

|  |  |
| --- | --- |
|  |  |
|  | Exploring the Influence of Social Media on Cultural Perceptions and Attitudes: A Comprehensive Analysis and Ethical Consideration |

|  |  |
| --- | --- |
|  |  |
|  |  |

**UNIVERSITY OF HERTFORDSHIRE**

School of Physics, Engineering and Computer Science

7COM1039-0109-2022 - Advanced Computer Science Masters Project

Alen Jose: 20055011

*Supervised by: Imran Khan*

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science and Analytics at the University of Hertfordshire (UH).

It is my work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby grant permission for the report to be made available on the university website, provided that the source is acknowledged.

# 

# 

# 

# 

# 

# 

# Abstract

The study offers a thorough analysis of the complex and diversified effects of social media on individuals and communities. To limit its scope, the study primarily examines tweets about specific social issues that are trending on Twitter. Using various robust machine learning algorithms and pre-trained models, sentiment analysis is performed to better analyse the social media data. Special emphasis was placed on hyperparameter tuning to find the optimal model. An LSTM model was trained on a 1.6 million tweets dataset, and the model achieved an accuracy of 82%, which is a remarkable result.

The research uses Python frameworks and data collection methods to build and evaluate models that can effectively classify text data. The paper also explores the limitations and performance metrics of the machine learning models and provides a deeper understanding of their capacity in real-world scenarios. One of the key limitations of the research was the difficulty in capturing the specific vocabulary of data that is related to social topics. Additionally, social topics are often highly subtle and context-dependent, which can make it difficult to develop accurate and reliable methods for data analysis.

In addition to the technical contributions, the paper delves into the ethical considerations surrounding social media research. It critically examines the concepts of informed consent and questions the ethical implications of using publicly available data for academic research purposes.

# Table of Contents

[Abstract 3](#_Toc144143492)

[Table of Contents 4](#_Toc144143493)

[List of Figures 7](#_Toc144143494)

[1. Introduction 8](#_Toc144143495)

[2. Research Questions 10](#_Toc144143496)

[3. Literature Review 11](#_Toc144143497)

[3.1 Background Research 11](#_Toc144143498)

[3.2 Related Work 11](#_Toc144143499)

[3.3 Critical Analysis and Research Gap 14](#_Toc144143500)

[4. Tools and Technology 14](#_Toc144143501)

[4.1 Development Tools and Frameworks 14](#_Toc144143502)

[4.1.1 Python Frameworks 15](#_Toc144143503)

[4.2 System Specification 15](#_Toc144143504)

[5. Methodology 16](#_Toc144143505)

[5.1 Data Collection 16](#_Toc144143506)

[5.1.1 Method of Collection 16](#_Toc144143507)

[5.2 Data Pre-Processing 17](#_Toc144143508)

[5.3 Model Building and Evaluation 17](#_Toc144143509)

[5.3.1 Baseline Model 18](#_Toc144143510)

[5.3.2 Logistic Regression 19](#_Toc144143511)

[5.3.2.1 Choice of Vectorizer 19](#_Toc144143512)

[5.3.2.2 Finding the Optimal Iteration Step 20](#_Toc144143513)

[5.3.2.3 Results of Logistic Regression 20](#_Toc144143514)

[5.3.3 Stochastic Gradient Descent Classifier 20](#_Toc144143515)

[5.3.3.1 How SGD works. 21](#_Toc144143516)

[5.3.3.2 Limitations of SGD 21](#_Toc144143517)

[5.3.3.3 Model Training and Results 21](#_Toc144143518)

[5.3.4 Random Forest Classifier 22](#_Toc144143519)

[5.3.4.1 How the model works. 23](#_Toc144143520)

[5.3.4.2 Training Process 23](#_Toc144143521)

[5.3.4.3 Testing and Results 23](#_Toc144143522)

[5.3.5 Long Short-Term Memory (LSTM) 23](#_Toc144143523)

[5.3.5.1 How it works. 24](#_Toc144143524)

[5.3.5.2 Model Training 25](#_Toc144143525)

[5.3.5.3 Results 26](#_Toc144143526)

[5.3.6 BERT 28](#_Toc144143527)

[5.3.6.1 Model Implementation 28](#_Toc144143528)

[i) Stochastic Gradient Descent (SGD): 29](#_Toc144143529)

[ii) AdamW: 29](#_Toc144143530)

[5.3.6.2 Results 29](#_Toc144143531)

[5.3.7 RoBERTa 30](#_Toc144143532)

[5.3.7.1 Model Implementation 31](#_Toc144143533)

[5.3.7.2 Results 31](#_Toc144143534)

[First Run: 31](#_Toc144143535)

[Second Run: 31](#_Toc144143536)

[5.3.8 Evaluating and Choosing the Best-Fit Model for Sentiment Analysis 32](#_Toc144143537)

[5.3.8.1 Robust Generalisation 32](#_Toc144143538)

[5.3.8.2 Effective Early Stopping 32](#_Toc144143539)

[5.3.8.3 Importance of Stopwords 33](#_Toc144143540)

[5.3.8.4 Comprehensive Dataset Training 33](#_Toc144143541)

[5.3.8.5 Limitations of Pre-Trained Models 33](#_Toc144143542)

[5.4 Model Evaluation on Unlabelled BLM Tweets 33](#_Toc144143543)

[6. Ethical Considerations in Social Media Data Research 34](#_Toc144143544)

[6.1 Consent and Transparency 34](#_Toc144143545)

[6.1.1 Explicit Consent 34](#_Toc144143546)

[6.1.2 Implicit Consent 35](#_Toc144143547)

[6.2 Anonymity and De-identification 35](#_Toc144143548)

[6.2.1 Data Anonymization 35](#_Toc144143549)

[6.2.2 Reporting 35](#_Toc144143550)

[6.3 Fair Representation and Non-discrimination 35](#_Toc144143551)

[6.3.1 Bias 35](#_Toc144143552)

[6.3.2 Sensitive Topics 35](#_Toc144143553)

[7. Conclusion 36](#_Toc144143554)

[8. Limitations and Future Work 36](#_Toc144143555)

[8.1 Limitations 36](#_Toc144143556)

[8.2 Future Work 37](#_Toc144143557)

[9. Reference 38](#_Toc144143558)

[10. Appendix 41](#_Toc144143559)

[10.1 Data Collection (Snscrape) 41](#_Toc144143560)

[10.2 Data Pre-processing 43](#_Toc144143561)

[10.2.1 Remove Emoji and URL 43](#_Toc144143562)

[10.2.2 Translating non-English tweets to English 43](#_Toc144143563)

[10.3 Model Training 45](#_Toc144143564)

[10.3.1 Baseline Model 45](#_Toc144143565)

[10.3.2 Logistic Regression 45](#_Toc144143566)

[10.3.2.1 Model Training 46](#_Toc144143567)

[10.3.2.2 Find Optimal Iteration 46](#_Toc144143568)

[10.3.2.3 Final model after parameter tuning 46](#_Toc144143569)

[10.3.3 Stochastic Gradient Descent 47](#_Toc144143570)

[10.3.4 Random Forest Classifier 48](#_Toc144143571)

[10.3.5 LSTM Model 48](#_Toc144143572)

[10.3.5.1 Find the optimal vocab\_size 48](#_Toc144143573)

[10.3.5.2 Model initialization and training 49](#_Toc144143574)

[10.3.5.3 Adding a Drop Out Layer 50](#_Toc144143575)

[10.3.6 Tuning Pre-trained Models 50](#_Toc144143576)

[10.3.6.1. Setting up text 50](#_Toc144143577)

[10.3.6.2 Model Training 52](#_Toc144143578)

[10.4 Prediction Evaluation 53](#_Toc144143579)

# 

# List of Figures

[Figure 1 Global Social Media Usage Overview (Chaffey, 2023) 8](#_Toc144150722)

[Figure 2 Basic Sentiment Analyzer (Arora, 2022) 9](#_Toc144150723)

[Figure 3 Snapshot of Raw Twitter Data Collected Using Snscrape 16](#_Toc144150724)

[Figure 4 Baseline Model Results 18](#_Toc144150725)

[Figure 5 Evaluation of the Optimal Logistic Regression Model 19](#_Toc144150726)

[Figure 6 Stochastic Gradient Descent Performance 21](#_Toc144150727)

[Figure 7 Multi-Label Classification based on Random Forest (Wu et al., 2019) 21](#_Toc144150728)

[Figure 8 LSTM Model Architecture 23](#_Toc144150729)

[Figure 9 Frequency Distribution of Words in Training Data 24](#_Toc144150730)

[Figure 10 Cumulative Word Coverage with Marked 90% 25](#_Toc144150731)

[Figure 11 LSTM - Training and Validation metrics during the second run. 26](#_Toc144150732)

[Figure 12 LSTM model summary with dropout layers 27](#_Toc144150733)

[Figure 13 Validation Accuracy at Each Epoch and 500th Batch (BERT with SGD) 29](#_Toc144150734)

[Figure 14 Validation Accuracy at Each Epoch and at each 500th Batch (BERT with AdamW) 29](#_Toc144150735)

[Figure 15 First Run: RoBERTa validation accuracy (epoch, batch) 31](#_Toc144150736)

[Figure 16 Second Run: RoBERTa validation accuracy (epoch, batch) 31](#_Toc144150737)

[Figure 17 Comparative Harmonic Mean CDF of Positive and Negative Sentiments 33](#_Toc144150738)

# 1. Introduction

Social media platforms have grown popular in recent years and are now among the most popularly used platforms for communication and information sharing. According to statistics, it attracts billions of users worldwide *(Perrin., 2015)*. The extended usage of social media has changed how people perceive and interact with information shared on such platforms, particularly when it pertains to gender, race, and sexual orientation. This paper proposes different methods that could reveal how social media, cultural beliefs, and the attitudes that people have towards a social issue change with exposure to social media.

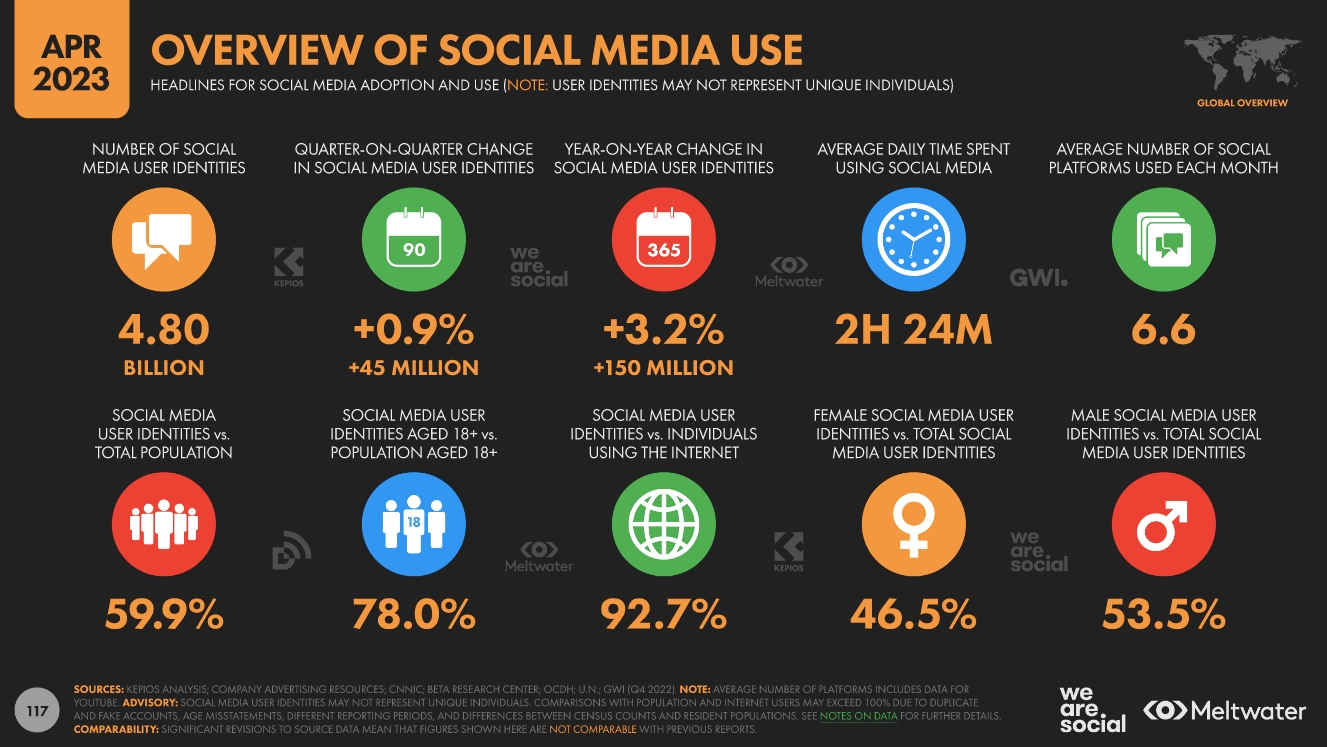


Figure 1 Global Social Media Usage Overview (Chaffey, 2023)

Social media is the perfect source of platform to study cultural trends and practices. The amount of user-generated data is broad and limitless. *(Chaffey., 2022)* It provides a way to shine a light on the diverse cultures, beliefs, and stories of individuals from different cultural backgrounds. Researchers could learn a lot about how cultural attitudes and opinions change over time by studying social media data. Instead of analysing every social media post, it might be useful to focus on social media data that is related to a particular social issue to gain better insights and narrow down the scope of the research.

The primary objective of this research is to train a model that accurately predicts the sentiments of the dataset it is presented with. To conclude, various algorithms were run and compared. The optimal model would then be applied to a dataset that has been scraped specifically for this research, one that contains data about a specific social topic. The results of this will be used to drive the research further. Shown below is how a sentiment analyzer works.

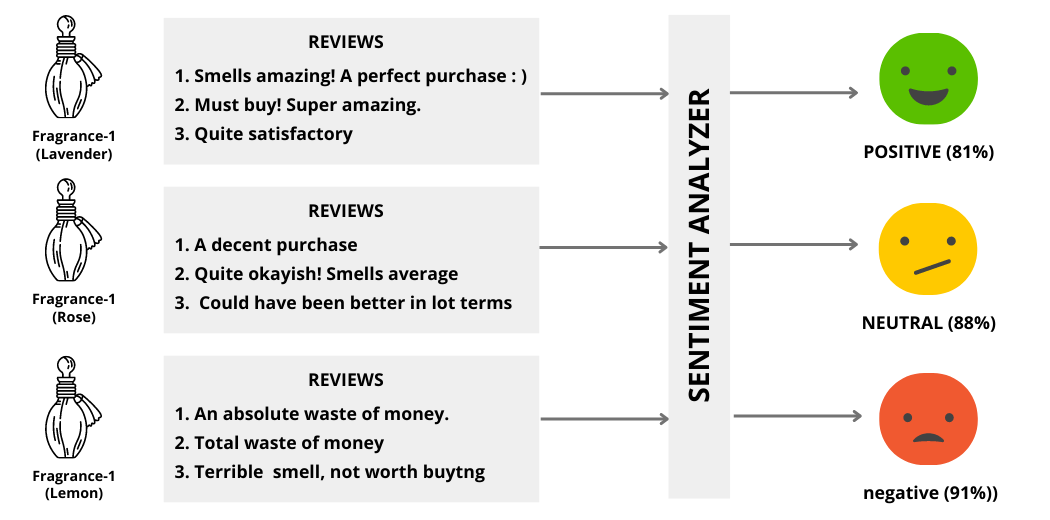


Figure 2 Basic Sentiment Analyzer (Arora, 2022)

The remainder of the paper will be structured as follows: The various research questions and their reasoning are described in Chapter 2. The background investigation conducted for this paper and the literature review are discussed in Section 3. Gaps in the literature are found and examined in the same section. Section 4 will cover the required tools for the study, the details of the system used for the study, and the techniques used to collect the data. Section 5 discusses the various methodologies and architectural designs used throughout the paper. The metrics that were used to compare the models and results will be part of this section. Section 6 dives into the professional and ethical problems that the project raises when dealing with social media data. The summary, self-evaluation, and difficulties encountered during the analysis are discussed in sections 7 and 8. The bibliography and appendix are at the end.

# 2. Research Questions

This paper examines the impact of social media by looking at how users perceive and respond to posts about a particular social issue, how those impacts are measured, the role machine learning algorithms play, and the ethical and cultural issues involved. The following research questions are put forth to accomplish these objectives:

1. **Machine Learning in Social Media Analysis:** What are the effective machine learning algorithms that can be used to detect patterns in social media content, and how can we optimise these algorithms to improve their accuracy and effectiveness? This question seeks to identify the most effective machine learning methods for studying social media content and investigates how to enhance them for improved effectiveness and accuracy.
2. **Influence and evolution of social media:** How much influence does social media have on affecting people's opinions towards a particular social issue, and how has this influence evolved? This question investigates the impact of social media on changing perspectives and its evolution over time, aiming to enhance the effectiveness of social media platforms.
3. **Ethical Considerations:** What are the ethical implications that need to be addressed while handling social media data to build a research model? By providing an answer, one can identify any potential ethical concerns, ensure that the study is conducted ethically, and protect the rights and privacy of the individuals whose data was used in it.

These questions could provide a detailed understanding of the influence of social media. The answers to the proposed questions might help guide research into developing a more sustainable social media tool, more precise social media analyses, and a greater awareness of the cultural and ethical ramifications of using social media for research.

# 3. Literature Review

## 3.1 Background Research

Over the past two decades, the influence of social media has evolved and has become a crucial channel for communication and information sharing *(Ling., 2020)*. Multiple research fields have taken an interest in this, especially sentiment analysis, a branch of Natural Language Processing. There has been much work done in this field, from traditional machine learning algorithms to complex neural networks. However, the challenge still exists around issues with culture-specific idioms, sarcasm, and other nuanced forms of expression that could confuse a model into predicting the wrong sentiments. The next section will go through literature that suggests the promising capability of machine learning models and other research that will focus solely on the social and psychological aspects of social media influence.

## 3.2 Related Work

A study *(Conover et al., 2011)* explored political polarisation on Twitter, focusing on the 2010 US midterm elections. The researchers found that echo chambers, created by users retweeting and mentioning others who share their beliefs, create segregated communities. This can lead to polarisation and extremism, as users are only exposed to information that confirms their existing beliefs. This can negatively impact democracy by preventing informed discussions on important issues. The study underscores the potential impact of social media platforms on shaping attitudes and beliefs, providing insights into cross-cultural perspectives on social media influence.

Another paper *(Pak et al., 2010)* developed a sentiment analysis and opinion mining corpus using Twitter data. They labelled tweets with their corresponding sentiment and collected data via the Twitter API. They used emoticons to classify text as positive, negative, or neutral. They extracted features from tweets and trained a Naive Bayes classifier using unigrams and bigrams. A 10-fold cross-validation was performed, using precision, recall, and F-measure as evaluation metrics. The model performed well but had limitations when users used emoticons sarcastically. The use of emoticons as a prediction criterion and metrics for model analysis was helpful. The Twitter data used in the study is highly correlated with the research, and the challenges proposed can be used as guidelines.

The paper *(Ravi et al., 2015)* discusses sentiment analysis and opinion mining using machine learning techniques. It covers document, sentence, and aspect-level sentiment analysis, which determines the overall sentiment of a document. The authors discuss various machine learning algorithms, including traditional methods like Naive Bayes, Support Vector Machines, and Decision Trees, as well as newer ones like Deep Learning. They also discuss the pros and cons of each method, pre-processing steps, and applications of sentiment analysis, such as social media monitoring, customer feedback, and product reviews. The paper also addresses challenges in sentiment analysis, such as handling implicit sentiments and negations. The research provides a comprehensive overview of sentiment analysis and opinion mining in various fields.

An author *(Balahur., 2013)* found a method that is specifically tailored for Twitter data which is what this research intends to use. The model considers the structure, length, and specific language of the data. The method is readily available in any language and can process tweets in real-time. The paper discusses various pre-processing techniques, as well as the potential use of minimal linguistic processing, making it easily transferable to other languages. They were able to determine that the best features to use are unigrams and bigrams together and that using sentiment dictionaries improved accuracy. This paper was useful to understand the research tailored for Twitter data which is crucial for effective analysis.

A paper *(Singh et al., 2022)* focused on sentiment analysis of the Twitter dataset that was related to the Farmer's protest in India. The authors used machine learning algorithms to analyse the sentiments and the study was aimed at understanding the depth of the protest and the various viewpoints of people on the issue. The paper discusses the challenges in performing sentiment analysis on such complex and emotionally charged topics. The paper employs various libraries and techniques for data extraction, pre-processing, and analysis. The use of the textblob library for sentiment analysis was quite useful and was decided to be used as a baseline model in the current research. The study also employed other algorithms such as Naive Bayes, Support Vector Machines and Logistic Regression. The paper concluded that Naive Bayes provided better results, which were calculated using precision and accuracy. They were also able to highlight the challenges and ethical considerations involved in handling social media data for research needs.

A study *(Zhang et al., 2023)* explores factors affecting urban happiness in Hangzhou, China. They used sentiment analysis, high-frequency words, and semantic network analysis to analyse comments on social media platforms. The study found that 47% of comments were positive, 2% neutral, and the rest negative. High-frequency words were found in each class. Economic factors like cost of living and income inequalities were identified as significant concerns affecting happiness. However, traffic and transport were common in both classes, indicating divided opinions on these topics. The study highlights the importance of understanding these factors in fostering urban happiness.

The study *(Immanuel et al., 2022)* addresses the rising issue of depression, a major contributor to suicide, and finds ways for early detection methods based on social media information. Using Twitter as a primary source, the paper proposes a lexicon-enhanced Long Short-Term Memory (LSTM) model to identify users who may be going through depression. The paper analyses existing methods and their limitations by comparing the high computational costs and difficulty faced while analysing short text. The paper discusses how the data was collected and then how it was divided into training, validation, and test sets and how the model was trained. The paper claims that their model can detect depression accurately and in a short time. They also proposed a system where the family members of the user are informed about the condition. The paper concludes by stating that their LE-LSTM model could be used as an early detection tool.

A paper *(Varughese et al., 2017)* titled "Analysing the Behavior of Youth to Sociality Using Social Media Mining" tries to understand the behaviour and interests of youth by implementing social media mining. The primary objective was to focus on the trending topics and classify sentiments using Natural Language Processing. The polarity of tweets related to each topic is calculated and compared. The study states how the approach can help in understanding public opinions on different matters and could be used by companies and governments for decision-making and performance improvement.

The research *(Boiy et al., 2009)* explores the issues of sentiment analysis when dealing with multilingual web texts, specifically English, Dutch, and French language. They utilised machine learning algorithms such as Support Vector Machines (SVM), and Multinomial Naïve Bayes (MNB), and tried to classify sentiments as positive, negative, or neutral. Despite the noisy data, the model they trained was able to predict the sentiments with an accuracy of 83% for English texts, and approximately 70% and 68% for Dutch and French texts. The methodology involved a 10-fold cross-validation and employed a cascaded approach for optimising feature computations. The paper also investigates the role of active learning techniques that offer marginal performance improvements.

A paper *(Imran et al., 2020)* offered an analysis of public sentiments and emotional comments towards the COVID-19 pandemic and this was done across different cultures. They utilised Deep Long Short-Term Memory (LSTM) models, to retrieve the sentiment polarity and emotions expressed in a tweet. Two datasets were used including one with trending hashtags and talks about emoticons as a possible validation method. This could potentially help with cross-verification of the model's performance. The paper doesn't dive deep into the details of the LSTM model but does emphasise its effectiveness in capturing the emotional response related to the pandemic.

The paper "Sentiment Analysis for the News Data Based on Social Media" approaches to understanding the public sentiments towards any news they see through social media platforms. The author *(Shahare, 2017)* used two methods, one was Naïve Bayes for sentiment classification and the other was the Levenshtein algorithm for text processing. This combination allowed the classifier to categorise the emotions into six different levels: anger, sadness, fear, joy, disgust, and surprise. The author also argued that their method could be scalable and provide real-time analysis of news data.

A paper *(Tufekci, 2014)* dives into challenges associated with analysing big data from social media platforms. The paper talks about the over-reliance by researchers on Twitter data and highlights that this focus can create bias and limit the generalisation process. The structure of Twitter data, such as short message lengths and hashtags might not be able to represent the entire social media users. The paper also discusses the shortcomings of the hashtag-based analysis, leading to skewed or incomplete insights. The author finishes by stating that a more rigorous approach needs to be taken, to acknowledge the bias and limitations produced by such datasets.

A paper (*Singh et al., 2020*) delves into the ethical challenges that are associated with using social media data. The authors mainly focused on data privacy and bias produced by algorithms. They argue that the lack of transparency is concerning and provides a mathematical framework that can be followed to construct a security threat model. They examine the limited privacy provided by social media sites such as Instagram, Snapchat, TikTok, Twitter, and LinkedIn. The paper concludes by discussing the need for a better transparent system while performing software development and suggests ethical considerations to be followed.

## 3.3 Critical Analysis and Research Gap

The literature review covers a range of topics from machine learning algorithms to ethical considerations in social media analysis. It lays the groundwork for the research questions mentioned in this paper and offers insights into various methodologies and frameworks constructed around social media analysis.

While extensive research has been performed around individual aspects of social media's influence, there is a clear lack of integrative studies that bridge these dimensions. Previous studies have focused on general sentiment analysis. So, a need for research that combines machine learning effectiveness, social influence, and ethical consideration all into one framework. Additionally, the rapid evolution of social media platforms and technologies indicates a need for a continuous adaptation and evaluation of existing models and methodologies.

# 4. Tools and Technology

## 4.1 Development Tools and Frameworks

The tools and frameworks used in this research were selected based on the research requirements and guided by the process of data collection, data pre-processing, model training, and result analysis. This section provides an overview of the development environment and frameworks used throughout the research.

This research employed Python, a programming language that is popular in the data science community due to its wide range of libraries and frameworks specifically for data science and analytics. Jupyter Notebook and Google Colab were used as platforms to run the code. Jupyter Notebook was used for less intense tasks that require only CPU and take less time to execute (*Jupyter Project, 2021*). Jupyter Notebook is a powerful tool for data analysis and presentation. It allows users to combine code, documentation, and visualisations in a single document. Google Colab is a cloud-based platform that provides high-performance GPUs for more computationally intensive tasks. (*Google Research, 2021*). The data was stored in Google Drive for the convenience of accessing it in Google Colab (*Google, 2021*).

### 4.1.1 Python Frameworks

The research utilised various Python frameworks and libraries, including Snscrape for tweet collection (*Snscrape, 2021*), Pandas for data manipulation, TensorFlow for neural network models, Keras for natural language processing, PyTorch for machine learning, Transformers for transformer-based models, and Matplotlib and Seaborn for data visualisation. These tools were used to efficiently analyse tweets, handle data manipulation, build neural network models, analyse text data, and create informative plots and charts.

## 4.2 System Specification

| **Component** | **Specification** |
| --- | --- |
| Device | MacBook Pro 2021 |
| Chip | Apple M1 Pro |
| Total CPU Cores | 8 |
| High-Performance CPU Cores | 6 |
| Energy Efficient CPU Cores | 2 |
| RAM | 16GB |
| GPU | Integrated GPU with 14 cores |
| Graphics and Computing API | Metal 3 |
| Capabilities | Advanced rendering and computation |
| Source | Apple, 2021 |

# 5. Methodology

The research methodology is divided into three steps, the first of which is the collection of relevant information (social media data), followed by analysis resulting in valuable insights. The third step, which is completed from start to finish, is considering the ethical implications while performing the analysis.

The steps in the data collection process include learning about the various social media platforms that are available and choosing the most significant ones, like Twitter, Instagram, and Facebook, where appropriate keyword-based searches and hashtags are used.

## 5.1 Data Collection

Data collection is crucial for sentiment analysis models and Twitter was found to be the most popular source. Other sources like Instagram, Facebook, and Reddit are challenging to analyse because of the way the data is structured. The research used manual scraping methods and publicly available datasets. Twitter data included text, metadata, geolocation, language information, and source.

### 5.1.1 Method of Collection

1. **Defining the search query:** For relevance, tweets were scraped based on a search string. The search string was constructed using specific keywords and hashtags that were related to the ‘Black Lives Matter’ movement. The topic was chosen because it was a popular topic that is receiving a lot of attention.
2. **Setting the time range and the limit:** The research collected tweets between January 1, 2021, and January 28, 2023, with a scraping limit of 100,000 to ensure relevance and accuracy.
3. **Using Snscrape:** The Snscrape library allows for automated data collection, which includes the tweet content, user information and tweet metadata.
4. **Data collection process:** Repeated pausing of 900 seconds after each 2000 tweets to avoid the chance of rate limiting by Twitter.
5. **Data extraction:** The data extracted from each tweet includes its date, time, text, unique ID, user information, replies, retweets, likes, quotes, language, source, original tweet ID, usernames, and hashtags. It also includes information about the user, followers, friends, statuses, and location.
6. **Data storage:** The tweets collected are stored in pandas dataframe for ease of data manipulation and analysis. The unlabelled data was inspected for quality and relevance, and a subset of tweets was manually reviewed, and then stored in a CSV file for easy processing.

A dataset containing 1.6 million tweets scraped from Twitter was downloaded from Kaggle (Kaggle, 2021) for model training. It was pre-labelled with negative and positive sentiments where 50% of tweets were positively labelled and 50% negatively labelled, ensuring a more robust model during training.

## 5.2 Data Pre-Processing

The data pre-processing stage involved transforming raw tweets into a clean and standard format. The scraped dataset was cleaned by removing unwanted columns, converting the date column to a datetime format, converting tweet text to lowercase, removing emojis, punctuations and URLs, and analysing language codes. The mentions and URLs were removed and HTML and UTF-8 BOM characters were decoded. Apostrophes were changed to represent negations. Empty tweet rows were monitored and eliminated as they materialised. Most tweets were translated into English for ease of analysis. These steps were crucial for training a clean and effective model. Similar steps were taken to clean the labelled dataset procured from Kaggle.

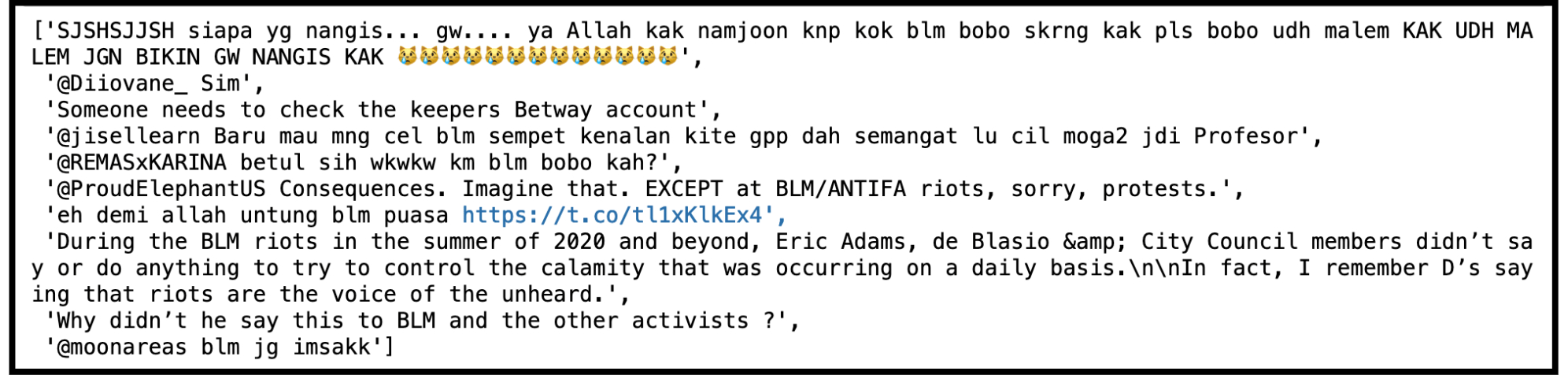


Figure 3 Snapshot of Raw Twitter Data Collected Using Snscrape

## 5.3 Model Building and Evaluation

The research explores machine learning techniques for analysing social media data. Various algorithms were trained by utilising the pre-labelled dataset. The findings can help researchers and practitioners select the most appropriate machine-learning techniques for their specific needs.

### 5.3.1 Baseline Model

A baseline model was trained for tracking the performance of more complex models. They are typically simple algorithms that need no tuning and can be implemented with minimum effort. For sentiment analysis, a baseline model can be a predefined vocabulary of words and their associated sentiment values (*Saif et al., 2012*). A Textblob model was used as a baseline as it provides a simple way to calculate the polarity scores for input text. The polarity is produced in the range -1 to 1, where positive texts are closer to 1, and negative are closer to -1. The ones closer to 0 are neutral texts (*Ahuja et al., 2017*). The performance of the model is calculated using accuracy, confusion matrix, and classification report metrics. Accuracy provides the ratio of correctly classified texts. The confusion matrix presents the number of true positives, true negatives, false positives, and false negatives while making predictions. Precision, F1-score, and recall are provided in the classification report, and these metrics are helpful for thoroughly analysing the performance of the model.

Textblob is a simple baseline model that can be vulnerable to misclassification when presented with text that contains slang, idioms, or other language constructs. It may also face issues with context-based sentiment where surrounding text is used to find the sentiment of a word. Despite these limitations, Textblob is useful as a starting point and can be compared as research progresses, making it a valuable tool for understanding and analysing the models.

The model's accuracy was 62.46% meaning that the model correctly predicted the sentiment of 62.46% of the tweets in the dataset. This is a reasonable accuracy rate, but there is room for improvement. The confusion matrix revealed that the model had many false positives and false negatives. This means that the model often misclassified tweets as positive or negative when they were neutral. The classification report revealed that the model performs better at predicting negative tweets accurately than positive tweets. This may be because negative tweets tend to be more explicit in their sentiment than positive tweets. The balanced F1-score is a clear indication that the model is not biased toward one target class.

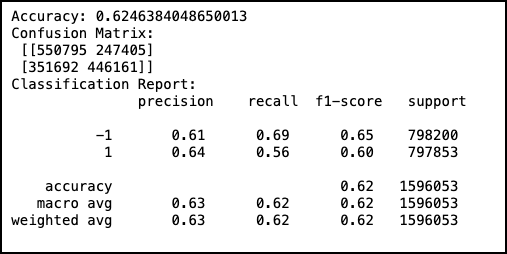


Figure 4 Baseline Model Results

### 5.3.2 Logistic Regression

Logistic regression is a popular choice for binary classification problems as it is computationally efficient and works well for linearly separable data. It predicts whether a text is positive or negative by using features extracted from the text, such as word frequency and TF-IDF scores (*Bhargava et al., 2017*).

The training text for the model can be represented as a vector of dimension V, where V is the vocabulary size. The model uses sparse vectors for tweets, potentially increasing training and prediction time. To reduce the computational cost, it uses token frequency, identifying more frequent words in each class for more efficient sentiment prediction.

Logistic regression utilises the ‘*logistic function’* to find the probability of a tweet belonging to a class. The logistic function, also known as the sigmoid function is as follows:

**;** x = linear combination of the input features and their corresponding weights.

#### 5.3.2.1 Choice of Vectorizer

Countvectorizer and Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer are both utilised for feature extraction. Countvectorizer converts text into a numerical form using a bag-of-words model, denoting each document as a vector of word counts. Each element represents a token in the vocabulary, with the value representing the word count in the entire document. TF-IDF considers both the frequency of a word in a document and the entire corpus. Both methods are part of the *‘sklearn.feature\_extraction.text’* module in the scikit-learn library.

* TF(w,d) =
* IDF(w) = log ()
* TF-IDF(w,d) = TF(w,d) × IDF(w)

Words that are specific to a document are given more weight by the TF-IDF vectorizer. Higher TF-IDF scores are assigned to words that are frequently used in one document but infrequently used in other documents. On the other hand, stopwords and other frequently occurring words in numerous documents receive a lower TF-IDF score. The model may be able to better capture the little details and semantics of the text data with the aid of these methods.

#### 5.3.2.2 Finding the Optimal Iteration Step

A logistic regression model was trained using varying iteration step values from 100 to 900, affecting its accuracy. The model's accuracy was evaluated on a validation set to confirm convergence. The max\_iter value showed that the model's accuracy increased with increasing iterations. However, after a certain number of iterations, the accuracy dropped, indicating that the optimal number of iterations (200) is crucial for the model's performance. The max\_iter value will be used as training progresses.

#### 5.3.2.3 Results of Logistic Regression

The study compared vectorizer models with and without stopwords, finding that models with stopwords performed better than those without stopwords. This is because stopwords preserve sentence context. Removing stopwords reduces the dimensionality of the feature space, which can prevent overfitting but also lead to information loss. Tf-idf was performing better than count-vectorizer, despite the computational costs.

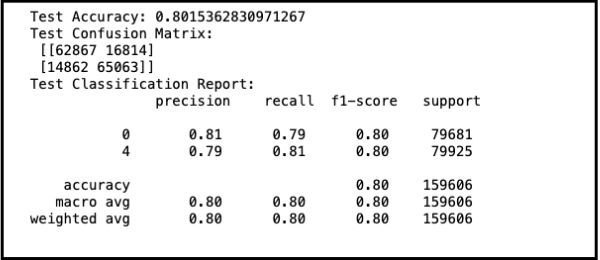


Figure 5 Evaluation of the Optimal Logistic Regression Model

### 5.3.3 Stochastic Gradient Descent Classifier

Stochastic gradient descent (SGD) is a linear classifier that uses stochastic gradient descent optimization to find optimal model parameters, especially for larger data and high-dimensional feature spaces, updating them iteratively by selecting a random subset of training data (*Goswami et al., 2021*).

#### 5.3.3.1 How SGD works.

The initial model parameters are set with random values. After each iteration, a random subset of the training data is used to calculate the gradient of the loss function concerning the model parameters. To minimise the loss function, the model weights are updated in the opposite direction of the gradient. The amount of change is controlled by a predefined learning rate. The process repeats until the model converges, which is confirmed when the change in loss after an iteration is below the threshold set or if the maximum number of iterations have been done. SGD classifiers are highly scalable and take up less memory as they run on a subset of data at a time. The stochastic nature of the optimization makes the model converge at a higher rate.

#### 5.3.3.2 Limitations of SGD

The performance of an SGD model is highly sensitive to hyperparameters, making it hard to find the optimal parameters. The subset training could mean that the gradient updates are noisy at times and may lead to oscillations in the parameter values. This could result in a slower convergence or produce noisy results. (*Goswami et al., 2021*)

#### 5.3.3.3 Model Training and Results

The model was trained on the whole dataset. The experiments were run over a set of feature counts from the list [5000, 10000, 20000, 50000, 100000]. For each of the feature count, a parameter grid was formed with different combinations of *‘alpha’* and *‘penalty’* parameters. The parameter grid had eight values for alpha: [1e−4, 1e−3, 1e−2, 1e−1, 1e0, 1e1, 1e2, 1e3] and three values for penalty: [‘l2’, ‘l1’, ‘elasticnet’].

The model was configured with a hinge loss function that replicated the functionality of a linear support vector machine (SVM). The model was set to run for 1000 iterations, with a tolerance of 0.001. The validation accuracies for the different maximum feature counts were as follows:

* 5000 features: 78.61%
* 10000 features: 78.84%
* 20000 features: 78.84%
* 50000 features: 78.87%
* 100000 features: 78.82%

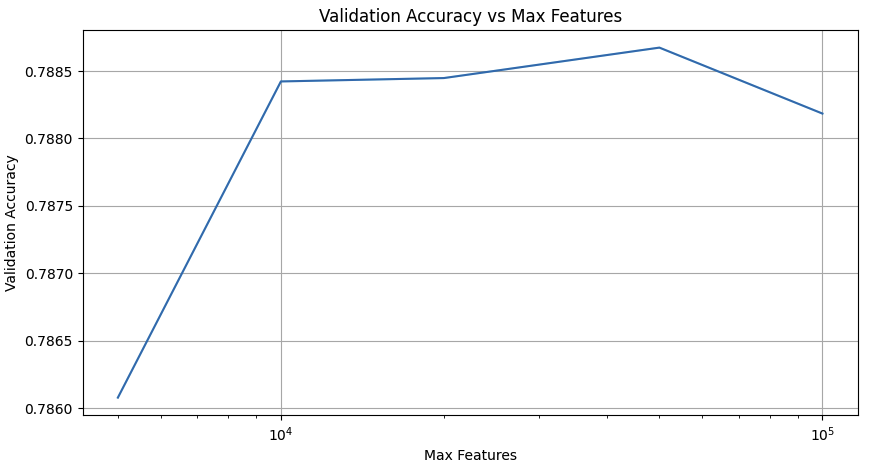


Figure 6 Stochastic Gradient Descent Performance

The optimal parameters were the same across each feature range, with an alpha of 0.0001 and a penalty of l2. The performance across all possible combinations ranged from 78.61% to 78.87%. While this performance is satisfactory, there are some possible reasons for the poor results that require further investigation and are beyond the scope of this research.

### 5.3.4 Random Forest Classifier

Random forest classifier is an ensemble learning technique that uses the ability of several decision trees to build a more precise and robust model. The model will be able to handle high-dimensional data and complex feature interactions (*Karthika et al., 2019*).

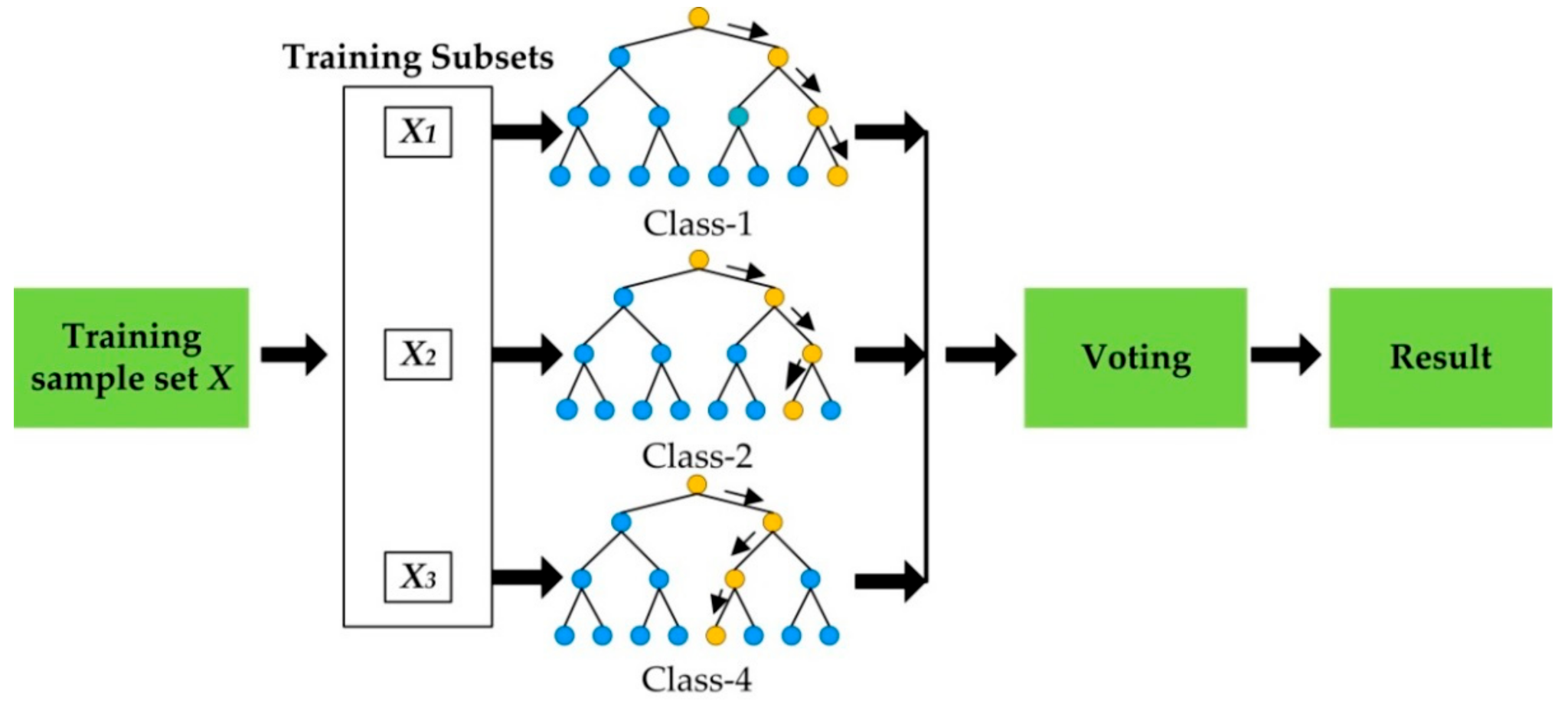


Figure 7 Multi-Label Classification based on Random Forest (Wu et al., 2019)

#### 5.3.4.1 How the model works.

1. **Bootstrapping**: Multiple bootstrap samples of training data are drawn with replacement, and a decision tree is trained using each sample.
2. **Feature** **Selection**: To maximise diversity among decision trees, a random subset of features is chosen at each split for each tree.
3. **Decision** **Tree**: Each decision tree is grown to its maximum depth and makes its predictions using the features.
4. **Voting**: After all the decision trees are voted, the class with the most votes is chosen as the final prediction

The model, when trained correctly, could attain high accuracy most times. This is because it is a combination of multiple decision trees to improve generalisation and reduce overfitting. The model is also capable of handling missing values during the training and prediction phase.

#### 5.3.4.2 Training Process

The dataset was randomised, and a 10% subset was selected as per computational limitations. The CountVectorizer was used for feature extraction and a RandomizedSearchCV for hyperparameter tuning. A parameter grid was used for feature extraction, and a range of features was set for maximum features. The model was trained on all possible values in the feature range, and the best parameters were calculated for each maximum feature count, ensuring accuracy. Best parameters: *n\_estimators* = 200, *min\_samples\_split* = 5, *min\_samples\_leaf* = 2, max\_depth = None.

#### 5.3.4.3 Testing and Results

Using 30% of the data for training and setting the best parameters, the model achieved 76.29% validation accuracy with a feature set size of 100,000. The test accuracy without stop words was 77.35%, but with stop words, it rose to 79.23%. The text classification model is accurate, but there's room for improvement. Exploring other feature extraction techniques like TF-IDF, using a comprehensive grid search or Bayesian optimization, or combining the model with Multinomial Naive Bayes could yield better results. These options could enhance the model's effectiveness.

### 5.3.5 Long Short-Term Memory (LSTM)

LSTM networks are a form of recurrent neural network (RNN) that was developed to address the shortcomings of traditional RNNs in learning long-term dependencies in sequential data. LSTM networks can do this by using a memory cell that can store information from previous inputs, which allows them to learn long-term patterns in data. Each LSTM unit has 3 parts: an input gate, a forget gate, and an output gate. (*Hochreiter et al., 1997*)

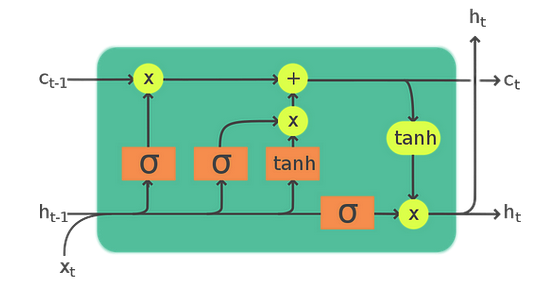


Figure 8 LSTM Model Architecture

#### 5.3.5.1 How it works.

The steps are as follows:

1. **Input Gate:** new input is added to the cell state.

= input gate; = current input; = previous hidden state.

and = weight matrices; = bias; = sigmoid activation function

1. **Forget Gate:** model determines if and how much of the previous cell should be retained.

= forget gate; and = weight matrices; = bias

1. **Cell State Update**: new input is produced based on the input and forget gates.

= current cell state; = previous cel state; and = weight matrices

1. **Output Gate:** measures how much of the updated cell state should be preserved and passed as the hidden state.

= output gate; = current hidden state; and = weight matrices

#### 5.3.5.2 Model Training

The pre-processing stage involves tokenizing the text into individual words, which are then lemmatized and stemmed to convert them to their base form. Lemmatization is the process of converting a word to its base form, also known as its lemma. For example, the words "ran," "run," and "running" are all derived from the word "run," so their lemma would be "run." Stemming is the process of removing the suffixes of a word to obtain a shorter word. For example, the words "history" and "historical" both stemmed from "histori".

To determine the vocabulary size, a hyperparameter for the model, and the cumulative frequency were calculated. The cumulative coverage was then calculated using the cumulative frequency. The plot revealed that a vocabulary size of 7000 was sufficient to cover almost 90% of the vocabulary of the entire document.

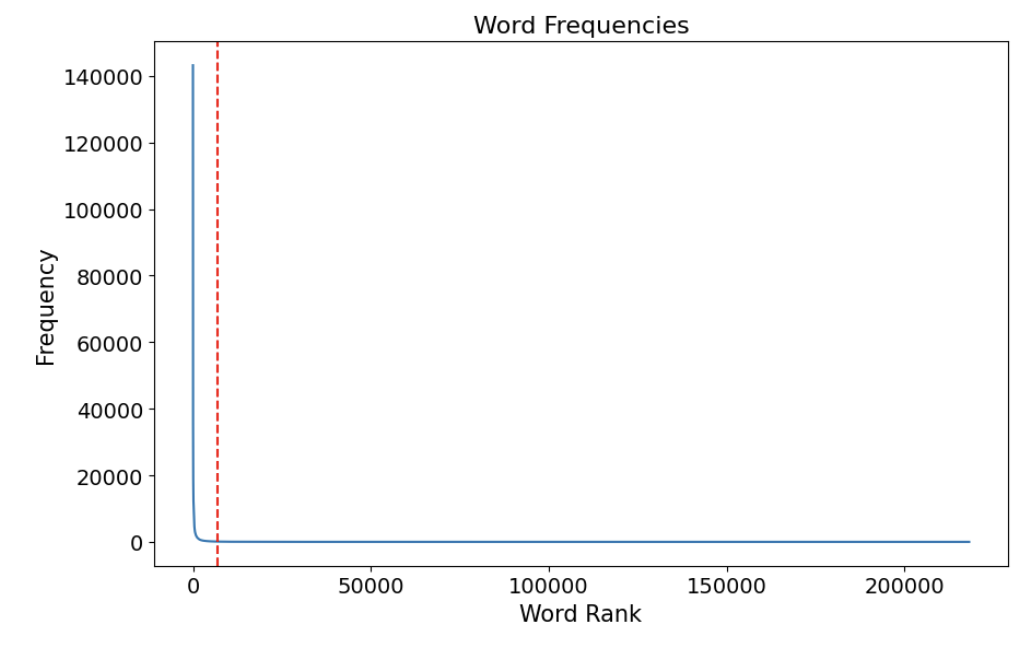


Figure 9 Frequency Distribution of Words in Training Data

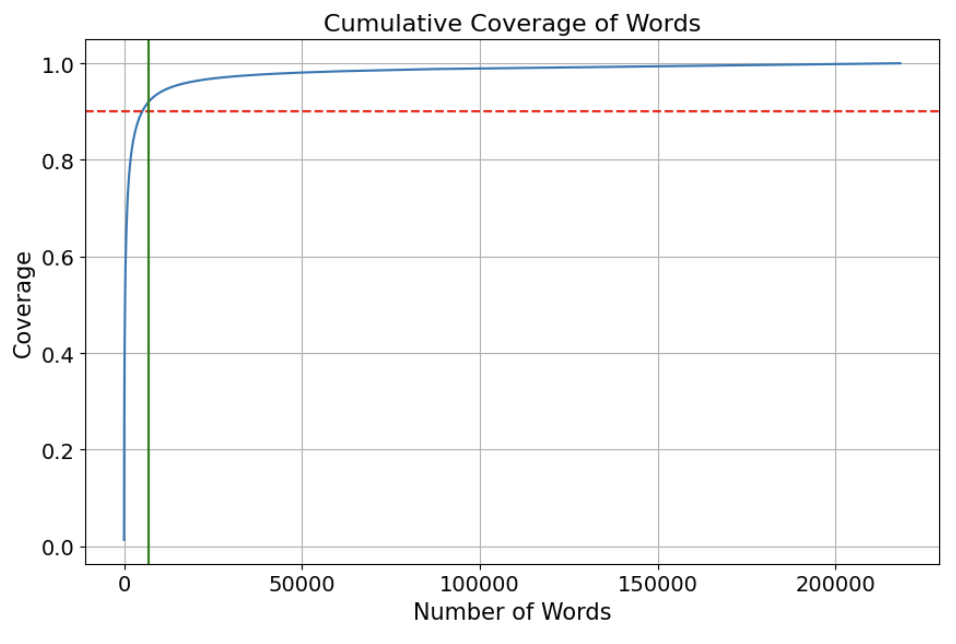


Figure 10 Cumulative Word Coverage with Marked 90%

After the vocabulary size was determined, the training sentence was tokenized into a sequence of integers. The sequence was then padded to maintain a consistent length. A special token, ‘<OOV>’, was used to represent out-of-vocabulary words.

The model is constructed using the Keras library and has the following layers:

* An embedding layer with the vocabulary size as an input dimension, an output dimension, and an input length equal to the maximum sequence length.
* A bidirectional long short-term memory (LSTM) layer used to learn patterns in the sequence.
* A dense layer with a ReLU activation function is used to introduce non-linearity.
* Finally, a dense layer with a sigmoid activation function is used to output the probability of the text being positive.

The model was compiled with binary cross-entropy loss, the Adam optimizer, and accuracy as the metric. The model summary reveals the parameters available for training.

#### 5.3.5.3 Results

The mode was initially implemented with two options: the first option was to remove stop words, and the second option was to retain stop words. The hyperparameters were set as vocab\_size = 7000, embedding\_dim = 100, max\_length = 200, Bidirectional LSTM layer units = 64, Dense layer 1 units = 24, output layer units = 1, default learning rate = 0.001.

**Without stopwords:**

The model was trained for 5 epochs on a padded training sequence with a validation split of 0.1 to monitor overfitting. It achieved a training accuracy of 81.11% by the fifth epoch and a validation accuracy of 78.39%. The early stopping mechanism was activated and the weights from epoch 2 were restored due to the model's unimproved validation loss. The model achieved a final training accuracy of 78.85% and a validation accuracy of 78.37%.

**With stopwords:**

The LSTM model was trained for eight epochs using an early stopping mechanism with a three-epoch patience. The training accuracy reached 80.64%, increasing to 85.71% by the sixth epoch. Validation accuracy started at 81.59% and peaked at 82.47% in the third epoch. The early stopping mechanism detected increasing loss after the third epoch, resulting in an 82.47% validation accuracy. The model performed better when stop words were retained.

During the second run, various hyperparameters were changed. The model achieved a slightly better validation accuracy of 82.73% but stopped early at epoch 6, like the first run. The validation loss was lower than the first run. The results indicated that increasing the embedding dimension drastically increased the training time with marginal improvement to the model, the same with higher LSTM units. Increasing the dense units had no significant impact. Sequence length and vocabulary size did not affect the results as well.

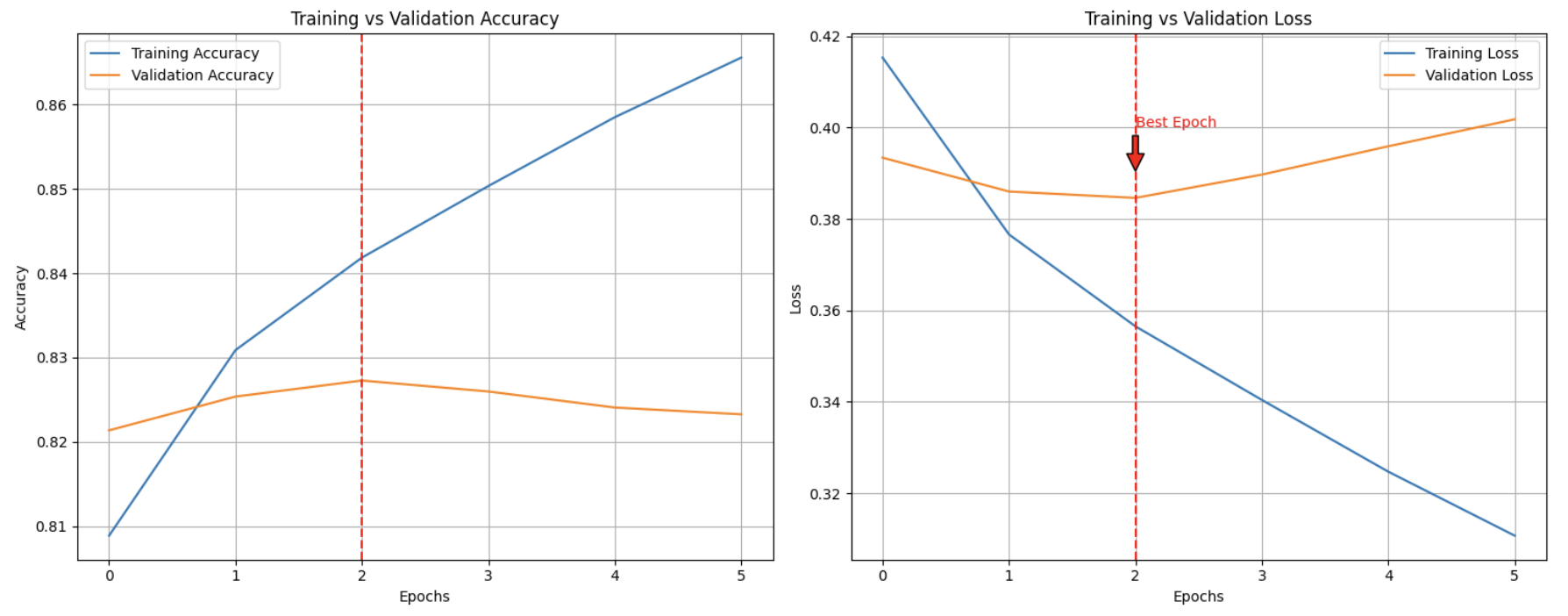


Figure 11 LSTM - Training and Validation metrics during the second run.

The third run was with a lower learning rate of 0.0001 from 0.001 while keeping all the other parameters from the first run. This model was able to achieve a validation accuracy of 81.95% and completed all epochs. The validation loss was at 0.3996.

The final run introduced a dropout layer to the first run, but despite the added regularisation, the model was only able to achieve a validation accuracy of 81.97% and a validation loss of 0.4004. The model ran all 8 epochs to completion.

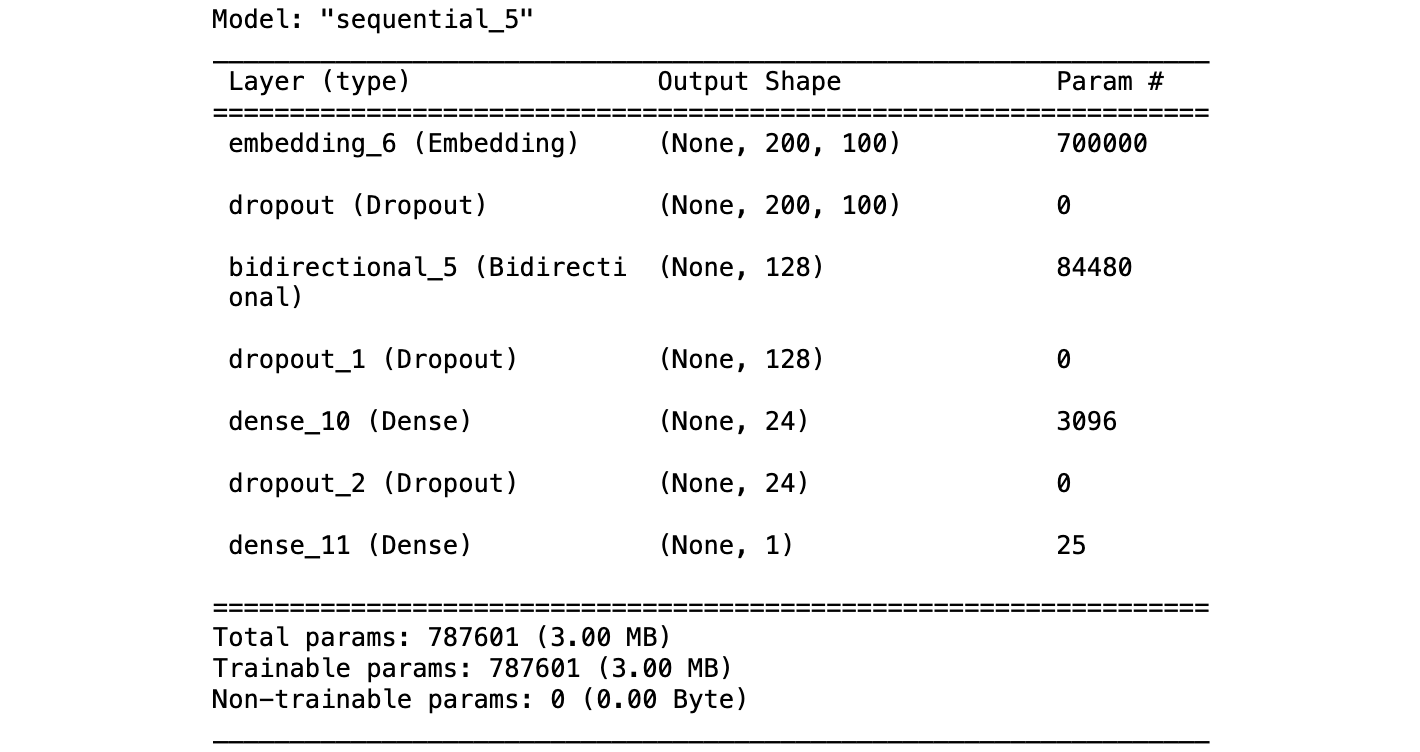


Figure 12 LSTM model summary with dropout layers

It was concluded that there is room for improvement after training the model with and without stopwords. Techniques such as regularisation (*Goodfellow et al., 2016*), and dropout layers (*Srivastava et al., 2014*) were explored but more extensive hyperparameter tuning can be explored to further improve the model. Experimenting with a pre-trained embedding like Word2Vec (Mikolov et al, 2013) or GloVe (*Pennington* et al., 2014) might help boost accuracy.

The traditional machine learning models discussed so far, such as logistic regression and SGD, all struggle when it comes to capturing the complex patterns found in text data. More advanced models like LSTMs perform better but may still fall short when capturing long-range dependencies and context-based analysis.

To address the limitations of the current approach, the paper explores the use of pre-trained models such as BERT and RoBERTa, which have been pre-trained on a massive amount of text data and require only a small amount of fine-tuning to the specific task at hand to achieve better results for the current problem.

### 5.3.6 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model (*Devlin et al., 2018*) that can be fine-tuned for a variety of tasks. It is a bidirectional model, which means that it can consider the context of words that come before and after a given word in a sentence. This makes it more accurate than unidirectional models, which can only consider the context of the words that come before a given word.

#### 5.3.6.1 Model Implementation

The dataset was subsetted to include 110,000 tweets from both the beginning and end to balance the two classes. The tweets were shuffled before training, then lemmatized, and stemmed to reduce dimensionality and improve model accuracy. The training, validation, and test data were tokenized using the BERT tokenizer and converted into PyTorch tensors for efficient batch processing. The training was accelerated using Google Colab's GPU capabilities.

SGD and AdamW were used as optimizers.

##### i) Stochastic Gradient Descent (SGD):

It is a traditional optimizer that is known for its simplicity and has a few hyperparameters to tune and could be used as a baseline to compare against a more complex optimizer like AdamW. The learning rate was set at 0.01. The BertForSequenceClassification model was used with pre-trained weights from the bert-base-uncased version. The model was trained and validated for 3 epochs. For each epoch, the model was first set to training mode. To monitor the model's learning process, the average training loss was printed out every 100 batches and the validation accuracy was calculated every 500 batches. The intermediate accuracies were stored for further analysis, and at the end of each epoch, the epoch's validation accuracy was calculated.

##### ii) AdamW:

AdamW is a modified version of Adam optimizer that is well-suited for NLP tasks. It is less sensitive to hyperparameter choices and combines the benefits of adaptive gradient descent algorithms and weight decay optimization. This results in a computationally efficient optimization algorithm that is more versatile and effective over multiple training epochs.

#### 5.3.6.2 Results

The BERT model with SGD optimizer showed a gradual decline in training loss from 0.6924 to 0.5193, resulting in a significant increase in validation accuracy from 72.45% to 75.86%. In the second epoch, the training loss slightly increased to 77.65%. By the final epoch, the training loss continued to decline, indicating effective learning. The model achieved a validation accuracy of 78.23%, indicating good generalisation of unseen data and a stable change in validation accuracy, indicating neither overfitting nor underfitting.

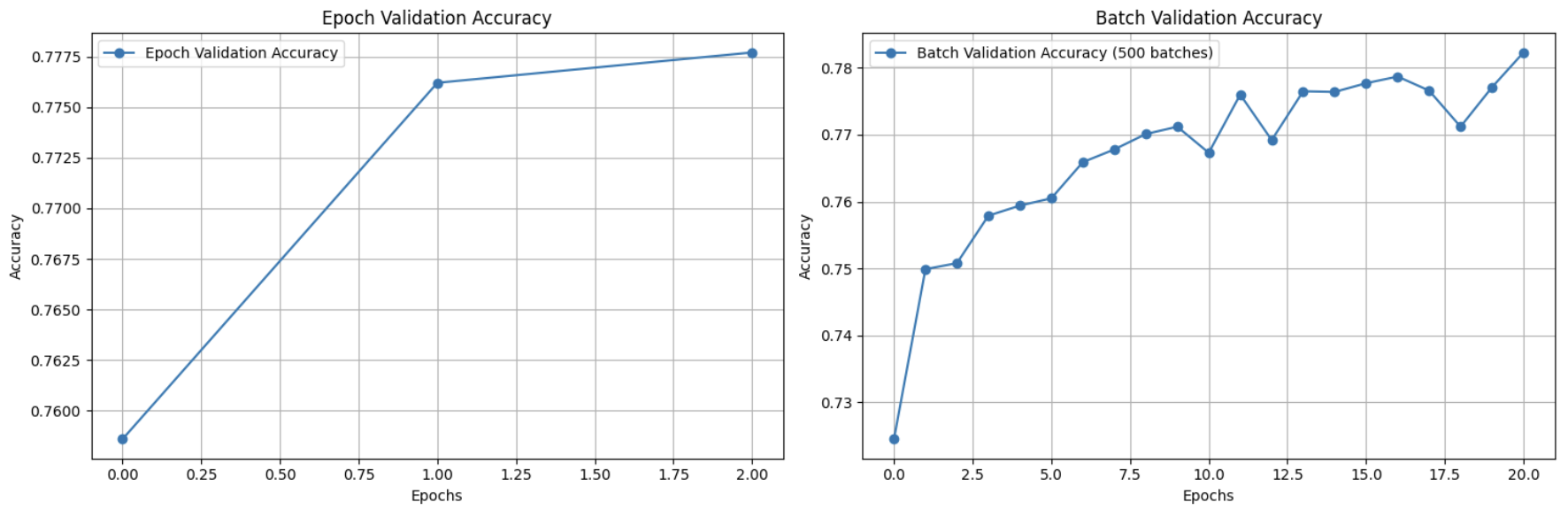


Figure 13 Validation Accuracy at Each Epoch and 500th Batch (BERT with SGD)

While the results look promising, there is still more work to be done, and more tuning to be performed to make sure the model has reached its expected potential.

Now when it comes to BERT with AdamW optimizer, the training loss began at 0.6899 and gradually declined to 0.3349 by the end of the third epoch. The validation accuracy started at a low of 0.4997 and improved consistently till it peaked at 0.7890 by the end of the third epoch.

The AdamW optimizer achieved higher validation accuracy across all epochs. AdamW also showed a general tendency to converge at a faster rate. The lower final training loss and higher validation accuracy could suggest better generalisation.

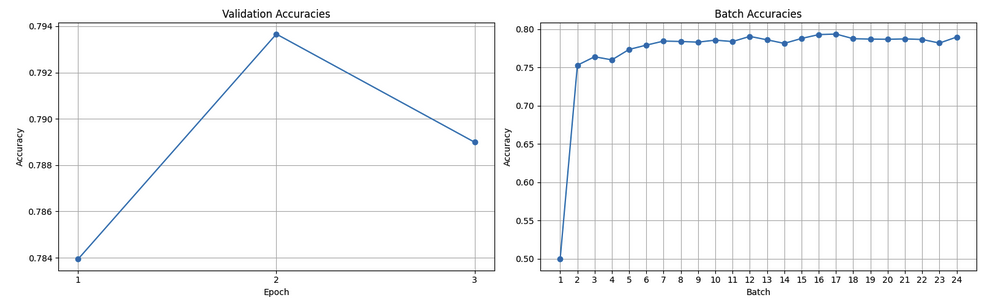


Figure 14 Validation Accuracy at Each Epoch and at each 500th Batch (BERT with AdamW)

This comparative analysis provides valuable insights into how effective each optimizer is for training BERT models. Due to computational limitations and extensive runtime, further tuning was not feasible to determine the choice of optimizer that would yield maximum accuracy.

### 5.3.7 RoBERTa

Robustly Optimised BERT (Bidirectional Encoder Representations from Transformers) Pre-Training Approach, is a variant of BERT that was created by Facebook AI. RoBERTa (Liao et al., 2021) uses multiple transformer block layers to capture context-based relations between words in a text. It uses dynamic masking, altering masked tokens between epochs, to learn more robust representations. RoBERTa is trained on a larger corpus and batch size, improving performance. It removes the Next Sentence Prediction (NSP) task, enhancing downstream tasks. Furthermore, it is trained for a higher number of iterations, allowing for optimised loss function and higher accuracy in performing tasks.

#### 5.3.7.1 Model Implementation

The RoBERTa model (*Liu et al., 2019*) for sentiment analysis is compiled by tokenizing data using RoBERTa's tokenizer, creating input IDs and attention masks, and converting them to PyTorch tensors. The model is initialised for sequence classification with two output labels, and a training loop is implemented to update the model weight and calculate the training loss and the validation accuracy at the end of each epoch.

The model was trained for 3 epochs and the results were analysed. Further training was done on the model by loading the model at a later stage and running it through more epochs.

#### 5.3.7.2 Results

The model was trained and validated in 2 separate runs, each run consisting of 3 epochs.

##### First Run:

The training loss was consistently decreasing across epochs. It started from 0.6887 and ended at 0.5580 after the first epoch, 0.5027 to 0.4970 by the second and 0.4812 to 0.4753 by the end of the last epoch. This was an indication that the model was learning effectively. The validation accuracy at the end of each epoch was 74.63%, 76.42% and 76.96%. The batch accuracy at each 500th batch was also analysed, and it started from 70.52% and increased to a peak of 79.59% by the end of the last epoch. The results were promising, so a second run was initiated.

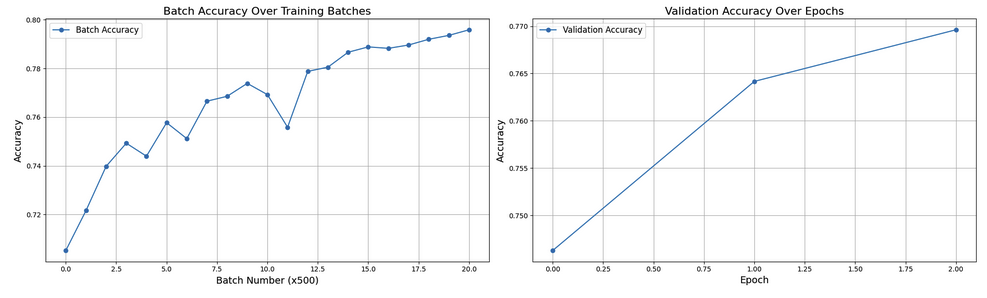


Figure 15 First Run: RoBERTa validation accuracy (epoch, batch)

##### Second Run:

During the second run, the model maintained a high level of validation accuracy and showed only a slight fluctuation. The validation accuracy rose from 77.84 to 78.19 but then declined a bit to 78.03%. The batch accuracy ranged from 77.11% to 80.89%, showing stable and accurate training.

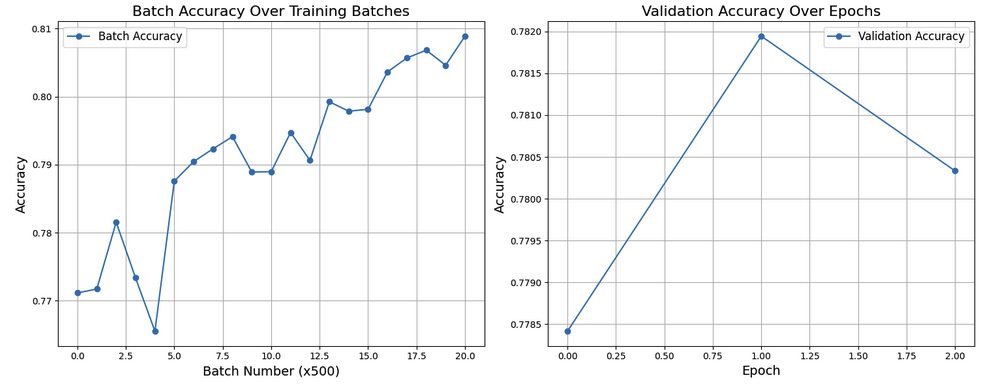


Figure 16 Second Run: RoBERTa validation accuracy (epoch, batch)

The Roberta model demonstrated efficient training and stable validation, allowing it to generalise and work well with unseen data. The model's learning capacity was evident as the training loss declined consistently. Validation accuracy remained high across different runs, indicating the model's ability to generalise well. However, a slight decrease in accuracy during the second run suggests the model may be overfitting the data. This is a minor fluctuation that should be considered when training or using the model on other data.

### 5.3.8 Evaluating and Choosing the Best-Fit Model for Sentiment Analysis

The goal of the research was to evaluate the performance of various ML algorithms, for sentiment analysis tasks on the Twitter dataset. After performing the training using various models from Logistic Regression, Stochastic Gradient Descent Classifier, Random Forest, and Long Short-Term Memory (LSTM) networks, they were evaluated. Moreover, experiments were conducted with pre-trained neural networks models like BERT and RoBERTa. Out of all the models LSTM was chosen and the reasoning is mentioned below:

#### 5.3.8.1 Robust Generalisation

The LSTM model showed higher training and validation accuracy, indicating that it was learning the underlying patterns in the data rather than fitting to the training set. This makes the model robust in generalising well when tested with unseen data.

#### 5.3.8.2 Effective Early Stopping

Overfitting is further prevented by the early stopping mechanism that is activated during training. As a result, the LSTM model is more trustworthy and resilient when used for sentiment analysis tasks.

#### 5.3.8.3 Importance of Stopwords

The LSTM model that retained stopwords performed better in both the training and validation phases. This suggests that including stopwords may be useful in capturing the semantic relations within a tweet, which is crucial for accurate sentiment analysis.

#### 5.3.8.4 Comprehensive Dataset Training

Unlike the pre-trained models, the LSTM model was trained on the entire dataset, leading to higher validation accuracy. This comprehensive training contributes to the model's ability to generalise well, which is particularly important for this research.

#### 5.3.8.5 Limitations of Pre-Trained Models

The pre-trained models like BERT and RoBERTa were only trained on a subset of the dataset. Although they showed promising results, training on a limited dataset can potentially hinder their ability to analyse the broader context within which sentiments are expressed in tweets.

By weighing these points, the LSTM model emerges as the most suitable and robust option for sentiment analysis in the context of this research.

## 5.4 Model Evaluation on Unlabelled BLM Tweets

The study used an LSTM model loaded from the local system that was run on an unlabelled dataset relevant to the research. The model was compiled using the 'Adam' optimizer and a binary cross-entropy loss function. The tweets were pre-processed and tokenized to convert text into a sequence and padded to a uniform length. The data was then processed through the model to generate predictions.

The study classified tweets as positive or negative and then verified their accuracy by plotting word clouds. To further understand sentiments, tweets were transformed into token count matrices using CountVectorizer, producing a feature frequency table for both sentiments. The frequency was normalised, and scatter plots were used to visualise divergence. Since no significant difference was found, harmonic mean and cumulative distribution functions (CDF) were used to understand the distribution. This exhaustive process provided a visual understanding of tweets in each class.

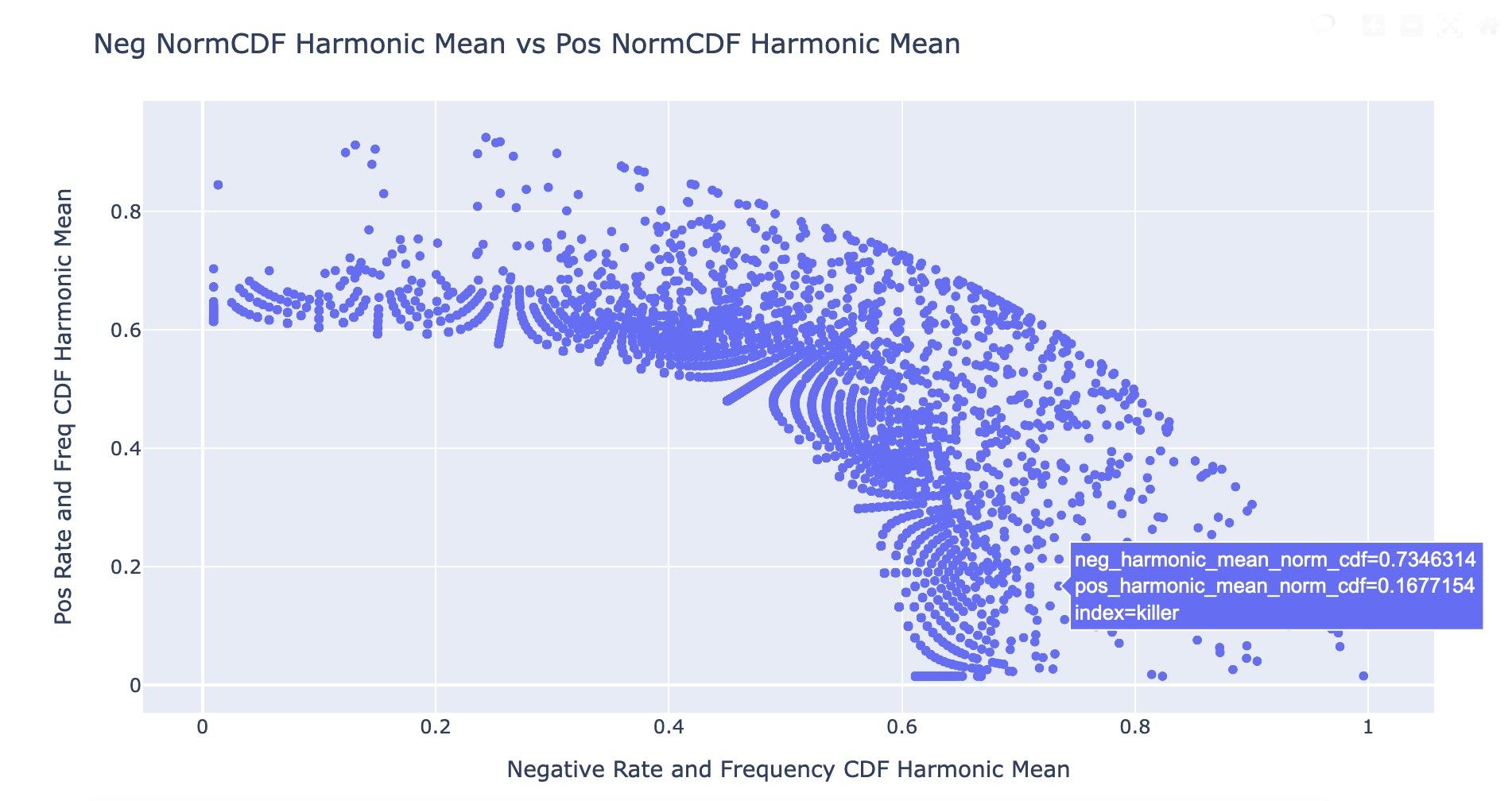


Figure 17 Comparative Harmonic Mean CDF of Positive and Negative Sentiments

Upon reviewing the plots and by manually comparing the predicted sentiments with the actual tweets, it was evident that the model had certain limits while trying to capture the nuances of the sentiments expressed in the text. These limitations will be discussed in depth in the limitations section of the paper.

# 6. Ethical Considerations in Social Media Data Research

Research that revolves around human-generated data, especially social media data, has always needed a strong ethical framework. This research aims to not just guide the researchers but also to protect the privacy, confidentiality and integrity of subjects involved in research.

## 6.1 Consent and Transparency

### 6.1.1 Explicit Consent

Twitter's terms of service (*Twitter Inc., 2023*) imply consent for the use of public data, but this is not the same as informed consent in the ethical sense of the term. Obtaining explicit informed consent can be challenging, due to the sheer volume of data being used and the difficulty of notifying everyone involved in the research. However, it is always best practice to inform participants of the research when and where possible.

### 6.1.2 Implicit Consent

Some researchers argue that social media users implicitly consent to the use of their data for research purposes when they post content on public platforms. However, this is not good practice. (*Fiesler et al., 2018*)

There are several reasons why this is not good practice. First, it is not clear that users understand that they are giving consent when they post content on social media. Second, even if users do understand that they are giving consent, they may not be aware of the full extent to which their data will be used. There are also instances where they may not be comfortable with this. (*Fiesler et al., 2018*)

## 6.2 Anonymity and De-identification

### 6.2.1 Data Anonymization

Anonymization is the process of removing or replacing personal information from a dataset that could identify a person. It is the first step that should be taken before moving forward with research. Researchers should be aware that full anonymity cannot be always guaranteed, and there are cases where the data can potentially identify a user. (*Zimmer., 2010*)

### 6.2.2 Reporting

It is considered best practice to avoid quoting posts by users that could allow readers to trace back to the original account when writing an academic publication. This is to protect the privacy of the users and to avoid any potential legal issues. (*Zimmer., 2010*)

## 6.3 Fair Representation and Non-discrimination

### 6.3.1 Bias

Social media analysis models are not objective and can reflect the biases that exist in society. It is the researcher's responsibility to be aware of these biases and to ensure that their methodology is robust enough to avoid misrepresenting any group. (*Tufekci., 2014*)

Some ways to mitigate biases include:

* Using a representative sample of data
* Using multiple models
* Using cross-validation
* Using interpretable models
* Evaluating the model's performance on held-out data.

### 6.3.2 Sensitive Topics

The utmost care must be taken to ensure that specific communities or groups are not marginalised or stigmatised while discussing sensitive topics, such as the "black lives matter" movement, which was used for this study. (*Tufekci., 2014*)

# 7. Conclusion

Social media sentiment analysis is an emerging field with an excess of problems that are yet to be solved. The focus of the study was to use machine learning algorithms to retrieve the sentiments within each tweet, and to analyse the impact of social media. For effective sentiment analysis, a model should be trained in a way that makes it generalizable to unseen data. Among the various models trained, Long Short-Term Memory (LSTM) networks showed the best performance metrics. Applying the trained model to the unlabelled dataset revealed issues with the model that could only be resolved by introducing the semantics and vocabulary of the unlabelled dataset and further tuning the model. This is an example of a model that performs well in a controlled environment but fails to generalise to real-world data. This is often because the model is trained on a dataset that is not representative of the real world, or that the model is not able to handle the noise and complexity of real-world data. The paper also dives into the ethical considerations to be taken while dealing with social media data, as it is sensitive and could be used to harm individuals or groups. By identifying where the research fell short, the study can be used as a starting point for deeper exploration.

# 8. Limitations and Future Work

## 8.1 Limitations

The current study was limited by several factors that affected the quality of the results and the extent to which the research question could be answered. One of the major issues was the changes made to Twitter's API policies, which hindered access to tweets. This change impacted all previously operational libraries that were used for scraping tweets. In addition to this issue, before the Snscrape stopped working, it was only possible to scrape tweets related to two social topics, BLM, and climate change. Unfortunately, the second dataset was lost during cleaning and could not be recovered. These limitations should be considered when interpreting the results of this study.

Another significant constraint was the nature of the unlabelled ‘Black Lives Matter’ dataset. The sentiment analysis model was not optimised for the special lexical tokens found in the scraped data. The words commonly associated with the Black Lives Matter movement might not have been well-represented in the training set. The chosen model, LSTM, is very sensitive to the sequence of words and might not be able to capture the social and political context, which is crucial for complex issues such as BLM. If the training dataset did not include many examples of social justice or activism-related content, then the model could be less effective at classifying such tweets. The tweets that were non-English when translated to English won't always hold the same semantic structure as tweets written in English. This could potentially break the model if the model considers the semantics while predicting.

## 8.2 Future Work

Despite the limitations posed in this research, there are several options for future research, that might be able to produce valuable answers to the research question proposed in this paper. One immediate step to consider is manually labelling a subset of the scraped data and using this labelled subset to fine-tune the sentiment model. This would introduce new unseen vocabulary that is needed for the model to correctly predict sentiments.

A more computationally powerful system that could run the complex models at a faster rate could help mitigate some of the limitations proposed. Models like BERT, RoBERTa which are computationally intensive need a powerful system to fine-tune the data. If the entire data is used for tuning the pre-trained model, that could help improve the accuracy of the model.

Another work to be considered is incorporating the multi-lingual approach (Dashtipour et al, 2016) to address the issues that occurred while translating the tweets and losing context in the process. A solution to this could mean a globally representative analysis, which is essential for social movements such as BLM.

# 9. Reference

1. Chaffey, D. (2023). Global social media statistics research summary 2023. available at https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/
2. Perrin, A., 2015. Social media usage. *Pew research centre*, *125*, pp.52-68.
3. Arora, S. 2022. Sentiment Analysis Using Python. Available at https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/
4. Conover, M., Ratkiewicz, J., Francisco, M., Gonçalves, B., Menczer, F. and Flammini, A., 2011. Political polarisation on twitter. In *Proceedings of the international aaai conference on web and social media* (Vol. 5, No. 1, pp. 89-96).
5. Zhang, X.F. and Chen, R.M., 2023, April. A Sentiment and Semantic Network Analysis-based Study on the Factors Influencing the" Sense of Urban Happiness":—With a Focus on Hangzhou. In *2023 2nd Asia Conference on Electrical, Power and Computer Engineering (EPCE)* (pp. 18-22). IEEE.
6. Ling, J., 2020. Coronavirus public sentiment analysis with BERT deep learning.
7. Varughese, B.M., Joseph, M.S., Thomas, P.E. and Sherly, K.K., 2017, June. Analyzing the behavior of youth to sociality using social media mining. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1231-1235). IEEE.
8. Immanuel, J.D., Ragavan, H.M., Rani, P.G., Niveditaa, K. and Manikandan, G., 2022, April. AI to detect social media users depression polarity score. In *2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)* (pp. 415-418). IEEE.
9. Pak, A. and Paroubek, P., 2010, May. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc* (Vol. 10, No. 2010, pp. 1320-1326).
10. Singh, A., Kalra, N., Singh, A. and Sharma, S., 2022, March. Sentiment analysis of Twitter data during Farmers' Protest in India through Machine Learning. In *2022 International Conference on Computer Science and Software Engineering (CSASE)* (pp. 121-126). IEEE.
11. JustAnotherMiner., 2021. snscrape: A Python library for scraping social networking services. GitHub. Available at: <https://github.com/JustAnotherArchivist/snscrape> (Accessed: 13 August 2023).
12. Ravi, K. and Ravi, V., 2015. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-based systems*, *89*, pp.14-46.
13. Balahur, A., 2013, June. Sentiment analysis in social media texts. In *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 120-128).
14. Boiy, E. and Moens, M.F., 2009. A machine learning approach to sentiment analysis in multilingual Web texts. *Information retrieval*, *12*, pp.526-558.
15. Shahare, F.F., 2017, June. Sentiment analysis for the news data based on the social media. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1365-1370). IEEE.
16. Jupyter Project, 2021. Project Jupyter. Available at:<https://jupyter.org/>
17. Google Research, 2021. Google Colaboratory. Available at:<https://colab.research.google.com/>
18. Google, 2021. Google Drive. Available at:<https://www.google.com/drive/>
19. Kaggle, 2021. Kaggle: Your Home for Data Science. Available at:<https://www.kaggle.com/>
20. Apple Inc., 2021. MacBook Pro: Apple M1 Pro. Available at:<https://www.apple.com/macbook-pro-14-and-16/specs/>
21. Saif, H., He, Y. and Alani, H., 2012. Semantic sentiment analysis of twitter. In *The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I 11* (pp. 508-524). Springer Berlin Heidelberg.
22. Liao, W., Zeng, B., Yin, X. and Wei, P., 2021. An improved aspect-category sentiment analysis model for text sentiment analysis based on RoBERTa. *Applied Intelligence*, *51*, pp.3522-3533.
23. Ahuja, S. and Dubey, G., 2017, August. Clustering and sentiment analysis on Twitter data. In *2017 2nd International Conference on Telecommunication and Networks (TEL-NET)* (pp. 1-5). IEEE.
24. Bhargava, K. and Katarya, R., 2017, October. An improved lexicon using logistic regression for sentiment analysis. In *2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN)* (pp. 332-337). IEEE.
25. Raza, G.M., Butt, Z.S., Latif, S. and Wahid, A., 2021, May. Sentiment analysis on COVID tweets: an experimental analysis on the impact of count vectorizer and TF-IDF on sentiment predictions using deep learning models. In *2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2)* (pp. 1-6). IEEE.
26. Goswami, M. and Sabata, P., 2021. Evaluation of ML-Based Sentiment Analysis Techniques with Stochastic Gradient Descent and Logistic Regression. *Trends in Wireless Communication and Information Security: Proceedings of EWCIS 2020*, pp.153-163.
27. Farisi, A.A., Sibaroni, Y. and Al Faraby, S., 2019, March. Sentiment analysis on hotel reviews using Multinomial Naïve Bayes classifier. In *Journal of Physics: Conference Series* (Vol. 1192, No. 1, p. 012024). IOP Publishing.
28. Karthika, P., Murugeswari, R. and Manoranjithem, R., 2019, April. Sentiment analysis of social media network using random forest algorithm. In *2019 IEEE international conference on intelligent techniques in control, optimization and signal processing (INCOS)* (pp. 1-5). IEEE.
29. Wu, X., Gao, Y. and Jiao, D., 2019. Multi-label classification based on random forest algorithm for non-intrusive load monitoring system. *Processes*, *7*(6), p.337.
30. Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, *9*(8), pp.1735-1780.
31. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), pp.1929-1958.
32. Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep learning*. MIT press.
33. Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*
34. *Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).*
35. *Fiesler, C., & Proferes, N. (2018). “Participant” Perceptions of Twitter Research Ethics.*
36. *Zimmer, M. (2010). “But the data is already public”: On the ethics of research in Facebook.*
37. *Tufekci, Z. (2014). Big Questions for Social Media Big Data.*
38. *Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.*
39. *Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.*
40. *Singh, L., Polyzou, A., Wang, Y., Farr, J. and Gresenz, C.R., 2020. Social Media Data-Our Ethical Conundrum. A Quarterly bulletin of the IEEE Computer Society Technical Committee on Database Engineering.*
41. *Twitter Inc. (2023). Terms of Service. Available at: https://twitter.com/en/tos (Accessed: 12/08/2023).*
42. *Dashtipour, K., Poria, S., Hussain, A., Cambria, E., Hawalah, A.Y., Gelbukh, A. and Zhou, Q., 2016. Multilingual sentiment analysis: state of the art and independent comparison of techniques. Cognitive computation, 8, pp.757-771.*

# 

# 

# 

# 10. Appendix

## 10.1 Data Collection (Snscrape)

|  |
| --- |
| search\_string = "BlackLivesMatter OR #BlackLivesMatter OR BLM OR #BLM" tweet\_limit = 100000  tweets = [] count = 0 sleep = 900  start = datetime(2021, 1, 1) end = datetime(2023, 1, 28)  for tweet in sntwitter.TwitterSearchScraper(search\_string).get\_items():  if len(tweets) != tweet\_limit:  tweet\_date = tweet.date.date()  if start <= tweet\_date <= end:  count += count   if count % 2000 == 0:  print("COUNT:", count, "Sleeping for", sleep/60, "minutes")  time.sleep(sleep)  tweets.append([ tweet.date, tweet.rawContent,  tweet.renderedContent,tweet.id, tweet.user.username,  tweet.user.displayname, tweet.user.id,  tweet.user.renderedDescription, tweet.user.verified,  tweet.user.followersCount, tweet.user.friendsCount,  tweet.user.statusesCount,tweet.user.location,  tweet.replyCount,tweet.retweetCount,  tweet.likeCount,  tweet.quoteCount,  tweet.lang,  tweet.source,  tweet.retweetedTweet,  tweet.quotedTweet,  tweet.mentionedUsers,  tweet.hashtags  ])  else:  break |

|  |
| --- |
| df = pd.DataFrame(tweets, columns=[  'Date','Tweet\_content','Tweet\_rendered\_content',  'tweet\_id','user\_name','Display\_name','user\_id',  'user\_description','verified','user\_follower\_count',  'user\_friend\_count','user\_statuses\_count','user\_location',  'tweet\_reply\_count','tweet\_retweet\_count','tweet\_like\_count',  'tweet\_quote\_count','tweet\_language','tweet\_source',  'rt\_original\_tweet\_id','quoted\_tweet\_original\_tweet\_id',  'tweet\_mentioned\_users','tweet\_hashtags']) |

## 10.2 Data Pre-processing

### 10.2.1 Remove Emoji and URL

|  |
| --- |
| def remove\_emoji(text):  # convert emoji to its base text (enclosed in :emoji:)  text = emoji.demojize(text)  # remove converted emoji  pattern = r":[^:\s]+:"  text = re.sub(pattern, "", text)  text = text.replace(" "," ")  return text  def remove\_url(text):  pattern = r"http\S+|www\S+"  text = re.sub(pattern, "", text)  return text |

### 

### 10.2.2 Translating non-English Tweets to English

|  |
| --- |
| def translate\_to\_target(x, source, target):  translated = GoogleTranslator(source=source,target=target)  .translate(x)  return translated  count = 0  def translate\_row(row,source\_language,target\_language,  display\_progress=False):  global count  count += 1  if display\_progress and count % 1000 == 0:  print("Processed”, count , “rows")   if row['tweet\_language'] == source\_language:  return translate\_to\_target(row['tweet\_rendered\_content']  ,source\_language, target\_language)  else:  return row['tweet\_rendered\_content']  # iterate over rows for index, row in df.iterrows():  df.at[index, 'tweet\_rendered\_content'] =  translate\_row(row,'id','en', display\_progress=True) |
|  |

|  |
| --- |
| pattern\_username = r'@\w+' pattern\_url = r'https?://\S+|www\.\S+' www\_url = r'www.[^ ]+' combine\_pattern = r'|'.join((pattern\_username,pattern\_url,www\_url)) pattern\_n = f"[^{re.escape(string.ascii\_letters + string.digits + string.punctuation + ' ')}]" def remove\_apostrophe(text):  # omit one-letter words  one\_letter\_pat = r'\b\w\b'  text = re.sub(one\_letter\_pat, '', text)  # omit apostrophes  text = text.replace("'", "")  return text.strip() |

|  |
| --- |
| i=0 def clean\_text(text):  global i  # removing @mentions and urls   text = re.sub(combined\_pattern,'',text)  # decoding HTML encode   text = BeautifulSoup(text,'lxml').get\_text()  # decoding utf-8 BOM  text = re.sub(pattern\_n, "", text)  # remove apostrophe  text = remove\_apostrophe(text)  # preserve only letters  text = re.sub("[^a-zA-Z]", " ", text)  # delete white spaces and lowercase all letters  text = (' '.join(text.split())).lower()  if(i%10000 == 0):  print('completed:'+ str(i))  i = i+1   return text |

## 10.3 Model Training

### 10.3.1 Baseline Model

Analytics Vidhya (2021) 'Sentiment Analysis with TextBlob and VADER', Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/10/sentiment-analysis-with-textblob-and-vader/>

|  |
| --- |
| # apply textblob and extract sentiment polarity def extract\_polarity(text):  return TextBlob(text).sentiment.polarity  # to match textblob output df['textblob\_target'] = df['target'].replace({0: -1, 4: 1})  # predicting sentiment polarity df['predicted\_sentiment'] = df['tweet\_text'].apply(extract\_polarity)  # predicting sentiment using predicted polarity df['predicted\_sentiment'] = df['predicted\_sentiment'].apply(lambda x: 1 if x > 0 else -1) |

### 

### 10.3.2 Logistic Regression

scikit-learn (n.d.) 'scikit-learn: Machine Learning in Python', scikit-learn. Available at: <https://scikit-learn.org>

#### 10.3.2.1 Model Training

|  |
| --- |
| # vectorizer instance vectorizer = CountVectorizer() # fit object to training data x\_train\_vectorized = vectorizer.fit\_transform(x\_train)  # logistic regression model model = LogisticRegression() model.fit(x\_train\_vect, y\_train)  # model validation x\_val\_vect = vectorizer.transform(x\_val) y\_val\_pred = model.predict(x\_val\_vect) |

#### 10.3.2.2 Find Optimal Iteration

|  |
| --- |
| accuracies = []  # range of max\_iter values to be tested max\_iter\_values = range(100, 1000, 100)  # iterating max\_iter list for max\_iter in max\_iter\_values:  # log reg model with current max\_iter value  log\_reg = LogisticRegression(max\_iter= max\_iter)   log\_reg.fit(x\_train\_tfidf, y\_train)  y\_val\_pred = logreg.predict(x\_val\_tfidf)  acc = accuracy\_score(y\_val, y\_val\_pred)  accuracies.append(acc)  best\_max\_iter = max\_iter\_values[np.argmax(accuracies)] |

#### 10.3.2.3 Final model after parameter tuning

|  |
| --- |
| vectorizer = TfidfVectorizer(max\_features=120000)  # data transformation x\_train\_tfidf = vectorizer.fit\_transform(x\_train) x\_test\_tfidf = vectorizer.transform(x\_test)  # logistic regression model log\_reg = LogisticRegression(max\_iter=300, solver='lbfgs') log\_reg.fit(x\_train\_tfidf, y\_train)  # test set pred y\_pred\_test = log\_reg.predict(x\_test\_tfidf) |

### 10.3.3 Stochastic Gradient Descent

scikit-learn (n.d.) 'Stochastic Gradient Descent', scikit-learn. Available at: <https://scikit-learn.org/stable/modules/sgd.html>

|  |
| --- |
| feature\_range = [5000, 10000, 20000, 50000, 100000] hyperparam\_grid = {  'alpha': [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3],  'penalty': ['l2', 'l1', 'elasticnet'], }  # SGD model (hinge loss function is equivalent to linear SVM) svm\_model = SGDClassifier(loss='hinge', max\_iter=1000, tol=1e-3) # store val acc and optimal param val\_accs = {} best\_params = {}  for max\_features in feature\_range:  vectorizer = CountVectorizer(max\_features=max\_features)  x\_train\_vec = vectorizer.fit\_transform(x\_train)  x\_val\_vec = vectorizer.transform(x\_val)  grid\_search = GridSearchCV(svm\_model, hyperparam\_grid,  cv=3, verbose=1)  grid\_search.fit(x\_train\_vec, y\_train)  best\_params[max\_features] = grid\_search.best\_params\_  y\_val\_pred = grid\_search.best\_estimator\_.predict(x\_val\_vec)  val\_accs[max\_features] = accuracy\_score(y\_val, y\_val\_pred) |

### 10.3.4 Random Forest Classifier

scikit-learn (n.d.) 'sklearn.ensemble.RandomForestClassifier', scikit-learn. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

|  |
| --- |
| hyperparam\_grid = {  'n\_estimators': [10, 50, 100, 200],  'max\_depth': [None, 10, 20, 30],  'min\_samples\_split': [2, 5, 10],  'min\_samples\_leaf': [1, 2, 4] }  feature\_range = [i for i in range(10000,130000,10000)] validation\_accs\_rf = []  for max\_features in feature\_range:  vectorizer = CountVectorizer(max\_features= max\_features,  stop\_words=stop\_words)  x\_train\_vect = vectorizer.fit\_transform(x\_train\_subset)  x\_val\_vect = vectorizer.transform(x\_val\_subset)  random\_search = RandomizedSearchCV(  estimator=RandomForestClassifier(),  param\_distributions=hyperparam\_grid,  n\_iter=10, cv=3, verbose=2, random\_state=42)   random\_search.fit(x\_train\_vect, y\_train\_subset)  y\_val\_pred = random\_search.best\_estimator\_.  predict(x\_val\_vect)  val\_accs\_rf.append(accuracy\_score(y\_val\_subset,  y\_val\_pred)) |

### 10.3.5 LSTM Model

#### 10.3.5.1 Find the optimal vocab\_size

|  |
| --- |
| sentences = df['tweet\_text'].tolist()  # tokenizing the sentences tokens = [word for sentence in sentences for word in sentence.split()]  # frequency of each token frequency = Counter(tokens)  # sorting tokens on freq sorted\_tokens = sorted(frequency.items(), key=lambda x: x[1], reverse=True)  # cumulative sum of word freq cumulative\_freq = np.cumsum([x[1] for x in sorted\_tokens])  # cumulative coverage cumulative\_coverage = cumulative\_freq / len(tokens) |
|  |

#### 10.3.5.2 Model initialization and training

Analytics Vidhya (2021) 'LSTM for Text Classification', Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/>

|  |
| --- |
| train\_sentences = X\_train.astype(str).tolist() train\_labels = y\_train.values val\_sentences = X\_val.astype(str).tolist() val\_labels = y\_val.values  # hyperparameters  vocab\_size = 7000 oov\_tok = '<OOV>' embedding\_dim = 100 max\_length = 200 padding\_type='post' trunc\_type='post'  tokenizer = Tokenizer(num\_words = vocab\_size, oov\_token=oov\_tok) tokenizer.fit\_on\_texts(train\_sentences) word\_index = tokenizer.word\_index  # convert train dataset to sequence and pad sequences train\_sequences = tokenizer.texts\_to\_sequences(train\_sentences) train\_padded = pad\_sequences(train\_sequences, padding=padding\_type, maxlen=max\_length, truncating=trunc\_type)  # convert test dataset to sequence and pad sequences val\_sequences = tokenizer.texts\_to\_sequences(val\_sentences) val\_padded = pad\_sequences(val\_sequences, padding=padding\_type, maxlen=max\_length, truncating=trunc\_type)  # model initialization model = keras.Sequential([  keras.layers.Embedding(vocab\_size, embedding\_dim, input\_length=max\_length),  keras.layers.Bidirectional(keras.layers.LSTM(64)),  keras.layers.Dense(24, activation='relu'),  keras.layers.Dense(1, activation='sigmoid') ]) # compile model model.compile(loss='binary\_crossentropy',  optimizer='adam',  metrics=['accuracy'])  model.summary() |

|  |
| --- |
| early\_stop = EarlyStopping(monitor='val\_loss', patience=3, verbose=1,  restore\_best\_weights=True) num\_epochs = 5 history = model.fit(train\_padded, train\_labels,  epochs=num\_epochs, verbose=1,  validation\_split=0.1,  callbacks=[early\_stop]) |

#### 10.3.5.3 Adding a Drop Out Layer

Saturn Cloud (n.d.) 'How to Implement Dropout in LSTM Neural Networks with TensorFlow', Saturn Cloud. Available at: <https://saturncloud.io/blog/how-to-implement-dropout-in-lstm-neural-networks-with-tensorflow/>

|  |
| --- |
| from tensorflow.keras.layers import Dropout  drop\_model = keras.Sequential([  keras.layers.Embedding(vocab\_size, embedding\_dim  ,input\_length=max\_length),  # dropout layer after Embedding  Dropout(0.5),keras.layers.Bidirectional(keras.layers.LSTM(64)),  # dropout layer after LSTM  Dropout(0.5), keras.layers.Dense(24, activation='relu'),  # dropout layer before final layer  Dropout(0.5), keras.layers.Dense(1, activation='sigmoid') ]) |

### 10.3.6 Tuning Pre-trained Models

#### 10.3.6.1. Setting up text

Hugging Face (n.d.) 'Main Classes — Transformers 4.10.0 Documentation', Hugging Face. Available at: <https://huggingface.co/docs/transformers/main_classes/tokenizer>

|  |
| --- |
| tokenizer = BertTokenizer.from\_pretrained(  'bert-base-uncased',  do\_lower\_case = True  )  # tokenize train, val, test : train\_encoded\_data = tokenizer.batch\_encode\_plus(  train\_df['tweet\_text'].tolist(),  add\_special\_tokens=True, return\_attention\_mask=True,  padding='max\_length', truncation=True,  max\_length=137, return\_tensors='pt' )  val\_encoded\_data = tokenizer.batch\_encode\_plus(  val\_df['tweet\_text'].tolist(),add\_special\_tokens=True,  return\_attention\_mask=True, padding='max\_length',  truncation=True, max\_length=137,  return\_tensors='pt' )  test\_encoded\_data = tokenizer.batch\_encode\_plus(  test\_df['tweet\_text'].tolist(),  add\_special\_tokens=True, return\_attention\_mask=True,  padding='max\_length',truncation=True,  max\_length=137, return\_tensors='pt' )  # defined input\_ids, attention\_masks, and labels train\_input\_ids = train\_encoded\_data['input\_ids'] train\_attention\_masks = train\_encoded\_data['attention\_mask'] train\_labels = torch.tensor(train\_df['target'].replace(4, 1).tolist())  val\_input\_ids = val\_encoded\_data['input\_ids'] val\_attention\_masks = val\_encoded\_data['attention\_mask'] val\_labels = torch.tensor(val\_df['target'].replace(4, 1).tolist())  # dataloader instances batch\_size = 40 train\_dataset = TensorDataset(train\_input\_ids, train\_attention\_masks, train\_labels) val\_dataset = TensorDataset(val\_input\_ids, val\_attention\_masks, val\_labels) test\_dataset = TensorDataset(test\_input\_ids, test\_attention\_masks, test\_labels)  train\_dataloader = DataLoader(train\_dataset, batch\_size=batch\_size, shuffle=True) val\_dataloader = DataLoader(val\_dataset, batch\_size=batch\_size, shuffle=False) test\_dataloader = DataLoader(test\_dataset, batch\_size=batch\_size, shuffle=False) |

#### 10.3.6.2 Model Training

Hugging Face (n.d.) 'Main Classes — Transformers 4.10.0 Documentation', Hugging Face. Available at: <https://huggingface.co/docs/transformers/main_classes/tokenizer>

|  |
| --- |
| # BERT model model\_bert\_SGD = BertForSequenceClassification.from\_pretrained('bert-base-uncased', num\_labels=2)  device = torch.device("cuda" if torch.cuda.is\_available() else "cpu") model\_bert\_SGD.to(device)  optimizer = optim.SGD(model\_bert\_SGD.parameters(), lr=0.01)  # number of epochs epochs = 3 # storing acc after every epoch accuracies\_sgd = []  # storing acc after every 500 batch batch\_accuracies\_sgd = []   for epoch in range(epochs):  model\_bert\_SGD.train()  total\_loss = 0  for i, batch in enumerate(train\_dataloader, 1):  batch = tuple(t.to(device) for t in batch)  b\_input\_ids, b\_input\_mask, b\_labels = batch   optimizer.zero\_grad()  outputs = model\_bert\_SGD(b\_input\_ids, attention\_mask=b\_input\_mask,  labels=b\_labels)  loss = outputs.loss  loss.backward()  optimizer.step()   total\_loss += loss.item()   if i % 100 == 0:  avg\_train\_loss = total\_loss / (i + 1)   if i % 500 == 0:  model.eval()  val\_accuracy = 0  val\_total = 0  with torch.no\_grad():  for val\_batch in val\_dataloader:  val\_batch = tuple(t.to(device) for t in val\_batch)  val\_input\_ids, val\_input\_mask, val\_labels = val\_batch   val\_outputs = model(val\_input\_ids,  attention\_mask=val\_input\_mask)  val\_logits = val\_outputs.logits  val\_predictions = torch.argmax(val\_logits, dim=1)   val\_accuracy += (val\_predictions ==  val\_labels).sum().item()  val\_total += val\_labels.size(0)   batch\_val\_accuracy = val\_accuracy / val\_total  batch\_accuracies\_sgd.append(batch\_val\_accuracy)  model.train()   model.eval()  val\_accuracy = 0  val\_total = 0  with torch.no\_grad():  for batch in val\_dataloader:  batch = tuple(t.to(device) for t in batch)  b\_input\_ids, b\_input\_mask, b\_labels = batch   outputs = model(b\_input\_ids, attention\_mask=b\_input\_mask)  logits = outputs.logits  predictions = torch.argmax(logits, dim=1)   val\_accuracy += (predictions == b\_labels).sum().item()  val\_total += b\_labels.size(0)   epoch\_val\_accuracy = val\_accuracy / val\_total  accuracies\_sgd.append(epoch\_val\_accuracy) |

## 

## 10.4 Prediction Evaluation

|  |
| --- |
| def add\_to\_list(tweet):  for word in tweet.split():  positive\_text.append(word)   df[df['target']==4].tweet\_text.apply(lambda x:add\_to\_list(x))  wordcloud = WordCloud(width=1600, height=800,max\_font\_size=200,colormap='magma').generate(' '.join(positive\_text))  # Display the generated image: plt.figure(figsize=(12,10)) plt.imshow(wordcloud, interpolation='bilinear') plt.axis("off") plt.show() |

REFERENCE:

<https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html>

|  |
| --- |
| cvec = CountVectorizer() cvec.fit(df.tweet\_text)  # cvec - count vectorizer neg\_matrix = cvec.transform(df[df.target == 0].tweet\_text) pos\_matrix = cvec.transform(df[df.target == 4].tweet\_text)  neg\_df = np.sum(neg\_matrix,axis=0) pos\_df = np.sum(pos\_matrix,axis=0)  negative = np.squeeze(np.asarray(neg\_df)) positive = np.squeeze(np.asarray(pos\_df))  term\_freq = pd.DataFrame([negative,positive],columns=cvec.get\_feature\_names\_out()).transpose()  term\_freq.columns = ['neg','pos']  term\_freq['total\_count'] = term\_freq['neg'] + term\_freq['pos']  from sklearn.feature\_extraction.text import CountVectorizer  cvec = CountVectorizer(stop\_words='english',max\_features=10000)  cvec.fit(df.tweet\_text)  # cvec - count vectorizer  neg\_matrix\_1 = cvec.transform(df[df.target == 0].tweet\_text)  pos\_matrix\_1 = cvec.transform(df[df.target == 4].tweet\_text)  neg\_df\_1 = np.sum(neg\_matrix\_1,axis=0)  pos\_df\_1 = np.sum(pos\_matrix\_1,axis=0)  negative\_1 = np.squeeze(np.asarray(neg\_df\_1))  positive\_1 = np.squeeze(np.asarray(pos\_df\_1))  term\_freq\_1 = pd.DataFrame([negative\_1,positive\_1],columns=cvec.get\_feature\_names\_out()).transpose()  # renaming the columns  term\_freq\_1.columns = ['neg','pos']  term\_freq\_1['total\_count'] = term\_freq\_1['neg'] + term\_freq\_1['pos']  term\_freq\_1['total'] = term\_freq\_1['pos'] + term\_freq\_1['neg']  term\_freq\_1['pos\_rate'] = term\_freq\_1['pos'] \* 1./term\_freq\_1['total']  term\_freq\_1['neg\_rate'] = term\_freq\_1['neg'] \* 1./term\_freq\_1['total']  term\_freq\_1['pos\_freq\_pct'] = term\_freq\_1['pos'] \* 1./term\_freq\_1['pos'].sum()  term\_freq\_1['neg\_freq\_pct'] = term\_freq\_1['neg'] \* 1./term\_freq\_1['neg'].sum()  # min-max :  term\_freq\_1['pos\_rate\_norm'] = (term\_freq\_1['pos\_rate'] - term\_freq\_1['pos\_rate'].min()) / (term\_freq\_1['pos\_rate'].max() - term\_freq\_1['pos\_rate'].min())  term\_freq\_1['pos\_freq\_norm'] = (term\_freq\_1['pos\_freq\_pct'] - term\_freq\_1['pos\_freq\_pct'].min()) / (term\_freq\_1['pos\_freq\_pct'].max() - term\_freq\_1['pos\_freq\_pct'].min())  term\_freq\_1['neg\_rate\_norm'] = (term\_freq\_1['neg\_rate'] - term\_freq\_1['neg\_rate'].min()) / (term\_freq\_1['neg\_rate'].max() - term\_freq\_1['neg\_rate'].min())  term\_freq\_1['neg\_freq\_norm'] = (term\_freq\_1['neg\_freq\_pct'] - term\_freq\_1['neg\_freq\_pct'].min()) / (term\_freq\_1['neg\_freq\_pct'].max() - term\_freq\_1['neg\_freq\_pct'].min())  # add a small constant to avoid division by zero  constant = 1e-10  # Calculate harmonic mean  term\_freq\_1['pos\_harmonic\_mean'] = hmean(term\_freq\_1[['pos\_rate', 'pos\_freq\_pct']].replace(0, constant), axis=1)  term\_freq\_1['neg\_harmonic\_mean'] = hmean(term\_freq\_1[['neg\_rate', 'neg\_freq\_pct']].replace(0, constant), axis=1)  def normcdf(x):  return norm.cdf(x, x.mean(), x.std())  term\_freq\_1['pos\_rate\_normcdf'] = normcdf(term\_freq\_1['pos\_rate'])  term\_freq\_1['pos\_freq\_pct\_normcdf'] = normcdf(term\_freq\_1['pos\_freq\_pct'])  term\_freq\_1['pos\_normcdf\_hmean'] = hmean([term\_freq\_1['pos\_rate\_normcdf'], term\_freq\_1['pos\_freq\_pct\_normcdf']])  # cumulative distribution function  def cumulative\_distribution\_function(value):  return norm.cdf(value, value.mean(), value.std())  # cdf for 'pos\_rate' and 'pos\_freq\_pct'  term\_freq\_1['norm\_pos\_rate\_cdf'] = cumulative\_distribution\_function(term\_freq\_1['pos\_rate'])  term\_freq\_1['norm\_pos\_frequency\_cdf'] = cumulative\_distribution\_function(term\_freq\_1['pos\_freq\_pct'])  # cdf for 'neg\_rate' and 'neg\_freq\_pct'  term\_freq\_1['norm\_neg\_rate\_cdf'] = cumulative\_distribution\_function(term\_freq\_1['neg\_rate'])  term\_freq\_1['norm\_neg\_frequency\_cdf'] = cumulative\_distribution\_function(term\_freq\_1['neg\_freq\_pct'])  # harmonic mean of the two new columns:  term\_freq\_1['pos\_harmonic\_mean\_norm\_cdf'] = hmean([term\_freq\_1['norm\_pos\_rate\_cdf'], term\_freq\_1['norm\_pos\_frequency\_cdf']])  term\_freq\_1['neg\_harmonic\_mean\_norm\_cdf'] = hmean([term\_freq\_1['norm\_neg\_rate\_cdf'], term\_freq\_1['norm\_neg\_frequency\_cdf']])  fig = px.scatter(term\_freq\_1,  x="neg\_harmonic\_mean\_norm\_cdf",  y="pos\_harmonic\_mean\_norm\_cdf",  hover\_data=[term\_freq\_1.index])  fig.update\_layout(  title='Neg NormCDF Harmonic Mean vs Pos NormCDF Harmonic Mean',  xaxis=dict(title='Negative Rate and Frequency CDF Harmonic Mean'),  yaxis=dict(title='Pos Rate and Freq CDF Harmonic Mean'),  hovermode="closest")  fig.show() |