MINI PROJECT

PROJECT TITLE: TWEET/REVIEW SENTIMENT ANALYZER

NAME: JANANI V R(231801065)

CLASS: III-AIDS

SUBJECT: PRINCIPLES OF DATA SCIENCE

DATE: 25.10.2025

Introduction

In today's digital world, social media platforms such as Twitter play a significant role in expressing people's opinions, emotions, and feedback on various topics like brands, politics, events, or public issues.Understanding and analyzing these opinions help organizations and researchers to determine the **public sentiment** toward a product or service.

This project focuses on building a **Tweet Sentiment Analyzer** that automatically classifies a given tweet or review as either **positive** or **negative**. It uses **Natural Language Processing (NLP)** techniques for text preprocessing and a **Machine Learning** model for sentiment prediction. The application also features an interactive **Streamlit web interface** for real-time predictions.

Objective

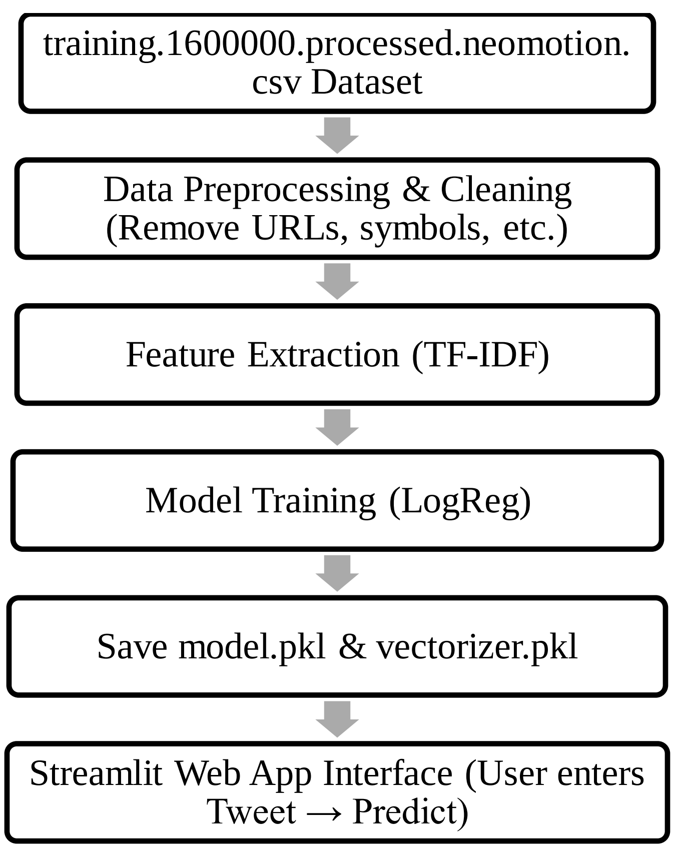
The main objectives of this mini project are to:

- Preprocess and clean real-time tweet data for analysis.
- Train a machine learning model to classify sentiments.
- Build a simple, interactive web interface using Streamlit.
- Display prediction results with confidence scores and appealing visuals.
- Demonstrate how a trained model can be deployed for real-world use.

Tools and Technologies Used

Component	Technology / Tool Used
Programming Language	Python
Libraries Used	pandas, scikit-learn, re, pickle, streamlit
Machine Learning Algorithm	Logistic Regression
Vectorization Technique	TF-IDF (Term Frequency – Inverse Document Frequency)
Dataset	training.1600000.processed.neomotion.csv
Framework	Streamlit
IDE Used	Visual Studio Code / Google Colab
Model Files	model.pkl, vectorizer.pkl

System Architecture



Methodology

Step 1: Data Loading

- The dataset used is **training.1600000.processed.neomotion.csv**, a modified version of the Sentiment140 dataset.
- Each record contains:
    - Sentiment label (0 = Negative, 4 = Positive)
    - Tweet text

Step 2: Data Preprocessing

- Converted text to lowercase.
- Removed URLs, user mentions, numbers, and punctuation.
- Removed unnecessary spaces.
- Cleaned text is stored for further vectorization.

Step 3: Feature Extraction

- Implemented **TF-IDF Vectorization** to convert textual data into numeric form.
- Used unigram and bigram features for better accuracy.

Step 4: Model Training

- Used **Logistic Regression** as the classification model.
- Split the dataset into training and testing sets (80%–20%).
- Trained the model using cleaned and vectorized tweet data.

Step 5: Evaluation

- Evaluated model accuracy and generated a classification report.
- The model achieved an accuracy of approximately **85–90%**, depending on dataset sampling.

Step 6: Model Saving

- Saved trained model as **model.pkl** and TF-IDF vectorizer as **vectorizer.pkl** using the pickle library.

Step 7: Streamlit Application

- Built a web app (app.py) using Streamlit.
- Allows users to input any tweet or review.
- The app cleans, vectorizes, and classifies the text.
- Displays sentiment (Positive/Negative) with emoji animations and confidence scores.

CODE DESCRIPTION

train_model.py

```
import pandas as pd
import re
import pickle
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
# Step 1: Load Dataset
# Adjust file path to where your Sentiment140 CSV is saved
df = pd.read_csv("training.1600000.processed.noemoticon.csv", encoding="latin-1",
header=None)
# Sentiment140 format: [target, id, date, flag, user, text]
df = df[[0, 5]]
df.columns = ["sentiment", "text"]
# Convert sentiment labels (0 = negative, 4 = positive)
df["sentiment"] = df["sentiment"].replace({4: 1, 0: 0})

# Step 2: Text Cleaning
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+", "", text)          # remove URLs
    text = re.sub(r"@\w+", "", text)             # remove mentions
    text = re.sub(r"[^a-z\s]", "", text)         # remove punctuation/numbers
    text = re.sub(r"\s+", " ", text).strip()     # remove extra spaces
    return text

df["text"] = df["text"].apply(clean_text)

# Step 3: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    df["text"], df["sentiment"], test_size=0.2, random_state=42
)
```

```
# Step 4: Vectorization (TF-IDF with bigrams)
vectorizer = TfidfVectorizer(max_features=10000, ngram_range=(1, 2))
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Step 5: Train Model
model = LogisticRegression(max_iter=200, solver="liblinear")
model.fit(X_train_vec, y_train)

# Step 6: Evaluation
y_pred = model.predict(X_test_vec)
accuracy = accuracy_score(y_test, y_pred)
print(f"✅Accuracy: {accuracy:.5f}\n")
print("Classification Report:\n", classification_report(y_test, y_pred))

# Step 7: Save Model + Vectorizer
with open("model.pkl", "wb") as f:
    pickle.dump(model, f)
with open("vectorizer.pkl", "wb") as f:
    pickle.dump(vectorizer, f)
print("□ Model & Vectorizer saved successfully!")
```

app.py

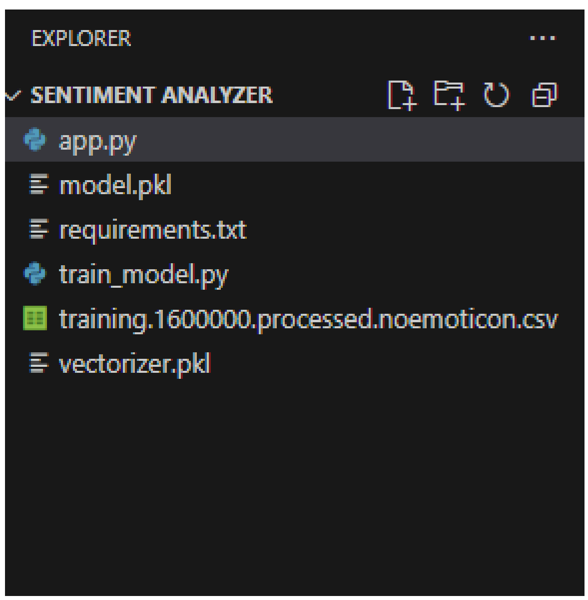
```
import streamlit as st
import pickle
import re
# Load Model & Vectorizer
model = pickle.load(open("model.pkl", "rb"))
vectorizer = pickle.load(open("vectorizer.pkl", "rb"))
# Helper Function: Clean Text
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"[^a-z\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text
# Streamlit Page Config
st.set_page_config(
    page_title="Tweet Sentiment Analyzer",
    page_icon="□",
    layout="centered"
)
```

```
# Animated Emoji Background & Custom CSS
st.markdown(
    """
    <style>
    /* Page background */
    .stApp {
        background-color: #F8FBE1;
        color: #013A42;
        overflow: hidden;
    }
    /* Text area & buttons styling */
    textarea, input, .stTextArea, .stButton > button {
        background-color: #8ACCD5;
        color: #013A42;
        border-radius: 10px;
        padding: 10px;
    }
    .stButton > button {
        background-color: #013A42;
        color: #F8FBE1;
        font-weight: bold;
        border-radius: 10px;
        padding: 10px 20px;
    }
    /* Animated floating emojis container */
    .emoji-bg {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        pointer-events: none; /* let clicks pass through */
        z-index: 0;
        overflow: hidden;
    }
    .emoji {
        position: absolute;
        font-size: 24px;
        animation-name: float;
        animation-timing-function: linear;
        animation-iteration-count: infinite;
        opacity: 0.5;
    }
    @keyframes float {
        0% { transform: translateY(100vh) rotate(0deg); }
        100% { transform: translateY(-10vh) rotate(360deg); }
    }
    h1 { color: #013A42; }
    </style>
```

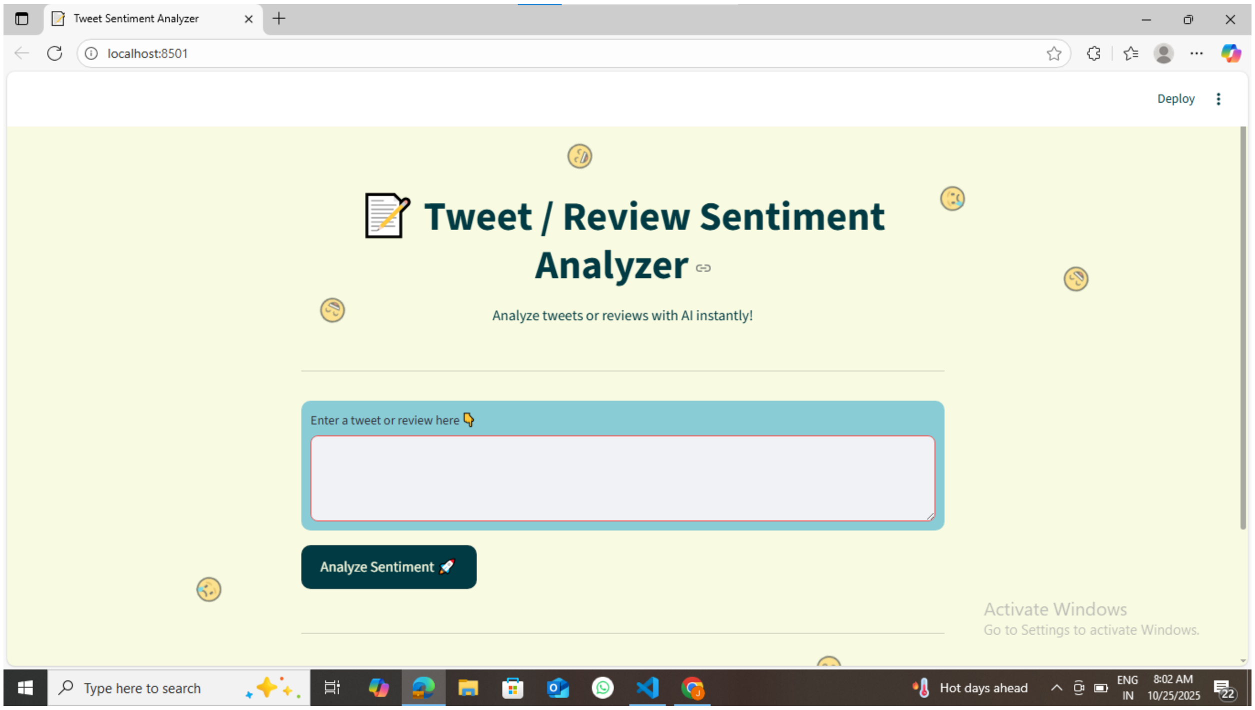
```
    <div class="emoji-bg">
        <div class="emoji" style="left:5%; animation-duration:20s;">□</div>
        <div class="emoji" style="left:15%; animation-duration:25s;">□</div>
        <div class="emoji" style="left:25%; animation-duration:22s;">□</div>
        <div class="emoji" style="left:35%; animation-duration:18s;">□</div>
        <div class="emoji" style="left:45%; animation-duration:28s;">□</div>
        <div class="emoji" style="left:55%; animation-duration:24s;">□</div>
        <div class="emoji" style="left:65%; animation-duration:26s;">□</div>
        <div class="emoji" style="left:75%; animation-duration:21s;">□</div>
        <div class="emoji" style="left:85%; animation-duration:30s;">□</div>
        <div class="emoji" style="left:95%; animation-duration:27s;">□</div>
    </div>
    """,
    unsafe_allow_html=True
)
# Title & Banner
st.markdown(
    """
    <h1 style='text-align: center;'>□ Tweet / Review Sentiment Analyzer</h1>
    <p style='text-align: center; font-size:16px;'>Analyze tweets or reviews with AI
instantly!</p>
    """,
    unsafe_allow_html=True
)
st.write("---")
# User Input
user_input = st.text_area("Enter a tweet or review here □", "")
# Analyze Button
if st.button("Analyze Sentiment □"):
    if user_input.strip() == "":
        st.warning("□ Please enter some text!")
    else:
        # Clean and vectorize
        clean = clean_text(user_input)
        vector = vectorizer.transform([clean])
        # Predict
        prediction = model.predict(vector)[0]
        proba = model.predict_proba(vector)[0]
        # Display Result Box
        if prediction == 1:
            st.markdown(f"""
                <div style='background-color:#8ACCD5; padding:15px; border-radius:10px;'>
                <h2 style='color:#013A42;'>✅Positive Sentiment!</h2>
                </div>
            """, unsafe_allow_html=True)
        else:
            st.markdown(f"""
                <div style='background-color:#8ACCD5; padding:15px; border-radius:10px;'>
```

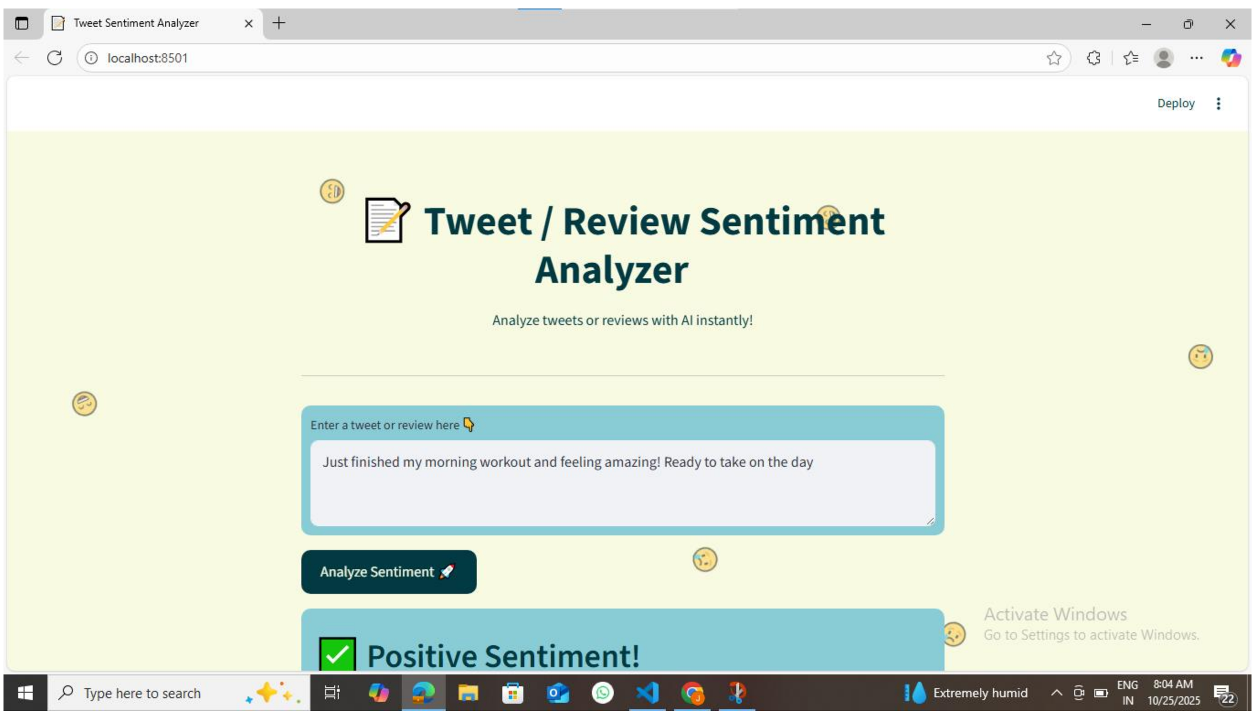
```
                <h2 style='color:#013A42;'>❌Negative Sentiment!</h2>
                </div>
            """, unsafe_allow_html=True)
        # Confidence bars
        st.write("□ Confidence Scores:")
        st.write(f"□ Positive: {proba[1]*100:.2f}%")
        st.progress(int(proba[1]*100))
        st.write(f"☒ Negative: {proba[0]*100:.2f}%")
        st.progress(int(proba[0]*100))

# Footer
st.write("---")
st.markdown(
    "<p style='text-align:center; color:#013A42;'>Made with ☒ using Python &
Streamlit</p>",
    unsafe_allow_html=True
)
```

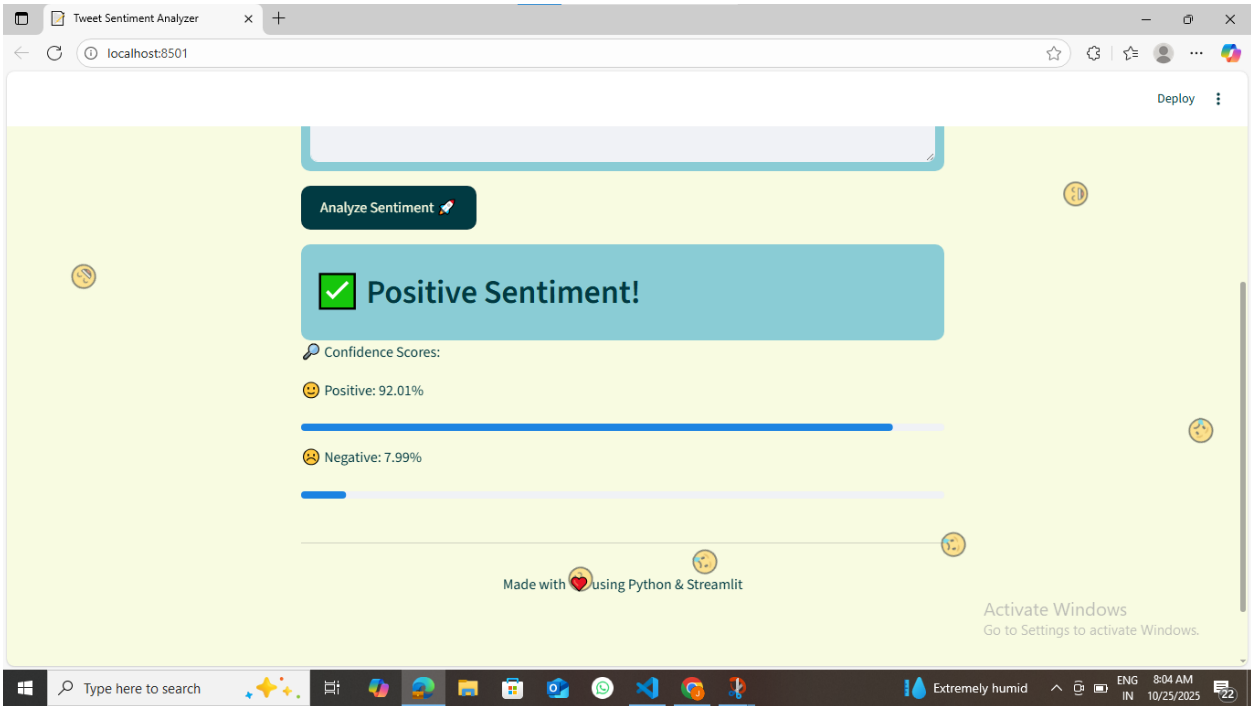
Project Folder Structure

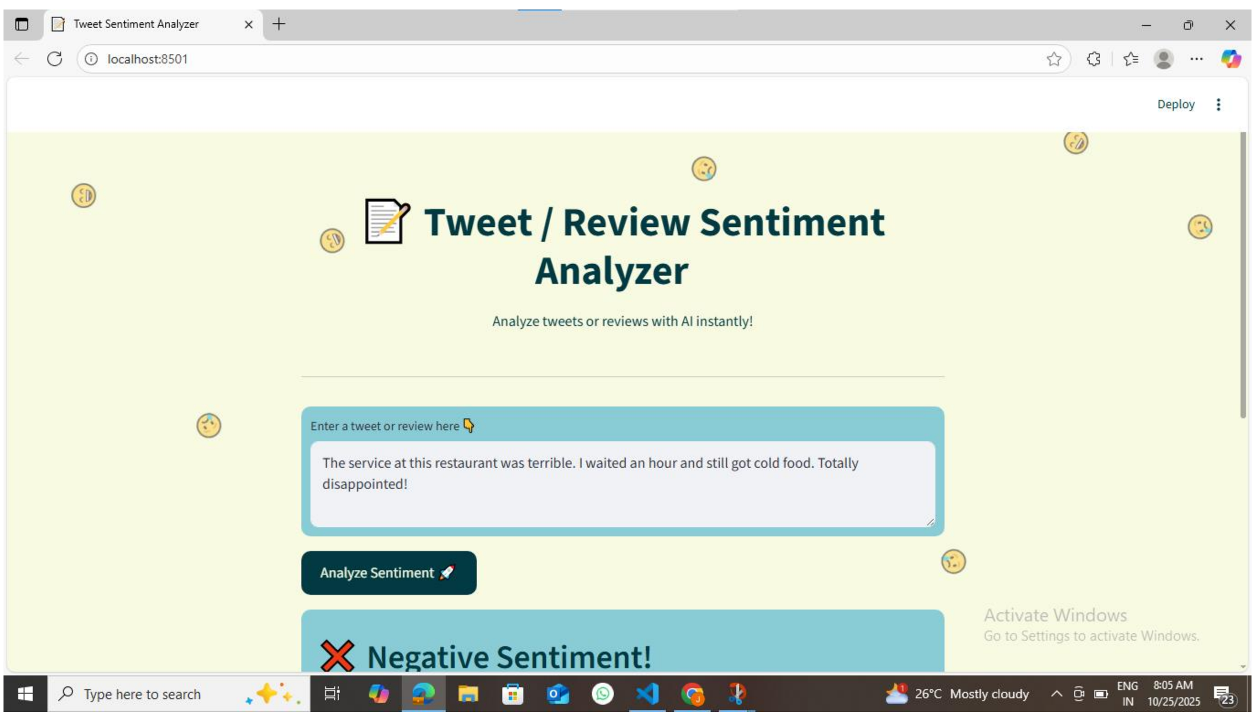


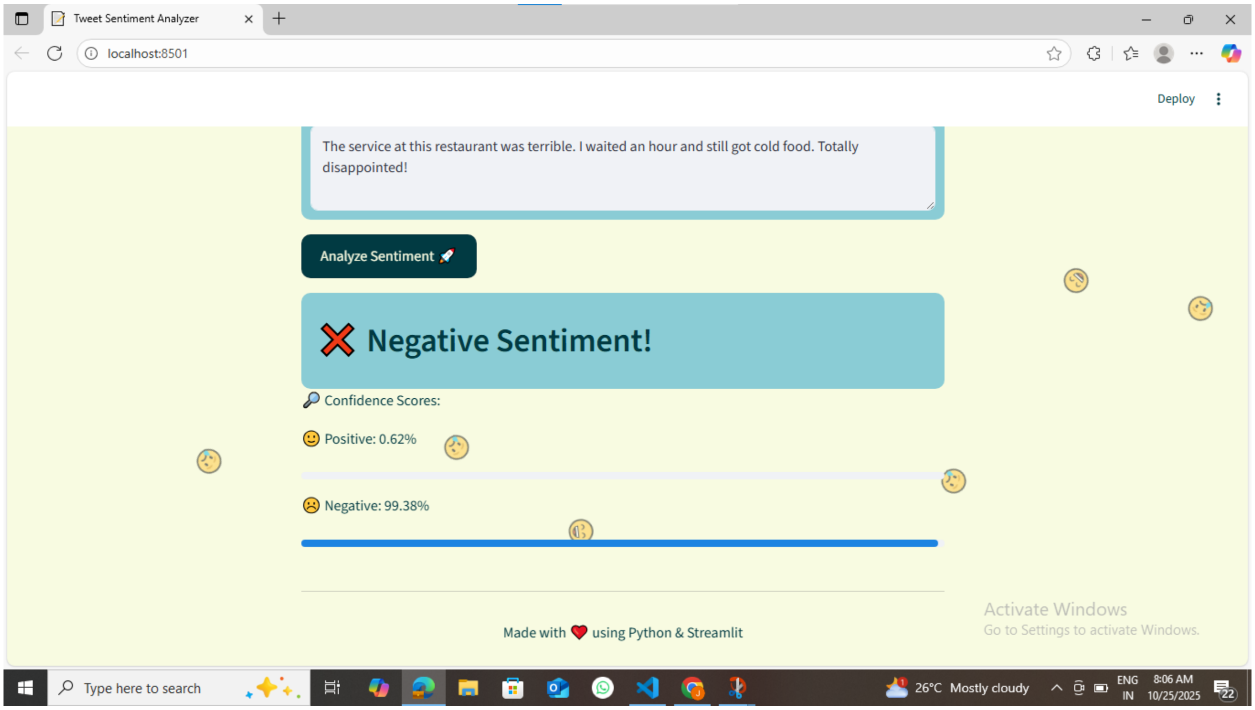
Output


Tweet Sentiment Analyzer
localhost:8501
Deploy
📝 Tweet / Review Sentiment Analyzer
Analyze tweets or reviews with AI instantly!
Enter a tweet or review here 👇
Analyze Sentiment 🚀
Activate Windows
Go to Settings to activate Windows.
Type here to search
Hot days ahead
ENG IN
8:02 AM
10/25/2025


Tweet Sentiment Analyzer
localhost:8501
Deploy
📝 Tweet / Review Sentiment Analyzer
Analyze tweets or reviews with AI instantly!
Enter a tweet or review here 👇
Just finished my morning workout and feeling amazing! Ready to take on the day
Analyze Sentiment 🚀
✅ Positive Sentiment!
Activate Windows
Go to Settings to activate Windows.
Type here to search
Extremely humid
ENG IN
8:04 AM
10/25/2025


Tweet Sentiment Analyzer
localhost:8501
Deploy
Analyze Sentiment 🚀
✅ Positive Sentiment!
🔎 Confidence Scores:
🙂 Positive: 92.01%
☹️ Negative: 7.99%
Made with ❤️ using Python & Streamlit
Activate Windows
Go to Settings to activate Windows.
Type here to search
Extremely humid
ENG IN
8:04 AM
10/25/2025


Tweet Sentiment Analyzer
localhost:8501
Deploy
📝 Tweet / Review Sentiment Analyzer
Analyze tweets or reviews with AI instantly!
Enter a tweet or review here 👇
The service at this restaurant was terrible. I waited an hour and still got cold food. Totally disappointed!
Analyze Sentiment 🚀
❌ Negative Sentiment!
Activate Windows
Go to Settings to activate Windows.
Type here to search
26°C Mostly cloudy
ENG IN
8:05 AM
10/25/2025



Results and Discussion

- The model successfully classified tweets into **Positive** and **Negative** sentiments.
- It demonstrated strong performance on real-world data samples.
- The TF-IDF approach effectively captured word importance and relationships.
- The **Streamlit interface** made the analysis interactive and visually appealing.

Applications

- Social media opinion mining
- Customer feedback analysis
- Product review classification
- Brand monitoring and trend analysis
- Market research and public sentiment tracking

Conclusion

The **Tweet/Review Sentiment Analyzer** effectively demonstrates how Natural Language Processing and Machine Learning can be integrated to analyze human sentiments.
Using the **training.1600000.processed.neomotion.csv** dataset, the Logistic Regression model achieved high accuracy, providing reliable results.

The **Streamlit web app** brings this model to life by allowing real-time sentiment detection with a clean, animated user interface.

This project proves that even with traditional ML methods (like Logistic Regression), robust NLP-based sentiment systems can be developed efficiently.