Sentiment Analysis Report

1. Introduction

This project aims to classify 1,600,000 automatically annotated tweets as positive or negative using two machine learning models based on Neural Networks. The models used are a sequential Dense model and a Convolutional Neural Network with accuracies around 77-78%.

2. Data Preprocessing

Observations made on the data:

- Data has no null values
- The target column is well balanced as the target classes have an equal count.

Steps taken to clean the data:

- ***Removing unrequired characters*:** HTML tags, punctuations, numbers, single characters and multiple spaces were removed.
- ***Splitting into training and testing data:*** The dataset is divided into the training set and testing set with 0.8 and 0.2 proportions of the whole dataset.
- ***Tokenization**:* Tokenization vectorizes each text into a sequence of integers. It has been applied to training and testing data.
  Additionally, pad_sequences are applied to the list of integers to convert them into a 2D NumPy array with dimensions (number of samples, maxlen). Sequences that are shorter than maxlen are padded with 0 until they are maxlen long. Sequences longer than maxlen are truncated so that they fit the desired length. In this project, the value of maxlen is assigned as 100, meaning each sequence is of length 100.
- ***Word Embeddings:***
  Word Embeddings captures the context of a word in a document, semantic and syntactic similarity, relation with other words. A pre-trained word embedding **GloVe** was used in this project. It generates word embeddings by aggregating global word co-occurrence matrices from the given corpus.
  The GloVe file with 100 dimensions was used in this project. The file contains words and their corresponding vectors with 100 dimensions. An embedding matrix is further created to find the similarity between words using this file.

3. Modelling

Two neural networks based models were used in this project. Their architecture, performance and their losses are compared.

Model 1: Dense Model

The first model used was a simple sequential dense model. The model architecture contains:

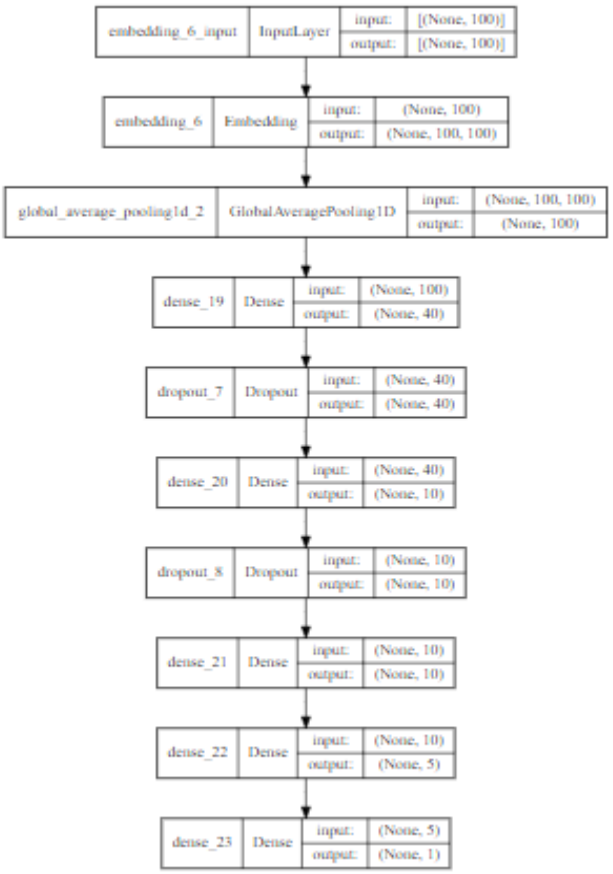
- *Embedding layer:* The first hidden layer, used for generating an embedding vector for each input sequence.
- *GlobalAveragePooling1D layer:* It takes a 2-dimensional tensor as input and computes a single average value for each of the input channels (the second dimension).
- *Dense layers*: Further 5 dense layers are used as fully connected layers for classification.

- ➢ *Dropout layers:* Two dropout layers with 0.5 rates were used as an effective regularization method to avoid overfitting and improve generalization.

Overall, the model contains 8 hidden layers and one output layer.
Parameters used for compiling the model were:

1. Optimizer: Adam
2. Loss: binary_crossentropy
3. Metrics: Accuracy

The architecture of the model is visualized as follows:



Performance Evaluation of the model:

The classification report generated for the model:

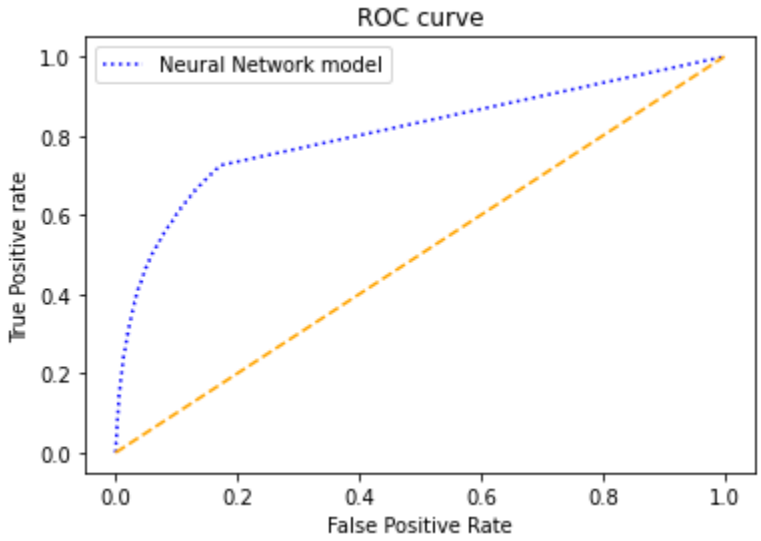
	precision	recall	f1-score	support
0	0.69	0.90	0.78	15958
1	0.86	0.59	0.70	16042
accuracy			0.75	32000
macro avg	0.77	0.75	0.74	32000
weighted avg	0.77	0.75	0.74	32000

Confusion matrix:

```
[[14399  1559]
 [ 6508  9534]]
```

Accuracy of the model:
Training dataset: **79.92 %**
Testing dataset: **74.79 %**

AUC score of the model: **0.805** (Close to 1, the model can separate the classes well)
Roc curve:



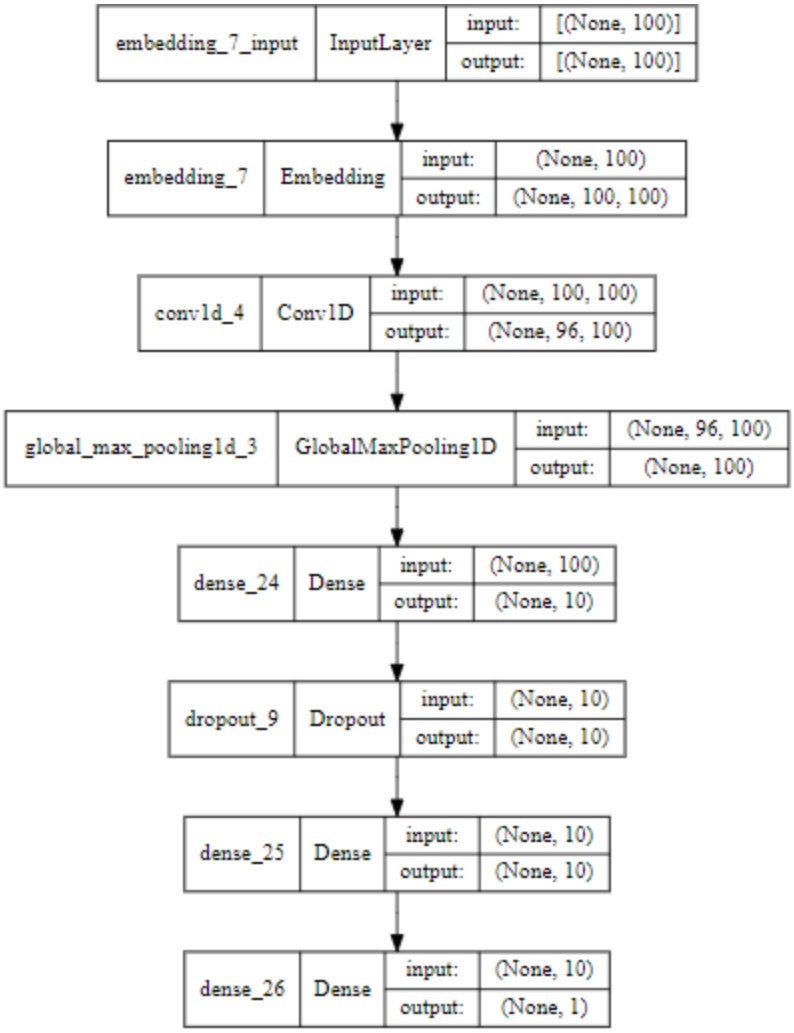
Model 2: Convolutional Neural Networks model (CNN)
The second model used was a CNN model. The model architecture contains:

- *Embedding layer:* The first hidden layer, used for generating an embedding vector for each input sequence.
- *Conv1D layer:* It was used to convolve data into smaller feature vectors.
- *GlobalMaxPooling1D:* It takes a 2-dimensional tensor as input and computes the maximum value for each of the input channels (the second dimension).
- *Dense layers:* Further 3 dense layers are used as fully connected layers for classification.
- *Dropout layer:* One dropout layer with 0.5 rate was used as an effective regularization method to avoid overfitting and improve generalization.

Overall, the model contains 6 hidden layers and one output layer.
Parameters used for compiling the model were:

1. Optimizer: Adam
2. Loss: binary_crossentropy
3. Metrics: Accuracy

The architecture of the model is visualized as follows:



Performance Evaluation of the model:

The classification report generated for the model:

	precision	recall	f1-score	support
0	0.75	0.85	0.79	15958
1	0.83	0.71	0.76	16042
accuracy			0.78	32000
macro avg	0.79	0.78	0.78	32000
weighted avg	0.79	0.78	0.78	32000

Confusion matrix:

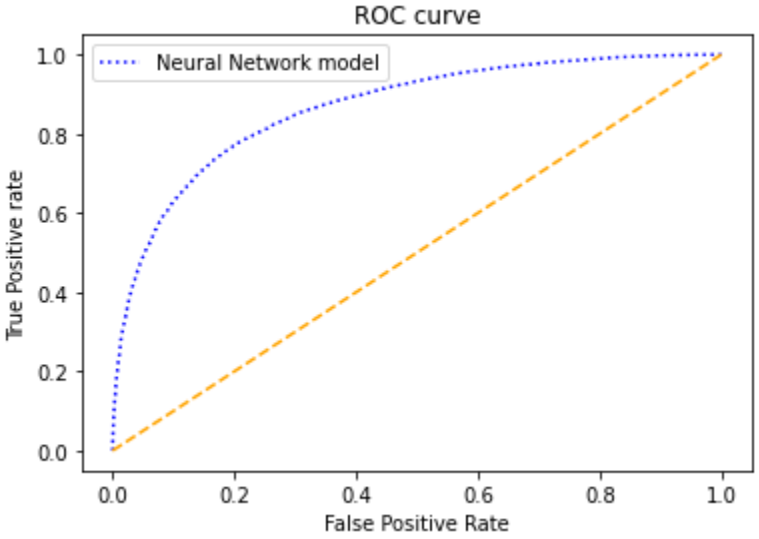
```
[[13574  2384]
 [ 4646 11396]]
```

Accuracy of the model:

Training dataset: **82.56 %**

Testing dataset: **78.03 %**

AUC score of the model: **0.865**
Roc curve:



4. Comparing the performance of two models:

Models/Performance	**Training set Accuracy**	**Testing set Accuracy**	**AUC score**
Dense Model	79.92 %	74.79 %	0.805
CNN Model	**82.56 %**	**78.03 %**	**0.865**

It can be seen that the CNN model performs slightly better than the simple dense model on both the training as well as the testing datasets.
The reason why the CNN model performs better than Dense could be because:

- Convolutional Layer utilizes fewer parameters by requiring input values to share parameters. The Dense Layer employs a linear operation, which means that each output is formed by the function based on each input.
- The Convolutional layer implements a set of filters to perform the convolution operation, which has a small size. The output of the convolution layers is formed by a small number of inputs that depends on the size of the filter, and the weights are shared across all words. In other words, the output is created by using the same coefficients for all words and the neighboring words as input.

5. Conclusion:

The two neural networks models performed really well in classifying the sentiments of the tweets with an accuracy of 77-78%. There were a few challenges while developing the project:

- On removing the stop words, the accuracy of both the models dropped. So the stop words were not removed in this case.
- It was hard to cross the accuracy threshold of more than 80%. The current accuracies of the models have been improved by hyper-parameter tuning.

Overall working on the project was a great experience as it taught me to try and experiment with different models and approaches to get the best results.