**Sentiment Analysis Model using IMDB Dataset**

**Introduction**

In this NLP learning task, the primary goal is to create and implement a sentiment analysis model utilizing the IMDB dataset. Sentiment analysis, a branch of natural language processing (NLP), revolves around the classification of movie reviews into positive or negative sentiments based on their textual content. The overall development process encompasses various stages, spanning from collecting the necessary data to deploying the finalized model. This journey offers a holistic exploration of NLP tasks, involving the manipulation of text data, training of machine learning models, and the creation of a user-friendly interface for practical deployment.

**Gather the Dataset (IMDB Dataset)**

The IMDB dataset comprises 50,000 rows and 2 columns, forming a robust foundation for sentiment analysis model development. Each row corresponds to a movie review, while the two columns include the 'review' column containing the text of the reviews and the 'sentiment' column providing binary labels, denoting either positive or negative sentiment. This extensive dataset captures a diverse range of opinions, enabling the training and evaluation of the sentiment analysis model on a substantial and representative sample of movie reviews. The large scale of the dataset ensures a comprehensive exploration of language nuances and sentiment expressions in the context of film critiques.

**Exploratory Data Analysis (EDA)**

In the process of Exploratory Data Analysis (EDA), an investigation into the dataset's sentiment label distribution was conducted. By assessing the count of labels in each target class (positive and negative), the balance of the dataset was determined. The dataset displays a balance between positive and negative sentiments, with a roughly equal distribution. This equilibrium is advantageous for training the sentiment analysis model, as it ensures exposure to a diverse array of opinions, contributing to a more comprehensive and unbiased model performance during both training and evaluation.

**Preprocess the Text**

Text preprocessing is a crucial step in preparing textual data for analysis. In this context, the following techniques were employed:

1. Remove Missing Values: Any instances of missing data were eliminated to ensure a clean dataset, as missing values can hinder model training.

2. Lowercase: All text was converted to lowercase. This normalization step ensures uniformity, preventing the model from treating uppercase and lowercase versions of the same word differently.

3. Remove Special Characters and Numbers: Extraneous characters and numerical digits were excluded. This helps focus the model on the textual content by eliminating irrelevant symbols and numbers.

4. HTML Tags Removal: If present, HTML tags were stripped from the text. This is essential when dealing with web-based datasets to eliminate any HTML-specific elements.

5. Expanding Contradictions: Expressions like "isn't" were expanded to their full form ("is not"). This aids in maintaining the integrity of words and enhances the model's understanding of negations.

6. Tokenization: Tokenization involves breaking down text into individual words or tokens. This process facilitates the model's analysis by providing a structured representation of the text.

7. Stemming and Lemmatization: Both are techniques to reduce words to their base or root form. While stemming is more aggressive and involves removing prefixes or suffixes, lemmatization considers the word's context and transforms it into its base form. For this task, lemmatization was chosen to preserve the context and improve the model's comprehension.

8. Stop Words Removal: Stop words are common words (e.g., "the," "and," "is") that may not contribute significantly to the meaning of a sentence. Removing these words helps focus the model on more meaningful content.

**Embedding of Text**

Embedding of text is like translating words into a special language that computers easily understand. Each word gets its unique code a set of numbers that represents its meaning. This helps the computer make sense of words and their relationships in a way that's useful for analyzing text.

**Count Vectorization**:

Count Vectorization is like counting how many times each word appears in a bunch of sentences. Imagine making a list of all the words and marking how many times each one shows up. This list becomes a set of numbers, showing the frequency of words. It's a simple way for the computer to understand the words in our text.

**TF-IDF (Apply TF-IDF on the count vectorized data):**

TF-IDF is a smart way of looking at words. It not only counts how often a word appears in one sentence but also considers how unique and important it is in all the sentences. It's like giving more weight to words that are special and less weight to common ones. This helps the computer focus on what matters when understanding and analyzing text.

**Train Test Split**

The dataset was split into training and testing sets, ensuring distinct subsets for model training and evaluation. The standard 70-30 split was applied, providing a robust foundation for assessing model performance

**Train Classical Machine Learning Models**

Classical machine learning models, including Logistic Regression, Decision Tree, SVM, Naive Bayes, and K-Nearest Neighbors, were implemented. Training logs showcased the iterative learning process, with each model capturing different aspects of the data.

**Evaluate Models**

The model evaluation involved assessing performance on the test dataset. Classification reports provided a detailed overview of model precision, recall, accuracy, F1-score, and AUC-ROC score, aiding in the selection of the best-performing model.

**Model Selection and Saving**

Analyzing the evaluation metrics led to the selection of the best-performing model. The chosen model was saved for future use, ensuring reproducibility and efficiency.

**Backend Development**

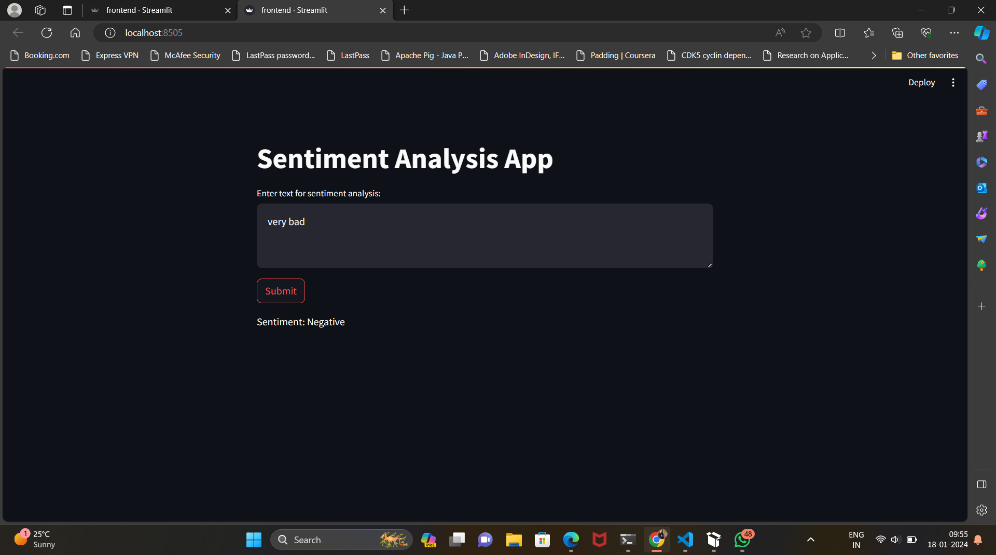
Backend development involved creating functions for loading the model and preprocessing data. These functions were organized within a class for modularity and ease of use.

**Frontend Development (Streamlit)**

The frontend was developed using Streamlit, creating a user-friendly interface. Users could input text, and upon clicking submit, the text was processed by the backend, with results displayed on the website.

**Deployment: Deploy the website locally**

After building and training our sentiment analysis model, the next step is deployment, which means making our model accessible for others to use. In this case, we focused on deploying the model through a website locally, meaning it can be accessed on a personal computer.



**Conclusion**

In the culmination of this NLP learning task, a comprehensive journey was undertaken, navigating through the multifaceted stages of sentiment analysis model development. The success of this endeavor stems from adeptly comprehending the intricacies of the dataset, implementing robust text preprocessing techniques, and training diverse machine learning models. The pivotal deployment phase, where a user-friendly interface was created using Streamlit, showcased the application of acquired knowledge in a practical setting. This holistic approach illuminated the interconnected nature of data exploration, model training, and deployment, emphasizing the need for a unified strategy for ineffective sentiment analysis. Through this task, a profound familiarity with fundamental NLP concepts and tools was cultivated, laying a solid foundation for continued exploration and refinement of natural language processing techniques. This experiential journey not only enhanced technical proficiency but also instilled a deeper appreciation for the nuanced art of deciphering sentiments from textual data.