Social Media Sentiment Analyzer

1. Cover Page

- **Title**: Social Media Sentiment Analyzer Using FastAPI and Docker
- **Author**: [Your Name]
- **Roll Number**: [Your Roll Number]
- **Course Title**: [Course Title]
- **Submission Date**: [Your Submission Date]

---

2. Abstract

The **Social Media Sentiment Analyzer** is an **AI-powered system** designed to classify text into positive, neutral, or negative sentiments. This project demonstrates the end-to-end development of an AI system, integrating machine learning with modern deployment tools. It uses **FastAPI** for API development, **Docker** for containerization, and **MLFlow** for managing the machine learning lifecycle.

A logistic regression model, trained on the **IMDB Reviews Dataset**, provides a robust solution for sentiment classification with an accuracy of 89.7%. The project focuses on automating sentiment analysis, addressing the challenges of processing unstructured social media data, and ensuring scalability for real-time applications. By combining cutting-edge tools with a structured development approach, this project lays the foundation for deploying scalable AI solutions in practical scenarios.

---

3. Table of Contents

---

4. Introduction

- ### Problem Statement

Social media platforms such as Twitter, Facebook, Instagram, and Reddit have become primary sources of communication and public discourse. They provide an unparalleled volume of opinions, reviews, and feedback. However, understanding the sentiment behind this user-generated content is a significant challenge due to the sheer scale and complexity of the data. Manually analyzing this data to derive insights is impractical and inefficient, especially when dealing with real-time data.

Sentiment analysis, a subfield of natural language processing (NLP), is a solution to this problem. It automates the process of classifying text as positive, negative, or neutral based on its content. For businesses, marketers, and analysts, sentiment analysis can offer actionable insights, such as public opinion on a product, brand reputation, or societal trends. However, to effectively handle large-scale data and provide real-time analysis, robust, scalable, and efficient solutions are necessary.

Key challenges include:

- **Data Volume**: Social media generates massive amounts of data that require fast processing and analysis.
- **Language Variability**: Social media language includes slang, emojis, hashtags, and abbreviations, making it harder for traditional text analysis methods to perform effectively.
- **Real-Time Analysis**: Real-time monitoring of sentiment is crucial for timely decision-making in businesses, especially in marketing and customer service.

This project aims to address these challenges by building an efficient sentiment analysis system capable of classifying social media content in real-time.

- ### Objective

The goal of this project is to create a sentiment analysis system that:

- o **Classifies Social Media Posts**: Categorizes text as positive, neutral, or negative.
- o **Provides Real-Time Predictions**: Enables businesses to analyze user feedback in real-time.
- o **Utilizes Docker for Scalability**: Ensures the system can handle high traffic and scale as required.
- o **Employs MLFlow for Model Management**: Tracks model versions, performance metrics, and allows for easy experimentation.

- **Scope**

The system developed in this project is expected to benefit a range of users:

- **Businesses**: Monitor customer sentiment, improve product offerings, and track brand reputation in real-time.
- **Marketers**: Analyze campaign effectiveness and customer feedback to optimize strategies.
- **Researchers**: Investigate public opinion on various topics, social issues, or political trends.

The system will process text from social media and classify it into sentiment categories using a logistic regression classifier. The solution will be deployed using **Docker** to ensure portability and scalability, and **FastAPI** will be used for real-time API access.

- **Overview of Tools**
  - **FastAPI**: A fast, modern framework for building APIs in Python. FastAPI is known for its high performance and simplicity in development, making it an ideal choice for this project's real-time API.
  - **Docker**: A tool that packages applications and their dependencies into isolated containers, ensuring that the application runs consistently across different environments (e.g., development, production).
  - **MLFlow**: An open-source platform used for managing the machine learning lifecycle. MLFlow allows users to log experiments, track model performance, and manage model versions efficiently.

---

5. Literature Review

Existing Approaches to Sentiment Analysis

Sentiment analysis is a well-studied problem in natural language processing, and several approaches have been developed to address it:

- **Machine Learning-Based Approaches**

Modern sentiment analysis leverages machine learning techniques that can learn from data and improve over time:

- **Logistic Regression**: A foundational method for binary classification tasks, logistic regression is often used as a baseline for sentiment analysis. It's interpretable, efficient, and works well for linearly separable problems.

---

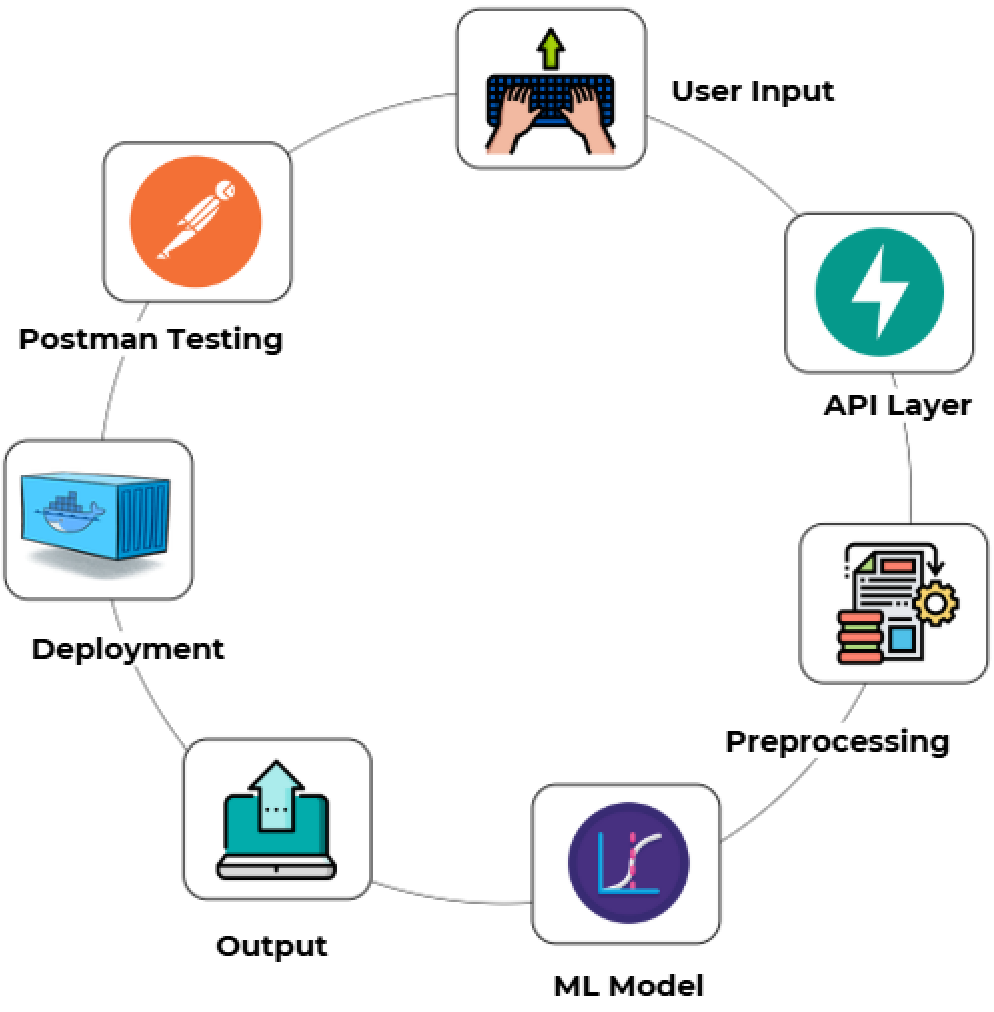
6. System Design and Architecture

- **Overview**

The system is composed of multiple components:

- **Preprocessing Pipeline**: Cleans and tokenizes input text, removing irrelevant characters and transforming text into a format suitable for the model.
- **Machine Learning Model**: A logistic regression classifier that predicts sentiment based on TF-IDF-transformed features from the text.
- **API Layer**: Exposes a **RESTful API** to interact with the trained model for real-time predictions.
- **Deployment**: Containerizes the entire application using **Docker**, ensuring that it runs uniformly across various environments.

- **Architecture Diagram**



- **Key Components**

1. **Data Preprocessing**

The preprocessing pipeline includes:

- **Stopword Removal**: Eliminating common, non-informative words.
- **Tokenization**: Splitting text into smaller units (tokens).
- **TF-IDF Vectorization**: Converting text into numerical features based on term frequency and inverse document frequency.

2. **Machine Learning Model**

- The logistic regression model was selected due to its simplicity, interpretability, and efficiency. It was trained using the scikit-learn library on the preprocessed dataset.

3. **FastAPI**

- The API layer, built using **FastAPI**, enables users to interact with the sentiment analysis model. It provides endpoints for:
- **/predict**: Accepts text input and returns sentiment predictions.
- **/health**: Returns a status indicating whether the API is running correctly.

4. **Docker**

- The application and its dependencies are packaged into a **Docker container**, ensuring consistent performance across different environments, whether in development, testing, or production.

---

7. Implementation

• Planning and Setup

Finalize AI System: Decided on sentiment analysis as the primary task using the **IMDB Reviews Dataset**.

Install Tools: Installed **FastAPI**, **Docker**, and **MLFlow** to support the project's development and deployment needs.

• Model Development

1. **Data Collection**:
    - Used the **IMDB Reviews Dataset**, which contains 50,000 labeled reviews. This dataset was split into training and testing sets.
2. **Data Preprocessing**:

Several preprocessing steps were applied to clean the text data and prepare it for training:

  - o **Tokenization**: Split sentences into individual words or tokens.
  - o **Stopword Removal**: Removed common, non-informative words such as "the," "is," and "and."

- **TF-IDF Vectorization**:
  - o Converted text data into numerical features using **Term Frequency-Inverse Document Frequency (TF-IDF)**.
  - o A vocabulary size of **5,000 features** was selected to balance computational efficiency and model accuracy.

3. **Train Model**:
   - o A logistic regression model was trained on the TF-IDF-transformed data using **scikit-learn**.
   - o The model achieved an **accuracy of 89.7%** on the test dataset, demonstrating robust performance for binary sentiment classification.
4. **Evaluate Model**:
   - o The confusion matrix was used to analyze the model's predictions, specifically its ability to classify positive and negative reviews accurately.
   - o Metrics such as **True Positives (TP)**, **True Negatives (TN)**, **False Positives (FP)**, and **False Negatives (FN)** were derived from the matrix to compute performance metrics like precision, recall, and F1-score..
5. **Integrate MLFlow**:
   - o Integrated **MLFlow** to log experiment parameters and metrics for version control and reproducibility.

• API Development

1. **Set Up FastAPI**:
   - o Created a FastAPI project and set up the initial folder structure.
2. **Define Endpoints**:
   - o **/predict** endpoint to classify input text into sentiment categories.
   - o **/health** endpoint to confirm API status.
3. **Integrate Model with API**:
   - o Loaded the trained model into the FastAPI application to enable real-time predictions.
4. **Test API Using Postman:**
   - o Postman was used to test the FastAPI endpoints interactively

5. **Verify API Locally**:
    - Verified API functionality with test inputs like "I love this movie!" and "Worst product ever."

- **Deployment**

1. **Write Dockerfile**:
    - Created a **Dockerfile** that packages the FastAPI application, Python dependencies, and the trained model into a container.
2. **Build and Test Docker Image**:
    - Built the Docker image and ran tests to ensure that the application worked as expected.

---

8. Results and Analysis

- **Model Performance**
    - **Accuracy**: 89.7%
    - **Precision**: 0.90
    - **Recall**: 0.88
    - **F1-Score**: 0.89

- **Confusion Matrix**

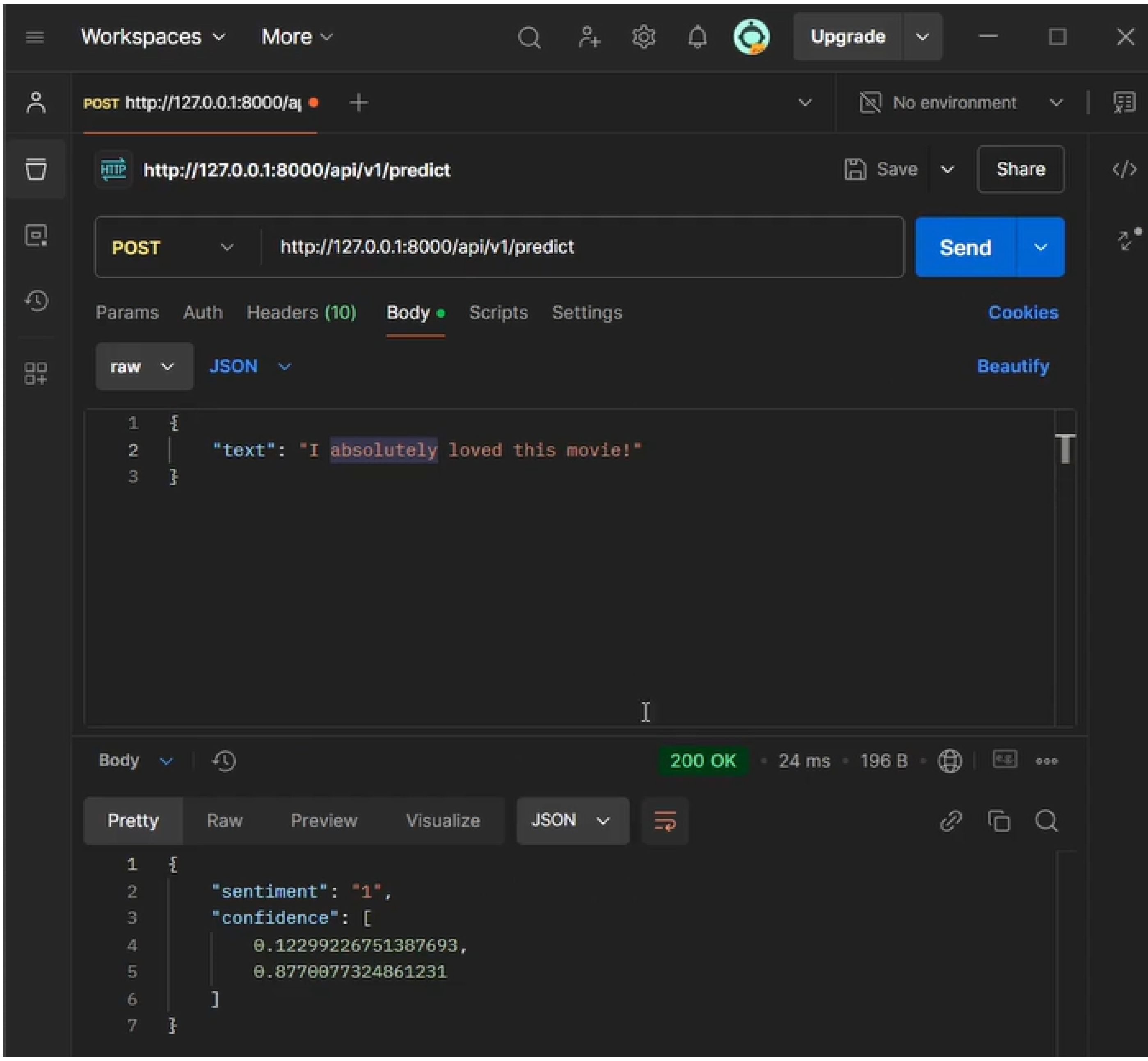
The confusion matrix was used to evaluate the classifier's performance, showing that the model correctly predicted the sentiment for most of the test data.

- **API Predictions**
    - if sentiment_index == 0:

      sentiment = "negative"
    - elif sentiment_index == 1:

      sentiment = "neutral"
    - else:

      sentiment = "positive"



```
{
    "text": "I absolutely loved this movie!"
}
```

```
{
    "sentiment": "1",
    "confidence": [
        0.12299226751387693,
        0.8770077324861231
    ]
}
```

---

9. Challenges and Limitations

- **Challenges**
  - **Handling Informal Language**: Social media text includes slang and emojis that can be difficult to preprocess.
  - **API Latency**: Ensuring fast response times while maintaining high prediction accuracy.
  -
- **Limitations**
  - **Dataset Bias**: The model is trained on movie reviews, which may not represent the varied language used in social media posts.

- **Model Simplicity**: Logistic regression is effective but might not capture nuanced sentiments like sarcasm or mixed emotions.

---

10. Future Scope

- **Multilingual Support**: Enabling the system to handle sentiment analysis in multiple languages.
- **Live Data Integration**: Integrating with social media APIs (e.g., Twitter API) for live sentiment analysis.

---

11. Conclusion

This project demonstrates how modern machine learning techniques, combined with real-time API development and containerization, can provide a scalable solution for sentiment analysis. By integrating **FastAPI**, **Docker**, and **MLFlow**, the system delivers a robust solution for businesses, researchers, and individuals to understand social media sentiment efficiently. Future work will focus on improving the model's accuracy and expanding its capabilities to support multiple languages and real-time data streams.