Mini Project 2: Sentiment140 Twitter Dataset Analysis

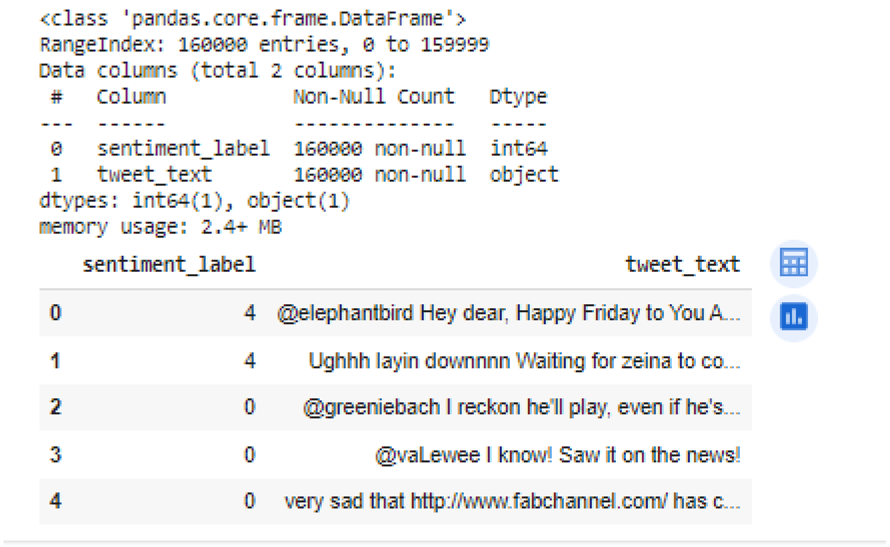
Introduction

In this project, I aim to conduct sentiment analysis on the Sentiment140 dataset, a publicly available dataset created by three graduate students at Stanford University: Alec Go, Richa Bhayani, and Lei Huang. The dataset consists of approximately 1,60,000 automatically annotated tweets, making it a valuable resource for sentiment analysis tasks. Sentiment analysis involves classifying text into different sentiment categories, such as positive, negative, and essential for various applications, including social media monitoring, customer feedback analysis, and market research.

In this report, I compare the performance of the following models: Simple Neural Network, Convolutional Neural Network (CNN), Bidirectional LSTM, Random Forest Classifier, and Logistic Regression Model.

Data Overview

The dataset contains two crucial columns: "sentiment_label," denoting the sentiment (0 for negative and 4 for positive), and "tweet_text," containing the text of the tweets.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 160000 entries, 0 to 159999
Data columns (total 2 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   sentiment_label  160000 non-null  int64 
 1   tweet_text       160000 non-null  object
dtypes: int64(1), object(1)
memory usage: 2.4+ MB
```

	sentiment_label	tweet_text
0	4	@elephantbird Hey dear, Happy Friday to You A...
1	4	Ughhh layin downnnn Waiting for zeina to co...
2	0	@greeniebach I reckon he'll play, even if he's...
3	0	@vaLewee I know! Saw it on the news!
4	0	very sad that http://www.fabchannel.com/ has c...

Figure: data information

Sentiment Label: This column contains integer values representing the sentiment of the tweet. A label of 0 typically indicates a negative sentiment, while a label of 4 corresponds to a positive sentiment. These labels were automatically annotated during the creation of the dataset.

Tweet Text: The tweet text column contains the actual text of the tweets. It encompasses a wide range of topics, opinions, and expressions shared by Twitter users.

Data Preprocessing

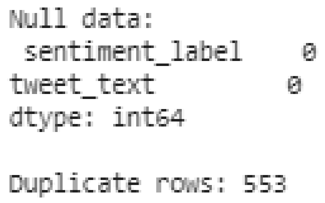
In the data preprocessing phase, several steps were undertaken to clean and prepare the dataset for modeling. This section outlines the methods applied to ensure the data's quality and suitability for training machine learning models.

- Handling Missing Data:

  Before proceeding with preprocessing, the dataset was checked for missing values. Fortunately, no missing data was found, eliminating the need for imputation or removal of incomplete entries.

- Handling Duplicate Data:

  Duplicate rows in the dataset were identified to ensure data integrity and avoid bias in model training. Initially, the dataset was scanned for duplicates, revealing the presence of duplicate rows.



```
Null data:
 sentiment_label    0
tweet_text         0
dtype: int64

Duplicate rows: 553
```

Figure: No of duplicate rows

  In total, 553 duplicate rows were found. These duplicate entries were subsequently removed to maintain the dataset's integrity and prevent redundancy in the training data.

- Text Preprocessing:

  The text data in the 'tweet_text' column underwent preprocessing to standardize its format and remove irrelevant information. This involved the following steps: HTML Tag Removal, Removing Punctuations and Numbers, Single Character Removal, Removing Multiple Spaces, and finally, any multiple consecutive spaces were replaced with a single space to standardize the text's format and ensure consistency.

- Target Label Transformation:

  The sentiment labels were transformed to binary values to facilitate binary classification. Positive sentiment labels (denoted by '4') were mapped to '1', while negative sentiment labels (denoted by '0') were preserved as '0'. This transformation simplifies the sentiment analysis task by framing it as a binary classification problem.

- Data Splitting:

  The preprocessed data was split into input features (X) and target labels (y) arrays. The input features (X) consist of the preprocessed tweet text, while the target labels (y) represent the binary sentiment labels.

- Final Dataset Preparation:

  After preprocessing, the dataset was ready for model training and evaluation. The input features (X) and target labels (y) were appropriately formatted and ready to be fed into machine learning models for sentiment analysis.

Exploratory Data Analysis (EDA)

In this section, I conducted exploratory data analysis (EDA) to gain insights into the distribution and characteristics of the Sentiment140 dataset. EDA is crucial for understanding the dataset's structure, identifying patterns, and informing subsequent modeling decisions.

Distribution of Sentiment Labels:

I began by examining the distribution of sentiment labels in the dataset. The sentiment labels were mapped to their corresponding sentiment categories: "Negative" for label 0 and "Positive" for label 4. The distribution revealed a balance between positive and negative sentiments, with positive sentiments slightly outnumbering negative sentiments.

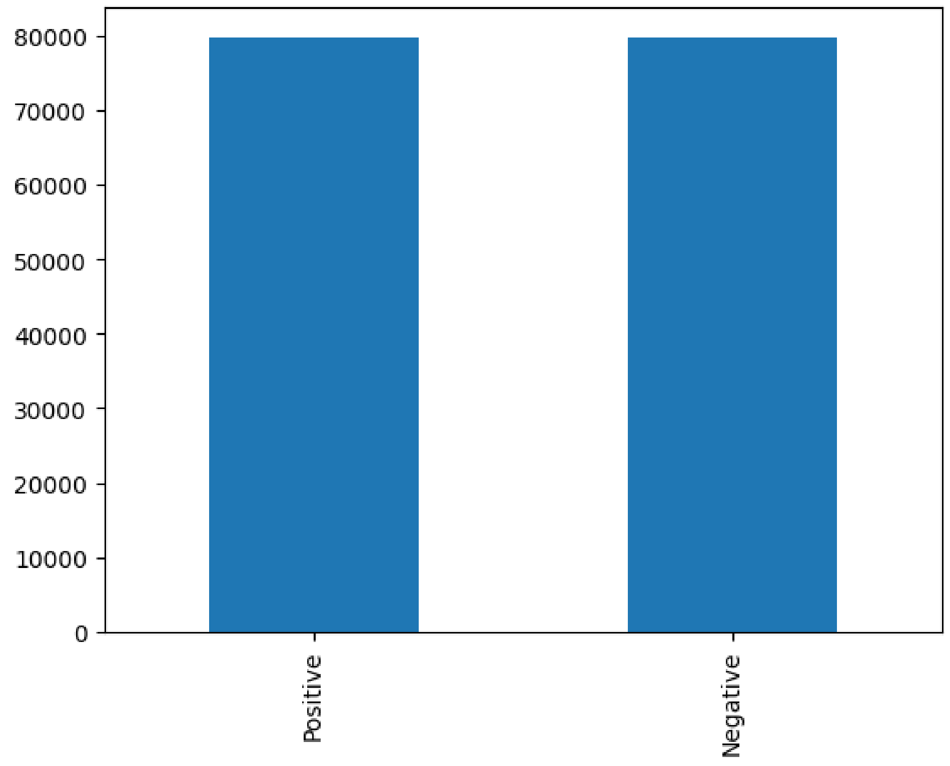


Figure: Distribution of the sentiment label

This balanced distribution ensures that the dataset is suitable for training sentiment analysis models without bias towards any sentiment class.

Distribution of Text Length:

Next, we analyzed the distribution of text lengths in the tweet text data. A histogram plot illustrated the distribution of text lengths, showing the frequency of tweets corresponding to different text length ranges.

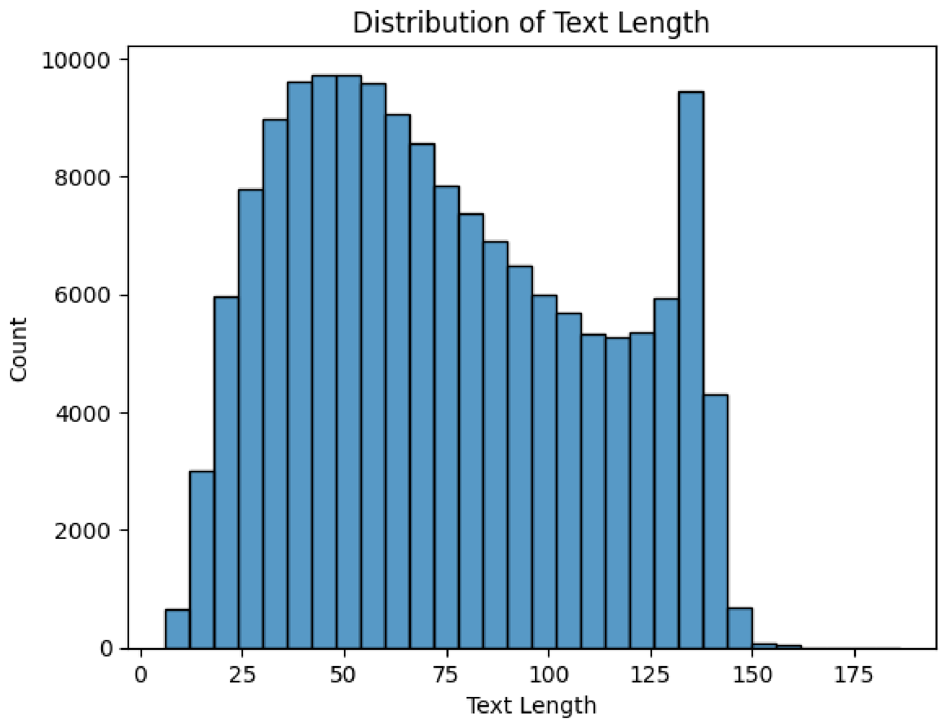


Figure: Distribution of Text Length

The distribution appeared to be right-skewed, with most tweets having relatively short text lengths. This insight is valuable for understanding the variability in tweet lengths and may influence feature engineering decisions during model development.

Word Frequency Analysis:

We conducted word frequency analysis to identify the most common words used in positive and negative tweets separately. The top 20 most frequent words in both positive and negative tweets were determined using the Counter module.

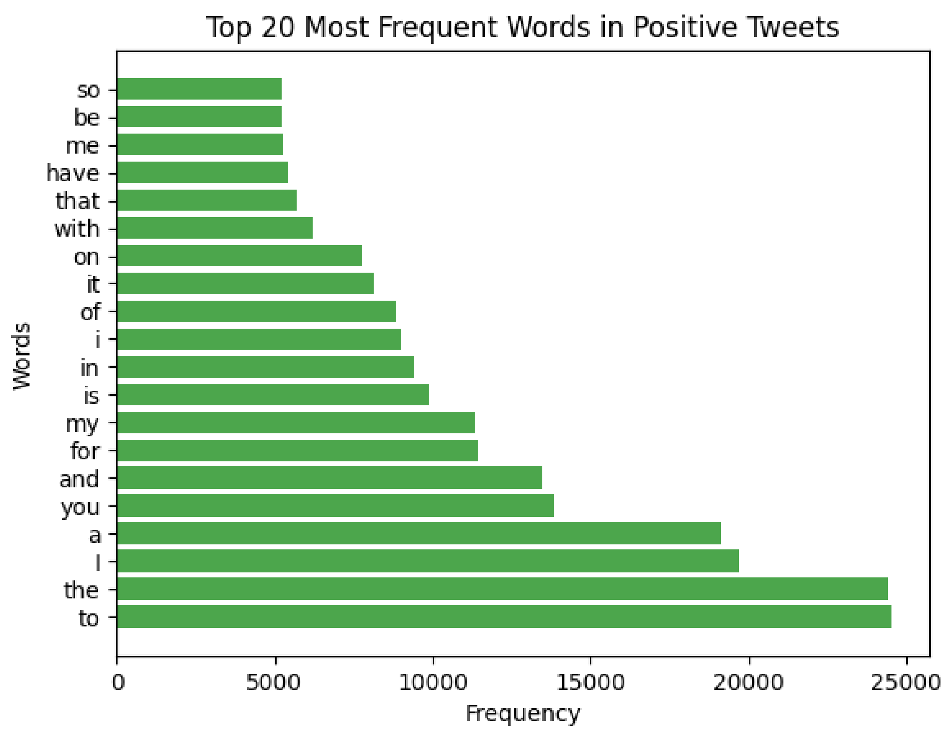


Figure: Top 20 Most frequent words in positive tweets

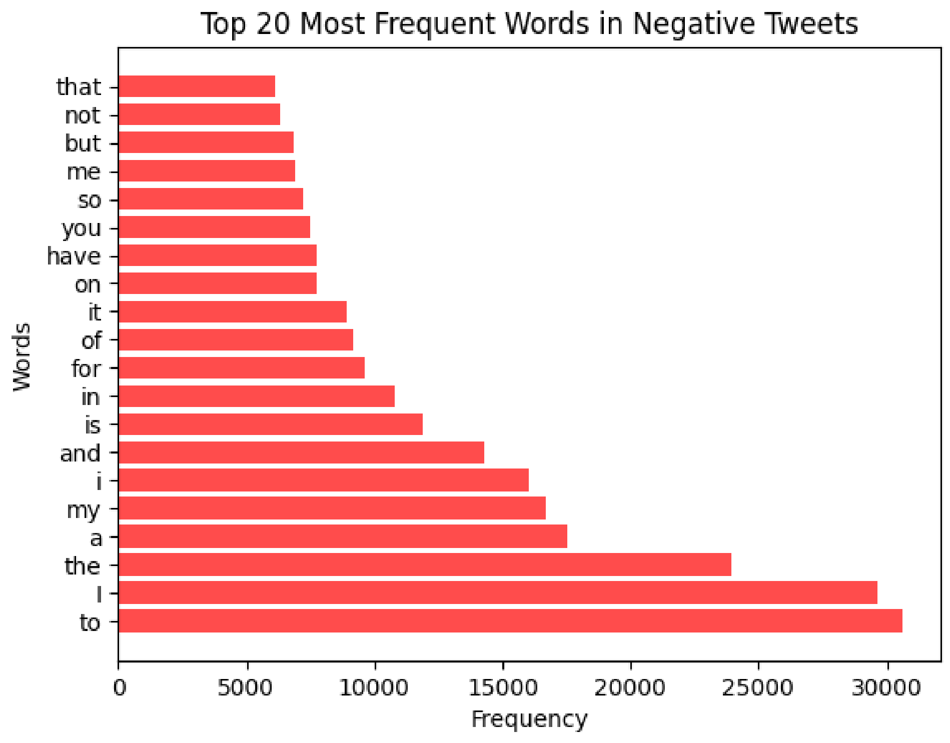


Figure: *Top 20 Most frequent words in negative tweets*

Visualization of word frequencies through horizontal bar plots provided insights into the vocabulary associated with each sentiment category. In positive tweets, words such as "my," "i," and "have" were prevalent, indicating positive sentiment expression. Conversely, negative tweets exhibited words like "not," "but" and "no," reflecting negative sentiment expressions.

The exploratory data analysis provided valuable insights into the distribution and characteristics of the Sentiment140 dataset. The balanced distribution of sentiment labels, the variability in tweet lengths, and the identification of common words associated with each sentiment category are essential considerations for subsequent model development and feature engineering. These insights lay the groundwork for building effective sentiment analysis models capable of accurately classifying tweet sentiments based on their textual content.

Dataset Splitting and Text Tokenization

The dataset was split into training and testing sets using the train_test_split function from the scikit-learn library. A test size of 0.2 was specified, indicating that 20% of the data would be reserved for testing, while the remaining 80% would be used for training. This splitting strategy ensures that the model's performance can be adequately evaluated on unseen data.

Text tokenization involves converting textual data into numerical sequences that can be processed by machine learning models. To achieve this, we utilized the Tokenizer class from the Keras library. The tokenizer was fitted on the training data to learn the vocabulary and assign numerical indices to each unique word. I specified a maximum number of words to keep based on word frequency, limiting the vocabulary size to improve computational efficiency and mitigate overfitting.

To ensure uniform input dimensions across all sequences, we performed sequence padding using the pad_sequences function from Keras. Sequences were padded or truncated to a maximum length of 100 tokens, ensuring consistency in sequence length. Padding was applied at the end of sequences, and any sequences exceeding the maximum length were truncated.

Modelling

As suggested by the professor, I tried using several models and changing several parameters to get the good accuracy level. Among which i will list a few here.

Simple Neural Network (SNN):

The SNN consists of an embedding layer, followed by several dense layers with dropout for regularization. The SNN achieved a test accuracy of approximately 76.38%. SNN is simple to implement and train but may struggle with capturing complex patterns in text data.

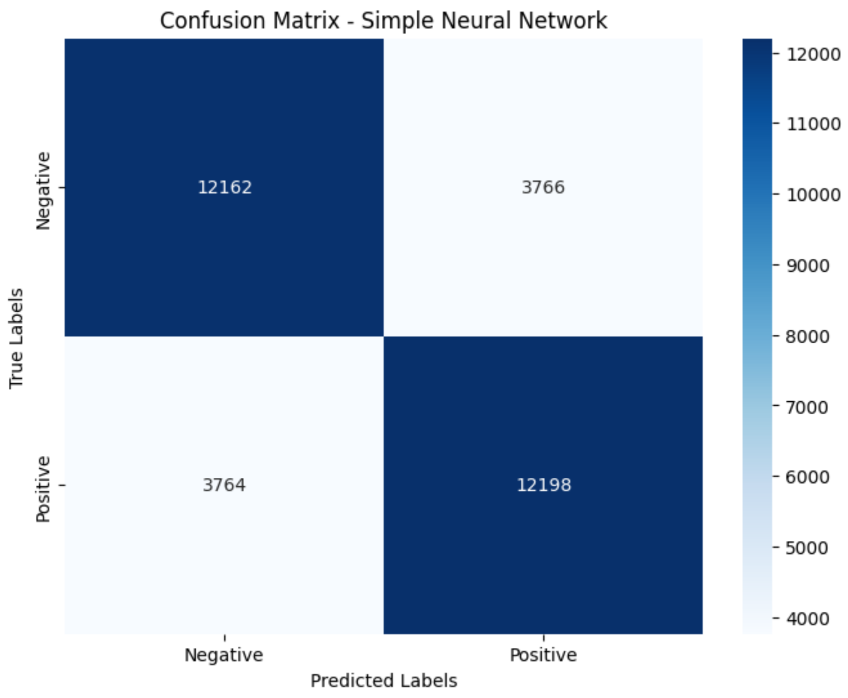


Figure: Confusion matrix for Simple Neural Network

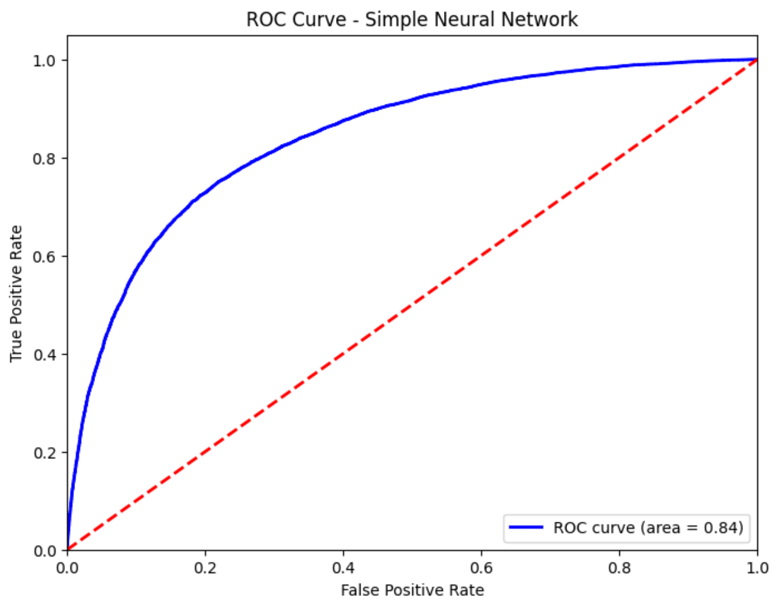


Figure: ROC Curve for Simple Neural Network

Convolutional Neural Network (CNN):

The CNN employs convolutional and pooling layers to extract features from the text. The CNN achieved a test accuracy of approximately 73.62%. The CNN is effective in capturing spatial features but may not capture long-range dependencies in text data as effectively as recurrent models.

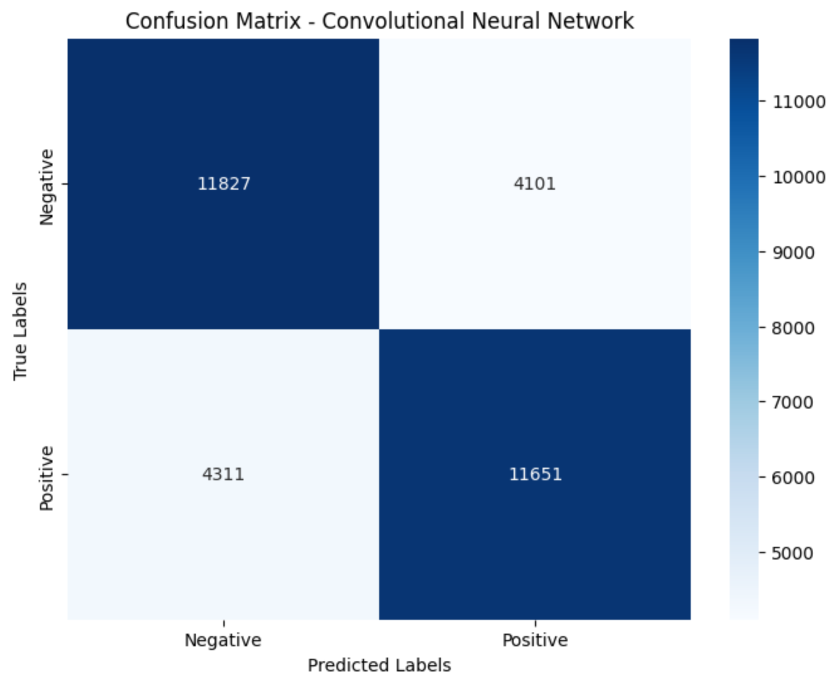


Figure: Confusion matrix for Convolutional Neural Network

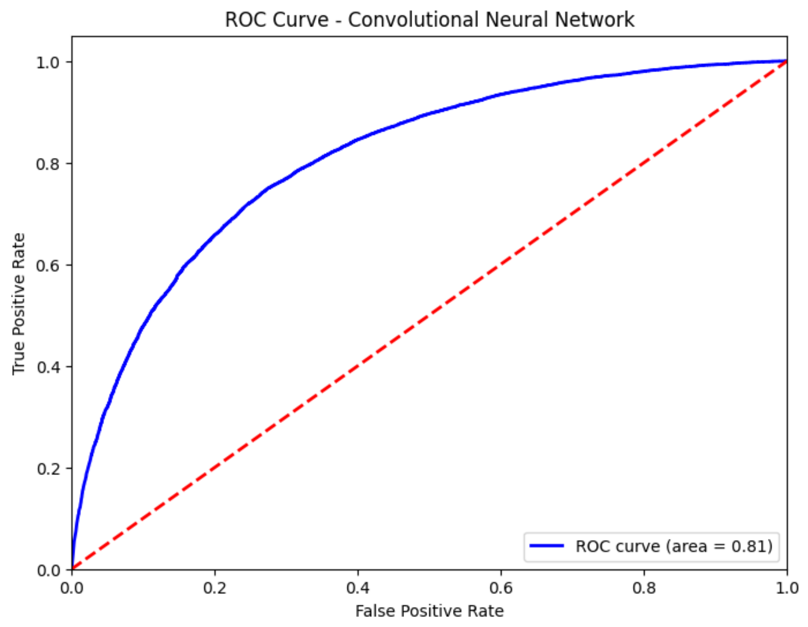


Figure: ROC Curve for Convolutional Neural Network

Bidirectional LSTM:

The Bidirectional LSTM uses bidirectional recurrent layers to capture sequential information from both directions. The Bidirectional LSTM achieved a test accuracy of approximately 78.77%. The Bidirectional LSTM is effective in capturing long-range dependencies but may be computationally expensive to train.

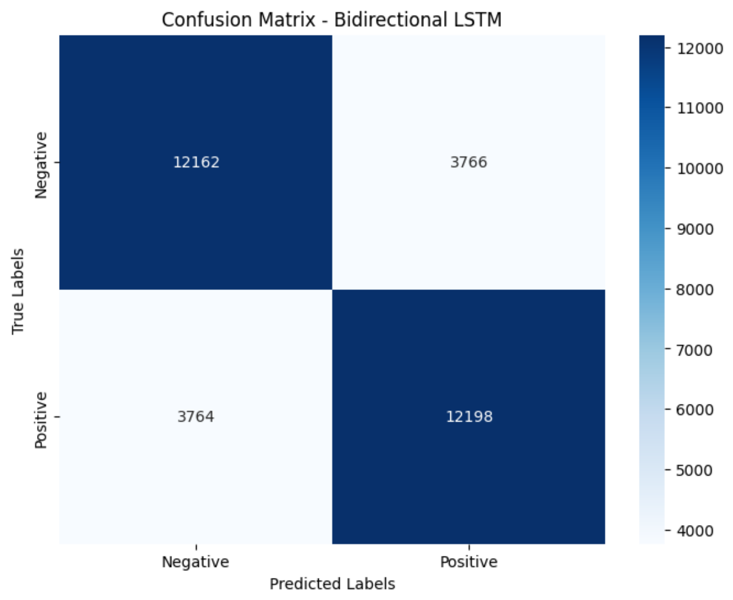


Figure: Confusion matrix for Bidirectional LSTM

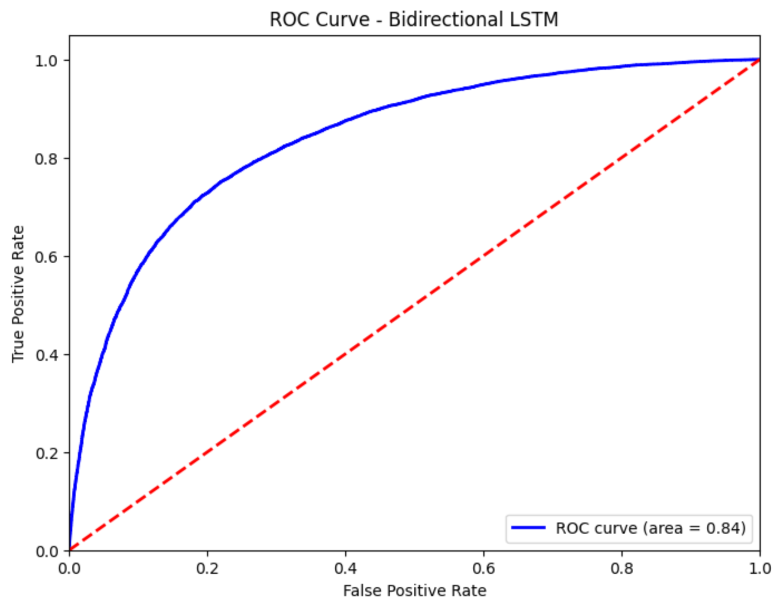


Figure: ROC Curve for Bidirectional LSTM

Random Forest Classifier:

The Random Forest Classifier is an ensemble of decision trees trained on TF-IDF features. The Random Forest Classifier achieved a test accuracy of approximately 76.19%. The Random Forest Classifier is robust and works well with high-dimensional data but may be overfit with large feature spaces.

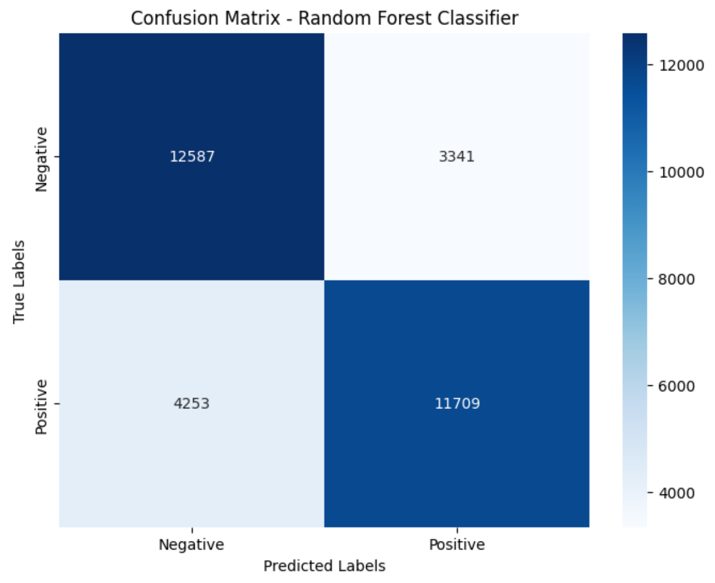


Figure: Confusion Matrix for Random Forest Classifier

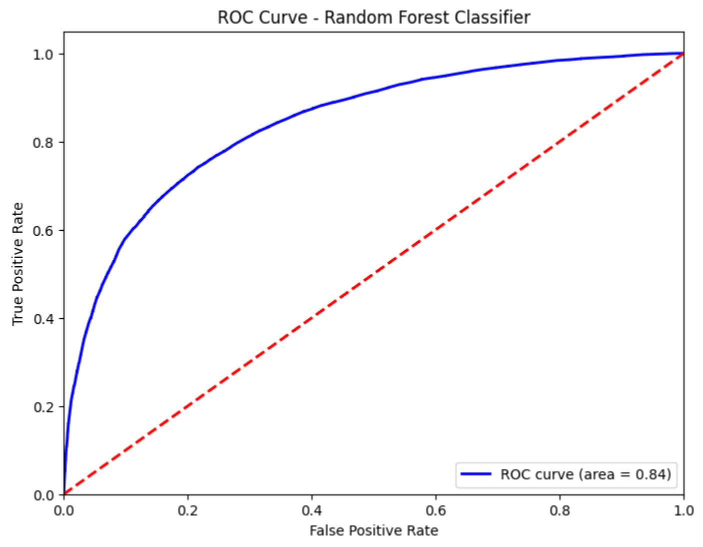


Figure: Confusion Matrix for Random Forest Classifier

Logistic Regression Model:

The Logistic Regression Model performs binary classification using logistic regression. The Logistic Regression Model achieved a test accuracy of approximately 77.65%. Logistic Regression is simple, interpretable, and computationally efficient but may underperform when the relationship between features and target is non-linear.

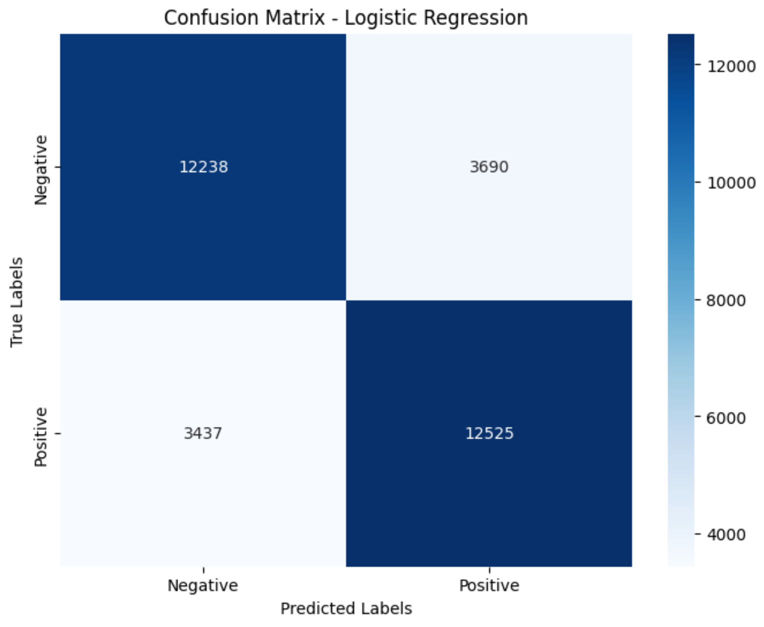


Figure: Confusion Matrix for Logistic Regression Model

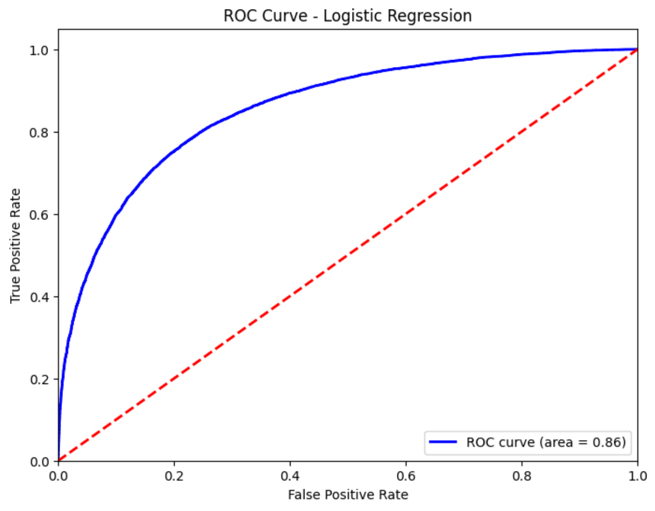


Figure: ROC Curve for Logistic Regression Model

The performance of each model was evaluated based on metrics such as accuracy, ROC AUC score, and the shapes of their ROC curves. One model might have performed better than the others due to its ability to capture specific patterns or relationships present in the data. For example, CNN might excel in capturing spatial features, while the Bidirectional LSTM might better capture temporal dependencies. The Random Forest Classifier, on the other hand, could perform well due to its ability to handle non-linearity and feature interactions effectively.

It can be improved by hyperparameter Tuning where each model could benefit from further hyperparameter tuning to optimize performance. Feature Engineering by exploring additional features or alternative representations of the data could enhance model performance. And ensembling methods like combining the predictions of multiple models through ensemble methods like stacking or boosting could potentially improve overall performance.

Conclusions

In conclusion, I compared and evaluated the performance of various machine learning models for sentiment analysis on the Sentiment140 dataset. Among the models tested, the Bidirectional LSTM achieved the highest test accuracy of 78.77%, followed closely by the Logistic Regression Model with an accuracy of 77.65%. These results suggest that models utilizing sequential information, such as LSTMs, are effective for sentiment analysis tasks. However, the choice of the best model depends on factors such as computational resources, interpretability, and the specific requirements of the application.

The scientific bottleneck for this project for me was the complexity of data for dealing with complex relationships and patterns present in the data, ensuring the quality and reliability of the data used for model training and evaluation, by preventing models from memorizing the training data and generalizing well to unseen data, and ensuring that the models provide interpretable results that can be understood and trusted by stakeholders. Importantly, it was difficult to get a good accuracy level in whatever model I used.

Overall, it was an amazing experience doing this project besides the time-consuming factor and various model training. I had a chance to learn various parameters that we may need to deal with machine learning projects.