PRACTICAL JOURNAL

in

WEB DATA ANALYTICS

Submitted by

KSMSCIT005 HITESH BHANUSHALI

for the award of the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

PART – II

DEPARTMENT OF INFORMATION TECHNOLOGY
KISHINCHAND CHELLARAM COLLEGE
(Affiliated to University of HSNCU)
MUMBAI,400020
MAHARASHTRA
2024-25

SUBJECT CODE – BSIT606B

WEB DATA ANALYTICS

KISHINCHAND CHELLARAM COLLEGE

CHURCHGATE, MUMBAI – 400 020.

DEPARTMENT OF INFORMATION TECHNOLOGY

M.SC.I.T PART- II

CERTIFICATE

This is to certify that the Practical conducted by Mr. HITESH BHANUSHALI for M.Sc. (IT) Part- II Semester- III, Seat No: KSMSCIT005 at Kishinchand Chellaram College in partial fulfillment for the MASTERS OF SCIENCE (INFORMATION TECHNOLOGY). Degree Examination for Semester III has been periodically examined and signed, and the course of term work has been satisfactorily carried out for the year 2024 - 2025. This Practical journal had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Signature	Signature	Signature
Lecturer-In-Charge	External Examiner	Course Coordination
Guided By	Examined By	Certified By

College Stamp

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20
M.Sc (I.T.) Part-1 Semester II

INDEX

Sr. No.	TITLE	Date	Page No.	Signature
	PRACTICALS			
1	Perform Spam Classifier	15/06/24	1-2	
2	Apriori Algorithm implementation in case study.	07/07/24	3-4	
3	Develop a basic crawler for the web search for user -defined keywords.	07/07/24	5	
4	Sentiment analysis for reviews by customers and visualize the same.	24/08/24	6-7	
5	Link Analysis and PageRank a) Implement the PageRank algorithm to rank web pages based on link analysis. b) Apply the PageRank algorithm to a small web graph and analyze the results.	24/08/24	8	
6	Scrape an online Social Media Site for Data. Use python to scrape information from twitter.	24/08/24	9-10	
7	Web Crawling and Indexing a) Develop a web crawler to fetch and index web pages. b) Handle challenges such as robots.txt, dynamic content, and crawling delays.	14/09/24		
8	Develop a focused crawler for local search.	14/09/24		
	ASSIGNMENTS			
1	Assignment 1	24/07/24		
2	Assignment 2	04/08/24		
3	Assignment 3	28/09/24		

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc (I.T.) Part-2 Semester III

PRACTICAL - 1

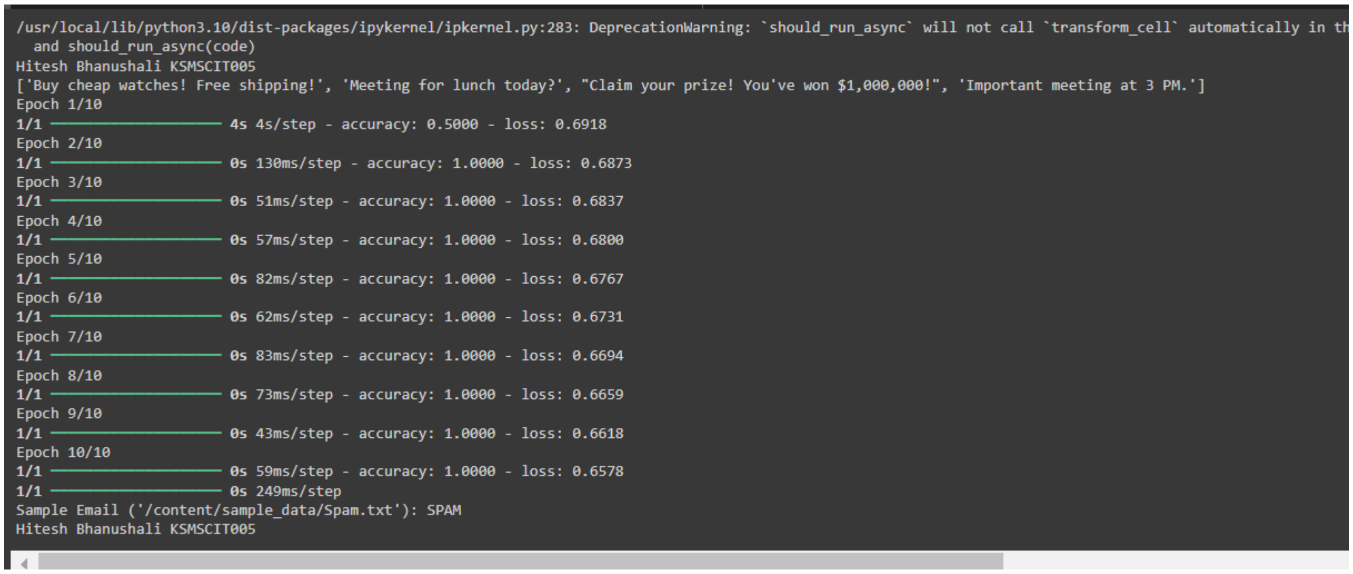
Aim: Perform Spam Classifier

Code:

```
print("Hitesh Bhanushali KSMSCIT005")
import tensorflow as tf
import numpy as np # Added import for NumPy
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
# Step 1: Prepare a dataset of labeled emails (spam and non-spam)
emails = [ "Buy cheap watches! Free shipping!",
        "Meeting for lunch today?",
         "Claim your prize! You've won $1,000,000!",
         "Important meeting at 3 PM.",
         ]
labels = [1, 0, 1, 0]
print(emails)
# Step 2: Tokenize and pad the email text data
max_words = 1000
max_len = 50
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(emails)
sequences = tokenizer.texts_to_sequences(emails)
X_padded = pad_sequences(sequences, maxlen=max_len, padding="post", truncating="post")
# Step 3: Define the neural network model
model = tf.keras.Sequential([
   tf.keras.layers.Embedding(input_dim=max_words, output_dim=16, input_length=max_len),
   tf.keras.layers.Flatten(), tf.keras.layers.Dense(16, activation='relu'),
   tf.keras.layers.Dense(1, activation='sigmoid')
   ])
# Compile the model
```

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
training_data = np.array(X_padded)
training_labels = np.array(labels)
model.fit(training_data, training_labels, epochs=10)
file_path = "Spam.txt"
with open(file_path, "r", encoding="utf-8") as file:
  sample_email_text = file.read()
# Tokenize and pad the sample email text
sequences_sample = tokenizer.texts_to_sequences([sample_email_text])
sample_email_padded = pad_sequences(sequences_sample, maxlen=max_len, padding="post", truncating="post")
# Use the trained model to make predictions
prediction = model.predict(sample_email_padded)
# Set a classification threshold (e.g., 0.5)
threshold = 0.5
#Classify the sample email based on the threshold
if prediction > threshold: print(f"Sample Email ('{file_path}'): SPAM")
else: print(f"Sample Email ('{file_path}'): NOT SPAM")
print("Hitesh Bhanushali KSMSCIT005")
```

Output:



```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in th
  and should_run_async(code)
Hitesh Bhanushali KSMSCIT005
['Buy cheap watches! Free shipping!', 'Meeting for lunch today?', "Claim your prize! You've won $1,000,000!", 'Important meeting at 3 PM.']
Epoch 1/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - accuracy: 0.5000 - loss: 0.6918
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 130ms/step - accuracy: 1.0000 - loss: 0.6873
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 1.0000 - loss: 0.6837
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 1.0000 - loss: 0.6800
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 1.0000 - loss: 0.6767
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 1.0000 - loss: 0.6731
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step - accuracy: 1.0000 - loss: 0.6694
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step - accuracy: 1.0000 - loss: 0.6659
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 1.0000 - loss: 0.6618
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 1.0000 - loss: 0.6578
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 249ms/step
Sample Email ('/content/sample_data/Spam.txt'): SPAM
Hitesh Bhanushali KSMSCIT005
```

PRACTICAL - 2

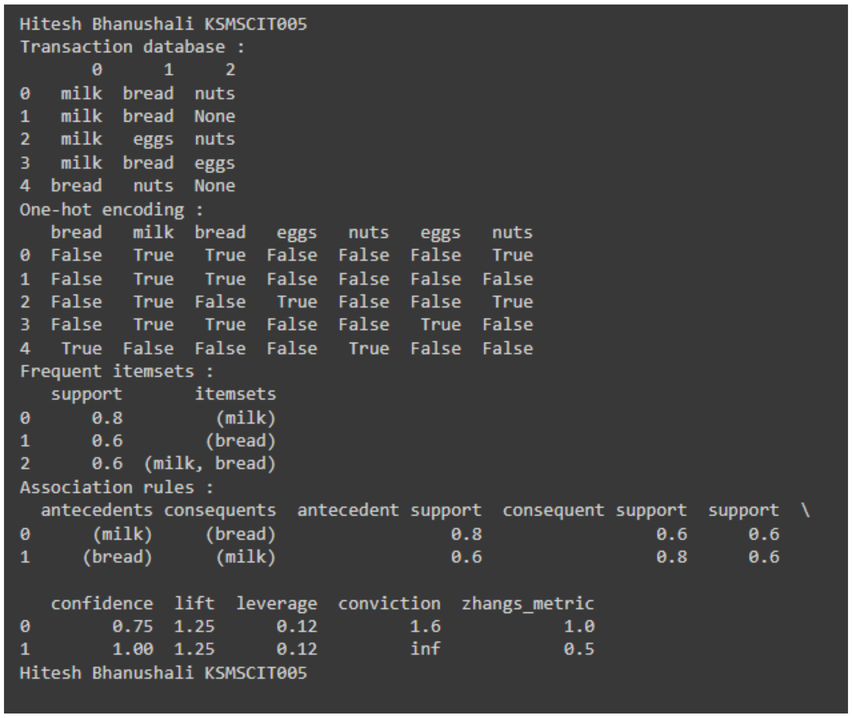
Aim: Apriori Algorithm implementation in case study.

Code:

```
print("Hitesh Bhanushali KSMSCIT005")
#required libraries for association rules
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import pandas as pd
import numpy as np
dataset = [
   ['milk','bread','nuts'],
   ['milk','bread'],
   ['milk','eggs','nuts'],
   ['milk','bread','eggs'],
   ['bread','nuts']]
df = pd.DataFrame(dataset)
print("Transaction database :")
print(df)
#Perform one-hot encoding (Convert items to column)
df_encoding = pd.get_dummies(df, prefix ='',prefix_sep='')
print("One-hot encoding :")
print(df_encoding)
# Find frequent itemsets using Apriori algo
frequent_itemsets = apriori(df_encoding, min_support=0.5, use_colnames=True)
print("Frequent itemsets :")
print(frequent_itemsets)
```

```
# Generate association rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print("Association rules :")
print(rules)
print("Hitesh Bhanushali KSMSCIT005")
```

Output:



```
Hitesh Bhanushali KSMSCIT005
Transaction database :
       0      1     2
0   milk  bread  nuts
1   milk  bread  None
2   milk   eggs  nuts
3   milk  bread  eggs
4  bread   nuts  None
One-hot encoding :
   bread   milk  bread   eggs   nuts   eggs   nuts
0  False   True   True  False  False  False   True
1  False   True   True  False  False  False  False
2  False   True  False   True  False  False   True
3  False   True   True  False  False   True  False
4   True  False  False  False   True  False  False
Frequent itemsets :
   support       itemsets
0      0.8         (milk)
1      0.6        (bread)
2      0.6  (milk, bread)
Association rules :
  antecedents consequents  antecedent support  consequent support  support  \
0      (milk)     (bread)                 0.8                 0.6      0.6
1     (bread)      (milk)                 0.6                 0.8      0.6

   confidence  lift  leverage  conviction  zhangs_metric
0        0.75  1.25      0.12         1.6            1.0
1        1.00  1.25      0.12         inf            0.5
Hitesh Bhanushali KSMSCIT005
```

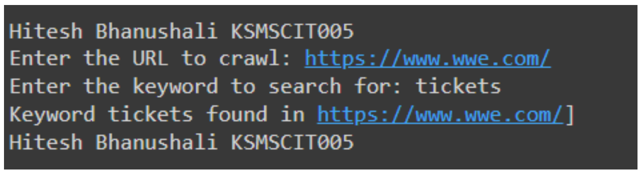
PRACTICAL - 3

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

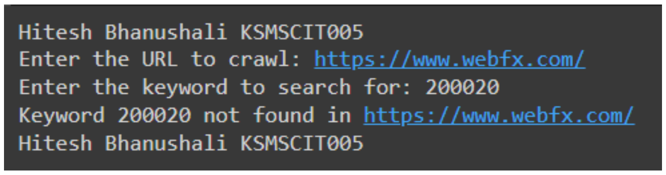
Code:

```
import requests
from bs4 import BeautifulSoup
print("Hitesh Bhanushali KSMSCIT005")
def check_word_in_webpage (url, word):
  response = requests.get(url)
  if response.status_code == 200:
    soup = BeautifulSoup (response.content, 'html.parser')
    text_content = soup.get_text()
    if word.lower() in text_content.lower():
      print(f"The word '{word}' is present in the webpage.")
    else:
        print (f"The word '{word}' is not present in the webpage.")
  else:
    print("Failed to retrieve webpage.")
url= input("Enter the url you want to Scrap: ")
word_to_check = input("Enter the text you want to know which is present or not: ")
check_word_in_webpage (url, word_to_check)
print("Hitesh Bhanushali KSMSCIT005")
```

Output:



```
Hitesh Bhanushali KSMSCIT005
Enter the URL to crawl: https://www.wwe.com/
Enter the keyword to search for: tickets
Keyword tickets found in https://www.wwe.com/]
Hitesh Bhanushali KSMSCIT005
```



```
Hitesh Bhanushali KSMSCIT005
Enter the URL to crawl: https://www.webfx.com/
Enter the keyword to search for: 200020
Keyword 200020 not found in https://www.webfx.com/
Hitesh Bhanushali KSMSCIT005
```

PRACTICAL - 4

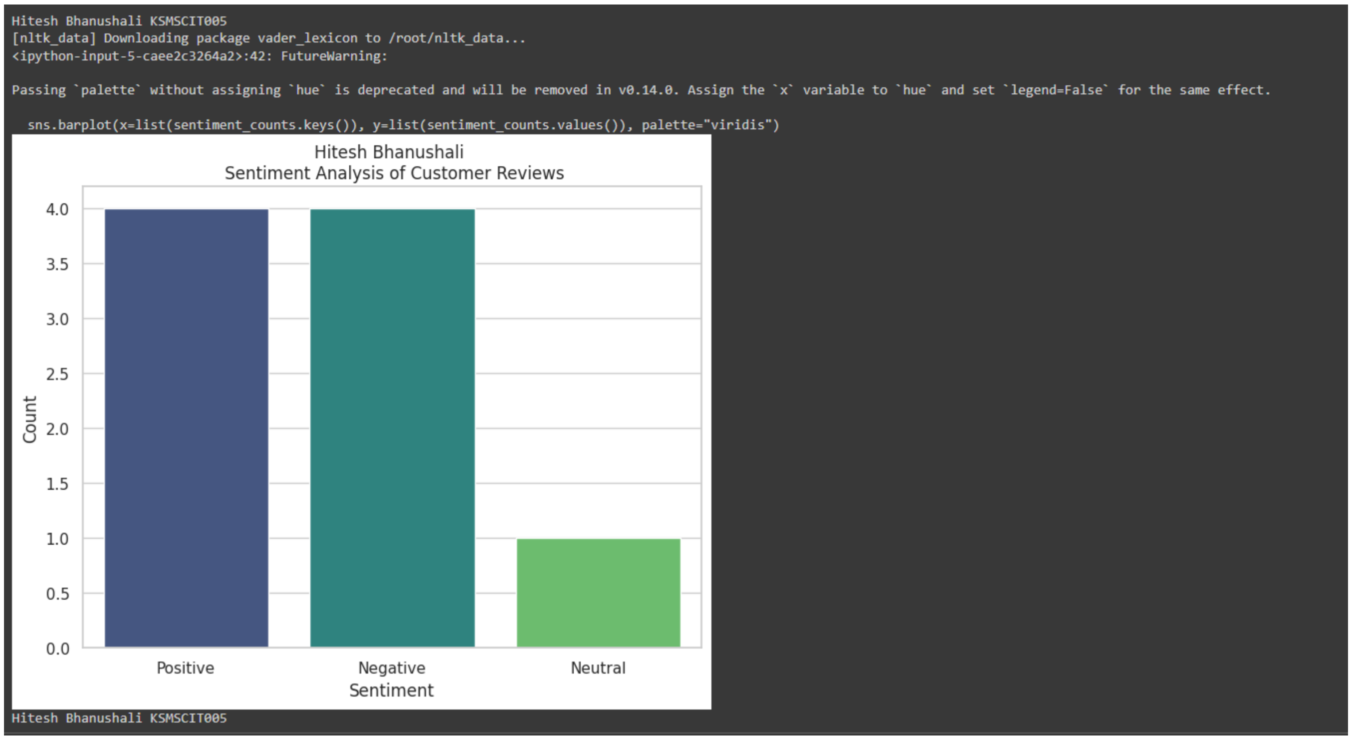
Aim: Sentiment analysis for reviews by customers and visualize the same.

Code:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import matplotlib.pyplot as plt
import seaborn as sns
# Step 1: Download VADER lexicon if not already done
print("Hitesh Bhanushali KSMSCIT005")
nltk.download('vader_lexicon')
# Step 2: Initialize the VADER sentiment intensity analyzer
sia= SentimentIntensityAnalyzer()
#Step 3: List of customer reviews
reviews = [
"The product quality is amazing, I'm very satisfied!",
"Terrible service, I will never buy from here again.",
"Decent product, but shipping was too slow.",
"Absolutely love it! Will recommend to everyone.",
"Not worth the money, very disappointing.",
"Great experience overall but could improve the packaging"
"Mediocre, not what I expected.",
"Excellent value for the price, highly recommended.",
"Worst purchase I've made this year.",
"It's okay, nothing special."]
# Step 4: Analyze sentiment for each review
```

```
sentiments = []
for review in reviews:
  sentiment_score = sia.polarity_scores (review)
  compound_score = sentiment_score['compound']
  if compound_score >= 0.05:
    sentiments.append('Positive')
  elif compound_score <= -0.05:
    sentiments.append('Negative')
  else:
    sentiments.append('Neutral')
# Step 5: Count the occurrences of each sentiment
sentiment_counts = {
'Positive': sentiments.count('Positive'), 'Negative': sentiments.count('Negative'),
'Neutral': sentiments.count('Neutral')   }
#Step 6: Visualize
sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))
sns.barplot(x=list(sentiment_counts.keys()), y=list(sentiment_counts.values()), palette="viridis")
plt.title('Hitesh Bhanushali \n Sentiment Analysis of Customer Reviews')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
print("Hitesh Bhanushali KSMSCIT005")
```

Output:



```
Hitesh Bhanushali KSMSCIT005
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
<ipython-input-5-caee2c3264a2>:42: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(sentiment_counts.keys()), y=list(sentiment_counts.values()), palette="viridis")
```

```
Hitesh Bhanushali KSMSCIT005
```

PRACTICAL - 5

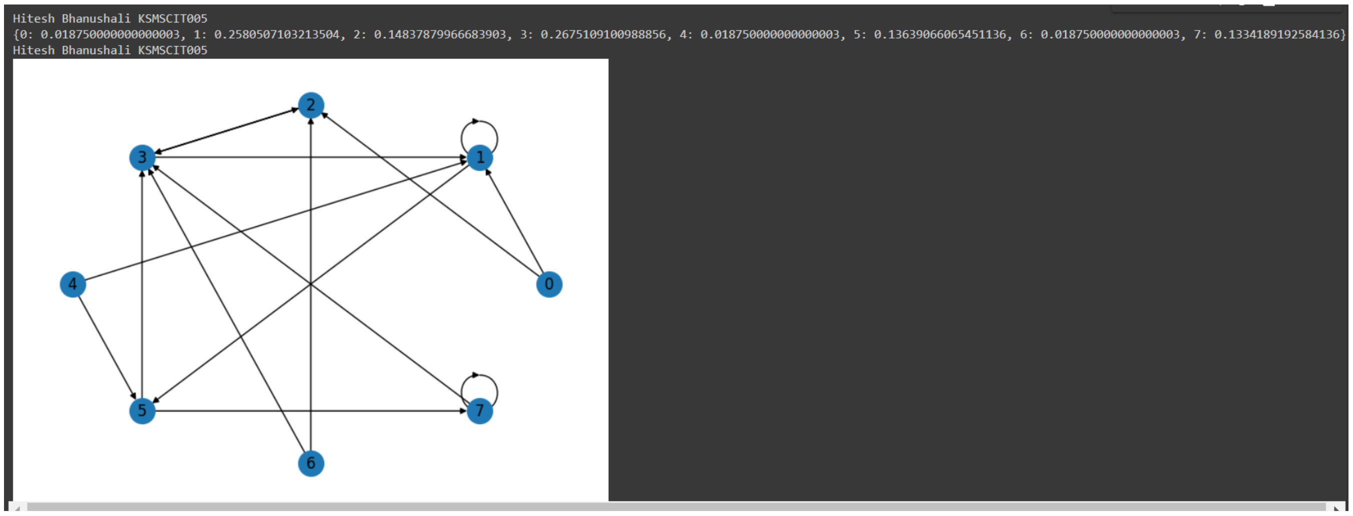
Aim: Link Analysis and PageRank

a) **Implement the PageRank algorithm to rank web pages based on link analysis.**
b) **Apply the PageRank algorithm to a small web graph and analyze the results.**

Code:

```
import networkx as nx

print("Hitesh Bhanushali KSMSCIT005")

G = nx.random_k_out_graph(n=8, k=2, alpha=0.75)

def draw_graph(G):

   nx.draw_circular(G, node_size=400, with_labels=True)

draw_graph(G)

ranks_pr = nx.pagerank(G)

print(ranks_pr)

print("Hitesh Bhanushali KSMSCIT005")
```

Output:



```
Hitesh Bhanushali KSMSCIT005
{0: 0.018750000000000003, 1: 0.2580507103213504, 2: 0.14837879966683903, 3: 0.2675109100988856, 4: 0.018750000000000003, 5: 0.13639066065451136, 6: 0.018750000000000003, 7: 0.1334189192584136}
Hitesh Bhanushali KSMSCIT005
```

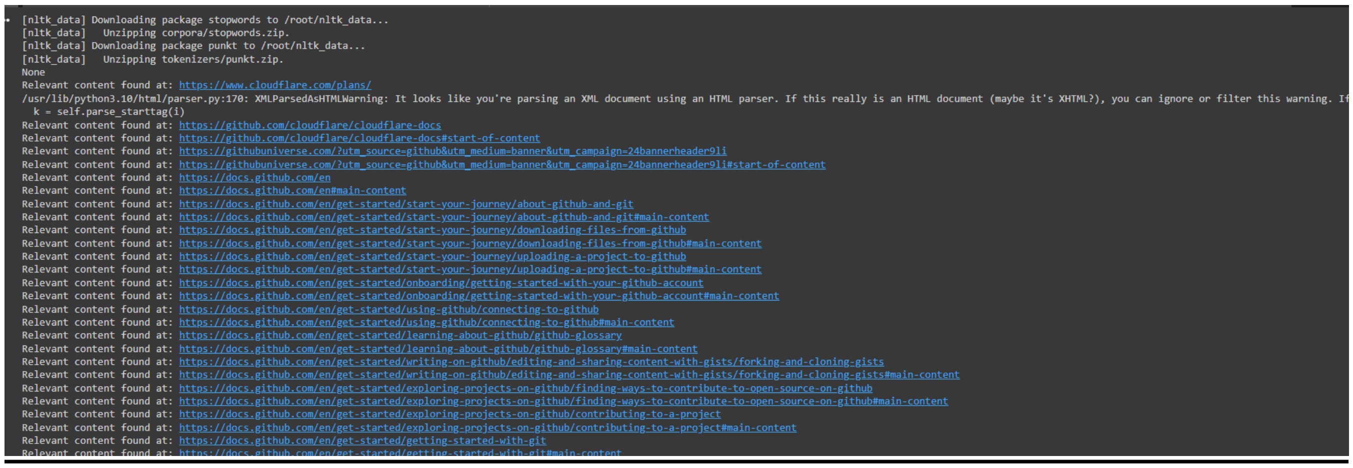
PRACTICAL - 6

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

Code:

```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
nltk.download('stopwords')
nltk.download('punkt')
import re
# Seed URLs
seed_urls = [ 'https://www.cloudflare.com/'
        'https://europa.eu/',
        'https://t.me/',
        'https://vk.com/',
        'https://brandbucket.com/']
# Keywords to focus on
keywords = ['restaurant', 'food', 'local']
# Stop words (to filter out common words)
stop_words = set(stopwords.words('english'))
# Visited URLS
visited = set()
def is_relevant(content, keywords):
  #Check if the content is relevant based on the keywords.
  words = word_tokenize(content.lower())
  words = [w for w in words if w.isalnum() and w not in stop_words]
  return any(keyword in words for keyword in keywords)
def crawl(url):
#Crawl a single webpage.
  try:
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    text = soup.get_text()
    # Check if the content is relevant
    if is_relevant(text, keywords):
      print(f"Relevant content found at: {url}")
      # Here you could save the content to a file or database
    # Extract links and follow them
    for link in soup.find_all('a', href=True):
      new_url = urljoin(url, link['href'])
      if new_url not in visited and re.match(r'^https?://', new_url):
        visited.add(new_url)
        crawl(new_url)
  except requests.exceptions.RequestException as e:
      print(f"Error crawling {url}: {e}")
# Start crawling from the seed URLS
for url in seed_urls:
  if url not in visited:
    visited.add(url)
    print(crawl(url))
```

Output:



```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
None
Relevant content found at: https://www.cloudflare.com/plans/
/usr/lib/python3.10/html/parser.py:170: XMLParsedAsHTMLWarning: It looks like you're parsing an XML document using an HTML parser. If this really is an HTML document (maybe it's XHTML?), you can ignore or filter this warning. If
  k = self.parse_starttag(i)
Relevant content found at: https://github.com/cloudflare/cloudflare-docs
Relevant content found at: https://github.com/cloudflare/cloudflare-docs#start-of-content
Relevant content found at: https://githubuniverse.com/?utm_source=github&utm_medium=banner&utm_campaign=24bannerheader9li
Relevant content found at: https://githubuniverse.com/?utm_source=github&utm_medium=banner&utm_campaign=24bannerheader9li#start-of-content
Relevant content found at: https://docs.github.com/en
Relevant content found at: https://docs.github.com/en#main-content
Relevant content found at: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git
Relevant content found at: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git#main-content
Relevant content found at: https://docs.github.com/en/get-started/start-your-journey/downloading-files-from-github
Relevant content found at: https://docs.github.com/en/get-started/start-your-journey/downloading-files-from-github#main-content
Relevant content found at: https://docs.github.com/en/get-started/start-your-journey/uploading-a-project-to-github
Relevant content found at: https://docs.github.com/en/get-started/start-your-journey/uploading-a-project-to-github#main-content
Relevant content found at: https://docs.github.com/en/get-started/onboarding/getting-started-with-your-github-account
Relevant content found at: https://docs.github.com/en/get-started/onboarding/getting-started-with-your-github-account#main-content
Relevant content found at: https://docs.github.com/en/get-started/using-github/connecting-to-github
Relevant content found at: https://docs.github.com/en/get-started/using-github/connecting-to-github#main-content
Relevant content found at: https://docs.github.com/en/get-started/learning-about-github/github-glossary
Relevant content found at: https://docs.github.com/en/get-started/learning-about-github/github-glossary#main-content
Relevant content found at: https://docs.github.com/en/get-started/writing-on-github/editing-and-sharing-content-with-gists/forking-and-cloning-gists
Relevant content found at: https://docs.github.com/en/get-started/writing-on-github/editing-and-sharing-content-with-gists/forking-and-cloning-gists#main-content
Relevant content found at: https://docs.github.com/en/get-started/exploring-projects-on-github/finding-ways-to-contribute-to-open-source-on-github
Relevant content found at: https://docs.github.com/en/get-started/exploring-projects-on-github/finding-ways-to-contribute-to-open-source-on-github#main-content
Relevant content found at: https://docs.github.com/en/get-started/exploring-projects-on-github/contributing-to-a-project
Relevant content found at: https://docs.github.com/en/get-started/exploring-projects-on-github/contributing-to-a-project#main-content
Relevant content found at: https://docs.github.com/en/get-started/getting-started-with-git
Relevant content found at: https://docs.github.com/en/get-started/getting-started-with-git#main-content
```