SENTIMENT ANALYSIS

Steps:

1. Data Collection
2. Data Preprocessing
3. Model Training
4. Model Evaluation
5. Deployment using Flask

CODE:

```
import random
import nltk
from nltk.corpus import movie_reviews, stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pickle
from flask import Flask, request, jsonify
```

```
# Download necessary NLTK data
nltk.download('movie_reviews')
nltk.download('stopwords')
nltk.download('punkt')
# Data Collection and Preprocessing
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)
def preprocess(document):
    tokens = word_tokenize(document)
    tokens = [word.lower() for word in tokens if
word.isalpha() and word not in stopwords.words('english')]
    return ' '.join(tokens)
documents = [(' '.join(doc), category) for (doc, category) in
documents]
documents = [(preprocess(doc), category) for (doc,
category) in documents]
# Split into training and testing sets
```

```
train_documents = documents[:1500]
test_documents = documents[1500:]
train_texts, train_labels = zip(*train_documents)
test_texts, test_labels = zip(*test_documents)
label_mapping = {'pos': 1, 'neg': -1}
train_labels = [label_mapping[label] for label in train_labels]
test_labels = [label_mapping[label] for label in test_labels]
# Feature Extraction
vectorizer = TfidfVectorizer(max_features=5000)
train_vectors = vectorizer.fit_transform(train_texts)
test_vectors = vectorizer.transform(test_texts)
# Model Training
model = LogisticRegression(max_iter=1000)
model.fit(train_vectors, train_labels)
# Model Evaluation
test_predictions = model.predict(test_vectors)
accuracy = accuracy_score(test_labels, test_predictions)
print(f'Accuracy: {accuracy}')
# Save the trained model and vectorizer
```

```
with open('sentiment_model.pkl', 'wb') as f:
    pickle.dump(model, f)
with open('vectorizer.pkl', 'wb') as f:
    pickle.dump(vectorizer, f)
# Flask Deployment
app = Flask(__name__)
with open('sentiment_model.pkl', 'rb') as f:
    model = pickle.load(f)
with open('vectorizer.pkl', 'rb') as f:
    vectorizer = pickle.load(f)
@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    review = data.get('review')
    if review:
        preprocessed_review = preprocess(review)
        vectorized_review =
vectorizer.transform([preprocessed_review])
        prediction = model.predict(vectorized_review)[0]
        sentiment = 'positive' if prediction == 1 else 'negative'
        return jsonify({'review': review, 'sentiment':
sentiment})
    else:
        return jsonify({'error': 'No review provided'}), 400
if __name__ == '__main__':
    app.run(debug=True)
```