1. Loading and Preprocessing of Data

The First Step is to load the dataset in the pandas dataframe format for ease of doing the further analysis. As the dataset is in the text file format in the respective Train and Test folder, each file need to read and stored in a dataframe format. In order to achieve that a load function is used to convert the text file to pandas DataFrame format. In the second step, data cleaning and Preprocessing is required. The data cleaning is required because the raw text file contains HTML tags along with other non-letter symbols which are not usefull for Sentiment Analysis. In addition, the datasets also contains emojis which are helpfull for Sentiment analysis and will be usefull in this case and so not all of them are removed. In order to preprocess the dataset Python Regex library is used to clean the dataset. Word Capitals are also converted to lower case as capitalization of words doesn't contain much semantic information with exceptions in some cases.

2. Tokenization and Padding for the Text Data

In order for us to train any model to learn the sentiments we need to convert the words to Vectors in a Token Format. In the First step, Tokenization is performed and it is done using the Keras Preprocessing Library called Tokenizer it takes the text data and applies the texts_to_sequence method and then the Tokens are given out by the function. For each of the word the Tokenizer gives out a unique Integer to represent it. These are then converted into Input features and given to the Neural Network which is discussed below. This is done for both the Test data and the Train data. In the second step Padding is performed on the Tokens generated by the Tokenizer funtion and it mainly padds the data to match the lenght for each of the samples as the lenght of the samples vary and it needs to have the same lenght for each of the input to be given to the Neural Network.

3. Model Design for Sentiment Analysis

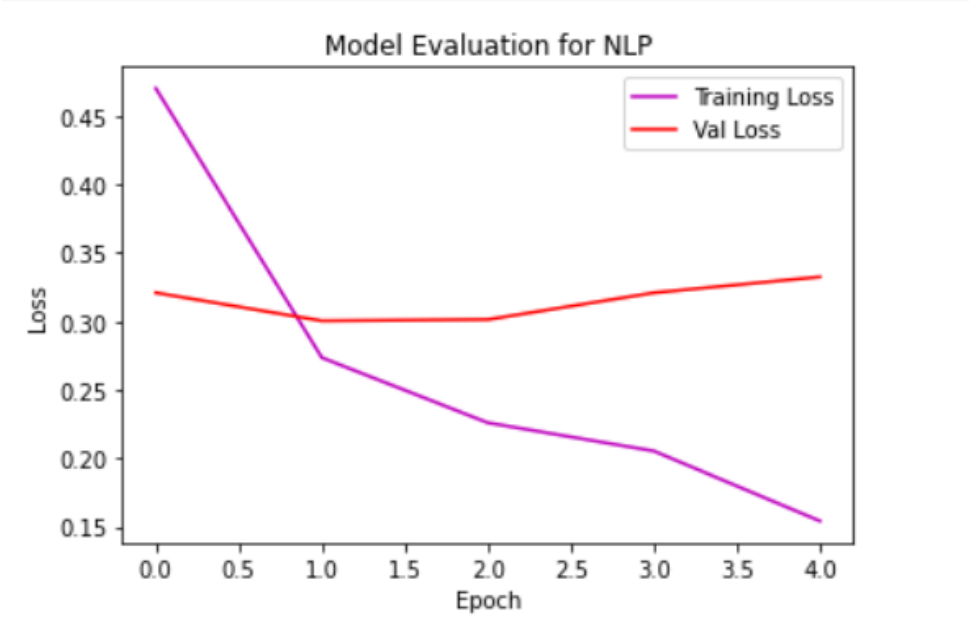
In order to obtain a good accuracy score on the IMDB movie dataset Long Short Term Memory (LSTM) Neural Network along with one Embedding layer is is used. The main aim of Embedding layer is to automatically learn the sailent features to represent the words for the dataset. In addition, one important condition here to keep in mind is that the embedding dimension should be very small than number of unique words in the dataset. The condition is represented as embedding_size << unique_words. For curiosity purpose the above method uses unique words to represent the features from the text data. It is obvious to think that the Embedding layer can be replaced by one hot encoding however it is not very efficient for large or very large dataset as the encoding values becomes sparse. In addition, there are advantages of using Embedding layer from that of One hot Encoding. The firs advantage is that the Embedding layer reduces the dimensionality of feature space to decrease the overall effect of the curse of dimensionality for the dataset as they are large. The second advantage is that the extracted sailent features from the dataset can used by the Neural Networks to learn it. After the Embedding Layer we use a layer of LSTM with 100 units in it. The number of units used is obtained by trail and error after running several iterations by keeping in mind that the Neural Network doesn't overfit. The last layer is a dense layer for representing the output of the Neural Network. As the output is either 1 (Positive Review) or 0 (Negative Review) it can be represented by an individual Neuron. In addition, for trainig the Neural Network a batch size of 64 is used for Training for 5 Epochs. This is the Neural Network for Sentiment Analysis.

4. Output Disscusion

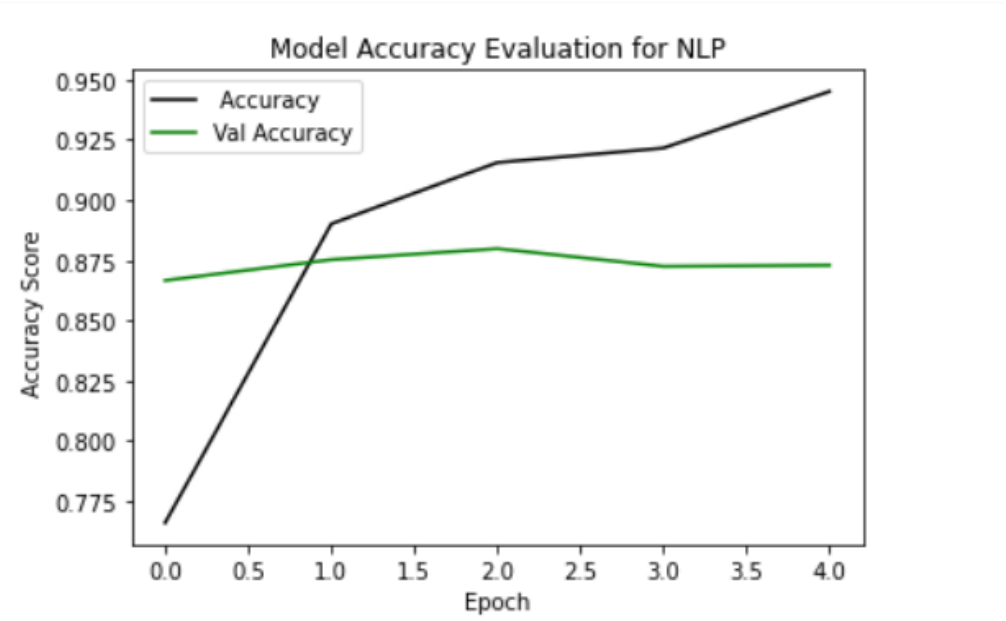
The loss value is shown below

```
Epoch 1/5
25000/25000 [==============================] - 264s 11ms/step - loss: 0.5014 - accura
cy: 0.7469 - val_loss: 0.3471 - val_accuracy: 0.8534
Epoch 2/5
25000/25000 [==============================] - 251s 10ms/step - loss: 0.2956 - accura
cy: 0.8808 - val_loss: 0.3051 - val_accuracy: 0.8758
Epoch 3/5
25000/25000 [==============================] - 242s 10ms/step - loss: 0.2327 - accura
cy: 0.9122 - val_loss: 0.3150 - val_accuracy: 0.8697
Epoch 4/5
25000/25000 [==============================] - 235s 9ms/step - loss: 0.1831 - accurac
y: 0.9343 - val_loss: 0.3135 - val_accuracy: 0.8833
Epoch 5/5
25000/25000 [==============================] - 256s 10ms/step - loss: 0.1527 - accura
cy: 0.9463 - val_loss: 0.3351 - val_accuracy: 0.8770
```

As we can see that the loss value for Training is decreasing and this value reaches a low of 0.1527 at Epoch 5 and the Training Accuracy of 94.63% is achieved. The Validation loss decreases from 0.3471 in Epoch 1 to 0.3051 at Epoch 2 and then slightly inceases after this. At Epoch 5 a value of 0.3351 is achieved which is lower than that of Epoch 1. The final Validation Accuracy is 87.70%. The plots for Validation loss and Trainig loss is shown in the Figure below:



The plot for Validation Accuracy and Trainig Accurayc is shown below:



From the above plot it is clear that the Accracy Score for Sentiment Analysis on Training set is lower when Compared to Validation set. However, the Accuracy Score obatined by the model on the Test Set which is unseen is 87.696% which is a Satisfactory result by the model.