Natural Language Processing

Programming Assignment-1

Group Members:-

Deepeanshi Jindal(**M20MA202**)

Sushil Kumar Lakhiwal(**M20MA205**)

Dataset details:-

For our task we choose the dataset of Movie rating dataset(IMDB dataset) which contains 50,000 movie reviews in it.Which is the task of Binary Class Classification.
This dataset contains two labels(Class) for sentiment analysis as Positive and Negative reviews.

Task- 1 and 2 :

Preprocessing of the dataset:

We have taken 30,000 data points as taking whole dataset is very computationally expensive and memory crashing issues was occurring in the system.
Now we started preprocessing there is no null values in the dataset although there were some duplicate values in the dataset so we remove all the duplicate values after removing there were 29854 data points after this we have made a preprocessing_text function in which we done tokenization of the text, removing stopwords, converting tokens into lowercase format, removing punctuation marks in the data and in the end removing empty tokens from the data and joined the tokens back.
We imported label encoder to convert string data labels into numerical values of like 0 and 1.

Train_val_test split:-

Then we have defined train validation and test split by taking 20% data as test data and 25% data as validation data and the rest data 55% for training purposes.

Task-3:

Developing ML Models:

Now we started the third part of our assignment in which we first use the count vectorizer class from the sklearn library to convert text data into numerical expression that can be use by ML algo. We have train this on training data then transformed train,test and validation data from string to numerical values.
Then we have trained Naive Bayes classifier on training data and found and printed its accuracy, precision, F1-score and Confusion Matrix and Recall on test and validation data.

After it we use the TF-IDF to convert the text data into numerical representations and we transform train, test and validation data into numerical values.
Then we apply Naive Bayes classifier on train, test and validation and calculate accuracy, precision, F1-score and Confusion Matrix and Recall on test and validation data.
After training Naive bayes classifier, we have trained Decision Tree classifier on training data and printed its accuracy, precision, F1-score and Confusion Matrix and Recall on test and validation data.
All the observations are given below that can be seen from there.

NAIVE BAYES Classifier Accuracies:

The following is Validation data Accuracy with Count Vectorizer:-

```
Accuracy: 0.8520080321285141
Precision: 0.8548320199087516
Recall: 0.8419117647058824
F1 score: 0.8483227001440626
Confusion matrix:
 [[2182  350]
```

```
 [ 387 2061]]
```

The following is Test data Accuracy with Count Vectorizer:-

```
Accuracy: 0.8598393574297188
Precision: 0.870887130362349
Recall: 0.8434852763210973
F1 score: 0.8569672131147541
Confusion matrix:
 [[2191  310]
 [ 388 2091]]
```

Colab Link:-

https://colab.research.google.com/drive/1Lb4YWpY-DwBxK7Wo-g5jFNMizmc9MvUF?usp=sharing

The following is validation data accuracy with TF-IDF:

```
Accuracy: 0.8630522088353414
Precision: 0.8621821164889254
Recall: 0.8586601307189542
F1 score: 0.8604175194433074
Confusion matrix:
 [[2196  336]
 [ 346 2102]]
```

The Following is Test Accuracy with TF-IDF:

```
Accuracy: 0.8678714859437751
Precision: 0.8723926380368098
Recall: 0.8604275917708754
F1 score: 0.8663688058489033
Confusion matrix:
 [[2189  312]
 [ 346 2133]]
```

Colab Link:-

https://colab.research.google.com/drive/1zX1ZfFTOeepxDNPV_FhwxMTFOYIDgBdN?usp=sharing

Decision Tree Classifier Accuracies:-

The following is validation data accuracy with TF-IDF:

```
Accuracy: 0.6991978609625669
Precision: 0.6998626373626373
Recall: 0.6875843454790823
F1 score: 0.6936691626957113
Confusion matrix:
 [[1073  437]
 [ 463 1019]]
```

The following is Test data accuracy with TF-IDF:

```
Accuracy: 0.6829268292682927
Precision: 0.7058823529411765
Recall: 0.6606217616580311
F1 score: 0.6825025092004016
Confusion matrix:
 [[1024  425]
 [ 524 1020]]
```

Google Colab Link:-

https://colab.research.google.com/drive/1yRnOpfDGEqLo_RWZRTyEg62oXowGnk66?usp=sharing

Deep Neural Networks:-

HyperParameters:-

learning _rate=0.001

Loss=Binary Cross Entropy

Epochs around 10 with earlystopping

TAsk-4 :

RNN Model:

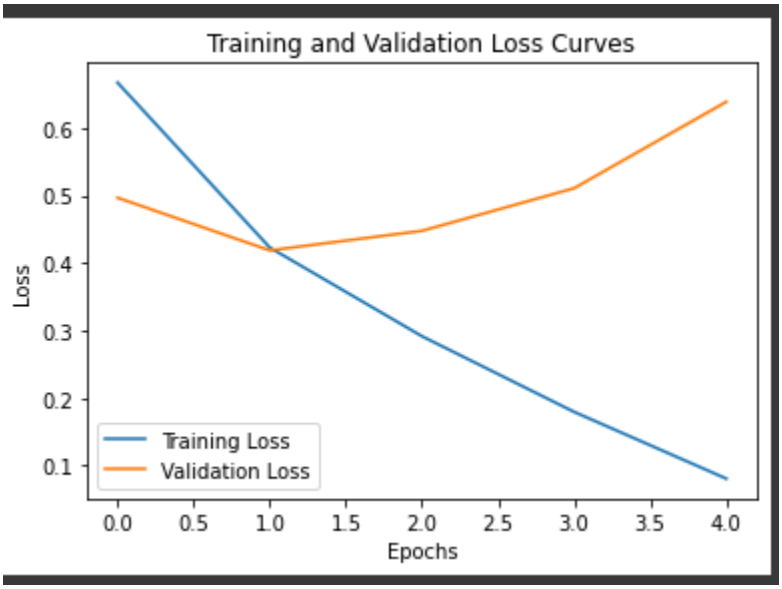
After the preprocessing we first use one hot encoding technique to convert into textual reviews from the data into a numerical expression. we have taken vocab size=5000. Then we have use sequence.pad_sequences() function to pad each review in the reviews list to length 100.

Then we have split the data into train , test and validation set taking 30% testing set 25% validation set and rest for training then we have trained a simple RNN model for sentiment analysis using keras library the model architecture consists of embedding layer that maps each integer encoded word to a dense vector of a embedding dimension a simple RNN layer with 64 hidden nodes and we use dropout layer to prevent the model from overfitting and the dense layer with single output node.And EarlyStopping callback is added to stop training when the validation loss does not improve till 3 more epochs.

We have taken binary cross entropy as loss function, Sigmoid as activation function and Adam optimizer with embedding dimension=32.

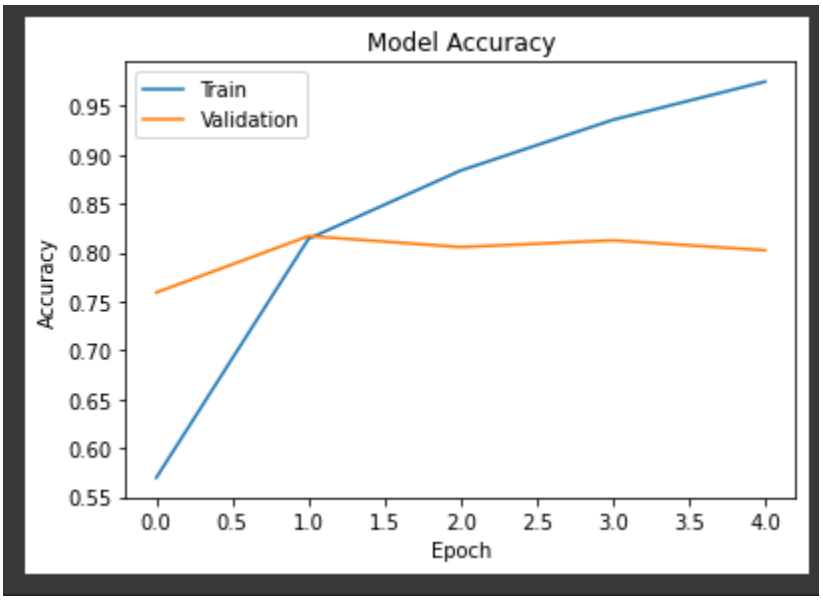
And now we have found validation and training loss and accuracy on each epoch till EarlyStopping and printed accuracy,precision, recall, F1-Score, confusion Matrix on test data And the graphs of loss and accuracy and the observations are given below-

```
Epoch 1/10
245/245 [==============================] - 14s 50ms/step - loss: 0.6673 - accuracy:
0.5697 - val_loss: 0.4968 - val_accuracy: 0.7592
Epoch 2/10
245/245 [==============================] - 11s 46ms/step - loss: 0.2918 - accuracy:
0.8841 - val_loss: 0.4478 - val_accuracy: 0.8057
Epoch 4/10
245/245 [==============================] - 10s 40ms/step - loss: 0.1797 - accuracy:
0.9360 - val_loss: 0.5109 - val_accuracy: 0.8126
Epoch 5/10
245/245 [==============================] - 12s 48ms/step - loss: 0.0809 - accuracy:
0.9751 - val_loss: 0.6389 - val_accuracy: 0.8025
Epoch 5: early stopping
280/280 [==============================] - 3s 8ms/step
Test Accuracy:  0.8060734620966842
```



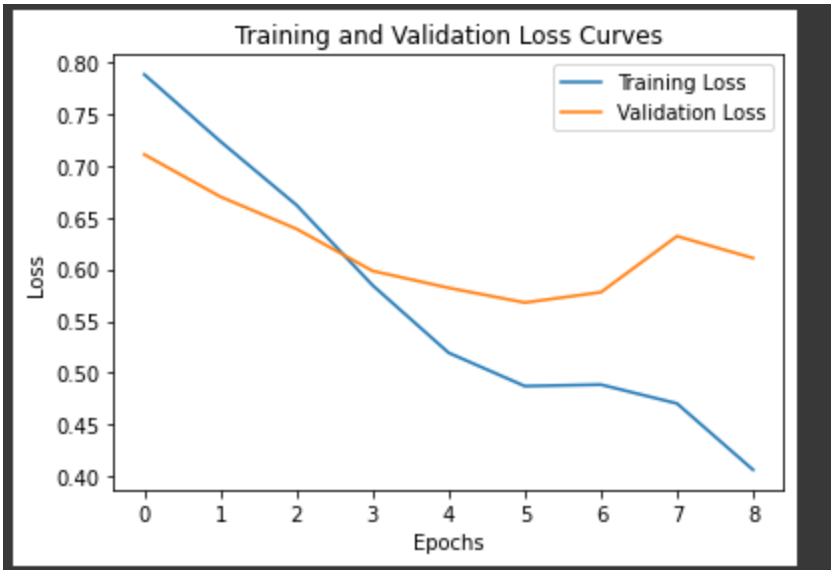
Observations on the test data:-

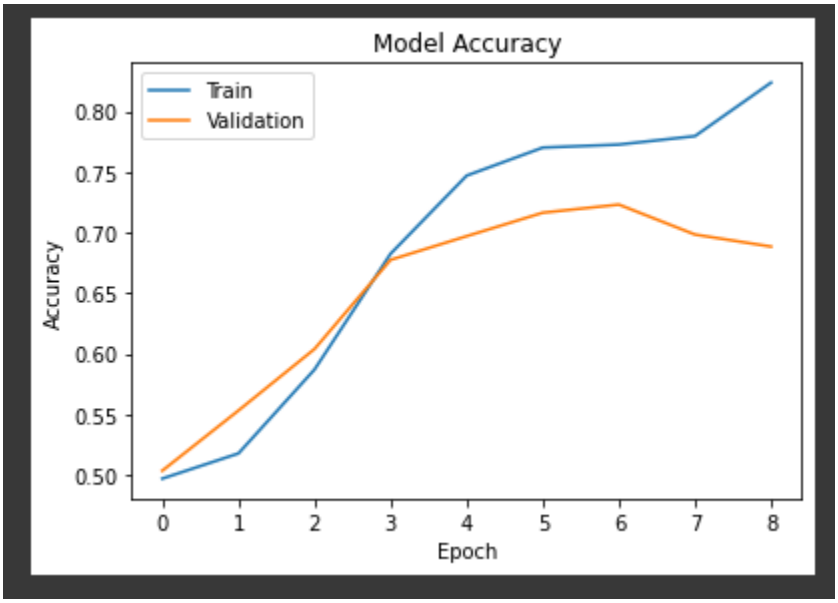
```
Accuracy: 0.8060734620966842
Precision: 0.7965554359526372
Recall: 0.8236865538735529
F1 score: 0.8098938382401225
Confusion matrix:
 [[3520  945]
 [ 792 3700]]
```



Now we have trained the Same model on 256 hidden nodes and the observations and graphs are as follows:-

```
Epoch 1/15
245/245 [==============================] - 57s 220ms/step - loss: 0.7882 - accuracy:
0.4972 - val_loss: 0.7109 - val_accuracy: 0.5037
Epoch 2/15
245/245 [==============================] - 49s 201ms/step - loss: 0.7235 - accuracy:
0.5179 - val_loss: 0.6701 - val_accuracy: 0.5533
Epoch 3/15
245/245 [==============================] - 52s 211ms/step - loss: 0.6620 - accuracy:
0.5871 - val_loss: 0.6390 - val_accuracy: 0.6040
Epoch 4/15
245/245 [==============================] - 49s 202ms/step - loss: 0.5845 - accuracy:
0.6826 - val_loss: 0.5984 - val_accuracy: 0.6775
Epoch 5/15
245/245 [==============================] - 53s 214ms/step - loss: 0.5193 - accuracy:
0.7469 - val_loss: 0.5820 - val_accuracy: 0.6970
Epoch 6/15
245/245 [==============================] - 47s 192ms/step - loss: 0.4872 - accuracy:
0.7700 - val_loss: 0.5678 - val_accuracy: 0.7164
Epoch 7/15
245/245 [==============================] - 53s 217ms/step - loss: 0.4886 - accuracy:
0.7725 - val_loss: 0.5778 - val_accuracy: 0.7231
Epoch 8/15
245/245 [==============================] - 48s 197ms/step - loss: 0.4704 - accuracy:
0.7795 - val_loss: 0.6320 - val_accuracy: 0.6984
Epoch 9/15
245/245 [==============================] - 53s 218ms/step - loss: 0.4063 - accuracy:
0.8236 - val_loss: 0.6108 - val_accuracy: 0.6884
Epoch 9: early stopping
280/280 [==============================] - 12s 43ms/step
Test Accuracy:  0.6844925756391649
Precision: 0.6854407836153161
Recall: 0.6854407836153161
F1 score: 0.6854407836153161
Confusion matrix:
 [[3052 1413]
 [1413 3079]]
```





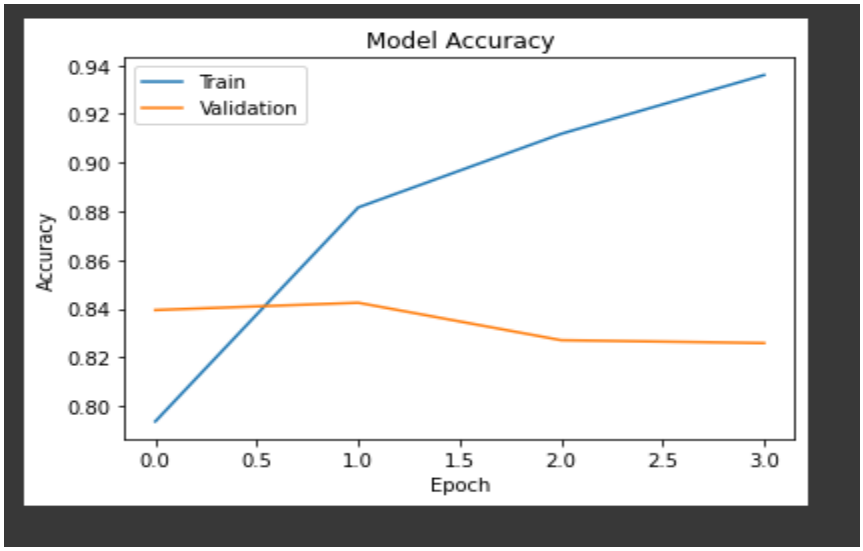
Colab link:-

https://colab.research.google.com/drive/19duJmA-94eZfJZ77HpD5mD92cpsMPkje?usp=sharing

We found that RNN model with 64 hidden nodes is giving good accuracy in compared to RNN with 256 hidden nodes.so we have used 64 hidden nodes for further coming models.

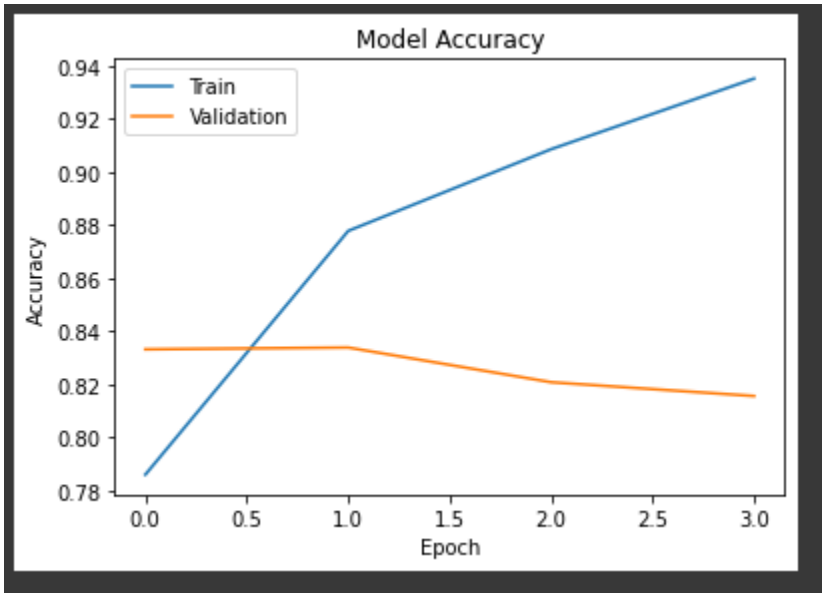
4(b):-In LSTM we have used 20% test data ,25% validation data and rest for training.Now we have trained LSTM model with 1 layer and with same architecture we followed for RNN and the observations and plots are as follows:-

```
Epoch 1/10
560/560 [==============================] - 44s 70ms/step - loss: 0.4362 - accuracy:
0.7936 - val_loss: 0.3702 - val_accuracy: 0.8394
Epoch 2/10
560/560 [==============================] - 40s 72ms/step - loss: 0.2899 - accuracy:
0.8816 - val_loss: 0.3713 - val_accuracy: 0.8424
Epoch 3/10
560/560 [==============================] - 41s 74ms/step - loss: 0.2216 - accuracy:
0.9118 - val_loss: 0.4062 - val_accuracy: 0.8270
Epoch 4/10
560/560 [==============================] - 40s 72ms/step - loss: 0.1742 - accuracy:
0.9360 - val_loss: 0.4822 - val_accuracy: 0.8258
Epoch 4: early stopping
187/187 [==============================] - 3s 16ms/step
Test Accuracy:  0.8223078211354882
Precision: 0.8094783715012722
Recall: 0.8463584968407051
F1 score: 0.8275077223215738
Confusion matrix:
 [[2365  599]
 [ 462 2545]]
```



4(c): Now we have trained LSTM model with 2 layer and with same architecture we followed for RNN and the observations and plots are as follows:-

```
Epoch 1/10
560/560 [==============================] - 106s 171ms/step - loss: 0.4415 -
accuracy: 0.7860 - val_loss: 0.4007 - val_accuracy: 0.8332
Epoch 2/10
560/560 [==============================] - 84s 151ms/step - loss: 0.2991 - accuracy:
0.8778 - val_loss: 0.4141 - val_accuracy: 0.8339
Epoch 3/10
560/560 [==============================] - 89s 160ms/step - loss: 0.2315 - accuracy:
0.9086 - val_loss: 0.4110 - val_accuracy: 0.8208
Epoch 4/10
560/560 [==============================] - 85s 151ms/step - loss: 0.1733 - accuracy:
0.9351 - val_loss: 0.4488 - val_accuracy: 0.8156
Epoch 4: early stopping
187/187 [==============================] - 8s 39ms/step
Test Accuracy:  0.8114218723831854
Precision: 0.7980982567353407
Recall: 0.8373794479547721
F1 score: 0.8172671210645895
Confusion matrix:
 [[2327  637]
 [ 489 2518]]
```

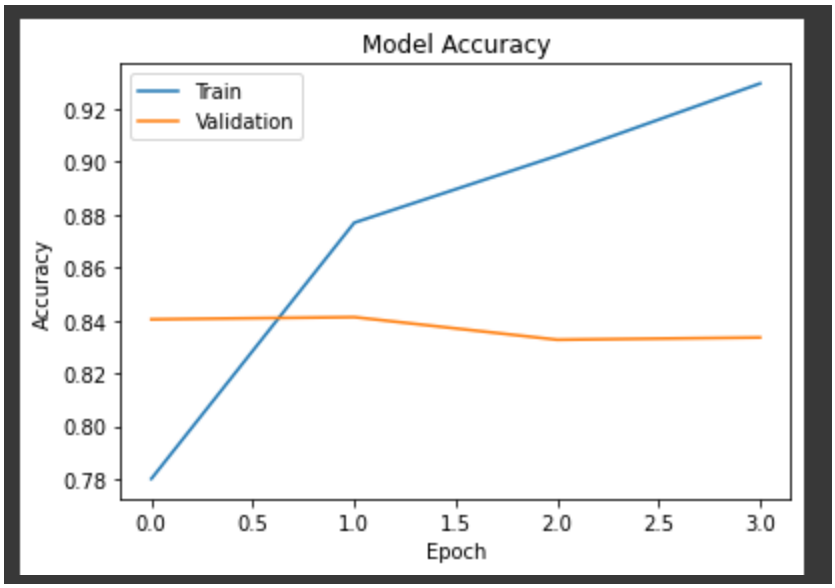


Google Colab

link:-https://colab.research.google.com/drive/1FTbhrqiZC5r6FTmkS5_6e5GGlYqwxWAO?usp=sharing

4(d):-Now we have trained 1 layer Bi-LSTM model on the same architecture and the observations and plots are as follows:-

```
Epoch 1/10
560/560 [==============================] - 91s 149ms/step - loss: 0.4563 - accuracy:
0.7801 - val_loss: 0.3743 - val_accuracy: 0.8404
Epoch 2/10
560/560 [==============================] - 87s 156ms/step - loss: 0.3028 - accuracy:
0.8769 - val_loss: 0.3686 - val_accuracy: 0.8412
Epoch 3/10
560/560 [==============================] - 83s 149ms/step - loss: 0.2445 - accuracy:
0.9022 - val_loss: 0.3890 - val_accuracy: 0.8327
Epoch 4/10
560/560 [==============================] - 86s 154ms/step - loss: 0.1876 - accuracy:
0.9295 - val_loss: 0.4122 - val_accuracy: 0.8335
Epoch 4: early stopping
187/187 [==============================] - 9s 41ms/step
Test Accuracy:  0.8283369619829174
Precision: 0.8268469656992085
Recall: 0.8337213169271699
F1 score: 0.8302699122371253
Confusion matrix:
 [[2439  525]
 [ 500 2507]]
```



Google Colab

Link:-https://colab.research.google.com/drive/1s_LQqOtYr7cE9Z7MEDRjUMFV361cJSO_?usp=sharing

Comparison between 4(b) and 4(d)

Both the models LSTM with one layer and BI-LSTM is giving same results as both the codes are giving same accuracy with same batch size

So both the models are giving almost same results.

4(e):- Word2Vec embedding:-

We have coded to load the word2vec embeddings from google news and used it in Bi-LSTM model as input embedding and get the observations as follows-

```
 precision: 0.8399 - recall: 0.8324
Loss:  0.34746497700868008
Accuracy:  0.85457676498230543
Precision:  0.8476739879886865
Recall:  0.8223874950943578782
```

Google colab link:

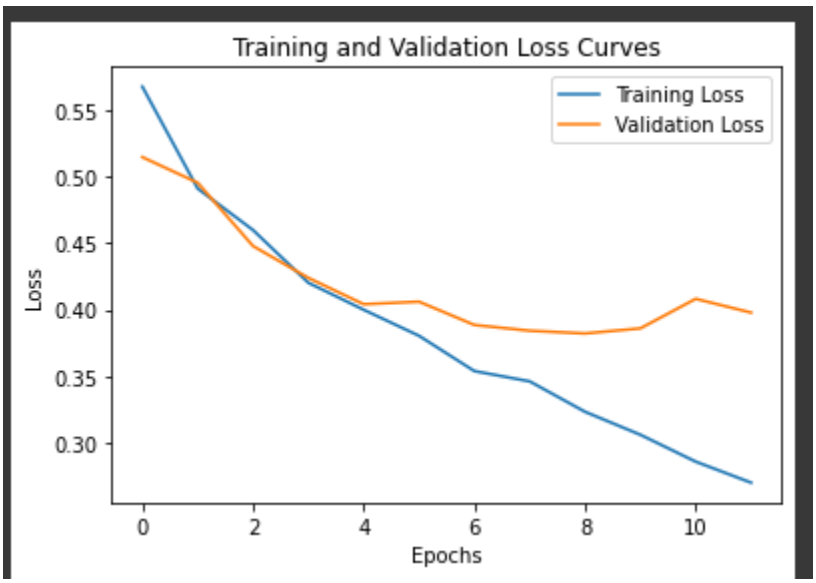
https://colab.research.google.com/drive/1oT--fjG78yxaAOn7GRlSPgKBOsr1F8S5?usp=sharing

4(f):-Glove Embedding:-

Now we have download and use Glove embedding on BI-Directional model and the observations and the plots are as follows:-

```
Found 106524 unique tokens.
(29854, 100)
Shape of data tensor: (29854, 100)
Shape of label tensor: (29854,)
Number of samples in training set: 13434
Number of samples in validation set: 4478
Found 400000 word vectors.
Train...
Epoch 1/16
210/210 [==============================] - 15s 23ms/step - loss: 0.5673 - accuracy: 0.7019 - precision:
0.7186 - recall: 0.6999 - val_loss: 0.5145 - val_accuracy: 0.7494 - val_precision: 0.7811 - val_recall: 0.6839
Epoch 2/16
210/210 [==============================] - 3s 17ms/step - loss: 0.4909 - accuracy: 0.7643 - precision: 0.7827
- recall: 0.7523 - val_loss: 0.4953 - val_accuracy: 0.7564 - val_precision: 0.8254 - val_recall: 0.6424
Epoch 3/16
210/210 [==============================] - 3s 13ms/step - loss: 0.4598 - accuracy: 0.7878 - precision: 0.7993
- recall: 0.7821 - val_loss: 0.4478 - val_accuracy: 0.7946 - val_precision: 0.8131 - val_recall: 0.7604
Epoch 4/16
210/210 [==============================] - 3s 13ms/step - loss: 0.4202 - accuracy: 0.8112 - precision: 0.8214
- recall: 0.8047 - val_loss: 0.4239 - val_accuracy: 0.8106 - val_precision: 0.7869 - val_recall: 0.8443
Epoch 5/16
210/210 [==============================] - 3s 13ms/step - loss: 0.4000 - accuracy: 0.8241 - precision: 0.8346
- recall: 0.8207 - val_loss: 0.4042 - val_accuracy: 0.8142 - val_precision: 0.8094 - val_recall: 0.8177
Epoch 6/16
210/210 [==============================] - 4s 18ms/step - loss: 0.3805 - accuracy: 0.8292 - precision: 0.8384
- recall: 0.8249 - val_loss: 0.4060 - val_accuracy: 0.8240 - val_precision: 0.8080 - val_recall: 0.8447
Epoch 7/16
210/210 [==============================] - 3s 13ms/step - loss: 0.3540 - accuracy: 0.8476 - precision: 0.8556
- recall: 0.8436 - val_loss: 0.3886 - val_accuracy: 0.8303 - val_precision: 0.8021 - val_recall: 0.8711
Epoch 8/16
210/210 [==============================] - 3s 15ms/step - loss: 0.3463 - accuracy: 0.8504 - precision: 0.8571
- recall: 0.8511 - val_loss: 0.3843 - val_accuracy: 0.8234 - val_precision: 0.8395 - val_recall: 0.7963
Epoch 9/16
210/210 [==============================] - 4s 19ms/step - loss: 0.3235 - accuracy: 0.8603 - precision: 0.8657
- recall: 0.8598 - val_loss: 0.3822 - val_accuracy: 0.8321 - val_precision: 0.8168 - val_recall: 0.8521
```

Epoch 10/16
210/210 [==============================] - 4s 18ms/step - loss: 0.3062 - accuracy: 0.8704 - precision: 0.8750 - recall: 0.8712 - val_loss: 0.3860 - val_accuracy: 0.8296 - val_precision: 0.8287 - val_recall: 0.8234
Epoch 11/16
210/210 [==============================] - 3s 15ms/step - loss: 0.2861 - accuracy: 0.8796 - precision: 0.8812 - recall: 0.8828 - val_loss: 0.4082 - val_accuracy: 0.8274 - val_precision: 0.7813 - val_recall: 0.9037
Epoch 12/16
210/210 [==============================] - 3s 13ms/step - loss: 0.2703 - accuracy: 0.8871 - precision: 0.8884 - recall: 0.8906 - val_loss: 0.3979 - val_accuracy: 0.8356 - val_precision: 0.8230 - val_recall: 0.8489
Epoch 12: early stopping
374/374 [==============================] - 3s 8ms/step - loss: 0.3973 - accuracy: 0.8343 - precision: 0.8340 - recall: 0.8324
Loss: 0.3972749710083008
Accuracy: 0.8342823386192322
Precision: 0.8339879512786865
Recall: 0.8323926329612732
Confusion Matrix:
[[4965 988]
 [991 4998]]
F1 Score: 0.8347390396659709



Google Colab Link:-

https://colab.research.google.com/drive/17h99_WBEepvarunfKmOCatBb8WaqgSWm?usp=sharing

Comparison between 1 layer Bi-LSTM model and with glove embedding Bi-LSTM model:

Bi-LSTM model with 1 layer with glove embedding is giving slightly more good accuracy in comparison to Bi-LSTM model without glove embedding as accuracy of model with glove embedding is good compare to without glove embedding model.

Future Scope and Challenges faced: difficulties are faced due to large amount of data , memory clashing problem was occurring . to solve this we have taken a big subset of this data set of around 30000 datapoints. More work can be done on this dataset by taking different types of embedding o different models and compare their results in order to get best model.

Each member in the group has worked actively. Data preprocessing , RNN, Bi-directional, Glove embedding,, word2vec is done by Deepanshi Jindal, Data preprocessing, Decision tree, Naive bayes, LSTM is done by Sushil kumar Lakhiwal. Moreover we have solved the errors and difficulties of big data set together. So we have equally contributed on this assignment.