Sentiment Analysis on Tweets

Ahmad Kamal Baig

1 Introduction

Twitter is a social platform where users express their thoughts in short messages called tweets, making it a rich source for understanding public sentiment. However, the informal and diverse language used in tweets, including slang and abbreviations, poses a challenge for accurately analyzing these sentiments.

To address this, we're applying Natural Language Processing (NLP) techniques to perform sentiment analysis on a selected set of tweets. By doing so, we aim to decode the emotions behind the tweets, providing valuable insights into public opinion on various topics. This analysis will help businesses, policymakers, and researchers make informed decisions based on the collective mood of Twitter users.

2 Data Processing

2.1 Dataset

The dataset used in this project contains 10% of original Sentiment 140 Tweets Dataset. The dataset contains **16000** rows and only two columns, namely:

1. **Sentiment Analysis** (Integer)
2. **Tweet Text** (String)

2.2 Binary Encoding

For our preprocessing steps, we first started with binary encoding. Our sentimental analysis column had only two distinct values, **0** for representing **negative sentiment** and **4** for **positive sentiment**. As this big gap of values may hinder performance of our classical machine learning models, we decided to encode them.

2.3 Removing Usernames

During our exploratory research, we found that most tweets contained usernames. Primary use of these usernames is to reply to someone in their tweets, while the secondary use is to tag someone. In most cases, the username doesn't contribute to the context of the tweet, therefore, we decided to remove them altogether.

2.4 Removing URLs

Website URLs doesn't contribute to the tweet as a whole, therefore, we decided to replace such URLs with the keyword "**website**" in order to retain the general context.

2.5 Removing Non Alphabetic Characters

We removed non-alphabetic characters from the dataset to streamline the input and concentrate on the linguistic content. This decision helped to diminish noise and variability in the data, which allowed the model to focus more effectively on understanding the semantic essence of the text. This step can be beneficial for enhancing the model's performance by reducing the complexity of the input features and ensuring consistency across the dataset.

2.6 Removing Extra Spaces

The removal of redundant spaces helped in reducing unnecessary computational complexity and ensured that the model's attention is directed towards the meaningful content of the text.

2.7 Converting words to lower case

This normalization technique allows us to treat words with the same meaning equally, regardless of their case usage in the original text. This prevents the model from considering the same words in different cases as distinct entities. This approach aids in simplifying the linguistic analysis and enhances the model's ability to learn from the textual data more effectively.

2.8 Removing Consecutive Letters

This adjustment was particularly targeted at informal text characteristics, such as the elongation of words for emphasis (e.g., converting "soooo" to "so"), to maintain the linguistic integrity of the words while minimizing noise. This step is instrumental in ensuring that the model interprets variations of a word caused by

such repetitions as a single, consistent entity. This may help in improving model's accuracy.

2.9 Lemmatization

We also applied lemmatization, a process where we converted words to their base or root form. For example, "running" turns into "run". This can help model to understand that different forms of a word are essentially the same, making it easier for the model to learn from the text. This step is crucial for simplifying the data and improving the model's ability to accurately interpret and process the language.

2.10 TF-IDF Vectorization

TF-IDF stands for "Term Frequency-Inverse Document Frequency". In our model, we used TF-IDF Vectorization to transform text into numerical values. It helps our model understand which words are important in a text by considering how often a word appears in a document and how unique the word is across all documents. This technique allows the model to pay more attention to meaningful words and less to common words that appear in many documents, like "the" or "is", making it better at analyzing and learning from the text.

3 Modeling

3.1 Logistic Regression

- **Overview**: Logistic Regression is a fundamental statistical approach used for binary classification problems. It models the probability of a binary outcome based on one or more predictor variables.
- **Reason for choosing:** Logistic regression has been chosen because of its interpretability, efficiency and performance.
- **Steps to perform**: We used scikit-learn's base model of logistic regression for this purpose. The dataset was split into **80-20 ratio** for training and testing respectively. The hyper parameters for this model involved setting **max_iterations** to **100**.
- **Accuracy**: The logistic regression model achieved an overall accuracy of **80%**, indicating correct predictions for the majority of cases.
- **Confusion Matrix**: The model correctly predicted negative sentiment and positive sentiment for 12759 and 12690 cases respectively, while incorrectly predicting 3243 negative sentiments and 3308 positive sentiments.

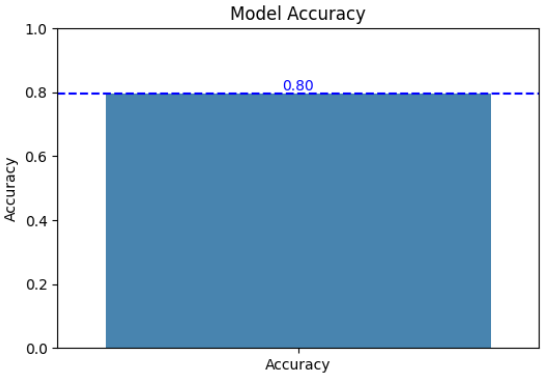

Figure 1: Logistic Regression Accuracy

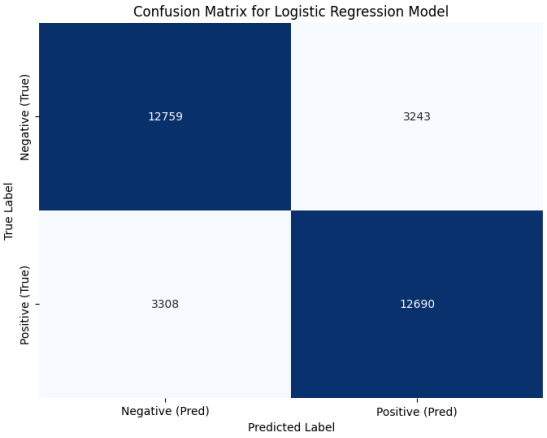

Figure 2: Linear Regression Confusion Matrix

- **Classification Report**: Precision, recall, and F1-score for negative sentiments are 0.79, 0.8, and 0.8 respectively; for positive sentiments, they are 0.8, 0.79, and 0.79 respectively. *This indicates that there is a good balance between precision and recall.*

3.2 LSTM

- **Overview:** Long Short-Term Memory (LSTM) is an advanced type of Recurrent Neural Network (RNN) architecture designed to handle the vanishing and exploding gradient problems associated with traditional RNNs. LSTMs are particularly adept at learning long-term dependencies such as in NLP.
- **Reason for choosing:** The LSTMs are very suitable for long sequences of data.
- **Steps to perform**: We use KERAS to train our model of LSTM. First we use a

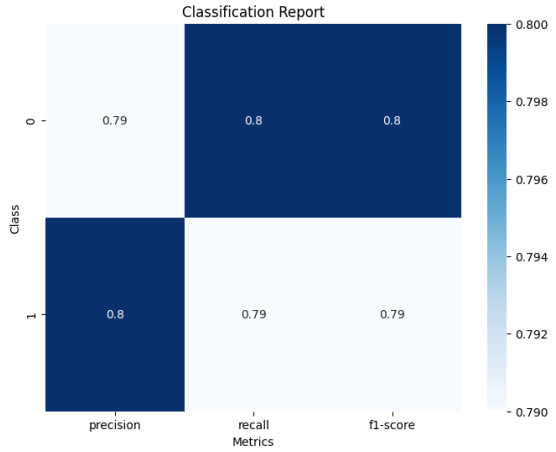

Figure 3: Linear Regression Classification Report

tokenizer to convert long sentences into individual tokens. Then we use **padding** so that all examples are of the same size. After this preprocessing, we define our model's structure and train it.

- **Architecture**: Our model contains a sequential layer with embedding layer. This is followed by an LSTM layer. In order to add some kind of **regularization**, we apply a dropout layer. At the end, we apply a fully connected layer with **1 output** with **sigmoid** activation.

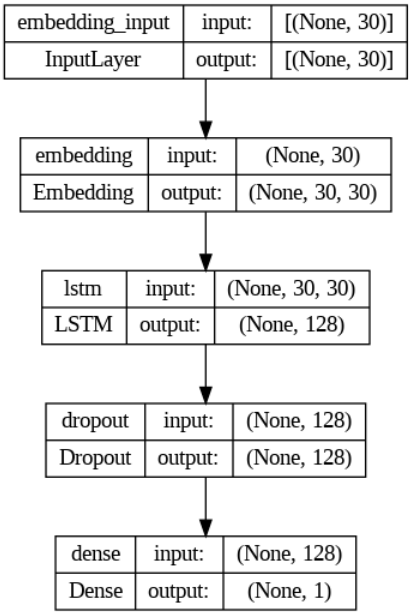

Figure 4: LSTM Architecture

- **Accuracy**: The model exhibits an average accuracy of **77%**, indicating that *it is struggling to train on a relatively small dataset.*

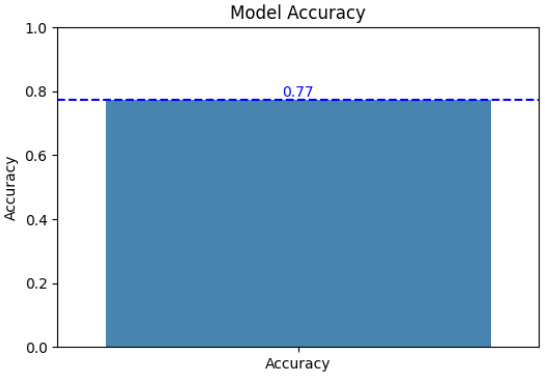

Figure 5: LSTM Accuracy

- **Confusion Matrix**: The model correctly predicted negative sentiment and positive sentiment for 12933 and 11826 cases respectively, while incorrectly predicting 3069 negative sentiments and 4172 positive sentiments. *This indicates that model is having difficulty in detecting tweets with positive sentiment.*

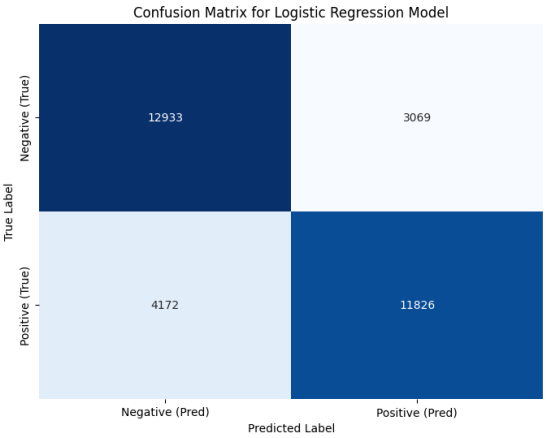

Figure 6: LSTM Confusion Matrix

- **Classification Report**: Precision, recall, and F1-score for negative sentiments are 0.76, 0.81, and 0.78 respectively; for positive sentiments, they are 0.79, 0.74, and 0.77 respectively. *This indicates that there is a minor imbalance between negative sentiment and positive sentiment class' metrics.*

3.3 Comparison

3.3.1 Which model performed better?

1. Logistic regression outperformed LSTM by **3% in accuracy**

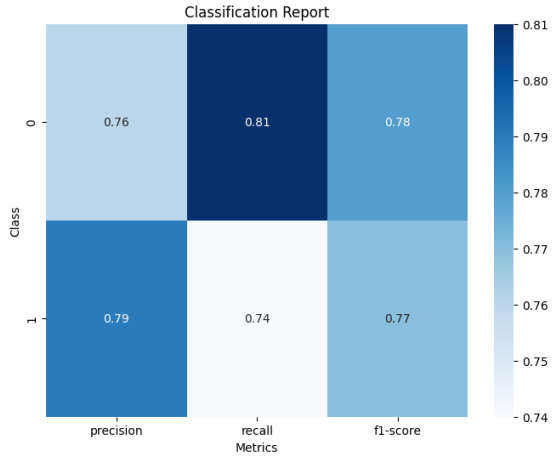

Figure 7: Random Forest Classification Report

2. Accuracy shows LSTM is struggling to train on a **small dataset**.
3. Confusion matrix shows LSTM is struggling with **positive tweets**.

3.3.2 What can be done better?

1. Larget Dataset
2. Better pre-processing i.e. tokenization
3. Strong regularization for LSTM

4 Conclusion

4.0.1 Scientific Bottlenecks

1. Small dataset
2. Uncleaned tweets

4.0.2 Techniques used to overcome

1. Applied several preprocessing techniques
2. Used simpler model (i.e. logistic regression)