Task2:-

🔧 **Project Overview:-**

Objective: Build a web app that allows users to upload and analyze textual data using NLTK for natural language processing and pandas for data handling. They are provide:-

●Preprocessing tools

●Statistical insights (frequency, collocations, sentiment)

●Visualization tools

●An interactive and dynamic user interface via Streamlit or Flask

📁 **1. Project Structure:-**

```
Text_analytics_app/
│
├── app.py or streamlit_app.py        # Main web app logic (UI + routing)
├── nlp_pipeline.py                   # Text processing and analysis module
├── static/                           # CSS, JS, images (only for Flask)
├── templates/                        # HTML templates (only for Flask)
├── data/
│   └── sample_data.txt               # Example dataset
├── output screenshots/               # Output visualizations (static examples)
└── README.md                         # User guide + setup instructions
```

⚙️ **2. Nlp_pipeline.py: NLP Processing Module**

Functions should include:-

```
Import nltk
Import pandas as pd
From nltk.corpus import stopwords
From nltk import FreqDist, pos_tag, word_tokenize, collocations
From nltk.stem import WordNetLemmatizer
From nltk. Sentiment import SentimentIntensityAnalyzer
```

```
Def preprocess(text: str) -> pd.DataFrame:
    # Tokenize, lowercase, remove stopwords, lemmatize
    # Return DataFrame with tokens, POS tags
    ...

Def compute_freq_dist(tokens: list) -> FreqDist:
    ...

Def compute_collocations(tokens: list) -> list:
    ...

Def compute_sentiment_scores(texts: list) -> pd.DataFrame:
    # Use VADER for sentiment (compound, pos, neg, neu)
    ...
```

📈 **3. Visualizations:-**

Use either matplotlib, seaborn, or plotly to show:

●Top N N-grams (bigrams, trigrams)

●Sentiment trend line (for multiple inputs or sections)

●Optional: POS distribution, word clouds

🖼️ **4. UI Pages (Flask or Streamlit)**

Option 1:- Flask

□/upload route (Data Explorer):

《》 Upload file

《》 Display raw text and token table

□/dashboard route (Analysis Dashboard):

《》 Show word frequency, collocations, sentiment chart

《》 Use Bootstrap for styling

Option 2:-Streamlit (easier, recommended for faster setup)

《》 Two tabs/pages using st.sidebar.radio or st.page:

《》 Data Explorer: File upload, text preview, tokens table

《》 Analysis Dashboard: Frequency charts, sentiment scores, collocations

📁 5. Example Dataset:-

●*Provide a file like movie_reviews.txt or tweets_sample.txt in the data/ folder.*

●Should contain several text entries (one per line or JSON).

📷 6. Output Screenshots:-

●*Screenshot 1: Data upload and preview*

●Screenshot 2: Frequency bar plot

●Screenshot 3: Sentiment trend visualization

📘 7. README / User Guide:-

Content to include:-

●Project description

●Installation: pip install -r requirements.txt

●How to run: python app.py or streamlit run streamlit_app.py

●Screenshots

●Sample output

●Troubleshooting

✅ **Key Libraries:-**

Bash

Pip install nltk pandas matplotlib plotly streamlit flask

Python

Import nltk
Nltk. download('punkt')
Nltk.download('averaged_perceptron_tagger')
Nltk.download('stopwords')
Nltk.download('wordnet')
Nltk.download('vader_lexicon')

☆A starter version of nlp_pipeline.py?

☆Streamlit or Flask version of the app UI?

☆A complete zip package template?

✅ **1. Starter Version of nlp_pipeline.py**

 # nlp_pipeline.py

Import nltk
Import pandas as pd
From nltk.corpus import stopwords, wordnet
From nltk.tokenize import word_tokenize
From nltk.stem import WordNetLemmatizer
From nltk import FreqDist, bigrams, trigrams, pos_tag
From nltk.sentiment import SentimentIntensityAnalyzer

```
Nltk.download('punkt')
Nltk.download('stopwords')
Nltk.download('wordnet')
Nltk.download('averaged_perceptron_tagger')
Nltk.download('vader_lexicon')

Stop_words = set(stopwords.words('english'))
Lemmatizer = WordNetLemmatizer()
Sia = SentimentIntensityAnalyzer()

Def preprocess(text: str) -> pd.DataFrame:
    Tokens = word_tokenize(text.lower())
    Tokens = [t for t in tokens if t.isalpha() and t not in stop_words]
    Lemmas = [lemmatizer.lemmatize(token) for token in tokens]
    Pos_tags = pos_tag(lemmas)
    Return pd.DataFrame(pos_tags, columns=["Token", "POS"])

Def compute_freq_dist(tokens: list) -> pd.DataFrame:
    Freq = FreqDist(tokens)
    Return pd.DataFrame(freq.most_common(20), columns=["Token", "Frequency"])

Def compute_ngrams(tokens: list, n: int = 2) -> pd.DataFrame:
    If n == 2:
        Ngrams_list = list(bigrams(tokens))
    Elif n == 3:
        Ngrams_list = list(trigrams(tokens))
    Else:
        Raise ValueError("Only bigrams and trigrams supported.")
    Freq = FreqDist(ngrams_list)
    Return pd.DataFrame(freq.most_common(20), columns=["N-gram", "Frequency"])

Def compute_sentiment_scores(text: str) -> dict:
    Return sia.polarity_scores(text)
```

✅ 2. Streamlit Version of the Web App

```
# streamlit_app.py

Import streamlit as st
Import pandas as pd
From nlp_pipeline import preprocess, compute_freq_dist, compute_ngrams,
compute_sentiment_scores
```

```
Import matplotlib.pyplot as plt

St.set_page_config(page_title="Text Analytics App", layout="wide")

St.title("🔍 NLP-Powered Text Analytics")

Menu = st.sidebar.radio("Navigation", ["📂 Data Explorer", "📊 Analysis Dashboard"])

Uploaded_file = st.sidebar.file_uploader("Upload a text file (.txt)", type="txt")

If uploaded_file:
    Text = uploaded_file. Read().decode("utf-8")
    Token_df = preprocess(text)
    Tokens = token_df["Token"].tolist()
Else:
    St.warning("Upload a .txt file to begin.")
    St.stop()

If menu == "📂 Data Explorer":
    St.subheader("Raw Text Preview")
    St.text_area("Uploaded Text", value=text[:1000], height=200)

    St.subheader("Token Table (first 50)")
    St.dataframe(token_df.head(50))

Elif menu == "📊 Analysis Dashboard":
    St.subheader("Top Word Frequencies")
    Freq_df = compute_freq_dist(tokens)
    St.bar_chart(freq_df.set_index("Token"))

    St.subheader("Top Bigrams")
    Bigram_df = compute_ngrams(tokens, n=2)
    St.dataframe(bigram_df)

    St.subheader("Sentiment Analysis")
    Sentiment = compute_sentiment_scores(text)
    St.write(sentiment)
    St.bar_chart(pd.DataFrame(sentiment, index=[0]))
```

✅ 3. Complete Zip Package

- *Prepare a downloadable .zip file with:*

- Streamlit_app.py
- Nlp_pipeline.py
- Data/sample.txt
- README.md with setup instructions
- Requirements.txt
- Optional: - output screenshots