Practical 1

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

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 the 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.

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

import requests # requests module allows to send HTTP request. import csv

from bs4 import BeautifulSoup import mysql.connector as mysql

req = requests.get( "https://[www.flipkart.com/search?q=mobiles&sid=tyy%2C4io&as=on&as-](http://www.flipkart.com/search?q=mobiles&sid=tyy%2C4io&as=on&as-)

show=on&otracker=AS\_QueryStore\_OrganicAutoSuggest\_1\_2\_na\_na\_na&otracker1=AS\_QueryStore\_O rganicAutoSuggest\_1\_2\_na\_na\_na&as-pos=1&as- type=RECENT&suggestionId=mobiles%7CMobiles&requestId=d880cadf-96c3-472d-b134- 4fc05ac82697&as-backfill=on")

soup = BeautifulSoup(req.content, "html.parser") # print(soup.prettify())

res = soup.head print(res.text)

all\_products = []

products = soup.findAll("div", {"class": "\_3pLy-c row"}) #print(products)

mydb = mysql.connect( host="localhost", user="root", password="root", database="flipkartdb )

mycursor = mydb.cursor()

for product in products:

mname = product.select("div > div.\_4rR01T")[0].text.strip() print(mname)

mprice = product.select("div > div.\_30jeq3")[0].text.strip()

x = mprice.split("₹") print(x[1]) print(mprice)

sql = "INSERT INTO products (Mobile\_Name, Price) VALUES (%s, %s)" val = (mname, x[1])

mycursor.execute(sql, val) mydb.commit()

all\_products.append({ "Name": mname, "Price": mprice

})

print("Record Inserted Successfully...") mydb.close()

print(all\_products)

keys = all\_products[0].keys() print(keys)

with open('flipkart.csv', 'w', newline="", encoding="utf-8") as output\_file: dict\_writer = csv.DictWriter(output\_file, keys)

dict\_writer.writeheader() dict\_writer.writerows(all\_products)

Practical 2

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

import tweepy import pandas as pd

api\_key = 'api\_key' api\_key\_secret = 'api\_key\_secret'

access\_token = 'access\_token' access\_token\_secret = 'access\_token\_secret'

auth = tweepy.OAuthHandler(api\_key, api\_key\_secret) auth.set\_access\_token(access\_token, access\_token\_secret)

api = tweepy.API(auth)

public\_tweets = api.home\_timeline()

# print(public\_tweets) # print all homepage tweets

# print(public\_tweets[0].text) # print first tweet only

# print(public\_tweets[0].created\_at) # print timing of tweets

# print(public\_tweets[0].user.name) # print username

data = []

col = ['Time', 'User', 'Tweet']

for tweet in public\_tweets:

print(tweet.text) # print all tweets print(tweet.created\_at) # print timing for tweets print(tweet.user.name) # print user name

data.append([tweet.created\_at, tweet.user.name, tweet.text])

print(data)

df = pd.DataFrame(data, columns=col) # print(df)

df.to\_csv('tweets.csv', index=False)

Practical 3

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

import networkx as nx import random import numpy as np

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

def nodes\_sorted(g, points):

t = np.array(points) t

= np.argsort(-t) return t

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

g = nx.DiGraph()

N = 4

g.add\_nodes\_from(range(N)

)

g = add\_edges(g, 0.4) points = random\_Walk(g)

sorted\_by\_points = nodes\_sorted(g, points) print("PageRank using Random Walk Method") print(sorted\_by\_points)

PageRank using Random Walk Method [2 3 0 1]

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 inbuilt pagerank method 2, 3, 0, 1,

Practical 4

AIM: Apriori Algorithm implementation in case study

import pandas as pd

from apyori import apriori

store\_data = pd.read\_csv('Dataset.csv') print(store\_data.shape)

col = ['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'] records = []

for i in range(len(store\_data)): records.append([str(store\_data.values[i, j]) for j in range(0, 4)])

print(records)

rules = apriori(transactions=records, min\_support=0.03, min\_confidence=0.2, min\_lift=3, min\_length=2) results = list(rules)

print(results)

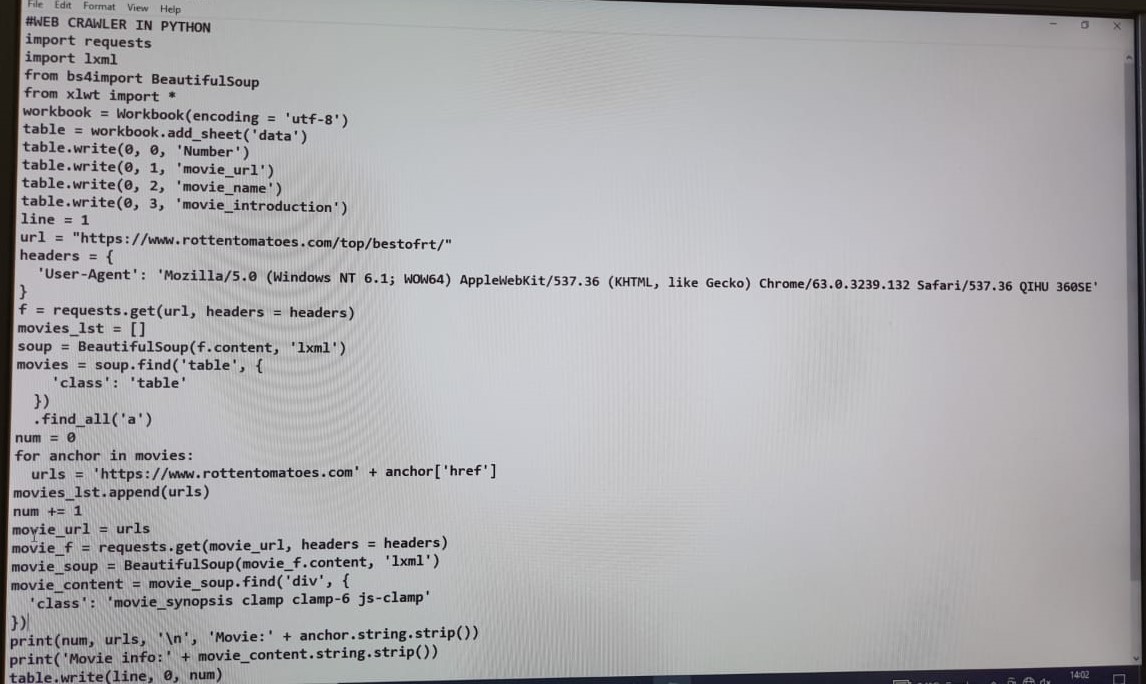
def apriori(results):

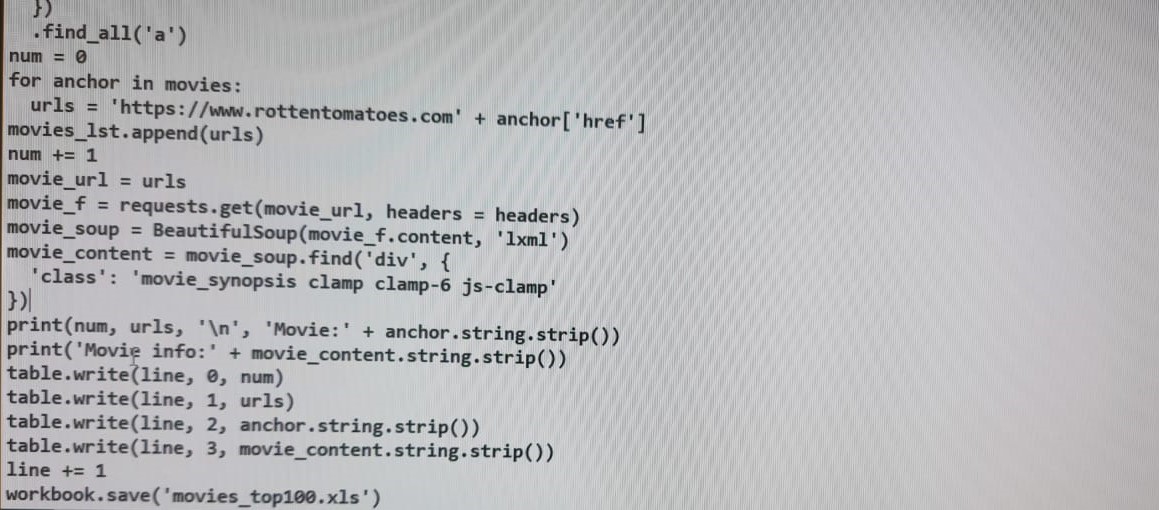
lhs = [tuple(result[2][0][0])[0] for result in results] rhs = [tuple(result[2][0][1])[0] for result in results] supports = [result[1] for result in results] confidences = [result[2][0][2] for result in results] lifts = [result[2][0][3] for result in results]

return list(zip(lhs, rhs, supports, confidences, lifts))

Practical 5

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





Practical 6

AIM:Develop a focused crawler for local search

import requests import lxml

from bs4 import BeautifulSoup

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

url = "https://[www.rottentomatoes.com/top/bestofrt/](http://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)

f = requests.get(url, headers = headers) #source page "rottontomatoes site" soup = BeautifulSoup(f.content,'html') #parse in normal html

movies = soup.find\_all('div',{'class':'discovery-

tiles wrap'})[1].find\_all\_next('a') #take out all movies links available in the page

movies\_lst = [] num = 0

for x in range(len(movies)):

try:

url = 'https://www.rottentomatoes.com' + movies[x]["href"]

movie\_f = requests.get(url, headers = headers) # visit every movie link and access there page movies\_lst.append(url)

num += 1

movie\_page = BeautifulSoup(movie\_f.content,'html')

title = movie\_page.find('h1', {'class': 'scoreboard title'}) # take out movie title movie\_content = movie\_page.find('div', {'class': 'movie\_synopsis'}) # take out movie content print(f"{num} {url}

\nMovie: \nMovie: {title.text}\ninfo: {movie\_content.getText()}") # now print the 1. url of movie, title, content

except: continue

1. <https://www.rottentomatoes.com/m/the_lost_city>

Movie:

Movie: The Lost

City info: Reclusive author Loretta Sage writes about exotic places in her popular adventure novels that feature a handsome cover model named Alan. While on tour promoting her new book with Alan, Loretta

Practical 7

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

pip install matplotlib pandas nltk textblob

import nltk nltk.download('vader\_lexicon') nltk.download('movie\_reviews'

) nltk.download('punkt')

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA

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)

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

{'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

blobber = Blobber(analyzer=NaiveBayesAnalyzer())

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

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

**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)

**content**

-

0.000000

0.613050

0.241100

0.394659

0.250000

- 0.400000

0.941300

-

-0.214151

0.800000

-

-

-0.087933

0.500000

-

**nltk**

**textblob\_bayes**

**textblob**

**0** I love love love love this kitten 0.957100

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

0.040076

0.000000

0.476700

0.350000

-0.574270

-

were broken

1. Trashy television shows are some of my favorites 0.421500
2. I'm seeing a Kubrick film tomorrow, I hear not so

great things

about it.

0.800000

0.717875

-

0.629600