ALBUKHARY INTERNATIONAL UNIVERSITY

Course Code	**CCS3153**	Course Name	**Natural Language Processing**
Lecturer	**Assoc Prof. Ts Dr Nik Adilah Hanin Zahri**		
Semester / Year	**7/2025**	Submission Date	**01/08/2025**
Assessment	**Project**	Weightage	**20%**

TITLE: NLP EMOTION CLASSIFIER USING DISTILBERT

STUDENT'S NAME	**STUDENT ID**
SHAHABA ALAM	AIU22102004

Table of Contents

1.0 Introduction

1.1. Introduction to the Corpus

The "Emotions" dataset represents the data used in this project, and it is a public dataset found on Kaggle. The dataset is specially structured for sentiment analyses, emotion classifications, and textual mining-related applications in the case of social media communicative environments. This dataset offers deep insight into the many emotional states that Twitter messages can convey.

This dataset is purely textual data scraped from Twitter messages and was labeled for each emotion. Thus, these data labels are supposed to define the expressiveness of every tweet within a discrete scale: sadness, joy, love, anger, fear, and surprise; hence, these texts have one of these emotive categories represented in numeric format from 0 to 5. Such a dataset has become an excellent source of training models that could teach how to find and categorize emotional content on text.

This dataset consists of 416,809 rows, each with two columns. The first column represents the "text" attribute, that is, the content of the Twitter message. The second column is for the labels, describing the primary emotion expressed by the tweet, from a range between 0 for sadness and 5 for surprise. The data is structured in a pandas.DataFrame to ensure it can be manipulated and processed with ease for model training. There are no null values in this dataset, hence highly complete and reliable for analytical purposes.

Objective of Sentiment Analysis Task

This Sentiment Analysis Task is being undertaken with a key purpose: the development of a system which will categorize data from texts according to predefined emotional classes. For the said project, a usage of higher-end NLP techniques, coupled with advanced machine learning models, has been incorporated with an intention to systematic analyses and then classification of textual data, exactly of Twitter messages' emotional content into six distinct feelings-sadness, joy, love, anger, fear, and surprise.

Goals:

- To refine and expand data preprocessing methods to improve the quality of input data for the sentiment analysis model.
- To enhance the precision and accuracy of the emotion classification model through advanced machine learning techniques.
- To create an intuitive and engaging user interface that allows users to easily interact with the sentiment analysis tool.

Scope of the Task:

- The project will focus on refining existing data preprocessing techniques to enhance the quality and integrity of the input data.
- Machine learning techniques with pre-trained hugging face models will be employed to develop and fine-tune the emotion classification model.
- A major component of the project will be designing and developing a user-friendly interface that facilitates easy interaction with the sentiment analysis tool.

2.0 Framework Design

2.1. Overview of the Analytical Model

Our analytical framework for emotion detection is structured in such a way that it efficiently processes and classifies textual data through an end-to-end approach: from data preprocessing to cleaning and normalization of text, advanced NLP techniques for feature extraction, and a mix of machine learning and deep learning models for robust emotion classification. It involves a very rigorous evaluation method, such as precision, recall, and F1-score metrics, with confusion matrices to give an in-depth analysis of the performance. Further, we integrate a user-friendly interface into the model that can easily interact and provide real-time classification results to increase user engagement and understanding of emotional contexts in social media texts.

2.2. Preprocessing Methods

Cleaning the text means the important part of preparing our data for analysis. It consists of the application of regular expressions with the purpose of taking out superfluous text elements. This means stripping the HTML tags, URLs, emojis, numeric values, and non-alphanumeric characters from the dataset. These operations are crucial in making sure that extra information that might lower the ability of a model to perform accurate emotional content analysis does not appear in the text data, thus increasing the precision and speed with which the model executes.

Normalization is one big preprocessing function that our pipeline performs in order to make the input texts uniform, include all data in lowercase format, and remove excessive spaces. In this regard, it might further help in accomplishing the consistency for the input data provided to the models. Standardization of text normalizes variability in data input because of format or other structural differences; therefore, these cleaned data may turn out to be uniform.

Lemmatization in our preprocessing workflow reduces words to their base or dictionary forms using efficient linguistic algorithms provided by SpaCy. This helps the model focus on semantic meaning rather than morphological differences between words. Converting the different words "running," "ran," and "runs" into their lemma "run" decreases the feature space the model has to handle, thereby increasing the efficiency in processing and performances of the model.

Stop words removal can be considered a major step in cleaning up the dataset to filter out words that do not add much value to the analysis, such as "the," "is," and "and." We make use of the extensive list of stop words provided by SpaCy, which helps in streamlining the text in a way that the model can focus on the most effective and meaningful parts of the text. This step effectively reduces data noise and enhances the focus and efficiency of the model in detecting emotions.

2.3. Feature Engineering in NLP

BoW is a well-known representation of text data that finds its applications in natural language processing. Text data in this representation are represented based on the frequency of each word appearing in a document without taking into consideration the order of words. This representation technique transforms raw text into a fixed-length set of features, where every feature corresponds to a unique word in the corpus. Each feature corresponds to the frequency of a word in a given document; that allows the model to extract information about word frequencies to study and classify texts.

In the implementation, CountVectorizer is used from sklearn to transform preprocessed text data into a BoW model. An instance of CountVectorizer is created, then fitted to the column processed_text of the DataFrame holding the preprocessed text. This includes developing a vocabulary of known words, and the documents will be transformed into vectors of features.

TF-IDF is a numerical statistic that is supposed to reflect how important a word is to a document in a collection or corpus. It extends the BoW model by not only counting the frequency of the words but giving weight to words that occur frequently in a particular document and not across the documents. It helps in differentiating the importance of words in certain contexts and dampens the effect of frequently used words that may not carry much meaning. TF-IDF is implemented using the TfidfVectorizer from scikit-learn, which converts a collection of raw documents to a matrix of TF-IDF features.

2.4. Classification Algorithms

Logistic regression is a statistical model that is very common for both binary and multi-class classification problems. This model performs its classification by estimating probabilities using the logistic function. The logistic function takes any real-valued number and maps it onto a value between 0 and 1, which then can be treated as a probability to be part of a certain class. It works great for cases where a linear decision boundary is sufficient because it is relatively easy to learn and provide reasonable results while being very interpretable.

Naive Bayes Classifier is a family of simplistic probabilistic classifiers that is based upon Bayes' theorem but with strong independence assumptions among features. They are really extremely fast and efficient on very big data and even on many kinds of text classifications. These tend to perform remarkably well in such situations even being simplistic in design, most the time producing the best models under its own assumptions.

DistilBERT is a distilled version of BERT, lighter and faster, while retaining most of the key properties of BERT. "Base-Uncased" means the version that handles text in lowercase, i.e., it does not differentiate between uppercase and lowercase; thus, it is suitable for tasks where case does not carry semantic importance. The purpose behind DistilBERT is the reduction of the model size to make it faster in processing, and therefore suitable for computational resource-constrained environments where performance can't be drastically compromised.

2.5. Handling Class Imbalance

To handle class imbalance in our dataset, we will employ the Synthetic Minority Over-sampling Technique, or SMOTE. The SMOTE generates synthetic samples for the minority classes aiming at a class-balanced distribution. In creating new artificial instances of samples from minority classes, SMOTE makes sure all the classes in any one dataset have approximately equal representation. This balanced distribution is important because it prevents the model from getting biased toward the majority class and enhances its capability of predicting the outcomes of the minority class more accurately, thus making the classification performance much more effective and fair.

2.6. Evaluation Methods

In assessing our models, we use different important metrics and visualization techniques in order to critically analyze performance. We use Precision, Recall, F1-Score, and Support as the main metrics. These metrics describe the accuracy of the model to predict each class, the balance between sensitivity and specificity, and harmonic mean of precision and recall, respectively, thereby giving a wide view of the performance of the model. Support is the number of actual occurrences of each class present in the test dataset, thus giving context to the metric scores.

For visualization, Confusion Matrices are carried out, which show a visual sense of model performance for each class. These include the numbers of correct and incorrect predictions broken down by class, understanding which classes are predicted with good accuracy and which classes have the potential to give errors. Thus, this combination of approaches-depth with metrics, visually-allows us to go through the comprehensive evaluation of our classification models.

2.7. Framework Diagram

The analytical pipeline in figure 2.0 begins with data collection, where relevant data is gathered; this acts as the backbone for the entire project. Some of the preprocessing steps involved include lemmatization, which makes words simple; for example, tokenization has to do with breaking up text or speech into bits for manageable portions. After normalizing the text, frequent stop words should be gotten rid of and shift focus on better terms that mean more in the text. These preprocessed data then go into feature engineering, where word embeddings like Word2Vec, GloVe, BERT, TF-IDF, and Bag of Words transform the text into numerical formats understandable by machine learning models.

The pipeline also involves a stage to handle class imbalance, balancing the dataset so that it does not bias toward classes with higher frequencies. Then, the train-test split method will divide the data into separate sets to train and test the model. This approach applies a range of classification algorithms: Decision Tree, DistilBERT, Naive Bayes, and Logistic Regression. It looks for the most effective model on this task. Each of these models has undergone optimization to fine-tune its parameters for the best performance.

The process gets completed with the evaluation of the models, where these are assessed basis certain metrics such as accuracy and F1-score on their effectiveness. A completion check follows that decides if the model meets the required standards or if more adjustments have to be made. In the case that happens in iteration until the model performs well, it reaches the return point for deployment or for further refinement on the basis of outcomes. This structured approach ensures a comprehensive analysis and development of a robust model, aiming for high accuracy and reliability in practical applications.

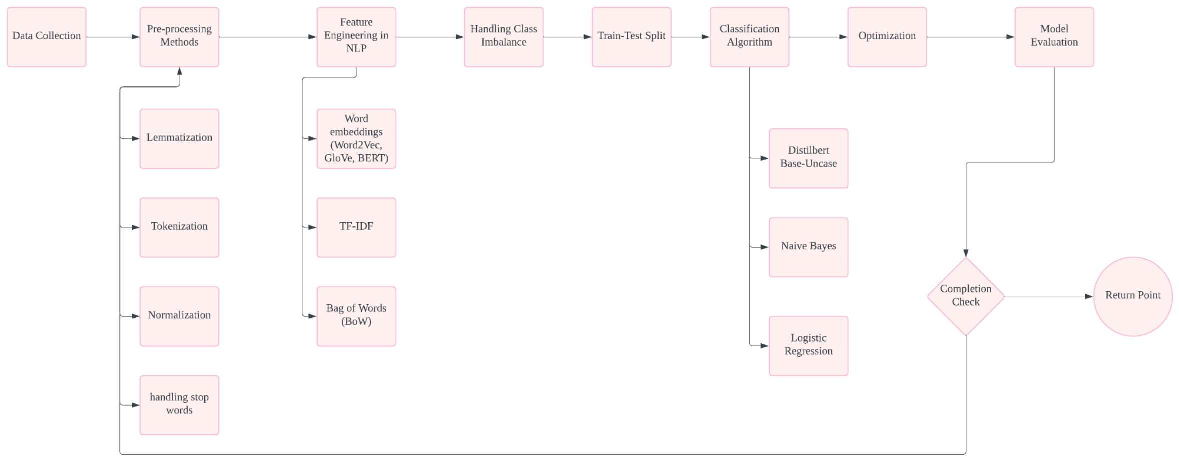


Figure 2.0: Steps in Building an Emotion Detection Model

3.0 Implementation

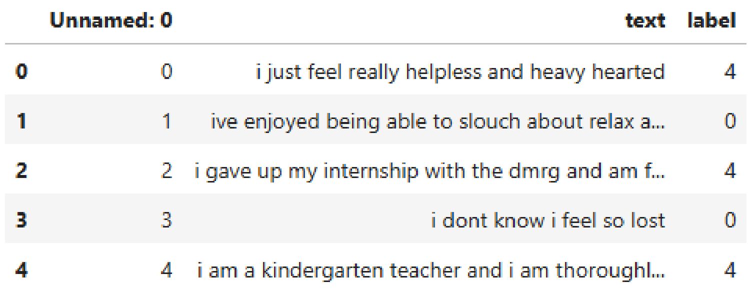
3.1. Overview

In the following, we describe how we apply the emotion detection system according to the detailed framework of Figure 2.0: First, data preprocessing by applying complex NLP methods. Adding to this, it moves to feature engineering by using techniques like TF-IDF and embedding, moving to the training of different types of the ML models like Logistic Regression and DistilBERT. A lot of model evaluation is performed by using precision metrics and confusion matrices. In parallel, we offer a friendly and interactive interface for real-time analysis so that the system becomes optimized, easily accessible, and highly effective in emotion classification.

3.2. Code Implementation

3.2.1. Data Acquisition and Initial Processing

In the first step of our sentiment analysis project, we attained and preprocessed our dataset using the Kaggle API, then used pandas to load and structure it. A preliminary look at the dataset gives an overview of its structure-a fast view of the first few entries is documented as Figure 3.0 and a detailed summary of data types and columns. We then convert these numerical labels to descriptive categorical labels of the emotions, such as sadness, joy, and fear, using a predefined mapping; this greatly improves readability in our analyses. Thus, this preparation provides a well-structured dataset that will be useful later in the deeper analysis, laying a solid foundation for subsequent stages of the project.



	Unnamed: 0	text	label
0	0	i just feel really helpless and heavy hearted	4
1	1	ive enjoyed being able to slouch about relax a...	0
2	2	i gave up my internship with the dmrg and am f...	4
3	3	i dont know i feel so lost	0
4	4	i am a kindergarten teacher and i am thoroughl...	4

Figure 3.0: First few rows of our dataset

3.2.2. Data Preprocessing

In the data preprocessing phase, depicted in Figure 3.1 as a heatmap of missing values, we've employed comprehensive techniques to ensure the dataset is pristine and well-prepared for analysis. Firstly, we visualize missing data and manage duplicates using duplicate_rows to check. We removed 686 exact duplicate rows to refine our dataset to 416,123 entries.

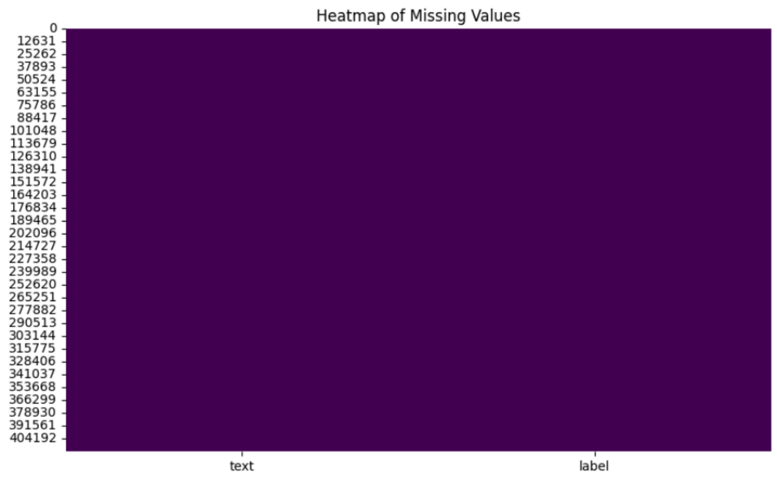


Figure 3.1: Plotting missing values

Data Cleaning: Implementing regex-based cleaning, normalization, lemmatization, and stop words removal.

3.2.3 Text Lengths

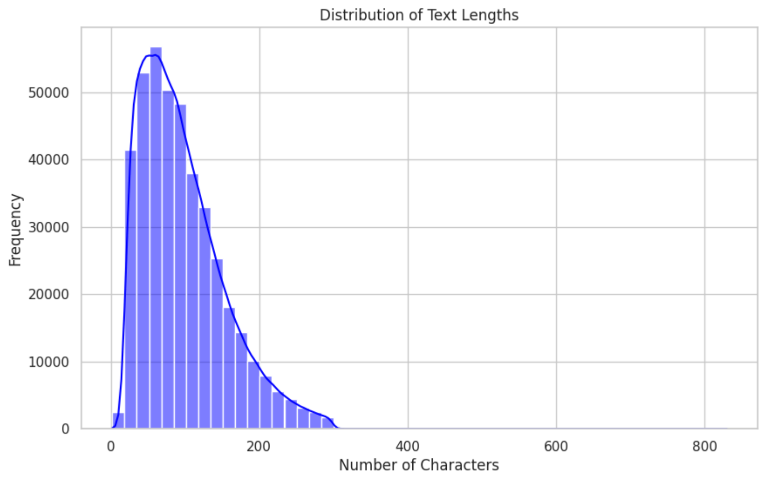


Figure 3.2: Distribution of text lengths

Figure 2.3 displays the distribution of text length by character number of entry within the dataset. The histogram shows the general feature of how the text length variable is distributed; it includes the KDE curve fitted through Kernel Density Estimation. Positive skewness implies the majority of entries are shorter, and hence the most common range for all entries would range from 50 to 100 characters. This, therefore, means that the greater majority of text entries are succinct in nature. Longer texts above 400 characters are realized, these happen very much fewer times by the tapering tail of this distribution. A comprehension of this distribution is quite valuable during preprocessing in machine learning with text for informing decisions about whether truncation or padding is required for compliance with algorithm input, mainly neural networks.

3.2.4 Handling Outlier

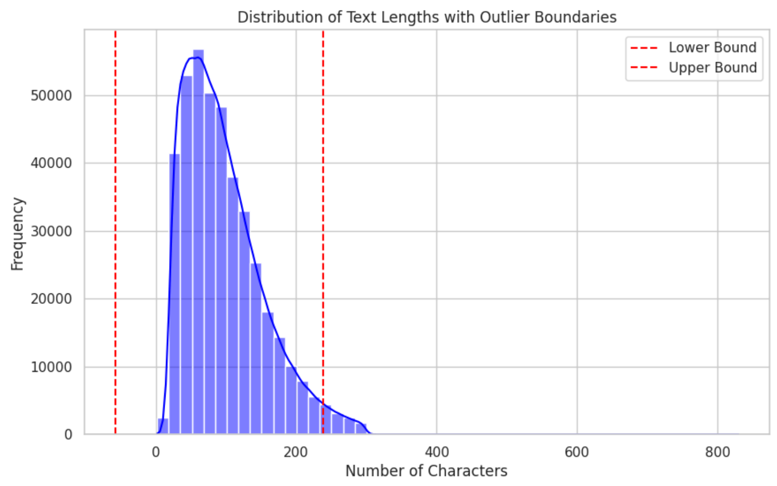
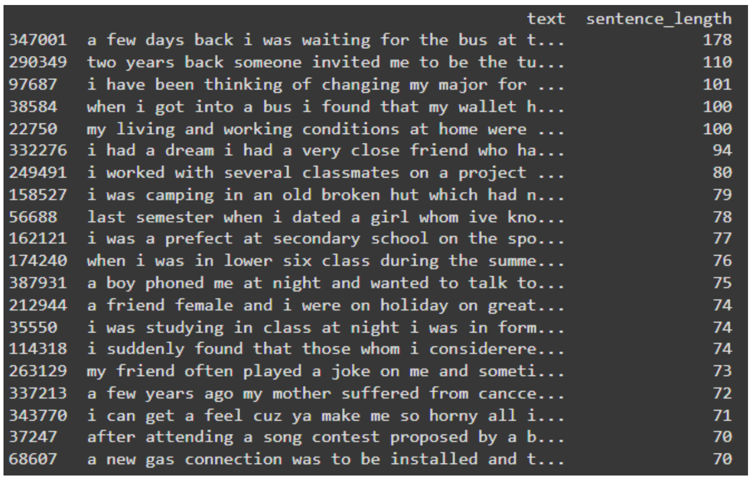


Figure 3.3: Identify outliers

The figure shows the distribution of text lengths in the dataset, with outlier boundaries marked by red dashed lines. These boundaries are determined using the Interquartile Range (IQR) method, where the lower bound is $Q1 - 1.5 \times IQR$ and the upper bound is $Q3 + 1.5 \times IQR$. Most text lengths fall within these bounds, with a peak around 50–100 characters, indicating that the majority of entries are concise. The distribution is positively skewed, with a small number of extremely short or long text entries identified as outliers.

Outlier detection is essential in this study to improve model performance and data quality. Outliers can introduce noise, bias, and inconsistency, potentially leading to overfitting or suboptimal model behavior. By identifying and managing these entries, the dataset becomes more representative and reliable, ensuring that machine learning models focus on meaningful patterns in the majority of the data. This process also enhances preprocessing strategies and ensures better generalization for emotion detection tasks.



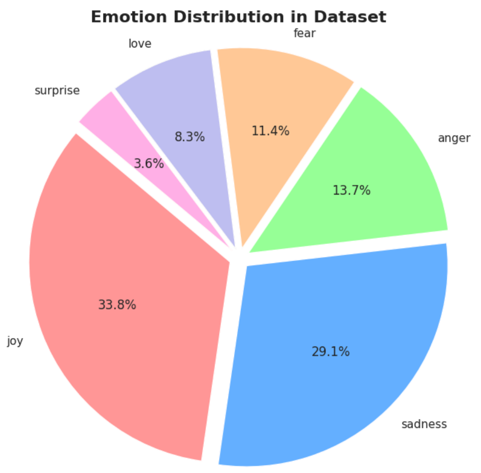
```
                                                   text  sentence_length
347001  a few days back i was waiting for the bus at t...              178
290349  two years back someone invited me to be the tu...              110
97687   i have been thinking of changing my major for ...              101
38584   when i got into a bus i found that my wallet h...              100
22750   my living and working conditions at home were ...              100
332276  i had a dream i had a very close friend who ha...               94
249491  i worked with several classmates on a project ...               80
158527  i was camping in an old broken hut which had n...               79
56688   last semester when i dated a girl whom ive kno...               78
162121  i was a prefect at secondary school on the spo...               77
174240  when i was in lower six class during the summe...               76
387931  a boy phoned me at night and wanted to talk to...               75
212944  a friend female and i were on holiday on great...               74
35550   i was studying in class at night i was in form...               74
114318  i suddenly found that those whom i considerere...               74
263129  my friend often played a joke on me and someti...               73
337213  a few years ago my mother suffered from cancce...               72
343770  i can get a feel cuz ya make me so horny all i...               71
37247   after attending a song contest proposed by a b...               70
68607   a new gas connection was to be installed and t...               70
```

Figure 3.4: Top 20 Longest Sentences

After analyzing the dataset, we applied two specific filtering steps: enhancing data quality and focusing on meaningful text entries. First, the text entries whose sentence lengths are 1, 2, or 3 were identified and removed from the dataset. Such extremely short entries are unlikely to provide enough contextual information for meaningful emotion analysis. This step in filtering cleans the dataset of text entries with substantial content to support robust machine learning and reduces the noise that too short sentences may introduce. Following that, the index was reset on the filtered dataset for coherence and cleanliness, and this updated dataset was saved for later use.

Further, we located and eliminated a small subset of the top 20 longest sentences of the dataset. These include the entries with a sentence length of 178 and others above 90, which are considered as potential outliers due to their extremely long nature. Extremely long sentences can distort model training since they may have an outsized influence or introduce patterns not representative of the majority of the data. The result is a more balanced dataset representative of common lengths of text; removing these entries thus further increases the quality of this dataset for emotion detection tasks.

3.2.5 Distribution of Emotions



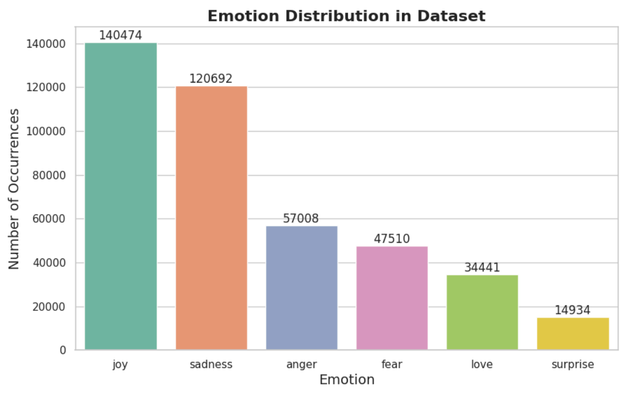


Figure 3.3: Emotion Distribution in the Dataset

The two visualizations together give an overview of the distribution of emotions in the dataset. The first bar chart presents the absolute frequency of each class of emotion-joy, sadness, anger, fear, love, surprise-by count, while the second pie chart gives a complementary overview of the proportional distribution of the emotions in percent.

These visualizations show that joy is the most frequent emotion (33.8%), followed by sadness (29.1%), while surprise is the least represented class (3.6%). This is very important for the study because understanding the distribution of the emotions points out an imbalance between classes, which might affect the performance of the machine learning models. Balancing such imbalance with the resampling technique makes sure that each class gets an equal say, hence letting the models generalize well without being biased toward dominant classes. Further, these insights guide data preprocessing, modeling decisions, and evaluation strategies.

Indeed, the bar chart and pie chart clearly show that the data is imbalanced, with dominant emotional classes like joy at 33.8%, sadness at 29.1%, and underrepresented ones like surprise at 3.6%. Class imbalance will result in biased machine learning models doing well on dominant classes, not the minority ones. This visualization hence helps in finding such imbalances that can be treated with appropriate preprocessing techniques, like oversampling using SMOTE or class weighting.

3.2.6. Word Cloud

The word cloud visualizes the most frequently occurring words in the fear class of the dataset. It provides an intuitive way to identify key terms associated with this emotion. Larger words represent higher frequencies, indicating that they appear more often in the text entries labeled as fear. For example, words like "feel," "think," "want," "little," and "time" dominate the word cloud, suggesting they are central to the emotional context of fear in the dataset.

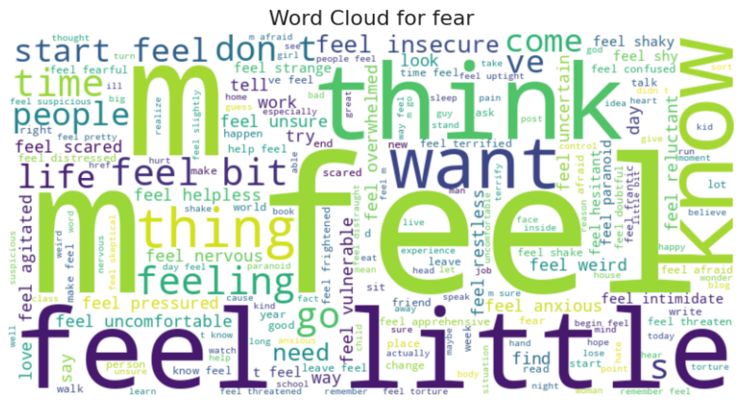


Figure 3.4: Word Cloud for Emotion Fear

This visualization highlights the linguistic patterns and vocabulary associated with specific emotions, offering valuable insights into how people express fear. Such patterns can be used to guide feature selection, improve text preprocessing, and inform model development for emotion detection tasks.

3.2.7. Feature Extraction

Bag of Words (BoW): We have used the BoW model, through an implementation of the CountVectorizer of scikit-learn; this being a pretty common approach in NLP to translate textual data into numerical. It starts initializing a CountVectorizer that processes texts to produce a matrix of token counts-the mathematical representation through which each document gets transformed into a vector by word occurrence frequency. This will use the fit_transform method on the 'processed_text' column of a DataFrame to learn the vocabulary and encode the text as a sparse matrix of integers. In this case, X_bow.shape will give the number of documents and the extent of the vocabulary learned from data. This is a simple approach that allows text data to be processed by classic machine learning algorithms which will focus on word frequencies while losing information about word order.

TF-IDF: From this, we came up with the Bag of Words model, for which we then used the Term Frequency-Inverse Document Frequency, TF-IDF model, using the TfidfVectorizer from scikit-learn. The TF-IDF is a statistical method that analyzes a word's importance in a specific document against an entire corpus. In such a way, weights are changed such that weightier imports are given to words that are frequent in the documents, yet not as frequent across different documents. We have applied TfidfVectorizer to the 'processed_text' column of our DataFrame using default settings. This gave us a matrix where each row represents a document and each column represents a unique word, with values in the matrix representing TF-IDF scores. The feature matrix X_tfidf that resulted was of shape (415,059, 62,325), meaning our dataset consists of 415,059 documents and a vocabulary of 62,325 unique words. It forms a full but efficient representation matrix of text data, which is appropriate for subsequent machine learning tasks.

3.2.8. Handling Class Imbalance

We applied SMOTE-which is the Synthetic Minority Over-sampling Technique-imbalance treatment from the imblearn.over_sampling module on a given dataset. It creates synthetic samples to create a balanced class distribution instead of just duplicating existing samples of the minority class. A SMOTE instance is initialized, where a certain random_state was used for reproducibility:. This approach first applies the fit_resample to features matrix and labels vector, effectively performing actual change in the class distribution of this dataset through the over-sampling of the minority class based on similarities in feature space. Afterwards, it prints out the shape and class distribution of both original and the resampled datasets so it would become clear by a comparison what a difference in the balance of classes it makes when actually applying SMOTE.

Table 3.0: Class Distribution Before and After SMOTE Application

	Dataset Shape	**Class Distribution**
Original Dataset	(415,059, 62,325)	'joy': 140,474, 'sadness': 120,692, 'anger': 57,008, 'fear': 47,510, 'love': 34,441, 'surprise': 14,934
Resampled Dataset	(842,844, 62,325)	'fear': 140,474, 'sadness': 140,474, 'love': 140,474, 'joy': 140,474, 'surprise': 140,474,

		'anger': 140,474

Initially, the dataset consists of 415,059 samples and 62,325 features. The class distribution is notably imbalanced, with 'joy' being the most prevalent label (140,474 instances) and 'surprise' the least (14,934 instances). After applying SMOTE, each class in the resampled dataset has an equal count of 140,474 instances, totaling 842,844 samples. This equalization is achieved by synthesizing new samples in the minority classes based on the features of existing samples, thus providing a more balanced dataset for model training which could enhance the performance and fairness of predictive models.

3.3. Machine Learning Models

First, to prepare the dataset for machine learning, label encoding was performed on categorical labels to change them into a numerical format. The reason for this is that machine learning algorithms require the input of numeric values for the target variable. The LabelEncoder from Scikit-Learn was used for the transformation of emotion labels such as joy, sadness, and anger into their corresponding integer forms like 0, 1, and 2, respectively. Such a transformation has made the dataset appropriate for the models, but with the assurance that the original categorical information is retained and can always be retrieved if needed.

The dataset has been divided into training and test subsets using the Scikit-Learn function train_test_split to effectively evaluate the different models. This split was to take 80% for the training set and 20% for the testing set. In case of any class imbalance, the stratify parameter will ensure that the distribution of classes in both the training and testing sets is the same as in the original dataset. This approach has been important to keep a representative sample of each class in both subsets to help the models generalize during evaluation.

Three machine learning models were trained to predict emotions from text data: Logistic Regression, Naive Bayes, and DistilBERT.

3.3.1 Logistic Regression

Logistic Regression is a linear model generally used in classification, including multi-class problems like emotion detection. It predicts probabilities for each class using the softmax function, hence it allows it to handle multiple classes effectively. In this study, the training dataset for the Logistic Regression model was obtained by resampling due to the class imbalance problem. This involves fitting the model parameters to maximize the likelihood of observing the true labels given the input features. The parameter max_iter=1000 ensures convergence with adequate iterations. A classification report including key evaluation metrics includes precision, recall, F1-score, and support that together give insight into the performance of the model across different emotion categories. Another reason for the popularity of Logistic Regression is that it offers interpretability, computational efficiency, and feature importance. Although it works well for emotion detection tasks, the linear nature might be limiting for complex and nonlinear patterns; further research of more advanced models is required.

3.3.1 Naive Bayes

Multinomial Naive Bayes is a probabilistic classifier based on Bayes' theorem, and it is designed especially for discrete features that are usually shared in general by text classification tasks, such as word counts or term frequencies. Additionally, this classifier assumes that features are conditionally independent given the class label. It is computationally efficient and easy to implement.

The Multinomial Naive Bayes model was then trained on the resampled data in order to handle class imbalance. It infers class probabilities by learning the likelihood of a feature given a class and the prior of the classes. This approach is very effective for datasets whose features are word frequencies, for example, TF-IDF or Bag-of-Words vectors.

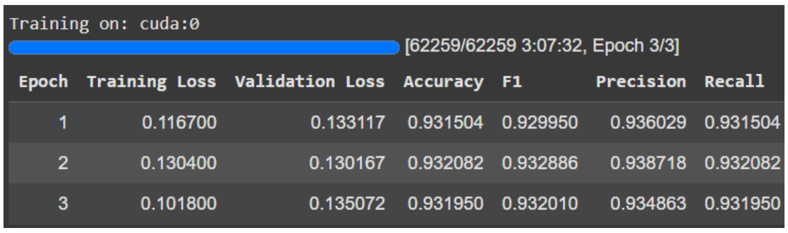
Multinomial Naive Bayes is apt for emotion detection tasks as it utilizes word/term frequency for class discrimination among different classes of emotions. Although the model is simple and efficient, the independent feature assumption in it may at times restrict it from capturing complex dependencies among features. Often, however, it provides a strong baseline for text classification due to its ease of implementation and good performance.

3.3.1 DistilBERT

DistilBERT is a compact and efficient version of the BERT model. It is designed to retain 97% of BERT's performance with 40% fewer parameters and running 60% faster. Therefore, DistilBERT will be the best choice for big NLP jobs, such as emotion detection, where computational efficiency and memory are so important.

Now, the data has been split into training and testing sets; the text feature tokenized, using distilbert-base-uncased by Hugging Face, was pad, truncate, and convert texts into input ids and attention mask accordingly. Label encoding should be done for matching model criteria using LabelEncoder. Load pre-trained DistilBERT model and its tokenizer, with its number of labels corresponding to multi-class classification matching emotion classes of this dataset have started doing finetuning on the pre-trained DistilBERT model. Training was done on a batch size of 16, learning rate of $2e - 5$, and weight decay of 0.01 to avoid overfitting, for 3 epochs with evaluation after each epoch. The best model was chosen using the F1-score. Training, evaluation, and optimization were performed using the Hugging Face Trainer API, while performance metrics-accuracy, precision, recall, and F1-score-were computed using a custom compute_metrics function.

The model has been trained to predict the emotion labels of the test set by analyzing the tokenized text inputs. DistilBERT captures the context of the words and phrases, hence turning in very effective for emotion detection tasks where nuances in language are crucial to distinguish between emotions like joy, sadness, and anger.



```
Training on: cuda:0
[62259/62259 3:07:32, Epoch 3/3]
```

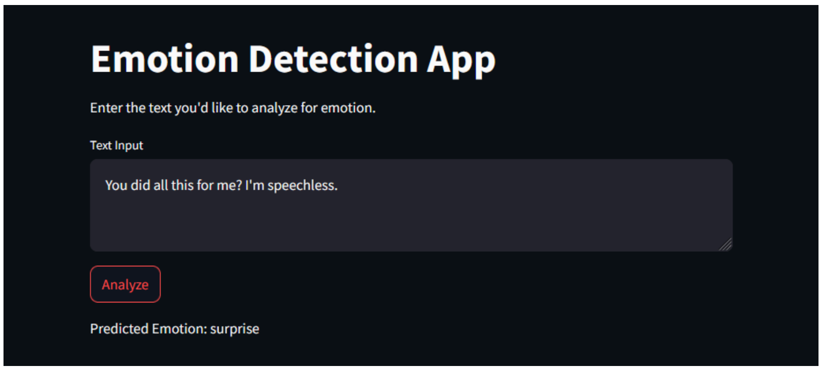
Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.116700	0.133117	0.931504	0.929950	0.936029	0.931504
2	0.130400	0.130167	0.932082	0.932886	0.938718	0.932082
3	0.101800	0.135072	0.931950	0.932010	0.934863	0.931950

Figure 3.4: Training and Validation Metrics for DistilBERT

Figure 3.4 summarizes the training and validation performance of the DistilBERT model over three epochs. Indeed, all key metrics of training and validation loss, accuracy, F1-score, precision, and recall keep improving. The training loss is linearly going down, while the validation loss remains very low, hence the model learns effectively with good generalization. The final accuracy is 93.19% with an F1-score of 0.932, showing a strong balance between precision and recall. Both precision and recall are over 93%, which points to the model's reliability in identifying emotion classes with minimal errors.

3.4 Deployment Using Streamlit

The emotion detection model was deployed as a web application using Streamlit, a lightweight and user-friendly framework for creating interactive web apps. This deployment enables real-time emotion analysis of user-provided text inputs, making the model accessible to end-users through an intuitive interface. Key Features include: Interactive input, Prediction and output which uses our pre-trained model as shown in our figure 4.1 below.


Emotion Detection App

Enter the text you'd like to analyze for emotion.

Text Input

You did all this for me? I'm speechless.

Predicted Emotion: surprise

Figure 4.2: Streamlit Interface

4. Evaluation and Analysis

4.1. Sentiment Scores Visualization

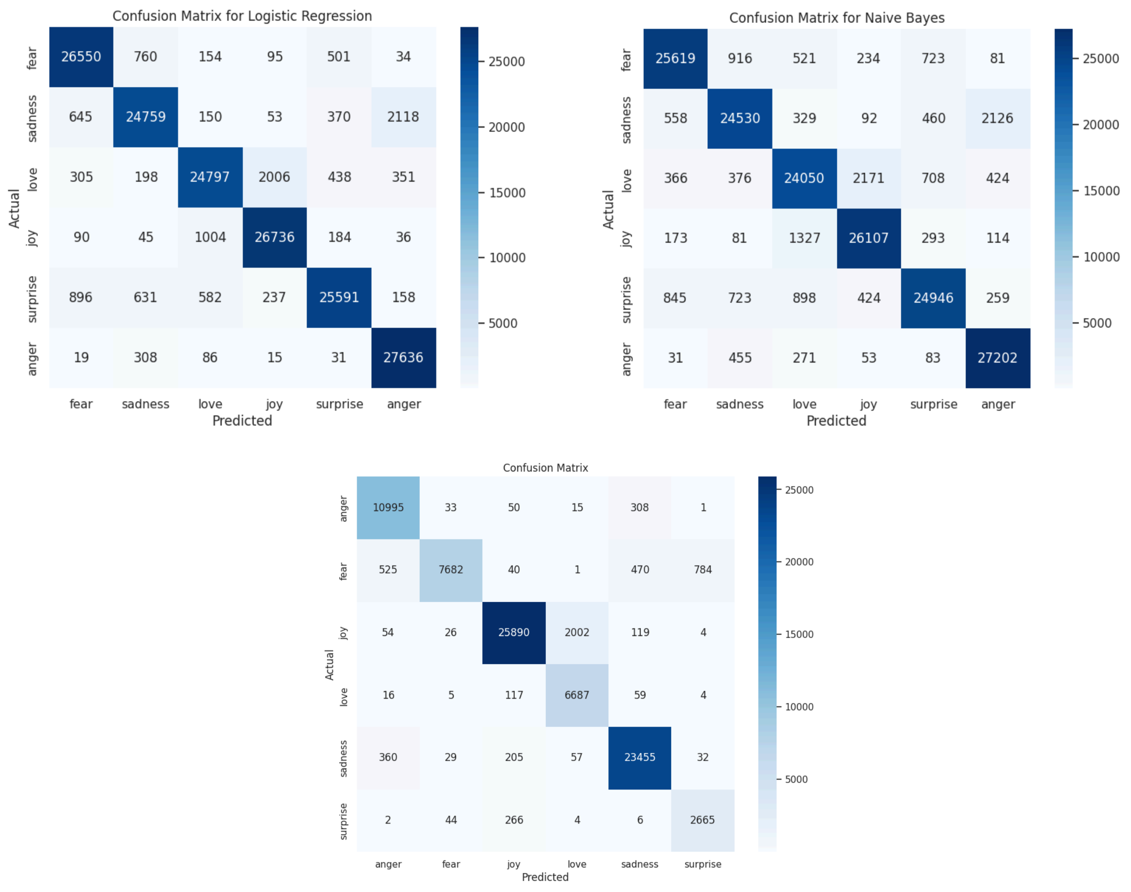


Figure 4.0: Confusion Matrix for Models

Figure 4.0 shows stark features of relative strengths and challenges for each classifier in the performance comparison between Naive Bayes, Logistic Regression, and DistilBERT on the emotion classification task. Naive Bayes has the best performance for the class "joy" but poorly distinguishes classes "fear" and "anger", hence needs improvement under the feature independence assumption. Logistic Regression does a pretty good job of classifying joy and surprise but commits errors, just like other models, in confusing emotions with closely related sentiments, such as love and joy. By utilizing more refined neural network architecture, DistilBERT outperforms most of the models on most categories of emotions, reflecting a strength in dealing with complex classifications. However, it also struggles with challenges of similar positive emotions, which are indicative of areas where further model refinement could yield improvements.

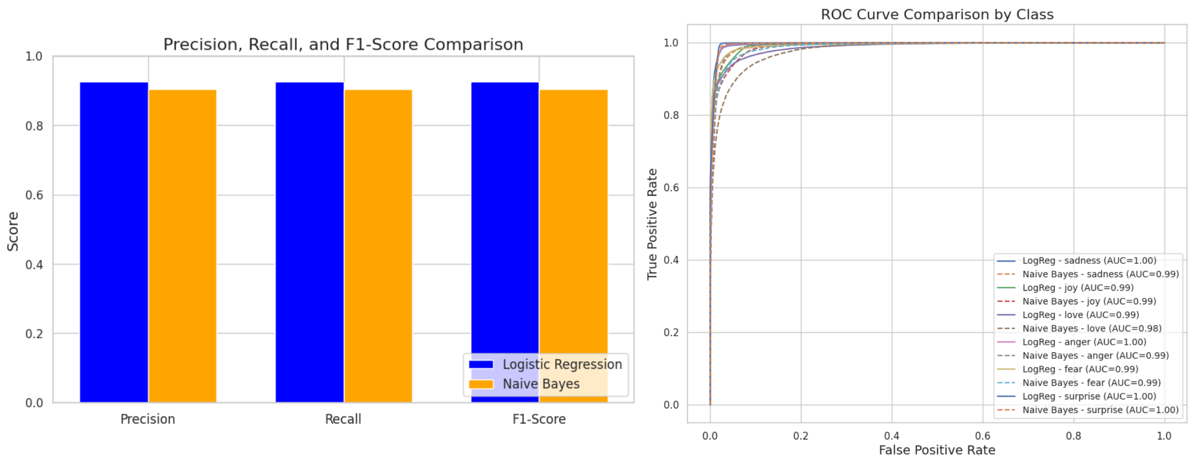


Figure 4.1: Precision, Recall, F1-Score and ROC Curve Comparison

This bar chart compares the precision, recall, and F1-score of the Logistic Regression and Naive Bayes models. Both models demonstrate very strong performance in all three metrics, although Logistic Regression is slightly better than Naive Bayes. Logistic Regression outperforms Naive Bayes on precision, recall, and F1-score, reflecting better handling of the multi-class emotion detection task. This chart underlines the fact that Logistic Regression is robust in balancing false positives and false negatives while preserving strong predictive power.

The ROC curve figure complements the bar chart shown below, displaying the Receiver Operating Characteristic (ROC) curves for each of the emotion classes across both Logistic Regression and Naive Bayes models. Each curve plots the trade-off between the TPR and FPR for every class. AUC values further support this classification performance of each class, and the closer to 1.0, the more predictiveness there is within the values. Logistic Regression has, in most of the classes, presented an AUC that is slightly higher than Naive Bayes, thus proving a better capability to distinguish between emotion classes.

4.2. Comparative Analysis of Machine Learning Methods

Table 4.0: Classification Report for all models

Model	Emotion	Precision	Recall	F1-Score	Support
Naive Bayes	Anger	0.93	0.91	0.92	28094

	Fear	0.91	0.87	0.89	28094
	Joy	0.88	0.86	0.87	28094
	Love	0.90	0.93	0.91	28094
	Sadness	0.92	0.89	0.90	28094
	Surprice	0.99	0.97	0.93	28094
Logistic Regression	Anger	0.93	0.95	0.94	28094
	Fear	0.93	0.88	0.90	28094
	Joy	0.93	0.88	0.90	28094
	Love	0.92	0.95	0.93	28094
	Sadness	0.94	0.91	0.93	28094
	Surprice	0.91	0.98	0.95	28094
DistilBERT	Anger	0.92	0.96	0.94	11402
	Fear	0.98	0.81	0.89	9502
	Joy	0.97	0.93	0.95	28095
	Love	0.76	0.97	0.85	6888
	Sadness	0.96	0.97	0.97	24138
	Surprice	0.76	0.89	0.82	2987

The Naive Bayes model results in 90% overall accuracy, with a precision of 0.90, recall of 0.90, and F1-score of 0.90 on average. This model is very consistent among all emotion classes, particularly excelling in anger with a high F1-score of 0.92 and surprise at 0.93. However, it gives slightly lower recall for the emotions of fear and joy, at 0.87 and 0.86, respectively. That, in turn, suggests that the Naive Bayes, though powerful, sometimes lacks the capturing of some instances related to both fear and joy-probably because of the assumption of independence of features that might be a gross simplification with text data.

Logistic Regression outperforms Naive Bayes on all fronts: an overall accuracy of 93% with a macro average precision, recall, and F1-score of 0.93. This is the best performance among all classes, with surprise being exceptionally high at 0.95 F1-score and sadness at 0.93 F1-score. It outperforms Naive Bayes for almost all emotions, especially for Joy and Fear, at recall values of 0.88 in both, hence capturing the subtleties of these emotions better. Thus, this model is generally well-balanced for the task at hand.

DistilBERT, with its deep learning architecture, results in an overall accuracy of 93% with a macro average precision of 0.89, recall of 0.92, and F1-score of 0.90. It gives very high precision and recall for sadness (F1-score: 0.97) and anger (F1-score: 0.94). For love and surprise, it performs slightly worse, yielding a lower precision of 0.76 each. Despite this, its ability to handle complex language patterns gives it an edge in nuanced emotional analysis, making it particularly effective for capturing subtle emotions like sadness and anger.

Naive Bayes, Logistic Regression, and DistilBERT have different strengths and weaknesses regarding emotion classification. Naive Bayes works well for emotions like anger and surprise, which have clear lexical markers, but fails to capture subtle emotions like fear and joy because of its simplistic feature independence assumption. Logistic Regression has consistent performance on all emotions, especially performing the best on sadness and surprise, but struggles with closely related emotions such as love and joy. DistilBERT shows better performance in complex emotions like sadness and anger by utilizing its deep learning architecture, but struggles with love and surprise because of overlapping expressions and dependency on pre-trained embeddings.

Misclassifications across all models reveal common challenges. Emotions such as love and joy often share sentiment and vocabulary, which leads to frequent confusion. Similarly, fear and sadness are misclassified because of their shared negativity in expression, and surprise is sometimes confused with anger due to similar emotional intensity. These patterns point to a need for more nuanced handling of overlapping and subtle emotional expressions.

4.4. Critical Analysis

Our report discusses the performance comparison of three models-DistilBERT, logistic regression, and Naive Bayes-on classifying emotions within text data. Each of these algorithms has different strengths and weaknesses with respect to the performance classification of many emotional expressions. From the results of our experiment, DistilBERT turns out to be the strongest of these models, producing the highest F1-score for a number of complex emotions like sadness and anger. Logistic Regression, while generalizing quite well, like in the case of F1-scores of about, tends to excel especially in categorizing surprise and sadness, taking great advantage of the patterns in text. Naive Bayes, while posting slightly low scores, does face difficulty differentiating between these somewhat overlapping expressions of emotions and thus has its scores degraded.

More precisely, a set of steps undertaken during the cleaning-normalization-lemmatization phases has given a boost to their performances appreciably. These techniques have helped, especially for Logistic Regression and Naive Bayes, which are extremely sensitive to clean and normalized input data in order to avoid misclassifications between similar emotions like love and joy. The balancing of classes with SMOTE has been crucial and has come up with a high degree of improvement in recall scores for minority classes, allowing the models to perform better on categories of emotions that are not as frequent. With these enhancements, however, the models still remain sensitive to the quality of input data and to the artefactual patterns introduced by SMOTE, which might not represent real variability well, particularly for Naive Bayes and Logistic Regression. This therefore calls for continued development in model techniques that handle overlapping emotions, enhance contextual embeddings, and conform to the variability in real data for the overall optimization of emotion classification systems.

5. Conclusion and Future Work

This project successfully developed and evaluated an emotion detection system that classifies textual data into six emotional categories. The objectives, including refining data preprocessing, improving model performance, and creating a user-friendly interface, were effectively met. Key findings indicate that DistilBERT, a transformer-based model, outperformed traditional methods like Logistic Regression and Naive Bayes in handling nuanced and complex emotional expressions, achieving an accuracy of 93%. Logistic Regression also showed strong performance with balanced precision and recall across all classes, while Naive Bayes provided a computationally efficient baseline. The data preprocessing steps, such as normalization, lemmatization, and handling class imbalance with SMOTE, significantly enhanced the quality of the input data and improved model performance. These findings demonstrate the practical applicability of the emotion detection system in areas like social media monitoring, customer sentiment analysis, and real-time feedback systems.

To further enhance the system, several future directions can be pursued. Integrating more advanced transformer-based architectures, such as BERT or RoBERTa, could improve the system's ability to capture subtle emotional nuances and overlapping expressions. Expanding the dataset to include larger or more diverse sources, such as multilingual data or domain-specific corpora, would enhance the generalization and robustness of the models. Deployment of the system can be achieved by developing APIs or integrating the models into user-friendly platforms, enabling real-time emotion detection in practical applications. Additionally, incorporating contextual information or exploring multi-modal data (e.g., combining text with images or audio) could provide richer emotion detection capabilities, opening avenues for broader and more impactful real-world use cases. These enhancements would ensure the continued development and scalability of the emotion detection system.

6. References

Nidula Elgiriyewithana. (2024). Emotions. Kaggle.com. https://www.kaggle.com/datasets/nelgiriyewithana/emotions

samanfatima7. (2024, March 17). BiLSTM RNN Delivers 94% Accuracy. Kaggle.com; Kaggle. https://www.kaggle.com/code/samanfatima7/crushing-it-bilstm-rnn-delivers-94-accuracy

Spelmen, V. S., & Porkodi, R. (2018, March). A review on handling imbalanced data. In *2018 international conference on current trends towards converging technologies (ICCTCT)* (pp. 1-11). IEEE.

Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. *Social network analysis and mining*, *11*(1), 81.

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other subject, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that I have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.

Name : Shahaba Alam

Date : 01-05-25

Signature :

[illegible]