Flask API Documentation

Sentiment Prediction API

Purpose of the Flask API

The Flask API is used to **deploy the trained machine learning model** so that users can send review text and receive **sentiment predictions** in real time.

This API:

- Accepts review text as JSON input
- Cleans the text
- Converts it using TF-IDF
- Predicts sentiment using the trained Logistic Regression model
- Returns the result as a JSON response

Technologies Used

- **Flask** – Web framework for API creation
- **Scikit-learn** – Machine learning model
- **Joblib** – Load saved model and vectorizer
- **TF-IDF** – Text vectorization
- **Python** – Backend language

Model and Vectorizer Loading

model = joblib.load("LogisticRegression.pkl")

tfidf = joblib.load("tfidf_vectorizer.pkl")

- **Explanation**
  - Loads the **trained Logistic Regression model**
  - Loads the **TF-IDF vectorizer**
  - Ensures the API can predict without retraining

Text Cleaning Function

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'http\S+|www\S+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text
```

- **Purpose**
  - Converts text to lowercase
  - Removes URLs
  - Removes special characters and numbers
  - Ensures clean input for prediction

API Endpoints

- **Home Endpoint**

URL: /
Method: GET

```
@app.route("/", methods=["GET"])
def home():
    return "<h2>Flask API running!</h2>"
```

- **Purpose**
  - Confirms that the API server is running successfully

Prediction Endpoint

URL: /predict
Method: POST
Content-Type: application/json

Request Format (Input)

```
{
  "name": "Alice",
  "review": "This product is very good and comfortable"
}
```

- **Input Description**
  - name → User name (optional)
  - review → Customer review text (**mandatory**)

Prediction Logic

review_clean = clean_text(review)

review_vec = tfidf.transform([review_clean])

prediction = model.predict(review_vec)[0]

- **Explanation**
  1. Cleans the input review text
  2. Converts text into TF-IDF numerical form
  3. Predicts sentiment using the trained model

Sentiment to Rating Mapping

```
SENTIMENT_MAP = {
   "negative": 0,
   "neutral": 1,
   "positive": 2
}
```

- **Purpose**
  - Converts text sentiment into a numeric rating
  - Makes output easy to understand and use in applications

Response Format (Output)

```
{
  "name": "Alice",
  "review": "This product is very good and comfortable",
  "predicted_sentiment": "positive",
  "predicted_rating": 2
}
```

◇ **Output Description**

- predicted_sentiment → Model prediction
- predicted_rating → Numeric form of sentiment

Error Handling

The API handles common errors such as:

- Missing JSON data
- Empty review text
- Internal server errors

return jsonify({"error": "Review text is empty"}), 400

Benefit

- Prevents API crashes
- Provides meaningful error messages to users

Testing the API

The API is tested using test.py with the **requests** library.

response = requests.post(url, json=payload)

print(response.json())

Purpose

- Sends a POST request to the API
- Displays the predicted sentiment result

How to Run the API

python api.py

API runs at:

http://127.0.0.1:5000