اونيۏرسيتي مليسيا ڤهڠ السلطان عبد الله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

Centre for
Mathematical Sciences
Pusat Sains Matematik

BSD3523:
MACHINE LEARNING

GROUP PROJECT REPORT
(Sentiment Analysis for Malaysian Political Tweets)

COMPANY NAME : SENTIMENTSPEAK SOLUTION

Authored by: TAN CHEK CHENG (SD21031)
NURUL ATHIRAH BINTI RAMLI (SD21015)
KHAIRUL IMRAN BIN KHALIP (SD21002)
MUHAMMAD HAZIQ AKMAL BIN LOKMAN (SD21060)
LEE FU FONG (SD21026)

Table of Contents

1.0 Introduction

1.1 About Company

Welcome to SentimentSpeak Solutions, a cutting-edge startup consulting company founded by a dynamic team passionate about leveraging data analytics to decode sentiments in the realm of Malaysian politics on Twitter. Our company is committed to addressing the existing gap in sentiment analysis research specific to Malay language tweets related to Malaysian politics. Our CEO, Tan Chek Cheng, serves as the visionary leader, steering the company's strategic direction and ensuring alignment with core objectives. A team of dynamic specialists complements Tan, including Lee Fu Fong, the proficient Data Scientist responsible for crafting advanced machine learning algorithms for sentiment analysis. Nurul Athirah Binti Ramli, our Data Analyst, meticulously oversees data pre-processing, optimizing datasets for precise sentiment analysis.

Our team's synergy is evident in the distinct roles shoulder by each member. Muhammad Haziq Akmal Bin Lokman, the Statistician, employs statistical methods to extract meaningful insights and ensures robust evaluation of machine learning models. Khairul Imran Bin Khalip, in the pivotal role of Product Manager, focuses on the overall commercialization of the model, crafting a user-friendly GUI and working with clients who need NLP related tasks involving Malay language. SentimentSpeak Solutions addresses the scarcity of specialized sentiment analysis models and intuitive interfaces, providing comprehensive insights for politicians, analysts, and the general public.

Our ongoing project involves developing a bespoke sentiment analysis model for Malay language tweets, specifically those related to Malaysian politics on Twitter. Leveraging diverse machine learning techniques, including Support Vector Machine, Naive Bayes, Logistic Regression, Random Forest, and Long Short-Term Memory Networks, our goal is to offer a nuanced understanding of public sentiment in the realm of Malaysian politics on Twitter.

SentimentSpeak Solutions is poised to revolutionise sentiment analysis, contributing invaluable tools and insights to the field of Malaysian politics on social media. With a dedicated team and a clear vision, we aspire to reshape how sentiments are analysed and understood in the digital landscape. Our journey involves bridging the gap in sentiment analysis research, fostering accessibility and accuracy, and creating a lasting impact on the comprehension of public sentiments in the intricate domain of Malaysian politics.

1.2 Project Description

Twitter, a global microblogging and social networking platform, is a rich source of data for sentiment analysis due to its vast user base and the concise nature of tweets. Recognizing the significant presence of Malay-speaking users on Twitter and the limited research conducted in the Malay language, we embarked on a project to develop a sentiment analysis model specifically for Malay language tweets. This project focuses on Malaysian political tweets, providing an unique perspective on public sentiment in a critical societal domain. The first stage consisted of assembling a large dataset of tweets from Twitter's API with an emphasis on political debate in Malaysia and the Malay language. Data preprocessing was performed to clean and prepare the textual content for analysis, encompassing tasks such as tokenization, removing hashtag symbol, stop words, user @mention, punctuations, 'RT' (retweet) words, emoji, and specific words. After eliminating redundant elements, such as retweets and commonly used phrases, the dataset undergoes refinement. This process involves the extraction of retweeted content, as it often duplicates text from other users. Given that sentiment analysis benefits from original, individual expressions, this step ensures a focus on unique content within the tweets.

The project's importance lies in its ability to provide insight into the public's thoughts in the essential field of Malaysian politics, especially among Malay-speaking Twitter users. This project focused on filling a gap in the research on sentiment analysis in the Malay language by utilising a variety of machine learning techniques. It also provided insights into the multifaceted views and attitudes that people in Malaysia had about political conversation on Twitter. The project's results have significant impact for politicians, analysts, and the general public. They may support decision-making procedures and provide a more thorough understanding of public sentiment about Malaysian politics in the realm of digital media.

In summary, various machine learning models, especially Support Vector Machine, Naive Bayes, Logistic Regression, Random Forest, and Long Short-Term Memory Networks, were trained using this dataset. Each model was put via an in-depth evaluation process to see how well it could reliably classify sentiment in the context of

Malay political tweets from Malaysia using measures including accuracy, precision, recall, and F1-score.

1.3 Problem Statement

Even though there are a lot of Malay-speaking users on Twitter and the site is widely used for political debate in Malaysia, there is a distinct lack of sentiment analysis research that is explicitly focused on tweets in the Malay language that are related to politics in Malaysia. Accurately evaluating the public's sentiments towards Malaysian politics in the internet domain is difficult due to a lack of comprehensive research and specialised sentiment analysis models developed for this linguistic and geopolitical context. Politicians, analysts, and the general public are unable to have an in-depth comprehension of the diverse thoughts and feelings that are communicated on Twitter about Malaysian political discourse in the Malay language due to the lack of such trained models.

Furthermore, the absence of a user-friendly Graphical User Interface (GUI) amplifies this issue, as it prevents direct user interaction with sentiment analysis algorithms. Without an uncomplicated interface, people don't have an easy way to interact with and understand the results of sentiment analysis for their text inputs regarding Malaysian political discussions in Malay. As a result, the current gap in specialised sentiment analysis models and intuitive interfaces limits the accessibility and accuracy in comprehending sentiments expressed in Malay tweets about Malaysian politics.

Moreover, there are not many tools that understand the Malay language well enough for sentiment analysis, especially in Malaysian politics. The tools we have now often miss the subtleties and different ways Malaysians express feelings, making it hard to tell if a word is negative, neutral, or positive.

This implies a significant challenge for us which we demand tools that truly understand the online debate about politics of Malaysians. The various terms and phrases that are restricted to Malay are not captured by the existing tools. It is rarely

seen to find a tool that is proficient in Malay, particularly in the field of politics. Creating one would help people use their language to express feelings accurately while encouraging comprehending Malaysians' perspectives on political matters.

1.4 Project Scope

The project is situated within the rapidly evolving landscape of social media analytics, specifically focusing on Twitter, a global platform for microblogging and social networking. Twitter, with its 368 million active users per month as of 2022, provides a rich source of data for sentiment analysis. However, despite the significant presence of Malay-speaking users on Twitter, there is a notable gap in research of sentiment analysis tools for the Twitter user that spoke in Malay language. This is primarily due to the lack of research on sentiment analysis which focused only on political issues in Twitter's application. This project, therefore, aims to bridge this gap by focusing on understanding politics' matters in Malay language tweets, particularly those related to Malaysian politics.

Plus, we are making a friendly interface where people can type in their text, and the tool will tell them how people might feel about it. We are working hard to make sure this interface understands Malay well, so it can figure out the feelings expressed in Malaysian political tweets on Twitter. We will keep trying and fixing things to make this interface easy to use and accurate, making it a special tool to understand what people really think about politics in Malaysia.

1.5 Objectives

The objectives of this project are:

i. To develop a Sentiment Analysis Model for Malay Language Tweets: The primary objective of the project is to develop a sentiment analysis model that can accurately classify Malay language tweets into positive or negative sentiments.

ii. To compare and evaluate the performance of five different models: Naive Bayes, Logistic Regression, Support Vector Machine, Random Forests, and Long Short-Term Memory Networks.

iii. To develop a GUI for the best evaluated model, for better user experience where users can interact directly with the model by inserting their texts and observing the predicted sentiment.

1.6 Limitation of Project

One limitation of this project could be the dependency on the quality and quantity of available data. If the dataset lacks diversity or is limited in size, it might affect the model's ability to generalise well to all variations of Malay language used in political tweets. The model's capacity to generalise to any variation of Malay language used in political tweets could be limited if the dataset is small or unorganised. Moreover, the accuracy of the sentiment analysis could be impacted if the dataset doesn't adequately record all nuances of sentiments expressed in Malaysian political discourse. Other than that, the complexity of language and cultural nuances may be another problem. Like any other language, Malay has idioms, sarcasm, context-specific meanings, and nuances that the models may find difficult to appropriately comprehend. Ensuring the sentiment analysis models understand these nuances completely might be a persistent challenge.

2.0 Project Workflow

2.1 Machine Learning Pipeline

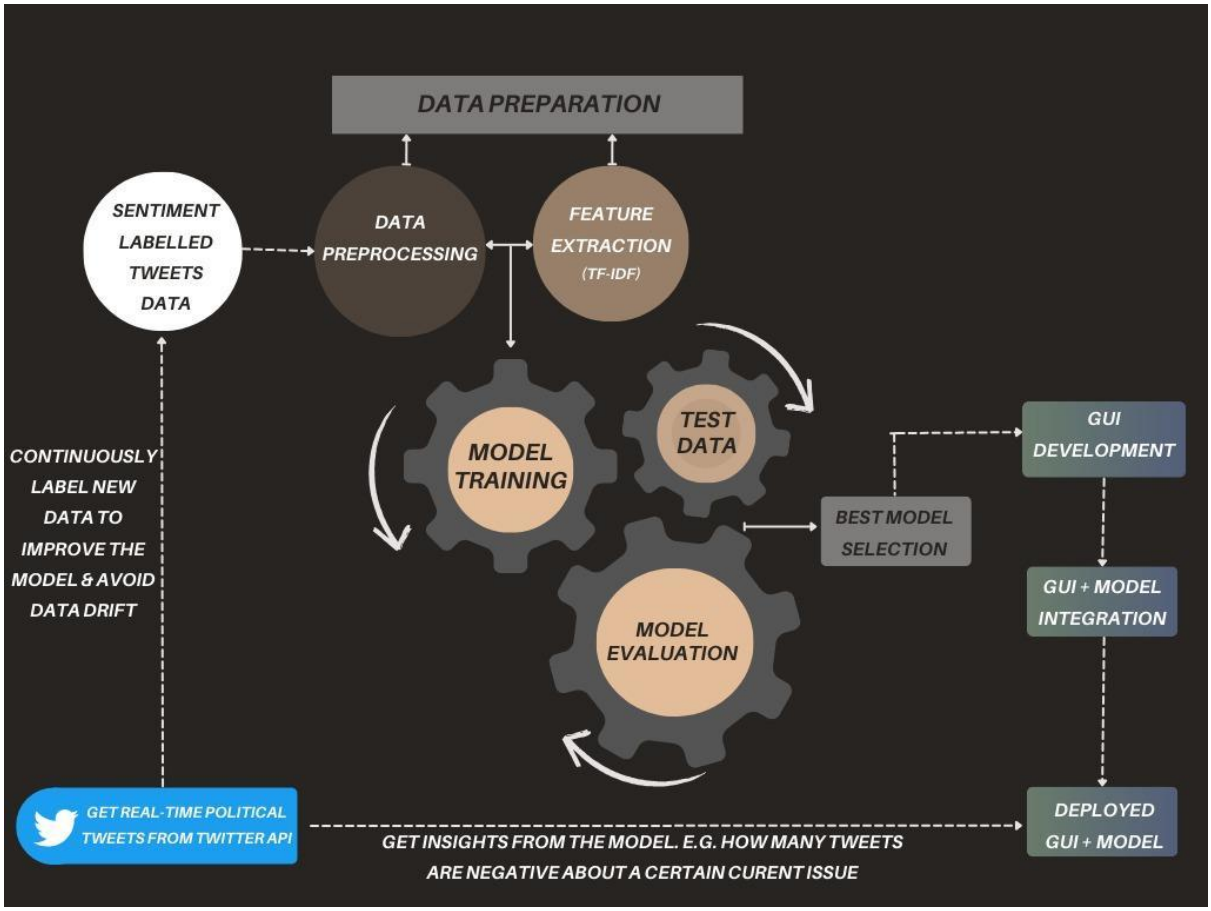


Figure 2.1.1 Machine Learning Pipeline

This ML pipeline represents a dynamic and robust system that not only performs sentiment analysis on tweet data but also adapts to new data trends and provides actionable insights into public opinion on political matters.

1. *Sentiment Labelled Tweets Data*
   The pipeline starts with a dataset of tweets that have already been labelled according to sentiment.

2. *Data Preparation*

- Data Preprocessing
  This step involves cleaning the tweet data by removing special characters, links, stopwords, hashtags, and emojis to standardize the text.

- Feature Extraction

Two methods are utilized here. The `CountVectorizer` is used to convert the text data into a matrix of token counts. Meanwhile, the `TfidfVectorizer` is employed to transform the text into a matrix of TF-IDF features.

3. *Model Training*
   Using the features extracted, five different machine learning models are trained: Naive Bayes, Support Vector Machine (SVM), Logistic Regression, Random Forest, and Long Short-Term Memory (LSTM) networks.

4. *Model Evaluation*

- Test Data
  A designated set of labelled tweets, which the models have not previously encountered, is used to assess their performance.

- *Best Model Selection*
  Each model is evaluated using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, Mean Squared Error (MSE), Receiver Operating Characteristic (ROC) curve, and Area Under the ROC Curve (AUC). The LSTM model emerges as the best performer across these metrics and is selected for deployment.

5. *GUI Development*
   A graphical user interface is developed, creating an interactive platform for the sentiment analysis application.

6. *GUI + Model Integration*
   The LSTM model is integrated into the GUI, making it a user-friendly application for sentiment analysis.

7. *Deployed GUI + Model*
   The application with the integrated GUI and LSTM model is then deployed, making it accessible for users to analyze sentiments of real-time tweets.

8. *Continuous Improvement*
   The model is continuously updated with new labelled data to improve its accuracy and to mitigate the issue of data drift, ensuring that the model stays relevant over time.

9. *Real-Time Data Acquisition*
   The deployed application uses the Twitter API to obtain real-time political tweets, which are then fed into the LSTM model for sentiment analysis.

10. *Insights and Analytics*
    The application provides insights, such as quantifying the volume of negative tweets about specific current political issues, thereby offering a snapshot of public sentiment in real-time.

2.2 Data Description

The data for this project comprises Malay tweets, each annotated with a sentiment label (Positive, Neutral, or Negative) by human annotators. The dataset includes several attributes:

Table 2.2.1 Data Description of Malaysian Political Tweets with Malay Language

ATTRIBUTES	DATA TYPE	EXAMPLE	DESCRIPTION
text	String	"Apsal Mayor Paris nak sibuk urusan politik Malaysia ni. Hello?"	The actual text content of the tweet in Malay language.
id	integer	"80741"	A unique identifier for each tweet.
sentiment	String	"Neutral","Negative"," Positive"	The sentiment or emotional tone expressed in the tweet.
annotator	integer	"27"	The identifier of the annotator who assigned the sentiment label to the tweet.
annotation_id	Integer	"4335"	An unique identifier assigned to each sentiment annotation.
created_at	datetime	2022-02-15T08:08:13.568662Z	The timestamp indicates when the sentiment annotation was created.

updated_at	datetime	2022-02-15T08:08:13.568662Z	The timestamp indicates when the sentiment annotation was last updated.
lead_time	float	1.87	The duration between the creation and update of the sentiment annotation.

The project's primary goal is to provide a robust and accurate sentiment analysis tool for Malay language tweets, thereby contributing to the broader field of machine learning models and offering valuable insights for various stakeholders, including political candidates, parties, and companies conducting market research.

2.3 Data Pre-Processing

In this assignment, our main goal is to get the tweets ready for sentiment analysis by doing some data preprocessing. We are specifically focusing on the steps needed to clean up the tweets for analysis.

First things first, we are getting rid of common stopwords. Those words that pop up a lot but do not really tell us much about sentiment. We have got a list of these words, and we are going to carefully document each one that we have removed.

Next up, we are aiming for pure text by waving goodbye to emojis. We will walk you through exactly how we are figuring out which emojis are in the tweets and then getting rid of them. It is a bit technical, but we will make sure it makes sense.

Besides stopwords and emojis, there's other stuff in the tweets that will not help us understand sentiment. We are cleaning house by identifying and removing these irrelevant elements. We will spell out what these elements are and why we are tossing them out.

Now, here's the interesting part – we're keeping certain words, like 'tak', even though they're technically considered stopwords. Why? Because these words can seriously affect sentiment, so they get a pass. We will explain the reasoning behind keeping them and why they matter in the grand scheme of sentiment analysis

2.4 Data Analysis

(a) Sentiment Class Distribution

Before:

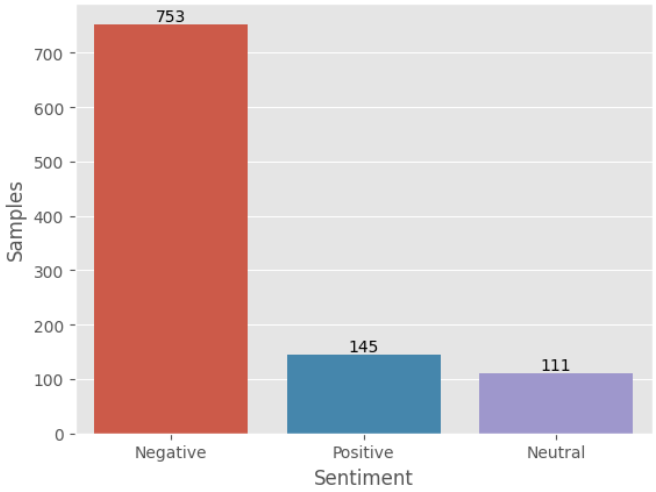


Figure 2.4.1 Class Distribution Before Resampling

A bar plot was created to visualise the distribution of sentiments that shows the number of positive, neutral, and negative tweets, indicating *imbalance sentiment distribution*. This visualisation was crucial for training a well-performing sentiment analysis model. If we train a model on a dataset with more negative than positive labels, the model might learn that predicting "negative" most of the time leads to the correct answer. As a result, when the trained model is used to predict sentiment on new, unseen data, it may be biassed towards predicting it as negative, even when that's not true.

After:

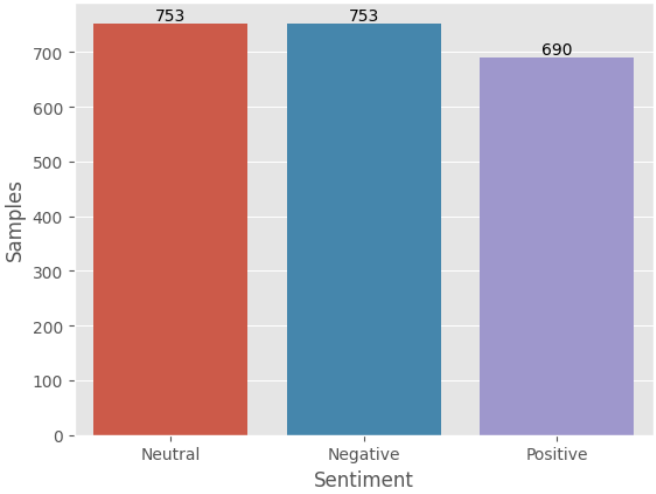


Figure 2.4.2 Class Distribution After Resampling

The dataset is balanced by resampling. Based on figure 2.4.2, as we can see a class distribution above is the balanced sentiment distribution with equal number of positive and negative labels. Hence, the model is less likely to develop a bias towards predicting a specific sentiment. This balance in the data ensures that the model learns to make accurate predictions without favouring one class over the other. We have another dataset which is a scraped data of Malay tweets. The difference between this dataset and previous one is these tweets are random, not specifically political tweets. We will use this data to populate our data frame so that the data is balanced.

The reason we choose a resampling method to handle the imbalance data is because the other way has its pros and cons. There are few ways that could have been used such as undersampling , oversampling and combination of both. Undersampling might decrease the overall performance of the model due to loss of information. As a matter of fact, undersampling requires removing some data from the majority class to balance the classes. On the contrary, an oversampling method may lead to overfitting as it makes exact copies of existing data. Over sampling will duplicate examples from the minority class or generate new synthetic examples. A common technique for this is SMOTE. Otherwise, a combination of both can be performed simultaneously. For instance,

undersampling for the majority class, oversampling for the minority class. To sum up, we use a resampling method to avoid the above issues.

(b) Number of characters in tweets

It represents the length of a tweet in terms of the number of characters used.

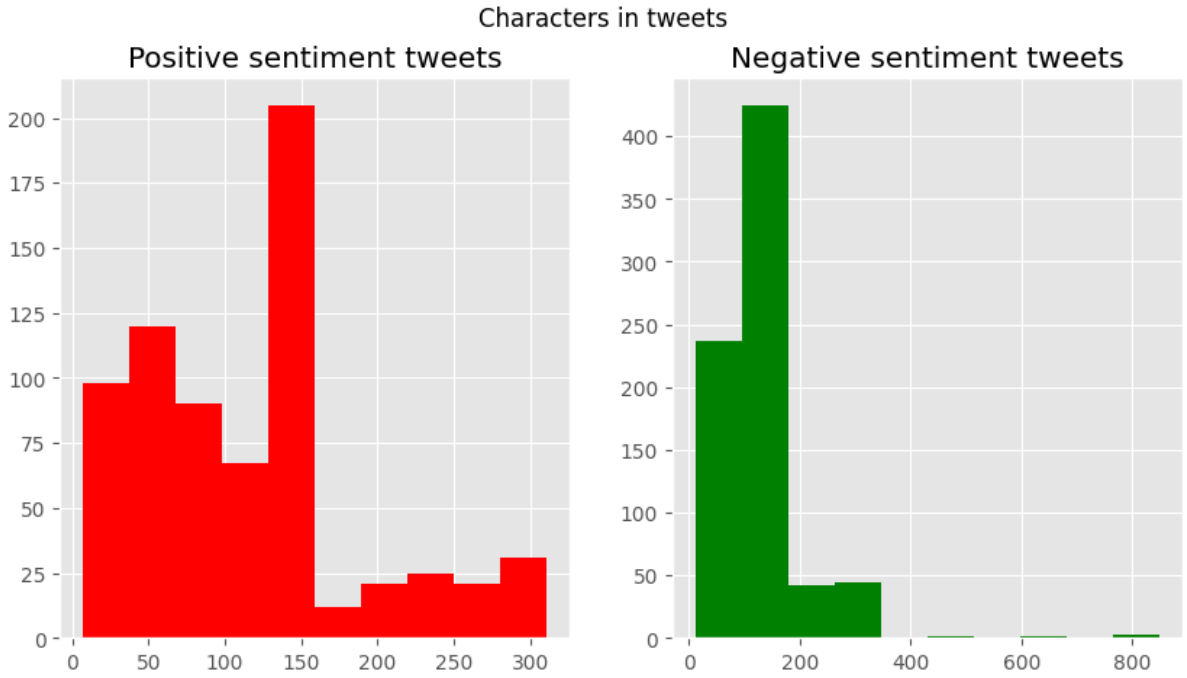


Figure 2.4.3 Number of Characters Positive Sentiment Tweets and Negative Sentiment Tweets

Based on figure 2.4.3 for Positive sentiment tweets, we can see the highest number of characters that were used in it were above 200 characters for almost 150 tweets while for Negative sentiment over 400 characters were used in 100 to 150 tweets shows that Negative sentiment really gives a lot of complaints in a tweet. Obviously, Negative sentiment tweets uses more characters than Positive sentiment tweets.

(c) Number of Words in Tweets:

It refers to the count of words present in each tweet, representing the length of a tweet in terms of the number of words used.

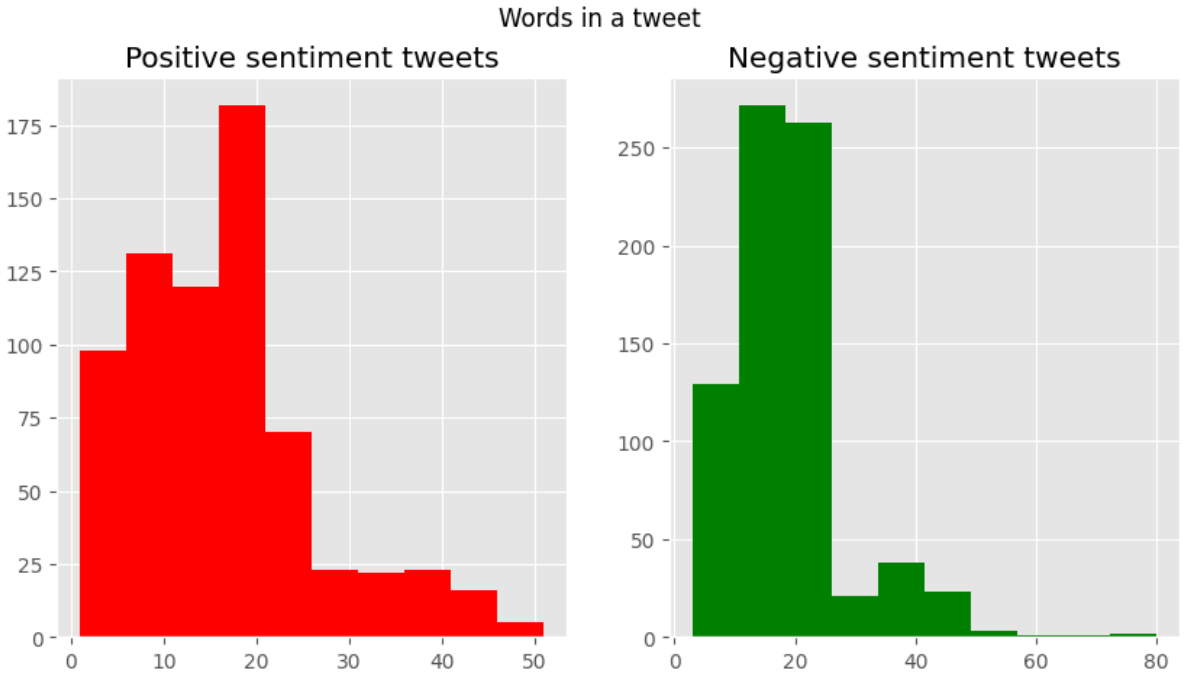


Figure 2.4.4 Number of Words Positive Sentiment Tweets and Negative Sentiment Tweets

To be compared, Negative sentiment tweet uses more words the Positive sentiment tweets. The highest number of words used in Positive and Negative sentiment tweets is more than 175 and 250 respectively. The number of Negative tweets the used most word count is in the range of 10 to 30 and about 20 positive tweets. On the contrary, the least word count also used in Negative sentiment tweet. 60 to 80 Negative tweets used the least words.

(d) Average Word Length in Each Tweets:

The average word length in a tweet serves as a metric that provides insights into the linguistic characteristics and writing style of users.

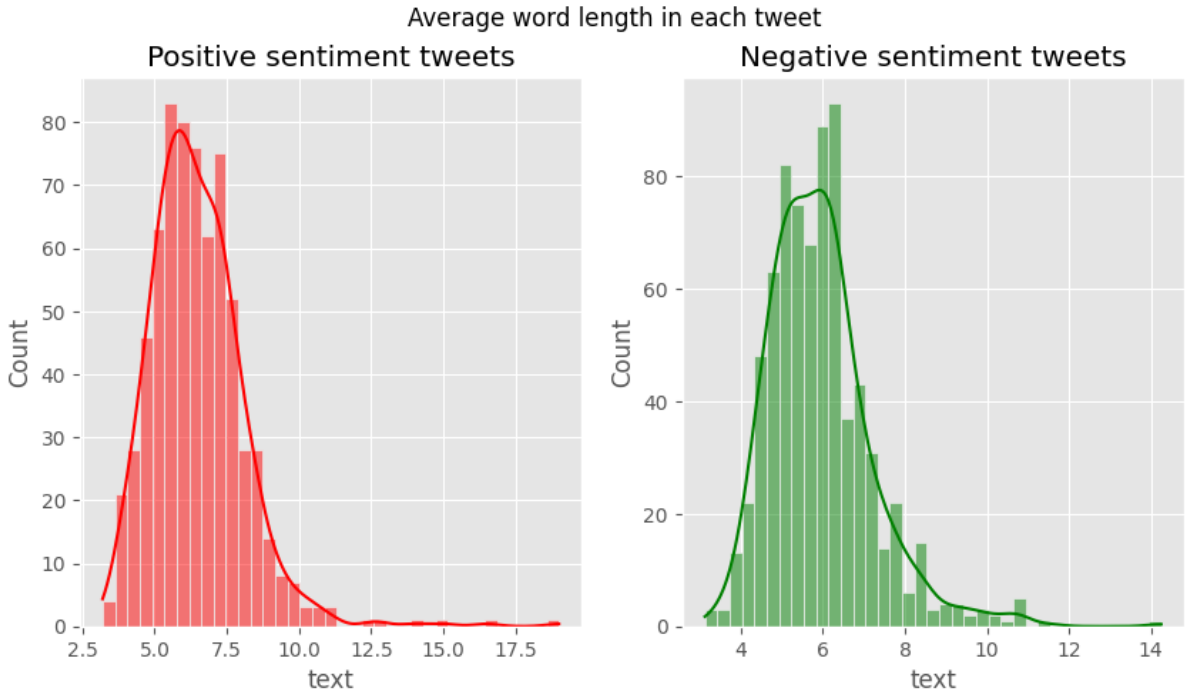


Figure 2.4.5 Average of Word Length Positive Sentiment Tweets and Negative Sentiment Tweets

The highest average word length of the negative sentiment tweets are more than positive tweets. Both graphs are right skewed. Positive sentiment tweets graph is at the peak when the word length is 5 but negative sentiment tweet graph is at the peak when the word length is 6. To sum up, tweets expressing negative sentiment tend to have longer word length than those conveying positive sentiment.

(e) Word Cloud in all tweets:

A Word Cloud is a visual representation of text data where the size of each word corresponds to its frequency or importance in the given text. In the context of all tweets, a Word Cloud provides a summarised and visually appealing way to understand the most common or significant words used in the entire set of tweets.

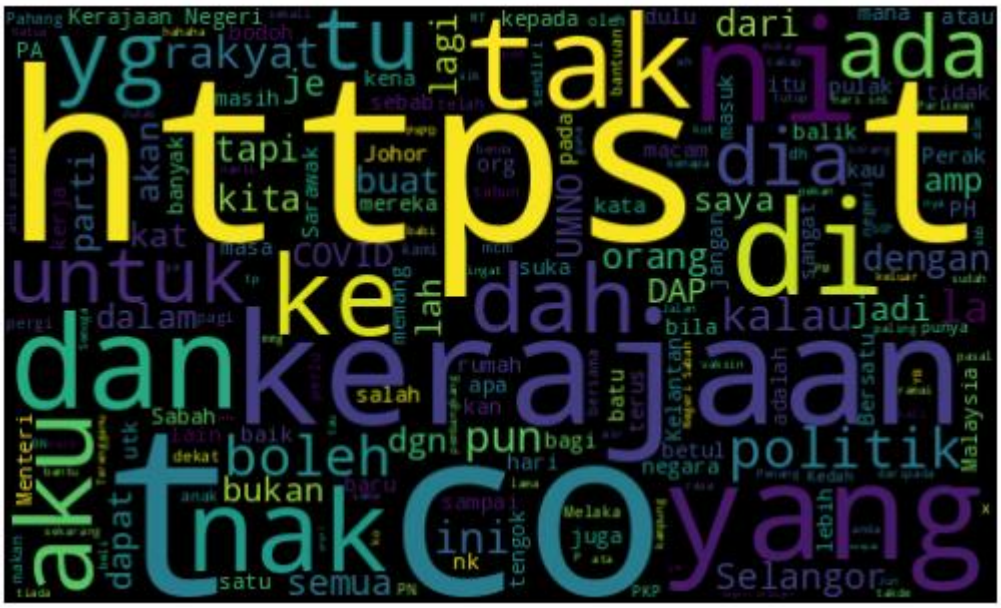


Figure 2.4.6 Word Cloud

The bigger the size of the words in Word Cloud the more frequent the words are used. The word in the biggest size is ‘https’ which also refers to the primary protocol used to send data between a web browser and a website , which means website links are mostly included in both classes of tweets. Word ‘kerajaan’ , is moderately used in the tweets. The least used words are ‘makan’ and ‘bantu’. In addition, there are few states included in the Word Cloud like ‘Kelantan’ , ‘Selangor’ ,’Sabah’ ‘Melaka’and ‘Sarawak’. The most mentioned will be the state of ‘Selangor’. Political party names like ‘DAP’ ,’PH’, ‘PN’ and ‘UMNO’ also appear in WordCloud where DAP and UMNO are the biggest.

(f) N-gram Analysis

A technique used in NLP to analyze the sequential patterns of words in a text. An n-gram can be thought of as a sliding window of n words that moves across the text, extracting each group of words it encounters. For example, let's consider the sentence: "I love to code."

A 1-gram (unigram) analysis would involve looking at each individual word separately: "I", "love", "to", "code". A 2-gram (bigram) analysis would involve examining pairs of words: "I love", "love to", "to code". A 3-gram (trigram) analysis would involve considering groups of three consecutive words: "I love to", "love to code".

By analyzing these n-grams, we can gain insights into the frequency and patterns of word combinations within a text. For example, in the bigram "tak puas," the word "tak" changes the sentiment of "puas" to negative.

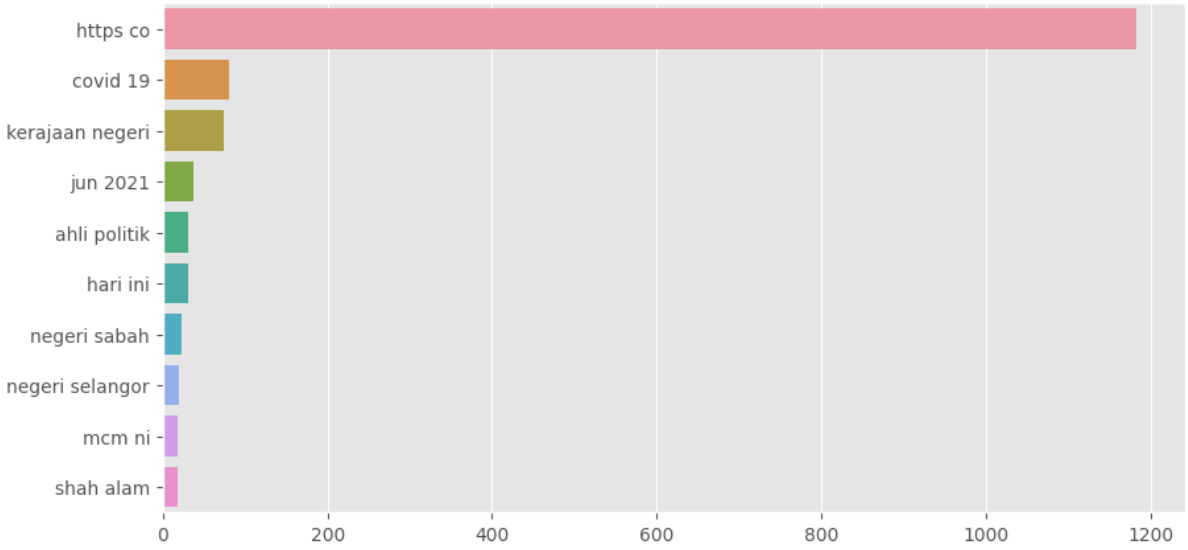


Figure 2.4.7 N-Gram Analysis

(g) List of Malay Stopwords

```
# List of malay stop words
STOP_WORDS = set("""
ada adalah adanya adapun agak agaknya agar akan akankah akhir akhiri akhirnya
aku akulah amat amatlah anda andalah antar antara antaranya apa apaan apabila
apakah apalagi apatah artinya asal asalkan atas atau ataukah ataupun awal
awalnya
bagai bagaikan bagaimana bagaimanakah bagaimanapun bagi bagian bahkan bahwa
bahwasanya baik bakal bakalan balik banyak byk bapak baru bawah beberapa begini
beginian beginikah beginilah begitu begitukah begitulah begitupun bekerja
belakang belakangan belum belumlah benar benarkah benarlah berada berakhir
berakhirlah berakhirnya berapa berapakah berapalah berapapun berarti berawal
berbagai berdatangan beri berikan berikut berikutnya berjumlah berkali-kali
berkata berkehendak berkeinginan berkenaan berlainan berlalu berlangsung
berlebihan bermacam bermacam-macam bermaksud bermula bersama bersama-sama
bersiap bersiap-siap bertanya bertanya-tanya berturut berturut-turut bertutur
berujar berupa besar betul betulkah biasa biasanya bila bilakah bisa bisakah
boleh bolehkah bolehlah buat bukan bukankah bukanlah bukannya bulan bung
cara caranya cukup cukupkah cukuplah cuma
dahulu dalam dan dapat dari daripada datang dekat demi demikian demikianlah
dengan dgn depan di dia diakhiri diakhirinya dialah diantara diantaranya diberi
diberikan diberikannya dibuat dibuatnya didapat didatangkan digunakan
diibaratkan diibaratkannya diingat diingatkan diinginkan dijawab dijelaskan
dijelaskannya dikarenakan dikatakan dikatakannya dikerjakan diketahui
diketahuinya dikira dilakukan dilalui dilihat dimaksud dimaksudkan
dimaksudkannya dimaksudnya diminta dimintai dimisalkan dimulai dimulailah
dimulainya dimungkinkan dini dipastikan diperbuat diperbuatnya dipergunakan
diperkirakan diperlihatkan diperlukan diperlukannya dipersoalkan dipertanyakan
dipunyai diri dirinya disampaikan disebut disebutkan disebutkannya disini
disinilah ditambahkan ditandaskan ditanya ditanyai ditanyakan ditegaskan
ditujukan ditunjuk ditunjuki ditunjukkan ditunjukkannya ditunjuknya dituturkan
dituturkannya diucapkan diucapkannya diungkapkan dong dua dulu
empat enggak enggaknya entah entahlah
guna gunakan
```

Figure 2.4.8 (a) List of Malay Stopwords

```
hal hampir hanya hanyalah hari harus haruslah harusnya hendak hendaklah
hendaknya hingga
ia ialah ibarat ibaratkan ibaratnya ibu ikut ingat ingat-ingat ingin inginkah
inginkan ini inikah inilah itu itukah itulah
jadi jadilah jadinya jangan jangankan janganlah jauh jawab jawaban jawabnya
jelas jelaskan jelaslah jelasnya jika jikalau juga jumlah jumlahnya justru
kala kalau kalaulah kalaupun kalian kami kamilah kamu kamulah kan kapan
kapankah kapanpun karena karenanya kasus kata katakan katakanlah katanya ke
keadaan kebetulan kecil kedua keduanya keinginan kelamaan kelihatan
kelihatannya kelima keluar kembali kemudian kemungkinan kemungkinannya kenapa
kepada kepadanya kesampaian keseluruhan keseluruhannya keterlaluan ketika
khususnya kini kinilah kira kira-kira kiranya kita kitalah kok kurang
lagi lagian lah lain lainnya lalu lama lamanya lanjut lanjutnya lebih lewat
lima luar
macam maka makanya makin malah malahan mampu mampukah mana manakala manalagi
masa masalah masalahnya masih masihkah masing masing-masing mau maupun
melainkan melakukan melalui melihat melihatnya memang memastikan memberi
memberikan membuat memerlukan memihak meminta memintakan memisalkan memperbuat
mempergunakan memperkirakan memperlihatkan mempersiapkan mempersoalkan
mempertanyakan mempunyai memulai memungkinkan menaiki menambahkan menandaskan
menanti menanti-nanti menantikan menanya menanyai menanyakan mendapat
mendapatkan mendatang mendatangi mendatangkan menegaskan mengakhiri mengapa
mengatakan mengatakannya mengenai mengerjakan mengetahui menggunakan
menghendaki mengibaratkan mengibaratkannya mengingat mengingatkan menginginkan
mengira mengucapkan mengucapkannya mengungkapkan menjadi menjawab menjelaskan
menuju menunjuk menunjuki menunjukkan menunjuknya menurut menuturkan
menyampaikan menyangkut menyatakan menyebutkan menyeluruh menyiapkan merasa
mereka merekalah merupakan meski meskipun meyakini meyakinkan minta mirip
misal misalkan misalnya mula mulai mulailah mulanya mungkin mungkinkah
nah naik namun nanti nantinya nyaris nyatanya
oleh olehnya
pada padahal padanya pak paling panjang pantas para pasti pastilah penting
pentingnya per percuma perlu perlukah perlunya pernah persoalan pertama
pertama-tama pertanyaan pertanyakan pihak pihaknya pukul pula pun punya
rasa rasanya rata rupanya
saat saatnya saja sajalah saling sama sama-sama sambil sampai sampai-sampai
```

Figure 2.4.8 (b) List of Malay Stopwords

```
sampaikan sana sangat sangatlah satu saya sayalah se sebab sebabnya sebagai
sebagaimana sebagainya sebagian sebaik sebaik-baiknya sebaiknya sebaliknya
sebanyak sebegini sebegitu sebelum sebelumnya sebenarnya seberapa sebesar
sebetulnya sebisanya sebuah sebut sebutlah sebutnya secara secukupnya sedang
sedangkan sedemikian sedikit sedikitnya seenaknya segala segalanya segera
seharusnya sehingga seingat sejak sejauh sejenak sejumlah sekadar sekadarnya
sekali sekali-kali sekalian sekaligus sekalipun sekarang sekarang sekecil
seketika sekiranya sekitar sekitarnya sekurang-kurangnya sekurangnya sela
selain selaku selalu selama selama-lamanya selamanya selanjutnya seluruh
seluruhnya semacam semakin semampu semampunya semasa semasih semata semata-mata
semaunya sementara semisal semisalnya sempat semua semuanya semula sendiri
sendirian sendirinya seolah seolah-olah seorang sepanjang sepantasnya
sepantasnyalah seperlunya seperti sepertinya sepihak sering seringnya serta
serupa sesaat sesama sesampai sesegera sesekali seseorang sesuatu sesuatunya
sesudah sesudahnya setelah setempat setengah seterusnya setiap setiba setibanya
setidak-tidaknya setidaknya setinggi seusai sewaktu siap siapa siapakah
siapapun sini sinilah soal soalnya suatu sudah sudahkah sudahlah supaya
tadi tadinya tahu tahun tambah tambahnya tampak tampaknya tandas tandasnya
tanpa tanya tanyakan tanyanya tapi tegas tegasnya telah tempat tengah tentang
tentu tentulah tentunya tepat terakhir terasa terbanyak terdahulu terdapat
terdiri terhadap terhadapnya teringat teringat-ingat terjadi terjadilah
terjadinya terkira terlalu terlebih terlihat termasuk ternyata tersampaikan
tersebut tersebutlah tertentu tertuju terus terutama tetap tetapi tiap tiba
tiba-tiba tidakkah tidaklah tiga tinggi toh tunjuk turut tutur tuturnya
ucap ucapnya ujar ujarnya umum umumnya ungkap ungkapnya untuk usah usai
waduh wah wahai waktu waktunya walau walaupun wong
yaitu yakin yakni yang yg
""".split()
)
```

Figure 2.4.8 (c) List of Malay Stopwords

A stop word function is a function or method used in NLP to filter out stop words from text. Stop words are common words that typically do not carry much meaning and are often removed during text analysis or text mining tasks. Removing stop words helps reduce noise and focuses on more meaningful and significant words in the text.

Although, 'tidak','tak', 'takde', 'takda', 'tk', 'x', are stopwords, we do not remove them from the training data because it has important meaning as it can flip the sentiment. For example, 'not good' is a negative sentiment, but if we remove 'not' as it is technically a stopword, we will get 'good' which is a positive sentiment.

(h) The most common stopwords in Negative tweets

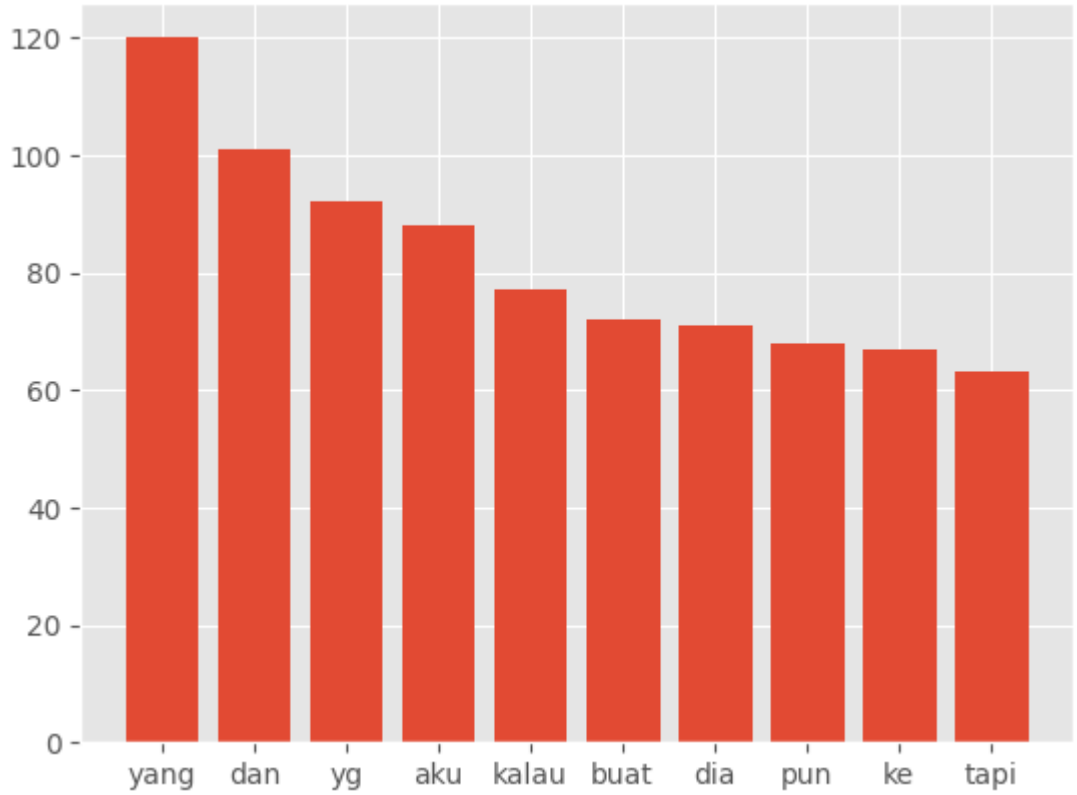


Figure 2.4.9 Most Common Stopwords in Negative Tweets

By identifying figure 2.4.9, you can gain insights into the language patterns and characteristics of negative sentiment expressed in the dataset. This analysis can help in tasks such as sentiment analysis, understanding user feedback, or identifying common negative expressions.

(i) The most common stopwords in Positive tweets

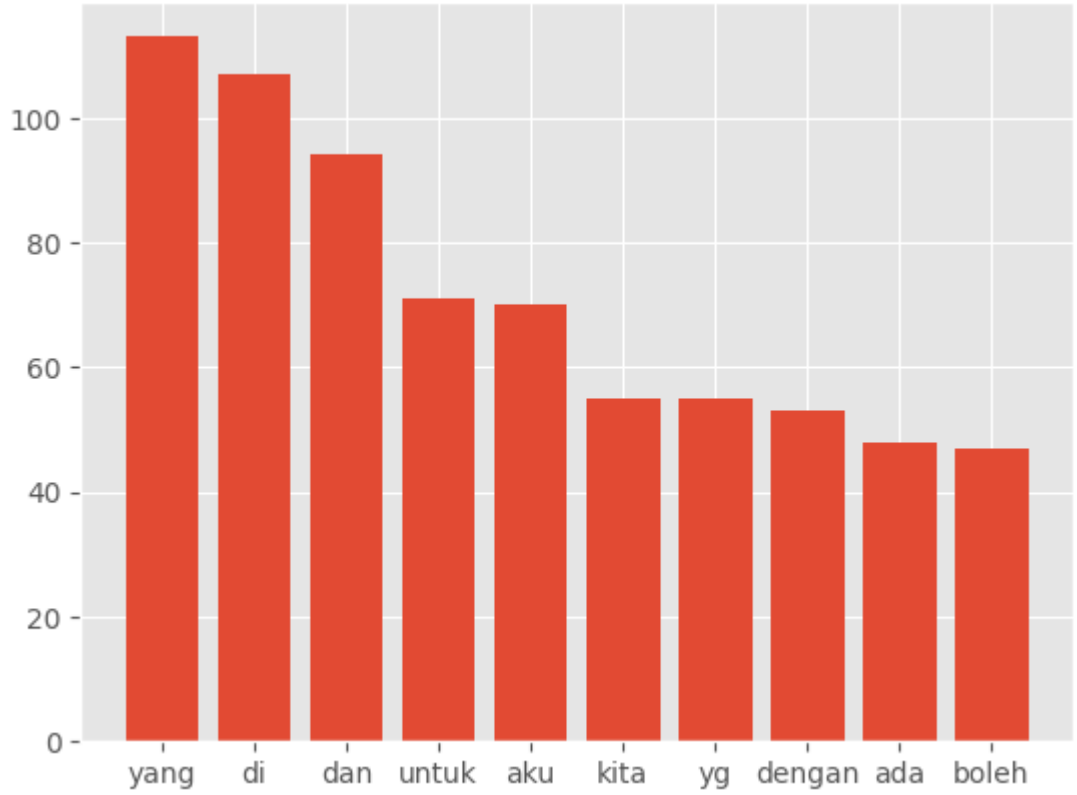


Figure 2.4.10 Most Common Stopwords in PositiveTweets

The purpose of this function is to identify and analyse the stopwords that commonly appear in positive tweets. By identifying these stopwords, you can gain insights into the language patterns and characteristics of positive sentiment expressed in the dataset. This analysis can help in tasks such as sentiment analysis, understanding user feedback, or identifying common positive expressions.

(j) The Most Common Words (Excl Stopwords) in Negative and Positive Tweets:

Negative:

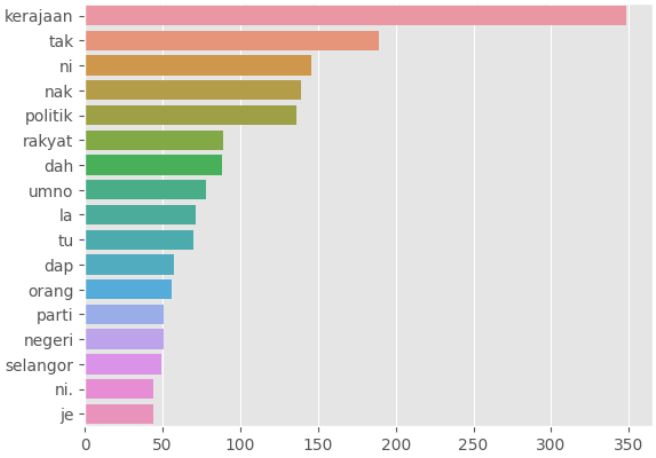


Figure 2.4.11(a) The Common Words in Negative Tweets

Positive:

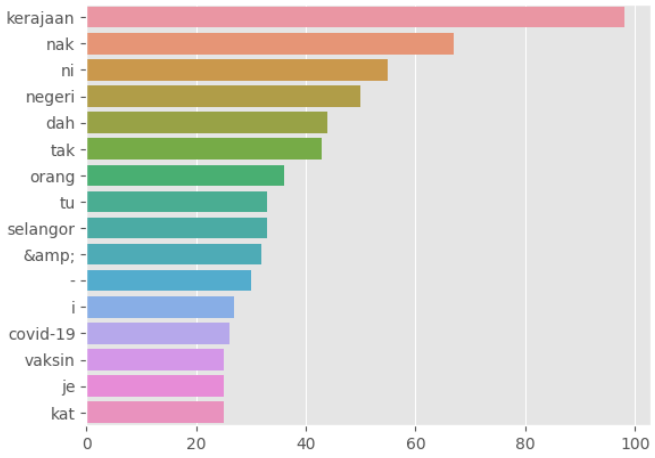


Figure 2.4.11(b) The Common Words in Positive Tweets

The most common words, excluding stopwords, in both negative and positive tweets were analysed. This analysis provided insights into the words that are commonly associated with negative and positive sentiments.

The analysis of these results showed that the sentiment analysis model was capable of accurately predicting the sentiment of Malay language tweets. The model was sensitive to the nuances of the Malay language, including the use of slang and

dialect differences. The visualisations created during the exploratory data analysis phase provided valuable insights into the data, such as the distribution of sentiments, the length of tweets, and the common words used in negative tweets. These insights were crucial in understanding the data and developing a well-performing sentiment analysis model. The successful prediction of sentiments by the model demonstrated its potential applications in various fields, including politics and market research.

(k) Word Cloud: Before vs After Data Cleaning

Figure 2.4.12 Word Cloud: Before vs After Data Cleaning

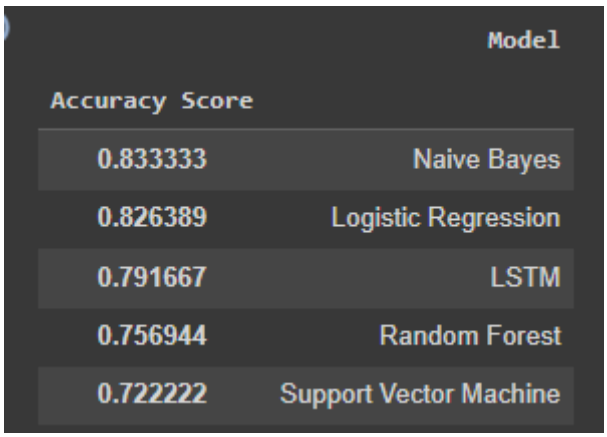
2.5 Machine Learning Algorithm

In developing the machine learning algorithm after comprehensive data preprocessing and meticulous data analysis, The sentiment analysis process undertaken here is characterized by a meticulous blend of feature engineering, model training, and thorough evaluation across diverse classifiers. Two crucial lists, namely BAD_WORDS and GOOD_WORDS, have been meticulously crafted to discern negative and positive sentiments, facilitating a granular analysis of prevalent words in each sentiment category. Moving beyond, an array of classifiers, including Naive Bayes, LSTM (Deep Learning RNN), Random Forest, SVM, and Logistic Regression, have been employed. Each classifier undergoes an exhaustive training process, culminating in predictions and comprehensive evaluations utilizing a diverse set of metrics. The LSTM model, a deep learning approach, involves intricate steps such as data reshaping and the creation of a neural network through the Keras framework.

The evaluation metrics deployed span traditional measures like accuracy, classification report, ROC AUC score, and ROC curve, complemented by regression-based metrics like mean squared error (MSE) and its root mean squared error (RMSE). To enhance interpretability, insightful visualisations, particularly focused on ROC curves, have been meticulously crafted. These visual aids serve to facilitate an intuitive understanding of the delicate balance between true positive and false positive rates. This multifaceted approach ensures a robust sentiment analysis, unravelling the distinctive strengths and nuances inherent in each classifier under consideration. The entire codebase for the machine learning algorithm is also available for reference in the appendix located at the end of the report.

2.6 Interpretation of the results

In terms of interpreting the results, we compare the performance of all models through metrics such as accuracy, classification report, AUC and ROC curve, as well as MSE and RMSE.



Accuracy Score	Model
0.833333	Naive Bayes
0.826389	Logistic Regression
0.791667	LSTM
0.756944	Random Forest
0.722222	Support Vector Machine

Figure 2.6.1 Accuracy Score for 5 Models

The accuracy scores for the various models indicate the proportion of correctly predicted sentiments within the dataset. Naive Bayes achieved the highest accuracy with a score of 83.33%, followed closely by Logistic Regression at 82.64%. The LSTM model demonstrated a respectable accuracy of 79.17%. Random Forest and Support Vector Machine yielded lower accuracy scores of 75.69% and 72.22%, respectively. This suggests that Naive Bayes and Logistic Regression performed relatively well in

capturing the sentiment patterns present in the data, outperforming the other models in terms of overall accuracy.

Model	precision	recall	f1-score	support
Naive Bayes	0.833620	0.833333	0.830707	144
Logistic Regression	0.825927	0.826389	0.826117	144
LSTM	0.792677	0.791667	0.786132	144
Random Forest	0.761794	0.756944	0.758455	144
SVM	0.746061	0.722222	0.725252	144

Figure 2.6.2 Model Performance for each Sentiment Analysis Classifier

The classification report provides a comprehensive overview of the model performance for each sentiment analysis classifier. Precision represents the accuracy of positive predictions, recall indicates the ability to capture all positive instances, and the F1-score is a harmonic mean that balances precision and recall. Additionally, support denotes the number of instances in each class.

Naive Bayes and Logistic Regression exhibit comparable performance, with both models achieving high precision, recall, and F1-scores around 83%, 83%, and 83%, respectively. This indicates their effectiveness in correctly identifying positive and negative sentiments. The LSTM model, although slightly trailing behind, maintains a solid performance with precision, recall, and F1-score values around 79%. Random Forest and SVM exhibit lower precision, recall, and F1-scores, suggesting a comparatively less accurate prediction of sentiment.

Overall, this detailed examination of the classification report underscores the nuanced performance of each model across various evaluation metrics. Naive Bayes and Logistic Regression emerge as top performers, while LSTM demonstrates strong competence. Random Forest and SVM, although providing reasonable predictions, exhibit a lower level of accuracy in comparison to the other models.

Model	AUC
LSTM	0.860657
Logistic Regression	0.816999
Naive Bayes	0.813672
Random Forest	0.753479
SVM	0.733817

Figure 2.6.3 AUC Values of each 5 Models

The Area Under the Curve (AUC) values for the Receiver Operating Characteristic (ROC) curves offer insights into the models' ability to discriminate between positive and negative sentiments. A higher AUC suggests a better trade-off between true positive rate and false positive rate, indicating superior model performance.

In this context, the LSTM model stands out with an AUC of 0.860657, signifying strong discrimination capabilities. Logistic Regression and Naive Bayes closely follow with AUC values of 0.816999 and 0.813672, respectively, reinforcing their effectiveness in distinguishing sentiments. Random Forest and SVM exhibit lower AUC values of 0.753479 and 0.733817, respectively, indicating a relatively weaker discriminatory power compared to the other models.

In summary, the AUC results align with the overall model performance, highlighting the LSTM model as the most effective in sentiment discrimination, followed closely by Logistic Regression and Naive Bayes. Random Forest and SVM, while still providing reasonable discrimination, demonstrate a comparatively lower ability to distinguish between positive and negative sentiments.

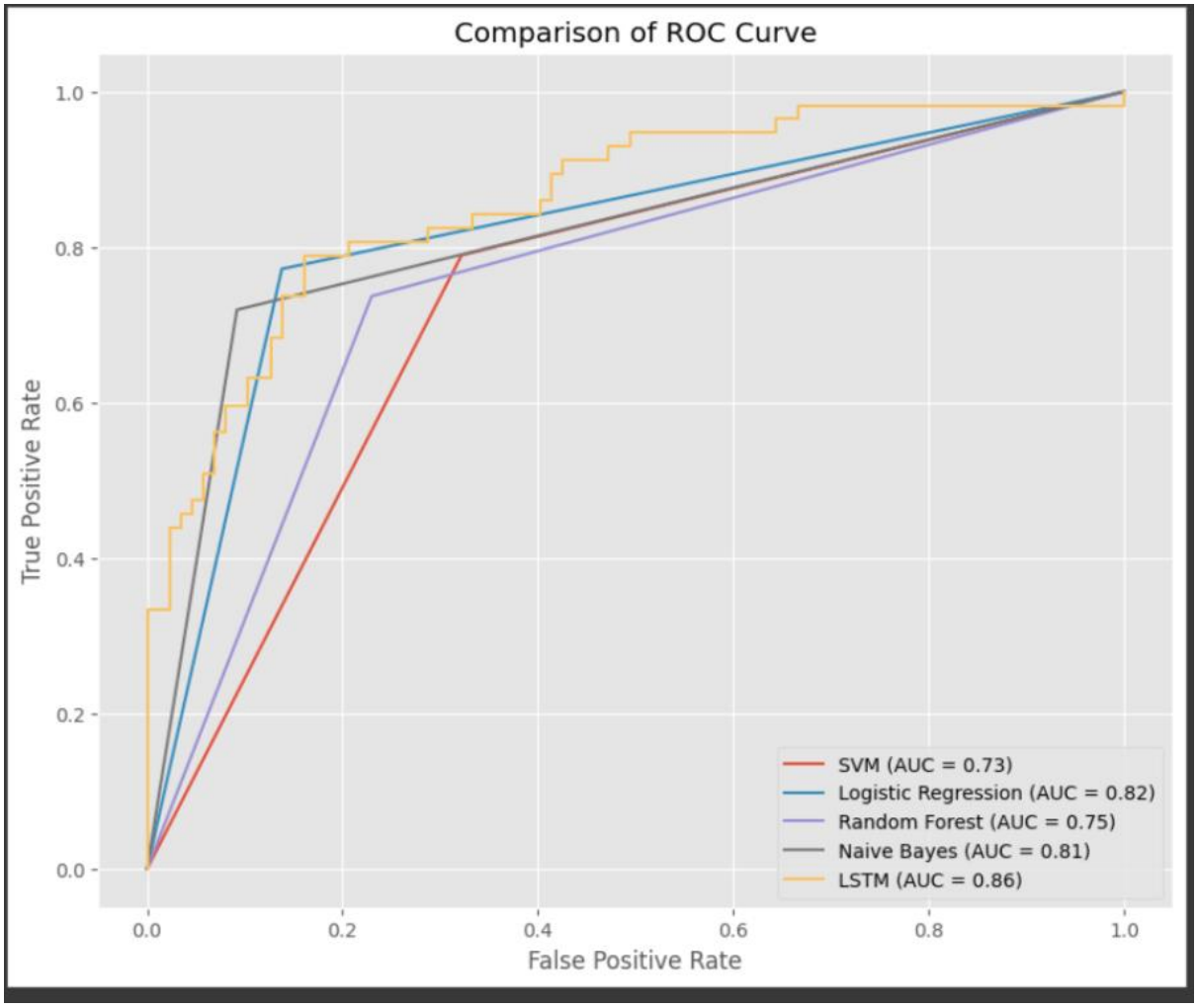


Figure 2.6.4 Comparison of ROC Curve each 5 Models

This ROC curve illustrates the comparative performance of multiple machine learning models in a sentiment analysis task, including SVM, Logistic Regression, Random Forest, Naive Bayes, and LSTM. The y-axis represents the true positive rate (sensitivity), while the x-axis represents the false positive rate (1-specificity). Among the showcased models, SVM exhibits the lowest Area Under Curve (AUC) value at 0.73, indicating a relatively diminished ability to differentiate between positive and negative sentiments.

On the other hand, Logistic Regression performs notably better with an AUC of 0.82, showcasing a balanced identification of both true positives and negatives. Random Forest falls in between SVM and Logistic Regression, boasting a 0.75 AUC, signifying better performance than SVM but not as strong as Logistic Regression in this sentiment

analysis task. Naive Bayes closely aligns with Logistic Regression, demonstrating a slightly lower AUC of 0.81, emphasizing its effectiveness in accurate sentiment classification.

Remarkably, LSTM emerges as the top-performing model with the highest AUC value of 0.86, showcasing unparalleled effectiveness in distinguishing between positive and negative sentiments. In summary, the sentiment analysis project highlights LSTM as the most potent model for this task, evident from its superior AUC value compared to other models.

Model	MSE	RMSE
LSTM	0.159977	0.399972
Naive Bayes	0.166667	0.408248
Logistic Regression	0.173611	0.416667
Random Forest	0.243056	0.493007
SVM	0.277778	0.527046

Figure 2.6.5 (MSE) and (RMSE) Values each 5 Models

The Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) values provide insights into the predictive accuracy and precision of the machine learning models in the sentiment analysis task.

LSTM exhibits the lowest MSE of 0.159977 and the corresponding RMSE of 0.399972. These values suggest that the LSTM model's predictions are, on average, close to the actual sentiments, indicating a high level of accuracy and minimal prediction errors. The lower MSE and RMSE for LSTM compared to other models imply superior performance in minimizing the squared differences between predicted and actual sentiment values.

Naive Bayes follows closely with an MSE of 0.166667 and RMSE of 0.408248. This indicates a slightly higher prediction error compared to LSTM but still suggests good predictive accuracy.

Logistic Regression demonstrates an MSE of 0.173611 and RMSE of 0.416667. While marginally higher than Naive Bayes, it reflects a reasonable level of accuracy and precision in sentiment predictions.

Random Forest exhibits a higher MSE of 0.243056 and RMSE of 0.493007, indicating a relatively larger prediction error compared to the previous models. Although the error is higher, it is still within an acceptable range, showcasing the model's ability to make reasonably accurate predictions.

SVM shows the highest MSE of 0.277778 and RMSE of 0.527046, suggesting a larger overall prediction error compared to the other models. While SVM's performance in sentiment analysis is acceptable, the higher MSE and RMSE values indicate a somewhat lower precision in predicting sentiment compared to the other models.

In summary, the MSE and RMSE metrics provide a quantitative assessment of the models' predictive accuracy, with lower values indicating better performance. LSTM stands out with the lowest MSE and RMSE, suggesting superior accuracy in sentiment predictions, while SVM shows a comparatively higher prediction error.

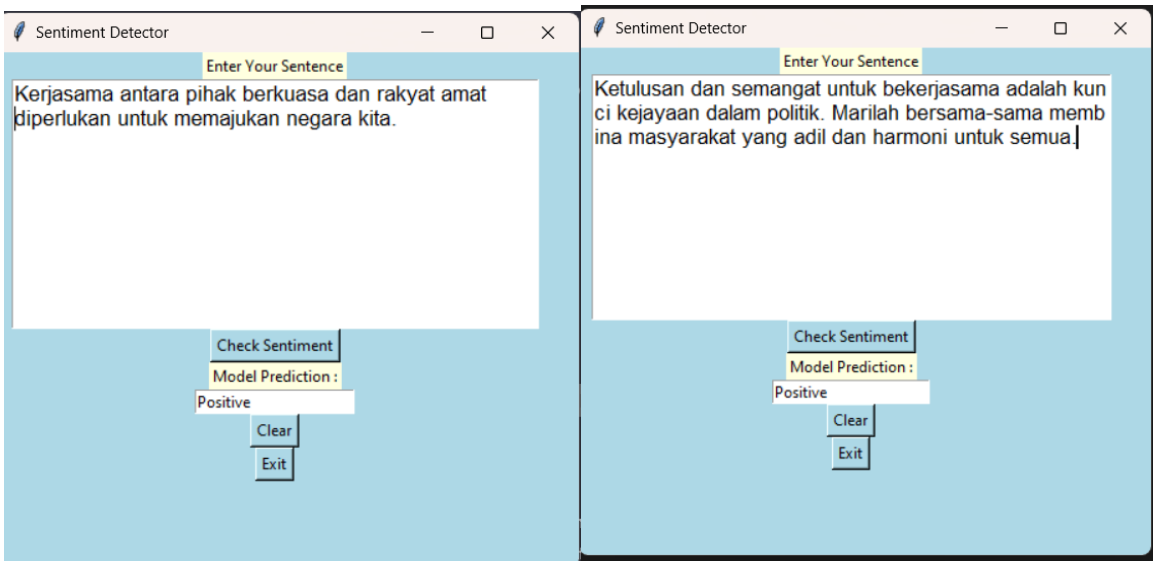
Conclusion of interpretation:

The comparative analysis of various machine learning models for sentiment analysis reveals the LSTM model as the overall top performer. It demonstrates exceptional accuracy, precision, recall, and F1-score values, along with a high AUC in the ROC curve. Additionally, it exhibits the lowest MSE and RMSE values, indicating superior predictive accuracy and minimal prediction errors.

Logistic Regression and Naive Bayes also perform remarkably well, with comparable metrics to LSTM, showcasing their strong capabilities in sentiment analysis. Random Forest and SVM provide reasonable predictions, but their performance falls behind the top-performing models.Based on these findings, LSTM emerges as the optimal choice for sentiment analysis tasks, offering superior performance in capturing sentiment patterns and making accurate predictions.

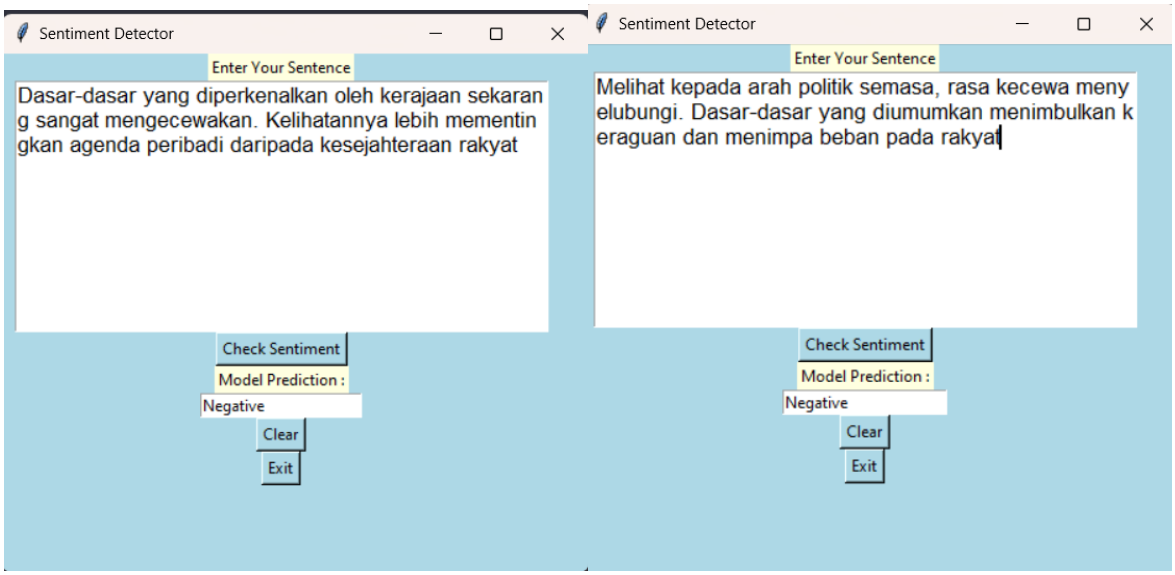
Moreover, this comprehensive analysis highlights the importance of evaluating models using multiple metrics to gain a holistic understanding of their performance. Accuracy alone may not provide a complete picture, and examining metrics such as precision, recall, F1-score, AUC, MSE, and RMSE provides a more nuanced assessment of model capabilities.

GUI -Positive Tweets



During our collaborative study, we inputted data encompassing a range of Malaysian political tweets. The machine learning model demonstrated accurate predictions for sentiment analysis based on the entered tweets. This successful outcome underscores the reliability and effectiveness of the model in assessing and predicting sentiment of Malaysian political tweets in diverse scenarios.

GUI- Negative Tweets



In contrast, our findings revealed that the predictive accuracy of the model extends beyond just identifying positive sentiment in Malaysian political tweets; it also demonstrates precision in predicting cases classified as negative sentiment. The model's ability to provide accurate predictions for both negative and positive sentiment instances underscores its robustness and reliability across diverse political viewpoints.

2.7 Conclusion and recommendation

In conclusion, the SentimentSpeak Solutions project has successfully bridged the gap in sentiment analysis research for Malay language tweets related to Malaysian politics on Twitter. Leveraging a combination of machine learning techniques, including Support Vector Machine, Naive Bayes, Logistic Regression, Random Forest, and Long Short-Term Memory Networks (LSTM), the project aimed to provide a nuanced understanding of public sentiments specifically within the Malay-speaking audience. The LSTM model emerged as the top performer, demonstrating balanced accuracy, precision, recall, and AUC in the ROC curve. The developed graphical user interface (GUI) enhances accessibility and user interaction, contributing to a user-friendly sentiment analysis tool.

Predicting political sentiment in the Malaysian context offers numerous benefits, including empowering politicians with insights into public perceptions, assisting analysts in understanding the diverse attitudes of the population, and providing the general public with a tool to gauge sentiment in political discourse. The accurate identification of sentiment can support informed decision-making processes, enhance communication strategies, and contribute to a more transparent political landscape.

For future improvements and recommendation, the project can benefit from acquiring additional labelled data from Twitter, further enriching the model's training set and improving its generalization capabilities. Hyperparameter tuning for the LSTM, the current best-performing model, is recommended to optimize its performance. Additionally, future research endeavours could explore the utilization of large language models, tapping into the advancements in natural language processing and deep learning. These strategies collectively ensure the continued relevance, accuracy, and effectiveness of the sentiment analysis tool in navigating the intricate domain of Malaysian politics on social media.

3. References

The following references were consulted for this project:

1. A. (2021, May 10). *NLP 📝 GloVe, BERT, TF-IDF, LSTM. . . 📝 Explained*. Kaggle. https://www.kaggle.com/code/andreshg/nlp-glove-bert-tf-idf-lstm-explained/notebook
2. F. (2023, February 16). *Natural Language Processing (NLP) for Beginners*. Kaggle. https://www.kaggle.com/code/faressayah/natural-language-processing-nlp-for-beginners/notebook
3. N. (2018, June 5). *Python NLTK sentiment analysis*. Kaggle. https://www.kaggle.com/code/ngyptr/python-nltk-sentiment-analysis
4. P. (2020, May 28). *Getting started with NLP - A general Intro*. Kaggle. https://www.kaggle.com/code/parulpandey/getting-started-with-nlp-a-general-intro
5. Naw, N. (2018, October 12). *Twitter Sentiment Analysis Using Support Vector Machine and K-NN Classifiers*. International Journal of Scientific and Research Publications. https://doi.org/10.29322/ijsrp.8.10.2018.p8252
6. *Sentiment Analysis of Noisy Malay Text: State of Art, Challenges and Future Work*. (2020). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8967092

Source of dataset:

1. Malay political tweets data: https://github.com/huseinzol05/malay-dataset/blob/master/sentiment/supervised-twitter-politics/data.csv

2. Malay random tweets data: https://github.com/huseinzol05/malay-dataset/tree/master/sentiment/supervised-twitter

3. The stop words used in the analysis were extracted from a Malaysian contributor to the spaCy library. The details can be found in the following pull request: https://github.com/explosion/spaCy/pull/12602/files

3. Appendix

Exploratory Data Analysis :

```
[ ]  #!pip install emoji
```

```
import pandas as pd

from matplotlib import pyplot as plt
plt.style.use('ggplot')
import seaborn as sns
import numpy as np
from math import sqrt

from nltk.util import ngrams
from collections import defaultdict
from collections import  Counter
from wordcloud import WordCloud

import re
import string
import emoji
import gensim # Word Embeddings

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Models
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import LSTM, Input, Dense, Dropout

# Metrics
from sklearn import metrics
from sklearn.metrics import classification_report, accuracy_score, roc_curve, auc, mean_squared_error
```

Data Preview

```
[ ]  df = pd.read_csv('train.csv')

    # Check for null values in the DataFrame
    null_counts = df.isnull().sum()

    # Print the null value counts
    print(null_counts)

    print(len(df.index), ' rows of data is clean from Null values')

    df.head(3)
```

```
text            0
id              0
sentiment       0
annotator       0
annotation_id   0
created_at      0
updated_at      0
lead_time       0
dtype: int64
1009  rows of data is clean from Null values
```

1009 rows of data is clean from null values

	text	id	sentiment	annotator	annotation_id	created_at	updated_at	lead_time
0	Bersatu kita teguh, bercerai kita roboh.	81037	Neutral	27	4231	2022-02-14T15:58:43.745680Z	2022-02-14T15:58:47.080664Z	282799.830
1	semuanya bersatu Bersatu Indonesia Kuat	81001	Neutral	27	4195	2022-02-14T15:56:20.283992Z	2022-02-14T15:56:23.113936Z	282655.843
2	Dulu aku Jenis yang Tak suka sgt tgk mp-mp sho...	80998	Negative	27	4192	2022-02-14T15:55:57.558702Z	2022-02-14T15:56:03.446298Z	282636.179

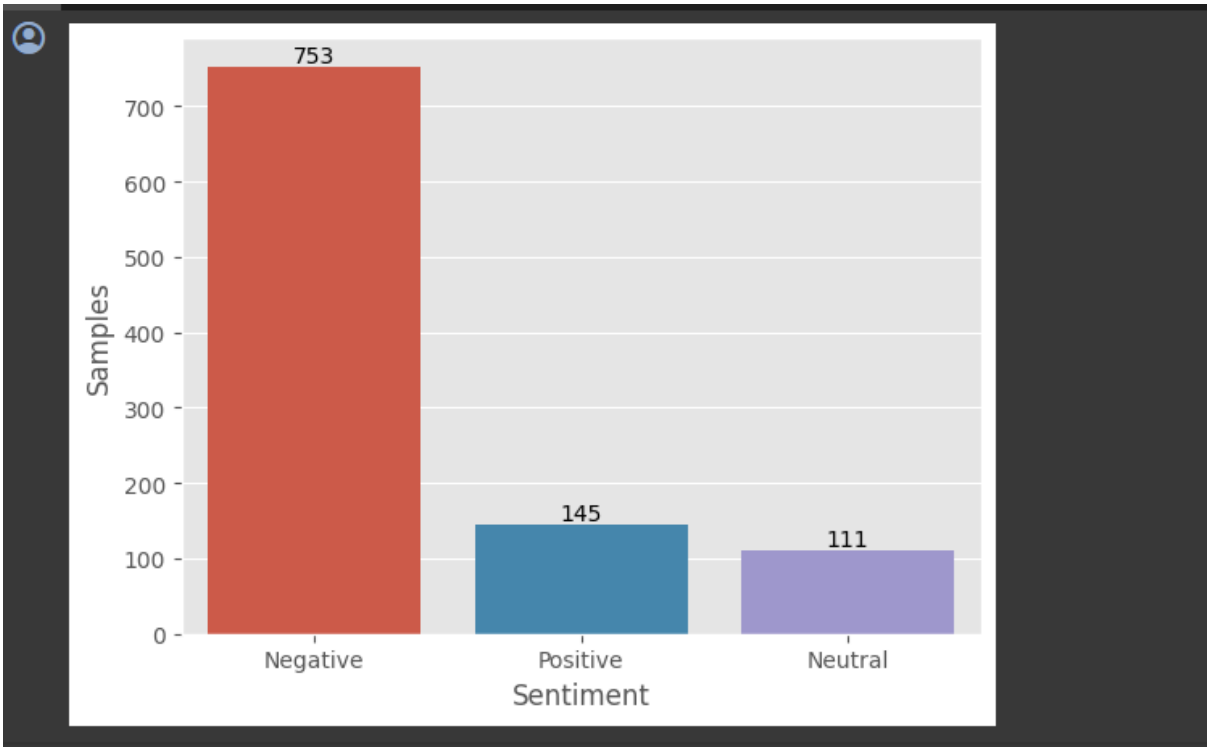
```
df
```

	text	id	sentiment	annotator	annotation_id	created_at	updated_at	lead_time
0	Bersatu kita teguh, bercerai kita roboh.	81037	Neutral	27	4231	2022-02-14T15:58:43.745680Z	2022-02-14T15:58:47.080664Z	282799.830
1	semuanya bersatu Bersatu Indonesia Kuat	81001	Neutral	27	4195	2022-02-14T15:56:20.283992Z	2022-02-14T15:56:23.113936Z	282655.843
2	Dulu aku Jenis yang Tak suka sgt tgk mp-mp sho...	80998	Negative	27	4192	2022-02-14T15:55:57.558702Z	2022-02-14T15:56:03.446298Z	282636.179
3	Kini360: Pengamal media Kelantan terima dos pe...	80968	Neutral	27	3923	2022-02-12T16:57:58.774007Z	2022-02-12T16:58:04.642062Z	113557.353
4	Kini360: Sabah lanjutkan PKP 3.0 hingga 28 Jun...	80907	Negative	27	3862	2022-02-12T16:53:52.561250Z	2022-02-12T16:53:57.517652Z	113310.237
...	...	...	...	...	...	...	...	...
1004	Kerajaan kita bukan serba boleh, tapi serba bo...	80189	Negative	27	2178	2022-01-28T04:16:20.626712Z	2022-01-28T04:16:20.626726Z	1.683
1005	Sudah sudah la kak mas woi. Kamu tahu tak semu...	80188	Negative	27	2177	2022-01-28T04:16:18.556006Z	2022-01-28T04:16:18.556020Z	1.015
1006	Nape nak kena tampal gambar Dia pulak...Guna d...	80187	Negative	27	2176	2022-01-28T04:16:17.145907Z	2022-01-28T04:16:17.145921Z	2.036
1007	Bendera putih tu bukannya nak harap bantuan ke...	80186	Negative	27	2175	2022-01-28T04:16:14.729682Z	2022-01-28T04:16:14.729696Z	2.969
1008	Menggelabah masing-masing nak beraya kan. Lepa...	80185	Negative	27	2174	2022-01-28T04:16:11.343549Z	2022-01-28T04:16:11.343588Z	34.702

1009 rows × 8 columns

Sentiment Class Distribution

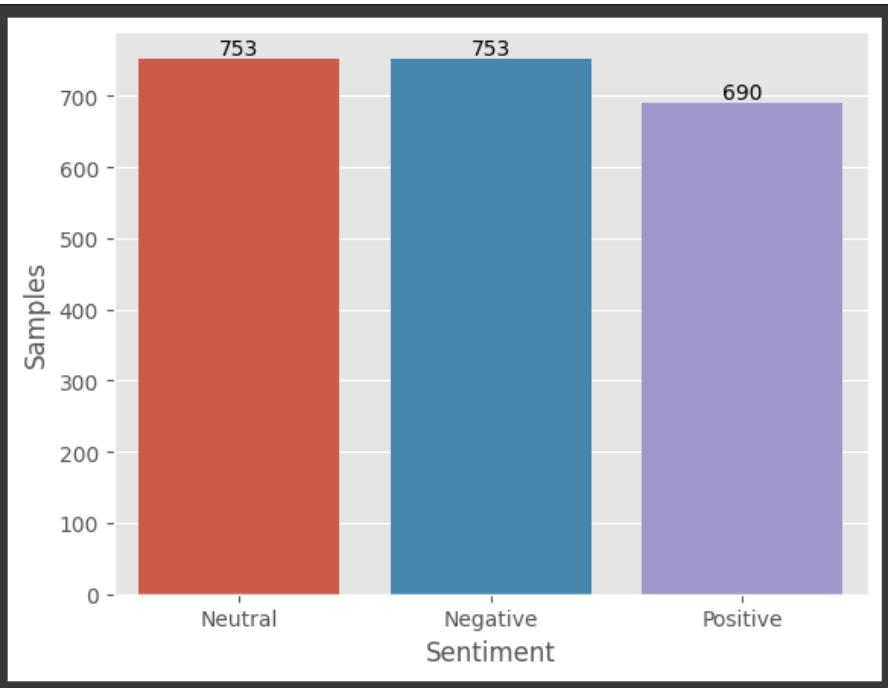
```
[ ] # Count the occurrences of each class
    class_counts = df['sentiment'].value_counts()

    # Create a bar plot
    ax = sns.barplot(x=class_counts.index, y=class_counts)
    plt.ylabel('Samples')
    plt.xlabel('Sentiment')

    # Add count annotations above each bar
    for i, count in enumerate(class_counts):
        ax.annotate(str(count), xy=(i, count), ha='center', va='bottom')

    plt.show()
```



Resampling

We have a 2nd csv which is a scraped data of malay tweets (the difference is these tweets are random, not specifically political tweets like 1st csv). We will use this data to populate our dataframe so that the data is balanced.

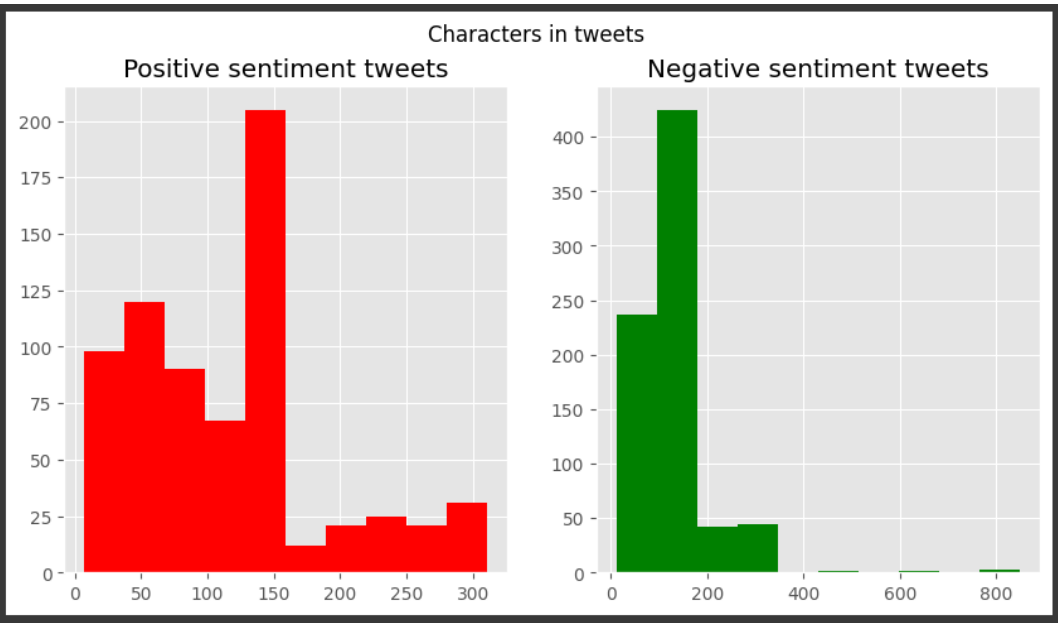
```
[ ]  # Load the second dataframe
     df2 = pd.read_csv('train2.csv')

     # Filter positive and neutral tweets from the second dataframe
     positive_tweets = df2[df2['sentiment'] == 'Positive']
     neutral_tweets = df2[df2['sentiment'] == 'Neutral']

     # Randomly sample 545 positive and 642 neutral tweets from the second df
     positive_tweets = positive_tweets.sample(min(len(positive_tweets), 545))
     neutral_tweets = neutral_tweets.sample(min(len(neutral_tweets), 642))

     # Concatenate the sampled tweets with the initial dataframe
     df = pd.concat([df, positive_tweets, neutral_tweets], ignore_index=True)

     # Count the occurrences of each class
     class_counts = df['sentiment'].value_counts()

     # Create a bar plot
     ax = sns.barplot(x=class_counts.index, y=class_counts)
     plt.ylabel('Samples')
     plt.xlabel('Sentiment')

     # Add count annotations above each bar
     for i, count in enumerate(class_counts):
         ax.annotate(str(count), xy=(i, count), ha='center', va='bottom')

     plt.show()
```



Exploratory Data Analysis

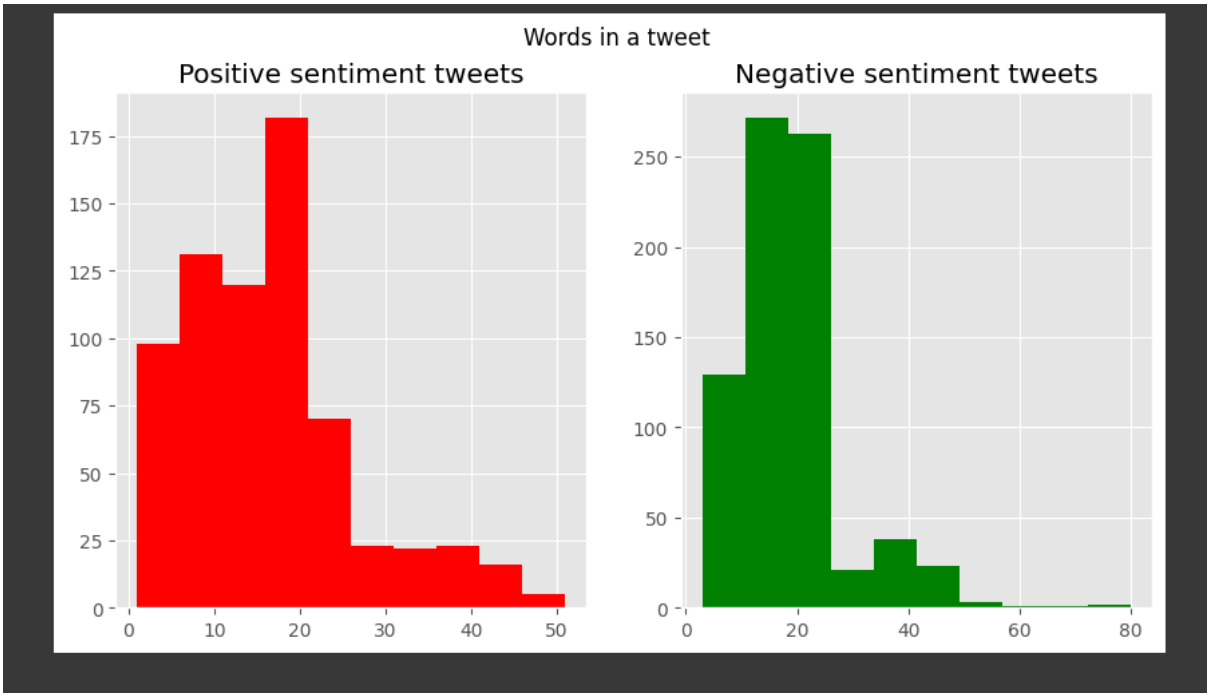
Number of characters in tweets

```
[ ]  # Create subplots
     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

     # Filter tweets with sentiment value 1 (positive) and calculate the length of text
     tweet_len_pos = df[df['sentiment'] == 'Positive']['text'].str.len()
     ax1.hist(tweet_len_pos, color='red')
     ax1.set_title('Positive sentiment tweets')

     # Filter tweets with sentiment value 0 (negative) and calculate the length of text
     tweet_len_neg = df[df['sentiment'] == 'Negative']['text'].str.len()
     ax2.hist(tweet_len_neg, color='green')
     ax2.set_title('Negative sentiment tweets')

     # Set overall title
     fig.suptitle('Characters in tweets')

     # Display the plot
     plt.show()
```



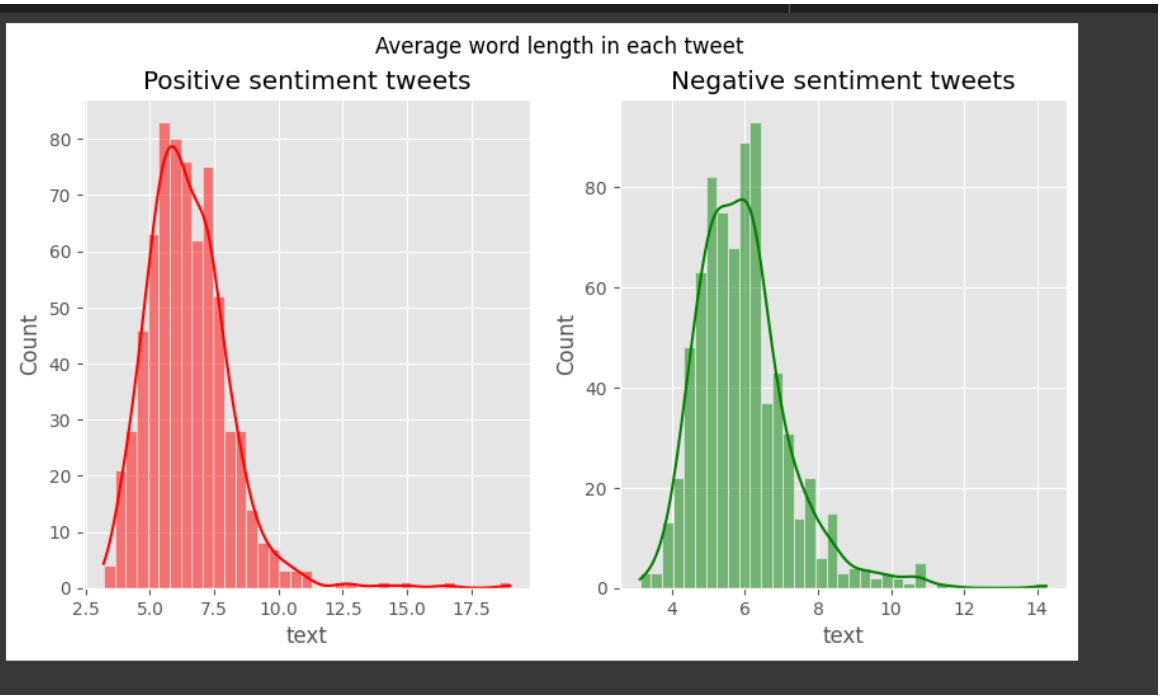
Number of words in tweets

```
[ ]  # Create subplots
     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

     # Filter tweets with sentiment 'positive' and calculate the number of words in text
     tweet_len_pos = df[df['sentiment'] == 'Positive']['text'].str.split().map(lambda x: len(x))
     ax1.hist(tweet_len_pos, color='red')
     ax1.set_title('Positive sentiment tweets')

     # Filter tweets with sentiment 'negative' and calculate the number of words in text
     tweet_len_neg = df[df['sentiment'] == 'Negative']['text'].str.split().map(lambda x: len(x))
     ax2.hist(tweet_len_neg, color='green')
     ax2.set_title('Negative sentiment tweets')

     # Set overall title
     fig.suptitle('Words in a tweet')

     # Display the plot
     plt.show()
```



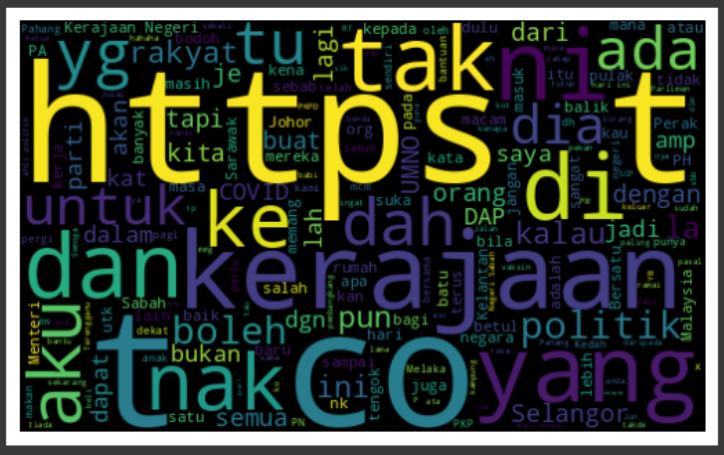
Average word length in a tweet

```
[ ]  # Create subplots
     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))

     # Filter tweets with sentiment 'positive' and calculate the average word length
     word_pos = df[df['sentiment'] == 'Positive']['text'].str.split().apply(lambda x: [len(i) for i in x])
     sns.histplot(word_pos.map(lambda x: np.mean(x)), ax=ax1, color='red', kde=True)
     ax1.set_title('Positive sentiment tweets')

     # Filter tweets with sentiment 'negative' and calculate the average word length
     word_neg = df[df['sentiment'] == 'Negative']['text'].str.split().apply(lambda x: [len(i) for i in x])
     sns.histplot(word_neg.map(lambda x: np.mean(x)), ax=ax2, color='green', kde=True)
     ax2.set_title('Negative sentiment tweets')

     # Set overall title
     fig.suptitle('Average word length in each tweet')

     # Display the plot
     plt.show()
```



Word Cloud in all tweets

```
[ ]  # Plot the Word Cloud
     allwords= ''.join( [twts for twts in df["text"]])
     wordCloud = WordCloud (width=500, height=300, random_state=21, max_font_size=119).generate(allwords)

     plt.imshow(wordCloud, interpolation = "bilinear")
     plt.axis('off')

     plt.show()
```



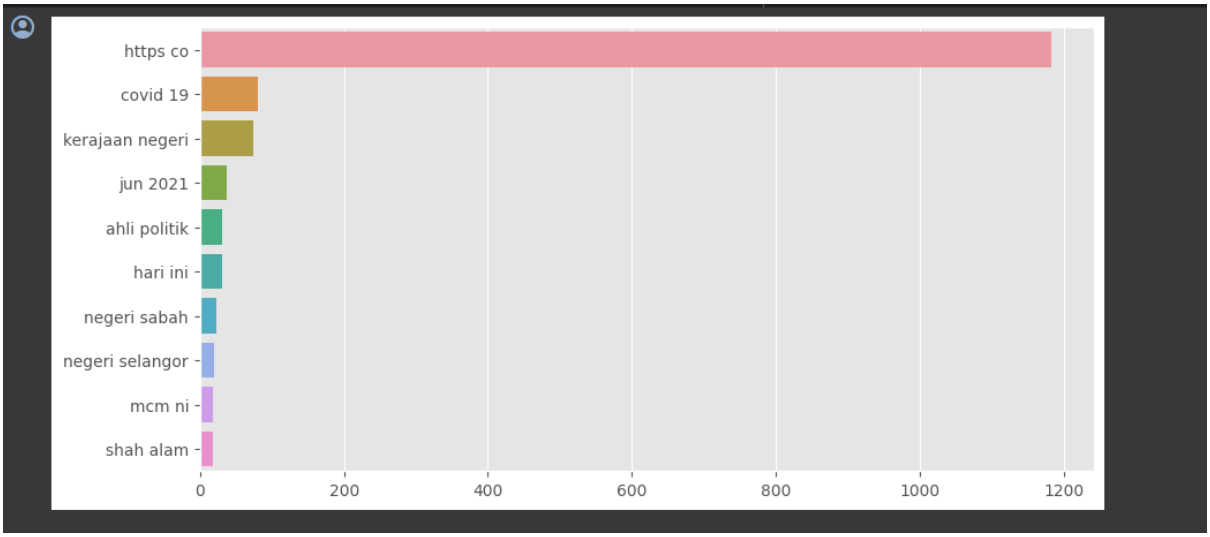
N-gram analysis

A technique used in NLP to analyze the sequential patterns of words in a text. An n-gram can be thought of as a sliding window of n words that moves across the text, extracting each group of words it encounters. For example, let's consider the sentence: "I love to code."

A 1-gram (unigram) analysis would involve looking at each individual word separately: "I", "love", "to", "code". A 2-gram (bigram) analysis would involve examining pairs of words: "I love", "love to", "to code". A 3-gram (trigram) analysis would involve considering groups of three consecutive words: "I love to", "love to code".

By analyzing these n-grams, we can gain insights into the frequency and patterns of word combinations within a text. For example, in the bigram "tak puas," the word "tak" changes the sentiment of "puas" to negative.

```
[ ]  def get_top_tweet_bigrams(corpus, n=None):
         vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
         bag_of_words = vec.transform(corpus)
         sum_words = bag_of_words.sum(axis=0)
         words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
         words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
         return words_freq[:n]

     plt.figure(figsize=(10, 5))
     top_tweet_bigrams = get_top_tweet_bigrams(df['text'])[:10]
     x, y = map(list, zip(*top_tweet_bigrams))
     sns.barplot(x=y, y=x)
     plt.show()
```



```
[ ]  # List of malay stop words
     STOP_WORDS = set("""
     ada adalah adanya adapun agak agaknya agar akan akankah akhir akhiri akhirnya
     aku akulah amat amatlah anda andalah antar antara antaranya apa apaan apabila
     apakah apalagi apatah artinya asal asalkan atas atau ataukah ataupun awal
     awalnya
     bagai bagaikan bagaimana bagaimanakah bagaimanapun bagi bagian bahkan bahwa
     bahwasanya baik bakal bakalan balik banyak byk bapak baru bawah beberapa begini
     beginian beginikah beginilah begitu begitukah begitulah begitupun bekerja
     belakang belakangan belum belumlah benar benarkah benarlah berada berakhir
     berakhirlah berakhirnya berapa berapakah berapalah berapapun berarti berawal
     berbagai berdatangan beri berikan berikut berikutnya berjumlah berkali-kali
     berkata berkehendak berkeinginan berkenaan berlainan berlalu berlangsung
     berlebihan bermacam bermacam-macam bermaksud bermula bersama bersama-sama
     bersiap bersiap-siap bertanya bertanya-tanya berturut berturut-turut bertutur
     berujar berupa besar betul betulkah biasa biasanya bila bilakah bisa bisakah
     boleh bolehkah bolehlah buat bukan bukankah bukanlah bukannya bulan bung
     cara caranya cukup cukupkah cukuplah cuma
     dahulu dalam dan dapat dari daripada datang dekat demi demikian demikianlah
     dengan dgn depan di dia diakhiri diakhirinya dialah diantara diantaranya diberi
     diberikan diberikannya dibuat dibuatnya didapat didatangkan digunakan
     diibaratkan diibaratkannya diingat diingatkan diinginkan dijawab dijelaskan
     dijelaskannya dikarenakan dikatakan dikatakannya dikerjakan diketahui
     diketahuinya dikira dilakukan dilalui dilihat dimaksud dimaksudkan
     dimaksudkannya dimaksudnya diminta dimintai dimisalkan dimulai dimulailah
     dimulainya dimungkinkan dini dipastikan diperbuat diperbuatnya dipergunakan
     diperkirakan diperlihatkan diperlukan diperlukannya dipersoalkan dipertanyakan
     dipunyai diri dirinya disampaikan disebut disebutkan disebutkannya disini
     disinilah ditambahkan ditandaskan ditanya ditanyai ditanyakan ditegaskan
     ditujukan ditunjuk ditunjuki ditunjukkan ditunjukkannya ditunjuknya dituturkan
     dituturkannya diucapkan diucapkannya diungkapkan dong dua dulu
     empat enggak enggaknya entah entahlah
     guna gunakan
     hal hampir hanya hanyalah hari harus haruslah harusnya hendak hendaklah
     hendaknya hingga
     ia ialah ibarat ibaratkan ibaratnya ibu ikut ingat ingat-ingat ingin inginkah
     inginkan ini inikah inilah itu itukah itulah
     jadi jadilah jadinya jangan jangankan janganlah jauh jawab jawaban jawabnya
     jelas jelaskan jelaslah jelasnya jika jikalau juga jumlah jumlahnya justru
     kala kalau kalaulah kalaupun kalian kami kamilah kamu kamulah kan kapan
```

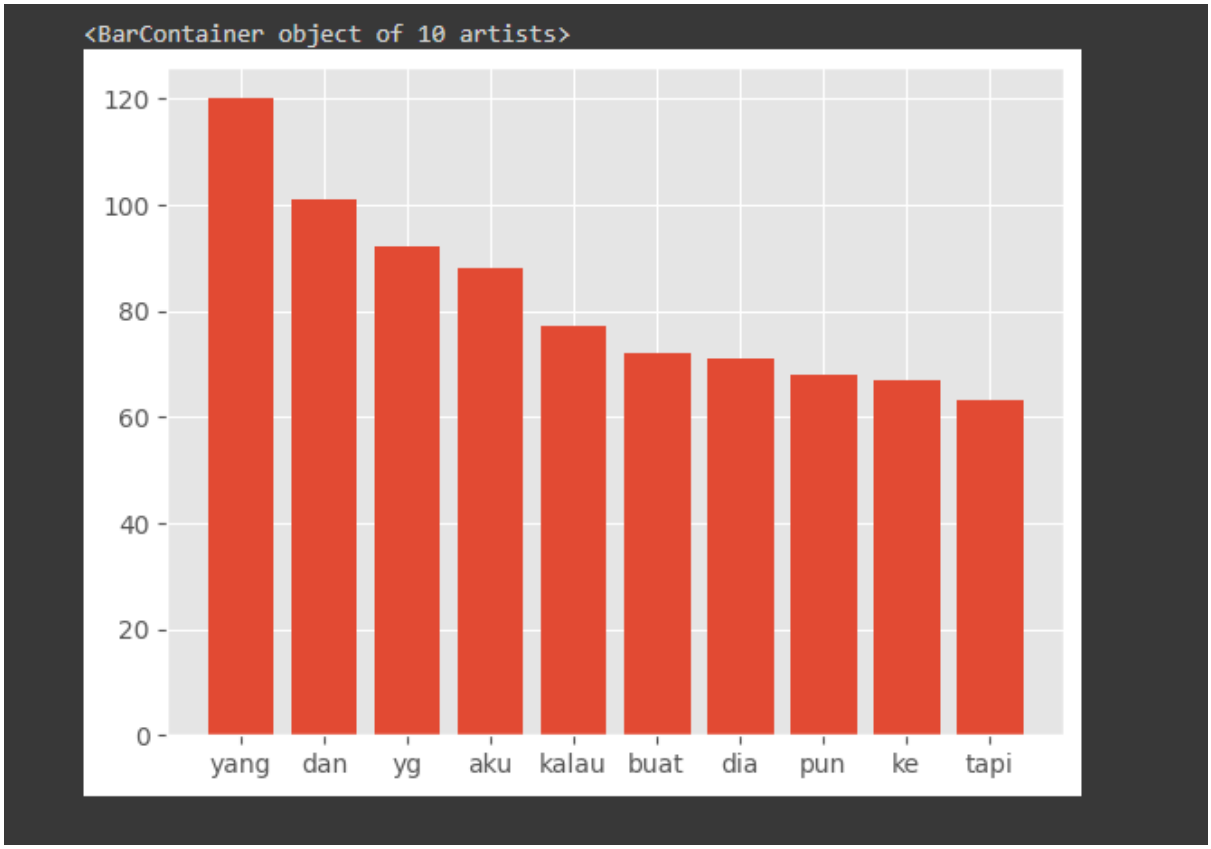
```
macam maka makanya makin malah malahan mampu mampukah mana manakala manalagi
masa masalah masalahnya masih masihkah masing masing-masing mau maupun
melainkan melakukan melalui melihat melihatnya memang memastikan memberi
memberikan membuat memerlukan memihak meminta memintakan memisalkan memperbuat
mempergunakan memperkirakan memperlihatkan mempersiapkan mempersoalkan
mempertanyakan mempunyai memulai memungkinkan menaiki menambahkan menandaskan
menanti menanti-nanti menantikan menanya menanyai menanyakan mendapat
mendapatkan mendatang mendatangi mendatangkan menegaskan mengakhiri mengapa
mengatakan mengatakannya mengenai mengerjakan mengetahui menggunakan
menghendaki mengibaratkan mengibaratkannya mengingat mengingatkan menginginkan
mengira mengucapkan mengucapkannya mengungkapkan menjadi menjawab menjelaskan
menuju menunjuk menunjuki menunjukkan menunjuknya menurut menuturkan
menyampaikan menyangkut menyatakan menyebutkan menyeluruh menyiapkan merasa
mereka merekalah merupakan meski meskipun meyakini meyakinkan minta mirip
misal misalkan misalnya mula mulai mulailah mulanya mungkin mungkinkah
nah naik namun nanti nantinya nyaris nyatanya
oleh olehnya
pada padahal padanya pak paling panjang pantas para pasti pastilah penting
pentingnya per percuma perlu perlukah perlunya pernah persoalan pertama
pertama-tama pertanyaan pertanyakan pihak pihaknya pukul pula pun punya
rasa rasanya rata rupanya
saat saatnya saja sajalah saling sama sama-sama sambil sampai sampai-sampai
sampaikan sana sangat sangatlah satu saya sayalah se sebab sebabnya sebagai
sebagaimana sebagainya sebagian sebaik sebaik-baiknya sebaiknya sebaliknya
sebanyak sebegini sebegitu sebelum sebelumnya sebenarnya seberapa sebesar
sebetulnya sebisanya sebuah sebut sebutlah sebutnya secara secukupnya sedang
sedangkan sedemikian sedikit sedikitnya seenaknya segala segalanya segera
seharusnya sehingga seingat sejak sejauh sejenak sejumlah sekadar sekadarnya
sekali sekali-kali sekalian sekaligus sekalipun sekarang sekarang sekecil
seketika sekiranya sekitar sekitarnya sekurang-kurangnya sekurangnya sela
selain selaku selalu selama selama-lamanya selamanya selanjutnya seluruh
seluruhnya semacam semakin semampu semampunya semasa semasih semata semata-mata
semaunya sementara semisal semisalnya sempat semua semuanya semula sendiri
sendirian sendirinya seolah seolah-olah seorang sepanjang sepantasnya
sepantasnyalah seperlunya seperti sepertinya sepihak sering seringnya serta
serupa sesaat sesama sesampai sesegera sesekali seseorang sesuatu sesuatunya
sesudah sesudahnya setelah setempat setengah seterusnya setiap setiba setibanya
setidak-tidaknya setidaknya setinggi seusai sewaktu siap siapa siapakah
siapapun sini sinilah soal soalnya suatu sudah sudahkah sudahlah supaya
tadi tadinya tahu tahun tambah tambahnya tampak tampaknya tandas tandasnya
tanpa tanya tanyakan tanyanya tapi tegas tegasnya telah tempat tengah tentang
tentu tentulah tentunya tepat terakhir terasa terbanyak terdahulu terdapat
```

```
sekali sekali-kali sekalian sekaligus sekalipun sekarang sekarang sekecil
seketika sekiranya sekitar sekitarnya sekurang-kurangnya sekurangnya sela
selain selaku selalu selama selama-lamanya selamanya selanjutnya seluruh
seluruhnya semacam semakin semampu semampunya semasa semasih semata semata-mata
semaunya sementara semisal semisalnya sempat semua semuanya semula sendiri
sendirian sendirinya seolah seolah-olah seorang sepanjang sepantasnya
sepantasnyalah seperlunya seperti sepertinya sepihak sering seringnya serta
serupa sesaat sesama sesampai sesegera sesekali seseorang sesuatu sesuatunya
sesudah sesudahnya setelah setempat setengah seterusnya setiap setiba setibanya
setidak-tidaknya setidaknya setinggi seusai sewaktu siap siapa siapakah
siapapun sini sinilah soal soalnya suatu sudah sudahkah sudahlah supaya
tadi tadinya tahu tahun tambah tambahnya tampak tampaknya tandas tandasnya
tanpa tanya tanyakan tanyanya tapi tegas tegasnya telah tempat tengah tentang
tentu tentulah tentunya tepat terakhir terasa terbanyak terdahulu terdapat
terdiri terhadap terhadapnya teringat teringat-ingat terjadi terjadilah
terjadinya terkira terlalu terlebih terlihat termasuk ternyata tersampaikan
tersebut tersebutlah tertentu tertuju terus terutama tetap tetapi tiap tiba
tiba-tiba tidakkah tidaklah tiga tinggi toh tunjuk turut tutur tuturnya
ucap ucapnya ujar ujarnya umum umumnya ungkap ungkapnya untuk usah usai
waduh wah wahai waktu waktunya walau walaupun wong
yaitu yakin yakni yang yg
""".split()
)
# Credit to a malay contributor to the spaCy library: https://github.com/explosion/spaCy/pull/12602/files

# Although, 'tidak','tak', 'takde', 'takda', 'tk', 'x', are stopwords,
# we don't remove them from the training data because it has important meaning as it can flip the sentiment.

# For eg, 'not good' is a negative sentiment, but if we remove 'not' as it's technically a stopword,
# we will get 'good' which is a positive sentiment.
```

```
[ ]  # Create corpus list for all words in the text

     def create_corpus(sentiment):
         corpus=[]

         for x in df[df['sentiment']==sentiment]['text'].str.split():
             for i in x:
                 corpus.append(i.lower())
         return corpus
```

Analyze the most common stopwords in Negative tweets.

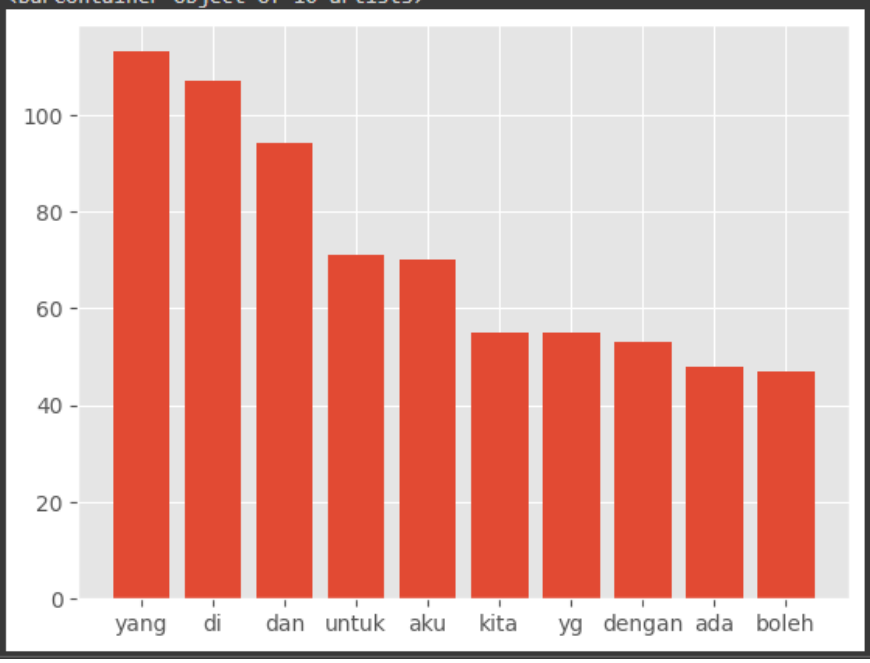
```
[ ]  negative_corpus=create_corpus('Negative')

     dic=defaultdict(int)
     for word in negative_corpus:
         if word in STOP_WORDS:
             dic[word]+=1

     top=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]

     x,y=zip(*top)
     plt.bar(x,y)
```


<BarContainer object of 10 artists>
120
100
80
60
40
20
0
yang
dan
yg
aku
kalau
buat
dia
pun
ke
tapi

Analyze the most common stopwords in Positive tweets.

```
[ ]  positive_corpus=create_corpus("Positive")

     dic=defaultdict(int)
     for word in positive_corpus:
         if word in STOP_WORDS:
             dic[word]+=1

     top=sorted(dic.items(), key=lambda x:x[1],reverse=True)[:10]

     x,y=zip(*top)
     plt.bar(x,y)
```

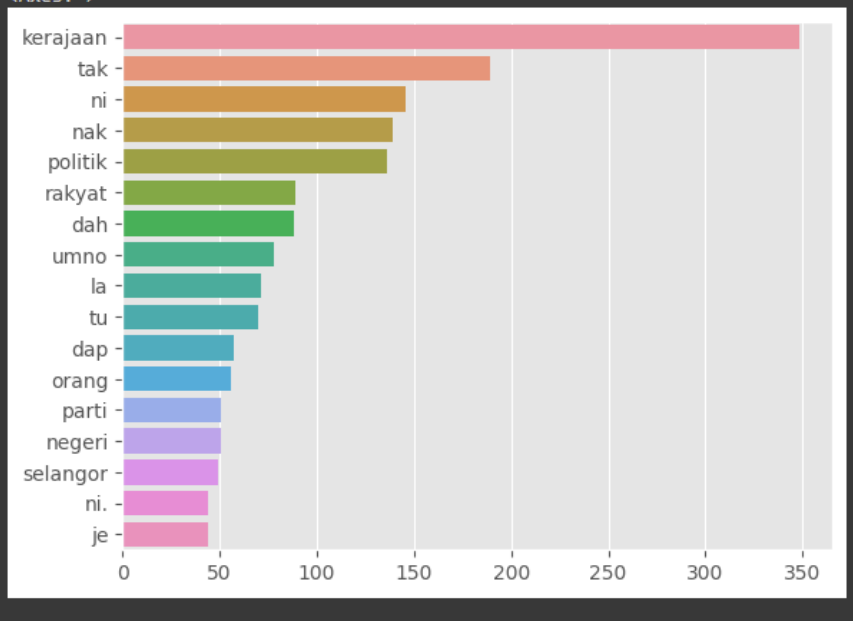
```
<BarContainer object of 10 artists>
```



Analyze the most common words (excl stopwords) in Negative tweet

```
[ ]  counter = Counter(negative_corpus)
     most = counter.most_common()
     x = []
     y = []
     for word, count in most[:40]:
         if word not in STOP_WORDS:
             x.append(word)
             y.append(count)

     sns.barplot(x=y,y=x)
```

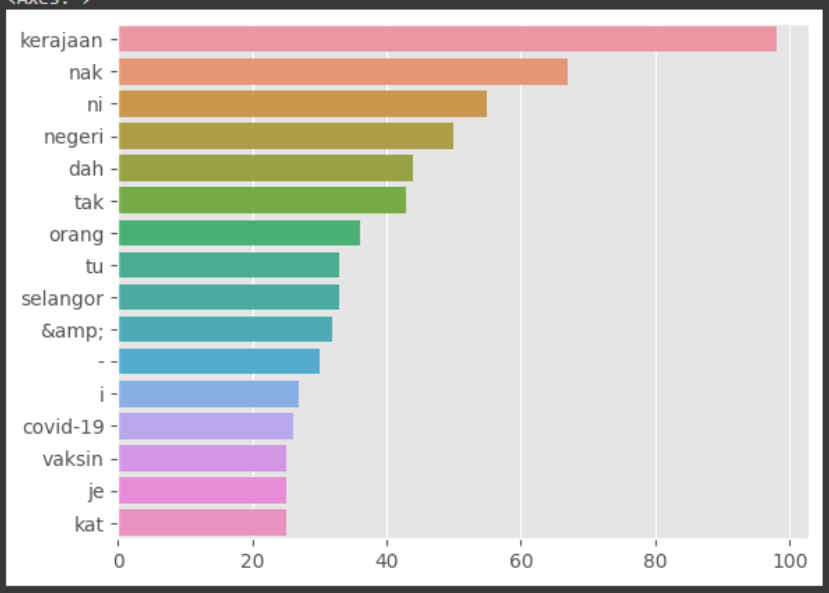
```
<Axes: >
```



Analyze the most common words (excl stopwords) in Positive tweets.

```
[ ]  counter = Counter(positive_corpus)
     most = counter.most_common()
     x = []
     y = []
     for word, count in most[:40]:
         if word not in STOP_WORDS:
             x.append(word)
             y.append(count)

     sns.barplot(x=y,y=x)
```

```
<Axes: >
```



Data Preprocessing:

Data Cleaning

```
[ ]  df_dirty = df.copy() # save a copy of df before cleaning
```

Remove stopwords

```
[ ]  print('Before: ', df.iloc[11]['text'])

     def removeStopWords(text):
         words = text.split()
         filtered_words = [word for word in words if word.lower() not in STOP_WORDS]
         return ' '.join(filtered_words)

     df['text'] = df['text'].apply(removeStopWords)
     print('After: ', df.iloc[11]['text'])
```

```
Before:  Bekas MB Perak, Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN yang baru. #GetaranMY | #TERKINI
After:  Bekas MB Perak, Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN baru. #GetaranMY | #TERKINI
```

Remove URL

```
[ ]  print('Before:', df.iloc[5]['text'])

     def cleanLink(text):
         text = re.sub(r'https?:\/\/\S+', '', text)  # remove any type of link
         return text

     df['text'] = df['text'].apply(cleanLink)
     print('After:', df.iloc[5]['text'])
```

```
Before: Lastly, pi jumpa doktor kerajaan, ckp doktor kau tak nak ubat. Tolong jgn buang duit negara bayar subsidi https://t.co/2jIzHiXXSA
After: Lastly, pi jumpa doktor kerajaan, ckp doktor kau tak nak ubat. Tolong jgn buang duit negara bayar subsidi
```

Remove hashtag symbol

```
[ ]  print('Before:', df.iloc[11]['text'])

     def cleanHashtag(text):
       text = re.sub(r'#', '', text) # remove any #hashtag
       return text

     # After clean hashtag
     df['text'] = df['text'].apply(cleanHashtag)
     print('After:', df.iloc[11]['text'])
```

```
Before: Bekas MB Perak, Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN baru. #GetaranMY | #TERKINI
After: Bekas MB Perak, Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN baru. GetaranMY | TERKINI
```

Remove user @mention

```
[ ]  print('Before:', df.iloc[20]['text'])

     def cleanMention(text):
         text = re.sub(r'@[A-Za-z0-9]+', '', text)  # remove any @mention
         return text

     # After clean @ mention
     df['text'] = df['text'].apply(cleanMention)
     print('After:', df.iloc[20]['text'])
```

```
Before: @limlipeng durian Raub pengundi, sokong DAP. Ahaks lkshinaislam durianraub
After:  durian Raub pengundi, sokong DAP. Ahaks lkshinaislam durianraub
```

Remove punctuations

```
[ ]  print('Before:', df.iloc[11]['text'])

     def cleanPunctuations(text):
         text = re.sub('[' + string.punctuation + ']', '', text)  # Remove punctuations
         return text

     # After clean punctuation
     df['text'] = df['text'].apply(cleanPunctuations)
     print('After:', df.iloc[11]['text'])
```

```
Before: Bekas MB Perak, Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN baru. GetaranMY | TERKINI
After: Bekas MB Perak Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN baru GetaranMY  TERKINI
```

Remove 'RT' (Retweet) word as it's meaningless

```
[ ]  print('Before:', df.iloc[14]['text'])

     def cleanRT(text):
         text = re.sub(r'RT[\s]+', '', text) # remove any 'RT' word (stands for Retweet)
         return text

     df['text'] = df['text'].apply(cleanRT)
     print('After:',df.iloc[14]['text'])
```

```
Before: RT   Tumpang promote Hi korang jom support bisnes partner i If korang ph
After: Tumpang promote Hi korang jom support bisnes partner i If korang ph
```

Remove emoji

```
# Before clean emoji
emoji_text = 'Hello! 🌈 😊😊😊 ❌🔧'
print('Before:', emoji_text)

def cleanEmoji(text):
  # emoticons (facial expression like smiley) are not removed since we will replace it with text representation, only other noisy emojis are removed
  emoji_pattern = re.compile("["
                             u"\U0001F300-\U0001F5FF"  # symbols & pictographs i.e. 🌟 ⚡ 🌈 💡 🔧
                             u"\U0001F680-\U0001F6FF"  # transport & map symbols
                             u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                             u"\U00002702-\U000027B0"  # additional symbols and shapes i.e. ✂ ❌ ✓
                             u"\U000024C2-\U0001F251"  # miscellaneous symbols i.e. ♓ 🔀 🔁
                             "]+", flags=re.UNICODE)
  text = emoji_pattern.sub(r'', text)

  # Preserves the sentiment conveyed by the emoticons by converting them into textual format like :grinning_face:
  text = emoji.demojize(text)

  # Remove colons(:) in :grinning_face: but do not remove the underscore so that vectorization will consider them as 1 word
  # Add a space around each colon to separate consecutive textual emojis 😊😊😊 and then remove extra spaces
  text = re.sub(r'(:\w+:| +)', lambda match: ' ' if match.group(0).isspace() else f' {match.group(0)} ', text.strip())

  return text

# Example after clean emoji
print('After:', cleanEmoji(emoji_text))

df['text'] = df['text'].apply(cleanEmoji)
```

```
Before: Hello! 🌈 😊😊😊 ❌🔧
After: Hello!  :smiling_face_with_smiling_eyes:  :smiling_face_with_smiling_eyes:  :smiling_face_with_smiling_eyes:
```

Remove specific words

These words occur frequently across different sentiment classes and do not provide discriminative information.

```
print('Before:', df.iloc[11]['text'])

def removeSpecificWords(text):
    num_words = [
        "satu", "dua", "tiga", "empat", "lima", "enam", "tujuh", "lapan", "sembilan",
        "sepuluh", "sebelas", "belas", "puluh", "ratus", "ribu", "juta", "million", "billion",
        "trillion", "kuadrilion", "kuintilion", "sekstilion", "septilion", "oktilion", "nonilion",
        "desilion"
    ]

    currencies = ["usd", "rm", "myr", "rp", "idr", "rmb", "sgd"]

    months = [
        "januari", "februari", "mac", "april", "mei", "jun", "julai", "ogos", "september",
        "oktober", "november", "disember", "january", "february", "march", "may", "june",
        "july", "august", "october", "december", "jan", "feb", "mac", "jun", "sept", "okt", "nov", "dis"
    ]

    entities = [
    "kerajaan", "parti", "umno", "bn", "ph", "pkr", "pas", "pn", "dap", "malaysia", "negara", "persekutuan",
    "wilayah", "daerah", "dun", "parlimen", "negeri", "selangor", "pahang", "perak",
    "terengganu", "perlis", "johor", "kelantan", "kedah", "pulau", "penang", "melaka", "sabah", "sarawak"
    ]


    # Split the text into individual words
    words = text.lower().split()
```

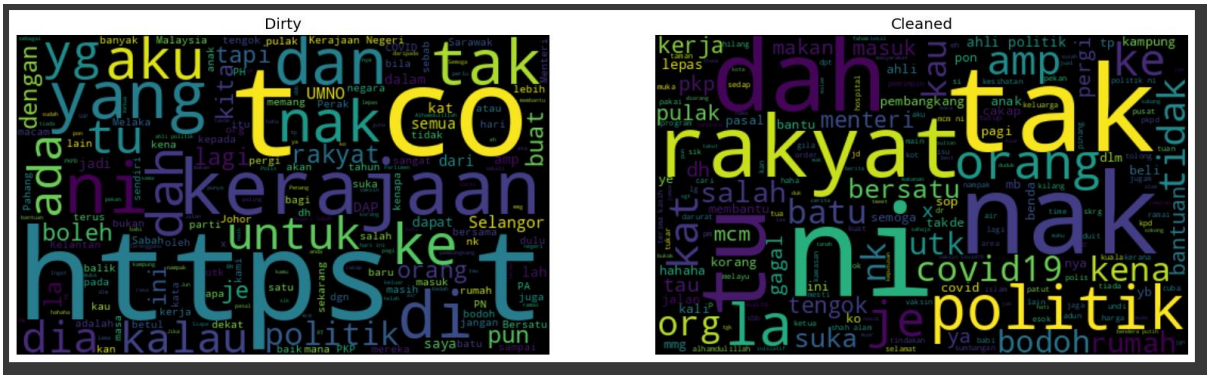
```
        # Remove the specific words
        cleaned_words = [word for word in words if word not in num_words
                         and word not in currencies
                         and word not in months
                         and word not in entities]

        # Join the cleaned words back into a single string
        cleaned_text = ' '.join(cleaned_words)

        return cleaned_text

    # Apply the function to the 'text' column
    df['text'] = df['text'].apply(removeSpecificWords)
    print('After:', df.iloc[11]['text'])
```

```
Before: Bekas MB Perak Datuk Seri Zambry Abdul Kadir dilantik Setiausaha Agung BN baru GetaranMY TERKINI
After: bekas mb datuk seri zambry abdul kadir dilantik setiausaha agung baru getaranmy terkini
```

```
[ ]  def preprocessing(text):
        text = removeStopWords(text)
        text = cleanLink(text)
        text = cleanHashtag(text)
        text = cleanPunctuations(text)
        text = cleanRT(text)
        text = cleanEmoji(text)
        text = removeSpecificWords(text)
        return text
```

Word Cloud: Before vs After data cleaning

```
[ ]  allwords_dirty = ' '.join([twts for twts in df_dirty["text"]])
     allwords_cleaned = ' '.join([twts for twts in df["text"]])

     wordCloud_dirty = WordCloud(width=500, height=300, random_state=21, max_font_size=119).generate(allwords_dirty)
     wordCloud_cleaned = WordCloud(width=500, height=300, random_state=21, max_font_size=119).generate(allwords_cleaned)

     fig, ax = plt.subplots(1, 2, figsize=(20, 10))  # 1 row, 2 columns

     # Display the word cloud of pre-cleaned tweets:
     ax[0].imshow(wordCloud_dirty, interpolation='bilinear')
     ax[0].set_title('Dirty')
     ax[0].axis('off')

     # Display the word cloud of cleaned tweets:
     ax[1].imshow(wordCloud_cleaned, interpolation='bilinear')
     ax[1].set_title('Cleaned')
     ax[1].axis('off')

     plt.show()
```



Calculate the number of unique words

```
[ ] unique_words = set(" ".join(df['text']).split())
    num_unique_words = len(unique_words)
    print("Number of unique words:", num_unique_words)
```

```
Number of unique words: 8384
```

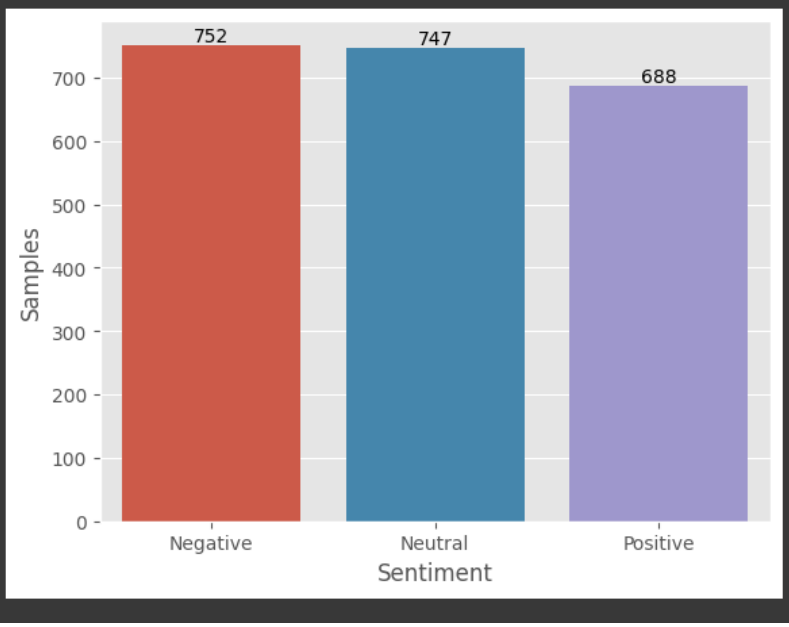
Delete column text with empty string ' ' after we've removed various words in data cleaning

```
[ ] # Find indices of empty string rows
    empty_string_indices = df[df['text'].str.strip().eq('')].index
    print('Before:',len(empty_string_indices))

    # Delete rows with empty strings
    df = df.drop(empty_string_indices)

    # Verify that empty string rows are removed
    empty_strings = df['text'].str.strip().eq('').sum()
    print('After:',empty_strings)
```

```
Before: 9
After: 0
```

Check sentiment class distribution after rows deleted

```
[ ] # Count the occurrences of each class
    class_counts = df['sentiment'].value_counts()

    # Create a bar plot
    ax = sns.barplot(x=class_counts.index, y=class_counts)
    plt.ylabel('Samples')
    plt.xlabel('Sentiment')

    # Add count annotations above each bar
    for i, count in enumerate(class_counts):
        ax.annotate(str(count), xy=(i, count), ha='center', va='bottom')

    plt.show()
```



```
[ ] # Convert sentiment values (Negative, Positive, Neutral) string into integer (0,1,2) for faster processing
    unique_sentiments = df['sentiment'].unique()
    print(unique_sentiments)

    sentiment_mapping = {'Negative': 0, 'Positive': 1, 'Neutral': 2, 'negative': 0, 'positive': 1, 'neutral': 2}

    # Replace the values in the 'sentiment' column using the mapping
    df['sentiment'] = df['sentiment'].map(sentiment_mapping)
    print(df['sentiment'].dtype)
    df.head(3)
```

```
['Neutral' 'Negative' 'Positive']
int64
```

	text	id	sentiment	annotator	annotation_id	created_at	updated_at	lead_time
0	bersatu teguh bercerai roboh	81037	2	27	4231	2022-02-14T15:58:43.745680Z	2022-02-14T15:58:47.080664Z	282799.830
1	bersatu bersatu indonesia kuat	81001	2	27	4195	2022-02-14T15:56:20.283992Z	2022-02-14T15:56:23.113936Z	282655.843
2	jenis tak suka sgt tgk mpmp showoff diorang ke...	80998	0	27	4192	2022-02-14T15:55:57.558702Z	2022-02-14T15:56:03.446298Z	282636.179

Drop neutral sentiment

The goal is to only differentiate positive and negative tweets, which simplifies the machine learning process. Moreover, in politics and product reviews, neutral opinions are not significant for analysis.

```
[ ] # Remove neutral sentiment since this is not relevant to our machine learning use case
    df = df[df['sentiment'] != 2]
```

Machine Learning

```
[ ] # Create a positive & negative corpus of unique words so we can analyze the most common words in them for feature engineering
    def create_corpus(df, sentiment):
        corpus = []

        for x in df[df['sentiment'] == sentiment]['text'].str.split():
            for i in x:
                corpus.append(i.lower())

        word_freq = Counter(corpus)
        most_common_words = word_freq.most_common()  # This will return a list of tuples, where each tuple is a word and its frequency, sorted by frequency

        return most_common_words  # Returning the sorted list of tuples

    # Just to run and check it once. We will hide the printed unique words since the output is too long

    # sorted_negative_corpus = create_corpus(df, 0)
    # print(sorted_negative_corpus)

    # sorted_positive_corpus = create_corpus(df, 1)
    # print(sorted_positive_corpus)
```

```
BAD_WORDS = ['babi', 'bodoh', 'bodoe', 'anjay', 'anjir', 'njir', 'tenok', 'cipan', 'belen', 'murahan', 'seranah', 'ludah', 'tercyduk', 'sibuk', 'umngok', 'dapig', 'santau', 'mabok', 'sawan', 'butoh', 'anak haram', 'duit haram', 'har

print('Here we make our own list of', len(BAD_WORDS), 'words that tend to be used in negative tweets')

GOOD_WORDS = ['alhamdulilah', 'alhamdulillah', 'masyaallah', 'tabarakallah', 'allahuakhbar', 'allohumma', 'aman', 'ameen', 'amiinn', 'aamiin', 'aminn', 'nikmat', 'subhanallah', 'bantuan', 'best', 'beautiful', 'cantik', 'doa', 'hehe'

print('Here we make our own list of', len(GOOD_WORDS), 'words that tend to be used in positive tweets')

print('Total: ', len(BAD_WORDS) + len(GOOD_WORDS))
```

```
Here we make our own list of 677 words that tend to be used in negative tweets
Here we make our own list of 192 words that tend to be used in positive tweets
Total:  869
```

```
# Create binary feature for presence of bad words
df['swear_present'] = df['text'].apply(lambda x: any(word in x for word in BAD_WORDS))

# Create binary feature for presence of good words
df['good_present'] = df['text'].apply(lambda x: any(word in x for word in GOOD_WORDS))

# Create feature for proportion of bad words
df['swear_proportion'] = df['text'].apply(lambda x: sum(x.count(word) for word in BAD_WORDS) / len(x.split()) if x and len(x.split()) > 0 else 0)

# Create feature for proportion of good words
df['good_proportion'] = df['text'].apply(lambda x: sum(x.count(word) for word in GOOD_WORDS) / len(x.split()) if x and len(x.split()) > 0 else 0)

# df['good_proportion'] = df['good_proportion'].apply(lambda x: x * 3)  # Increase the weight by multiplying by a factor
```

```
<ipython-input-114-42bfaf72fc88>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['swear_present'] = df['text'].apply(lambda x: any(word in x for word in BAD_WORDS))
```

Machine Learning Algorithm:

```
[ ]  # Split the dataset into training and test data
     X = df[['text', 'swear_present', 'swear_proportion', 'good_present', 'good_proportion']]
     y = df['sentiment']
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=70)
```

```
[ ]  # Create an instance of TfidfVectorizer, which converts a collection of raw documents into a matrix of TF-IDF features.
     # The vectorizer is configured to remove English stop words (for tweets mixed with malay & english words) since we have already removed Malay stop words
     # and consider both individual words and groups of 2 to 3 words (n-grams).
     vectorizer = TfidfVectorizer(stop_words='english', ngram_range=(1,3))
```

```
[ ]  # Fit and transform the training data
     X_train_tfidf = vectorizer.fit_transform(X_train['text'])
     X_train_swears = X_train[['swear_present', 'swear_proportion', 'good_present', 'good_proportion']].values
     X_train_final = pd.concat([pd.DataFrame(X_train_tfidf.toarray()), pd.DataFrame(X_train_swears)], axis=1)

     # Transform the test data
     X_test_tfidf = vectorizer.transform(X_test['text'])
     X_test_swears = X_test[['swear_present', 'swear_proportion', 'good_present', 'good_proportion']].values
     X_test_final = pd.concat([pd.DataFrame(X_test_tfidf.toarray()), pd.DataFrame(X_test_swears)], axis=1)

     # Set seed for reproducibility
     np.random.seed(42)
```

Naive Bayes Classifier

```
[ ]  # Initialize the Naive Bayes classifier
     nb = MultinomialNB()

     # Train the model
     nb.fit(X_train_final, y_train)

     # Test the model
     y_pred_nb = nb.predict(X_test_final)
```

Model evaluation of Naive Bayes

```
[ ]  accuracy_nb= accuracy_score(y_test, y_pred_nb)
     print(f" Model Accuracy for Naive Bayes Classifier: {accuracy_nb:2f}")
```

```
 Model Accuracy for Naive Bayes Classifier: 0.833333
```

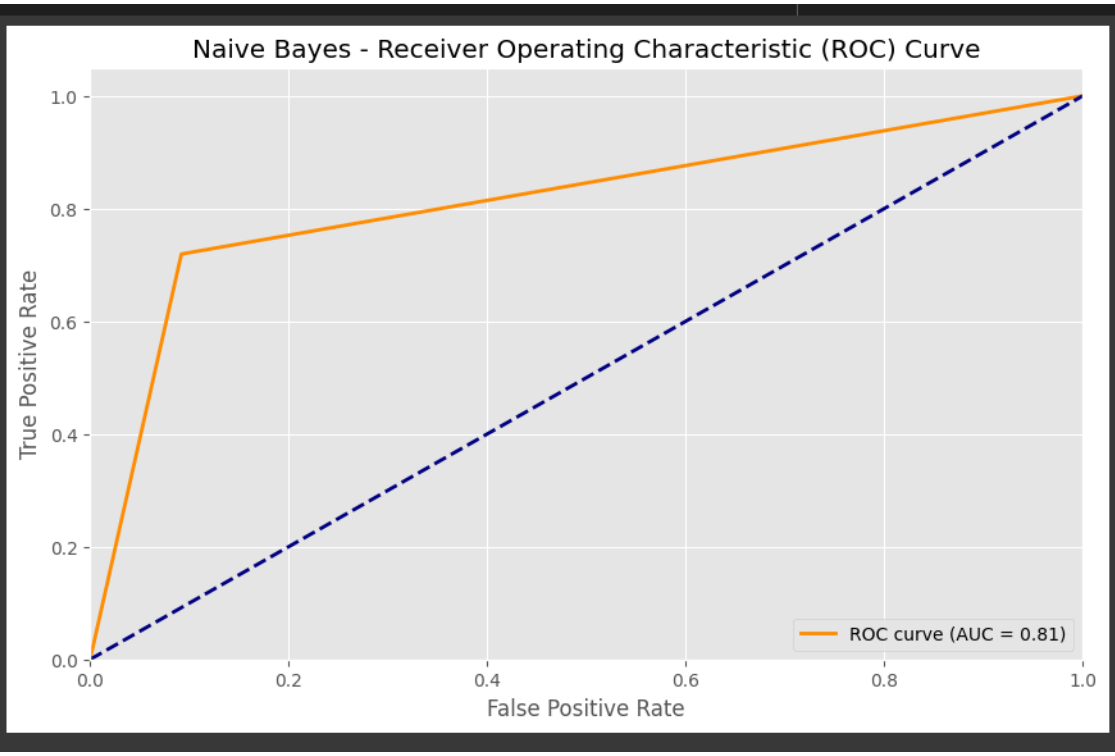
```
[ ]  # Print the classification report
     print(classification_report(y_test, y_pred_nb))
```

```
                precision    recall  f1-score   support

             0       0.83      0.91      0.87        87
             1       0.84      0.72      0.77        57

      accuracy                           0.83       144
     macro avg       0.83      0.81      0.82       144
  weighted avg       0.83      0.83      0.83       144
```

```
[ ]  # Calculate and print ROC AUC score
     roc_auc = metrics.roc_auc_score(y_test, y_pred_nb)
     print(f'ROC AUC score: {roc_auc:.2f}')
```

```
     ROC AUC score: 0.81
```

```
[ ]  fpr, tpr, thresholds = roc_curve(y_test, y_pred_nb)

     # Calculate the area under the ROC curve
     roc_auc = auc(fpr, tpr)

     # Plot the ROC curve
     plt.figure(figsize=(10, 6))
     plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Naive Bayes - Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc='lower right')
     plt.show()
```




```
[ ]  # MSE and RMSE for Naive Bayes
     mse_nb = mean_squared_error(y_test, y_pred_nb)
     rmse_nb = np.sqrt(mse_nb)
     print(f"\nMean Squared Error (MSE) for Naive Bayes: {mse_nb:.2f}")
     print(f"Root Mean Squared Error (RMSE) for Naive Bayes: {rmse_nb:.2f}")
```

```
     Mean Squared Error (MSE) for Naive Bayes: 0.17
     Root Mean Squared Error (RMSE) for Naive Bayes: 0.41
```

LSTM (Long Short-Term Memory) - Deep Learning RNN

```
[ ]  # Set the timesteps and input_dim variables
     timesteps = 4
     input_dim = X_train_final.shape[1]

     # Create a new variable to store the reshaped data
     X_train_final_reshaped = X_train_final.to_numpy().astype('float32').reshape((-1, timesteps, input_dim))
     X_test_final_reshaped = X_test_final.to_numpy().astype('float32').reshape((-1, timesteps, input_dim))
     y_train_reshaped = y_train.to_numpy().reshape((-1, timesteps, 1))
     y_test_reshaped = y_test.to_numpy().reshape((-1, timesteps, 1))

     # Create an input layer for the LSTM network
     input_layer = Input(shape=(timesteps, input_dim), name='input_layer')

     # Add the LSTM layer with 64 units, dropout, and recurrent_dropout
     lstm = LSTM(64, dropout=0.2, recurrent_dropout=0.2, return_sequences=True)(input_layer)

     # Add a dense layer to the LSTM output
     dense = Dense(32, activation='relu')(lstm)

     # Add a dense output layer with a sigmoid activation function
     output = Dense(1, activation='sigmoid')(dense)

     # Create the model
     model = Model(inputs=input_layer, outputs=output)

     # Compile the model with binary cross-entropy loss and Adam optimizer
     model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

     # Train the model
     model.fit(tf.convert_to_tensor(X_train_final_reshaped), y_train_reshaped, epochs=10, batch_size=64)

     # Test the model
     y_pred_lstm = model.predict(X_test_final_reshaped)
```

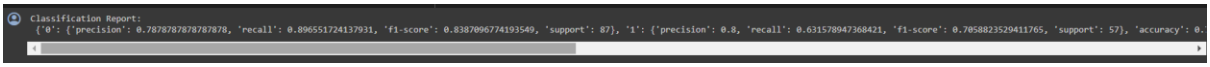
```
Epoch 1/10
6/6 [==============================] - 6s 471ms/step - loss: 0.6921 - accuracy: 0.5139
Epoch 2/10
6/6 [==============================] - 3s 496ms/step - loss: 0.6791 - accuracy: 0.6767
Epoch 3/10
6/6 [==============================] - 4s 679ms/step - loss: 0.6604 - accuracy: 0.7886
Epoch 4/10
6/6 [==============================] - 3s 482ms/step - loss: 0.6343 - accuracy: 0.8279
Epoch 5/10
6/6 [==============================] - 3s 480ms/step - loss: 0.5990 - accuracy: 0.8418
Epoch 6/10
6/6 [==============================] - 3s 477ms/step - loss: 0.5542 - accuracy: 0.8727
Epoch 7/10
6/6 [==============================] - 4s 740ms/step - loss: 0.5031 - accuracy: 0.8804
Epoch 8/10
6/6 [==============================] - 3s 484ms/step - loss: 0.4417 - accuracy: 0.9020
Epoch 9/10
6/6 [==============================] - 3s 473ms/step - loss: 0.3744 - accuracy: 0.9267
Epoch 10/10
6/6 [==============================] - 3s 458ms/step - loss: 0.3080 - accuracy: 0.9475
2/2 [==============================] - 0s 63ms/step
```

Model Evaluation of LSTM

```
[ ]  # Flatten the predicted and actual values to make them compatible with evaluation functi
     y_pred_flat = y_pred_lstm.flatten()
     y_test_flat = y_test_reshaped.flatten()

     # Convert predicted probabilities to binary predictions using a threshold (0.5 in this c
     y_pred_binary = (y_pred_flat > 0.5).astype(int)

     # Calculate accuracy
     accuracy_lstm = accuracy_score(y_test_flat, y_pred_binary)
     print(f'Model Accuracy for LSTM: {accuracy_lstm:.4f}')
```

```
Model Accuracy for LSTM: 0.7917
```

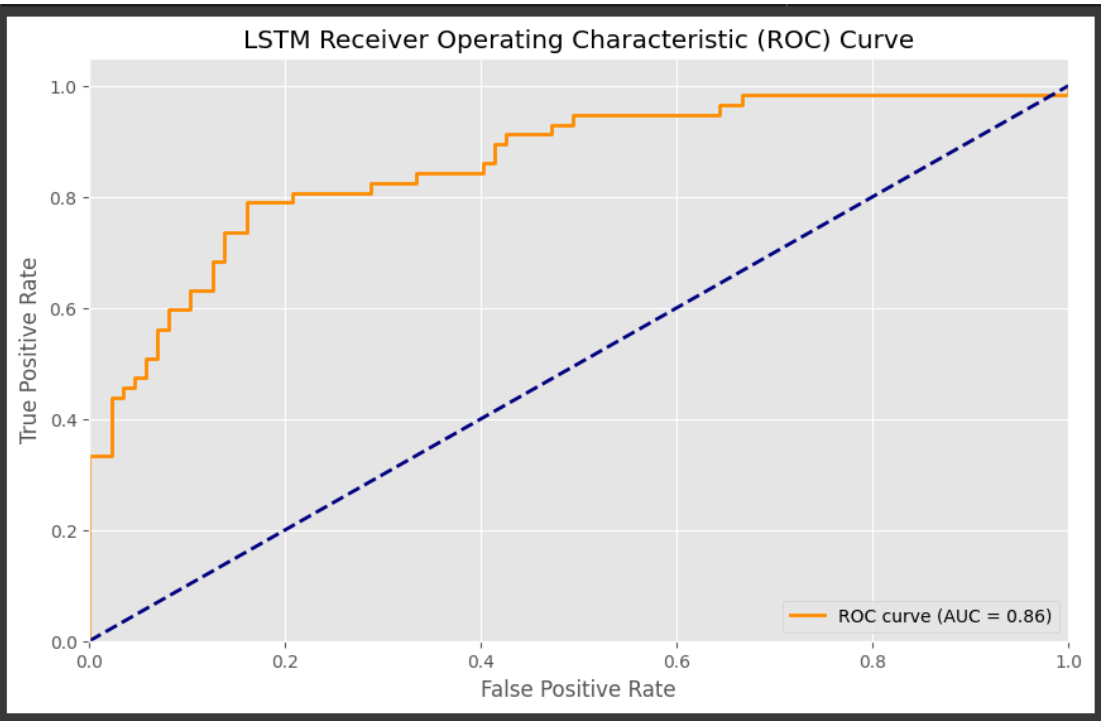
```
[ ]  # Print the classification report
     report_lstm = classification_report(y_test_flat, y_pred_binary, output_dict=True)
     print('Classification Report:\n', report_lstm)
```




```
[ ]  # Calculate and print ROC AUC score
     roc_auc_lstm = metrics.roc_auc_score(y_test_flat, y_pred_flat)
     print(f'ROC-AUC Score: {roc_auc_lstm:.4f}')
```

```
ROC-AUC Score: 0.8607
```

```
[ ]  # Calculate the ROC curve
     fpr_lstm, tpr_lstm, thresholds = roc_curve(y_test_flat, y_pred_flat)

     # Calculate the area under the ROC curve
     roc_auc = auc(fpr_lstm, tpr_lstm)

     # Plot the ROC curve
     plt.figure(figsize=(10, 6))
     plt.plot(fpr_lstm, tpr_lstm, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('LSTM Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc='lower right')
     plt.show()
```



The curve is not smooth because we need to Flatten the y_test and y_pred since LSTM input requires a 3D array

```
[ ]  # Calculate Mean Squared Error (MSE)
     mse_lstm = mean_squared_error(y_test_flat, y_pred_flat)

     # Calculate Root Mean Squared Error (RMSE)
     rmse_lstm = np.sqrt(mse_lstm)

     # Print the results
     print(f"\nMean Squared Error (MSE) for LSTM: {mse_lstm:.4f}")
     print(f"Root Mean Squared Error (RMSE) for LSTM: {rmse_lstm:.4f}")
```

```
Mean Squared Error (MSE) for LSTM: 0.1600
Root Mean Squared Error (RMSE) for LSTM: 0.4000
```

Random Forest Classifier

```
[ ]  # Initialize the Random Forest classifier
     rf_classifier = RandomForestClassifier(random_state=42)

     # Train the model
     rf_classifier.fit(X_train_final, y_train)

     # Test the model
     y_pred_rf = rf_classifier.predict(X_test_final)
```

Model Evaluation of Random Forest

```
[ ] accuracy_rf= accuracy_score(y_test, y_pred_rf)
    print(f" Model Accuracy for Random Forest Classifier: {accuracy_rf:4f}")
```

```
 Model Accuracy for Random Forest Classifier: 0.756944
```

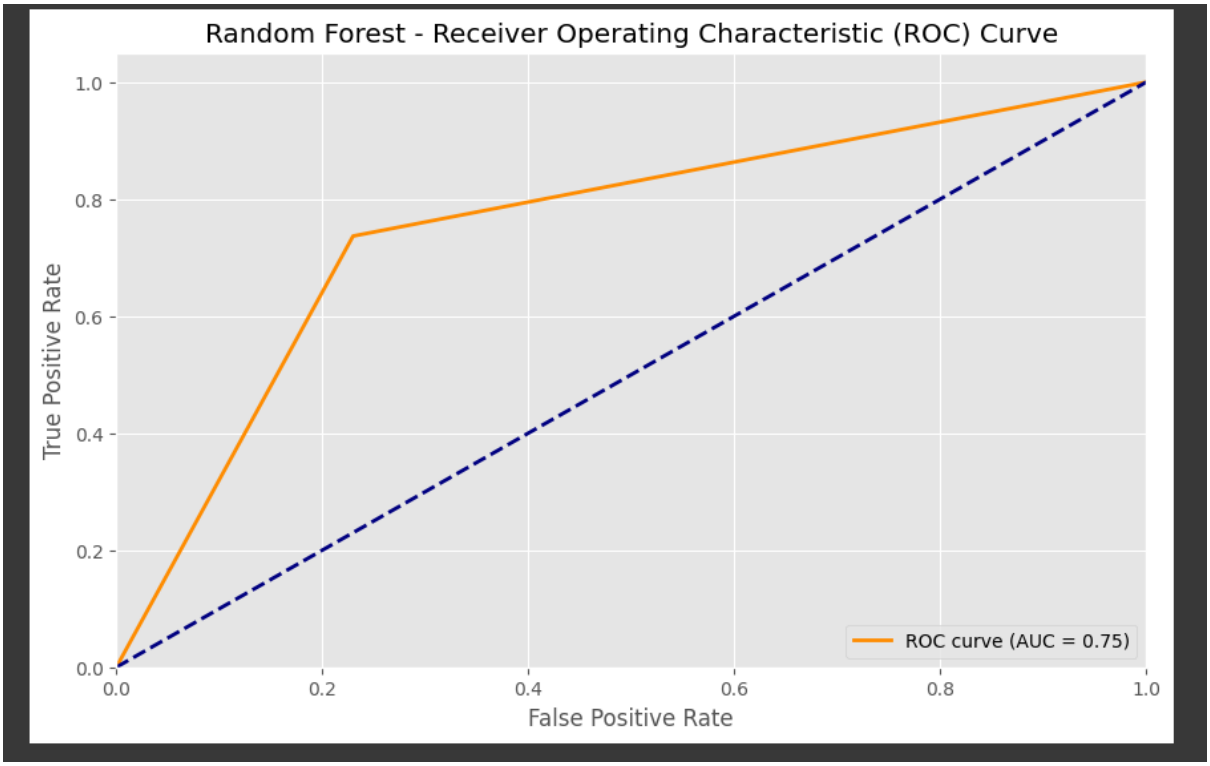
```
[ ] # Print the classification report
    print("Random Forest Classifier:")
    print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Classifier:
              precision    recall  f1-score   support

           0       0.82      0.77      0.79        87
           1       0.68      0.74      0.71        57

    accuracy                           0.76       144
   macro avg       0.75      0.75      0.75       144
weighted avg       0.76      0.76      0.76       144
```

```
[ ] # Calculate and print ROC AUC score
    roc_auc = metrics.roc_auc_score(y_test, y_pred_rf)
    print(f'ROC AUC score:', roc_auc)
```

```
ROC AUC score: 0.7534785238959468
```

```
[ ] fpr, tpr, thresholds = roc_curve(y_test, y_pred_rf)

    # Calculate the area under the ROC curve
    roc_auc = auc(fpr, tpr)

    # Plot the ROC curve
    plt.figure(figsize=(10, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Random Forest - Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```




```
[ ]  # Random Forest
     mse_rf = mean_squared_error(y_test, y_pred_rf)
     rmse_rf = np.sqrt(mse_rf)
     print(f"\nMean Squared Error (MSE) for Random Forest: {mse_rf:.2f}")
     print(f"Root Mean Squared Error (RMSE) for Random Forest: {rmse_rf:.2f}")
```

```
Mean Squared Error (MSE) for Random Forest: 0.24
Root Mean Squared Error (RMSE) for Random Forest: 0.49
```

Support Vector Machine

```
[ ]  # Initialize the Support Vector Machine classifier
     svm_classifier = SVC(random_state=42)

     # Train the SVM model
     svm_classifier.fit(X_train_final, y_train)

     # Test the SVM model
     y_pred_svm = svm_classifier.predict(X_test_final)
```

Model Evaluation of SVM Classifier

```
[ ]  accuracy_svm= accuracy_score(y_test, y_pred_svm)
     print(f" Model Accuracy for SVM Classifier: {accuracy_svm:}")
```

```
 Model Accuracy for SVM Classifier: 0.7222222222222222
```

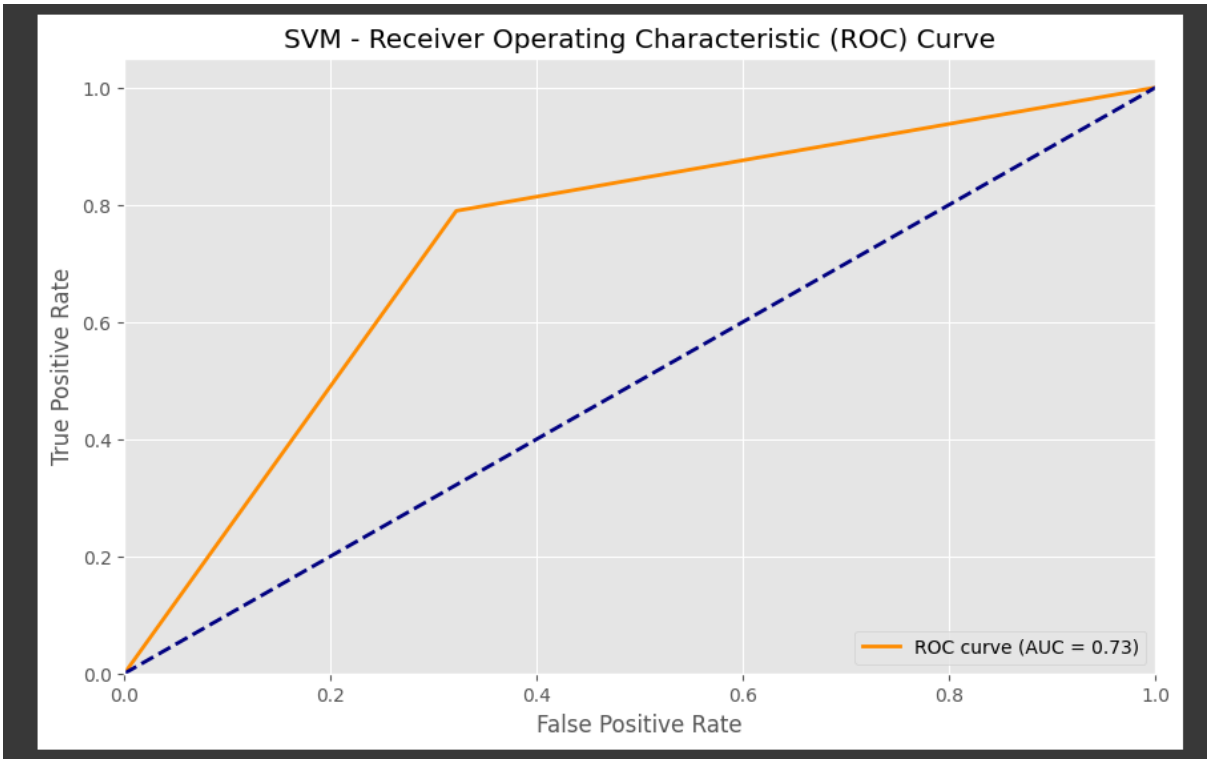
```
[ ]  # Print the classification report for SVM
     print("Support Vector Machine Classifier:")
     print(classification_report(y_test, y_pred_svm))
```

```
Support Vector Machine Classifier:
              precision    recall  f1-score   support

           0       0.83      0.68      0.75        87
           1       0.62      0.79      0.69        57

    accuracy                           0.72       144
   macro avg       0.72      0.73      0.72       144
weighted avg       0.75      0.72      0.73       144
```

```
[ ]  # Calculate and print ROC AUC score for SVM
     roc_auc_svm = metrics.roc_auc_score(y_test, y_pred_svm)
     print('ROC AUC score (Support Vector Machine):', roc_auc_svm)
```

```
ROC AUC score (Support Vector Machine): 0.7338173018753781
```

```
[ ]  fpr, tpr, thresholds = roc_curve(y_test, y_pred_svm)

     # Calculate the area under the ROC curve
     roc_auc = auc(fpr, tpr)

     # Plot the ROC curve
     plt.figure(figsize=(10, 6))
     plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('SVM - Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc='lower right')
     plt.show()
```




```
[ ]  # SVM
     mse_svm = mean_squared_error(y_test, y_pred_svm)
     rmse_svm = np.sqrt(mse_svm)
     print(f"\nMean Squared Error (MSE) for Support Vector Machine (SVM): {mse_svm:.2f}")
     print(f"Root Mean Squared Error (RMSE) for Support Vector Machine (SVM): {rmse_svm:.2f}")
```

```
Mean Squared Error (MSE) for Support Vector Machine (SVM): 0.28
Root Mean Squared Error (RMSE) for Support Vector Machine (SVM): 0.53
```

Logistic Regression

```
[ ]  # Initialize the Logistic Regression classifier
     logreg_classifier = LogisticRegression(random_state=42)

     # Train the Logistic Regression model
     logreg_classifier.fit(X_train_final, y_train)

     # Test the Logistic Regression model
     y_pred_logreg = logreg_classifier.predict(X_test_final)
```

Model Evaluation of Logistic Regression

```
[ ]  accuracy_logreg= accuracy_score(y_test, y_pred_logreg)
     print(f" Model Accuracy for Logistic Regression Classifier: {accuracy_logreg:}")
```

```
 Model Accuracy for Logistic Regression Classifier: 0.8263888888888888
```

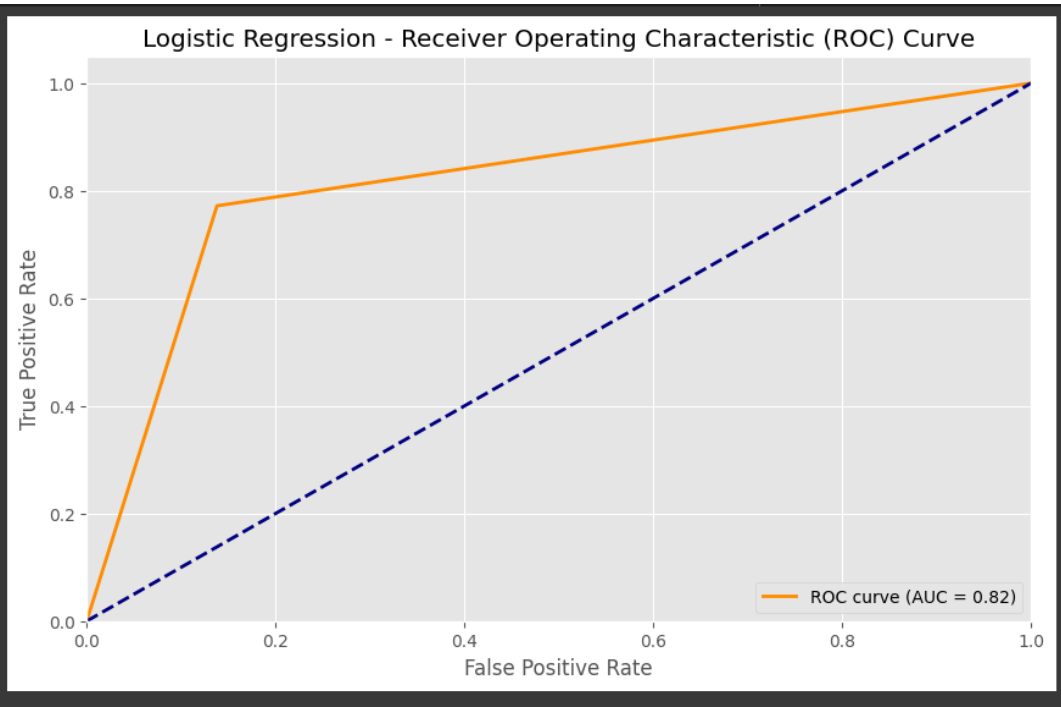
```
[ ]  # Print the classification report for Logistic Regression
     print("Logistic Regression Classifier:")
     print(classification_report(y_test, y_pred_logreg))
```

```
Logistic Regression Classifier:
              precision    recall  f1-score   support

           0       0.85      0.86      0.86        87
           1       0.79      0.77      0.78        57

    accuracy                           0.83       144
   macro avg       0.82      0.82      0.82       144
weighted avg       0.83      0.83      0.83       144
```

```
[ ]  # Calculate and print ROC AUC score for Logistic Regression
     roc_auc_logreg = metrics.roc_auc_score(y_test, y_pred_logreg)
     print('ROC AUC score (Logistic Regression):', roc_auc_logreg)
```

```
ROC AUC score (Logistic Regression): 0.8169993950393224
```

```
[ ]  fpr, tpr, thresholds = roc_curve(y_test, y_pred_logreg)

     # Calculate the area under the ROC curve
     roc_auc = auc(fpr, tpr)

     # Plot the ROC curve
     plt.figure(figsize=(10, 6))
     plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc_auc))
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Logistic Regression - Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc='lower right')
     plt.show()
```



```
[ ]  # Logistic Regression
     mse_logreg = mean_squared_error(y_test, y_pred_logreg)
     rmse_logreg = np.sqrt(mse_logreg)
     print(f"\nMean Squared Error (MSE) for Logistic Regression: {mse_logreg:.2f}")
     print(f"Root Mean Squared Error (RMSE) for Logistic Regression: {rmse_logreg:.2f}")
```

```
Mean Squared Error (MSE) for Logistic Regression: 0.17
Root Mean Squared Error (RMSE) for Logistic Regression: 0.42
```

```
[ ]  X_train_final.shape[1]
```

```
28016
```

Model Comparison:

Comparison the performance of all the models by using accuracy,classification report, AUC and ROC curve, MSE and RMSE.

Accuracy Score

```
[ ]  results = pd.DataFrame({
         'Model':['Support Vector Machine', 'Logistic Regression', 'Random Forest', 'Naive Bayes', 'LSTM'],
         'Accuracy Score':[accuracy_svm, accuracy_logreg, accuracy_rf, accuracy_nb, accuracy_lstm]})
     result_df = results.sort_values(by = 'Accuracy Score', ascending = False)
     result_df = result_df.set_index('Accuracy Score')
     result_df
```

Accuracy Score	Model
0.833333	Naive Bayes
0.826389	Logistic Regression
0.791667	LSTM
0.756944	Random Forest
0.722222	Support Vector Machine

Classification Report

Classification Report

```
[ ]  # Create a DataFrame for Classification Report
     classification_reports = []

     # Support Vector Machine
     report_svm = classification_report(y_test, y_pred_svm, output_dict=True)
     classification_reports.append({'Model': 'SVM', **report_svm['weighted avg']})

     # Logistic Regression
     report_logreg = classification_report(y_test, y_pred_logreg, output_dict=True)
     classification_reports.append({'Model': 'Logistic Regression', **report_logreg['weighted avg']})

     # Random Forest
     report_rf = classification_report(y_test, y_pred_rf, output_dict=True)
     classification_reports.append({'Model': 'Random Forest', **report_rf['weighted avg']})

     # Naive Bayes
     report_nb = classification_report(y_test, y_pred_nb, output_dict=True)
     classification_reports.append({'Model': 'Naive Bayes', **report_nb['weighted avg']})

     # LSTM
     classification_reports.append({'Model': 'LSTM', **report_lstm['weighted avg']})

     # Create DataFrame
     classification_df = pd.DataFrame(classification_reports)
     classification_df = classification_df.set_index('Model')

     # Sort DataFrame by Score in descending order
     classification_df = classification_df.sort_values(by='f1-score', ascending=False)

     # Print the DataFrame
     print(classification_df)
```

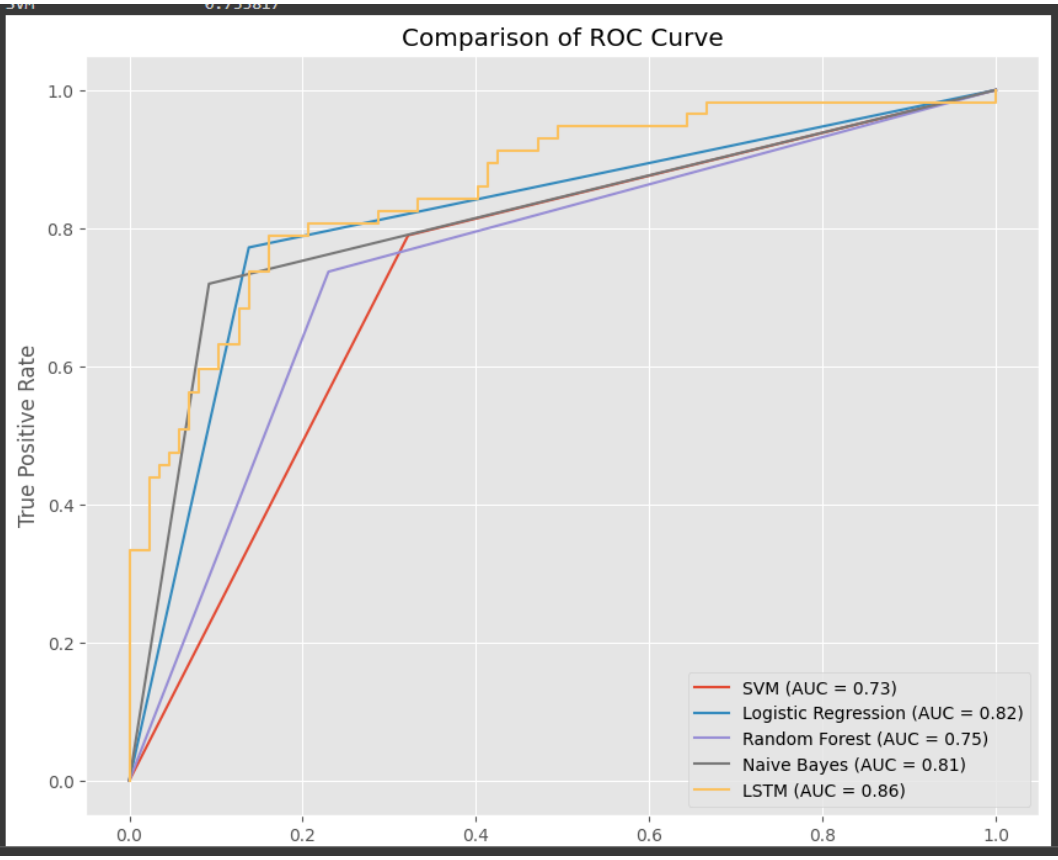
Model	precision	recall	f1-score	support
Naive Bayes	0.833620	0.833333	0.830707	144
Logistic Regression	0.825927	0.826389	0.826117	144
LSTM	0.792677	0.791667	0.786132	144
Random Forest	0.761794	0.756944	0.758455	144
SVM	0.746061	0.722222	0.725252	144

AUC and ROC curve

```
# Create a DataFrame for AUC
auc_results = []

# SVM
fpr_svm, tpr_svm, _ = roc_curve(y_test, y_pred_svm)
roc_auc_svm = auc(fpr_svm, tpr_svm)
auc_results.append({'Model': 'SVM', 'AUC': roc_auc_svm})

# Logistic Regression
fpr_logreg, tpr_logreg, _ = roc_curve(y_test, y_pred_logreg)
roc_auc_logreg = auc(fpr_logreg, tpr_logreg)
auc_results.append({'Model': 'Logistic Regression', 'AUC': roc_auc_logreg})

# Random Forest
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)
auc_results.append({'Model': 'Random Forest', 'AUC': roc_auc_rf})

# Naive Bayes
fpr_nb, tpr_nb, _ = roc_curve(y_test, y_pred_nb)
roc_auc_nb = auc(fpr_nb, tpr_nb)
auc_results.append({'Model': 'Naive Bayes', 'AUC': roc_auc_nb})

# LSTM
auc_results.append({'Model': 'LSTM', 'AUC': roc_auc_lstm})

# Create DataFrame
auc_df = pd.DataFrame(auc_results)
auc_df = auc_df.set_index('Model')

# Sort DataFrame by AUC in descending order
auc_df = auc_df.sort_values(by='AUC', ascending=False)

# Print the DataFrame
print(auc_df)
```

```
# Plot ROC curves
plt.figure(figsize=(10, 8))
plt.plot(fpr_svm, tpr_svm, label=f'SVM (AUC = {roc_auc_svm:.2f})')
plt.plot(fpr_logreg, tpr_logreg, label=f'Logistic Regression (AUC = {roc_auc_logreg:.2f})')
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {roc_auc_rf:.2f})')
plt.plot(fpr_nb, tpr_nb, label=f'Naive Bayes (AUC = {roc_auc_nb:.2f})')
plt.plot(fpr_lstm, tpr_lstm, label=f'LSTM (AUC = {roc_auc_lstm:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Comparison of ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

Model	AUC
LSTM	0.860657
Logistic Regression	0.816999
Naive Bayes	0.813672
Random Forest	0.753479
SVM	0.733817



MSE and RMSE

```
[ ] # Create a DataFrame for MSE and RMSE
    mse_rmse_results = []

    # SVM
    mse_svm = mean_squared_error(y_test, y_pred_svm)
    rmse_svm = sqrt(mse_svm)
    mse_rmse_results.append({'Model': 'SVM', 'MSE': mse_svm, 'RMSE': rmse_svm})

    # Logistic Regression
    mse_logreg = mean_squared_error(y_test, y_pred_logreg)
    rmse_logreg = np.sqrt(mse_logreg)
    mse_rmse_results.append({'Model': 'Logistic Regression', 'MSE': mse_logreg, 'RMSE': rmse_logreg})

    # Random Forest
    mse_rf = mean_squared_error(y_test, y_pred_rf)
    rmse_rf = np.sqrt(mse_rf)
    mse_rmse_results.append({'Model': 'Random Forest', 'MSE': mse_rf, 'RMSE': rmse_rf})

    # Naive Bayes
    mse_nb = mean_squared_error(y_test, y_pred_nb)
    rmse_nb = np.sqrt(mse_nb)
    mse_rmse_results.append({'Model': 'Naive Bayes', 'MSE': mse_nb, 'RMSE': rmse_nb})

    # LSTM
    mse_rmse_results.append({'Model': 'LSTM', 'MSE': mse_lstm, 'RMSE': rmse_lstm})

    # Create DataFrame
    mse_rmse_df = pd.DataFrame(mse_rmse_results)
    mse_rmse_df = mse_rmse_df.set_index('Model')

    # Sort DataFrame by MSE in ascending order
    mse_rmse_df = mse_rmse_df.sort_values(by='MSE', ascending=True)

    # Print the DataFrame
    print(mse_rmse_df)
```

```
                          MSE      RMSE
Model
LSTM                 0.159977  0.399972
Naive Bayes          0.166667  0.408248
Logistic Regression  0.173611  0.416667
Random Forest        0.243056  0.493007
SVM                  0.277778  0.527046
```

```
# Check to see where the model failed to predict positive/negative tweets from highest accuracy model
results_df = X_test.copy()
results_df['true_labels'] = y_test
results_df['predicted_labels'] = y_pred_nb

# Print false positives
false_positives = results_df[(results_df['predicted_labels'] == 1) & (results_df['true_labels'] == 0)]
print("False Positives:")
print(false_positives['text'])

# Print false negatives
false_negatives = results_df[(results_df['predicted_labels'] == 0) & (results_df['true_labels'] == 1)]
print("\nFalse Negatives:")
print(false_negatives['text'])
```

```
False Positives:
119    laaaa cepat nye naik nak tengok geng bergabung...
521    vibes drama jaman jaman 1900 an terang an dram...
696                       kuat kudrat darurat nak hajat
988    sembako pasar tradisional tidak kenakan ppn te...
589    rakyat disuruh menjaga diri membisu tidak inis...
918    buseet jaman udah teknologi 50 jualan nya buki...
761    team alzakry paria no 1 keling penjilat lubang...
325             bang bawa membaca sikit stfc menasihati u
Name: text, dtype: object
```

```
False Negatives:
1311    tbh sincerely abg enjoy sngt pusat kuarantin s...
1308       tindakan wajar pujian dicontohi adun ahli lain
1076    biar orang kita orang orang batu balas batu nisan
1495    brpa thn training nk baling tubrpe kali baling...
253           peruntuk rm300 pesakit covid19 kategori b40
1447    bodoh la ni nak solat kat suraumasjid tutup la...
1307              thread ringkas harta diisytihar sebaga
1304    program dipancarkan langsung menerusi media so...
1248               afro aliff haiqal mukhairi kuat power
1256                        kpk kuat 75 orang anti nkri
1340    belang dok sedor ni week 13 doh habis doh kut ...
835     kinerja kpk masyarakat mendukung menu kpk bers...
1499    160365 tubuh nya terpecah belah kaku batu mulu...
1029    ramai kongsi post ni insyaallah cepatlah dana ...
1384    dh rentas support sektor pelancongan sendiri t...
1469    menjumpai kegagalan tidak peduli besok seterus...
Name: text, dtype: object
```

GUI

GUI from a real Twitter user to check the predicted sentiment

> Kerajaan gagal. Kerajaan Muhyiddin gagal. Gagal, gagal total. Melingkuplah korang semua.
> — IAN ZAINAL (@IANZAINAL) May 4, 2021

In [*]:

```
from tkinter import *

# Define the function to detect sentiment
def detect_sentiment():
    # Get the input text from the text area
    input_text = textArea.get("1.0", "end-1c")

    # Preprocess the input text
    preprocessed_text = preprocessing(input_text)

    # Transform the preprocessed text using the same vectorizer
    input_text_tfidf = vectorizer.transform([preprocessed_text])
    input_text_features = [[any(word in preprocessed_text for word in BAD_WORDS),
                            sum(preprocessed_text.count(word) for word in BAD_WORDS) / len(preprocessed_text.split()) if len(pre
                            any(word in preprocessed_text for word in GOOD_WORDS),
                            sum(preprocessed_text.count(word) for word in GOOD_WORDS) / len(preprocessed_text.split()) if len(pr

    input_text_final = pd.concat([pd.DataFrame(input_text_tfidf.toarray()), pd.DataFrame(input_text_features)], axis=1)

    # Predict the sentiment of the input text
    prediction = nb.predict(input_text_final)[0]

    # Map the prediction to the corresponding sentiment label
    sentiment = {0: 'Negative', 1: 'Positive'}
    predicted_sentiment = sentiment[prediction]

    # Update the GUI with the predicted sentiment
    overallField.delete(0, END)
    overallField.insert(10, predicted_sentiment)

# Define the function to clear all fields
def clearAll():
    overallField.delete(0, END)
    textArea.delete(1.0, END)

# Define the function to exit the GUI
def exit():
    gui.destroy()

# Create the GUI
gui = Tk()
gui.config(background="light blue")
gui.title("Sentiment Detector")
gui.geometry("450x400")

# Create the widgets
enterText = Label(gui, text="Enter Your Sentence", bg="light yellow")
textArea = Text(gui, height=10, width=45, font="lucida 13")
check = Button(gui, text="Check Sentiment", fg="black", bg="light blue", command=detect_sentiment)
overall = Label(gui, text="Model Prediction :", bg="light yellow")
overallField = Entry(gui)
clear = Button(gui, text="Clear", fg="black", bg="light blue", command=clearAll)
Exit = Button(gui, text="Exit", fg="black", bg="light blue", command=exit)

# Add the widgets to the GUI
enterText.grid(row=0, column=2)
textArea.grid(row=1, column=2, padx=10, sticky=W)
check.grid(row=2, column=2)
overall.grid(row=3, column=2)
overallField.grid(row=4, column=2)
clear.grid(row=5, column=2)
Exit.grid(row=6, column=2)

# Start the GUI event loop
gui.mainloop()
```

Centre for Mathematical Sciences Pusat Sains Matematik UNIVERSITI MALAYSIA PAHANG AL-SULTAN ABDULLAH **SUBJECT: BSD3523 MACHINE LEARNING**	**GROUP PROJECT** **DUE DATE:** Group Project Report: 12/1/2024 1100 PM Group Project Presentation: 15/1/2024 (Sec 01G & 02G) 16/1/2024 (Sec 03G)	**MARKS**: 80(20%)

CLO	Description	PLO mapping	Marks	Percentage
CLO2	Evaluate the machine learning model/pipeline in solving real world problems.	PLO2: Able to develop analytical and critical thinking by utilizing Data Analytics knowledge in solving various problems. C6: Critical Thinking and Problem Solving.	20	5%
CLO3	Construct the programming codes or workflows using appropriate machine learning tools.	PLO3: Able to design, implement and manage data resources using various Data Analytics technologies. P4: Modern Tools Usage	30	7.5%
CLO4	Develop leadership skill in grouping assessment.	PLO6: Able to take responsibility as a leader effectively. A3: Leadership	15	3.75%
CLO5	Integrate machine learning knowledges to the project and future problems.	PLO7: Able to show enthusiasm, independent learning, intellectual, self-control, confident and professionalism in completing the task. A4: Lifelong Learning	15	3.75%

Summary

To make real progress along the path toward becoming a machine learning engineer, it is helpful to apply all your machine learning knowledge to a project task. In this group project, students will:

- Identify real-world problems where machine learning can have impact.
- Implement machine learning tools on real data and evaluate the performance.
- Show their leadership skills in the project presentation.

INSTRUCTIONS:

1. Set up a group that consists of **FOUR (4)** or **FIVE (5)** members.

2. Imagine that you, the group leader, and the rest of members in a group are having a start-up consulting company. Create a name of your company which is also your group name. Assign specific role and responsibility for each of the member in a group, for example, CEO, Co-founder, CTO, consultant, machine learning engineer, or data analyst.

3. As a consulting company or firm, you have a client that need your expertise in completing a machine learning project (title of the project will be given by lecturer). Solve the project by using all the knowledge that you have learn in the class.

4. Perform all analysis using Python. Additional tools to be used are recommended.

5. Make sure to complete all the machine learning process starting from data acquisition until model evaluation.

6. Write your report based on the report format provided below.

7. Submit your draft report in **pdf** format at **KALAM** platform. Make sure to attach your coding as an appendix.

8. Publish your findings in the Medium platform (optional but additional marks will be given).

9. **Remember to submit by the dateline. LATE submission will be deducted by 20% of the total marks.**

Plagiarism:

Your attention is drawn to the no-plagiarism policy. These covers using ChatGPT, copy from your friends, plagiarism from published works, and any other attempts to gain an unfair advantage in assessment. The work you submit must conform to this policy. If plagiarism is discovered, any parties involved will take equal blame (will get zero mark). For writing that using ChatGPT, 50% marks will be de deducted from CLO2.

Report format

1. Cover (use the cover page provided by lecturer)
2. Table of content.
3. Introduction (About the start-up company and role for each group member).
4. Project description, problem statement, objectives, scope, and limitation of the project.
5. Data description, data pre-processing, and data analysis.
6. Developing of machine learning algorithm.
7. Interpretation of the results.
8. Conclusion and recommendations.
9. References (Cite every single reference used in completing the project).
10. Appendices (Python code and proof of publishing on Medium).

There are no minimum or maximum pages of the report. Please use Times New Roman (size 12) or Arial (size 11) for your report. Make sure all paragraphs are **justified** and use 1.15 spacing. Name all the figures and tables included in the report.

RUBRICS FOR PROJECT REPORT

CLO2/PLO2: Evaluate the machine learning model/pipeline in solving real world problems.

Item Assessed	Weak 0	Very Poor 1	Poor 2	Fair 3	Good 4	Excellent 5	Weightage (W)	Score
Description and explanation of the selected project, problem to be solved, objectives and limitation of the project.	No description and explanation of the selected project, problem to be solved, objectives and limitation of the project.	Poorly describe and explain about the project selected with no problem to be solved, objectives and limitation included.	Poorly describe and explain about the project selected, problem to be solved, objectives and limitation of the project.	Fairly describe and explain about the project selected, problem to be solved, objectives and limitation of the project.	Clearly describe and explain about the project selected, problem to be solved, objectives and limitation of the project.	Excellently describe and explain about the project selected, problem to be solved, objectives and limitation of the project.	1	
Explanation of the project methodology.	No explanation on the project methodology.	Limited explanation on the project methodology.	Poor explanation on the project methodology.	Fair explanation on the project methodology.	Good explanation on the project methodology.	Excellent explanation on the project methodology.	0.5	
Explanation on the data analysis/ data pre-processing results.	Unable to provide any explanation on the data analysis/ data pre-processing results.	Limited explanation on the data analysis/ data pre-processing results.	Able to explain the data analysis/ data pre-processing results but unclear and inaccurate.	Able to explain the data analysis/ data pre-processing results clearly but inaccurate.	Able to explain the data analysis/ data pre-processing results clearly.	Able to explain the data analysis/ data pre-processing results excellently.	1	
Explanation on the model evaluation and provide a comparison between models.	Unable to provide any explanation on the model evaluation and a comparison between models.	Limited explanation on the model evaluation and no comparison between models.	Poor explanation on the model evaluation but provide a comparison between models.	Fair explanation on the model evaluation and provide a comparison between models.	Good explanation on the model evaluation and provide a comparison	Excellent explanation on the model evaluation and provide a comparison	1	
Concluding remarks.	No concluding remarks provided.	Limited concluding remarks provided and inaccurate.	Concluding remarks provided but unclear and inaccurate.	Concluding remarks provided but partly inaccurate.	Clear and good concluding remarks provided.	Very clear and excellent concluding remarks provided.	0.5	
						Total Score	**/20**	

CLO3/PLO3: Construct the programming codes or workflows using appropriate machine learning tools.

Item Assessed	Weak 0	Very Poor 1	Poor 2	Fair 3	Good 4	Excellent 5	Weightage (W)	Score
Ability to construct machine learning pipeline that suit the project involved.	Unable to construct machine learning pipeline.	Able to construct the machine learning pipeline but the pipeline is wrong.	Able to construct the machine learning pipeline but there is missing process in a pipeline.	Able to construct machine learning pipeline fairly.	Good in constructing the machine learning pipeline.	Able to construct machine learning pipeline excellently.	2	
Ability to evaluate and deploy the model.	Unable to evaluate and deploy the model.	Able to evaluate the model but unable to deploy the model.	Poorly able to evaluate and deploy the model.	Fairly able to evaluate and deploy the model.	Able to evaluate and deploy the model nicely.	Able to evaluate and deploy the model excellently.	1	
Easy to understand the codes constructed.	No code constructed.	Difficult to follow the structure and flow of the codes.	Fairly difficult to follow the structure and flow of the codes.	Fairly easy to follow the structure and flow of the codes.	Easily to follow the structure and flow of the codes.	Well easily to follow the structure and flow of the codes.	1	
Code execution.	No code executed.	Code does not work or has major warning.	Code mostly works but has minor error that effect on the results.	Code mostly works and has minor error but does not affect the results.	Code works with no error at all but did not complete.	Code works completely and excellently with no error and correct results.	1	
Ability to visualise the results obtained (ROC Curve, prediction, etc.)	Unable to visualise any results obtained.	Able to visualize some results obtained with unclear figures.	Able to visualize some results obtained with clear figures.	Able to visualise all the results obtained but the figures are not clear with wrong labels on the axis.	Able to visualise all the results obtained with clear figures but wrong labels on the axis.	Able to visualise all the results obtained excellently with clear figures and correct labels on the axis.	1	
						Total Score	**/30**	

CLO5/PLO7: Integrate machine learning knowledges to the project and future problems

Item Assessed	Weak 0	Very Poor 1	Poor 2	Fair 3	Good 4	Excellent 5	Weightage (W)	Score
Show advanced machine learning skills in completing the project.	No advanced machine learning skill is shown.	Limited advanced machine learning skills is shown in completing the project.	Poorly shown advanced machine learning skills in completing the project.	Moderately shown advanced machine learning skills in completing the project.	Show advanced machine learning skills in completing the project.	Show excellent and advanced machine learning skills in completing the project.	1	
Reflection from prior learning towards life experience.	Do not review any prior learning at any level.	Reviews prior learning at a surface level, but without clarifying meaning or indicating a broader perspective about educational or life events.	Reviews prior learning with limited capacity, giving minor clarification or broad perspective.	Reviews prior learning and shows clarification or broad perspective about educational or life events.	Reviews prior learning in some depth, revealing clear meaning and indicating broad perspectives related to educational events.	Review prior learning in depth to reveal significantly changed perspectives about educational and life experiences.	1	
Propose future study/ extended study from the proposed solution.	No proposal has been made.	The recommendations that have been made are inappropriate.	Poorly proposed the recommendations that can be improved.	Fairly proposed the recommendations that can be improved.	Good recommendations on the proposal but can be improved.	A few brilliant recommendations have been proposed.	1	
						Total Score	**/15**	

RUBRIC FOR PROJECT PRESENTATION.

CLO4/PLO6: Develop leadership skill in grouping assessment.

Item Assessed	Weak 0	Very Poor 1	Poor 2	Fair 3	Good 4	Excellent 5	Weightage (W)	Score
Demonstrate ability to coordinate tasks and resources based on the role.	Unable to coordinate tasks.	Limited ability to coordinate tasks and resources based on the role.	Able to demonstrate ability to coordinate tasks and resources poorly.	Able to demonstrate ability to coordinate tasks and resources fairly.	Able to demonstrate ability to coordinate tasks and resources nicely.	Able to demonstrate ability to coordinate tasks and resources effectively.	1	
Leadership skill from the role in the task.	Unable to show leadership skill based on the role given.	Poor leadership skill is shown.	Able to show leadership skill but very minimal.	Moderate leadership skill is shown.	Able to show true leadership skill based on the role given with help from others.	Able to show true leadership skill based on the role given without help from others.	2	
						Total Score	**/15**	