0.0413 - val_accuracy: 0.9892 - 988ms/epoch - 7ms/step
Epoch 41/50
140/140 - 1s - loss: 0.0361 - accuracy: 0.9883 - val_loss: 0.0464 - val_accuracy: 0.9883 - 1s/epoch - 8ms/step
Epoch 42/50
140/140 - 1s - loss: 0.0342 - accuracy: 0.9888 - val_loss: 0.0411 - val_accuracy: 0.9901 - 1s/epoch - 9ms/step
Epoch 43/50
140/140 - 1s - loss: 0.0341 - accuracy: 0.9892 - val_loss: 0.0412 - val_accuracy: 0.9892 - 1s/epoch - 8ms/step
Epoch 44/50
140/140 - 1s - loss: 0.0328 - accuracy: 0.9897 - val_loss: 0.0421 - val_accuracy: 0.9874 - 1s/epoch - 9ms/step
Epoch 45/50
140/140 - 1s - loss: 0.0335 - accuracy: 0.9890 - val_loss: 0.0420 - val_accuracy: 0.9910 - 1s/epoch - 9ms/step
Epoch 46/50
140/140 - 1s - loss: 0.0324 - accuracy: 0.9892 - val_loss: 0.0423 - val_accuracy: 0.9910 - 1s/epoch - 8ms/step
Epoch 47/50
140/140 - 1s - loss: 0.0326 - accuracy: 0.9892 - val_loss: 0.0412 - val_accuracy: 0.9901 - 1s/epoch - 9ms/step
Epoch 48/50
140/140 - 1s - loss: 0.0306 - accuracy: 0.9899 - val_loss: 0.0424 - val_accuracy: 0.9874 - 1s/epoch - 8ms/step
Epoch 49/50
140/140 - 1s - loss: 0.0314 - accuracy: 0.9895 - val_loss: 0.0419 - val_accuracy: 0.9910 - 1s/epoch - 9ms/step
Epoch 50/50
140/140 - 1s - loss: 0.0301 - accuracy: 0.9901 - val_loss: 0.0415 - val_accuracy: 0.9883 - 1s/epoch - 7ms/step
```

```
In [20]: results = model.evaluate(X_test, y_test)
         loss = results[0]
         accuracy = results[1]

         35/35 [==============================] - 0s 7ms/step - loss: 0.0415 - accuracy: 0.9883

In [21]: print(f"[+] Accuracy: {accuracy*100:.2f}%")

         [+] Accuracy: 98.83%
```

```
In [22]: from keras.preprocessing import sequence
         from keras.utils import pad_sequences

In [23]: #Defining the function
         def get_predictions(txts):
             txts = tokenizer.texts_to_sequences(txts)
             txts = pad_sequences(txts, maxlen=max_length)
             preds = model.predict(txts)
             if(preds[0] > 0.5):
                 print("SPAM MESSAGE")

             else:
                 print('NOT SPAM')
```

```
In [25]: txts=["You have won a free ticket to las vegas. Contact now"]

         get_predictions(txts)

         1/1 [==============================] - 0s 93ms/step
         SPAM MESSAGE

In [26]: txts=["Hey there call me asap!!"]

         get_predictions(txts)

         1/1 [==============================] - 0s 56ms/step
         NOT SPAM
```

Practical 4

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

Theory:

What is Text Mining?

Text Mining is the process of deriving meaningful information from natural language text.

What is Webpage Pre-processing?

Preprocessing stage helps to clean the records and determine the interesting user patterns and session creation. Data pre-processing is an important job of Web usage mining application. So, data must be processed before applying data mining methods to determine user access patterns

from web log.

Code:

In [1]:
```
import requests
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
```

In [2]:
```
# IMDB's homepage
imdb_url = 'https://www.imdb.com'
```

In [3]:
```
# Use requests to retrieve data from a given URL
imdb_response = requests.get(imdb_url)
```

In [4]:
```
# Parse the whole HTML page using BeautifulSoup
imdb_soup = BeautifulSoup(imdb_response.text, 'html.parser')
```

In [9]:
```
# Title of the parsed page
imdb_soup.title
```

In [11]:
```
imdb_soup.title.string
```

In [12]:
```
trailers = imdb_soup.find('div', {'class': 'ab_hero'})
```

In [15]:
```
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()
```

In [16]:
```
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()
```

In [17]:
```
# Find all links
links = [link.get('href') for link in imdb_soup.find_all('a')]
```

In [18]:
```
# Add homepage and keep the unique links
fixed_links = set([''.join([imdb_url, link]) for link in links if link])
```

In [19]:
```
# Box Office Mojo - UK Weekend box office
boxofficemojo_url = 'https://www.boxofficemojo.com/intl/uk/?yr=2019&wk=33&currency=local'
```

In [20]:
```
# Use requests to retrieve data from a given URL
bom_response = requests.get(boxofficemojo_url)
```

In [21]:
```
# 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'))}")
table = bom_soup.find_all('table')[0]
table
table.find_all('tr')[0].contents
table.find_all('tr')[0].text
```

In [22]:
```
# Print text "consumes" the newline characters
print(table.find_all('tr')[0].text)
```

In [25]:
```
table.find_all('tr')[0].text.split('\n')
```

In [29]:
```
print(f'(MOVIES, COLUMNS) -> {data.shape}')
for row in table.find_all('tr')[1:-1]:
    s = pd.Series([data.text for data in row.find_all('td')])
    lst.append(s)
```

In [28]:
```
data = pd.concat(lst, axis=1).T
data.head(2)
```

In [30]:
```
print(f'% OF MISSING VALUES PER COLUMN\n{(data.isnull().sum() / data.shape[0]) * 100}')
```

Output:

In [9]:
```
# Title of the parsed page
imdb_soup.title
```

Out[9]:
```
<title>403 Forbidden</title>
```

In [11]:
```
imdb_soup.title.string
```

In [21]:
```
# 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'))}")
table = bom_soup.find_all('table')[0]
table
table.find_all('tr')[0].contents
table.find_all('tr')[0].text
```

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

Out[21]:
```
'DatesTop 10 Gross%± LWOverall Gross%± LWReleases#1 Release\nGenre\nBudget\nRunning Time\nWeekLong Weekend\n'
```

In [22]:
```
# Print text "consumes" the newline characters
print(table.find_all('tr')[0].text)
```

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

In [25]:
```
table.find_all('tr')[0].text.split('\n')
```

Out[25]:
```
['DatesTop 10 Gross%± LWOverall Gross%± LWReleases#1 Release',
 'Genre',
 'Budget',
 'Running Time',
 'WeekLong Weekend',
 '']
```

In [28]:
```
data = pd.concat(lst, axis=1).T
data.head(2)
```

Out[28]:

	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

In [29]:
```
print(f'(MOVIES, COLUMNS) -> {data.shape}')
```

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

In [30]:
```
print(f'% OF MISSING VALUES PER COLUMN\n{(data.isnull().sum() / data.shape[0]) * 100}')
```

```
% 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 5

Aim: Apriori Algorithm implementation in case study.

Theory:

What is the Apriori algorithm?

- It is part of an Unsupervised Machine Learning algorithm that is a part of association-rule based learning.
- The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions.
- With the help of these association rules, it determines how strongly or how weakly two objects are connected.
- For example: a frequent itemset{Chicken, Clothes, Milk} [sup = 3/7] and one rule from the frequent itemset Clothes → Milk, Chicken [sup = 3/7, conf =3/3].
- This algorithm uses a breadth-first search and Hash Tree to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset. This algorithm was given by R. Agrawal and Srikant in the year 1994.
- It is mainly used for market basket analysis and helps to find those products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.
- To use the Apriori algorithm, two steps have to be performed:

  Step 1: Mining all frequent itemsets:

  - A frequent itemset is an itemset whose support is ≥ minsup.
  - The key idea is to use the apriori property (downward closure property), where any subsets of a frequent itemset are also frequent itemsets.
  - The items in I are sorted in lexicographic order (which is a total order). The order is used throughout the algorithm in each itemset.
  - {w[1], w[2], …, w[k]} represents a k-itemset w consisting of items w[1], w[2], …, w[k], where w[1] < w[2] < … < w[k] according to the total order.
  - The apriori candidate generation uses the candidate-gen function that takes Fk-1 and returns a superset (called the candidates) of the set of all frequent k-itemsets. It has two steps:

  a. join step: Generate all possible candidate itemsets Ck of length k
  b. prune step: Remove those candidates in Ck that cannot be frequent.

  Step 2: Generating rules from frequent itemsets:

  - One more step is needed to generate association rules.
  - For each frequent itemset X, For each proper nonempty subset A of X,
  - Let B = X - A.

- A → B is an association rule if:

a. Confidence(A → B) ≥ minconf,
b. support(A → B) = support(A∪B) = support(X)
c. confidence(A → B) = support(A ∪ B) / support(A)

Case Study:

Minimum Support = 50%

Minimum Confidence = 70%

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 6

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

Theory:

What is a Basic Crawler?

The basic crawler is a sequential crawler. It goes through a list of URLs called as seeds. Seeds can be any list of starting URLs. In its simple form, a crawler starts from a set of seed pages and then uses the links within them to fetch other pages. The process repeats until a sufficient number of pages are visited or some other objective is achieved. The order of pages visited by the crawlers is determined by a frontier data structure. The stop criterion can be anything.

Code:

In [ ]:
```
import requests
url = 'https://en.wikipedia.org/wiki/Stock_market'
```

In [ ]:
```
# Connect to the url using requests.get
response = requests.get(url)
response.status_code
```

In [ ]:
```
# Add a timeout to prevent hanging
response = requests.get(url, timeout=3)
response.status_code
```

In [ ]:
```
import requests
url = 'https://en.wikipedia.org/wiki/Stock_market'
```

In [ ]:
```
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')
```

In [ ]:
```
# Print out the contents of a request's response
print(f"{'---'*20}\n\tContents of Response.items():\n{'---'*20}")

for k,v in response.headers.items():
    print(f"{k:{25}}: {v:{40}}") # Note: add :{number} inside of a

for k,v in response.headers.items():
    print(f"{k}: {v}") # Note: add :{number} inside of a

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

In [ ]:
```
from bs4 import BeautifulSoup

# Define Url and establish connection
url = 'https://en.wikipedia.org/wiki/Stock_market'
response = requests.get(url, timeout=3)
```

In [ ]:
```
# Feed the response's .content into BeautifulSoup
page_content = response.content
soup = BeautifulSoup(page_content,'lxml') #'html.parser')
```

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

In [ ]:
```
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
```

In [ ]:
```
title = soup.head.title
print(title.parent.name)
```

In [ ]:
```
results = soup.find_all()
results
```

Output:

In [2]:
```
# Connect to the url using requests.get
response = requests.get(url)
response.status_code
```

Out[2]: 200

In [3]:
```
# Add a timeout to prevent hanging
response = requests.get(url, timeout=3)
response.status_code
```

Out[3]: 200

In [5]:
```
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')
```

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

In [6]:
```
# Print out the contents of a request's response
print(f"{'---'*20}\n\tContents of Response.items():\n{'---'*20}")

for k,v in response.headers.items():
    print(f"{k:{25}}: {v:{40}}") # Note: add :{number} inside of a

for k,v in response.headers.items():
    print(f"{k}: {v}") # Note: add :{number} inside of a

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

```
------------------------------------------------------------
        Contents of Response.items():
------------------------------------------------------------
date                      : Fri, 26 May 2023 11:47:58 GMT
vary                      : Accept-Encoding,Cookie,Authorization
server                    : ATS/9.1.4
x-content-type-options    : nosniff
content-language          : en
last-modified             : Thu, 25 May 2023 08:36:52 GMT
content-type              : text/html; charset=UTF-8
content-encoding          : gzip
age                       : 4909
x-cache                   : cp5024 hit, cp5024 hit/7
x-cache-status            : hit-front
server-timing             : cache;desc="hit-front", host;desc="cp5024"
strict-transport-security: max-age=106384710; includeSubDomains; preload
report-to                 : { "group": "wm_nel", "max_age": 604800, "endpoints": [{ "url": "https://intake-logging.wikimedia.or
g/v1/events?stream=w3c.reportingapi.network_error&schema_uri=/w3c/reportingapi/network_error/1.0.0" }] }
nel                       : { "report_to": "wm_nel", "max_age": 604800, "failure_fraction": 0.05, "success_fraction": 0.0}
set-cookie                : WMF-Last-Access=26-May-2023;Path=/;HttpOnly;secure;Expires=Tue, 27 Jun 2023 12:00:00 GMT, WMF-Last-A
ccess-Global=26-May-2023;Path=/;Domain=.wikipedia.org;HttpOnly;secure;Expires=Tue, 27 Jun 2023 12:00:00 GMT, WMF-DP=713;Path=/;
HttpOnly;secure;Expires=Sat, 27 May 2023 00:00:00 GMT, GeoIP=IN:MH:Mumbai:19.07:72.89:v4; Path=/; secure; Domain=.wikipedia.org
x-client-ip               : 2402:3a80:676:d1f:a4a3:ff44:bf5b:beca
cache-control             : private, s-maxage=0, max-age=0, must-revalidate
accept-ranges             : bytes
content-length            : 70920
date: Fri, 26 May 2023 11:47:58 GMT
vary: Accept-Encoding,Cookie,Authorization
server: ATS/9.1.4
x-content-type-options: nosniff
content-language: en
last-modified: Thu, 25 May 2023 08:36:52 GMT
content-type: text/html; charset=UTF-8
content-encoding: gzip
age: 4909
x-cache: cp5024 hit, cp5024 hit/7
x-cache-status: hit-front
server-timing: cache;desc="hit-front", host;desc="cp5024"
strict-transport-security: max-age=106384710; includeSubDomains; preload
report-to: { "group": "wm_nel", "max_age": 604800, "endpoints": [{ "url": "https://intake-logging.wikimedia.org/v1/events?strea
m=w3c.reportingapi.network_error&schema_uri=/w3c/reportingapi/network_error/1.0.0" }] }
nel: { "report_to": "wm_nel", "max_age": 604800, "failure_fraction": 0.05, "success_fraction": 0.0}
set-cookie: WMF-Last-Access=26-May-2023;Path=/;HttpOnly;secure;Expires=Tue, 27 Jun 2023 12:00:00 GMT, WMF-Last-Access-Global=26
-May-2023;Path=/;Domain=.wikipedia.org;HttpOnly;secure;Expires=Tue, 27 Jun 2023 12:00:00 GMT, WMF-DP=713;Path=/;HttpOnly;secur
e;Expires=Sat, 27 May 2023 00:00:00 GMT, GeoIP=IN:MH:Mumbai:19.07:72.89:v4; Path=/; secure; Domain=.wikipedia.org
x-client-ip: 2402:3a80:676:d1f:a4a3:ff44:bf5b:beca
cache-control: private, s-maxage=0, max-age=0, must-revalidate
accept-ranges: bytes
content-length: 70920
Status code: 200
Status code:                  200
Status code: ----------------200
```

In [9]:
```
# Preview soup contents using .prettify()
print(soup.prettify()[:2000])
```

```
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-enabled vector-feature-main-menu-pinned-disabled vector-feature-limited-width-enabled vector-feature-limited-width-content-enabled vector-feature-zebra-design-disabled" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Stock market - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-enabled vector-feature-main-menu-pinned-disabled vector-feature-limited-width-enabled vector-feature-limited-width-content-enabled vector-feature-zebra-design-disabled";(function(){var cookie=document.cookie.match(/(?:^|; )enwikimwclientprefs=([^;]+)/);if(cookie){var featureName=cookie[1];document.documentElement.className=document.documentElement.className.replace(featureName+'-enabled',featureName+'-disabled');}})());RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","May","June","July","August","September","October","November","December"],"wgRequestId":"31028b5a-7be9-4813-93b7-af1e5bac1b8f","wgCSPNonce":false,
"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Stock_market","wgTitle":"Stock market","wgCurRevisionId":1151433545,"wgRevisionId":1151433545,"wgArticleId":52328,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 errors: missing periodical","Webarchive template wayback links","CS1 maint: multiple names: authors list","Wikipedia indefinitely semi-protected
```

In [10]:
```
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
```

```
<a class="mw-jump-link" href="#bodyContent">Jump to content</a>
```

```


<div class="vector-header-container">
<header class="vector-header mw-header">
<div class="vector-header-start">
<nav aria-label="Site" class="vector-main-menu-landmark" role="navigation">
<div class="vector-menu vector-dropdown vector-menu-dropdown vector-main-menu-dropdown mw-ui-icon-flush-left mw-ui-icon-flush-right" id="vector-main-menu-dropdown">
<input aria-haspopup="true" aria-label="Main menu" class="vector-menu-checkbox" data-event-name="ui.dropdown-vector-main-menu-dropdown" id="vector-main-menu-dropdown-checkbox" role="button" type="checkbox"/>
<label aria-hidden="true" class="vector-menu-heading mw-ui-button mw-ui-quiet mw-ui-icon-element" for="vector-main-menu-dropdown-checkbox" id="vector-main-menu-dropdown-label">
<span class="mw-ui-icon mw-ui-icon-menu mw-ui-icon-wikimedia-menu"></span>
<span class="vector-menu-heading-label">Main menu</span>
</label>
<div class="vector-menu-content vector-dropdown-content">
<div class="vector-unpinned-container" id="vector-main-menu-unpinned-container">
<div class="vector-main-menu vector-pinnable-element" id="vector-main-menu">
<div class="vector-pinnable-header vector-main-menu-pinnable-header vector-pinnable-header-unpinned" data-feature-name="main-menu-pinned" data-pinnable-element-id="vector-main-menu" data-pinned-container-id="vector-main-menu-pinned-container" data-unpinned-container-id="vector-main-menu-unpinned-container">
<div class="vector-pinnable-header-label">Main menu</div>
<button class="vector-pinnable-header-toggle-button vector-pinnable-header-pin-button" data-event-name="pinnable-header.vector-main-menu.pin">move to sidebar</button>
<button class="vector-pinnable-header-toggle-button vector-pinnable-header-unpin-button" data-event-name="pinnable-header.vector-main-menu.unpin">hide</button>
</div>
```

In [11]:
```
title = soup.head.title
print(title.parent.name)
```

```
head
```

In [12]:
```
results = soup.find_all()
results
```

Out[12]:
```
[<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vector
-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-enabled vector-feature-main
-menu-pinned-disabled vector-feature-limited-width-enabled vector-feature-limited-width-content-enabled vector-feature-zebra-de
sign-disabled" dir="ltr" lang="en">
 <head>
 <meta charset="utf-8"/>
 <title>Stock market - Wikipedia</title>
 <script>document.documentElement.className="client-js vector-feature-language-in-header-enabled vector-feature-language-in-mai
n-page-header-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinne
d-enabled vector-feature-main-menu-pinned-disabled vector-feature-limited-width-enabled vector-feature-limited-width-content-en
abled vector-feature-zebra-design-disabled";(function(){var cookie=document.cookie.match(/(?:^|; )enwikimwclientprefs=([^;]
+)/);if(cookie){var featureName=cookie[1];document.documentElement.className=document.documentElement.className.replace(feature
Name+'-enabled',featureName+'-disabled');}})();RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransf
ormTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","May","June","July","Augu
st","September","October","November","December"],"wgRequestId":"31028b5a-7be9-4813-93b7-af1e5bac1b8f","wgCSPNonce":false,
 "wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Stock_market","wgTitle":"Stoc
k market","wgCurRevisionId":1151433545,"wgRevisionId":1151433545,"wgArticleId":52328,"wgIsArticle":true,"wgIsRedirect":false,"w
gAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 errors: missing periodical","Webarchive template wa
yback links","CS1 maint: multiple names: authors list","Wikipedia indefinitely semi-protected pages","Articles with short descr
iption","Short description is different from Wikidata","Use mdy dates from June 2013","Articles containing potentially dated st
atements from 2016","All articles containing potentially dated statements","Articles with limited geographic scope from Novembe
r 2020","United States-centric","All accuracy disputes","Articles with disputed statements from November 2019","All articles wi
th unsourced statements","Articles with unsourced statements from February 2017","Articles with GND identifiers"
 ,"Articles with NDL identifiers","Stock market","Private sector","Financial markets","Capital (economics)"],"wgPageContentLang
uage":"en","wgPageContentModel":"wikitext","wgRelevantPageName":"Stock_market","wgRelevantArticleId":52328,"wgIsProbablyEditabl
e":false,"wgRelevantPageIsProbablyEditable":false,"wgRestrictionEdit":["autoconfirmed"],"wgRestrictionMove":["autoconfirme
d"],"wgFlaggedRevsParams":{"tags":{"status":{"levels":1}}},"wgVisualEditor":{"pageLanguageCode":"en","pageLanguageDir":"ltr","p
ageVariantFallbacks":"en"},"wgMFDisplayWikibaseDescriptions":{"search":true,"watchlist":true,"tagline":false,"nearby":true},"wg
WMESchemaEditAttemptStepOversample":false,"wgWMEPageLength":60000,"wgNoticeProject":"wikipedia","wgMediaViewerOnClick":true,"wg
MediaViewerEnabledByDefault":true,"wgPopupsFlags":10,"wgULSCurrentAutonym":"English","wgEditSubmitButtonLabelPublish":true,"wgC
entralAuthMobileDomain":false,"wgULSPosition":"interlanguage","wgULSisCompactLinksEnabled":true,
 "wgULSisLanguageSelectorEmpty":false,"wgWikibaseItemId":"Q475000","GEHomepageSuggestedEditsEnableTopics":true,"wgGETopicsMatch
ModeEnabled":false,"wgGEStructuredTaskRejectionReasonTextInputEnabled":false,"wgGELevelingUpEnabledForUser":false};RLSTATE={"sk
ins.vector.user.styles":"ready","ext.globalCssJs.user.styles":"ready","site.styles":"ready","user.styles":"ready","skins.vecto
```

```
script%7Cext.wikimediaBadges%7Cjquery.makeCollapsible.styles%7Cmediawiki.ui.button%2Cicon%7Cskins.vector.icons%2Cstyles%7Cwikib
ase.client.init&amp;only=styles&amp;skin=vector-2022" rel="stylesheet"/>
 <script async="" src="/w/load.php?lang=en&amp;modules=startup&amp;only=scripts&amp;raw=1&amp;skin=vector-2022"></script>
 <meta content="" name="ResourceLoaderDynamicStyles"/>
 <link href="/w/load.php?lang=en&amp;modules=site.styles&amp;only=styles&amp;skin=vector-2022" rel="stylesheet"/>
 <meta content="MediaWiki 1.41.0-wmf.9" name="generator"/>
 <meta content="origin" name="referrer"/>
 <meta content="origin-when-crossorigin" name="referrer"/>
 <meta content="origin-when-cross-origin" name="referrer"/>
 <meta content="max-image-preview:standard" name="robots"/>
 <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="width=1000" name="viewport"/>
 <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="/static/apple-touch/wikipedia.png" rel="apple-touch-icon"/>
 <link href="/static/favicon/wikipedia.ico" rel="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"/>
 <link href="https://en.wikipedia.org/wiki/Stock_market" rel="canonical"/>
 <link href="https://creativecommons.org/licenses/by-sa/3.0/" rel="license"/>
 <link href="/w/index.php?title=Special:RecentChanges&amp;feed=atom" rel="alternate" title="Wikipedia Atom feed" type="applicat
ion/atom+xml"/>
 <link href="//meta.wikimedia.org" rel="dns-prefetch"/>
 <link href="//login.wikimedia.org" rel="dns-prefetch"/>
 </head>
 <body class="skin-vector skin-vector-search-vue mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-Stock_market
rootpage-Stock_market skin-vector-2022 action-view"><a class="mw-jump-link" href="#bodyContent">Jump to content</a>
```

Practical 7

Aim: Develop a focused crawler for local search.

Theory:

What is a Focused Crawler?

A focused crawler is a web crawler that collects Web pages that satisfy some specific property, by carefully prioritizing the crawl frontier and managing the hyperlink exploration process.

Some predicates may be based on simple, deterministic, and surface properties. For example, a crawler's mission may be to crawl pages from only the .jp domain. Other predicates may be softer or comparative, e.g., "crawl pages about baseball", or "crawl pages with large PageRank".

An important page property pertains to topics, leading to 'topical crawlers'. For example, a topical crawler may be deployed to collect pages about solar power, swine flu, or even more abstract concepts like controversy while minimizing resources spent fetching pages on other topics. Crawl frontier management may not be the only device used by focused crawlers; they may use a Web directory, a Web text index, backlinks, or any other Web artifact.

Code:

In [1]:
```
import requests
import lxml
from bs4 import BeautifulSoup
```

In [2]:
```
url = "https://www.rottentomatoes.com/top/bestofrt/"
f = requests.get(url)
```

In [3]:
```
url = "https://www.rottentomatoes.com/top/bestofrt/"
header = {
  'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 QI
}
f = requests.get(url, headers = header)
```

In [4]:
```
soup = BeautifulSoup(f.content,'lxml')
```

In [5]:
```
mov = soup.find_all('div',{'discovery-tiles__wrap'})
print(mov)
movies = []
```

In [6]:
```
for mov1 in mov:
    movA = mov1.find_all('a')
    for mov2 in movA:
        print(mov2['href'])
        movies.append(mov2['href'])
movies_lst = []
num = 0
```

In [7]:
```
for a in movies:
    urls = 'https://www.rottentomatoes.com' + a
    print(urls)
    movies_lst.append(urls)
    num += 1
    movie_url = urls
    movie_f = requests.get(movie_url, headers = header)
    movie_soup = BeautifulSoup(movie_f.content, 'lxml')
    movie_content = movie_soup.find('p', {
         'data-qa': 'movie-info-synopsis'
     })
    print(num, urls, '\n', 'Movie:' + a.strip())
    print('Movie info:' + movie_content.string.strip())
```

Output:

In [5]:
```
mov = soup.find_all('div',{'discovery-tiles__wrap'})
print(mov)
movies = []
```

```
[<div class="discovery-tiles__wrap" data-gridpageadsmanager="tilesWrap" data-qa="discovery-media-list">
<div class="js-tile-link" data-ems-id="fde71436-f4ed-323f-b23d-6e0ce470d4ba" data-qa="discovery-media-list-item">
<tile-dynamic isvideo="true" skeleton="panel">
<img alt="John Wick: Chapter 4" class="posterImage" loading="lazy" slot="image" src="https://resizing.flixster.com/tpLtbDZQSwBp
b3jfwoI7zMiRMjE=/206x305/v2/https://resizing.flixster.com/ayiSi8HrUS3_i-Jkru1P2oMM58Q=/ems.cHJkLWVtcy1hc3NldHMvbW92aWVzLzhiODAw
YWIyLTM1MmItNGI1ZS1iOWQzLTdjZjAzMzI1MDc2Yy5qcGc="/>
<button class="transparent" data-ems-id="fde71436-f4ed-323f-b23d-6e0ce470d4ba" data-mpx-id="2173002307606" data-public-id="rJEs
fOcpmvMx" data-type="Movie" data-videoplayeroverlaymanager="btnVideo:click" slot="imageAction">
<span class="sr-only">Watch the trailer for John Wick: Chapter 4</span>
</button>
<a data-qa="discovery-media-list-item-caption" data-track="scores" href="/m/john_wick_chapter_4" slot="caption">
<score-pairs audiencescore="93" audiencesentiment="positive" criticscertified="criticscertified" criticsscore="94" criticssenti
ment="positive">
</score-pairs>
<span class="p--small" data-qa="discovery-media-list-item-title">
          John Wick: Chapter 4
        </span>
<span class="smaller" data-qa="discovery-media-list-item-start-date">
           Streaming May 23, 2023
         </span>
</a>
</tile-dynamic>
</div>
<div class="js-tile-link" data-ems-id="fd9653ac-e47a-4efa-b3ef-6454d9d62df4" data-qa="discovery-media-list-item">
<tile-dynamic isvideo="true" skeleton="panel">
<img alt="Sisu" class="posterImage" loading="lazy" slot="image" src="https://resizing.flixster.com/UkHAG8zVRtZmmt43wSG_CHeVdRY
=/206x305/v2/https://resizing.flixster.com/LUMlPb01hdWx1gtuOyg1fs2kMyI=/ems.cHJkLWVtcy1hc3NldHMvbW92aWVzL2M0NjVmNThhLTI3MDctNDk
wOS1iZDA4LTE1YmQwODE3Yzc1Ni5wbmc="/>
<button class="transparent" data-ems-id="fd9653ac-e47a-4efa-b3ef-6454d9d62df4" data-mpx-id="2174488643659" data-public-id="c_6o
R1AHBI26" data-type="Movie" data-videoplayeroverlaymanager="btnVideo:click" slot="imageAction">
<span class="sr-only">Watch the trailer for Sisu</span>
</button>



<a data-qa="discovery-media-list-item-caption" data-track="scores" href="/m/soft_and_quiet" slot="caption">
<score-pairs audiencescore="45" audiencesentiment="negative" criticscertified="criticscertified" criticsscore="86" criticssenti
ment="positive">
</score-pairs>
<span class="p--small" data-qa="discovery-media-list-item-title">
          Soft &amp;amp; Quiet
        </span>
<span class="smaller" data-qa="discovery-media-list-item-start-date">
           Streaming Nov 4, 2022
         </span>
</a>
</tile-dynamic>
</div>
<div class="js-tile-link" data-ems-id="bd5d85e4-2ec5-3a26-bc32-f3129e85bd1b" data-qa="discovery-media-list-item">
<tile-dynamic isvideo="true" skeleton="panel">
<img alt="Furious 7" class="posterImage" loading="lazy" slot="image" src="https://resizing.flixster.com/4nG_psSvLG7C82Qf2l5HKYp
tvL4=/206x305/v2/https://flxt.tmsimg.com/assets/p10679969_p_v8_av.jpg"/>
<button class="transparent" data-ems-id="bd5d85e4-2ec5-3a26-bc32-f3129e85bd1b" data-mpx-id="2170074691775" data-public-id="YIpj
r9d9VLOu" data-type="Movie" data-videoplayeroverlaymanager="btnVideo:click" slot="imageAction">
<span class="sr-only">Watch the trailer for Furious 7</span>
</button>
<a data-qa="discovery-media-list-item-caption" data-track="scores" href="/m/furious_7" slot="caption">
<score-pairs audiencescore="82" audiencesentiment="positive" criticscertified="criticscertified" criticsscore="81" criticssenti
ment="positive">
</score-pairs>
<span class="p--small" data-qa="discovery-media-list-item-title">
          Furious 7
        </span>
<span class="smaller" data-qa="discovery-media-list-item-start-date">
           Streaming Mar 1, 2016
         </span>
</a>
</tile-dynamic>
</div>
</div>]
```

In [6]:
```
for mov1 in mov:
    movA = mov1.find_all('a')
    for mov2 in movA:
        print(mov2['href'])
        movies.append(mov2['href'])
movies_lst = []
num = 0
```

```
/m/john_wick_chapter_4
/m/sisu_2022
/m/missing_2023
/m/dungeons_and_dragons_honor_among_thieves
/m/guy_ritchies_the_covenant
/m/air_2023
/m/spider_man_into_the_spider_verse
/m/leave_no_trace
/m/evil_dead_rise
/m/top_gun_maverick
/m/avatar_the_way_of_water
/m/the_little_mermaid_1989
/m/huesera_the_bone_woman
/m/everything_everywhere_all_at_once
/m/the_stranger_2022
/m/x_2022
/m/avengers_endgame
/m/polite_society
/m/m3gan
/m/creed_iii
/m/puss_in_boots_the_last_wish
/m/the_menu
/m/she_said
/m/infinity_pool_2023
/m/scream_vi
/m/avengers_infinity_war
/m/soft_and_quiet
/m/furious_7
```

In [7]:
```
for a in movies:
    urls = 'https://www.rottentomatoes.com' + a
    print(urls)
    movies_lst.append(urls)
    num += 1
    movie_url = urls
    movie_f = requests.get(movie_url, headers = header)
    movie_soup = BeautifulSoup(movie_f.content, 'lxml')
    movie_content = movie_soup.find('p', {
        'data-qa': 'movie-info-synopsis'
     })
    print(num, urls, '\n', 'Movie:' + a.strip())
    print('Movie info:' + movie_content.string.strip())
```

```
https://www.rottentomatoes.com/m/john_wick_chapter_4
1 https://www.rottentomatoes.com/m/john_wick_chapter_4
 Movie:/m/john_wick_chapter_4
Movie info:John Wick (Keanu Reeves) uncovers a path to defeating The High Table. But before he can earn his freedom, Wick must face off against a new enemy with powerful alliances across the globe and forces that turn old friends into foes.
https://www.rottentomatoes.com/m/sisu_2022
2 https://www.rottentomatoes.com/m/sisu_2022
 Movie:/m/sisu_2022
Movie info:During the last desperate days of WWII, a solitary prospector (Jorma Tommila) crosses paths with Nazis on a scorched-earth retreat in northern Finland. When the Nazis steal his gold, they quickly discover that they have just tangled with no ordinary miner. While there is no direct translation for the Finnish word "sisu", this legendary ex-commando will embody what sisu means: a white-knuckled form of courage and unimaginable determination in the face of overwhelming odds. And no matter what the Nazis throw at him, the one-man death squad will go to outrageous lengths to get his gold back -- even if it means killing every last Nazi in his path.
https://www.rottentomatoes.com/m/missing_2023
3 https://www.rottentomatoes.com/m/missing_2023
 Movie:/m/missing_2023
Movie info:When her mother (Nia Long) disappears while on vacation in Colombia with her new boyfriend, June's (Storm Reid) search for answers is hindered by international red tape. Stuck thousands of miles away in Los Angeles, June creatively uses all the latest technology at her fingertips to try and find her before it's too late. But as she digs deeper, her digital sleuthing raises more questions than answers... and when June unravels secrets about her mom, she discovers that she never really knew her at all.
https://www.rottentomatoes.com/m/dungeons_and_dragons_honor_among_thieves
4 https://www.rottentomatoes.com/m/dungeons_and_dragons_honor_among_thieves
 Movie:/m/dungeons_and_dragons_honor_among_thieves
Movie info:A charming thief and a band of unlikely adventurers undertake an epic heist to retrieve a lost relic, but things go dangerously awry when they run afoul of the wrong people. Dungeons & Dragons: Honor Among Thieves brings the rich world and playful spirit of the legendary roleplaying game to the big screen in a hilarious and action-packed adventure.
https://www.rottentomatoes.com/m/guy_ritchies_the_covenant
5 https://www.rottentomatoes.com/m/guy_ritchies_the_covenant
 Movie:/m/guy_ritchies_the_covenant
Movie info:Guy Ritchie's The Covenant follows US Army Sergeant John Kinley (Jake Gyllenhaal) and Afghan interpreter Ahmed (Dar Salim). After an ambush, Ahmed goes to Herculean lengths to save Kinley’s life. When Kinley learns that Ahmed and his family were not given safe passage to America as promised, he must repay his debt by returning to the war zone to retrieve them before the Taliban hunts them down first.
https://www.rottentomatoes.com/m/air_2023
6 https://www.rottentomatoes.com/m/air_2023
 Movie:/m/air_2023
Movie info:From award-winning director Ben Affleck, AIR reveals the unbelievable game-changing partnership between a then-rookie Michael Jordan and Nike's fledgling basketball division which revolutionized the world of sports and contemporary culture with the Air Jordan brand. This moving story follows the career-defining gamble of an unconventional team with everything on the line, the uncompromising vision of a mother who knows the worth of her son’s immense talent, and the basketball phenom who would become the greatest of all time.
https://www.rottentomatoes.com/m/spider_man_into_the_spider_verse
7 https://www.rottentomatoes.com/m/spider_man_into_the_spider_verse
 Movie:/m/spider_man_into_the_spider_verse
Movie info:Bitten by a radioactive spider in the subway, Brooklyn teenager Miles Morales suddenly develops mysterious powers that transform him into the one and only Spider-Man. When he meets Peter Parker, he soon realizes that there are many others who share his special, high-flying talents. Miles must now use his newfound skills to battle the evil Kingpin, a hulking madman who can open portals to other universes and pull different versions of Spider-Man into our world.
https://www.rottentomatoes.com/m/leave_no_trace
8 https://www.rottentomatoes.com/m/leave_no_trace
 Movie:/m/leave_no_trace
Movie info:A father and daughter live a perfect but mysterious existence in Forest Park, a beautiful nature reserve near Portland, Ore., rarely making contact with the world. But when a small mistake tips them off to authorities, they are sent on an increasingly erratic journey in search of a place to call their own.
```

https://www.rottentomatoes.com/m/infinity_pool_2023
24 https://www.rottentomatoes.com/m/infinity_pool_2023
 Movie:/m/infinity_pool_2023
Movie info:While staying at an isolated island resort, James (Alexander Skarsgård) and Em (Cleopatra Coleman) are enjoying a pe
rfect vacation of pristine beaches, exceptional staff, and soaking up the sun. But guided by the seductive and mysterious Gabi
(Mia Goth), they venture outside the resort grounds and find themselves in a culture filled with violence, hedonism, and untold
horror. A tragic accident leaves them facing a zero tolerance policy for crime: either you'll be executed, or, if you're rich e
nough to afford it, you can watch yourself die instead.
https://www.rottentomatoes.com/m/scream_vi
25 https://www.rottentomatoes.com/m/scream_vi
 Movie:/m/scream_vi
Movie info:Following the latest Ghostface killings, the four survivors leave Woodsboro behind and start a fresh chapter.
https://www.rottentomatoes.com/m/avengers_infinity_war
26 https://www.rottentomatoes.com/m/avengers_infinity_war
 Movie:/m/avengers_infinity_war
Movie info:Iron Man, Thor, the Hulk and the rest of the Avengers unite to battle their most powerful enemy yet -- the evil Than
os. On a mission to collect all six Infinity Stones, Thanos plans to use the artifacts to inflict his twisted will on reality.
The fate of the planet and existence itself has never been more uncertain as everything the Avengers have fought for has led up
to this moment.
https://www.rottentomatoes.com/m/soft_and_quiet
27 https://www.rottentomatoes.com/m/soft_and_quiet
 Movie:/m/soft_and_quiet
Movie info:Playing out in real time, SOFT & QUIET is a runaway train that follows a single afternoon in the life of an elementa
ry school teacher, Emily, who organizes an inaugural club meeting of like-minded women. When they all decide to move the meetin
g to Emily's house to keep the wine flowing, they stop at the local store to pick up refreshments. At the store, an altercation
breaks out between a woman from Emily's past and the group, leading to a volatile chain of events.
https://www.rottentomatoes.com/m/furious_7
28 https://www.rottentomatoes.com/m/furious_7
 Movie:/m/furious_7
Movie info:After defeating international terrorist Owen Shaw, Dominic Toretto (Vin Diesel), Brian O'Conner (Paul Walker) and th
e rest of the crew have separated to return to more normal lives. However, Deckard Shaw (Jason Statham), Owen's older brother,
is thirsty for revenge. A slick government agent offers to help Dom and company take care of Shaw in exchange for their help in
rescuing a kidnapped computer hacker who has developed a powerful surveillance program.

Practical 8

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

Theory:

What is a Sentiment Analysis?

Sentiment analysis (or opinion mining) is a natural language processing (NLP) technique used to determine whether data is positive, negative, or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback and understand customer needs.

Code:

In [ ]:
```
pip install matplotlib pandas nltk textblob
```

In [ ]:
```
import nltk
nltk.download('vader_lexicon')
nltk.download('movie_reviews')
nltk.download('punkt')

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
```

In [ ]:
```
sia = SIA()
sia.polarity_scores("This restaurant was great, but I'm not sure if I'll go there again.")

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

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

In [ ]:
```
from textblob import TextBlob
from textblob import Blobber
from textblob.sentiments import NaiveBayesAnalyzer
```

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

In [ ]:
```
blobber = Blobber(analyzer=NaiveBayesAnalyzer())
```

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

In [ ]:
```
import pandas as pd
pd.set_option("display.max_colwidth", 200)
```

In [ ]:
```
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
```

In [ ]:
```
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'],
    })
```

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

Output:

In [1]:
```
import nltk
nltk.download('vader_lexicon')
nltk.download('movie_reviews')
nltk.download('punkt')

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
```

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

In [2]:
```
sia = SIA()
sia.polarity_scores("This restaurant was great, but I'm not sure if I'll go there again.")

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

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

Out[2]:
```
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
```

In [4]:
```
blob = TextBlob("This restaurant was great, but I'm not sure if I'll go there again.")
blob.sentiment
```

Out[4]:
```
Sentiment(polarity=0.275, subjectivity=0.8194444444444444)
```

In [6]:
```
blob = blobber("This restaurant was great, but I'm not sure if I'll go there again.")
blob.sentiment
```

Out[6]:
```
Sentiment(classification='pos', p_pos=0.5879425317005774, p_neg=0.41205746829942275)
```

In [8]:
```
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
```

Out[8]:

	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

In [10]:
```
scores = df.content.apply(get_scores)
scores.style.background_gradient(cmap='RdYlGn', axis=None, low=0.4, high=0.4)
```

Out[10]:

	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