**Youtube Sentiment Analysis Web Application Project Report**

**1. Introduction**

This project presents a web-based sentiment analysis application that determines the sentiment (Positive, Negative, or Neutral) of user-submitted text. It combines front-end technologies (HTML, CSS, JavaScript) for user interaction with back-end technologies (Python, Flask) and machine learning for analysis.

**2. Objective**

The main objective of this project is to build a responsive and interactive application that:

* Allows users to input text.
* Processes the input using a machine learning model.
* Displays the sentiment result with user-friendly visual feedback.

## ****3. Technologies Used****

### **Frontend**

* **HTML**: Provides the structure of the web page.
* **CSS**: Styles the web page and improves user interface aesthetics.
* **JavaScript**: Handles user interactions, sentiment calculations (rule-based fallback), and sends/receives data to/from the server.

### **Backend**

* **Python**: Implements the logic of sentiment prediction using a trained ML model.
* **Flask**: A lightweight web framework that handles HTTP requests and connects the front end with the machine learning backend.

### **Machine Learning**

* **Model Used**: Logistic Regression (or another classification model).
* **Vectorization**: TF-IDF vectorizer converts text into numerical format for model prediction.
* **Libraries**: scikit-learn, joblib, and Flask.

## ****4. Project Structure****

sentiment\_analysis\_project/

│

├── static/

├─── css

│ └──── styles.css # CSS for styling

│ ├───── js

└──── script.js # JavaScript for interactivity

│

├── templates/

│ └── index.html # HTML page served by Flask

│

├── app.py # Flask application

├── Logistic Regression\_model.pkl # Trained ML model

├── tfidf\_vectorizer.pkl # TF-IDF vectorizer

**5. Workflow**

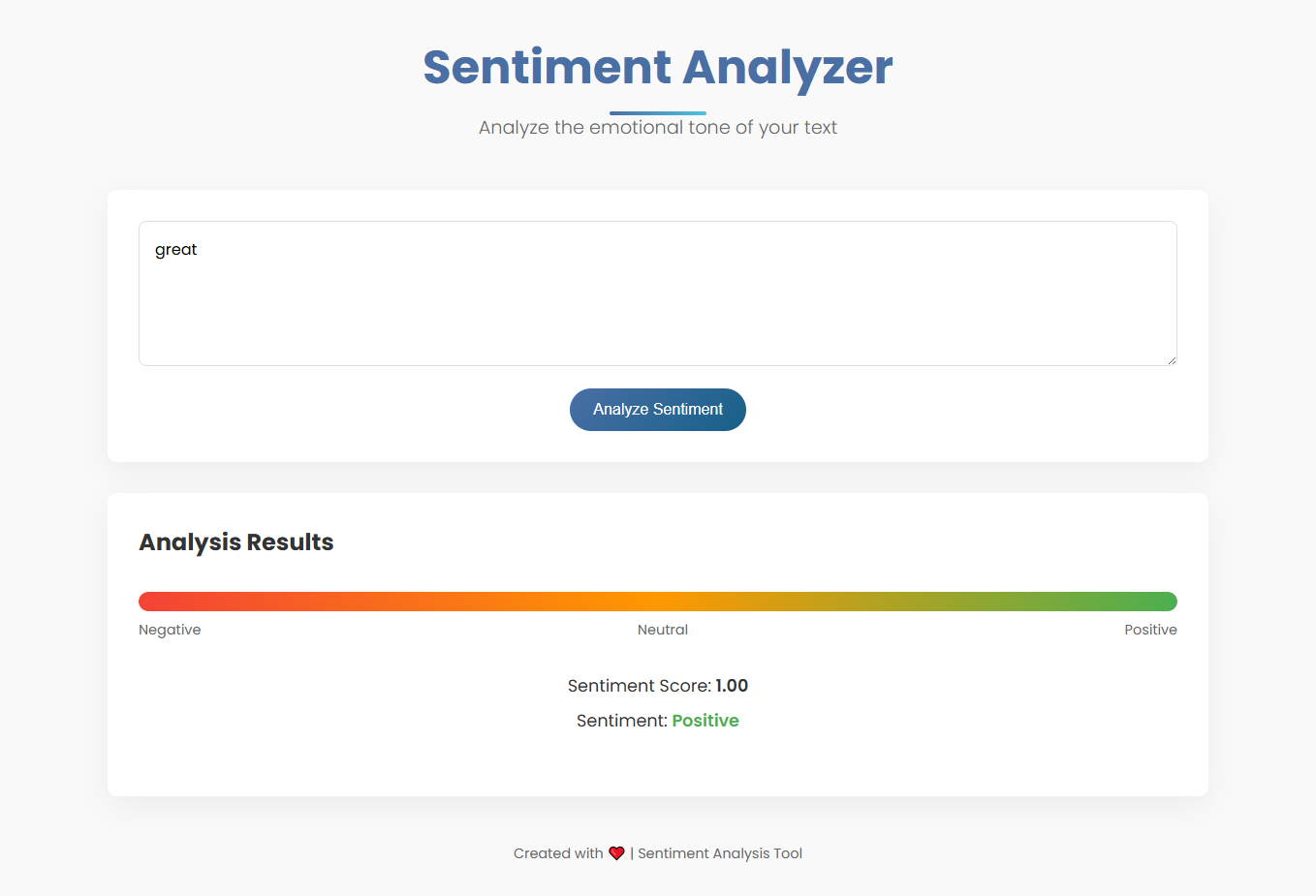
1. **User Interaction**:
   * The user enters a sentence and clicks the **Analyze** button.
   * JavaScript captures the text and sends it to the Flask server via a POST request.
2. **Data Processing**:
   * Flask receives the text and transforms it using the TF-IDF vectorizer.
   * The machine learning model predicts the sentiment class (0, 1, or 2).
3. **Response & Display**:
   * Flask sends the result back to the front end.
   * JavaScript updates the UI dynamically, showing the sentiment label, score, and relevant words.

**6. Conclusion**

This project is about how machine learning models understand emotions behind a piece of text. This application shows that whether text is positive, negative, or neutral. It’s fast, simple, and helpful for analyzing user feedback or social media posts.

**📊 Results:**

| **Classes** |  |  | **Negative** |  |  | **Neutral** |  | **Positive** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **Precision** | **Recall** | **F1-Score** | **Precision** | **Recall** | **F1-Score** |
| Naive Bayes | 70% | 78% | 85% | 82% | 71% | 53% | 60% | 67% | 78% | 72% |
| SVM | 80% | 86% | 94% | 90% | 81% | 72% | 76% | 85% | 88% | 87% |
| Logistic Regression | 80% | 85% | 91% | 88% | 80% | 70% | 74% | 82% | 87% | 85% |
| Random Forest | 90% | 94% | 96% | 95% | 87% | 72% | 79% | 80% | 97% | 85% |
| XG Boost | 90% | 96% | 94% | 95% | 83% | 84% | 83% | 87% | 88% | 87% |

**Screenshot**