Bahria University,

Karachi Campus

## LAB EXPERIMENT NO.

11

## LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Develop a Python script to scrape textual data from websites and perform sentiment analysis on this data. This could involve analyzing customer reviews, forum discussions, or news articles. |
| **2** | Use publicly available APIs (like Twitter API, OpenWeatherMap, or others) to fetch data and analyze specific trends or patterns. |
| **3** | Perform data mining from a collection of local files (like logs, text files, or CSVs) to extract meaningful information without using a database. |

Submitted On:

15 may 2024

(Date: DD/MM/YY)

**TASK # 1:** Develop a Python script to scrape textual data from websites and perform sentiment analysis on this data. This could involve analyzing customer reviews, forum discussions, or news articles.

import streamlit as st

from bs4 import BeautifulSoup

import requests

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

sentiment\_analyzer = SentimentIntensityAnalyzer()

def scrape\_website(url):

    try:

        response = requests.get(url)

        response.raise\_for\_status()

        soup = BeautifulSoup(response.content, 'html.parser')

        return soup.get\_text()

    except requests.exceptions.RequestException as e:

        st.error(f"Error fetching the URL: {e}")

        return ""

def analyze\_sentiment(text):

    return sentiment\_analyzer.polarity\_scores(text)

def main():

    st.title("Website Text Scraper and Sentiment Analyzer")

    url = st.text\_input("Enter the URL of the website to scrape")

    if st.button("Scrape and Analyze"):

        if url:

            text = scrape\_website(url)

            if text:

                st.write("### Scraped Text")

                st.write(text[:1000])  # Display first 1000 characters of the scraped text

                sentiment = analyze\_sentiment(text)

                st.write("### Sentiment Analysis")

                st.write(f"\*\*Positive:\*\* {sentiment['pos'] \* 100:.2f}%")

                st.write(f"\*\*Neutral:\*\* {sentiment['neu'] \* 100:.2f}%")

                st.write(f"\*\*Negative:\*\* {sentiment['neg'] \* 100:.2f}%")

                st.write(f"\*\*Overall Sentiment:\*\* {'Positive' if sentiment['compound'] >= 0.05 else 'Negative' if sentiment['compound'] <= -0.05 else 'Neutral'}")

        else:

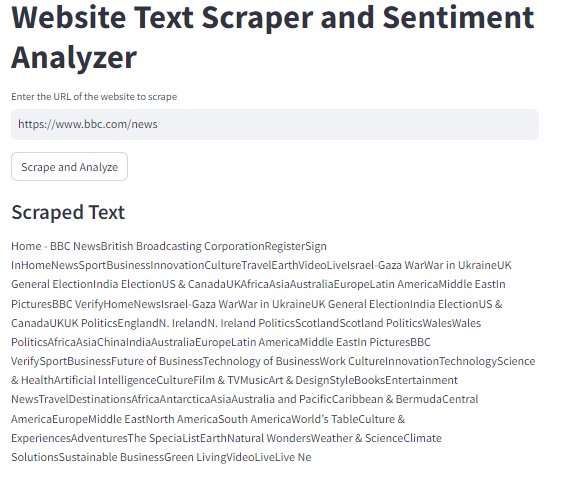
            st.error("Please enter a valid URL")

if \_\_name\_\_ == "\_\_main\_\_":

main()

A screenshot of a graph

Description automatically generated



**TASK # 2:** Use publicly available APIs (like Twitter API, OpenWeatherMap, or others) to fetch data and analyze specific trends or patterns.

import streamlit as st

import requests

import matplotlib.pyplot as plt

from datetime import datetime

def fetch\_current\_weather(api\_key, city):

    url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api\_key}&units=metric'

    response = requests.get(url)

    data = response.json()

    return data

def fetch\_historical\_weather(api\_key, city):

    historical\_url = f'http://api.openweathermap.org/data/2.5/forecast?q={city}&appid={api\_key}&units=metric'

    historical\_response = requests.get(historical\_url)

    historical\_data = historical\_response.json()

    return historical\_data

def plot\_temperature\_trend(dates, temperatures):

    fig, ax = plt.subplots(figsize=(10, 5))

    ax.plot(dates, temperatures, marker='o', linestyle='-')

    ax.set\_title('Temperature Trend for Next 5 Days')

    ax.set\_xlabel('Date')

    ax.set\_ylabel('Temperature (°C)')

    ax.tick\_params(axis='x', rotation=45)

    ax.grid(True)

    st.pyplot(fig)

def main():

    st.title('Weather Analysis App')

    api\_key = st.text\_input('Enter your OpenWeatherMap API key:')

    city = st.text\_input('Enter city name:')

    if api\_key and city:

        current\_weather\_data = fetch\_current\_weather(api\_key, city)

        if 'main' in current\_weather\_data:

            temperature = current\_weather\_data['main']['temp']

            humidity = current\_weather\_data['main']['humidity']

            pressure = current\_weather\_data['main']['pressure']

            wind\_speed = current\_weather\_data['wind']['speed']

            st.write(f'Current temperature in {city}: {temperature}°C')

            st.write(f'Humidity: {humidity}%')

            st.write(f'Pressure: {pressure} hPa')

            st.write(f'Wind Speed: {wind\_speed} m/s')

        historical\_data = fetch\_historical\_weather(api\_key, city)

        if 'list' in historical\_data:

            dates = []

            temperatures = []

            for forecast in historical\_data['list']:

                timestamp = forecast['dt']

                date = datetime.utcfromtimestamp(timestamp).strftime('%Y-%m-%d')

                dates.append(date)

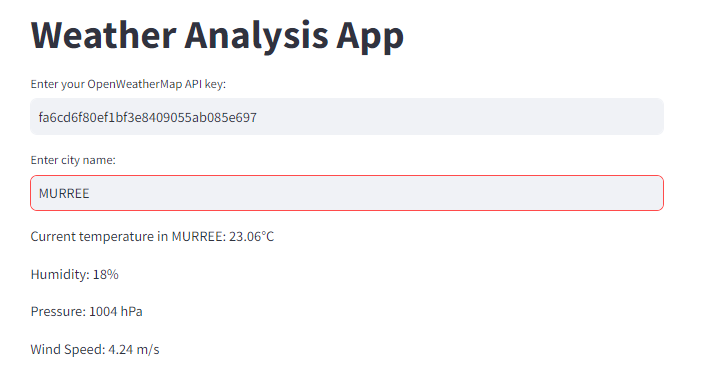
                temperatures.append(forecast['main']['temp'])

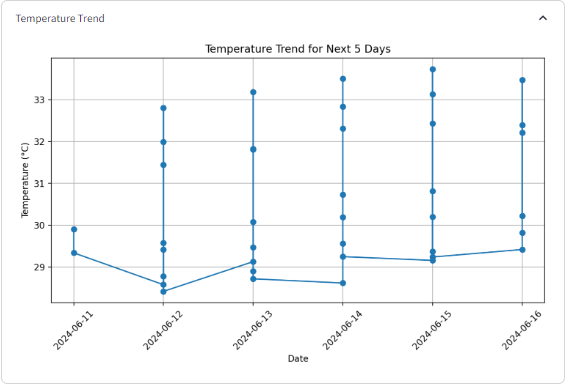
            with st.expander("Temperature Trend"):

                plot\_temperature\_trend(dates, temperatures)

if \_\_name\_\_ == "\_\_main\_\_":

    main()





**TASK#3:** Perform data mining from a collection of local files (like logs, text files, or CSVs) to extract meaningful information without using a database.

import streamlit as st

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

def load\_data(file):

    if file.name.endswith('.csv'):

        data = pd.read\_csv(file)

    elif file.name.endswith('.xlsx'):

        data = pd.read\_excel(file)

    else:

        st.error('Unsupported file format. Please upload a CSV or Excel file.')

        return None

    return data

def explore\_data(data):

    st.write("### Data Overview")

    st.write(data.head())

    st.write("Shape of the data:", data.shape)

    st.write("Summary statistics:", data.describe())

    st.write("### Data Visualization")

    st.write("#### Pairplot")

    sns.pairplot(data)

    st.pyplot()

    st.write("#### Correlation Heatmap")

    numeric\_data = data.select\_dtypes(include=['float64', 'int64'])

    corr = numeric\_data.corr()

    sns.heatmap(corr, annot=True, cmap='coolwarm')

    st.pyplot()

def main():

    st.title("Data Mining from Files")

    file = st.file\_uploader("Upload a file", type=['csv', 'xlsx'])

    if file is not None:

        data = load\_data(file)

        if data is not None:

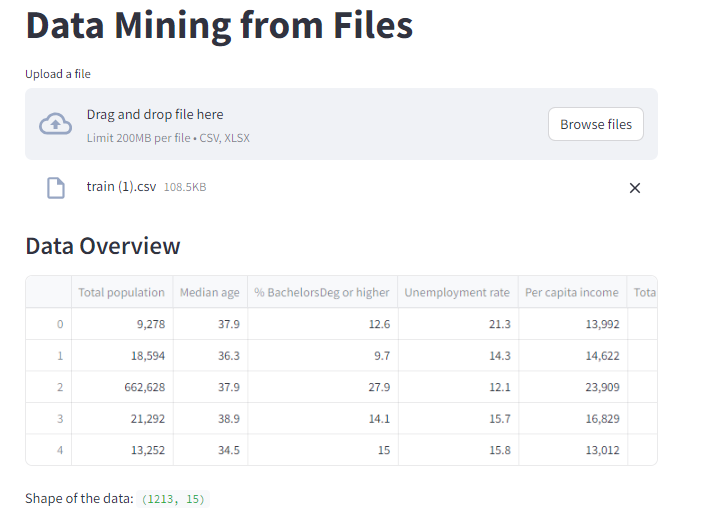
            explore\_data(data)

if \_\_name\_\_ == "\_\_main\_\_":

main()

A screenshot of a graph

Description automatically generated



A grid of blue dots

Description automatically generated

A screenshot of a graph

Description automatically generated