Hate Speech Detection Using NLP and Deep Learning

**Introduction:**

This project implements a machine learning pipeline to detect hate speech in text data. Using natural language processing (NLP) techniques and a Long Short-Term Memory (LSTM) model, the project classifies text into hate speech or non-hate speech categories. The system is built using Python, and the trained model is deployed using Streamlit for a user-friendly interface.

**Technical Stack:**

#### **Programming Language : Python**

#### **Libraries and Frameworks**

* **Data Preprocessing & Analysis**:
  + **Pandas** : For handling datasets and performing data cleaning.
  + **NumPy** : For numerical computations and array manipulations.
  + **NLTK**: For natural language processing, including stopword removal and tokenization.
  + **re** : For regular expression-based text cleaning.
* **Visualization**:
  + **Seaborn**: For data visualization to analyze dataset distribution.
* **Deep Learning**:
  + **TensorFlow/Keras** : To build, train, and evaluate the LSTM model.
* **Model Deployment**:
  + **Streamlit** : For creating a user-friendly web application interface for real-time predictions.

#### **Tools**

* **Jupyter Notebook**: For model experimentation and development.
* **Anaconda**: For managing the Python environment and dependencies.
* **GitHub**: For version control and code sharing.

#### **File Formats**

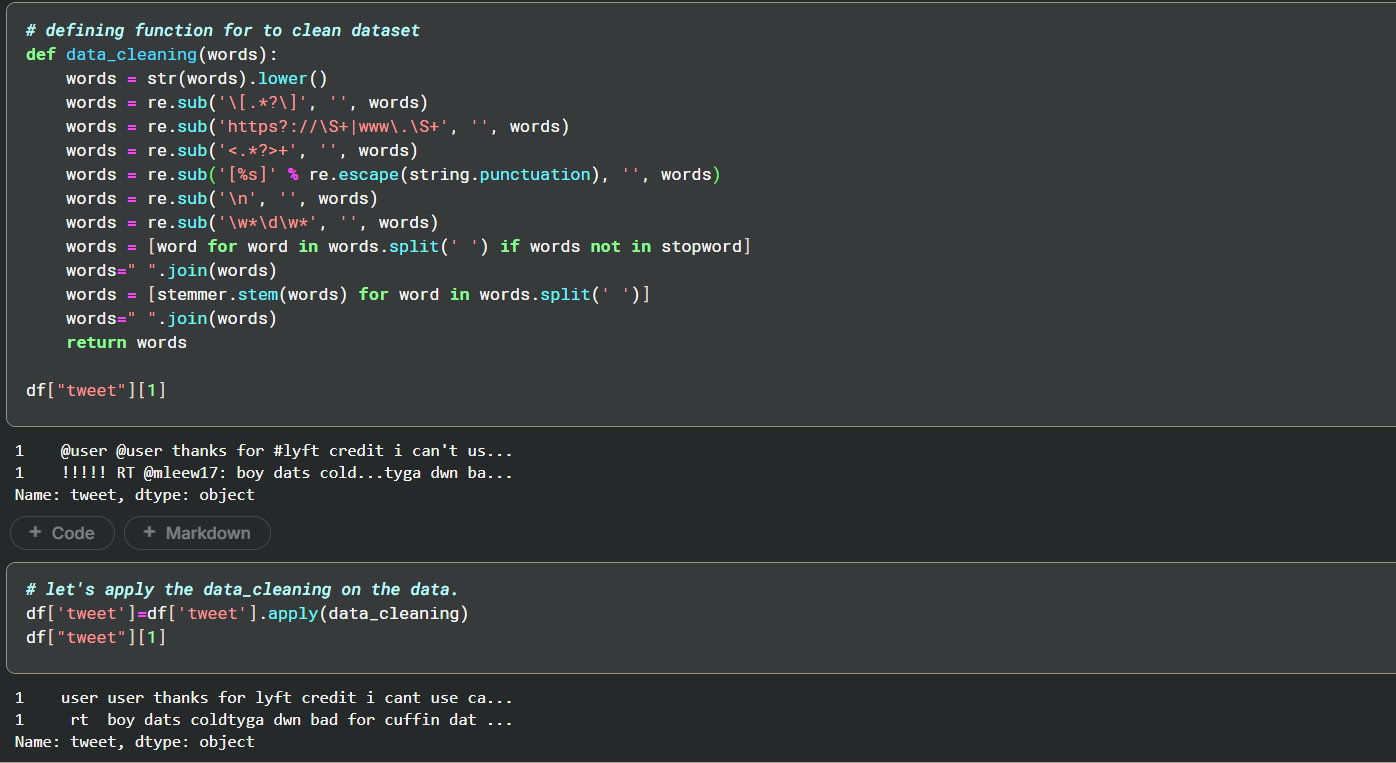
* **.csv** : For dataset storage and manipulation.
* **.h5** : For saving the trained model.
* **.pickle** : For saving the tokenizer.

**Features**

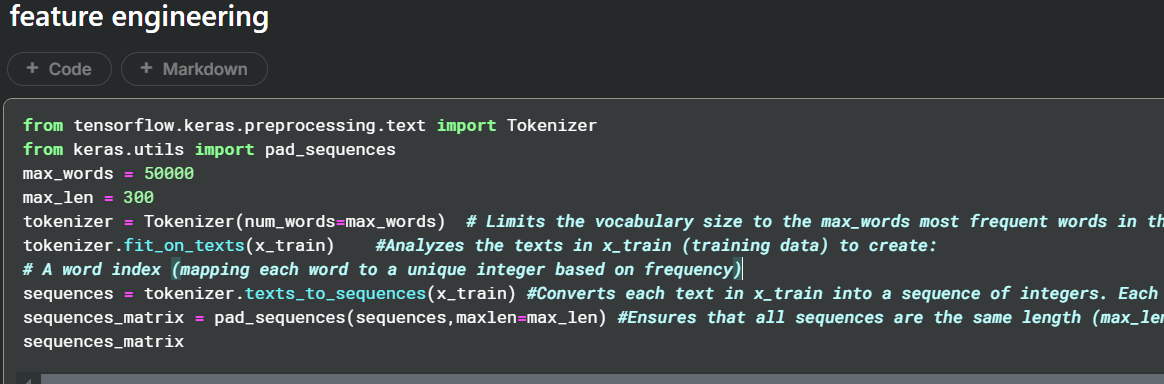
1. **Preprocessed and balanced dataset** using label transformation and concatenation.
2. **Custom text cleaning function** for removing unwanted patterns, punctuation, and stopwords.
3. **Deep learning-based classification** with an LSTM network.
4. **Deployed application** with a clean UI for real-time predictions.
5. Model storage and reusability using **.h5 format** and tokenizer serialization.

### **Project Workflow :**

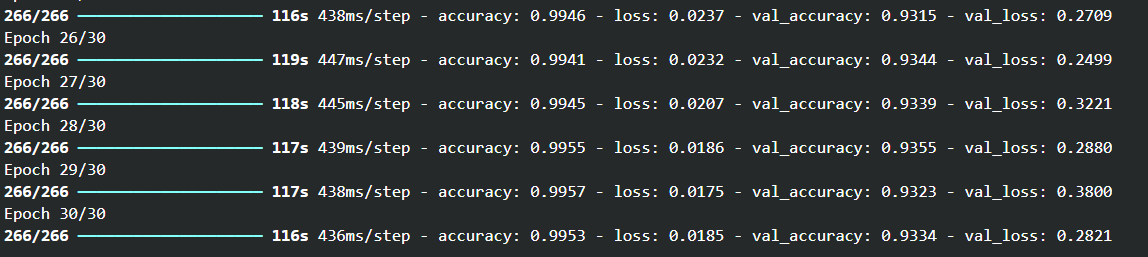
#### 1. **Data Preprocessing**

* **Datasets**:
  + Imbalanced dataset: imbalanced\_data.csv
  + Raw dataset: raw\_data.csv
* **Steps**:
  + Dropped unnecessary columns like id, Unnamed: 0, and others.
  + Transformed class labels: merged values to ensure binary classification.
  + Cleaned text data using regular expressions, stopword removal, and stemming. 

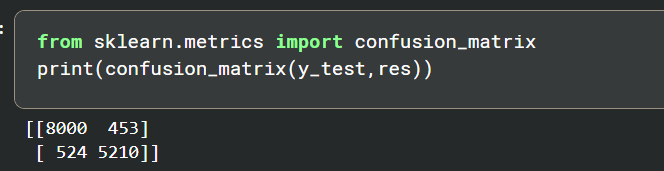
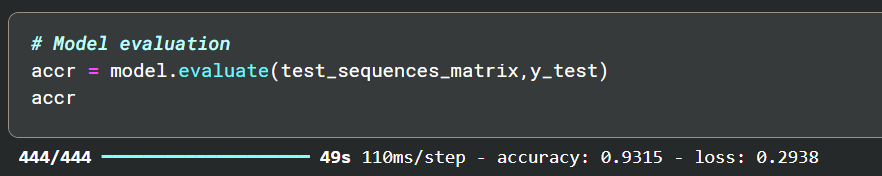
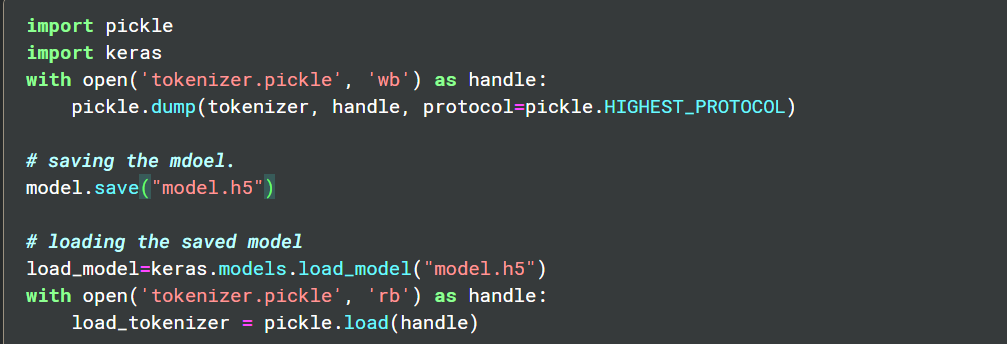
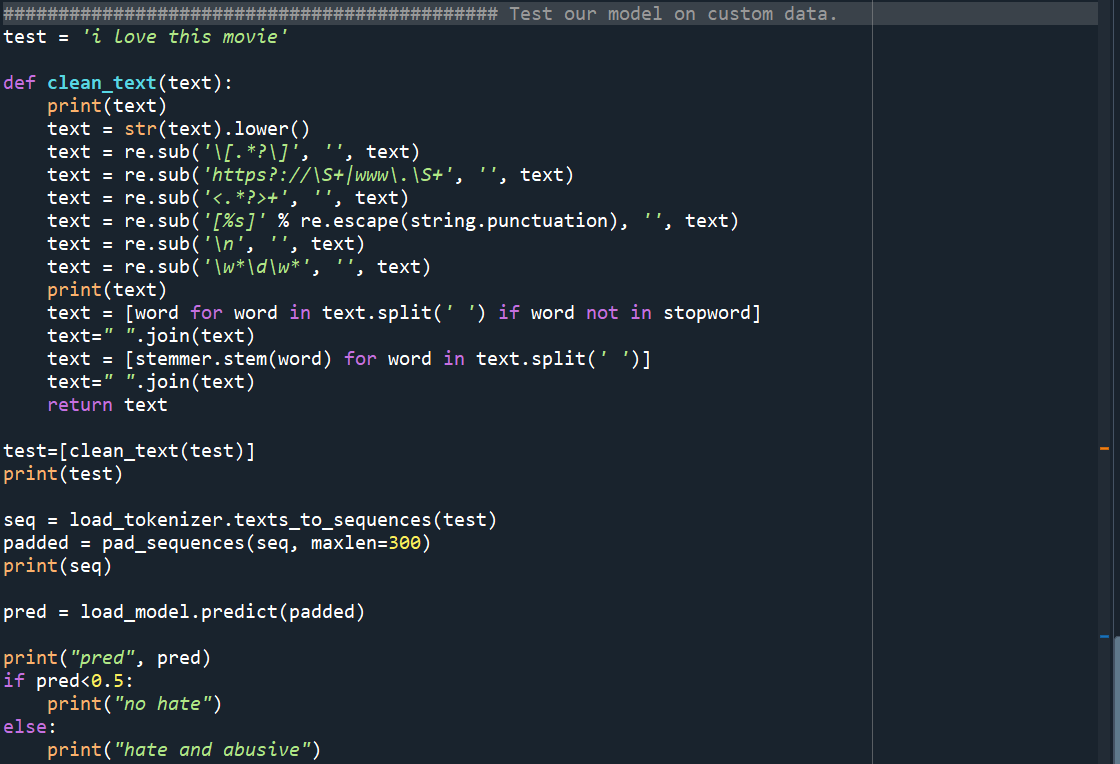
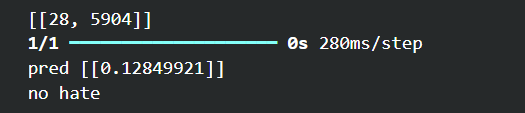
#### 2. **Feature Engineering**

* Tokenization using Keras tokenizer.
* Sequence padding to standardize text input length. 

#### 3. **Model Architecture**

* A sequential LSTM model with the following structure:
  + Embedding layer for text vectorization.
  + Spatial Dropout for regularization.
  + LSTM layer for capturing sequential dependencies.
  + Dense output layer with sigmoid activation for binary classification. 

#### 4. **Model Evaluation**

* Split the dataset into training and testing sets.
* Achieved accuracy and evaluated performance using a confusion matrix. 
* saving model and tokens in pickle and .h5 formate and loading the saved model for further use 
* Testing the model on unseen data/test data  

#### 5. **Deployment**

* Deployed using **Streamlit**, allowing real-time text input and prediction. 