**A**

**MINI PROJECT REPORT**

**ON**

**“SENTIMENT ANALYSIS”**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (AIML)**

**Submitted By**

**G Chandu 2453-21-748-025**

**P Adithya 2453-21-748-111**

**Under the guidance**

**Of**

**Mrs. VAISHALI**

**Assistant Professor,**

**Department of Computer Science and Engineering**

****

**AFFILIATED TO OSMANIA UNIVERSITY HYDERABAD**

****

**1.INTRODUCTION**

**1.1 Problem Statement**

**1.2 Motivation**

**2. PROPOSED MODEL – Algorithm and flowchart**

**3. REQUIREMENTS – software/hardware/dataset**

**4. IMPLEMENTATION**

**5. OUTPUT SCREENSHOTS**

**6. ADVANTAGES/ DISADVANTAGES**

**7. USE OF PROJECT**

**8. CONCLUSION**

**1.INTRODUCTION**

Sentiment Analysis, also known as opinion mining, is a subfield of Natural Language Processing (NLP) that focuses on identifying and categorizing opinions expressed in a piece of text. This process determines the writer's attitude towards a particular topic, product, or service, classifying the sentiment as positive, negative, or neutral. With the proliferation of social media and online reviews, sentiment analysis has become a crucial tool for businesses and researchers to understand public opinion and make data-driven decisions.

The objective of this mini project is to develop a sentiment analysis model that can automatically classify the sentiment of given text data. This project will involve data collection, preprocessing, model training, evaluation, and deployment. By the end of the project, you will have a functional sentiment analysis tool that can be applied to various text data sources.

**1.1.Problem Statement**

In today's competitive market, understanding customer sentiment is essential for business success. Companies often receive thousands of online reviews that provide valuable insights into customer opinions and areas for improvement. However, manually analyzing these reviews is impractical due to the sheer volume of data. This project aims to develop a sentiment analysis model to automatically classify customer reviews as positive, negative, or neutral. By leveraging natural language processing and machine learning techniques, the model will enable businesses to quickly gauge customer satisfaction, identify common issues, and make data-driven decisions to enhance their products and services.

**1.2.Motivation**

The motivation for this project stems from the need to efficiently harness the power of customer feedback in an era where consumer opinions are more accessible and abundant than ever. Businessesface the challenge of sifting through massive volumes of reviews to extract actionable insights, a task that is both labor intensive and prone to human error when done manually. By developing an automated sentiment analysis system, we can transform this wealth of unstructured data into valuable information, enabling businesses to swiftly understand customer sentiments, address issues proactively, and make informed decisions that enhance product quality and customer satisfaction. This project not only aims to improve operational efficiency but also to empower businesses with the tools needed to remain competitive and responsive in a fast-paced market environment.

**2. PROPOSED MODEL – Algorithm and flowchart**

The proposed sentiment analysis model is designed to automatically classify customer reviews into positive, negative, or neutral sentiments. Leveraging natural language processing (NLP) and machine learning techniques, the model preprocesses text data by converting it to lowercase, removing punctuation, tokenizing, and eliminating stopwords. Using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer, the preprocessed text is transformed into numerical feature vectors. A Multinomial Naive Bayes classifier is then trained on these vectors to learn the patterns associated with each sentiment class. The model is evaluated on both validation and test datasets to ensure its accuracy and robustness. Additionally, the model includes a function to predict the sentiment of new, unseen text inputs, enabling real-time sentiment analysis. This approach provides businesses with an efficient and reliable tool to understand customer opinions, helping them to make data-driven decisions and improve their products and services.

**Algorithm**

# Import necessary libraries

IMPORT nltk

FROM nltk.corpus IMPORT stopwords

FROM nltk.tokenize IMPORT word\_tokenize

IMPORT string

IMPORT pandas AS pd

FROM sklearn.feature\_extraction.text IMPORT TfidfVectorizer

FROM sklearn.naive\_bayes IMPORT MultinomialNB

FROM sklearn.metrics IMPORT classification\_report

# Download required NLTK data

nltk.download('punkt', quiet=True)

nltk.download('stopwords', quiet=True)

# Define function to preprocess text

FUNCTION preprocess\_text(text):

CONVERT text TO lowercase

REMOVE punctuation FROM text

TOKENIZE text INTO words

REMOVE stopwords FROM tokens

RETURN preprocessed text AS string

# Define function to load data from file

FUNCTION load\_data(file\_path):

INITIALIZE data AS empty list

INITIALIZE labels AS empty list

OPEN file\_path FOR READING

FOR EACH line IN file:

SPLIT line INTO text AND label

APPEND preprocess\_text(text) TO data

APPEND label TO labels

RETURN data, labels

# Load and preprocess data

train\_data, train\_labels = load\_data("train.txt")

val\_data, val\_labels = load\_data("val.txt")

test\_data, test\_labels = load\_data("test.txt")

# Feature extraction using TF-IDF

INITIALIZE vectorizer AS TfidfVectorizer()

X\_train = vectorizer.fit\_transform(train\_data)

X\_val = vectorizer.transform(val\_data)

X\_test = vectorizer.transform(test\_data)

# Train the Multinomial Naive Bayes model

INITIALIZE model AS MultinomialNB()

model.fit(X\_train, train\_labels)

# Evaluate the model on validation data

val\_predictions = model.predict(X\_val)

PRINT "Validation Set Performance:"

PRINT classification\_report(val\_labels, val\_predictions)

# Evaluate the model on test data

test\_predictions = model.predict(X\_test)

PRINT "Test Set Performance:"

PRINT classification\_report(test\_labels, test\_predictions)

# Define function to predict sentiment of new text

FUNCTION predict\_sentiment(text, model, vectorizer):

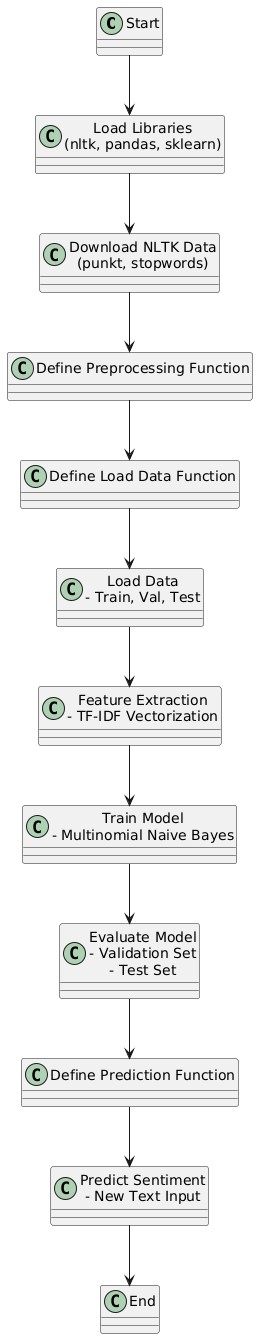
preprocessed\_text = preprocess\_text(text)

text\_vector = vectorizer.transform([preprocessed\_text])

prediction = model.predict(text\_vector)

RETURN prediction[0]

**Flowchart**



**3. REQUIREMENTS – software/hardware/dataset**

Software Requirements

1. Python:
   * Version: 3.6 or later.
   * Description: Python is the programming language used for developing the sentiment analysis model.
2. Libraries:
   * nltk:
     + Version: 3.5 or later.
     + Description: For natural language processing tasks such as tokenization and stopwords removal.
   * pandas:
     + Version: 1.0 or later.
     + Description: For data manipulation and loading.
   * scikit-learn:
     + Version: 0.24 or later.
     + Description: For machine learning tasks including feature extraction (TF-IDF) and model training (Naive Bayes).
   * Jupyter Notebook or IDE (e.g., PyCharm, VSCode):
     + Description: For writing and executing code in an interactive environment.
3. Data Files:
   * Format: .txt or other compatible formats.
   * Description: Text files containing labeled sentiment data for training, validation, and testing.
4. Optional Tools:
   * Visualization Libraries (e.g., matplotlib, seaborn):

Description: For visualizing model performance and results.

Hardware Requirements

1. Processor:
   * Type: Intel Core i5 or equivalent.
   * Description: A modern multi-core processor for efficient computation.
2. Memory (RAM):
   * Minimum: 8 GB.
   * Recommended: 16 GB or more.
   * Description: Sufficient memory is needed for handling large datasets and model training.
3. Storage:
   * Type: SSD (Solid State Drive) preferred.
   * Minimum: 10 GB of free space.
   * Description: For storing datasets, models, and software dependencies.
4. Graphics Card (Optional):
   * Type: NVIDIA GPU (e.g., GTX 1050 or higher) if working with large models or performing intensive computations.
   * Description: Not required for basic sentiment analysis, but beneficial for larger-scale machine learning tasks.
5. Internet Connection:
   * Description: Required for downloading libraries and datasets, and for any online services or cloud-based tools used in the project.

Dataset

* Dataset is taken from Kaggle,a dataset based on various emotions that

Are recorded from Tweets from various Users.

**4. IMPLEMENTATION**

**Step 1: Set Up the Environment**

1. Install Python: Ensure you have Python 3.6 or later installed on your system.
2. Create a Virtual Environment:
3. Install Required Libraries

**Step 2: Download NLTK Data**

python

Copy code

import nltk

nltk.download('punkt', quiet=True)

nltk.download('stopwords', quiet=True)

**Step 3: Preprocess the Data**

python

Copy code

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import string

def preprocess\_text(text):

text = text.lower()

text = text.translate(str.maketrans('', '', string.punctuation))

tokens = word\_tokenize(text)

stop\_words = set(stopwords.words('english'))

tokens = [token for token in tokens if token not in stop\_words]

return ' '.join(tokens)

**Step 4: Load and Prepare the Data**

python

Copy code

import pandas as pd

def load\_data(file\_path):

data = []

labels = []

with open(file\_path, 'r') as file:

for line in file:

text, label = line.strip().split(';')

data.append(preprocess\_text(text))

labels.append(label)

return data, labels

train\_data, train\_labels = load\_data('path\_to\_train.txt')

val\_data, val\_labels = load\_data('path\_to\_val.txt')

test\_data, test\_labels = load\_data('path\_to\_test.txt')

**Step 5: Feature Extraction and Model Training**

python

Copy code

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import classification\_report

# Feature extraction

vectorizer = TfidfVectorizer()

X\_train = vectorizer.fit\_transform(train\_data)

X\_val = vectorizer.transform(val\_data)

X\_test = vectorizer.transform(test\_data)

# Train model

model = MultinomialNB()

model.fit(X\_train, train\_labels)

# Evaluate model

val\_predictions = model.predict(X\_val)

print("Validation Set Performance:")

print(classification\_report(val\_labels, val\_predictions))

test\_predictions = model.predict(X\_test)

print("Test Set Performance:")

print(classification\_report(test\_labels, test\_predictions))

Step 6: Save the Model and Vectorizer

python

Copy code

import joblib

joblib.dump(model, 'sentiment\_model.joblib')

joblib.dump(vectorizer, 'vectorizer.joblib')

**Step 7: Define Prediction Function**

python

Copy code

def predict\_sentiment(text, model, vectorizer):

processed\_text = preprocess\_text(text)

text\_vector = vectorizer.transform([processed\_text])

prediction = model.predict(text\_vector)[0]

probabilities = model.predict\_proba(text\_vector)[0]

return prediction, probabilities, processed\_text, text\_vector

Step 8: Build the Streamlit App

Create a file named app.py:

python

Copy code

import streamlit as st

import joblib

from preprocess import preprocess\_text

# Load your trained model and vectorizer

model = joblib.load('sentiment\_model.joblib')

vectorizer = joblib.load('vectorizer.joblib')

def predict\_sentiment(text):

processed\_text = preprocess\_text(text)

text\_vector = vectorizer.transform([processed\_text])

prediction = model.predict(text\_vector)[0]

probabilities = model.predict\_proba(text\_vector)[0]

return prediction, probabilities, processed\_text, text\_vector

# Streamlit app

st.title('Sentiment Analysis')

# Text input

user\_input = st.text\_area("Enter your text here:")

if st.button('Analyze Sentiment'):

if user\_input:

sentiment, probabilities, processed\_text, text\_vector = predict\_sentiment(user\_input)

st.write(f"Original Text: {user\_input}")

st.write(f"Processed Text: {processed\_text}")

st.write(f"Predicted Sentiment: {sentiment}")

st.write("Class Probabilities:")

for i, prob in enumerate(probabilities):

st.write(f" Class {i}: {prob:.4f}")

else:

st.write("Please enter some text to analyze.")

# Display model information

st.write("Model Information:")

st.write(f"Model type: {type(model).\_\_name\_\_}")

st.write(f"Number of classes: {len(model.classes\_)}")

st.write(f"Classes: {model.classes\_}")

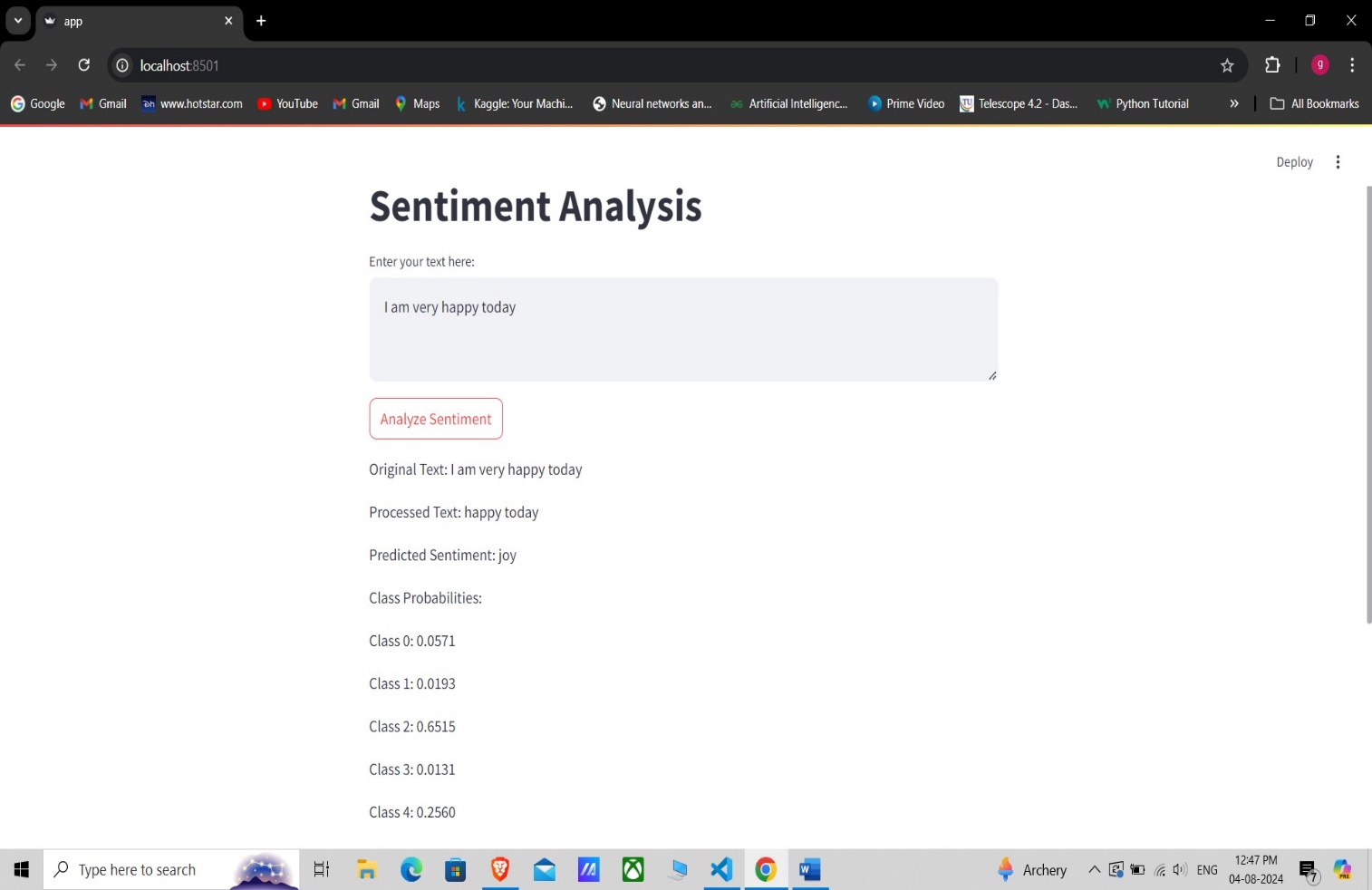
**Step 9: Run the Streamlit App**

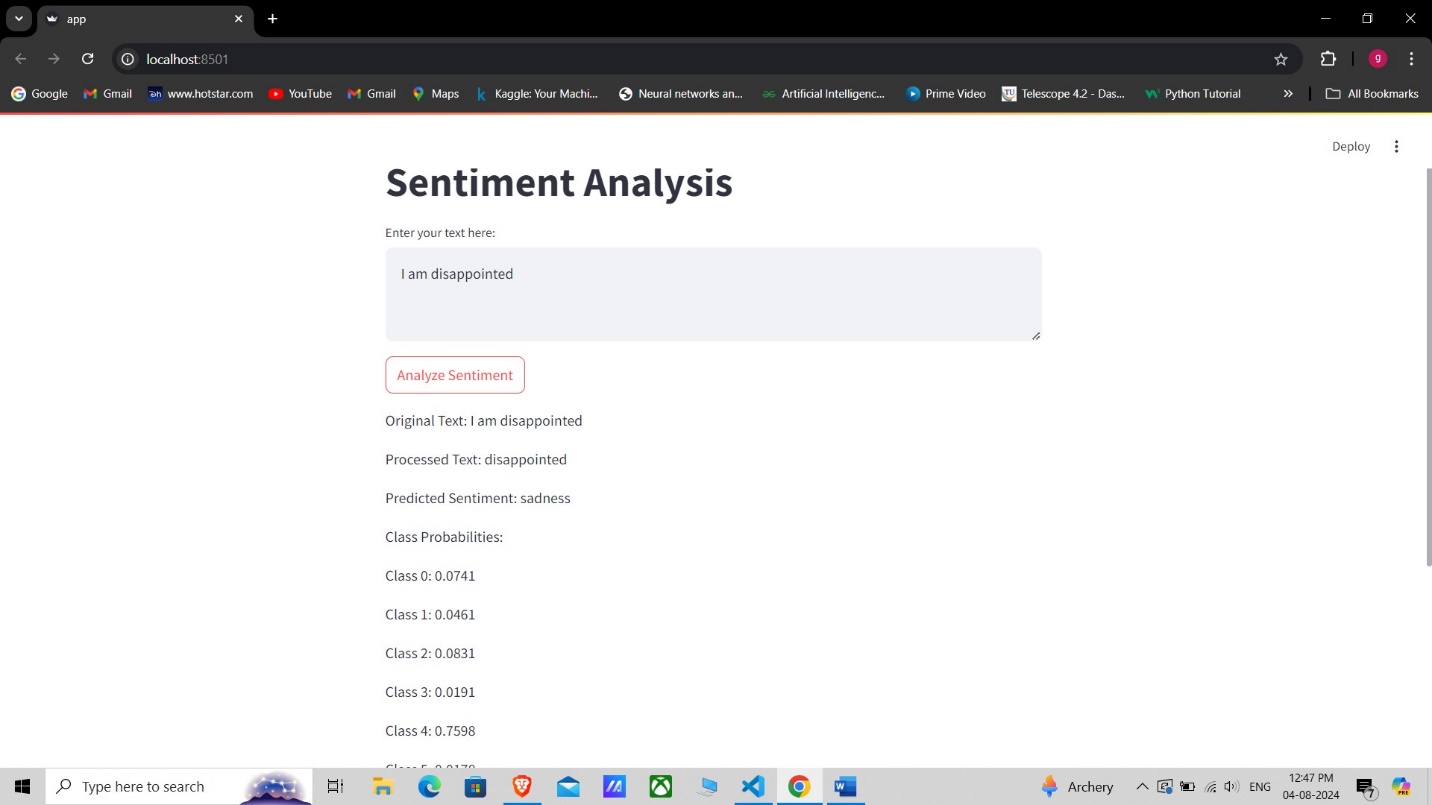
bash

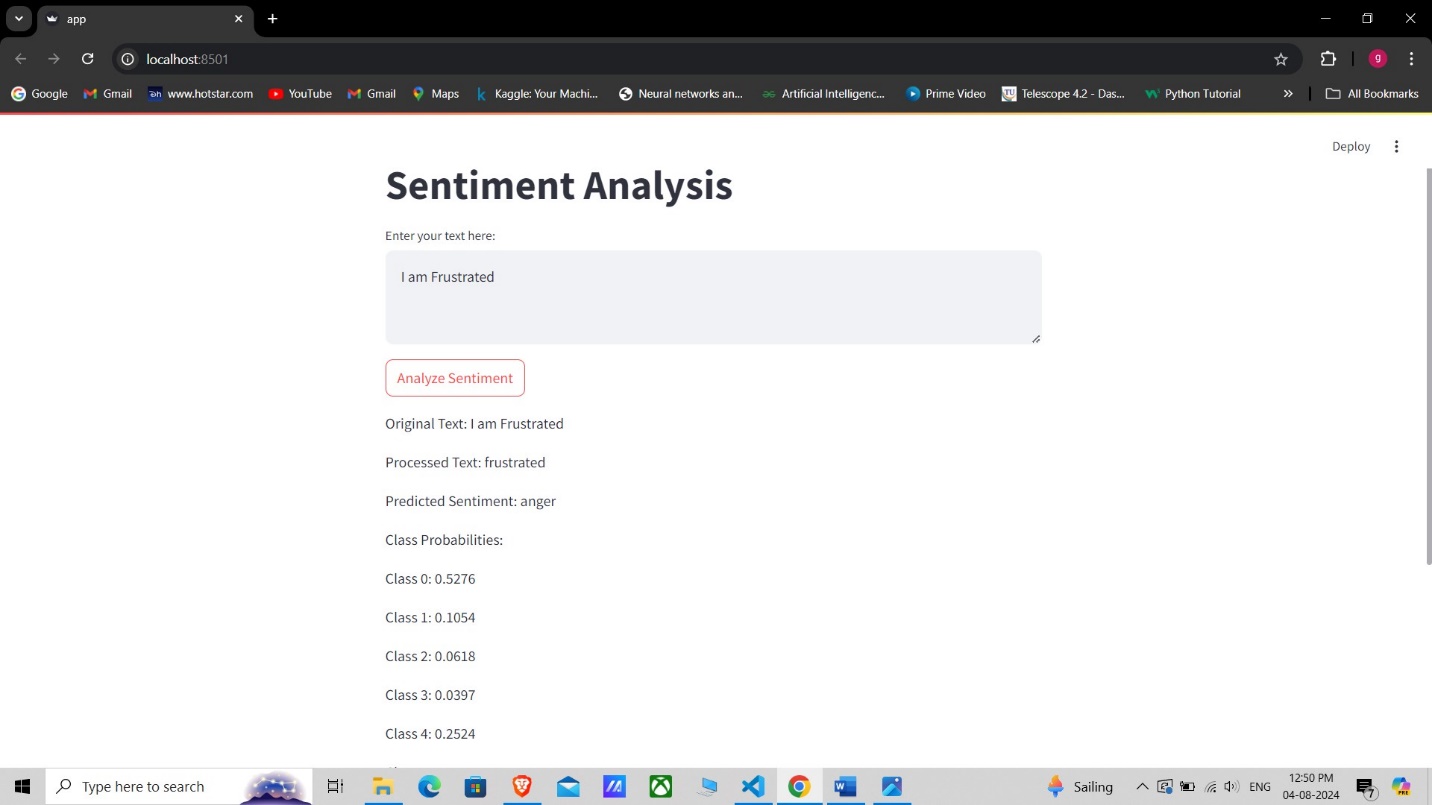
Copy code

streamlit run app.py

**4.OUTPUT SCREENSHOTS**

****





**6. ADVANTAGES/ DISADVANTAGES**

**Advantages of the Sentiment Analysis Project**

1. **Automated Sentiment Detection:**
   * The project automates the process of analyzing text to determine sentiment, saving significant time and effort compared to manual analysis.
2. **Real-Time Analysis:**
   * With the Streamlit app, users can get real-time sentiment analysis of text inputs, which is useful for live monitoring and decision-making.
3. **Improved Decision Making:**
   * Businesses and individuals can make data-driven decisions based on the sentiment analysis results, enhancing customer satisfaction and engagement strategies.
4. **Customizable and Extensible:**
   * The project is built using Python and popular libraries, making it easy to customize and extend for specific use cases or to improve model accuracy.
5. **Educational Value:**
   * The project serves as a great learning opportunity for understanding natural language processing (NLP), machine learning, and model deployment.

**Disadvantages of the Sentiment Analysis Project**

1. **Limited to Text Data:**
   * The model can only analyze text data, which means it cannot be used for other types of data like images or videos without significant modifications.
2. **Accuracy Depends on Data Quality:**
   * The accuracy of the model is highly dependent on the quality and representativeness of the training data. Poor quality data can lead to inaccurate predictions.
3. **Handling Sarcasm and Nuances:**
   * Sentiment analysis models often struggle with detecting sarcasm, irony, and nuanced expressions, which can lead to misclassification of sentiments.
4. **Language and Domain Limitations:**
   * The model is trained on English text and may not perform well on texts in other languages or from different domains without retraining on relevant datasets.
5. **Computational Resources:**
   * Training and deploying machine learning models can require significant computational resources, especially for large datasets, which might not be feasible for all users.

**7. USE OF PROJECT**

**Uses of the Sentiment Analysis Project**

1. **Customer Feedback Analysis:**
   * Businesses can analyze customer reviews, feedback, and survey responses to understand customer satisfaction and identify areas for improvement.
2. **Social Media Monitoring:**
   * Companies and organizations can monitor social media platforms to gauge public opinion and sentiment about their brand, products, or services.
3. **Market Research:**
   * Sentiment analysis can be used to analyze market trends and consumer preferences by examining online discussions and reviews.
4. **Product Improvement:**
   * By understanding the sentiment of customer feedback, companies can make data-driven decisions to enhance product features and address customer pain points.
5. **Brand Reputation Management:**
   * Sentiment analysis helps in tracking and managing a brand’s reputation by identifying positive or negative mentions and taking appropriate actions.
6. **Political Analysis:**
   * Sentiment analysis can be used to analyze public opinion on political issues, candidates, or policies by examining social media posts, news articles, and other textual data.
7. **Content Personalization:**
   * Content providers can use sentiment analysis to personalize recommendations based on the sentiment of previously consumed content.
8. **Customer Support:**
   * Sentiment analysis can be integrated into customer support systems to prioritize and address customer queries based on the sentiment and urgency of the messages.
9. **Financial Market Analysis:**
   * Financial analysts can use sentiment analysis to predict stock market trends and investor sentiment by analyzing news articles, financial reports, and social media posts.
10. **Employee Feedback and HR Management:**
    * Organizations can analyze employee feedback to gauge sentiment and improve workplace satisfaction and employee engagement.

**8. CONCLUSION**

The sentiment analysis project leverages advanced natural language processing (NLP) and machine learning techniques to transform raw text data into actionable insights. By automating the process of sentiment detection, this project enables businesses and organizations to efficiently analyze large volumes of textual data, such as customer reviews, social media posts, and feedback forms. The model's ability to provide real-time sentiment analysis is particularly valuable for applications in customer feedback analysis, brand reputation management, and market research.

While the project offers significant advantages, such as scalability, automation, and improved decision-making, it also has limitations that need careful consideration. Issues such as handling nuanced expressions, ensuring data quality, and addressing potential biases must be managed to maximize the model's effectiveness and fairness.

Overall, this sentiment analysis project serves as a powerful tool for understanding and interpreting human emotions expressed in text. It opens up numerous opportunities across various sectors, from enhancing customer satisfaction to predicting market trends. By continuing to refine and extend this model, it can be adapted to meet specific needs, driving better outcomes and deeper insights.

THANK YOU