EX.No: Date:	TRAINING A CUSTOM NLP MODEL AND DEPLOYING WITH FASTAPI

Aim

To train a custom NLP model for sentiment analysis and deploy it as a web API using FastAPI so that users can send text input and get real-time sentiment predictions.

Theory

- FastAPI is a modern and high-performance Python framework used to build APIs quickly and efficiently.
- It is based on Starlette for web handling and Pydantic for data validation.
- It provides automatic API documentation through Swagger UI and ReDoc.
- In this project, FastAPI is used to deploy a trained sentiment analysis NLP model so users can send text input and get real-time predictions.
- The trained model and vectorizer are loaded into the FastAPI application using joblib.
- When the user enters text, FastAPI transforms it using the vectorizer and predicts the sentiment using the model.
- The prediction result is returned as a JSON response to the client.
- The GET route serves the HTML page for the user interface, while the POST route processes user input and sends the prediction.
- Pydantic validates the input data structure to ensure correct processing.
- The Uvicorn ASGI server is used to run the FastAPI application and handle multiple requests efficiently.
- FastAPI is preferred because it is simple, fast, and well suited for deploying machine learning models as web services.
- It offers automatic validation, error handling, and interactive documentation for easy testing.

Implementation

1. Installation of Required Packages

- pip install fastapi
- pip install uvicorn
- pip install scikit-learn
- pip install joblib
- pip install pydantic

2. Training the Model (train_model.py)

- Prepare sample text data and their sentiment labels.
- Convert text into numeric form using CountVectorizer.
- Train a MultinomialNB model on the vectorized data.
- Save the model and vectorizer using joblib

Code:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
import joblib
# Step 1: Example training data
texts = [
    "I love this product",
    "This is amazing",
    "I hate this movie",
    "This is terrible",
    "It’s okay, not great",
    "Absolutely fantastic experience",
    "Worst day ever"
]
labels = ["positive", "positive", "negative", "negative", "neutral", "positive", "negative"]

 # Step 2: Convert text to numbers
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(texts)

# Step 3: Train model
model = MultinomialNB()
model.fit(X, labels)
```

```
# Step 4: Save model and vectorizer
joblib.dump(model, "sentiment_model.pkl")
joblib.dump(vectorizer, "vectorizer.pkl")
print("☑ Model trained and saved successfully!")
```

3. Deploying API with FastAPI (app.py)

1. Load the trained model and vectorizer.
2. Create FastAPI app
3. Define routes:
    - GET / → Serve frontend HTML page.
    - POST /predict → Accept user text and return sentiment prediction.

app.py

```
from fastapi import FastAPI
from fastapi.responses import HTMLResponse, FileResponse
from pydantic import BaseModel
import joblib
app = FastAPI()
model = joblib.load("sentiment_model.pkl")
vectorizer = joblib.load("vectorizer.pkl")
cLass TextInput(BaseModel):
    text: str
    @app.get("/", response_class=HTMLResponse)
def home():
  return FileResponse("index.html")
@app.post("/predict")
def predict_sentiment(data: TextInput):
  X = vectorizer.transform([data.text])
  prediction = model.predict(X)[0]
  return {"sentiment": prediction
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Sentiment Analyzer | FastAPI</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gradient-to-r from-blue-50 to-indigo-100 min-h-screen flex items-center justify-center font-sans">
  <div class="bg-white shadow-2xl rounded-2xl p-10 w-full max-w-md text-center">
    <h1 class="text-3xl font-bold text-indigo-700 mb-4">Sentiment Analyzer</h1>
    <p class="text-gray-600 mb-8">Analyze emotions in your text instantly using <b>FastAPI
</b></p>
    <div>
      <input id="userInput"
           type="text"
           placeholder="Type your sentence here..."
           class="w-full border-2 border-indigo-300 focus:border-indigo-600 rounded-lg px-4 py-3 text-gray-700 outline-none mb-4 transition duration-200"
     <button onclick="analyze()"
            class="w-full bg-indigo-600 hover:bg-indigo-700 text-white font-semibold py-3 rounded-lg shadow-md transition duration-200">
       Analyze Sentiment
     </button>
    </div>
    <div id="result" class="mt-6 text-lg font-semibold"></div>
    <footer class="mt-10 text-gray-500 text-sm">
      Built using <span class="text-indigo-600 font-semibold">FastAPI</span> 🚀
```

```
</footer>
  </div>
  <script>
    async function analyze() {
      const text = document.getElementById("userInput").value.trim();
      const resultDiv = document.getElementById("result");
      if (!text) {
  resultDiv.innerHTML = '<span class="text-red-500">⚠ Please enter some text!</span>';
        return;
      }
      resultDiv.innerHTML = '<span class="text-gray-500">Analyzing...</span>';
      try {
        const response = await fetch("/predict", {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({ text }),
        });
        const data = await response.json();
        let color = "text-gray-700";
        if (data.sentiment === "positive") color = "text-green-600";
        else if (data.sentiment === "negative") color = "text-red-600";
        else if (data.sentiment === "neutral") color = "text-yellow-600";
        resultDiv.innerHTML = `Sentiment: <span
class="${color}">${data.sentiment.toUpperCase()}</span>`;
      } catch (error) {
        resultDiv.innerHTML = '<span class="text-red-500">❌ Error connecting to
API.</span>';
      }
    }
```

```
</script>
</body>
</html>
```

Algorithm

Step 1: Train Model

1. Collect sample text and sentiment labels.
2. Convert text to numerical vectors using CountVectorizer.
3. Train a MultinomialNB classifier on the vectorized data.
4. Save the model and vectorizer using joblib.

Step 2: Build API

1. Create FastAPI application.
2. Load trained model and vectorizer.
3. Define / route to serve the HTML page.
4. Define /predict route to:
    - Accept user text input.
    - Transform text using the vectorizer.
    - Predict sentiment using the model.
    - Return JSON response.

Step 3: Frontend Interaction

1. User enters text in input box.
2. JavaScript sends POST request to /predict.
3. Receive sentiment prediction and display it with color coding.

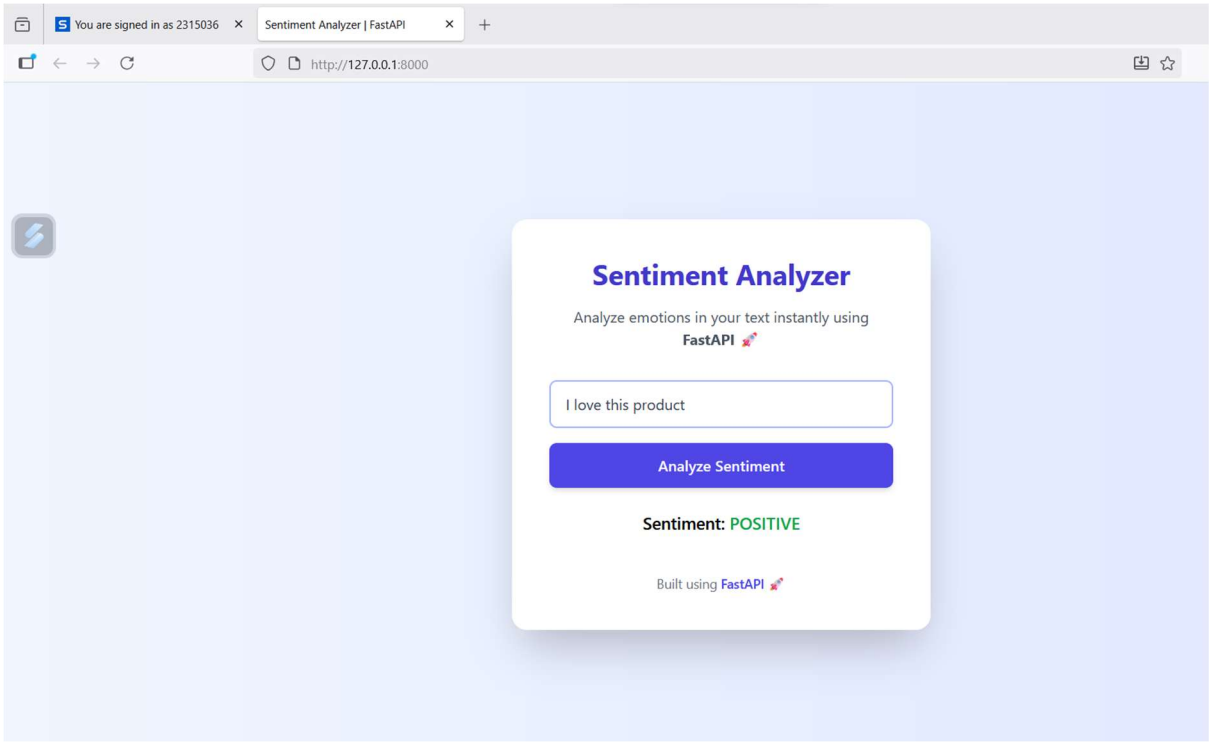
Step 4: Run the Application

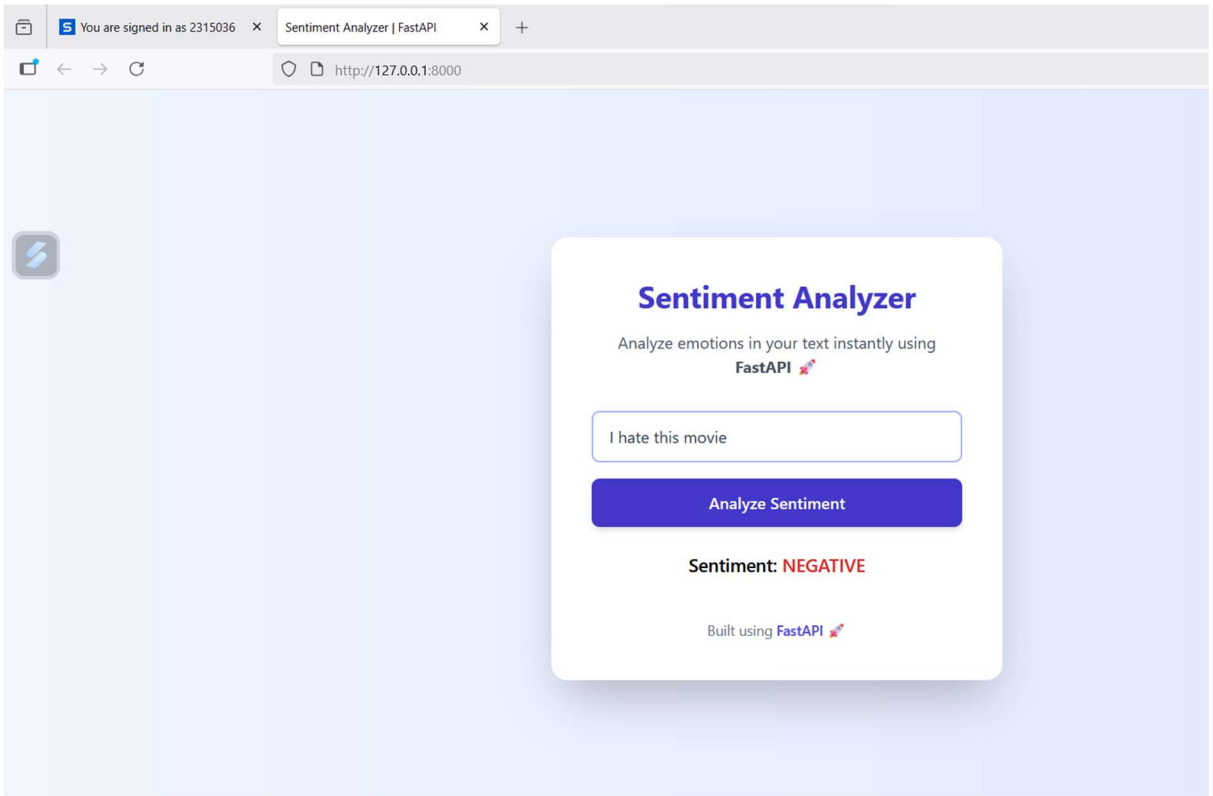
uvicorn app:app –reload

Open browser and navigate to http://127.0.0.1:8000

Enter text and observe sentiment predictions.

OUTPUT


You are signed in as 2315036
Sentiment Analyzer | FastAPI
http://127.0.0.1:8000
Sentiment Analyzer
Analyze emotions in your text instantly using FastAPI
I love this product
Analyze Sentiment
Sentiment: POSITIVE
Built using FastAPI


You are signed in as 2315036
Sentiment Analyzer | FastAPI
http://127.0.0.1:8000
Sentiment Analyzer
Analyze emotions in your text instantly using FastAPI
I hate this movie
Analyze Sentiment
Sentiment: NEGATIVE
Built using FastAPI

Problem Understanding and Objective Clarity (20)	Implementation and coding (40)	Output Evaluation and Discussion (30)	Viva (10)	Total (100)

Result

Thus ,the train a custom NLP model for sentiment analysis and deploy it as a web API using FastAPI so that users can send text input and get real-time sentiment predictions has been executed practically.