**# Sentiment Analysis: Text Classification with Flask Using Machine Learning**

**This project implements a sentiment analysis system utilizing machine learning techniques and Flask to create a web application. The primary objective is to classify customer reviews into positive or negative categories.**

**## Prerequisites**

**Python 3.x**

**Required Libraries: pandas, numpy, nltk, sklearn, flask**

**## Functionality**

**### Data Preprocessing:**

**Cleans text data by converting it to lowercase, removing non-alphanumeric characters, eliminating stop words, and applying lemmatization.**

**### Model Training:**

**\* Trains a Random Forest classification model using a labeled dataset of customer reviews.**

**\* Utilizes TF-IDF vectorization to convert text data into numerical features suitable for machine learning algorithms.**

**### Web Application:**

**\* Provides a user-friendly web interface for inputting reviews.**

**\* Analyzes the sentiment of the submitted review using the trained model and displays the result (positive or negative) on the webpage.**

**## Code Structure**

**### 1. Importing Libraries:**

**The necessary libraries are imported, including pandas for data manipulation, numpy for numerical operations, nltk for natural language processing, sklearn for machine learning algorithms and evaluation metrics, and flask for building the web application.**

**### 2. Preprocessing Function:**

**A function named preprocess\_text is defined to clean the review text. It converts the text to lowercase, removes non-alphabetic characters, tokenizes the text into words, eliminates stop words, and lemmatizes the words using the WordNetLemmatizer from NLTK.**

**### 3. Loading and Preprocessing Data:**

**A sample dataset is created containing reviews and their corresponding sentiments. The reviews are preprocessed using the preprocess\_text function, and the results are stored in a new column called cleaned\_review.**

**### 4. Converting Labels to Numeric:**

**The sentiment labels are converted into numeric values using LabelEncoder from sklearn.**

**### 5. Splitting the Data:**

**The preprocessed data is divided into training and testing sets using train\_test\_split from sklearn. The test\_size parameter is set to 0.2, indicating that 20% of the data will be reserved for testing, while the remaining 80% will be used for training.**

**### 6. TF-IDF Vectorization:**

**The cleaned reviews are transformed into a matrix of token counts using TfidfVectorizer from sklearn. This step converts the text data into a format that is suitable for machine learning algorithms.**

**### 7. Training the Random Forest Model:**

**A Random Forest Classifier is instantiated using RandomForestClassifier from sklearn. The classifier is trained on the training data using the fit method.**

**### 8. Evaluating the Model:**

**The trained model is evaluated on the testing data by making predictions. The predicted labels are compared with the actual labels using various evaluation metrics, such as accuracy, precision, recall, and F1-score, which are calculated using functions from sklearn.metrics.**

**### 9. Flask Web Application:**

**A Flask web application is established to expose the sentiment analysis functionality. The app object is created using Flask from the flask library.**

**### 10. HTML Template:**

**An HTML template is defined using triple-quoted strings. The template includes a textarea for entering reviews, a button to trigger the sentiment analysis, and a div to display the predicted sentiment.**

**### 11. Flask Routes:**

**Two Flask routes are defined:**

**The root route (/) renders the HTML template using render\_template\_string.**

**The /predict route accepts POST requests containing a review in JSON format, preprocesses the review, and returns the predicted sentiment using jsonify.**

**### 12. Running the Flask Application:**

**The sentiment analysis system is exposed as a web application using Flask. The Flask app is executed in debug mode using app.run(debug=True) if the script is run directly (not imported as a module).**

**### Summary**

**This code demonstrates a comprehensive sentiment analysis project that trains a Random Forest Classifier on a sample dataset, evaluates its performance, and exposes the functionality through a Flask web application. Users can input reviews, and the application will predict the sentiment (positive or negative) of the provided text.**

**## How to Use**

**1. Run the script: python app.py**

**2. Access the web application in your browser at: http://127.0.0.1:5000/**

**3. Enter a review in the provided text area.**

**4. Click the "Analyze Sentiment" button.**

**5. The predicted sentiment (positive or negative) will be displayed below the text area.**