A) Problem statement

To accurately classify film reviews as positive or negative with the use of Deep Learning Layers such as LSTM (Long short-term memory).

B) EDA and Data Pre-processing steps taken

→ Since our IMDb dataset has no null values, the initial step was to process all the text data to make it easier for the model to handle a rather large and wordy dataset.

→ To make it easier for the model to recognize patterns, all of the text data are shifted to lowercase. Characters other than alphabets are removed using regular expression commands. Word_tokenize function is then used to create "tokens" for each word. Lemmatizing these tokens reduces each word to its root form (running : run).

→ After simplifying the text data, the distribution of positive and negative sentiments are visualized using count plots. Also, the most frequent words are viewed with the help of word clouds.

→ The sentiment column is converted from categorical to numerical form (0's and 1's).

→ The processed text data is further tokenized into integers with the help of tokenizer. Only the top 10,000 frequently occurring words are considered for our sequential data. The data is also padded to a fixed length of 200 tokens per input, which is essential before training the model.

C) Model Training and Evaluation

→ A train-test split of 80-20 is designated for our sequential model. Also, a specific seed is set for data split so that our model learns more efficiently.

→ A sequential model with an embedding layer (for converting input into high dimensional vectors), two stacked LSTM's (for helping the model to understand both short and long term dependencies), dropout layer (for avoiding overfitting) and a dense layer (output and activation layer).

→ The embedding layer is responsible for creating semantic relationships between different set of words whereas the dense layer produces one singular output using a probabilistic function called sigmoid that is used in binary classification.

→ Binary crossentropy loss function is selected due to duality nature of the classification problem. Adam was chosen as the optimizer owing to its adaptive learning rates.

→ A early_stopping function is passed into our model to make sure that the best weights and biases are captured in case of overfitting across different epoch cycles.

D) Challenges faced

→ Passing the stopwords text database inside of clean_text function yielded a lot of computational cost since the complete database had to be summoned each time the function was called. Storing this as a global variable and assigning it to a set made sure that data fetching became faster (set datatypes support hashing and don't store duplicates).

→ Mapping categorical to numerical data was difficult since the values in sentiment column were formatted unevenly.

→ To capture the model with best weights and biases, I had to create a separate callback function to make sure that the most efficient model is retained at the end of training.

E) Summary

Deep Learning Algorithm used: Sequential / Stacked LSTM model

Accuracy: 88.79%

Precision: 89%

f1-score:89%

Key library methods in use →

- ✓ Keras: Sequential, Embedding, LSTM, Dense, Dropout, Tokenizer, pad_sequences, Early_stopping
- ✓ NLTK: word_tokenize, WordNetLemmatizer
- ✓ Wordcloud: WordCloud