Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News

Abstract

The public sentiment an important factor for influencing the financial world. X (formerly known as Twitter) have emerged as a crucial source for financial discussions. The goal is to create finance advanced sentiment analysis model for X's financial news postings through natural language processing (NLP) techniques. The primary challenge of this project is to handle unstructured data from X which includes slang, hashtags, abbreviations, and emojis, through rigorous pre-processing methods. The methodology consists of cleaning the text, training machine learning algorithms on financial-specific dataset and to classify them into bullish, bearish and neutral. The project will incorporate a visualisation tool to show public sentiment patterns and provide valuable insights of the market behaviour and support in predictive analytics for better decision-making in the financial realm. The goal is to build and analytical framework to improve the understanding of financial sentiments and support financial institution in risk management and strategic planning. This initiative is carried out to leverage the advanced NLP techniques to deliver comprehensive financial sentiment analysis with visualisations and improve data-driven decision-making in finance.

Keywords: Sentiment Analysis, Financial News, Natural Language Processing, Machine Learning, Predictive Analytics.

Contents

Table of Figures

Chapter 1: **Introduction**

In the dynamic world of finance, viewpoints and perspectives can change as rapidly as the stock market, therefore it is believed that it is crucial to access public sentiment accurately and immediately. The realm of finance is known for being extremely sensitive towards the response based on public opinion in finance, even the smallest shift in the sentiment changes drastically affects the direction of the market in the finance world (Mishev et al., 2020).

X (formerly known as Twitter) is an important platform for financial discussions and conversations, it is where people get the latest highlights for financial news. X provides an endless flow of information which are conveyed by users, businessmen, entrepreneurs, finance people and various individuals who are involved in finance. Therefore, X has a wide spectrum of comments from casual to professional opinions. The project aims to implement advanced sentiment analysis that is specifically used in analysing X's posts regarding financial news.

The reason for highlighting X is due to X plays a role in the market dynamics. The expressions from investor's opinions on X could substantially affect the financial decisions and market consequences. X is a platform which delivers news and information by investors and financial professionals which makes it a necessary tool for understanding the financial sentiment. Various studies have shown that sentiments on X can predict the market and financial phenomena (Kraaijeveld & De Smedt, 2020). Hence, this project will be optimising the predictive abilities of X's data to provide data-driven insights for investment strategies.

Analysing X's data can be challenging as the data is all unstructured. The text usually has slang, hashtags, abbreviations, and emojis which makes the traditional analytic techniques difficult to interpret their meanings accurately. This is why preprocessing is crucial for the raw X data, it is done to remove the noises of the text and keep the important text for analysis; by doing so, one can increase the accuracy of sentiment prediction (Pano & Kashef, 2020).

Besides that, classification is done with algorithms of machine learning. The algorithms will be trained using a dataset which consists of sentimental tests from X that consists of various sentiment expressions of bullish, bearish and neutral. As an example of a

bullish text “Just bought some $AAPL! The company’s new product launch makes the financial even stronger than ever!” the “stronger than ever” displays the user’s confidence in investing in this company. The ability to differentiate if the text data obtained is bullish, bearish or neutral is important as it represents a comprehensive and accurate sentiment (Tai et al., 2022).

Moreover, the project aims to integrate visualisations to analyse sentiment patterns and classification. These visualisations aid in providing valuable insights into the public’s sentiment in financial news which enables stakeholders to understand the phenomena in the current market. In addition to that, sentiment analysis is used to improve the predictions for the implication of finance risk management. For example, financial institutions can enhance their ability to anticipate a market’s change and implement risk strategies to mitigate potential losses (Dong & Liu, 2021).

Ultimately, the expected result of this project is to create a robust analytical framework to enhance the comprehension of financial attitudes. By employing advanced natural language processing (NLP) and visualisation tools, the framework can offer a comprehensive analysis which can be used for decision-making processes and strategic planning. As the market progresses every day, it is important to access the public sentiment regarding finance to obtain good outcomes. This project signifies an advancement of combining sentiment analysis with financial highlights in X to provide people with a potential tool for studying the dynamics of finance in X.

1.1 Problem Statement

X has become an important tool for analysing financial news, it exerts real-time influence on the market behavioural patterns and investment decisions. The huge amount of unstructured data from X is an obstacle to extracting and accessing sentiments which are related to financial news. This is due to the informal and casual language used which often consists of slang, hashtags and abbreviations.

Therefore, this explains why current sentiment analysis methods might face obstacles in handling the sentiment of financial data from X as these informal and casual languages potentially cause misinterpretation when conventional natural language processing (NLP) approaches (Adwan et al., 2020) are employed.

In addition to that, there is an absence of effective preprocessing techniques for handling the noise from the unstructured nature of X's data. This complicates the sentiment extraction and analysis which affects and causes limitations to the accuracy of the conventional NLP models which can be used to capture the sentiments expressed in X (Pano & Kashef, 2020).

Furthermore, the quality of the currents sentiment analysis models is often undermined by the data it is trained with. Financial sentiment research might suffer from lack of precision and recall, this is because of using generic datasets which do not cover the intricacies of financial jargon and finance specific languages (Mishev et al., 2020).

1.2 Justification of the Project

The project's significance is in its capacity to provide insights into public sentiment regarding financial news on X (formerly Twitter). It is important to comprehend and analyse the sentiments of the unpredictable financial texts where the opinion of the public can rapidly affect stock prices and investment choices. The project narrows the divide between qualitative analysis in sentiment by designing with applying an advanced sentiment analysis system. This system aids in enhancing predictive analytics and facilitates more informed decision-making processes in the financial industry. This might also help in risk management strategies and market investment planning. In fundamental terms, this project aims to provide a sentiment analytical framework which enhances the understanding of financial dynamics which benefits individuals and institutions in finance.

1.3 Methodology Overview

This project employs a methodology to develop a sentiment analysis system for financial news for X. The process begins with the data collection of X's relevant data in financial news. Then the data is pre-processed, this process starts with cleaning the text, tokenization, removing stop words, then word stemming or lemmatization, and normalisation. Once it is processed, the features will be extracted through TF-IDF and predefined embedding of words. For model training, the models used are traditional machine learning, lexicon-based, deep learning and hybrid. The best model will be used to train and evaluate through the metrics selected. Lastly, visualisation tools will display the sentient trends, provide insights into public sentiment regarding financial news, and improve decision-making.

1.4 Research Questions

1. How do different NLP techniques and methods address the primary challenges of processing X’s data for sentiment analysis in financial news?
2. Which type of approach can leverage and improve sentiment accuracy of finance news in X?
3. How to evaluate sentiment analysis accuracy on finance news in X to enhance prediction?

1.5 Objectives

1. To enhance accuracy in X's financial news sentiment analysis, this study explores various NLP techniques to address preprocessing challenges.
2. To develop a model that integrates the most effective NLP techniques for improving sentiment detection in X's financial news.
3. To develop and evaluate methods for improving sentiment analysis in Twitter financial news to enhance the accuracy.

1.6 Scope of the Project

The study focuses on analyzing the financial news-related texts on X by implementing advanced natural language processing (NLP) to evaluate public opinion. The scope consists of the collection of data from X and preprocessing it to handle slang, abbreviations, hashtags and emojis. The project will be training and evaluating the models through a dataset which consists of financial sentiments. The models used are traditional machine learning models, lexicon-based models and hybrid models. The performances are evaluated though metrics.

The results from the best model will be incorporated with visualization tools to analyze the sentiments and offer insights to the public sentiment regarding financial news and highlights. This project will be focusing on training the model to obtain the results and refining this report of the project based on the feedback from lecturers and supervisor. This project will not entail any direct engagement with end-users or experts. The studies made will be derived from the existing resources and literature.

1.7 Outline of Report

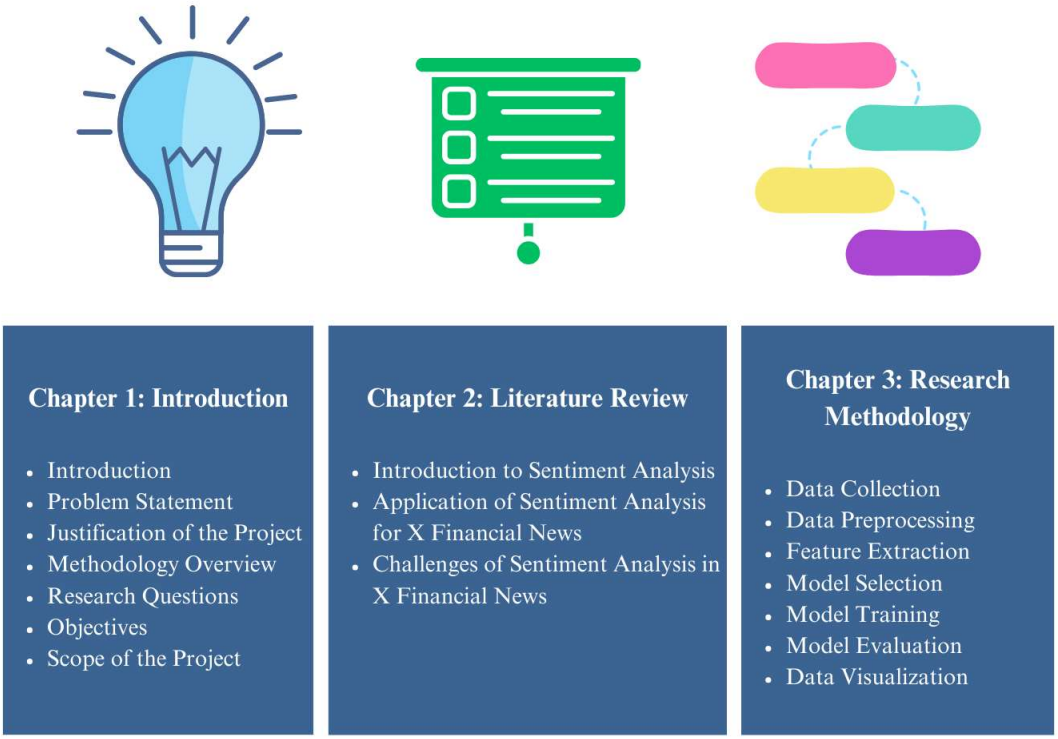


Figure 1 shows the outline of the project.

Chapter 1 is the Introduction. It will be discussing on the definition of understanding financial sentiment through X. This chapter consists of the Problem Statement which highlights the challenges faced when analyzing financial sentiment on X and the limitations faced in the existing methods. Justification of the Project explains the importance of financial sentiment insights. Methodology Overview discusses the overview flow of the research methodology. The research question displays the problems which are studied in this project and the Objectives reflect the Research Questions. Project scope will brief on how the project will be in this report

Chapter 2 is a Literature Review; it discusses the 3 topics which will be discussed which are as below:

- Sentiment Analysis Introductory
- Application of Sentiment Analysis for X Financial News
- Challenges of Sentiment Analysis in X Financial News

Chapter 3, is about Research Methodology which involves the steps for the execution of the finance sentiment analysis model.

Chapter 2: Literature Review

2.1 Sentiment Analysis Introductory

2.1.1 What is Sentiment Analysis?

Sentiment Analysis is a subcategory of natural language processing (NLP) which specifically distinguishes, extracts and classifies the sentiments that are expressed in text format. This process involves determining the sentiment polarity, in this project it will be bullish, bearish and neutral of the sentimental financial news from X. It is utilized in analytics, social behaviours, political analysis or even disaster forecasts. This is a computational treatment for analyzing and then classifying opinions, emotions and subjectivity in the text (Yuan et al., 2020). The goal is to classify the sentiments of individuals in X.

There are several approaches for sentiment analysis, which encompasses models of traditional machine learning and advanced deep learning. Traditional methods used are support vector machine (SVM) logistic regression and Lexicon-based approach depending on pre-established dictionaries of words with sentiment. Whereas convolutional neural networks (CNNs) and recurrent neural networks (RNNs), these deep learning models are good at understanding text pattern (Ha et al., 2019).

2.1.2 Approaches in Sentiment Analysis for Financial News on X

Sentiment analysis is beneficial in the financial domain, it can offer significant insights into public opinion and understand the financial trends and behaviours of the investors. Various approaches are employed to analyse the financial news on X:

i) Lexicon-Based Approaches:

The Lexicon-based approach uses predefined dictionaries of sentiment-bearing words to evaluate. The lexicon-based method may be straightforward to implement however it is limited by the comprehensiveness and domain-specificity of the lexicon (Habimana et al., 2019).

Lexicon-based techniques offer a significant benefit in terms of their simplicity and straightforwardness in implementation. They are unsupervised models, which makes them economical and efficient to implement. Furthermore, these models are highly interpretable. This enables the users to understand the process of how sentiment scores are derived from particular words in the text. As an example, Valence Aware Dictionary and sEntiment Reasoner (VADER) is tailored to handle social media text and it is widely used due to its robust performance in this field (Habimana et al., 2019).

Nevertheless, one might argue that lexicon-based methods are limited as it relies on the predefined dictionaries. Therefore, it might fail to capture the context when it is a sarcasm. As an example, financial jargon and the use of casual slangs or idiomatic expressions in the text can lead to inaccuracy in sentiment classification. Hence, this method alone might struggle with the dynamic and rapid change of the nature of language where new phrases and words frequently change like trends (Liu et al., 2020).

ii) Traditional Machine Learning:

This process involves extracting features from the text data using support vector machines (SVM), Naive Bayes, and random forests to categorise sentiment. This method can process a wide range of features and it is effective with sufficient labelled data.

The advantage of traditional machine learning methods resides in their extensive feature engineering, they can efficiently handle a wide range of feature datasets and obtain high accuracy with the dataset. The models can be trained and identify the complicated patterns in the text data, it have better accuracy compared to lexicon-based techniques. As an

example, support vector machines (SVM) and random forests have proven to categorise massive amounts of financial news in sentiment analysis tasks (Aziz & Starkey, 2020).

In contrast, traditional machine learning models requires extensive features for engineering that might be time consuming and complicated, and the model can be computationally intensive when it comes to handling large datasets. Furthermore, these models require continuous updates to maintain their performances (Ha et al., 2019). It may be argued that despite the model's effectiveness, the effort needed for model tunning and feature extraction may limit their adaptability and scalability in rapidly changing environments.

iii) Deep Learning Models:

Neural networks use numerous layers for acquiring text and classifying sentiment. Models such as LSTM, CNN, and transformers (e.g., BERT) can automatically learn data features. It is highly effective at capturing intricate patterns and context, resulting in exceptional performance that is at the forefront of the field.

Deep learning models offer significant benefits since they can autonomously acquire and extract complex characteristics from unprocessed text data, resulting in superior accuracy and resilience in sentiment categorization. These models have exceptional proficiency in catching the contextual and sequential characteristics of language, rendering them highly efficient for assessing social media and financial news. As an example, Yuan et al. (2020) mention that BERT has been adopted broadly for its performance in sentiment analysis.

However, labelled datasets are needed to train deep learning models. This can be a disadvantage for domains where acquiring large amounts of data can be expensive to obtain. In addition to that, deep learning models require significant computing resources which need large processing power and memory. Besides that, the model often operates as black boxes which makes the predictions less readable compared to more straightforward models. The lack of transparency can lead to problems in the financial context where explanations are crucial for trust and decision-making (Liu et al., 2020).

Iv) Hybrid Approaches:

By combining lexicon-based and deep or machine learning approaches, the combination can leverage the strengths of each technique, the hybrid approach can obtain higher accuracy by incorporating multimer sources.

Hybrid models provide the flexibility to use domain-specific features with learned features which might lead to higher and improved accuracy. For example, a hybrid model could integrate sentiment scores from lexicon with the features which are learned by deep learning models to enhance the performance. This combination can be effective in complex fields such as finance as it allows synthesis of data for accurate sentiment analysis (Mishev et al., 2020).

Although hybrid approaches have the potential to be effective, they might lead to complexity and might need extensive experimentation to determine the optimal combination of strategies. This makes hybrid approaches resource-intensive to develop and maintain. Moreover, the process of combining various models might lead to challenges of ensuring their compatibility and synergistic performance (Mishev et al., 2020). Therefore, one can comment that although hybrid models can achieve high accuracy, it is more complicated and resource intensive compared to other approaches. Therefore, careful consideration has to be taken before implementing hybrid model.

Approach	Description	Advantages	Disadvantages	Examples
Lexicon-Based Approaches	Use predefined dictionaries of sentiment-bearing words to determine sentiment scores.	- Easy to implement and interpret - No need for labeled training data - Cost-effective and quick to deploy	- Limited by comprehensiveness of lexicon - Struggle with context, sarcasm, and complex linguistic structures - May miss nuanced expressions and domain-specific jargon	VADER (Habimana et al., 2019)
Traditional Machine Learning	Obtain features in text data and apply in SVM, Naive Bayes, and random forests to	- Can handle diverse feature sets - Perform well with well-labeled datasets	- Require extensive feature engineering - Computationally intensive - May not generalize well to new data	SVM, Naive Bayes, Random Forest (Aziz & Starkey, 2020)

	classify sentiment.	- Improve accuracy over lexicon-based methods	- Require continuous updates	
Deep Learning	Use neural networks like LSTM, CNN, and transformers to learn representations of text data.	- Automatically learn intricate features - High accuracy and robustness - Capture contextual and sequential nature of language	- Require large, labeled datasets - Computationally intensive - Less interpretable - Operate as black boxes	LSTM, CNN, Transformers (BERT) (Yuan et al., 2020)
Hybrid Approaches	Combine lexicon-based, with traditional, deep learning methods.	- Leverage strengths of multiple methods - Higher accuracy by integrating multiple information sources - Effective in complex domains	- Increase complexity - Resource-intensive to develop and maintain - Require extensive experimentation to find optimal combination - Ensure compatibility and synergistic performance	Integrat3 lexicons and deep learning (Mishev et al., 2020)

Table 1 shows the summary of the 4 approaches that were discussed.

2.2 Applications and Sentiments in Financial News

2.2.1 Application of Sentiment Analysis for Financial News

Sentiment analysis comprises the computational recognition and classification of sentiments to evaluate a person's mood in specific topics, or events. This project will classify as bullish, bearish and neutral as this analysis is crucial for financial individuals who need fast access to the public sentiment to make informed trading and investment decisions (Mishev et al., 2020).

In addition to that, various sentiment analysis techniques have been implemented and applied in financial news. As an example, lexicon-based uses methods with predefined dictionaries to identify the sentiment-bearing of works, support vector machines (SVM) and random forests are trained on labelled data which is sentiment-based. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) recognise arrangements of texts (Aziz & Starkey, 2020).

i) Stock Market Analysis:

Sentiment analysis for financial news has often been used in stock markets to predict stock movements by analysing the correlation between sentiment and stock prices. Research has shown that financial sentiments can drastically influence the stock market's behaviour, especially during a financial crisis and pandemic, based on the finding of Valle-Cruz et al. (2021), the research shows a relationship between financial news and public opinion is strong during COVID-19 and H1N1 pandemics. Besides that, López-Cabarcos et al. (2019) mentioned in the research that investor sentiment had drastically affected Bitcoin during the speculative periods. Hence, one can assume that positive financial news regarding stocks can lead to a rise in stock prices, and negative news can cause a decline in the prices. This correlation proves the importance of accuracy in interpreting sentiment analysis for investment decisions.

ii) Cryptocurrency Markets:

Besides that, sentiment analysis is also applied in predicting the ups and downs of cryptocurrency prices in the market. Based on Kraaijeveld and Smedt (2020), sentiment analysis has demonstrated that public sentiments could perform price prediction on price returns for major cryptocurrencies like Ethereum and Bitcoin. This can be proven that sentiment analysis has potential as a predictive tool for digital asset markets. For instance,

during periods of the increase in public interest and positive sentiment in X, cryptocurrencies often experience significant price increases. In contrast, negative sentiment can cause a decline in prices, this shows that public opinions are sensitive in markets (Koltun & Yamshchikov, 2023).

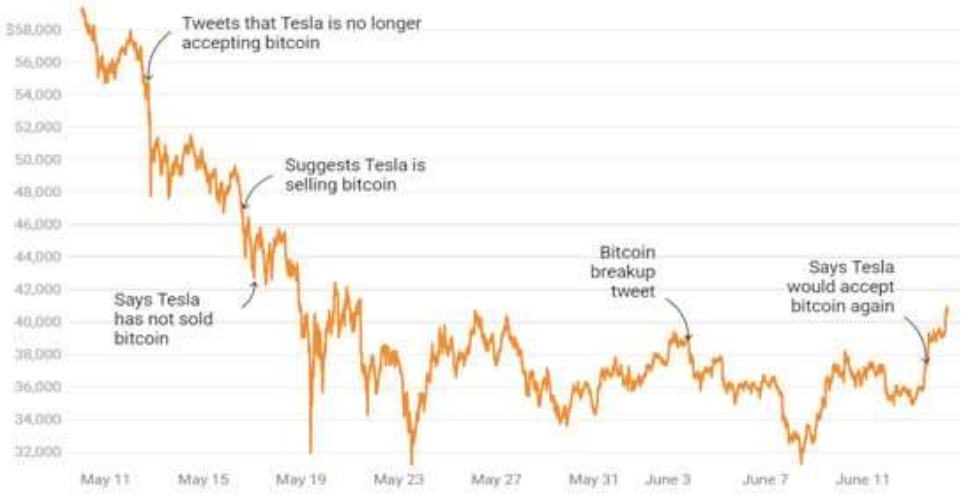


Figure 2 shows the BTC price in May–June 2021 according to Koltun and Yamshchikov (2023) in analysing the sentiment analysis for cryptocurrency.

iii) Financial News and Decision-Making:

Furthermore, financial sentiment analysis can assist professional traders and investors in detecting insights from news events. Based on, Parekh et al. (2022) a hybrid model is proposed to implement sentiment analysis using deep learning for cryptocurrency price prediction and it has shown improvements in accuracy for forecasting. This shows that the combination of different analytical methods can improve the analysis, which allows gain investors trust in analytical tools for decision-making.

However, one may argue that by integrating sentiment analysis into financial decision-making process may improve the capabilities of predicting the market's movements and adjust their investment strategies according to the situation. As an example, automated trading systems with sentiment analysis can respond more accurately to finance news events and potentially take advantage of on market inefficiencies before human traders can react (Ren, Wu, & Liu, 2019).

2.2.2 Bullish, Bearish, and Neutral Sentiments in Financial News

By understanding the sentiment in financial news, one can predict the market movements and the behavior of the market. Sentiments can be used to identify the bullish, bearish and neutral, each indicating different outlooks in finance:

i) Bullish Sentiment:

Bullish refers to a positive outlook where investors believed that there will be an increase in value for the market or product. X's data with bullish sentiment often shows positive language about how the future market is performing. This affects the decisions of other investors in buying or selling the asset. Chang et al. (2021) discussed that bullish sentiment in financial communities can drastically impact the stock market returns and volatility. During an announcement of positive earnings in the report, the sentiment tends to be bullish with causes the prices to spike up and investors are optimistic about it.

Figure 3 shows a source from CoinMartketCap in Betashares on how bullish sentiment is spotted in bitcoin ETF (Arzadon, 2023).

ii) Bearish Sentiment:

Bearish sentiment often signifies as a negative outlook where investors anticipate a crash in the market as the asset declines in value. The bearish news often produces pessimistic views on financial news which affects the investors to sell their assets. According to Xu et al. (2020), bearish sentiments will affect the market's volatility and returns specifically during periods of unstable economics. One can assume that, news of potential taking legal action against a company may cause a decline in value which declines the stock prices.

Figure 4 shows a source from Bloomberg that shows how bearish sentiment in the market looks based on a stock market stimulus (Bloomberg - Are You a Robot?, 2023).

iii) Neutral Sentiment:

The neutral sentiment shows a lack of strong conviction either way, this defines that the investors are unsure about the market's stability or direction. Neutral news usually shows a sign of balance and low volatility, neutral sentiment shows the importance of financial equilibrium. Bashir et al. (2021) discovered that financial news which express neutral sentiment will shape a more balanced perspective in the finance sentiment. Hence, neutral sentiments are often observed when the market is

consolidated, it is when neither positive nor negative news dominates, this brings a stable price.

In summary, sentiment analysis is important in financial news to provide insights into the public opinion and affect the market dynamics. By enhancing the analytical methods, one can improve their decision-making process and respond more efficiently to the fluctuations.

2.3 Challenges of Sentiment Analysis in Financial News

Although sentiment is an effective method to study the public sentiment for predicting market's movement, it faces several challenges when it is applied to analyze financial news. These challenges arise from the inherent complexity of financial language and the dynamic nature of the markets and limitations of analytical methodologies.

i) The complexity of Financial Languages

Financial news usually consists of jargon, idiomatic expressions and trending terms is difficult to interpret precisely. As an example, "bullish" and "bearish" may not be understood by models that are trained on general texts as these two terms are financially specific. Besides that, financial news frequently employs subtle financial language to express sentiments that can lead to misclassification by the algorithms of sentiment analysis. Based on Dong and Liu (2021) the complicated financial language is a challenge for sentiment analysis as the model must be able to analyse these terms in the text to provide accurate classification. Hence, one can argue that, to tackle the problems, sentiment analysis models must be incorporated with updated financial field languages for training to enhance the accuracy.

ii) Dynamic Nature in Finance News

The high dynamics in the financial realm can change the sentiments and trends rapidly in response to new information. The dynamics of the financial news makes it challenging for the sentiment analysis models to maintain accurate from time to time. Models which are trained with conventional historical data may struggle to project the future sentiments accurately as they are not updated with the latest trends. Mittal and Goel (2021) emphasized that the rapid change in financial markets require models which are capable to capture the latest trend patterns to maintain the accuracy.

iii) Ambiguity and Sarcasm Among Texts

According to the ambiguity and sarcasm which are observed in financial news can affect the performance of the sentiment analysis models as it causes incorrect sentiment classification, this is due to the sarcastic remarks are mistaken as the opposite. One can said that "The company's report is fabulous!" this may indicate that the person is sarcastically criticising on the poor performance of the company. This explains why Nimala et al. (2020) highlights that accurate interpretation of sarcasm and ambiguous

language is an obstacle in finance sentiment analysis. Therefore, detecting ambiguity and sarcasm is highly crucial, it requires capabilities in advanced natural language processing methods such as embeddings to aid in the model's performance.

iv) Domain-Specific Sentiment Lexicons

Building and maintaining a domain-specific sentiment lexicon is important for improving the prediction of financial news. However, developing can be said to be time-consuming as it requires specialisation in finance and linguistic background. This is due to generic sentiments are not enough to capture the specific financial sentiment and this causes the model to have a less accurate evaluation. The lexicon-based method can advance extensively with a dictionary that is customised for the finance domain. However, the effort put in to collect these resources can be time-consuming. Liu and Zhang (2019) emphasized that the development of domain-specific sentiment lexicon is important, but it is a resource-intensive and difficult task.

v) Interpretability of Machine Learning Models

Financial decision-makers depend on a clear and straightforward insight from the sentiment analysis tool to make effective decisions for the best results. However, advanced machine learning models such as deep learning tend to produce "black boxes" which affects their decision-making process. Consoli et al. (2021) recommended that implementing methods in feature-based sentiment can enhance interpretations. This can be done by applying sentiment scores to specific topics in the text. One can assume that improving the interpretation of these models is important for financial professionals to be confident with the predictions to base on the model. This ensures that sentiment analysis in finance is a practical application.

In a nutshell, although sentiment analysis provides the potential to improve the financial news analysis, it comes with several challenges which must be overcome to enhance the accuracy. These include the complexity of the financial language, the dynamic nature of financial news, the presence of ambiguity and sarcasm in texts, the necessity for domain-specific sentiment lexicons and the interpretability of machine learning models. It is believed that by consistently researching and developing advanced sentiment analysis models, one can overcome the challenges and increase the effectiveness and accuracy in the financial sector.

Chapter 3: Research Methodology

3.1 Introduction

This aim is improving financial insights by using NLP-driven sentiment analysis on X (previously known as Twitter) financial news. The technique described here takes an organised method from data collection to model assessment, resulting in rigorous and complete analysis. The data for this project will be sourced from the Financial Tweets Sentiment dataset on Hugging Face.

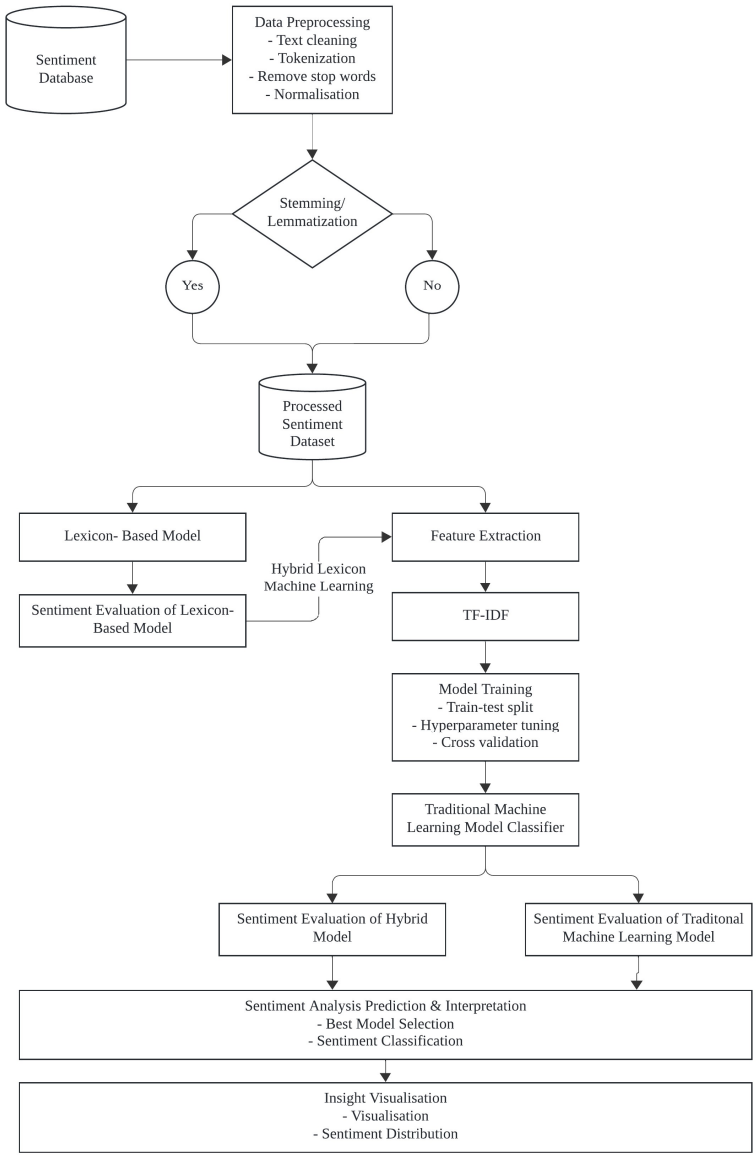


Figure 5 shows the research methodology pipeline used for sentiment analysis

The figure displays how the pipeline of the research methodology looks like for this project from data processing to model training to evaluation to insight visualization. Among the steps taken, there are steps involving text cleaning, stemming or lemmatization, feature extraction to improve the dataset quality for model training. These steps are performed to ensure high accuracy from the analysis.

Data Collection

The dataset chosen is from the Financial Tweets Sentiment dataset on Hugging face. This dataset will be focused on collecting texts with keywords regarding stock tickers, market indicators, and even of finance. By obtaining texts with these specific keywords, it can ensure that the dataset has a wide range of sentiments throughout various financial contexts.

The data from the dataset is taken from 2017 to 2022 from X, regarding financial news. The project will be focusing on the text which is collected which shows characteristics of bullish, bearish and neutral.

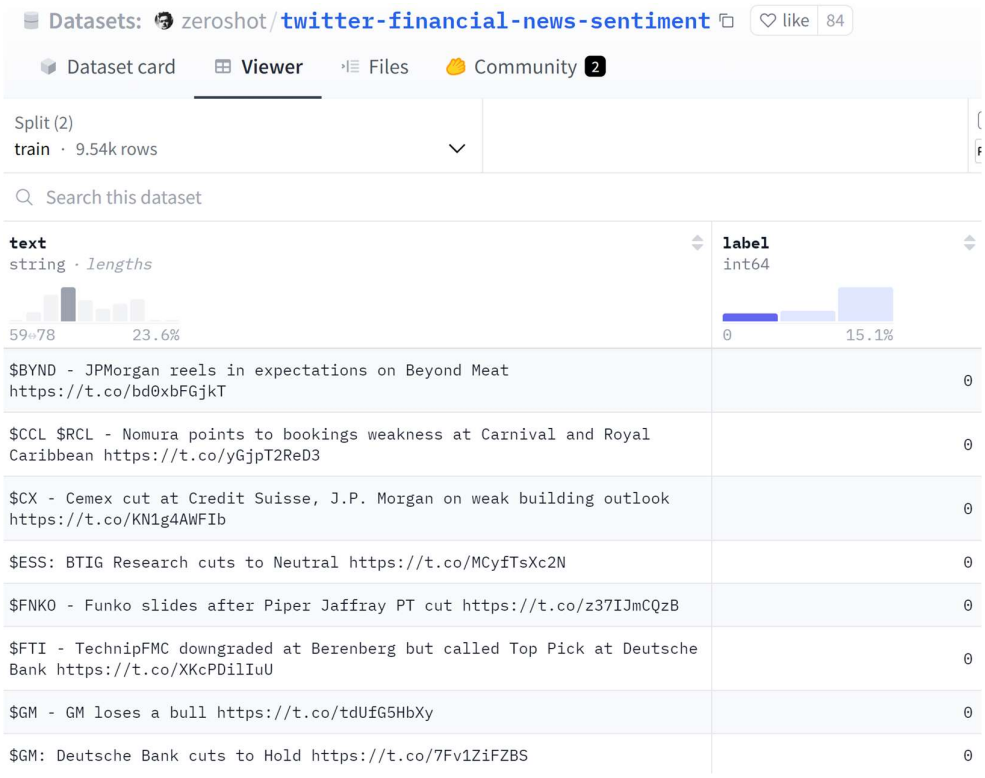


Figure 6 shows the dataset that will be used for sentiment analysis.

3.2 Data Preprocessing

This is an important process for preparing the dataset for analysis. This step involves cleaning and arranging the data to make it more fitting for processing and analysis. In the aspect of text data, preprocessing involves the following steps to ensure that the data is formatted properly for analysis:

i) **Noise Removal:** Noise removal is carried out to remove the redundant elements from the text data that has no aid to the analysis like URLs, mentions, hashtags, and special characters. The reason to perform this is due to these elements might bring unwanted variations and potentially distort the analysis results which affects the accuracy.

ii) **Tokenization:** This process is done to dismantle the text into small units in token form for words, phrases, or symbols. This step is important for further analysis because it supports understanding the structure of the text. Based on the text, it can extract meaningful information (Xue et al., 2022).

iii) **Stop Words Removal:** It is to remove common words, that have minimal meanings, and it is filtered out. By taking out “and” “is”, and “the”, can assist in focusing on the important words in the text (HaCohen-Kerner et al., 2020).

iv) **Stemming or Lemmatization:** The technique of stemming reduces the wordings to root forms like “dropping” to “drop”. Whereas for lemmatization change to words from the original form “studies” to “study”. This process is done to help standardize the words while reducing the variations to perform better analysis (Jabbar et al., 2020).

v) **Normalization:** Normalization is the step for converting all the text into lowercase, this is carried out to ensure that the texts are all uniform and consistent in the data. This step aids in treating similar words as identical and avoiding duplication based on the case sensitivity (Chouigui et al., 2021).

By implementing these steps for preprocessing, the text data will be cleaned, standardized, and optimized for further analysis. As a result, the data quality is improved, the noise is reduced and therefore it can be said that higher accuracy insights will be obtained for this sentiment analysis. As an example, converting “Stocks are growing” to “stock grow” can help with extracting the important sentiment information without adding redundant details.

3.3 Feature Extraction

This is performed to transform preprocessed texts into structures for training in machine learning models. Words are measured with Term Frequency-Inverse Document Frequency (TF-IDF) based on the word frequency and the number of areas in finance this word appeared in (Gite et al., 2023).

$$\mathrm{TF}(i, j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$\mathrm{IDF}(i) = \log \frac{\text{Total documents}}{\text{documents with term i}}$$

$$\mathrm{TF} - \mathrm{IDF} = \mathrm{TF} \times \mathrm{IDF}$$

Figure 7 shows the formula for TF-IDF (Sham & Mohamed, 2022).

Predefined word embeddings (GloVe and Word2Vec) are used to understand and acquire the semantic meanings and connections between the words. These methods are applied to demonstrate a representation of the text and enhance the accuracy. As an example, through using TF-IDF, the word “market” in financial news contexts can be weighted higher than “good”, this is due to the significance in the financial world of X (Alimova & Tutubalina, 2020).

3.4 Model Selection

Various models are evaluated by determining the most effective model for sentiment analysis. Lexicon-based models, for example, VADER are used as predefined dictionaries to assign sentiment scores (Borg & Boldt, 2020). VADER as a lexicon-model analyses each word in a text and assigns scores, then normalizes the sum to a smaller sum. For sentiment classification, a value limit 0.3 is applied instead of 0.05 by Sham and Mohamed (2022), as it is believed to provide the highest accuracy based on their report.

$$\text{Positive sentiment} : \text{compound}_{\text{score}} \geq 0.3$$

$$\text{Negative sentiment} : \text{compound}_{\text{score}} \leq -0.3$$

$$\text{Neutral sentiment} : \text{compound}_{\text{score}} < -0.3 \ \& \ \text{compound}_{\text{score}} > 0.3$$

Figure 8 shows the calculation for VADER (Sham & Mohamed, 2022).

Traditional machine learning models are used to categorise the sentiments based on the learning patterns obtained (Khiabani et al., 2019). Whereas deep learning models Long Short-Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) are used to obtain the complicated patterns in the data (Mishev et al., 2020).

As for hybrid models, it combines the features of the above approaches which are mentioned to increase the strength and reduce the weaknesses of the model. For instance, a hybrid model might consist of applying a lexicon-based approach to manage specific financial terms and use deep learning to leverage and capture the wide range of contextual meanings (Sham & Mohamed, 2022).

Model Type	Description	Citation
Lexicon-Based (VADER)	Uses a predefined dictionary for assigning sentiment scores	(Al-Abyadh et al., 2022)
Traditional ML (SVM, Naive Bayes)	Applies classification sentiments based on the learning patterns.	(Alqaryouti et al., 2020)
Deep Learning (LSTM, BERT)	Captures complex patterns in the data	(Kraaijeveld & Smedt, 2020)

Model Type	Description	Citation
Hybrid Models	Combination of lexicon-based and deep learning models to leverage strengths and mitigate weaknesses	(Mohamad Sham & Mohamed, 2022)

Table 2 shows the models which are selected for this project.

3.5 Model Training

During the model training phase, the prepared dataset is optimized for applying in selected models. This is performed to avoid overfitting or biases, this ensures that the machine learning model can accurately analyze the new data (Sevetlidis et al., 2022).

Train-test split will be done to split the data into a test and training set for performance evalaution, common 80/20 ratio or 70/30 it will be adjusted during the training to ensure the model performs well.

During the training phrase, the hyperparameters of the models will undergo tuning through cross-validation. Cross-validation splits the training set into subsets, then the model will be evaluated through the performance to find the most suitable hyperparameters. For instance, in an LSTM model, the number of layers can be modified, whereas in an SVM model, the regularization parameter can be fine-tuned to enhance accuracy (An et al., 2021).

Through the process of cross-validation, the models can obtain improved accuracy and enhanced ability to generalize to unfamiliar data by carefully adjusting hyperparameters. Ensuring the trained models function well in real-world applications and can accurately forecast fresh data is a vital step (Mohr & Van Rijn, 2023).

3.6 Model Evaluation

Evaluation is done to measure the machine learning models in their effectiveness in performing. The metrics are applied to analyze the model's performance and identify the areas which need to be improved and enhanced.

i) **Accuracy**:
Accuracy measures the percentage of correct occurrences out of the overall. However, accuracy alone is insufficient for datasets that are imbalanced when one trait is stronger than others (Hammad & El-Sankary, 2019).

ii) **Precision**:
This metric evaluates the accuracy of positives out of the total of predicted positives. It highlights the precision of positive predictions (Bustillo et al., 2020).

iii) **Recall** (Sensitivity):
Recall calculates the part of correct positive out from the origical class. It focuses on the model's capability to accurately recognize all appropriate instances (Li et al., 2019).

iv) **F1-Score**:
These measurements average precision and recall as it equalizes between these two metrics, by processing both the incorrect positive results and the incorrect negative results (Mortaz, 2020).

v) **ROC-AUC** (Receiver Operating Characteristic - Area Under the Curve):
This metric measures performance for categorisation problems at different thresholds. ROC-AUC displays the compromise among true positive rate and false positive rate (Kang et al., 2020).

vi) **Confusion Matrix**:
The metric measures the classification model performance. It provides the amount of predictions in true positive, true negative, false positive and false negative which are classified by classes, this assists in discovering where the model is making errors (Sidey-Gibbons & Sidey-Gibbons, 2019).

By acquiring these metrics and analyzing the confusion matrix, one can gain deeper insights on how the performance of the model. As an example, a high precision but low recall could be a sign that the model can identify positive cases, however it might miss the actual

positives. This observation shows that there is a need for further adjustments to improve overall model performance to obtain a better equilibrium between precision and recall.

After understanding the basic metrics, it is believed that these matrices might provide very generalized evaluations. Therefore, further exploration is done to understand advanced calculations. The chosen metrics will be used in Chapter 4 for results and evaluation.

i) **Matthews Correlation Coefficient (MCC):**
MCC evaluates the value of all the confusion matrices which are true positives, true negatives, false positives, and false negatives to provide a balance measure to all the matrixes (Chicco et al., 2021). Although MCC is used for dual classification, it can be modified for handling multi-class classification for implementation in the sentiment analysis dataset.

ii) Precision-Recall AUC:
Precision-Recall AUC is a plot between precision and recall for different thresholds, this allows the single metric which captures the trade-off to focus on the performance of the positive class (Miao & Zhu, 2021). Compared with ROC-AC it is more useful in the case of imbalanced datasets as it focuses on detecting the accuracy of the positive class which is beneficial for detecting bullish and bearish sentiments. Therefore, it may help in providing a clearer picture of the model's effectiveness for detecting true positives without being affected by the large number of true negatives.

iii) **Macro-Averaged F1-Score:**
Macro-Averaged F1-Scmeaningevaluates the F1-score separately from every class and then calculates the mean. This treats each classification equally regardless of the frequency level, which is important for a balanced performance across the classes in a dataset. It demonstrates a comprehensive measure of the model's precision and recall across the sentiment classes which is useful for overall model performance evaluation (Yilmaz et al., 2020).

3.7 Sentiment Analysis Prediction & Interpretation

This process utilises a high-performing sentiment analysis algorithm to categorise the texts into bullish, bearish, or neutral. This shows a crucial view regarding market opinion (Xing et al., 2019). By aggregating and evaluating these predictions, one can get a deeper understanding regarding the financial market sentiment patterns (Mishev et al., 2020). One can said that this can be fundamental in creating well-informed investing choices.

The recognition of increase in bearish texts during a company's results report as a possible sign of market declines it can be a potential indicator of market selloffs. This demonstrates a of sentiment analysis for predicting the market trends (Picasso et al., 2019).

This insight demonstrates the ability of the application of sentiment analysis in prediction for early signs of market shifts which allows investors to anticipate and may profit from the market's fluctuations (Xu et al., 2019).

By utilising the sentiment analysis, one can obtain useful insights from X's data which enables them to anticipate market trends and make data-driven decisions. This ability helps to analyse the market sentiment trends through earning announcement and based on the sentiment analysis, this allows one to adjust their market or investment strategies (Aziz & Starkey, 2020).

3.8 Insight Visualisation

Visualisation tools such as bar charts are used to display the sentiment analysis trend from the financial market. One can assume that this is an impactful technique for showing complicated information in a visual friendly and understandable manner (Tuarob et al., 2021). Through visuals, investors can easily understand the market dynamics based on the sentiment data and make investment decisions. As an example, bar charts are believed to be a straightforward method to show sentiment trends and patterns. The stakeholders can easily identify the outliers, correlations and trends regarding the market (Chen et al., 2020). Therefore, visualisation gives a complete overview of how sentiment progresses when used in various market events (Zucco et al., 2019).

By implementing visualisation tools into sentiment analysis, it improves the interpretability of the data and provide more efficient communication of insights to more people (Correia et al., 2022). It is safe to say that sentiment trends help in conveying complicated data into more impactful information which empowers investors and stakeholders to make data-base decisions base on the dynamics of the market sentiment (Ren et al., 2019).

In summary, this methodology implements advanced NL techniques for data analysis and visualisation to provide a comprehensive approach for analysing and understanding the financial sentiment on X. By leveraging the model through preprocessing, feature extraction methods. This allows the study to be delivered in a more accurate manner and provides data-driven insights for the financial world.

3.9 Prototype/Dashboard Proposal

A dashboard will be designed for the sentiment analysis model. The dashboard will show a section displaying the text from X, the customer ID column will be changed to ID numbers based on the number of API obtained and the project name would be removed, and comments would be the text which are used in sentiment analysis.

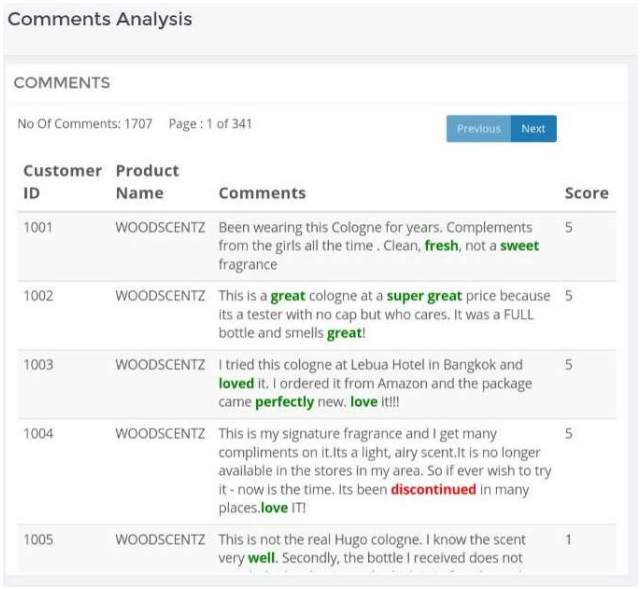


Figure 9 shows the analysis of the text.

Chapter 4: Results & Evaluation

This section will be discussing the implementation based on chapter 3 to analyse the data obtained from online resources which is huggingface. After that, the dataset will be handled before proceeding to modelling. Once the model is executed, it will be evaluated based on the results obtained and the highest performing model will be used for implementation in GUI.

4.1 Analysing and Preprocessing Data

4.4.1 Data Visualization

The dataset obtained has been loaded into the environment and then it is analysed through visualization in order to understand the structure. In order obtain a more reliable statistic, the URLS, mentions, hashtags and special characters are removed before visualized.

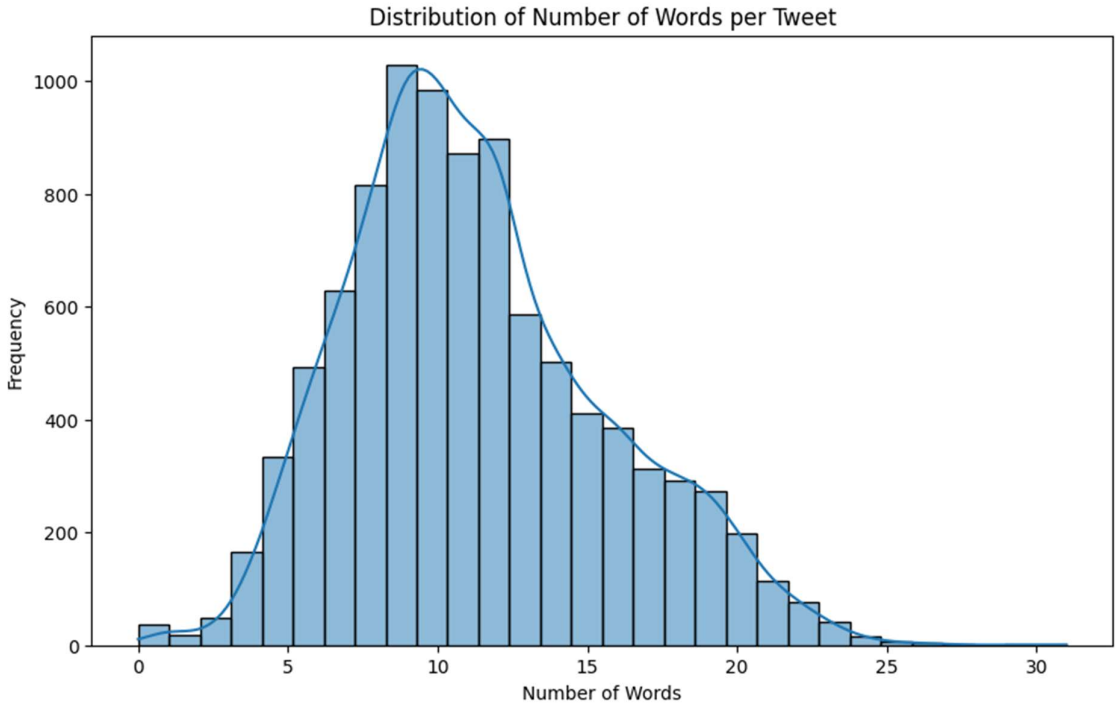


Figure 10 shows the chart for distribution of number of words per tweet.

4.1.2 Handling Missing Data

The next step is to check for the missing values in the dataset. The code prints the shape and features of the dataset to understand the structure. It is observed that there are no missing data in this collection of data.

```
# Check for missing values in the original dataset
print(dataset['train'].shape)
print(dataset['train'].features)

# Check for missing values in the DataFrame
print(dataset_df.isnull().sum())
```

```
(9543, 4)
{'text': Value(dtype='string', id=None), 'label': Value(dtype='int64', id=None), 'cleaned_tweet': Value(dtyp
text            0
```

Figure 11 shows the snippet for checking missing values.

4.1.3 FinBERT Initialization

FinBERT is initialized here for the pipeline and tokenizer. FinBERT is a language model that is pretrained specifically for financial sentiment analysis. It is created based on BERT architecture to understand and analyse financial linguistics.

```
# Initialize the FinBERT pipeline and tokenizer
pipe = pipeline("text-classification", model="ProsusAI/finbert")
tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
```

```
config.json: 100% 758/758 [00:00<00:00, 16.0kB/s]
pytorch_model.bin: 100% 438M/438M [00:04<00:00, 111MB/s]
tokenizer_config.json: 100% 252/252 [00:00<00:00, 3.50kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 3.58MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 2.90kB/s]
```

Figure 12 shows the initialization of FinBERT from ProsusAI.

4.1.4 Data PreProcessing

```
# Get the list of stop words
stop_words = set(stopwords.words('english'))

# Initialize lemmatizer
lemmatizer = WordNetLemmatizer()

# Function to preprocess text with lemmatization and normalization
def preprocess_text(text):
    text = text.lower()  # Normalize text to lowercase
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # Remove URLs
    text = re.sub(r'\@\w+|\#', '', text)  # Remove mentions and hashtags
    text = re.sub(r'[^A-Za-z0-9 ]+', '', text)  # Remove special characters
    words = text.split()
    words = [word for word in words if word not in stop_words]  # Remove stop words
    words = [lemmatizer.lemmatize(word) for word in words]  # Apply lemmatization
    return ' '.join(words)

# Add POS tagging
def pos_tagging(text):
    tokens = word_tokenize(text)
    return pos_tag(tokens)

dataset = dataset.map(lambda x: {"pos_tags": pos_tagging(x["cleaned_tweet"])})
```

Figure 13 shows the snippet of the steps in data preprocessing.

The dataset is the preprocessed with the following steps. A set of English stop words are obtained from the NTLK which is installed earlier in the library and downloads section is initialized here with the WordNetLemmatizer. Next, on the preprocessed text, the steps taken are as follows, the text is normalized to convert them into lowercase. Next, the URLs, mentions, hashtags and the special characters are removed. After that, the text is split and the stop words are removed from the texts, the remaining words are lemmatized.

POS tagging is applied here to tokenize the text and apply the part-of-speech (POS), it is done by mapping the dataset. This is to confirm the text data are all normalized, tagged, and cleaned.

```python
# Function to classify sentiment with adjustable thresholds
def classify_tweet_with_thresholds(tweet, bearish_threshold=0.4, bullish_threshold=0.6):
    results = pipe(tweet)
    for result in results:
        label = result['label']
        score = result['score']
        print(f"Debug: Label = {label}, Score = {score}")  # This is for debug

    # Use the highest scoring label
    top_result = results[0]
    label = top_result['label']
    score = top_result['score']

    # Check thresholds
    if label == 'negative':
        if score >= bearish_threshold:
            return 'Bearish'
    elif label == 'positive':
        if score >= bullish_threshold:
            return 'Bullish'
    return 'Neutral'
```

Figure 14 shows the steps to adjust the thresholds in the code.

After preprocessing, threshold function is done to classify the sentiments of the text data using the adjustable bullish and bearish thresholds. The text data is processed through a pipeline to show the sentiment labels and scores. The debugging results will be shown with the scores. The thresholds classifications are as follows which are negative is the bearish, positive is the bullish and neutral is neutral.

4.1.5 Analyze the sentiments with thresholds

```python
def analyze_sentiment_with_thresholds(example):
    preprocessed_text = preprocess_text_with_lemmatization(example["text"])
    sentiment = classify_tweet_with_thresholds(preprocessed_text)
    example["predicted_sentiment"] = sentiment
    return example
```

Figure 15 analyses the sentiments that contain thresholds.

After the text data is preprocessed, the next step is to classify the text data by analyzing it based on the thresholds implemented for sentiment classification. Then, the predicted sentiment is added to the example dictionary. This is done to ensure the structure is consistent for the sentiment labels with the given text data.

4.1.6 Preparing the Dataset

```
# Define the number of samples you want to keep
num_samples = 3000

# Function to get a random subset of the dataset
def get_random_subset(dataset, num_samples):
    import random
    indices = list(range(len(dataset)))
    random.shuffle(indices)
    selected_indices = indices[:num_samples]
    return dataset.select(selected_indices)

# Split the dataset into train and test
dataset = dataset['train'].train_test_split(test_size=0.2, seed=42)

# Get random subsets for train and test
train_subset = get_random_subset(dataset['train'], num_samples)
test_subset = get_random_subset(dataset['test'], int(num_samples * 0.2))

# Create a DatasetDict with the subsets
dataset_dict_subset = DatasetDict({
    'train': train_subset,
    'test': test_subset
})
```

Figure 16 shows the step taken to split the dataset for train test.

Then, 3000 sample size is determine as a subset for creating random subsets, it is used in the splitting of the data into the train and test sets. A random shuffle function is called to shuffle the dataset. The test set is set to 20% of the dataset and the data is split into train test sets. This is done to ensure the subsets are balanced and the model can be trained effectively and then the subsets are organized into a DatasetDict for structured handling.

4.1.7 Applying Sentiment Analysis

```
# Start timing for the sentiment analysis step
start_time = time.time()

# Apply the updated function to the dataset with thresholds
dataset_lemmatized_with_thresholds_subset = dataset_dict_subset.map(analyze_sentiment_with_thres

# End timing for the sentiment analysis step
end_time = time.time()
print(f"Time taken for sentiment analysis: {end_time - start_time} seconds")

# Display some results
print(dataset_lemmatized_with_thresholds_subset["train"][0])
```

Show hidden output

```
# Check the distribution of predicted sentiments
predicted_sentiments = dataset_lemmatized_with_thresholds_subset["train"]["predicted_sentiment"]
unique_sentiments, sentiment_counts = np.unique(predicted_sentiments, return_counts=True)
print(f"Predicted sentiment distribution: {dict(zip(unique_sentiments, sentiment_counts))}")
```

```
Predicted sentiment distribution: {'Bearish': 576, 'Bullish': 249, 'Neutral': 2175}
```

Figure 17 shows the steps to obtain the distribution of the sentiment.

Once the dataset is prepared, the start and end of the sentiment analysis with thresholds are done for timing the process and showing the results. Then the distribution of the predicted sentiments are checked to understand the composition of the dataset based on the three sentiments. It is crucial to analyze the distributions to obtain insights of the overall sentiment trend of the dataset for evaluating the performance.

4.1.8 Extracting data and encoding

```
# Extract text and labels from the dataset
df_train = pd.DataFrame(dataset_lemmatized_with_thresholds_subset["train"])
corpus = df_train["text"]
labels = df_train["predicted_sentiment"]

# Encode labels to numerical values
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(labels)

# Check the class distribution before splitting
unique_classes, class_counts = np.unique(y, return_counts=True)
print(f"Class distribution before splitting: {dict(zip(unique_classes, class_counts))}")
```

```
Class distribution before splitting: {0: 576, 1: 249, 2: 2175}
```

Figure 18 shows data extraction and encoding.

The dataset is then extracted based on the text and the predicted sentiment and the text data is encoded using numerical values for the labels. The distribution of the classification is checked again. It is done to verify that the encoding has been integrated to the sentiment distribution.

4.1.9 Splitting the data

```
# Initialize the TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(corpus)

# Split the data into training and testing sets using stratified split
train_indices, test_indices, y_train, y_test = train_test_split(df_train.index, y, test_size=0.2, random_st
X_train = X_tfidf[train_indices]
X_test = X_tfidf[test_indices]

# Check the distribution of classes in y_train and y_test
print("Training set class distribution:", np.bincount(y_train))
print("Test set class distribution:", np.bincount(y_test))
```

```
Training set class distribution: [ 461  199 1740]
Test set class distribution: [115  50 435]
```

Figure 19 shows the splitting of data after it is vectorised by TF-IDF.

TF-IDF Vectorizer is initialized here to transform the text data into numerical features. Then the next step is to split the data into two parts which is the training and testing sets. The training and testing set class distribution is then checked if it is balanced and not bias. The results is then printed out to verify before the next step.

4.1.10 Example Text Classification

```
# Example tweet to classify
tweet = "The company's lack of innovation will lead to stagnation and declining market share.."

# Preprocess the tweet
preprocessed_tweet = preprocess_text_with_lemmatization(tweet)

# Classify the tweet
sentiment = classify_tweet_with_thresholds(preprocessed_tweet)

# Display the classification result
print(f'The tweet is classified as: {sentiment}')
```

```
Debug: Label = negative, Score = 0.9650284051895142
The tweet is classified as: Bearish
```

Figure 20 shows the text classifications results.

This step is carried out after the data is well handled to check if the sentiments are correctly labelled before applying it into modelling. It is observed here that the classification can correctly identify a bearish sentiment.

4.2 Model Training

The models used here are based on the proposed models which are discussed in Chapter 3 research methodology. These models are SVM, LSTM, FinBERT, VADER, and Hybrid Model which is a combination of SVM and FinBERT. The features and embeddings are also referring to the chapter 3 research methodology as well.

4.2.1 SVM Model

The SVM model is embedded with the following features which are the GloVe, TF-IDF, and hyperparameter tuning in order to perform comparison. The highest-performance SVM will be selected to combine with FinBERT as a hybrid model. For the SVM that uses TF-IDF, it is executed by implementing SVM with TF-IDF in model training. For the GloVe, the GloVe Embeddings are loaded with the pre-trained vectors, and it is aggregated. Then for hyperparameter tuning, GridSearchCV is used to adjust the parameters through parameter grid with C, gamma and kernel. The model is then split into training and testing sets for model training.

4.2.2 FinBERT Model

The FinBERT model is sourced from Hugging Face and developed by ProsusAI, it is a pre-trained NLP model which is fine-tuned for the financial sentiment analysis (ProsusAI/Finbert · Hugging Face, n.d.), therefore it is chosen for this project. The FinBERT model initializes the FinBERT tokenizer for modelling. The text data is preprocessed through FinBERT's tokenisation and the subsample of the dataset is taken out to perform the train test set for model training.

4.2.3 Hybrid Model

The proposed hybrid model through research combines the best performance models which are SVM's highest-performance model with FinBERT model, it is then split for training and testing to perform model training. In this study, the one chosen will be the tuned SVM combne with the FinBERT model to become the hybrid model.

4.2.4 LSTM

The LSTM model is done by tokenizing the text and then applying padding and encoding the labels to become numerical values. Then, the parameters are implemented in order to allow LSTM to improve performance. Next, epoch and batch sizes are defined here, and early stopping is used to prevent overfitting.

4.2.5 VADER

VADER is a lexicon-based approach that is tuned for social media sentiments (Borg & Boldt, 2020). The classification of VADER is applied to the model and the classification labels, bullish, bearish and neutral are mapped and labelled numerically. The dataset is split into train and label for modelling. A confusion matrix is done here to study the sentiment analysis of VADER based on financial sentiment classifications.

4.3 Results & Evaluation of Models

A variety of evaluation metrics are used here to evaluate the models. The metrics used for SVM, FinBERT, LSTM, VADER and hybrid model are accuracy, precision, recall, and F1 score. The SVM models are also compared with the ROC curves to study the false positive and true positive rates of the SVM models. This evaluation will indicate the model's ability in differentiating among the classes.

4.3.1 SVM Evaluation

The figures below show the ROC curve for the SVM models.

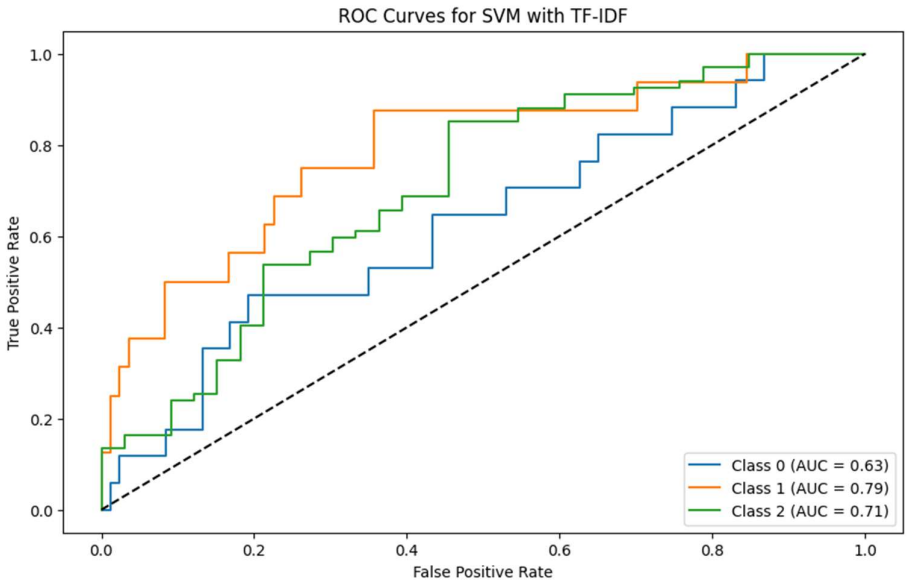


Figure 21 shows the ROC Curve for SVM with TF-IDF.

The SVM model that uses TF-IDF shows an AUC for Class 0 is 0.63, for Class 1 is 0.79, and for Class 2 is 0.71. The best performance based on this visual would be Class 1.

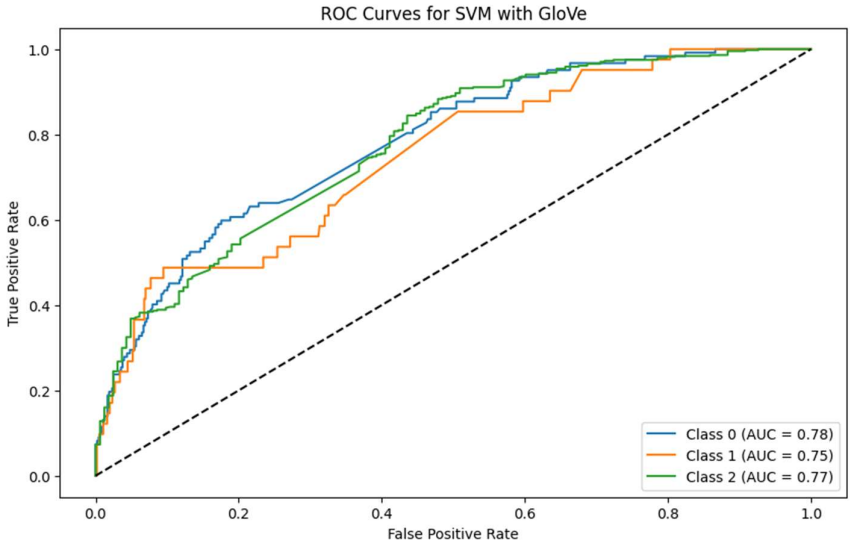


Figure 22 shows the ROC Curve for SVM with GloVe.

The SVM model using GloVe embeddings shows an AUC of Class 0 is 0.78, for Class 1 is 0.75, and for Class 2 is 0.77. This model shows improved performance for Class 0 and Class 2 compared to the tuned SVM without GloVe embeddings. The AUC value here suggested that GloVe embeddings improves the model's ability in classifying the sentiments.

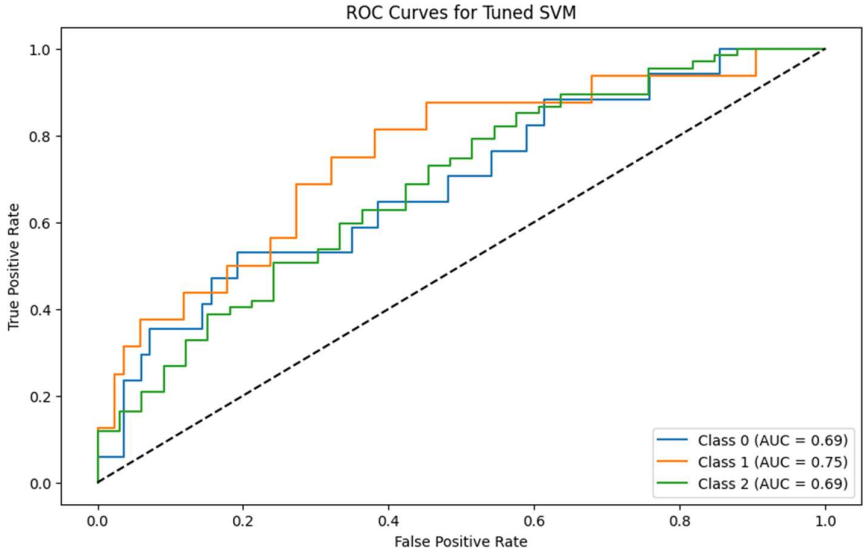


Figure 23 shows the ROC Curve for tuned SVM.

The figure shows ROC curve for the tuned SVM model. The AUC for Class 0 is 0.69, for Class 1 is 0.75, and for Class 2 is 0.69. This shows Class 1 is performing better than Class 0 and 2.

Model	Class 0 (AUC)	Class 1 (AUC)	Class 2 (AUC)
Tuned SVM	0.69	0.75	0.69
SVM with GloVe	0.78	0.75	0.77
SVM with TF-IDF	0.63	0.79	0.71

Table 3 shows the results of the AUC classes of the SVM models.

Based on the comparison of AUC of all 3 SVM models, it is observed that SVM with GloVe embeddings performs the best. As for the accuracy, precision, recall and F1 score, the following table shows the results. However, in this table, the accuracy of SVM with GloVe (0.7483) is the best, the precision, recall and F1 score, Tuned SVM performs the best.

SVM with	Accuracy	Precision	Recall	F1 Score
GloVe	0.7483	0.4811	0.3779	0.3653
TF-IDF	0.6800	0.5589	0.3542	0.3083
Tuned SVM	0.6900	0.7360	0.4372	0.4345

Table 4 shows the evaluation results of the SVM models.

Based on the results, GloVe is best in distinguishing capabilities based on AUC, while for exact classifications, Tuned SVM performs the best. As the SVM will be combined with FinBERT, FinBERT is evaluated with accuracy, precision, recall and F1. Hence, tuned SVM is selected for the hybrid model.

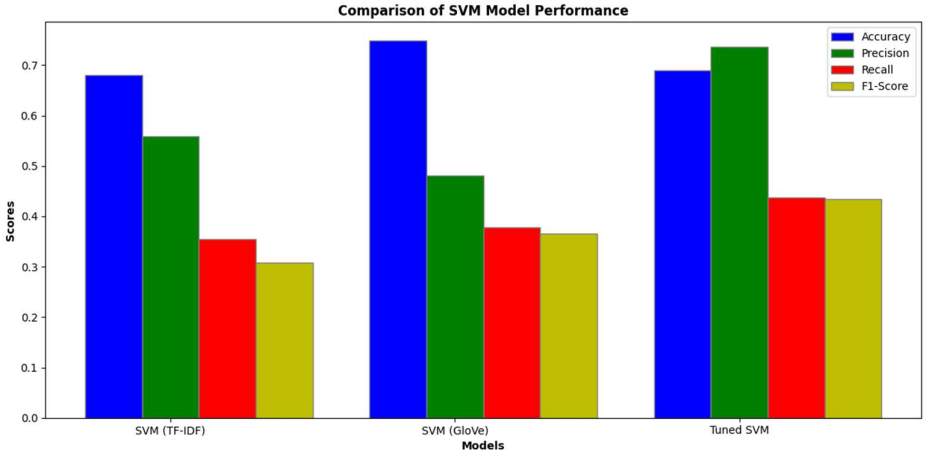


Figure 24 shows the comparison chart of the SVM models.

4.3.2 SVM, FinBERT, LSTM, Hybrid Model and VADER Evaluation

Model	Accuracy	Precision	Recall	F1 Score	MCC
SVM	0.69	0.7360	0.4372	0.4345	0.458
FinBERT	0.71	0.5283	0.5010	0.5063	0.427
LSTM	0.7197	0.4939	0.3794	0.3607	0.266
Hybrid Model (SVM+FinBERT)	0.7760	0.6871	0.5796	0.6162	0.454
VADER	0.4813	0.4370	0.4725	0.4382	0.149

Table 5 shows the evaluation results for the models.

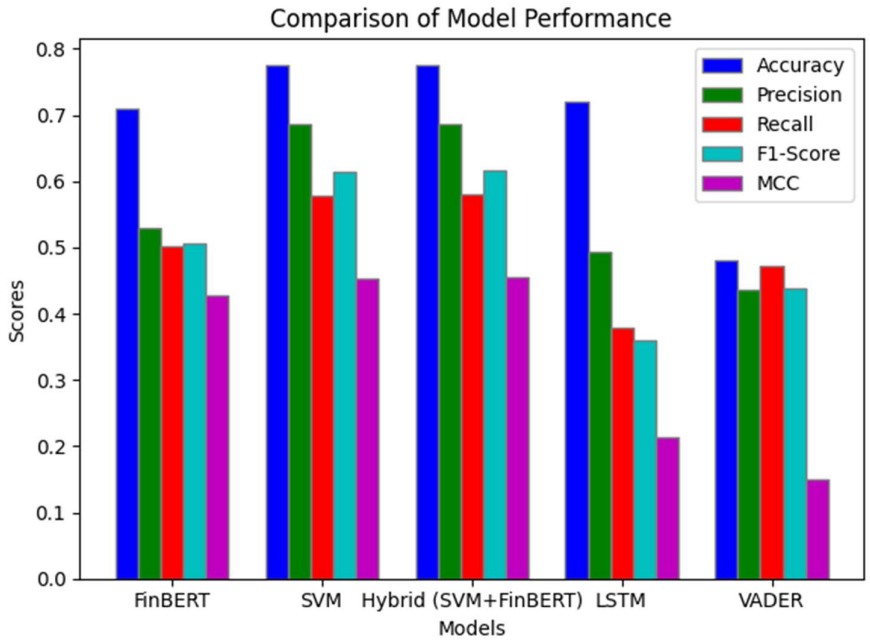


Figure 25 shows the chart of the comparison of the models.

It is observed that MCC is does not add value to the studies. MCC was initially implemented to be used due to the concern of imbalanced data, however the data used in this study is balanced. Therefore, MCC will be removed from the evaluation and the attention will be focused on the other four metrices instead.

Model	Accuracy	Precision	Recall	F1 Score
SVM	0.69	0.7360	0.4372	0.4345
FinBERT	0.71	0.5283	0.5010	0.5063
LSTM	0.7197	0.4939	0.3794	0.3607
Hybrid Model (SVM+FinBERT)	0.7760	0.6871	0.5796	0.6162
VADER	0.4813	0.4370	0.4725	0.4382

Table 6 shows the metrics results without MCC.

Based on the metrics evaluation table, it is observed that the performance results of the Hybrid Model which is SVM+FinBERT outperformed SVM, FinBERT, LSTM, and VADER models in accuracy, precision, recall, and F1 score. It can be said that hybrid models which combines models will enhance the sentiment performance.

The accuracy of the Hybrid Model is 0.7760. It is higher than that of the SVM's accuracy of 0.69, FinBERT's accuracy at 0.71, LSTM's accuracy 0.7197, and VADER accuracy 0.4813. Besides that, the precision, recall, and F1 score for the Hybrid Model show improvement over the other models. The particular one is the precision of the Hybrid Model is 0.6871, it outperforms FinBERT which has precision of 0.5283 and LSTM's precision at 0.4939. The Hybrid Model's recall is 0.5796 is highest compared to the other models. Therefore, the ranking of the models is as the follows: Hybrid Model (SVM+FinBERT) > FinBERT > SVM > LSTM > VADER.

4.4 Discussion

Based on this project, the analysis and evaluation made towards the various models has shown significant insights in their performance and suitability for this study. As mentioned in this paper, Hybrid model which combines SVM and FinBERT has outperformed the other singular models such as SVM, FinBERT, LSTM, and VADER in all the metrices used. The hybrid model has achieved accuracy of 0.77 in classifying the financial sentiments. However, a discussion is needed to talk about the process of obtaining these results.

During the modelling stage it was found out that some of the models require a long time from more than 30 minutes to hours to process and train the data and it requires high computational resources to run it. Therefore, the dataset had to be reduced into subsets in order to run in lesser time to obtain the results. It is time consuming to adjust the hyperparameter tuning for the models that required it as tuning will take time to wait for the model to run and show the results, often the results in the early attempts are not so useful.

SVM itself performed very weak compared to the SVMs that added embeddings and feature into the model. SVM requires a lot of effort in experimenting as various ways are tested to find which method of tuning performs the best and is efficient. In the end, GridSearch was implemented to tune the SVM model and it requires high computational resources to run it. Besides that, it was found out that LSTM requires a lot of time to process and run the training as it is a recurrent neural network (RNN). The epochs took a long time due to computational complexity when it processes the dataset as the training depends on the complexity of the model.

Furthermore, it was found out that VADER although it is a NLP model specializing for sentiment analysis in social media. It is not sensitive for the finance domain. The evaluation results for the model is very low. The accuracy of the model is only 0.48 and the other matrices perform quite poor, which makes it overall poorest performance compared to the other models.

As for the evaluation, it is discovered that the metrics that are used to evaluate in a study will change according to the different data that is used, as sometimes it can be balanced and sometimes it can be imbalanced as well. In the current case, MCC was redundant as the data used is balanced therefore it does not provide insights or values in this study.

4.5 Development of Prototype

A prototype GUI of a Financial Sentiment Analysis is developed and built in Google Colab using Gradio. The front end of the system is designed to have two options for the user. The first part would be the Upload CSV, it is created to allow users to upload csvs which contain financial text data, whereas for the second part, it is named as Analyze Text, this part is created to allow users to input text for sentiment classification. As for the backend, it is integrated with the hybrid model (SVM + FinBERT) for processing the data in Python environment.

4.5.1 Upload CSV Function

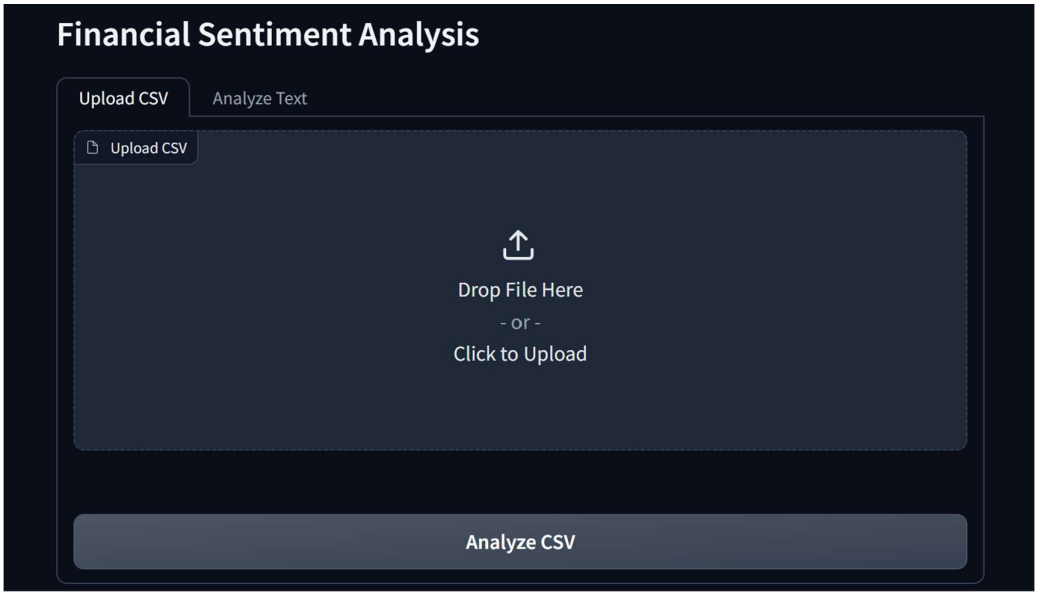


Figure 26 shows the GUI of the Financial Sentiment Analysis.

To use the Upload CSV function, the user can either choose Click to Upload or Drop File Here to import the csv file into the GUI.

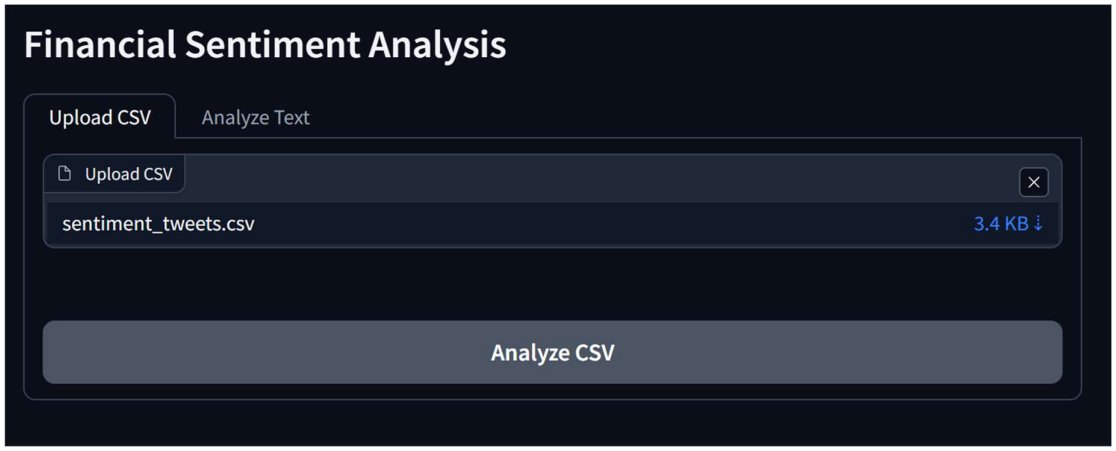


Figure 27 shows the method to upload the csv file.

Once the csv file is uploaded, the user proceeds by clicking the Analyse CSV button at the bottom.

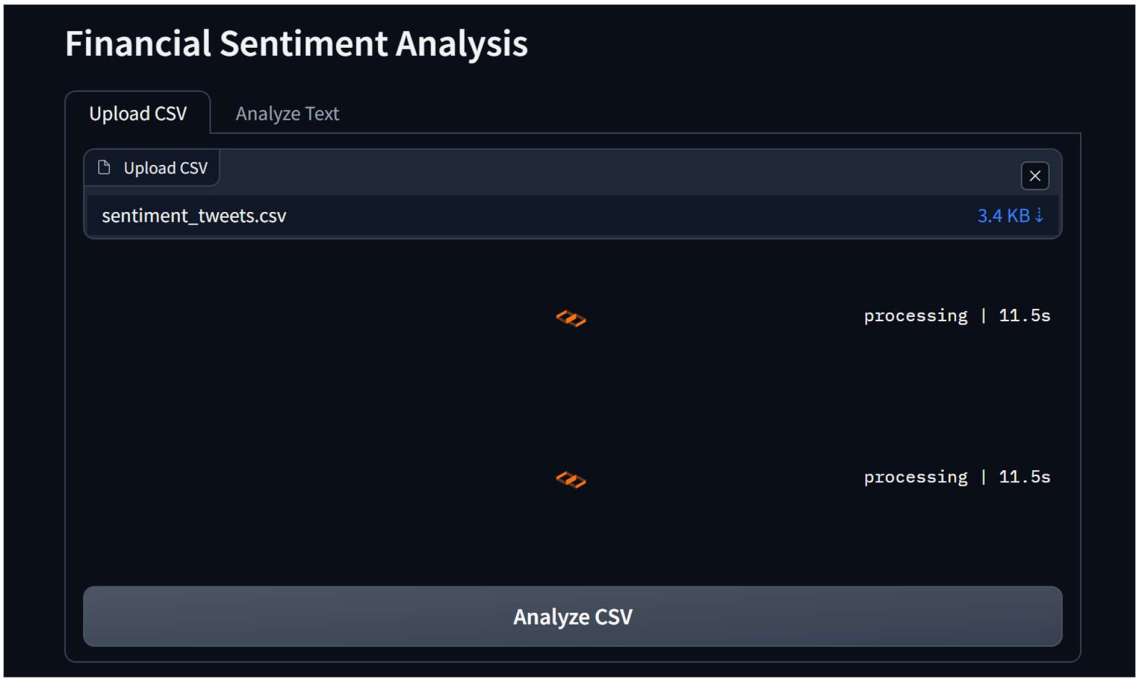


Figure 28 shows the csv file in process.

When the Analyze CSV button is clicked, the prototype will start processing the csv as shown in the above figure.

	text	predicted_sentiment
0	No major movements in the market.	neutral
1	Strong earnings reports are driving stocks up.	bullish
30	Profits are soaring in the market.	bearish
31	The economy is unchanged.	neutral
32	The stock market is stable.	neutral
33	The market remains neutral.	neutral
34	No major movements in the market.	neutral
35	The financial outlook is neutral.	neutral
36	The market is in a holding pattern.	neutral
37	No significant changes in the market today.	neutral
38	The financial outlook is negative.	neutral
39	The stock market is falling.	neutral
40	Weak earnings reports are driving stocks down.	bearish
41	The market is in a holding pattern.	neutral

Figure 29 show how the predicted sentiments look like for the csv.

When the running process is done, the results will be displayed. The user can scroll to see all the text that are classified.

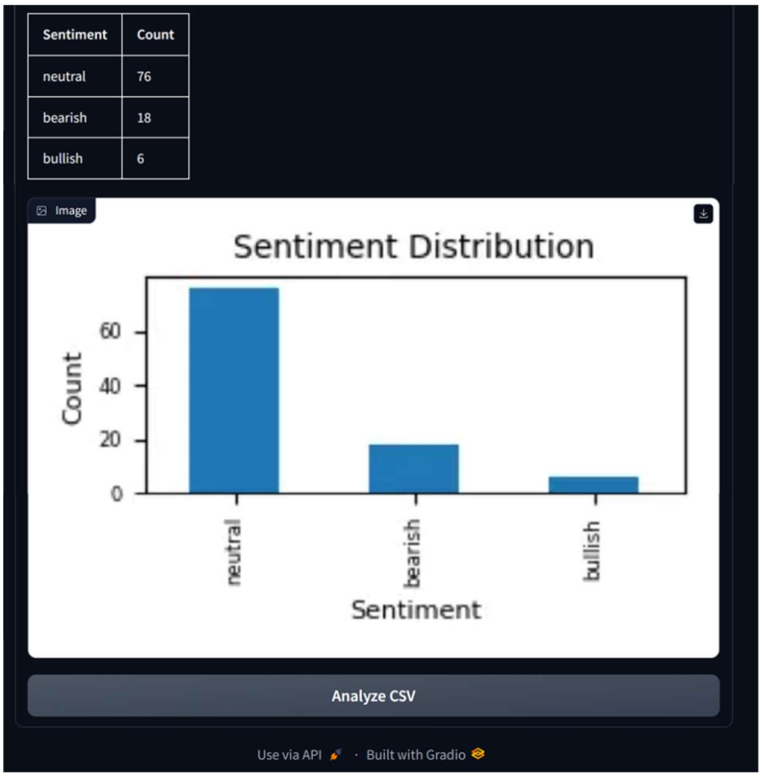


Figure 30 shows the results of the 3 classifications and a chart of the sentiment distribution.

At the bottom of the prototype after the text and predicted sentiments. The prototype will also show the total number in each sentiment classifications and displays a bar chart for visualisation of the csv file. This allows the user to understand the classification of the text data and also the total number of bullish, bearish and neutral text data that are collected in the csv file. This will enable to user to understand the overall classification structure of the text data collected.

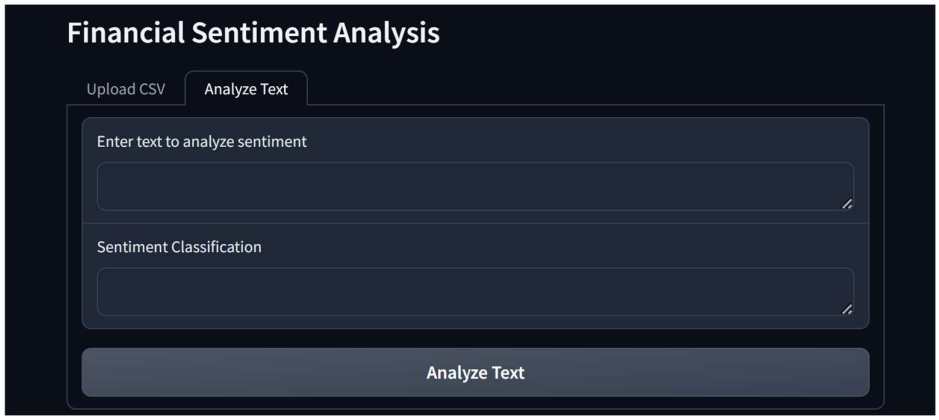
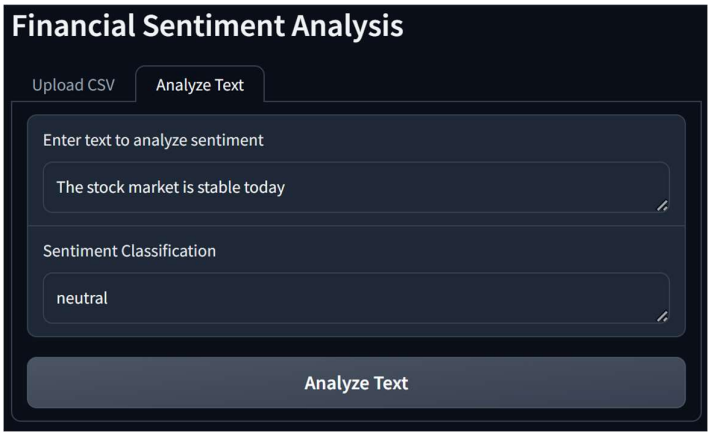
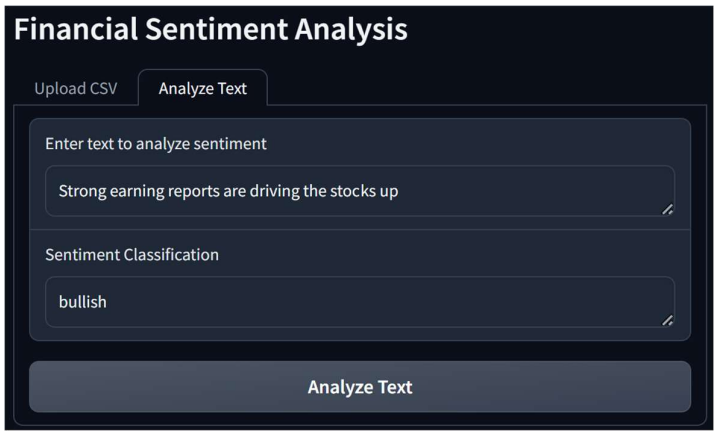


Figure 31 shows how the analyse text section looks like.

4.5.2 Analyze Text

The method to use Analyze Text is simple and straightforward for the user as the user is required to type the text in the Enter text to analyze sentiment part, and then click Analyze Text at the bottom. The figures below displays on how the Analyze Text is used.





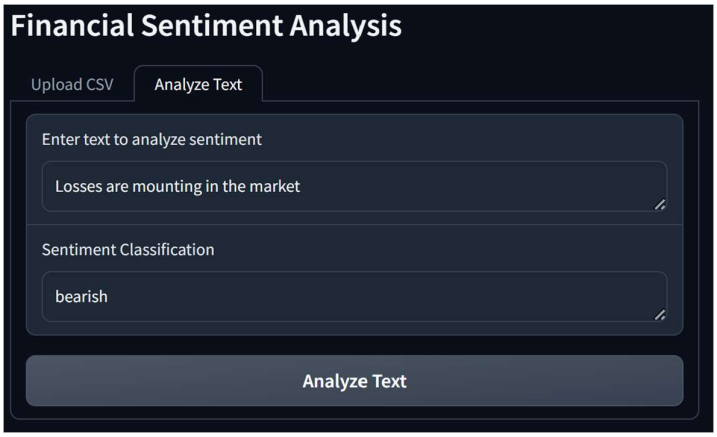


Figure 32 shows how the display of the analyzed text looks like based on the sentiments that are classified.

Chapter 5: Conclusion

This project highlights on the importance of using advance NLP techniques and machine learning to perform sentiment analysis for financial news on X to provide clear sentiment analysis to support decision making. The research problems and objectives are clearly defined and an extensive literature review is done to summarize the factors which might influence the prediction of sentiment analysis models and solutions are proposed to overcome it. Besides that, different financial contexts are studied to understand what are the factors that will affect sentiment analysis in the scenarios.

Furthermore, in research methodology, various models are proposed from SVM, FinBERT, LSTM, Hybrid and VADER in order to compare and evaluate to obtain the best model to be used for sentiment analysis and classification in financial texts. It was found out that Hybrid model performs the best in this study.

The hybrid model is then implemented into the GUI prototype. It is developed using Gradio in Google Colab for allowing users to understand financial sentiments based on the text data the user has collected. It can be said that this approach has demonstrated the importance of implementing advance machine learning models into the Financial Sentiment Analysis GUI while showcasing its application in real-life financial contexts.

5.2 Research Objective Achieved

This project has reflected on the research objectives which are stated in this project, the conclusion is summarized according to the research objectives:

1. As reflected to enhancing accuracy in X's financial news sentiment analysis, studies have been done in exploring different types of NLP techniques to handle the challenges in preprocessing. Based on Chapter 2 literature review, this paper has identified the challenges such as handling unstructured data, managing complicated linguistics and thus providing solutions to overcome it. The information collected has provided insights for developing the preprocessing pipeline. Therefore, the data collected has been well prepared for sentiment analysis.

2. This project has developed a model that integrates efficient NLP methods for enhancing the sentiment detection. In this study a model is created by combining SVM with FinBERT model to create a hybrid model to improve the performance in sentiment classification. It is observed that the hybrid model has a leveraged performance compared with the non-hybrid models.

3. The study has conducted a development of a hybrid model, and evaluations have been conducted to the model. In Chapter 4 result and evaluations, various models from SVM, FinBERT, LSTM, VADER and Hybrid model has been trained and evaluated with accuracy, precision, recall and F1-score accordingly and it was found that hybrid model performs the best in sentiment classification. The evaluation results show the practicability of the model as it proves that this model is workable in real-time financial contexts.

5.3 Contribution and Implications of Research

Finance institutions or organizations can benefit from this project as the project studies on enhancing sentiment analysis techniques which improves the interpretation and understanding of the financial context sentiments. The implication of advanced NLP methods and machine learning can allow one to handle unstructured financial news data with more accurate sentiment predictions, this will help to improve financial performance in decision making and financial planning.

As for researchers, this study conducted is believed to provide valuable references as it allows the researcher to further study this topic. The researcher is encouraged to perform further technical developments and refinement in improving this study by exploring more NLP methods and machine learning models to improve in advanced financial sentiments.

The prototype created plays a role as a practical tool for sentiment analysis which provides classifications based on the text data collected. The prototype is created as a web-based because it is scalable and easier to be integrated into developments.

5.4 Limitations

In this study, there are limitations in the dataset vocabulary as the dataset used in this study might not capture sufficient sentiment expressions in financial news, this may affect the accuracy and the classification ability of the sentiment analysis model. Besides that, the model training process relies intensively on computational resources, and it is highly time consuming. This limitation may affect the performance of an application and scalability when large datasets are used in real-life scenarios for analysis. Furthermore, hyperparameter tuning is not been fully investigated in this study. This limitation states that there might be hidden opportunities in enhancing the model's accuracy if the hyperparameters are adjusted more precisely.

5.5 Future Works

In future works, there is a need in retrieving more text data which are related to financial sentiments in order to increase the vocabulary and increase the accuracy of the sentiment analysis. It is believed that more datasets can increase more vocabulary which are related to finance sentiments. Therefore, the model will have better classification and understanding for a bigger range of sentiment expressions.

Besides that, further investigation on methods should be done to improve the model's efficiency during the training process. This can be done by improving the code with more optimised algorithms and methods to reduce the computational resources needed.

Lastly, in the future, further exploration for tuning the hyperparameter is needed as it will elevate the model's performance. To increase the accuracy of the model, further exploration of techniques such as applying gird search is recommended as it also improves the robustness of the model.

Chapter 6: Gantt Chart

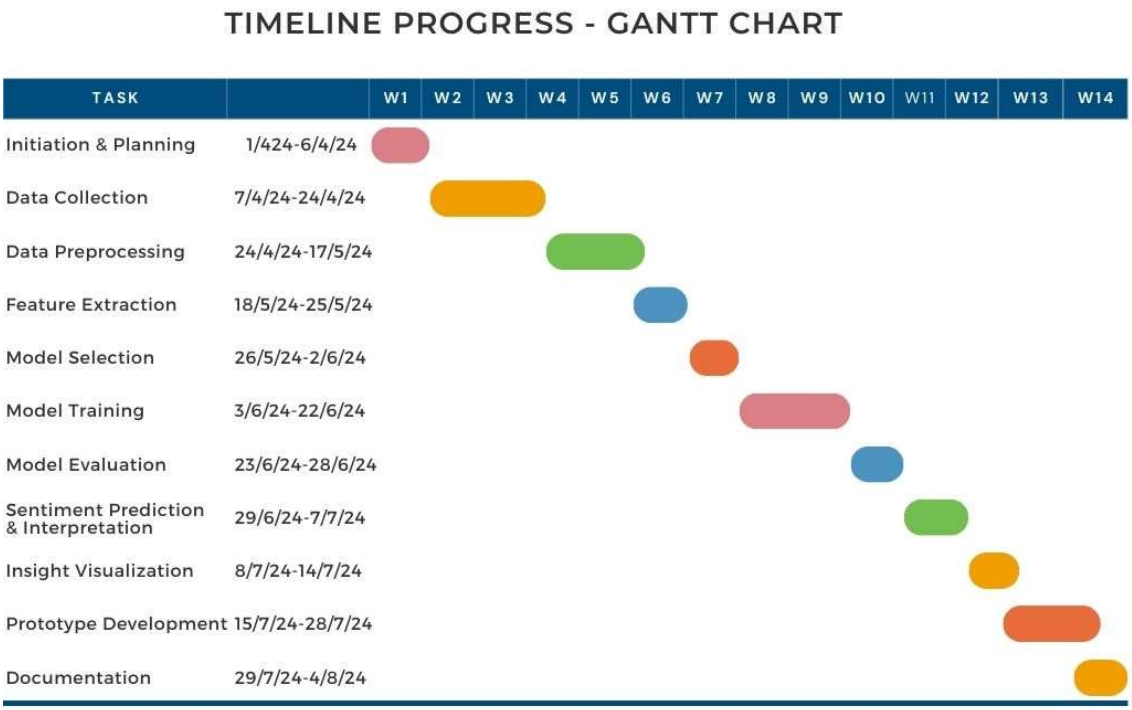


Figure 33 shows the Gantt Chart of the project.

Chapter 7: Reference

Alimova, I., & Tutubalina, E. (2020). Multiple features for clinical relation extraction: A machine learning approach. *Journal of Biomedical Informatics*, *103*, 103382. https://doi.org/10.1016/j.jbi.2020.103382

An, C., Park, Y. W., Ahn, S. S., Han, K., Kim, H., & Lee, S. K. (2021). Radiomics machine learning study with a small sample size: Single random training-test set split may lead to unreliable results. *PloS One*, *16*(8), e0256152. https://doi.org/10.1371/journal.pone.0256152

Arzadon, J. (2023, October 23). Bullish sentiment around crypto. *Betashares*. https://www.betashares.com.au/insights/bullish-sentiment-around-crypto/

Aziz, A. A., & Starkey, A. (2020). Predicting Supervise Machine Learning Performances for Sentiment Analysis Using Contextual-Based Approaches. *IEEE Access*, *8*, 17722–17733. https://doi.org/10.1109/access.2019.2958702

Bloomberg - Are you a robot? (2023, June 13). https://www.bloomberg.com/news/articles/2023-06-13/china-s-stimulus-runs-into-wall-of-doubts-in-a-bearish-market?embedded-checkout=true

Borg, A., & Boldt, M. (2020). Using VADER sentiment and SVM for predicting customer response sentiment. *Expert Systems With Applications*, *162*, 113746. https://doi.org/10.1016/j.eswa.2020.113746

Bustillo, A., Reis, R., Machado, A. R., & Pimenov, D. Y. (2020). Improving the accuracy of machine-learning models with data from machine test repetitions. *Journal of Intelligent Manufacturing*, *33*(1), 203–221. https://doi.org/10.1007/s10845-020-01661-3

Chang, J., Tu, W., Yu, C., & Qin, C. (2021). Assessing dynamic qualities of investor sentiments for stock recommendation. *Information Processing & Management*, *58*(2), 102452. https://doi.org/10.1016/j.ipm.2020.102452

Chen, J., Mao, Q., & Xue, L. (2020). Visual Sentiment Analysis With Active Learning. *IEEE Access*, *8*, 185899–185908. https://doi.org/10.1109/access.2020.3024948

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ. Computer Science, 7, e623. https://doi.org/10.7717/peerj-cs.623

Chouigui, A., Khiroun, O. B., & Elayeb, B. (2021). An Arabic Multi-source News Corpus: Experimenting on Single-document Extractive Summarization. *Arabian Journal for Science and Engineering*, *46*(4), 3925–3938. https://doi.org/10.1007/s13369-020-05258-z

Consoli, S., Barbaglia, L., & Manzan, S. (2021). Fine-Grained, Aspect-Based Sentiment Analysis on Economic and Financial Lexicon. *Social Science Research Network*. https://doi.org/10.2139/ssrn.3766194

Correia, F., Madureira, A. M., & Bernardino, J. (2022). Deep Neural Networks Applied to Stock Market Sentiment Analysis. *Sensors*, *22*(12), 4409. https://doi.org/10.3390/s22124409

Dong, S., & Liu, C. (2021). Sentiment Classification for Financial Texts Based on Deep Learning. *Computational Intelligence and Neuroscience*, *2021*, 1–9. https://doi.org/10.1155/2021/9524705

Fan, W. (2019). Dependencies for Graphs. *ACM Journal of Data and Information Quality*, *11*(2), 1–12. https://doi.org/10.1145/3310230

Gite, S., Patil, S., Dharrao, D., Yadav, M., Basak, S., Rajendran, A., & Kotecha, K. (2023). Textual Feature Extraction Using Ant Colony Optimization for Hate Speech Classification. *Big Data and Cognitive Computing*, *7*(1), 45. https://doi.org/10.3390/bdcc7010045

Ha, H., Prasad, P., Maag, A., & Alsadoon, A. (2019). Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review. *Expert Systems With Applications*, *118*, 272–299. https://doi.org/10.1016/j.eswa.2018.10.003

Habimana, O., Li, Y., Li, R., Gu, X., & Yu, G. (2019). Sentiment analysis using deep learning approaches: an overview. *Science China. Information Sciences/Science China. Information Sciences*, *63*(1). https://doi.org/10.1007/s11432-018-9941-6

HaCohen-Kerner, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. *PloS One*, *15*(5), e0232525. https://doi.org/10.1371/journal.pone.0232525

Hammad, I., & El-Sankary, K. (2019). Practical Considerations for Accuracy Evaluation in Sensor-Based Machine Learning and Deep Learning. *Sensors*, *19*(16), 3491. https://doi.org/10.3390/s19163491

Jabbar, A., Iqbal, S., Tamimy, M. I., Hussain, S., & Akhunzada, A. (2020). Empirical evaluation and study of text stemming algorithms. *Artificial Intelligence Review*, *53*(8), 5559–5588. https://doi.org/10.1007/s10462-020-09828-3

Kang, M. W., Kim, J., Kim, D. K., Oh, K. H., Joo, K. W., Kim, Y. S., & Han, S. S. (2020). Machine learning algorithm to predict mortality in patients undergoing continuous renal replacement therapy. *Critical Care*, *24*(1). https://doi.org/10.1186/s13054-020-2752-7

Khiabani, P. J., Basiri, M. E., & Rastegari, H. (2019). An improved evidence-based aggregation method for sentiment analysis. *Journal of Information Science*, *46*(3), 340–360. https://doi.org/10.1177/0165551519837187

Koltun, V., & Yamshchikov, I. P. (2023). Pump It: Twitter Sentiment Analysis for Cryptocurrency Price Prediction. *Risks*, *11*(9), 159. https://doi.org/10.3390/risks11090159

Kraaijeveld, O., & De Smedt, J. (2020). The predictive power of public Twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions & Money*, *65*, 101188. https://doi.org/10.1016/j.intfin.2020.101188

Li, J., Wang, L., Zhang, X., Liu, L., Li, J., Chan, M. F., Sui, J., & Yang, R. (2019). Machine Learning for Patient-Specific Quality Assurance of VMAT: Prediction and Classification Accuracy. *International Journal of Radiation Oncology, Biology, Physics*, *105*(4), 893–902. https://doi.org/10.1016/j.ijrobp.2019.07.049

Liu, H., Chatterjee, I., Zhou, M., Lu, X. S., & Abusorrah, A. (2020). Aspect-Based Sentiment Analysis: A Survey of Deep Learning Methods. *IEEE Transactions on Computational Social Systems*, *7*(6), 1358–1375. https://doi.org/10.1109/tcss.2020.3033302

López-Cabarcos, M. N., Pérez-Pico, A. M., Piñeiro-Chousa, J., & Šević, A. (2021). Bitcoin volatility, stock market and investor sentiment. Are they connected? *Finance Research Letters*, *38*, 101399. https://doi.org/10.1016/j.frl.2019.101399

Miao, J., & Zhu, W. (2021). Precision–recall curve (PRC) classification trees. Evolutionary Intelligence, 15(3), 1545–1569. https://doi.org/10.1007/s12065-021-00565-2

Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L. T., & Trajanov, D. (2020). Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access*, *8*, 131662–131682. https://doi.org/10.1109/access.2020.3009626

Mohr, F., & Van Rijn, J. N. (2023). Fast and Informative Model Selection Using Learning Curve Cross-Validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(8), 9669–9680. https://doi.org/10.1109/tpami.2023.3251957

Mortaz, E. (2020). Imbalance accuracy metric for model selection in multi-class imbalance classification problems. *Knowledge-based Systems*, *210*, 106490. https://doi.org/10.1016/j.knosys.2020.106490

Nimala, K., Jebakumar, R., & Saravanan, M. (2022). Retraction Note to: Sentiment topic sarcasm mixture model to distinguish sarcasm prevalent topics based on the sentiment bearing words in the tweets. *Journal of Ambient Intelligence & Humanized Computing/Journal of Ambient Intelligence and Humanized Computing*, *14*(S1), 51. https://doi.org/10.1007/s12652-022-03938-2

Pano, T., & Kashef, R. (2020). A Complete VADER-Based Sentiment Analysis of Bitcoin (BTC) Tweets during the Era of COVID-19. *Big Data and Cognitive Computing*, *4*(4), 33. https://doi.org/10.3390/bdcc4040033

Picasso, A., Merello, S., Ma, Y., Oneto, L., & Cambria, E. (2019). Technical analysis and sentiment embeddings for market trend prediction. *Expert Systems With Applications*, *135*, 60–70. https://doi.org/10.1016/j.eswa.2019.06.014

Ren, R., Wu, D. D., & Liu, T. (2019). Forecasting Stock Market Movement Direction Using Sentiment Analysis and Support Vector Machine. *IEEE Systems Journal*, *13*(1), 760–770. https://doi.org/10.1109/jsyst.2018.2794462

Sevetlidis, V., Pavlidis, G., Mouroutsos, S., & Gasteratos, A. (2022). Tackling Dataset Bias With an Automated Collection of Real-World Samples. *IEEE Access*, *10*, 126832–126844. https://doi.org/10.1109/access.2022.3226517

Sham, N. M., & Mohamed, A. (2022). Climate Change Sentiment Analysis Using Lexicon, Machine Learning and Hybrid Approaches. *Sustainability*, *14*(8), 4723. https://doi.org/10.3390/su14084723

Sidey-Gibbons, J. a. M., & Sidey-Gibbons, C. J. (2019). Machine learning in medicine: a practical introduction. *BMC Medical Research Methodology*, *19*(1). https://doi.org/10.1186/s12874-019-0681-4

Tai, W., Zhong, T., Mo, Y., & Zhou, F. (2022). Learning Sentimental and Financial Signals With Normalizing Flows for Stock Movement Prediction. *IEEE Signal Processing Letters*, *29*, 414–418. https://doi.org/10.1109/lsp.2021.3135793

Tuarob, S., Wettayakorn, P., Phetchai, P., Traivijitkhun, S., Lim, S., Noraset, T., & Thaipisutikul, T. (2021). DAViS: a unified solution for data collection, analyzation, and visualization in real-time stock market prediction. *Financial Innovation*, *7*(1). https://doi.org/10.1186/s40854-021-00269-7

Xing, F. Z., Cambria, E., & Zhang, Y. (2019). Sentiment-aware volatility forecasting. *Knowledge-based Systems*, *176*, 68–76. https://doi.org/10.1016/j.knosys.2019.03.029

Xu, Q., Wang, L., Jiang, C., & Liu, Y. (2019). A novel (U)MIDAS-SVR model with multi-source market sentiment for forecasting stock returns. *Neural Computing & Applications*, *32*(10), 5875–5888. https://doi.org/10.1007/s00521-019-04063-6

Yilmaz, S. F., Kaynak, E. B., Koç, A., Dibeklioglu, H., & Kozat, S. S. (2023). Multi-Label Sentiment Analysis on 100 Languages with Dynamic Weighting for Label Imbalance. IEEE Transactions on Neural Networks and Learning Systems, 34(1), 331–343. https://doi.org/10.1109/tnnls.2021.3094304

Yuan, J., Wu, Y., Lu, X., Zhao, Y., Qin, B., & Liu, T. (2020a). Recent advances in deep learning based sentiment analysis. *Science China. Technological Sciences/Science China. Technological Sciences*, *63*(10), 1947–1970. https://doi.org/10.1007/s11431-020-1634-3

Yuan, J., Wu, Y., Lu, X., Zhao, Y., Qin, B., & Liu, T. (2020b). Recent advances in deep learning based sentiment analysis. *Science China. Technological Sciences/Science China. Technological Sciences*, *63*(10), 1947–1970. https://doi.org/10.1007/s11431-020-1634-3

Zucco, C., Calabrese, B., Agapito, G., Guzzi, P. H., & Cannataro, M. (2019). Sentiment analysis for mining texts and social networks data: Methods and tools. *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery/Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, *10*(1). https://doi.org/10.1002/widm.1333

Appendices

Appendix I: Meeting Logs

TAYLOR'S UNIVERSITY | SCHOOL OF COMPUTER SCIENCE & ENGINEERING

MEETING RECORD

Instructions:

*__Note__: *Must be submitted at the end of the block (Block I & II) together with the* ***Project Report***.

Programme Name & Course Code:	12703 Capstone Project 01		
Semester: (*Please select box to* ***tick*** (✓))	✓ March ☐ August ☐ January	Block:	1
Student Name:	Thoo Xing En	Student ID No.:	0368698
Project Title:	**Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News**		
Principal Supervisor:	Dr Syeda Mariam Muzammal		
Co-Supervisor:			

Please include the following details in the ***summary of discussion*** column:
- Meeting with whom? (e.g., supervisory committee, Principal supervisor, or co-supervisor)
- Matters Discussed
- Proposed Action to be Taken

DATE	SUMMARY OF DISCUSSION	SIGNATURE OF THE RESPECTIVE SUPERVISOR
16/05/2024	- Discussion on the preliminary draft of the literature review - Showed the pipeline for the research methodology - Suggest to modify the problem statement and the objectives.	*Mariam*

TAYLOR'S UNIVERSITY | SCHOOL OF COMPUTER SCIENCE & ENGINEERING

MEETING RECORD

Instructions:

*__Note__: *Must be submitted at the end of the block (Block I & II) together with the* ***Project Report***.

Programme Name & Course Code:	12703 Capstone Project 01		
Semester: (*Please select box to* ***tick*** *(✓)*)	✓ March ☐ August ☐ January	Block:	1
Student Name:	Thoo Xing En	Student ID No.:	0368698
Project Title:	**Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News**		
Principal Supervisor:	Dr Syeda Mariam Muzammal		
Co-Supervisor:			

Please include the following details in the ***summary of discussion*** column:
- Meeting with whom? (e.g., supervisory committee, Principal supervisor, or co-supervisor)
- Matters Discussed
- Proposed Action to be Taken

DATE	SUMMARY OF DISCUSSION	SIGNATURE OF THE RESPECTIVE SUPERVISOR
26/04/2024	- Discussion on topic selection for the capstone report	*Mariam*

TAYLOR'S UNIVERSITY | SCHOOL OF COMPUTER SCIENCE & ENGINEERING

MEETING RECORD

Instructions:

*__Note__: *Must be submitted at the end of the block (Block I & II) together with the* ***Project Report***.

Programme Name & Course Code:	12703 Capstone Project 01		
Semester: (*Please select box to* ***tick*** *(✓)*)	✓ March ☐ August ☐ January	Block:	1
Student Name:	Thoo Xing En	Student ID No.:	0368698
Project Title:	**Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News**		
Principal Supervisor:	Dr Syeda Mariam Muzammal		
Co-Supervisor:			

Please include the following details in the ***summary of discussion*** column:
- Meeting with whom? (e.g., supervisory committee, Principal supervisor, or co-supervisor)
- Matters Discussed
- Proposed Action to be Taken

DATE	SUMMARY OF DISCUSSION	SIGNATURE OF THE RESPECTIVE SUPERVISOR
21/05/2024	- Change the themes to numerals for literature review and arrange the citations to the back. - Problem statement and objectives not accurate enough.	*Mariam*

TAYLOR'S UNIVERSITY | SCHOOL OF COMPUTER SCIENCE & ENGINEERING

MEETING RECORD

Instructions:

*__Note__: *Must be submitted at the end of the block (Block I & II) together with the* ***Project Report***.

Programme Name & Course Code:	**Masters In Applied Computing PRJ70404**		
Semester: (*Please select box to* ***tick*** (✓))	☐ March ✓ August ☐ January	Block:	2
Student Name:	**Thoo Xing En**	Student ID No.:	**0368698**
Project Title:	**Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News**		
Principal Supervisor:	Dr Syeda Mariam Muzammal		
Co-Supervisor:			

Please include the following details in the ***summary of discussion*** column:
- Meeting with whom? (e.g., supervisory committee, Principal supervisor, or co-supervisor)
- Matters Discussed
- Proposed Action to be Taken

DATE	SUMMARY OF DISCUSSION	SIGNATURE OF THE RESPECTIVE SUPERVISOR
10/07/2024	- Discussion the progress of early stage in capstone 2. - Show partially of the code and talk about experimental phrase of the code.	*Mariam*

TAYLOR'S UNIVERSITY | SCHOOL OF COMPUTER SCIENCE & ENGINEERING

MEETING RECORD

Instructions:

*__Note__: *Must be submitted at the end of the block (Block I & II) together with the* ***Project Report***.

Programme Name & Course Code:	**Masters In Applied Computing PRJ70404**		
Semester: (*Please select box to* ***tick*** (✓))	☐ March ✓ August ☐ January	Block:	2
Student Name:	**Thoo Xing En**	Student ID No.:	**0368698**
Project Title:	**Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News**		
Principal Supervisor:	Dr Syeda Mariam Muzammal		
Co-Supervisor:			

Please include the following details in the ***summary of discussion*** column:
- Meeting with whom? (e.g., supervisory committee, Principal supervisor, or co-supervisor)
- Matters Discussed
- Proposed Action to be Taken

DATE	SUMMARY OF DISCUSSION	SIGNATURE OF THE RESPECTIVE SUPERVISOR
29/07/2024	- Show updates on the completed code with the evaluation. - Discuss on the documentation and flow for the capstone report with some corrections to be done.	*Mariam*

TAYLOR'S UNIVERSITY | SCHOOL OF COMPUTER SCIENCE & ENGINEERING

MEETING RECORD

Instructions:

*__Note__: *Must be submitted at the end of the block (Block I & II) together with the **Project Report**.*

Programme Name & Course Code:	**Masters In Applied Computing PRJ70404**		
Semester: (*Please select box to **tick** (✓)*)	☐ March ✓ August ☐ January	Block:	2
Student Name:	**Thoo Xing En**	Student ID No.:	**0368698**
Project Title:	**Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News**		
Principal Supervisor:	Dr Syeda Mariam Muzammal		
Co-Supervisor:			

Please include the following details in the ***summary of discussion*** column:
- Meeting with whom? (e.g., supervisory committee, Principal supervisor, or co-supervisor)
- Matters Discussed
- Proposed Action to be Taken

DATE	SUMMARY OF DISCUSSION	SIGNATURE OF THE RESPECTIVE SUPERVISOR
31/7/2024	- Discuss the completed capstone report with amendments.	*Mariam*

Appendix II: Declaration & Copyright Form

Taylor's University

DECLARATION OF MAC CAPSTONE PROJECT REPORT AND COPYRIGHT

Author's full name : Thoo Xing En

Date of birth : 6th Feb 1998

Title : Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News

Academic Session : April 2024

I declare that this report is classified as :

☐ **CONFIDENTIAL** (Contains confidential information under the Official Secret Act 1972)*

☐ **RESTRICTED** (Contains restricted information as specified by the organization where research was done)*

☒ **OPEN ACCESS** I agree that my report to be published as online open access (full text)

I acknowledged that Taylor's University reserves the right as follows:

1. The report is the property of Taylor's University.
2. The Library of Taylor's University has the right to make copies for the purpose of research only.
3. The library has the right to make copies of the report for academic exchange.

SIGNATURE

Date : 04/08/24

Certified by:

SIGNATURE

Date :

NOTES :*If the report is CONFIDENTAL or RESTRICTED, please attach with the letterfrom the organization with period and reasons for confidentiality or restriction.

We hereby declare that we have read this project report and in our opinion, it is sufficient in terms of scope and quality for the award of the degree of **Master of Applied Computing.**

Signature : .....

Name of Supervisor : .....

Date : .....

Signature : .....

Name of Co-Supervisor: .....

Date : .....

Appendix III: Checklist Form

CHECKLIST 2

TAYLOR'S UNIVERSITY
SCHOOL OF COMPUTER SCIENCE & ENGINEERING

CHECKLIST

Submission Of Capstone Project II Report

All items must be submitted (latest) by **Week 7** (Block 1)

Please tick	Item	Remarks (date)
☑	A. 2 softcopies of reports (submit 1 to each supervisor)	04/08/24
☑	B. 1 Softcopy, if necessary	04/08/24
☑	C. Meeting record form	04/08/24
☑	D. Softcopy validation and submission form	04/08/24
☑	E. Turnitin report (similarity index must be below 20%)	04/08/24

*__Note__: *Items **2-5** to be submitted to the module leader.*

Appendix IV: Softcopy Validation Submission Form

SOFTCOPY VALIDATION & SUBMISSION

TAYLOR'S UNIVERSITY
SCHOOL OF COMPUTER SCIENCE & ENGINEERING

SOFTCOPY VALIDATION & SUBMISSION

Programme Name & Course Code: Masters in Applied Computing (*Programme Name*) PRJ70404 (*Course Code*)

Semester: (*Please select box to tick (✓)*): ☐ 1 ☒ 2 Block: 2

Project Title: Enhancing Financial Insights with NLP-Driven Sentiment Analysis on X's Financial News

I (Thoo Xing En) declare that the softcopy version of my project report is according to the format.

✍ *Signature of the student*

Name: Thoo Xing En

Contact No.: 0127219221

Date: 04/08/24

I (Thoo Xing En) verify that the softcopy version of the project report is according to the format.

✍ *Signature of the student*

Name: Thoo Xing En

Date: 04/08/24

Note: *this form must be submitted to the department together with two (2) softcopies.*

For official use only:			
Name:	Click or tap here to enter text.		
Signature:	✍	Date:	Click or tap to enter a date.

Appendix V: Dataset Preview

X	sentiment
$BYND - JPMorgan reels in expectations on Beyond Meat https://t.co/bd0xbFGjkT	2
$CCL $RCL - Nomura points to bookings weakness at Carnival and Royal Caribbean https://t.co/yGjpT2ReD3	2
$CX - Cemex cut at Credit Suisse, J.P. Morgan on weak building outlook https://t.co/KN1g4AWFIb	2
$ESS: BTIG Research cuts to Neutral https://t.co/MCyfTsXc2N	2
$FNKO - Funko slides after Piper Jaffray PT cut https://t.co/z37IJmCQzB	2
$FTI - TechnipFMC downgraded at Berenberg but called Top Pick at Deutsche Bank https://t.co/XKcPDilIuU	2
$GM - GM loses a bull https://t.co/tdUfG5HbXy	2
$GM: Deutsche Bank cuts to Hold https://t.co/7Fv1ZiFZBS	2
$GTT: Cowen cuts to Market Perform	2
$HNHAF $HNHPD $AAPL - Trendforce cuts iPhone estimate after Foxconn delay https://t.co/rlnEwzlzzS	2
$HOG - Moody's warns on Harley-Davidson https://t.co/LurHBEadeU	2
$HXL - Citing aero ties, Wells slashes PT on Hexcel https://t.co/wU5P2i8WBU	2
$I - Intelsat cut to Market Perform at Raymond James https://t.co/YsvsMSQRIb	2
$KRG: Compass Point cuts to Sell https://t.co/MCyfTsXc2N	2
$LK - Muddy Waters goes short Luckin Coffee https://t.co/8yrbwAjLKG	2
$MANT - ManTech downgraded ahead of difficult comps https://t.co/mJ1eSrsFXJ	2
$MDCO: Oppenheimer cuts to Perform	2
$MPLX $MPC - MPLX cut at Credit Suisse on potential dilution from Marathon strategic review https://t.co/0BFQy4ZU6W	2
$MSGN - Imperial downgrades MSG Networks amid sports-free airwaves https://t.co/Ul2S6XNXw8	2
$MTLS - Piper hits the Materialise sidelines https://t.co/qlFxZuhkrt	2
$NCBS: Hovde Group cuts to Market Perform	2
$NFLX - New Netflix bear steps out https://t.co/PdPkgLk0FQ	2
$SHOP - Shopify loses a bull https://t.co/eoq7Lchtk4	2
$STAY - Nomura Instinet loses confidence in Extended Stay America https://t.co/WrQXTC38nB	2
$TWLO - Twilio gets Street-low target on virus risk https://t.co/Ky3d8DxnjX	2
$VCLT $SPLB $IGLB - Guggenheim's Minerd sees more gloom ahead in stocks https://t.co/NKYQgy2o3J	2
$VIAC $VIACA $FOX - MoffettNathanson expects NFL TV costs to skyrocket https://t.co/APfxEqSSRF	2
$VIAC: Barrington Research cuts to Mkt Perform https://t.co/7Fv1ZiFZBS	2
$XLNX - Mizuho cuts XLNX target on near-term headwinds https://t.co/hsWCvJb1Ct	2
2U downgraded to sector weight from overweight at KeyBanc Capital	2
Sales of grounded Boeing 737 MAX pick up steam at Dubai Airshow	1
SourceDay Announces Record Growth as One of Austinâ€™s Fastest-Growing Companies	1
Tezos rallies into the top ten cryptocurrencies, ousting Stellar	1
This chart shows how Vanguardâ€™s explosive growth has â€˜taken on a life of its ownâ€™: https://t.co/8IOnrejNLy	1
UPDATE 1-Keystone operator TC Energy sees EBITDA exceeding C$10 bln in 2022	1
UPDATE 1-Qualcomm expects 5G phone sales to double in 2021	1
Why Walmart and Target are still winning with consumers https://t.co/TvPW2GzpQ6 by @DegrootMckenzie https://t.co/MHBf51cVBC	1
$CVX - Chevron ramps output at key Venezuela oil project as U.S. weighs sanctions https://t.co/o4g9WuvJfo	1
BP expects to keep Azeri ACG oil output stable as Shah Deniz gas powers ahead https://t.co/dcNqW5BxVN https://t.co/a0y3sr6K0u	1
$AAPL - Apple delays theatrical debut of 'The Banker' https://t.co/vAHWmIoXp8	0
$AAPL - Max out Apple's Mac Pro for $52,599 https://t.co/HIRuhy8Pqd	0
$ACER - Acer Therapeutics: Upcoming Catalysts. https://t.co/vseXJ9e1nH #business #investing #markets	0
$APLS - Apellis Pharmaceuticals (APLS) Investor Presentation - Slideshow. Read more and get updates on any stock!â€¦ https://t.co/wJL	0
$ATRO - Astronics unveils universal radio field test solution https://t.co/diC9vddtwB	0
$AUDC: AudioCodes enters into royalty buyout agreement with the Israel National Authority for Technology and... https://t.co/YVzJTV68b	0
$BA 737 MAX makes debut at Renton, Washington Factory - Bloomberg	0
$BGNE: BeiGene reports clinical data on Tislelizumab https://t.co/f1bfcfhxoh	0
$CMI - Cummins (CMI) Investor Presentation - Slideshow. Continue reading: https://t.co/4ropk5quxp #investing #stocks #business	0
$DAN - Dana (DAN) Presents At 7th Annual Industrials Conference - Slideshow. Get more updates here:â€¦ https://t.co/34xt9wQ6lG	0
$ENLC - EnLink Midstream (ENLC) Presents At RBC Capital Markets Midstream Conference - Slideshow. Continue reading:â€¦ https://t.c	0
$EPR: EPR Properties announces sale of charter school portfolio for approximately $454 mln; lowers FY19 FFO guidance https://t.co/i4G	0
$EQNR $SU - Siccar Point attracts bids as high as nearly $2B - Reuters https://t.co/CMSO3XjoN1	0
$INTI - Inhibitor Therapeutics to start prostate cancer treatment trial https://t.co/hKmHpHCXGJ	0
$ITRN - Margins suffer at Ituran Location in Q3 https://t.co/cCI94Csl3s	0

Dataset from 2017 – 2022 on financial sentiment.

Appendix VI: Source Codes

```
!pip install transformers[torch] accelerate -U
!pip install datasets
!pip install vaderSentiment
!pip install scikit-learn
!pip install matplotlib
!pip install seaborn
```

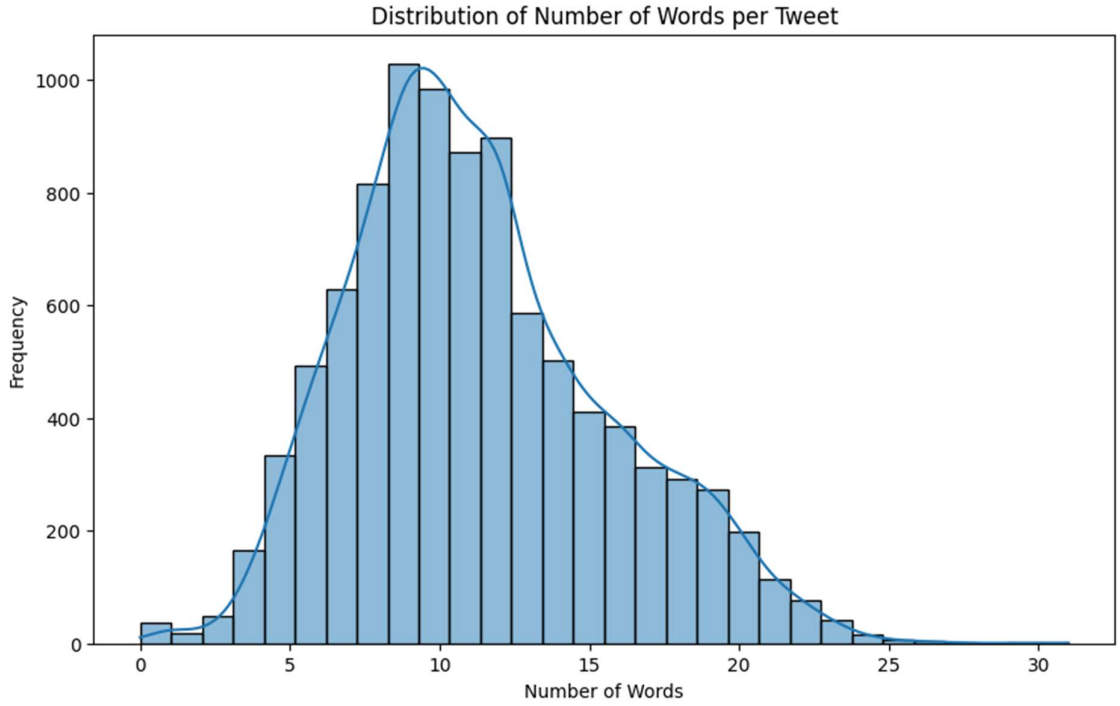
```
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-packages (0.32.1)
Collecting accelerate
  Downloading accelerate-0.33.0-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: transformers[torch] in /usr/local/lib/python3.10/dist-packages (4.42.4)
Collecting transformers[torch]
  Downloading transformers-4.43.3-py3-none-any.whl.metadata (43 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.7/43.7 kB 1.2 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from tra
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers[to
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers[
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transfor
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers[torch])
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.3
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hu
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from hugg
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->transformers[torch
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->transformers[to
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->transformers[torc
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch->transformers[torch])
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch->transformers[torch])
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch->transformers[torch])
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch->transformers[torch])
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch->transformers[torch])
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch->transformers[torch])
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch->transformers[torch])
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch->transformers[torch])
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl.metadata (1.8 kB)
```

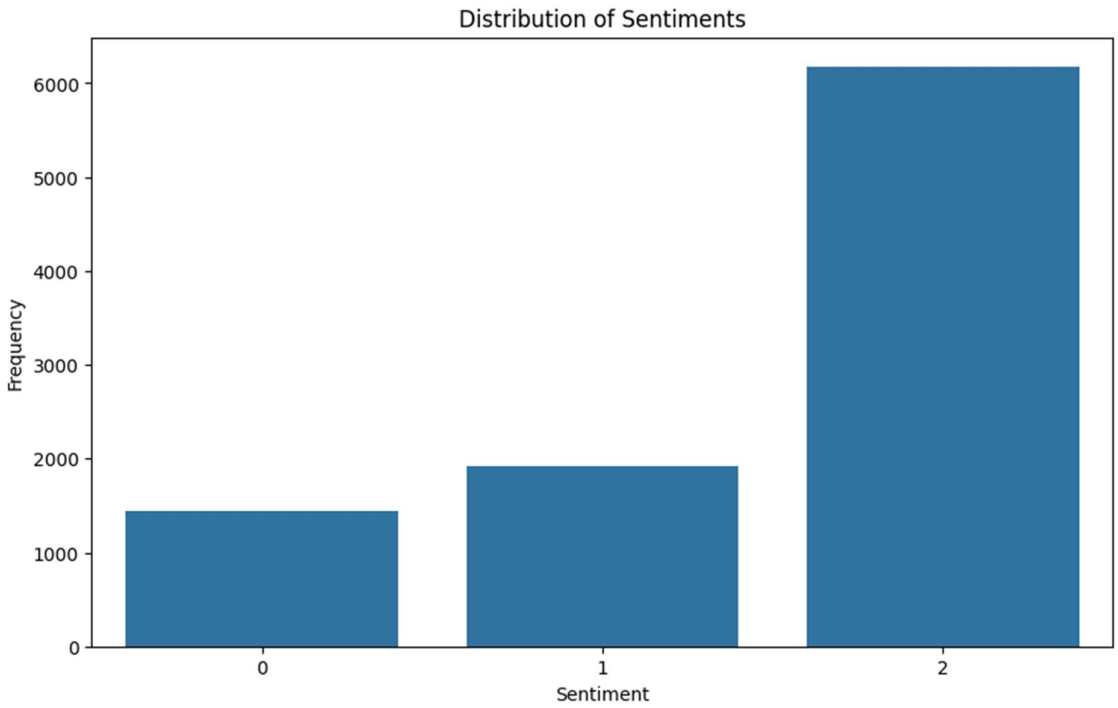
```
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl.metadata (572 bytes)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.32.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from reques
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSen
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->va
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->va
Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 2.7 MB/s eta 0:00:00
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.1
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lea
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (
```

```
import pandas as pd
import time
import re
from transformers import pipeline, AutoTokenizer, BertTokenizer, BertForSequenceClassification, Trainer, TrainingAr
from datasets import load_dataset, DatasetDict
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, matthews_corrcoef, precision_r
from sklearn.preprocessing import LabelEncoder
import gensim.downloader as api
import numpy as np
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import LSTM, Dense, Masking
import matplotlib.pyplot as plt
import seaborn as sns
import random

# Download NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
]  # Load the dataset
   dataset = load_dataset("zeroshot/twitter-financial-news-sentiment")

   # Visualize the distribution of number of words per tweet
   def preprocess_tweet(text):
       text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)  # Remove URLs
       text = re.sub(r'\@\w+|\#', '', text)  # Remove mentions and hashtags
       text = re.sub(r'[^A-Za-z0-9 ]+', '', text)  # Remove special characters
       return text

   # Preprocess the tweets
   dataset = dataset.map(lambda x: {"cleaned_tweet": preprocess_tweet(x["text"])})

   # Visualize number of words per tweet
   dataset_df = pd.DataFrame(dataset['train'])
   dataset_df['word_count'] = dataset_df['cleaned_tweet'].apply(lambda x: len(x.split()))

   plt.figure(figsize=(10, 6))
   sns.histplot(dataset_df['word_count'], bins=30, kde=True)
   plt.title('Distribution of Number of Words per Tweet')
   plt.xlabel('Number of Words')
   plt.ylabel('Frequency')
   plt.show()

   # Check the distribution of sentiments
   plt.figure(figsize=(10, 6))
   sns.countplot(x=dataset_df['label'])
   plt.title('Distribution of Sentiments')
   plt.xlabel('Sentiment')
   plt.ylabel('Frequency')
   plt.show()
```






```
# Initialize the FinBERT pipeline and tokenizer
pipe = pipeline("text-classification", model="ProsusAI/finbert")
tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
```

```
config.json: 100% 758/758 [00:00<00:00, 16.0kB/s]
pytorch_model.bin: 100% 438M/438M [00:04<00:00, 111MB/s]
tokenizer_config.json: 100% 252/252 [00:00<00:00, 3.50kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 3.58MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 2.90kB/s]
```

```python
# Get the list of stop words
stop_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

# preprocess text with lemmatization and normalization
def preprocess_text_with_lemmatization(text):
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    text = re.sub(r'\@\w+|\#', '', text)
    text = re.sub(r'[^A-Za-z0-9 ]+', '', text)
    words = text.split()
    words = [word for word in words if word not in stop_words]  # Remove stop words
    words = [lemmatizer.lemmatize(word) for word in words]  # Apply lemmatization
    return ' '.join(words)

# Function to classify sentiment with adjustable thresholds
def classify_tweet_with_thresholds(tweet, bearish_threshold=0.2, bullish_threshold=0.4):
    results = pipe(tweet)
    for result in results:
        label = result['label']
        score = result['score']
        print(f"Debug: Label = {label}, Score = {score}")

    # Use the highest scoring label
    top_result = results[0]
    label = top_result['label']
    score = top_result['score']

    # Check thresholds
    if label == 'negative':
        if score >= bearish_threshold:
            return 'Bearish'
    elif label == 'positive':
        if score >= bullish_threshold:
            return 'Bullish'
    return 'Neutral'
```

Part 3: Analyze Sentiment with Thresholds

```
[ ]  # Function to analyze sentiment with thresholds
     def analyze_sentiment_with_thresholds(example):
         preprocessed_text = preprocess_text_with_lemmatization(example["text"])
         sentiment = classify_tweet_with_thresholds(preprocessed_text)
         example["predicted_sentiment"] = sentiment
         return example
```

Part 4: Prepare Dataset

```
[ ]  # Define the number of samples you want to keep
     num_samples = 3000

     # Function to get a random subset of the dataset
     def get_random_subset(dataset, num_samples):
         import random
         indices = list(range(len(dataset)))
         random.shuffle(indices)
         selected_indices = indices[:num_samples]
         return dataset.select(selected_indices)

     # Split the dataset into train and test
     dataset = dataset['train'].train_test_split(test_size=0.2, seed=42)

     # Get random subsets for train and test
     train_subset = get_random_subset(dataset['train'], num_samples)
     test_subset = get_random_subset(dataset['test'], int(num_samples * 0.2))

     # Create a DatasetDict with the subsets
     dataset_dict_subset = DatasetDict({
         'train': train_subset,
         'test': test_subset
     })
```

5: Apply Sentiment Analysis

```
# Start timing for the sentiment analysis step
start_time = time.time()

# Apply the updated function to the dataset with thresholds
dataset_lemmatized_with_thresholds_subset = dataset_dict_subset.map(analyze_sentiment_with_thresholds)

# End timing for the sentiment analysis step
end_time = time.time()
print(f"Time taken for sentiment analysis: {end_time - start_time} seconds")

# Display some results
print(dataset_lemmatized_with_thresholds_subset["train"][0])
```

```
# Check the distribution of predicted sentiments
predicted_sentiments = dataset_lemmatized_with_thresholds_subset["train"]["predicted_sentiment"]
unique_sentiments, sentiment_counts = np.unique(predicted_sentiments, return_counts=True)
print(f"Predicted sentiment distribution: {dict(zip(unique_sentiments, sentiment_counts))}")
```

```
Predicted sentiment distribution: {'Bearish': 521, 'Bullish': 351, 'Neutral': 2128}
```

```
# Check the distribution of predicted sentiments
predicted_sentiments = dataset_lemmatized_with_thresholds_subset["train"]["predicted_sentiment"]
unique_sentiments, sentiment_counts = np.unique(predicted_sentiments, return_counts=True)
print(f"Predicted sentiment distribution: {dict(zip(unique_sentiments, sentiment_counts))}")

# Calculate percentage of each sentiment
total_count = np.sum(sentiment_counts)
percentage_distribution = {sentiment: (count / total_count) * 100 for sentiment, count in zip(unique_sentiments, ser

# Print the percentage distribution
print(f"Percentage distribution: {percentage_distribution}")
```

```
Predicted sentiment distribution: {'Bearish': 521, 'Bullish': 351, 'Neutral': 2128}
Percentage distribution: {'Bearish': 17.366666666666667, 'Bullish': 11.700000000000001, 'Neutral': 70.93333333333334}
```

```
# Extract text and labels from the dataset
df_train = pd.DataFrame(dataset_lemmatized_with_thresholds_subset["train"])
corpus = df_train["text"]
labels = df_train["predicted_sentiment"]

# Encode labels to numerical values
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(labels)

# Check the class distribution before splitting
unique_classes, class_counts = np.unique(y, return_counts=True)
print(f"Class distribution before splitting: {dict(zip(unique_classes, class_counts))}")
```

```
Class distribution before splitting: {0: 521, 1: 351, 2: 2128}
```

```
# Initialize the TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(corpus)

# Split the data into training and testing sets using stratified split
train_indices, test_indices, y_train, y_test = train_test_split(df_train.index, y, test_size=0.2, random_state=42, s
X_train = X_tfidf[train_indices]
X_test = X_tfidf[test_indices]

# Check the distribution of classes in y_train and y_test
print("Training set class distribution:", np.bincount(y_train))
print("Test set class distribution:", np.bincount(y_test))
```

```
Training set class distribution: [ 417  281 1702]
Test set class distribution: [104  70 426]
```

7: Example Tweet Classification

```
# Example tweet to classify
tweet = "The company's lack of innovation will lead to stagnation and declining market share.."

# Preprocess the tweet
preprocessed_tweet = preprocess_text_with_lemmatization(tweet)

# Classify the tweet
sentiment = classify_tweet_with_thresholds(preprocessed_tweet)

# Display the classification result
print(f'The tweet is classified as: {sentiment}')
```

```
Debug: Label = negative, Score = 0.9650284051895142
The tweet is classified as: Bearish
```

SVM WITH TF-IDF

```
# Train and evaluate SVM with TF-IDF
svm_model_tfidf = SVC(probability=True)
svm_model_tfidf.fit(X_train, y_train)

# Make predictions
y_pred_svm_tfidf = svm_model_tfidf.predict(X_test)
```

```
# Evaluate SVM Model with TF-IDF
accuracy_svm_tfidf = accuracy_score(y_test, y_pred_svm_tfidf)
precision_svm_tfidf = precision_score(y_test, y_pred_svm_tfidf, average='macro')
recall_svm_tfidf = recall_score(y_test, y_pred_svm_tfidf, average='macro')
f1_svm_tfidf = f1_score(y_test, y_pred_svm_tfidf, average='macro')
mcc_svm_tfidf = matthews_corrcoef(y_test, y_pred_svm_tfidf)

print(f'''SVM with TF-IDF -
Accuracy: {accuracy_svm_tfidf},
Precision: {precision_svm_tfidf},
Recall: {recall_svm_tfidf},
F1 Score: {f1_svm_tfidf},
MCC: {mcc_svm_tfidf}''')
```

```
SVM with TF-IDF -
Accuracy: 0.7183333333333334,
Precision: 0.7390572390572391,
Recall: 0.35091575091575095,
F1 Score: 0.3123819898329702,
MCC: 0.14660926306285205
```

```
# Load GloVe pre-trained vectors
glove_vectors = api.load("glove-wiki-gigaword-100")

def get_glove_embeddings(text):
    words = text.split()
    embeddings = [glove_vectors[word] for word in words if word in glove_vectors]
    return embeddings

glove_embeddings = [get_glove_embeddings(text) for text in corpus]

def aggregate_embeddings(embeddings):
    if embeddings:
        return np.mean(embeddings, axis=0)  # Averaging embeddings
    else:
        return np.zeros(100)  # Assuming GloVe vector size is 100

X_glove = np.array([aggregate_embeddings(embedding) for embedding in glove_embeddings])
```

```
[==================================================] 100.0% 128.1/128.1MB downloaded
```

```
# Split the data into training and testing sets
X_train_glove, X_test_glove, y_train_glove, y_test_glove = train_test_split(X_glove, y, test_size=0.2, random_state=
```

Evaluate SVM Model with GloVe

```
[ ]  # Train and evaluate SVM with GloVe
     svm_model_glove = SVC(probability=True)
     svm_model_glove.fit(X_train_glove, y_train_glove)

     # Make predictions
     y_pred_svm_glove = svm_model_glove.predict(X_test_glove)

     # Evaluate SVM Model with GloVe
     accuracy_svm_glove = accuracy_score(y_test_glove, y_pred_svm_glove)
     precision_svm_glove = precision_score(y_test_glove, y_pred_svm_glove, average='macro')
     recall_svm_glove = recall_score(y_test_glove, y_pred_svm_glove, average='macro')
     f1_svm_glove = f1_score(y_test_glove, y_pred_svm_glove, average='macro')
     mcc_svm_glove = matthews_corrcoef(y_test_glove, y_pred_svm_glove)

     print(f'''SVM with GloVe -
     Accuracy: {accuracy_svm_glove},
     Precision: {precision_svm_glove},
     Recall: {recall_svm_glove},
     F1 Score: {f1_svm_glove},
     MCC: {mcc_svm_glove}''')
```

```
SVM with GloVe -
Accuracy: 0.7433333333333333,
Precision: 0.45429556263362275,
Recall: 0.40064604149111194,
F1 Score: 0.3904260104093042,
MCC: 0.2866400359986444
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precisi
  _warn_prf(average, modifier, msg_start, len(result))
```

Hyperparameter Tuning and Evaluation

```
[ ]  from sklearn.svm import SVC
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, matthews_corrcoef, precision_
     from sklearn.model_selection import GridSearchCV

     # Define the parameter grid for GridSearchCV
     param_grid = {
         'C': [0.1, 1, 10, 100],
         'gamma': [1, 0.1, 0.01, 0.001],
         'kernel': ['rbf', 'linear']
     }

     # Initialize GridSearchCV with SVM model and parameter grid
     grid_search = GridSearchCV(SVC(class_weight='balanced', probability=True), param_grid, refit=True, verbose=2, cv=3

     # Perform the grid search
     grid_search.fit(X_train, y_train)

     # Get the best parameters
     print(f'Best Parameters: {grid_search.best_params_}')

     # Make predictions with the best model
     y_pred_svm_tuned = grid_search.best_estimator_.predict(X_test)
     y_pred_proba_svm_tuned = grid_search.best_estimator_.predict_proba(X_test)  # Get the predicted probabilities

     # Evaluate the best SVM model
     accuracy_svm_tuned = accuracy_score(y_test, y_pred_svm_tuned)
     precision_svm_tuned = precision_score(y_test, y_pred_svm_tuned, average='macro')
     recall_svm_tuned = recall_score(y_test, y_pred_svm_tuned, average='macro')
     f1_svm_tuned = f1_score(y_test, y_pred_svm_tuned, average='macro')
     mcc_svm_tuned = matthews_corrcoef(y_test, y_pred_svm_tuned)

     print(f'Tuned SVM - Accuracy: {accuracy_svm_tuned}, Precision: {precision_svm_tuned}, Recall: {recall_svm_tuned},

     # Compute Precision-Recall AUC for each class
     precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_pred_proba_svm_tuned[:, 1], pos_label=1)  # For
     pr_auc_svm_tuned = auc(recall_vals, precision_vals)

     print(f'Tuned SVM - Precision-Recall AUC: {pr_auc_svm_tuned}')
```

```
[CV] END .....................C=10, gamma=1, kernel=linear; total time=   2.9s
[CV] END .....................C=10, gamma=0.1, kernel=rbf; total time=   4.1s
[CV] END .....................C=10, gamma=0.1, kernel=rbf; total time=   3.2s
[CV] END .....................C=10, gamma=0.1, kernel=rbf; total time=   3.5s
[CV] END ....................C=10, gamma=0.1, kernel=linear; total time=   3.4s
[CV] END ....................C=10, gamma=0.1, kernel=linear; total time=   3.7s
[CV] END ....................C=10, gamma=0.1, kernel=linear; total time=   2.9s
[CV] END .....................C=10, gamma=0.01, kernel=rbf; total time=   3.3s
[CV] END .....................C=10, gamma=0.01, kernel=rbf; total time=   3.3s
[CV] END .....................C=10, gamma=0.01, kernel=rbf; total time=   4.2s
[CV] END ...................C=10, gamma=0.01, kernel=linear; total time=   3.0s
[CV] END ...................C=10, gamma=0.01, kernel=linear; total time=   2.8s
[CV] END ...................C=10, gamma=0.01, kernel=linear; total time=   2.8s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   3.8s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   3.9s
[CV] END .....................C=10, gamma=0.001, kernel=rbf; total time=   3.3s
[CV] END ...................C=10, gamma=0.001, kernel=linear; total time=   2.9s
[CV] END ...................C=10, gamma=0.001, kernel=linear; total time=   3.0s
[CV] END ...................C=10, gamma=0.001, kernel=linear; total time=   3.7s
[CV] END .....................C=100, gamma=1, kernel=rbf; total time=   3.4s
[CV] END .....................C=100, gamma=1, kernel=rbf; total time=   3.3s
[CV] END .....................C=100, gamma=1, kernel=rbf; total time=   3.4s
[CV] END ....................C=100, gamma=1, kernel=linear; total time=   3.8s
[CV] END ....................C=100, gamma=1, kernel=linear; total time=   2.9s
[CV] END ....................C=100, gamma=1, kernel=linear; total time=   2.9s
[CV] END .....................C=100, gamma=0.1, kernel=rbf; total time=   3.2s
[CV] END .....................C=100, gamma=0.1, kernel=rbf; total time=   3.8s
[CV] END .....................C=100, gamma=0.1, kernel=rbf; total time=   3.6s
[CV] END ...................C=100, gamma=0.1, kernel=linear; total time=   2.9s
[CV] END ...................C=100, gamma=0.1, kernel=linear; total time=   2.8s
[CV] END ...................C=100, gamma=0.1, kernel=linear; total time=   3.0s
[CV] END ....................C=100, gamma=0.01, kernel=rbf; total time=   4.1s
[CV] END ....................C=100, gamma=0.01, kernel=rbf; total time=   3.2s
[CV] END ....................C=100, gamma=0.01, kernel=rbf; total time=   3.3s
[CV] END ..................C=100, gamma=0.01, kernel=linear; total time=   2.9s
[CV] END ..................C=100, gamma=0.01, kernel=linear; total time=   3.7s
[CV] END ..................C=100, gamma=0.01, kernel=linear; total time=   3.1s
[CV] END ....................C=100, gamma=0.001, kernel=rbf; total time=   3.3s
[CV] END ....................C=100, gamma=0.001, kernel=rbf; total time=   3.3s
[CV] END ....................C=100, gamma=0.001, kernel=rbf; total time=   4.1s
[CV] END ..................C=100, gamma=0.001, kernel=linear; total time=   3.2s
[CV] END ..................C=100, gamma=0.001, kernel=linear; total time=   2.8s
[CV] END ..................C=100, gamma=0.001, kernel=linear; total time=   2.9s
Best Parameters: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
Tuned SVM - Accuracy: 0.7766666666666666, Precision: 0.6705200917959567, Recall: 0.5797146124610913, F1 Score:
Tuned SVM - Precision-Recall AUC: 0.45712946076536404
```

```python
param_grid = {
    'C': [0.1, 1, 10, 100, 1000],
    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
    'kernel': ['rbf', 'linear']
}

# Initialize GridSearchCV with SVM model and parameter grid
grid_search = GridSearchCV(SVC(class_weight='balanced', probability=True), param_grid, refit=True, verbose=2, cv=5)

# Perform the grid search
grid_search.fit(X_train, y_train)

# Get the best parameters
print(f'Best Parameters: {grid_search.best_params_}')

# Make predictions with the best model
y_pred_svm_tuned = grid_search.best_estimator_.predict(X_test)
y_pred_proba_svm_tuned = grid_search.best_estimator_.predict_proba(X_test)  # Get the predicted probabilities

# Evaluate the best SVM model
accuracy_svm_tuned = accuracy_score(y_test, y_pred_svm_tuned)
precision_svm_tuned = precision_score(y_test, y_pred_svm_tuned, average='macro')
recall_svm_tuned = recall_score(y_test, y_pred_svm_tuned, average='macro')
f1_svm_tuned = f1_score(y_test, y_pred_svm_tuned, average='macro')
mcc_svm_tuned = matthews_corrcoef(y_test, y_pred_svm_tuned)

print(f'Tuned SVM - Accuracy: {accuracy_svm_tuned}, Precision: {precision_svm_tuned}, Recall: {recall_svm_tuned}, F1

# Compute Precision-Recall AUC for each class
precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_pred_proba_svm_tuned[:, 1], pos_label=1)  # For cl
pr_auc_svm_tuned = auc(recall_vals, precision_vals)

print(f'Tuned SVM - Precision-Recall AUC: {pr_auc_svm_tuned}')
```

```
[CV] END .................C=100, gamma=0.0001, kernel=linear; total time=   4.0s
[CV] END .................C=100, gamma=0.0001, kernel=linear; total time=   4.0s
[CV] END .................C=100, gamma=0.0001, kernel=linear; total time=   5.0s
[CV] END .................C=100, gamma=0.0001, kernel=linear; total time=   3.9s
[CV] END .................C=100, gamma=0.0001, kernel=linear; total time=   4.0s
[CV] END ........................C=1000, gamma=1, kernel=rbf; total time=   5.7s
[CV] END ........................C=1000, gamma=1, kernel=rbf; total time=   4.8s
[CV] END ........................C=1000, gamma=1, kernel=rbf; total time=   4.6s
[CV] END ........................C=1000, gamma=1, kernel=rbf; total time=   5.8s
```

```
[CV] END .................C=1000, gamma=0.01, kernel=linear; total time=   4.3s
[CV] END .................C=1000, gamma=0.01, kernel=linear; total time=   4.5s
[CV] END ...................C=1000, gamma=0.001, kernel=rbf; total time=   4.3s
[CV] END ...................C=1000, gamma=0.001, kernel=rbf; total time=   4.5s
[CV] END ...................C=1000, gamma=0.001, kernel=rbf; total time=   5.1s
[CV] END ...................C=1000, gamma=0.001, kernel=rbf; total time=   4.3s
[CV] END ...................C=1000, gamma=0.001, kernel=rbf; total time=   4.5s
[CV] END ................C=1000, gamma=0.001, kernel=linear; total time=   4.8s
[CV] END ................C=1000, gamma=0.001, kernel=linear; total time=   4.0s
[CV] END ................C=1000, gamma=0.001, kernel=linear; total time=   3.9s
[CV] END ................C=1000, gamma=0.001, kernel=linear; total time=   4.9s
[CV] END ................C=1000, gamma=0.001, kernel=linear; total time=   4.0s
[CV] END ..................C=1000, gamma=0.0001, kernel=rbf; total time=   4.6s
[CV] END ..................C=1000, gamma=0.0001, kernel=rbf; total time=   5.6s
[CV] END ..................C=1000, gamma=0.0001, kernel=rbf; total time=   4.7s
[CV] END ..................C=1000, gamma=0.0001, kernel=rbf; total time=   4.6s
[CV] END ..................C=1000, gamma=0.0001, kernel=rbf; total time=   5.7s
[CV] END ................C=1000, gamma=0.0001, kernel=linear; total time=   4.0s
[CV] END ................C=1000, gamma=0.0001, kernel=linear; total time=   4.0s
[CV] END ................C=1000, gamma=0.0001, kernel=linear; total time=   4.5s
[CV] END ................C=1000, gamma=0.0001, kernel=linear; total time=   4.4s
[CV] END ................C=1000, gamma=0.0001, kernel=linear; total time=   3.9s
Best Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
Tuned SVM - Accuracy: 0.7783333333333333, Precision: 0.6792058124336605, Recall: 0.5804970850745499, F1 Score: 0.
```

```
[ ]  import lightgbm as lgb
     from sklearn.model_selection import GridSearchCV

     # Define the parameter grid for LightGBM
     param_grid_lgb = {
         'n_estimators': [100, 200, 300],
         'learning_rate': [0.01, 0.1, 0.2],
         'max_depth': [3, 5, 7],
         'num_leaves': [31, 63, 127],
         'subsample': [0.6, 0.8, 1.0],
         'colsample_bytree': [0.6, 0.8, 1.0]
     }

     # Initialize GridSearchCV with LightGBM model and parameter grid
     grid_search_lgb = GridSearchCV(lgb.LGBMClassifier(), param_grid_lgb, refit=True, verbose=2, cv=3)

     # Perform the grid search
     grid_search_lgb.fit(X_train, y_train)

     # Get the best parameters
     print(f'Best Parameters for LightGBM: {grid_search_lgb.best_params_}')

     # Make predictions with the best model
     y_pred_lgb_tuned = grid_search_lgb.best_estimator_.predict(X_test)
     y_pred_proba_lgb_tuned = grid_search_lgb.best_estimator_.predict_proba(X_test)  # Get the predicted probabilities

     # Evaluate the best LightGBM model
     accuracy_lgb_tuned = accuracy_score(y_test, y_pred_lgb_tuned)
     precision_lgb_tuned = precision_score(y_test, y_pred_lgb_tuned, average='macro')
     recall_lgb_tuned = recall_score(y_test, y_pred_lgb_tuned, average='macro')
     f1_lgb_tuned = f1_score(y_test, y_pred_lgb_tuned, average='macro')
     mcc_lgb_tuned = matthews_corrcoef(y_test, y_pred_lgb_tuned)

     print(f'Tuned LightGBM - Accuracy: {accuracy_lgb_tuned}, Precision: {precision_lgb_tuned}, Recall: {recall_lgb_tunec

     # Compute Precision-Recall AUC for each class
     precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_pred_proba_lgb_tuned[:, 1], pos_label=1)  # For cl
     pr_auc_lgb_tuned = auc(recall_vals, precision_vals)

     print(f'Tuned LightGBM - Precision-Recall AUC: {pr_auc_lgb_tuned}')
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

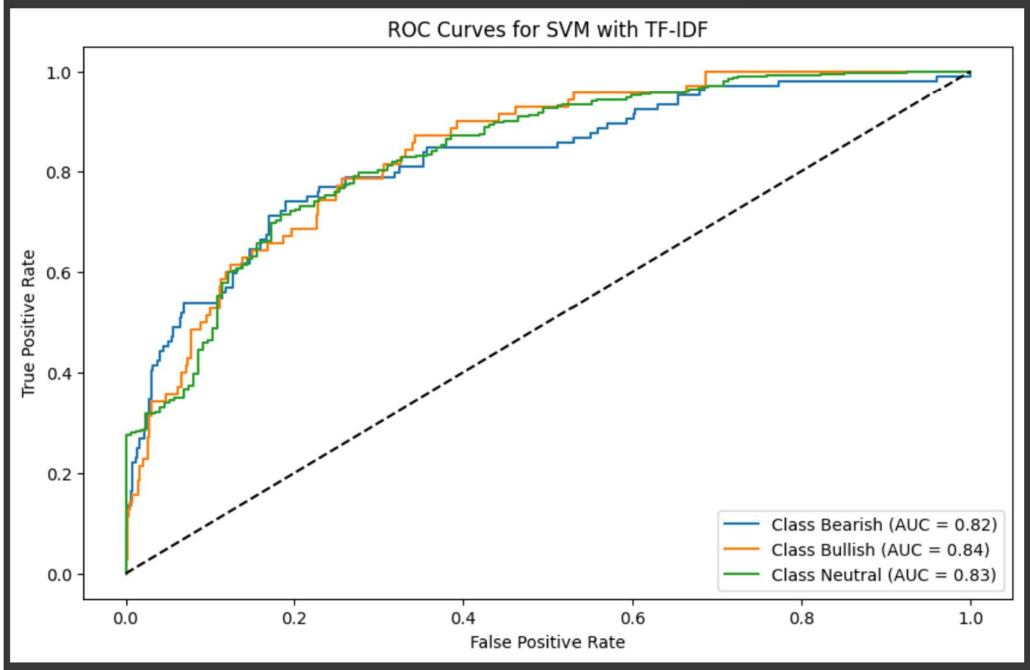
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Best Parameters for LightGBM: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100,
Tuned LightGBM - Accuracy: 0.7166666666666667, Precision: 0.5197079490557751, Recall: 0.39512287399611346, F1 Scor
Tuned LightGBM - Precision-Recall AUC: 0.21398654325469776
```

```python
from sklearn.ensemble import StackingClassifier

# Define the base models with their best estimators
estimators = [
    ('svm', grid_search.best_estimator_),
    ('lgb', grid_search_lgb.best_estimator_)
]

# Initialize StackingClassifier with a final estimator (meta-model), for example, a logistic regression classifie
from sklearn.linear_model import LogisticRegression
final_estimator = LogisticRegression()

stacking_clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator, cv=5)

# Fit the stacking model
stacking_clf.fit(X_train, y_train)

# Make predictions with the stacking model
y_pred_stack = stacking_clf.predict(X_test)
y_pred_proba_stack = stacking_clf.predict_proba(X_test)

# Evaluate the stacking model
accuracy_stack = accuracy_score(y_test, y_pred_stack)
precision_stack = precision_score(y_test, y_pred_stack, average='macro')
recall_stack = recall_score(y_test, y_pred_stack, average='macro')
f1_stack = f1_score(y_test, y_pred_stack, average='macro')
mcc_stack = matthews_corrcoef(y_test, y_pred_stack)

print(f'Stacking Model - Accuracy: {accuracy_stack}, Precision: {precision_stack}, Recall: {recall_stack}, F1 Sco

# Compute Precision-Recall AUC for each class
precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_pred_proba_stack[:, 1], pos_label=1)  # For cla
pr_auc_stack = auc(recall_vals, precision_vals)

print(f'Stacking Model - Precision-Recall AUC: {pr_auc_stack}')
```

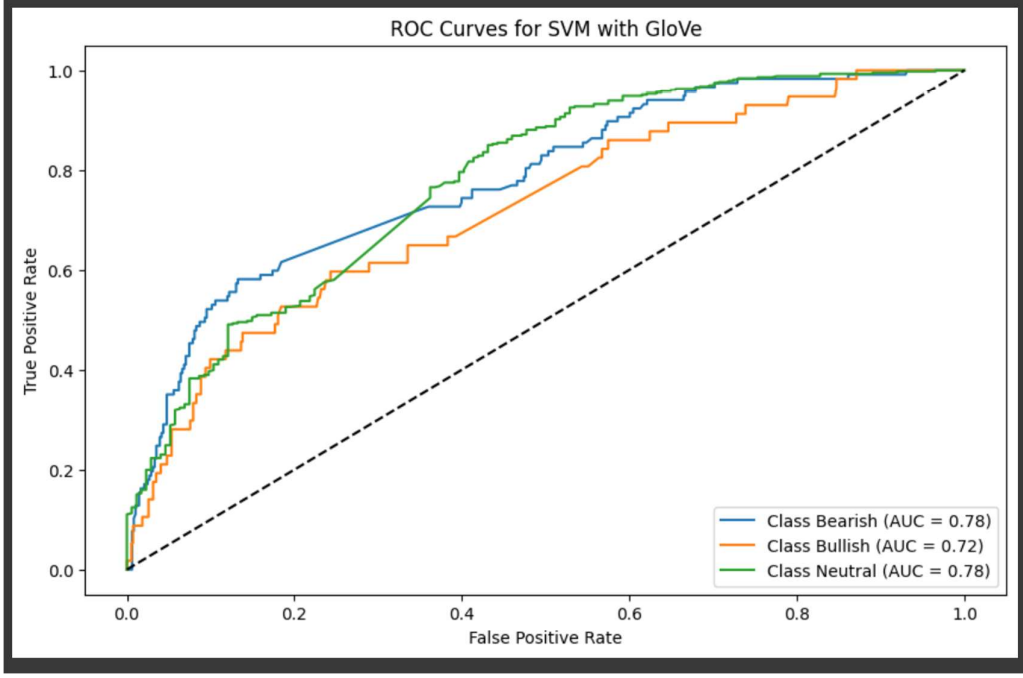
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Stacking Model - Accuracy: 0.7733333333333333, Precision: 0.6713197644649257, Recall: 0.550118660682041, F1 Score
Stacking Model - Precision-Recall AUC: 0.42545899309975294
```

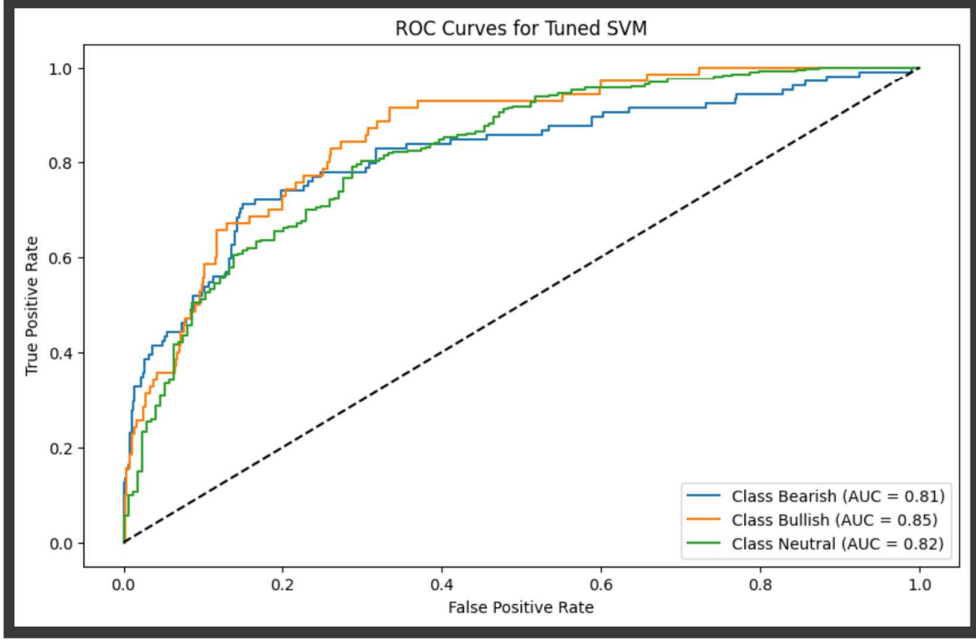
```python
# Calculate ROC curve and AUC for SVM with TF-IDF
y_probs_tfidf = svm_model_tfidf.predict_proba(X_test)
fpr_tfidf = {}
tpr_tfidf = {}
roc_auc_tfidf = {}

for i in range(len(label_encoder.classes_)):
    fpr_tfidf[i], tpr_tfidf[i], _ = roc_curve(y_test == i, y_probs_tfidf[:, i])
    roc_auc_tfidf[i] = auc(fpr_tfidf[i], tpr_tfidf[i])

# Plot ROC curves for SVM with TF-IDF
plt.figure(figsize=(10, 6))
for i in range(len(label_encoder.classes_)):
    plt.plot(fpr_tfidf[i], tpr_tfidf[i], label=f'Class {label_encoder.classes_[i]} (AUC = {roc_auc_tfidf[i]:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for SVM with TF-IDF')
plt.legend(loc='lower right')
plt.show()
```



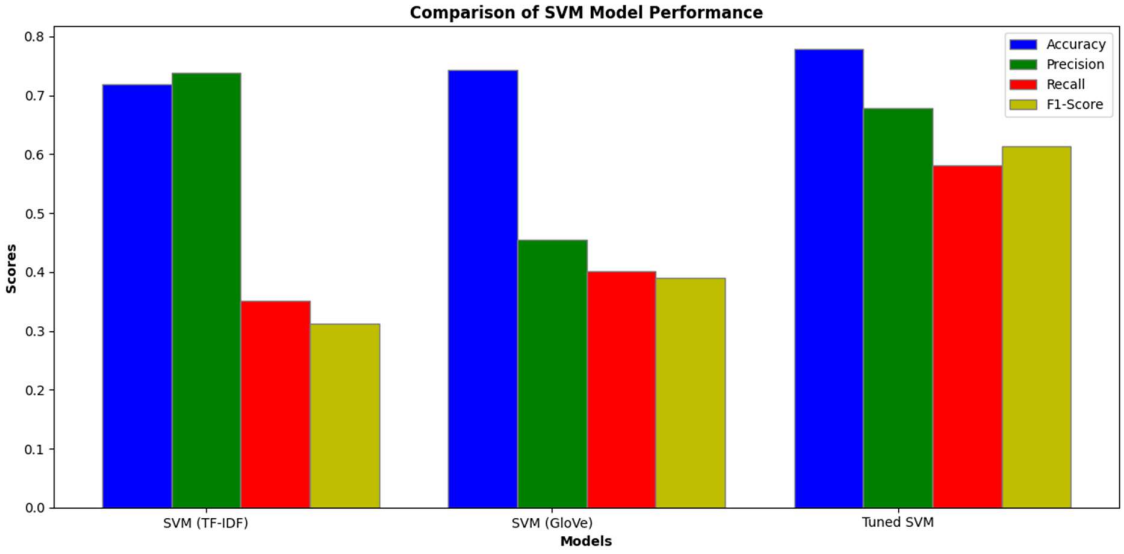
```python
# Calculate ROC curve and AUC for SVM with GloVe
y_probs_glove = svm_model_glove.predict_proba(X_test_glove)
fpr_glove = {}
tpr_glove = {}
roc_auc_glove = {}

for i in range(len(label_encoder.classes_)):
    fpr_glove[i], tpr_glove[i], _ = roc_curve(y_test_glove == i, y_probs_glove[:, i])
    roc_auc_glove[i] = auc(fpr_glove[i], tpr_glove[i])

# Plot ROC curves for SVM with GloVe
plt.figure(figsize=(10, 6))
for i in range(len(label_encoder.classes_)):
    plt.plot(fpr_glove[i], tpr_glove[i], label=f'Class {label_encoder.classes_[i]} (AUC = {roc_auc_glove[i]:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for SVM with GloVe')
plt.legend(loc='lower right')
plt.show()
```



```python
# Calculate ROC curve and AUC for Tuned SVM
y_probs_tuned = grid_search.best_estimator_.predict_proba(X_test)
fpr_tuned = {}
tpr_tuned = {}
roc_auc_tuned = {}

for i in range(len(label_encoder.classes_)):
    fpr_tuned[i], tpr_tuned[i], _ = roc_curve(y_test == i, y_probs_tuned[:, i])
    roc_auc_tuned[i] = auc(fpr_tuned[i], tpr_tuned[i])

# Plot ROC curves for Tuned SVM
plt.figure(figsize=(10, 6))
for i in range(len(label_encoder.classes_)):
    plt.plot(fpr_tuned[i], tpr_tuned[i], label=f'Class {label_encoder.classes_[i]} (AUC = {roc_auc_tuned[i]:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for Tuned SVM')
plt.legend(loc='lower right')
plt.show()
```



```python
# Data for the bar chart
models = ['SVM (TF-IDF)', 'SVM (GloVe)', 'Tuned SVM']
accuracy_scores = [accuracy_svm_tfidf, accuracy_svm_glove, accuracy_svm_tuned]
precision_scores = [precision_svm_tfidf, precision_svm_glove, precision_svm_tuned]
recall_scores = [recall_svm_tfidf, recall_svm_glove, recall_svm_tuned]
f1_scores = [f1_svm_tfidf, f1_svm_glove, f1_svm_tuned]

# Set the width of the bars
bar_width = 0.2

# Set the positions of the bars on the x-axis
r1 = np.arange(len(accuracy_scores))
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]
r4 = [x + bar_width for x in r3]

# Create the bar chart
plt.figure(figsize=(12, 6))

plt.bar(r1, accuracy_scores, color='b', width=bar_width, edgecolor='grey', label='Accuracy')
plt.bar(r2, precision_scores, color='g', width=bar_width, edgecolor='grey', label='Precision')
plt.bar(r3, recall_scores, color='r', width=bar_width, edgecolor='grey', label='Recall')
plt.bar(r4, f1_scores, color='y', width=bar_width, edgecolor='grey', label='F1-Score')

# Add labels, title, and legend
plt.xlabel('Models', fontweight='bold')
plt.ylabel('Scores', fontweight='bold')
plt.xticks([r + bar_width for r in range(len(accuracy_scores))], models)
plt.title('Comparison of SVM Model Performance', fontweight='bold')
plt.legend()

# Display the chart
plt.tight_layout()
plt.show()
```



```python
# Start timing for converting to DatasetDict and preprocessing
start_time = time.time()

def preprocess_text_for_bert(examples):
    return tokenizer(examples['text'], padding='max_length', truncation=True, max_length=64)

# Apply the preprocessing
dataset_dict_subset = dataset_lemmatized_with_thresholds_subset.map(preprocess_text_for_bert, batched=True)
dataset_dict_subset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])

# End timing for converting and preprocessing
end_time = time.time()
print(f"Time taken for converting and preprocessing: {end_time - start_time} seconds")

# Extract text from the dataset
corpus = dataset_lemmatized_with_thresholds_subset["train"]["text"]

# Initialize the TF-IDF vectorizer
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(corpus)

# Display TF-IDF matrix
print(X_tfidf.toarray())
print(vectorizer.get_feature_names_out())
```

```
Map: 100% 3000/3000 [00:00<00:00, 6311.31 examples/s]
Map: 100% 600/600 [00:00<00:00, 2146.59 examples/s]
Time taken for converting and preprocessing: 0.8785183429718018 seconds
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
['00' '000' '00pm' ... 'échelle' 'öltözik' 'übernahme']
```

```python
import numpy as np
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, matthews_corrcoef, precision_r
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer, DataCollato

# Initialize the FinBERT tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
model = AutoModelForSequenceClassification.from_pretrained("ProsusAI/finbert", num_labels=3)

# Preprocess the dataset for FinBERT
def preprocess_text_for_finbert(examples):
    encoding = tokenizer(examples['text'], padding='max_length', truncation=True, max_length=64)
    return encoding

# Assuming `dataset` is already loaded
dataset = dataset.map(preprocess_text_for_finbert, batched=True)

# Subsample the dataset to reduce the size
small_train_dataset = dataset['train'].shuffle(seed=42).select([i for i in list(range(0, 500))])
small_test_dataset = dataset['train'].shuffle(seed=42).select([i for i in list(range(500, 600))])

# Split the dataset into train and test
dataset_lemmatized = DatasetDict({'train': small_train_dataset, 'test': small_test_dataset})

# Set the format for PyTorch
dataset_lemmatized['train'].set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
dataset_lemmatized['test'].set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])

# Data collator
data_collator = DataCollatorWithPadding(tokenizer)

# Define metrics
def compute_metrics(p):
    preds = np.argmax(p.predictions, axis=1)
    precision = precision_score(p.label_ids, preds, average='macro')
    recall = recall_score(p.label_ids, preds, average='macro')
    f1 = f1_score(p.label_ids, preds, average='macro')
    mcc = matthews_corrcoef(p.label_ids, preds)
    precision_vals, recall_vals, _ = precision_recall_curve(p.label_ids, preds, pos_label=1)
    pr_auc = auc(recall_vals, precision_vals)
    return {
        'accuracy': accuracy_score(p.label_ids, preds),
        'precision': precision,
        'recall': recall,
        'f1': f1,
        'mcc': mcc,
        'pr_auc': pr_auc
    }
```

```python
# Training arguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=1,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    warmup_steps=200,
    weight_decay=0.01,
    logging_dir='./logs',
)

# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset_lemmatized['train'],
    eval_dataset=dataset_lemmatized['test'],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)

# Train the FinBERT model
trainer.train()

# Evaluate the FinBERT model
results = trainer.evaluate()

# Extract necessary metrics from results
accuracy_bert = results['eval_accuracy']
precision_bert = results['eval_precision']
recall_bert = results['eval_recall']
f1_bert = results['eval_f1']
mcc_bert = results['eval_mcc']
pr_auc_bert = results['eval_pr_auc']

# Display FinBERT metrics
print(f'FinBERT Accuracy: {accuracy_bert}')
print(f'FinBERT Precision (Macro): {precision_bert}')
print(f'FinBERT Recall (Macro): {recall_bert}')
print(f'FinBERT F1 Score (Macro): {f1_bert}')
print(f'FinBERT Matthews Correlation Coefficient: {mcc_bert}')
print(f'FinBERT Precision-Recall AUC: {pr_auc_bert}')
```

```
Map: 100% 7634/7634 [00:01<00:00, 5864.09 examples/s]
Map: 100% 1909/1909 [00:00<00:00, 3844.30 examples/s]
[125/125 09:24, Epoch 1/1]
Step  Training Loss
[25/25 00:24]
FinBERT Accuracy: 0.71
FinBERT Precision (Macro): 0.5283128636069813
FinBERT Recall (Macro): 0.500980500980501
FinBERT F1 Score (Macro): 0.5063336636164054
FinBERT Matthews Correlation Coefficient: 0.42705388093880614
FinBERT Precision-Recall AUC: 0.326451938865732
```

```python
from transformers import pipeline, AutoTokenizer

# Initialize the FinBERT pipeline and tokenizer
pipe = pipeline("text-classification", model="ProsusAI/finbert")
tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
```

```python
def preprocess_text_for_finbert(examples):
    encoding = tokenizer(examples['text'], padding='max_length', truncation=True, max_length=64)
    return encoding

dataset = dataset.map(preprocess_text_for_finbert, batched=True)
```

```
Map: 100% 7634/7634 [00:01<00:00, 6397.63 examples/s]
Map: 100% 1909/1909 [00:00<00:00, 6620.30 examples/s]
```

```python
small_train_dataset = dataset['train'].shuffle(seed=42).select([i for i in list(range(0, 500))])
small_test_dataset = dataset['train'].shuffle(seed=42).select([i for i in list(range(500, 600))])

dataset_lemmatized = {'train': small_train_dataset, 'test': small_test_dataset}

dataset_lemmatized['train'].set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
dataset_lemmatized['test'].set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
```

```python
# Example texts for prediction
texts = ["The market is looking bullish today.", "There is a lot of uncertainty in the market."]

# Use the pipeline for predictions
predictions = pipe(texts)

# Display predictions
for text, pred in zip(texts, predictions):
    print(f"Text: {text}")
    print(f"Prediction: {pred}")
```

```
Text: The market is looking bullish today.
Prediction: {'label': 'positive', 'score': 0.7777194976806641}
Text: There is a lot of uncertainty in the market.
Prediction: {'label': 'negative', 'score': 0.707795023918151 9}
```

SVM + Hybrid (SVM+FinBERT)

```
[ ]  # Analyze sentiments for the dataset using lemmatization
     def analyze_sentiment_with_lemmatization(example):
         preprocessed_text = preprocess_text_with_lemmatization(example["text"])
         result = pipe(preprocessed_text)
         example["predicted_sentiment"] = result[0]['label']
         return example

     # Apply the function to the dataset
     dataset_lemmatized = dataset.map(analyze_sentiment_with_lemmatization)

     # Create a DataFrame from the dataset
     df_train = pd.DataFrame(dataset_lemmatized["train"])

     # Extract text and labels from the dataset
     corpus = df_train["text"]
     labels = df_train["predicted_sentiment"]

     # Encode labels to numerical values
     label_encoder = LabelEncoder()
     y = label_encoder.fit_transform(labels)

     # Initialize the TF-IDF vectorizer
     vectorizer = TfidfVectorizer()
     X_tfidf = vectorizer.fit_transform(corpus)

     # Split the data into training and testing sets
     train_indices, test_indices, y_train, y_test = train_test_split(df_train.index, y, test_size=0.2, random_state=42)
     X_train = X_tfidf[train_indices]
     X_test = X_tfidf[test_indices]

     # Part 9: Train SVM Model with Class Weight Adjustment
     svm_model = SVC(class_weight='balanced')
     svm_model.fit(X_train, y_train)
     y_pred_svm = svm_model.predict(X_test)

     # Part 10: Hyperparameter Tuning and Evaluation

     # Define the parameter grid for GridSearchCV
     param_grid = {
         'C': [0.1, 1, 10, 100],
         'gamma': [1, 0.1, 0.01, 0.001],
         'kernel': ['rbf', 'linear']
     }
```

```
Map: 100% 7634/7634 [16:21<00:00, 8.84 examples/s]
Map: 100% 1909/1909 [03:53<00:00, 9.21 examples/s]
```

```
[ ] # Initialize GridSearchCV with SVM model and parameter grid
    grid_search = GridSearchCV(SVC(class_weight='balanced'), param_grid, refit=True, verbose=2, cv=3)

    # Perform the grid search
    grid_search.fit(X_train, y_train)

    # Get the best parameters
    print(f'Best Parameters: {grid_search.best_params_}')

    # Make predictions with the best model
    y_pred_svm = grid_search.best_estimator_.predict(X_test)
```

```
[CV] END .....................C=1, gamma=0.1, kernel=rbf; total time=   6.0s
[CV] END .....................C=1, gamma=0.1, kernel=rbf; total time=   6.4s
[CV] END .....................C=1, gamma=0.1, kernel=linear; total time=   4.2s
[CV] END .....................C=1, gamma=0.1, kernel=linear; total time=   4.7s
[CV] END .....................C=1, gamma=0.1, kernel=linear; total time=   4.8s
[CV] END ....................C=1, gamma=0.01, kernel=rbf; total time=   5.8s
[CV] END ....................C=1, gamma=0.01, kernel=rbf; total time=   6.9s
[CV] END ....................C=1, gamma=0.01, kernel=rbf; total time=   5.8s
[CV] END .....................C=1, gamma=0.01, kernel=linear; total time=   4.3s
[CV] END .....................C=1, gamma=0.01, kernel=linear; total time=   5.1s
[CV] END .....................C=1, gamma=0.01, kernel=linear; total time=   4.2s
[CV] END ....................C=1, gamma=0.001, kernel=rbf; total time=   6.2s
[CV] END ....................C=1, gamma=0.001, kernel=rbf; total time=   6.6s
[CV] END ....................C=1, gamma=0.001, kernel=rbf; total time=   5.8s
[CV] END .....................C=1, gamma=0.001, kernel=linear; total time=   5.2s
[CV] END .....................C=1, gamma=0.001, kernel=linear; total time=   4.4s
[CV] END .....................C=1, gamma=0.001, kernel=linear; total time=   4.2s
[CV] END ......................C=10, gamma=1, kernel=rbf; total time=   6.8s
[CV] END ......................C=10, gamma=1, kernel=rbf; total time=   5.6s
[CV] END ......................C=10, gamma=1, kernel=rbf; total time=   6.3s
[CV] END .....................C=10, gamma=1, kernel=linear; total time=   4.7s
[CV] END .....................C=10, gamma=1, kernel=linear; total time=   4.4s
[CV] END .....................C=10, gamma=1, kernel=linear; total time=   5.0s
[CV] END ....................C=10, gamma=0.1, kernel=rbf; total time=   5.4s
[CV] END ....................C=10, gamma=0.1, kernel=rbf; total time=   5.0s
[CV] END ....................C=10, gamma=0.1, kernel=rbf; total time=   6.1s
```

```
] # Evaluate the best SVM model
  accuracy_svm = accuracy_score(y_test, y_pred_svm)
  precision_svm = precision_score(y_test, y_pred_svm, average='macro')
  recall_svm = recall_score(y_test, y_pred_svm, average='macro')
  f1_svm = f1_score(y_test, y_pred_svm, average='macro')
  mcc_svm = matthews_corrcoef(y_test, y_pred_svm)

  # Display SVM metrics
  print(f'SVM Accuracy: {accuracy_svm}')
  print(f'SVM Precision (Macro): {precision_svm}')
  print(f'SVM Recall (Macro): {recall_svm}')
  print(f'SVM F1 Score (Macro): {f1_svm}')
  print(f'SVM Matthews Correlation Coefficient: {mcc_svm}')
```

```
SVM Accuracy: 0.7753765553372626
SVM Precision (Macro): 0.6864813011851897
SVM Recall (Macro): 0.5788989713148416
SVM F1 Score (Macro): 0.6151139248584611
SVM Matthews Correlation Coefficient: 0.4535832089112883
```

```python
# PUT SVM INTO HYBRID

# Function to ensure predictions have the correct shape
def ensure_correct_shape(preds, num_classes=3):
    if preds.ndim == 1:
        preds = np.expand_dims(preds, axis=-1)
    if preds.shape[1] != num_classes:
        preds = np.tile(preds, (1, num_classes // preds.shape[1]))
    return preds

# Example of a simple hybrid model by averaging predictions
def hybrid_model_predictions(preds_1, preds_2):
    combined_preds = (preds_1 + preds_2) / 2
    return np.argmax(combined_preds, axis=1)

# Hybrid Model: FinBERT + SVM
preds_svm = grid_search.best_estimator_.decision_function(X_test)
preds_svm = ensure_correct_shape(preds_svm, num_classes=3)

# Get FinBERT predictions and transform them appropriately
test_texts = df_train['text'].iloc[test_indices]  # Ensure the same number of samples
finbert_results = pipe(list(test_texts))
preds_finbert = np.array([res['score'] for res in finbert_results])
preds_finbert = ensure_correct_shape(preds_finbert, num_classes=3)

# Ensure the shapes are compatible
if preds_svm.shape != preds_finbert.shape:
    raise ValueError(f"Shape mismatch: preds_svm.shape={preds_svm.shape}, preds_finbert.shape={preds_finbert.shap

hybrid_preds = hybrid_model_predictions(preds_svm, preds_finbert)

# Evaluate Hybrid Model
accuracy_hybrid = accuracy_score(y_test, hybrid_preds)
precision_hybrid = precision_score(y_test, hybrid_preds, average='macro')
recall_hybrid = recall_score(y_test, hybrid_preds, average='macro')
f1_hybrid = f1_score(y_test, hybrid_preds, average='macro')
mcc_hybrid = matthews_corrcoef(y_test, hybrid_preds)

# Display Hybrid Model metrics
print(f'Hybrid Model Accuracy: {accuracy_hybrid}')
print(f'Hybrid Model Precision (Macro): {precision_hybrid}')
print(f'Hybrid Model Recall (Macro): {recall_hybrid}')
print(f'Hybrid Model F1 Score (Macro): {f1_hybrid}')
print(f'Hybrid Model Matthews Correlation Coefficient: {mcc_hybrid}')
```

```
Hybrid Model Accuracy: 0.7760314341846758
Hybrid Model Precision (Macro): 0.6871016612389149
Hybrid Model Recall (Macro): 0.5795932687967649
Hybrid Model F1 Score (Macro): 0.6161669805614832
Hybrid Model Matthews Correlation Coefficient: 0.4546259878334739
```

13: Prepare Data for LSTM

```
pip install numpy scikit-learn keras
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn)
```

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

# Tokenize the text data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(corpus)
sequences = tokenizer.texts_to_sequences(corpus)

# Pad the sequences
max_length = max(len(seq) for seq in sequences)
X_padded = pad_sequences(sequences, maxlen=max_length, padding='post')

# Split the data into training and testing sets
train_indices, test_indices, y_train, y_test = train_test_split(
    np.arange(len(labels)), labels, test_size=0.2, random_state=42, stratify=labels
)
X_train_padded = X_padded[train_indices]
X_test_padded = X_padded[test_indices]
```

```
# Encode labels to numerical values
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)

# Print shapes to verify
print(f'X_train_padded shape: {X_train_padded.shape}')
print(f'X_test_padded shape: {X_test_padded.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'y_test shape: {y_test.shape}')
```

```
X_train_padded shape: (6107, 35)
X_test_padded shape: (1527, 35)
y_train shape: (6107,)
y_test shape: (1527,)
```

```
import numpy as np
from keras.models import Sequential, Model
from keras.layers import Input, Embedding, Bidirectional, LSTM, Dropout, Dense
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
from keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, matthews_corrcoef, precision_

# Load GloVe pre-trained vectors
glove_vectors = api.load("glove-wiki-gigaword-100")
embedding_matrix = np.zeros((len(tokenizer.word_index) + 1, 100))
for word, i in tokenizer.word_index.items():
    if word in glove_vectors:
        embedding_vector = glove_vectors[word]
        embedding_matrix[i] = embedding_vector

# Define the LSTM model with improvements
def create_lstm_model(input_length, vocab_size, embedding_matrix):
    input_ = Input(shape=(input_length,))
    x = Embedding(input_dim=vocab_size, output_dim=100, weights=[embedding_matrix], input_length=input_length, tra
    x = Bidirectional(LSTM(units=100, return_sequences=True))(x)
    x = Dropout(0.2)(x)
    x = Bidirectional(LSTM(units=100))(x)
    x = Dropout(0.2)(x)
    x = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=input_, outputs=x)
    model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])
    return model

input_length = max_length
vocab_size = len(tokenizer.word_index) + 1

lstm_model = create_lstm_model(input_length, vocab_size, embedding_matrix)
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = lstm_model.fit(
    X_train_padded, y_train,
    epochs=20,
    batch_size=32,
    validation_data=(X_test_padded, y_test),
    callbacks=[early_stopping]
)

# Make predictions
y_pred_lstm = lstm_model.predict(X_test_padded)
y_pred_lstm = (y_pred_lstm > 0.5).astype(int)
```

```python
# Evaluate the LSTM model
accuracy_lstm = accuracy_score(y_test, y_pred_lstm)
precision_lstm = precision_score(y_test, y_pred_lstm, average='macro', zero_division=0)
recall_lstm = recall_score(y_test, y_pred_lstm, average='macro')
f1_lstm = f1_score(y_test, y_pred_lstm, average='macro')
mcc_lstm = matthews_corrcoef(y_test, y_pred_lstm)

# Calculate Precision-Recall AUC for LSTM
precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_pred_lstm, pos_label=1)
pr_auc_lstm = auc(recall_vals, precision_vals)

# Display LSTM metrics
print(f'LSTM Accuracy: {accuracy_lstm}')
print(f'LSTM Precision (Macro): {precision_lstm}')
print(f'LSTM Recall (Macro): {recall_lstm}')
print(f'LSTM F1 Score (Macro): {f1_lstm}')
print(f'LSTM Matthews Correlation Coefficient: {mcc_lstm}')
print(f'LSTM Precision-Recall AUC: {pr_auc_lstm}')
```

```
Epoch 1/20
191/191 [==============================] - 41s 182ms/step - loss: 0.0494 - accuracy: 0.7066 - val_loss: 0.0650 - va
Epoch 2/20
191/191 [==============================] - 32s 169ms/step - loss: -0.5943 - accuracy: 0.6995 - val_loss: -1.0230 -
Epoch 3/20
191/191 [==============================] - 33s 174ms/step - loss: -2.0749 - accuracy: 0.7100 - val_loss: -1.5726 -
Epoch 4/20
191/191 [==============================] - 34s 180ms/step - loss: 0.3170 - accuracy: 0.6900 - val_loss: 0.3069 - va
Epoch 5/20
191/191 [==============================] - 32s 168ms/step - loss: 0.1785 - accuracy: 0.6963 - val_loss: -0.0968 - va
Epoch 6/20
191/191 [==============================] - 32s 169ms/step - loss: -0.4405 - accuracy: 0.7084 - val_loss: -0.3139 -
Epoch 7/20
191/191 [==============================] - 35s 183ms/step - loss: -1.4594 - accuracy: 0.7143 - val_loss: -1.8920 -
Epoch 8/20
191/191 [==============================] - 33s 172ms/step - loss: -2.6429 - accuracy: 0.6985 - val_loss: -3.2579 -
Epoch 9/20
191/191 [==============================] - 33s 170ms/step - loss: -4.2957 - accuracy: 0.7043 - val_loss: -4.3055 -
Epoch 10/20
191/191 [==============================] - 32s 170ms/step - loss: -5.8475 - accuracy: 0.6967 - val_loss: -3.0366 -
Epoch 11/20
191/191 [==============================] - 35s 183ms/step - loss: -6.7417 - accuracy: 0.7038 - val_loss: -6.3274 -
Epoch 12/20
191/191 [==============================] - 32s 168ms/step - loss: -8.9111 - accuracy: 0.7100 - val_loss: -7.5755 -
Epoch 13/20
191/191 [==============================] - 33s 172ms/step - loss: -10.2011 - accuracy: 0.7036 - val_loss: -7.9393 -
Epoch 14/20
191/191 [==============================] - 32s 168ms/step - loss: -12.4795 - accuracy: 0.6966 - val_loss: -8.4383 -
Epoch 15/20
191/191 [==============================] - 35s 185ms/step - loss: -14.1232 - accuracy: 0.7061 - val_loss: -9.1503 -
Epoch 16/20
```

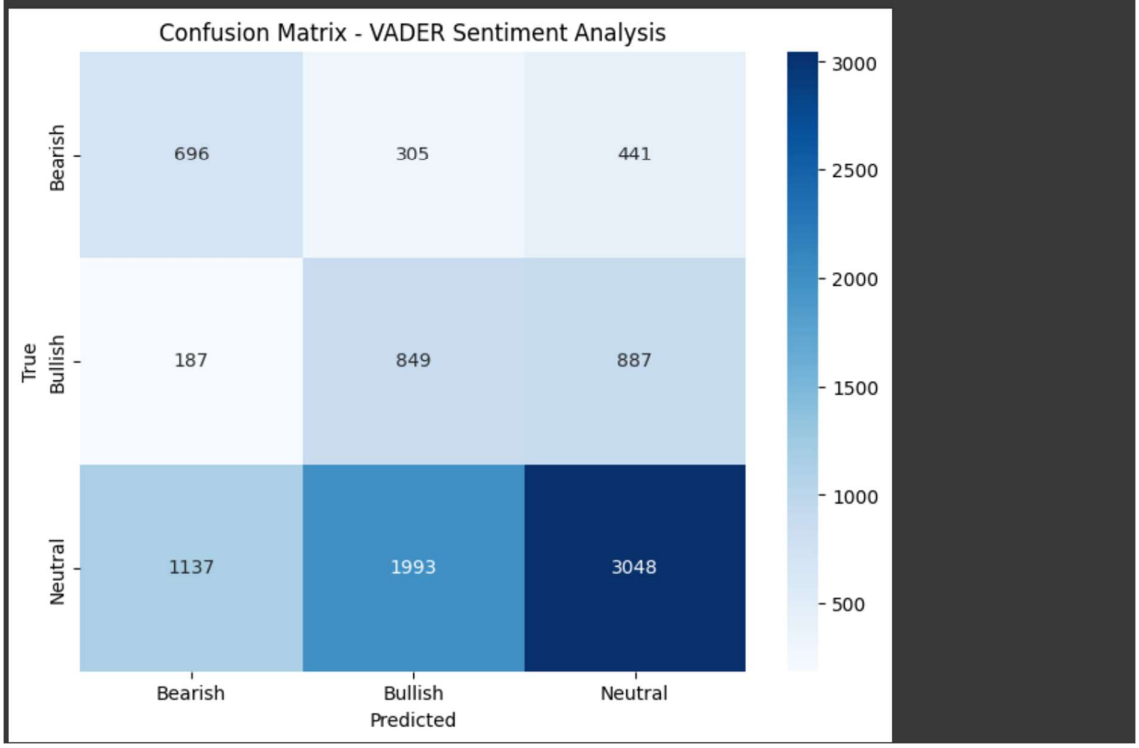
```
191/191 [==============================] - 32s 170ms/step - loss: -5.8475 - accuracy: 0.6967 - val_loss: -3.0366
Epoch 11/20
191/191 [==============================] - 35s 183ms/step - loss: -6.7417 - accuracy: 0.7038 - val_loss: -6.3274
Epoch 12/20
191/191 [==============================] - 32s 168ms/step - loss: -8.9111 - accuracy: 0.7100 - val_loss: -7.5755
Epoch 13/20
191/191 [==============================] - 33s 172ms/step - loss: -10.2011 - accuracy: 0.7036 - val_loss: -7.939
Epoch 14/20
191/191 [==============================] - 32s 168ms/step - loss: -12.4795 - accuracy: 0.6966 - val_loss: -8.438
Epoch 15/20
191/191 [==============================] - 35s 185ms/step - loss: -14.1232 - accuracy: 0.7061 - val_loss: -9.150
Epoch 16/20
191/191 [==============================] - 32s 170ms/step - loss: -16.1528 - accuracy: 0.7156 - val_loss: -11.75
Epoch 17/20
191/191 [==============================] - 32s 165ms/step - loss: -17.7740 - accuracy: 0.7092 - val_loss: -12.32
Epoch 18/20
191/191 [==============================] - 33s 171ms/step - loss: -20.1851 - accuracy: 0.7156 - val_loss: -13.64
Epoch 19/20
191/191 [==============================] - 33s 171ms/step - loss: -21.6822 - accuracy: 0.7190 - val_loss: -15.02
Epoch 20/20
191/191 [==============================] - 36s 186ms/step - loss: -24.5816 - accuracy: 0.7311 - val_loss: -16.28
48/48 [==============================] - 3s 42ms/step
LSTM Accuracy: 0.7301899148657498
LSTM Precision (Macro): 0.5046464646464647
LSTM Recall (Macro): 0.3992678382586135
LSTM F1 Score (Macro): 0.3897411555666484
LSTM Matthews Correlation Coefficient: 0.2668484741147831
LSTM Precision-Recall AUC: 0.861812627126776
```

```python
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, matthews_corrcoef
from datasets import load_dataset

# Load your dataset
dataset = load_dataset("zeroshot/twitter-financial-news-sentiment")

# Extract the text column from train split
texts = dataset['train']['text']

# Initialize VADER sentiment intensity analyzer
analyzer = SentimentIntensityAnalyzer()

# Function to classify sentiment using VADER
def classify_sentiment_vader(text):
    sentiment_score = analyzer.polarity_scores(text)
    compound = sentiment_score['compound']
    if compound >= 0.05:
        return 'Bullish'
    elif compound <= -0.05:
        return 'Bearish'
    else:
        return 'Neutral'

# Apply VADER sentiment analysis to your dataset
vader_sentiments = [classify_sentiment_vader(text) for text in texts]

# Add the VADER sentiments to your DataFrame
dataset['train'] = dataset['train'].add_column('vader_sentiment', vader_sentiments)

# Assuming your dataset has a 'label' column with the true sentiment labels
true_labels = dataset['train']['label']

# Map numerical labels to string labels
label_mapping = {0: 'Bearish', 1: 'Bullish', 2: 'Neutral'}
true_labels_mapped = [label_mapping[label] for label in true_labels]

# Function to evaluate VADER sentiment analysis
def evaluate_vader_sentiment(y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='macro')
    recall = recall_score(y_true, y_pred, average='macro')
    f1 = f1_score(y_true, y_pred, average='macro')
    mcc = matthews_corrcoef(y_true, y_pred)
    return accuracy, precision, recall, f1, mcc
```

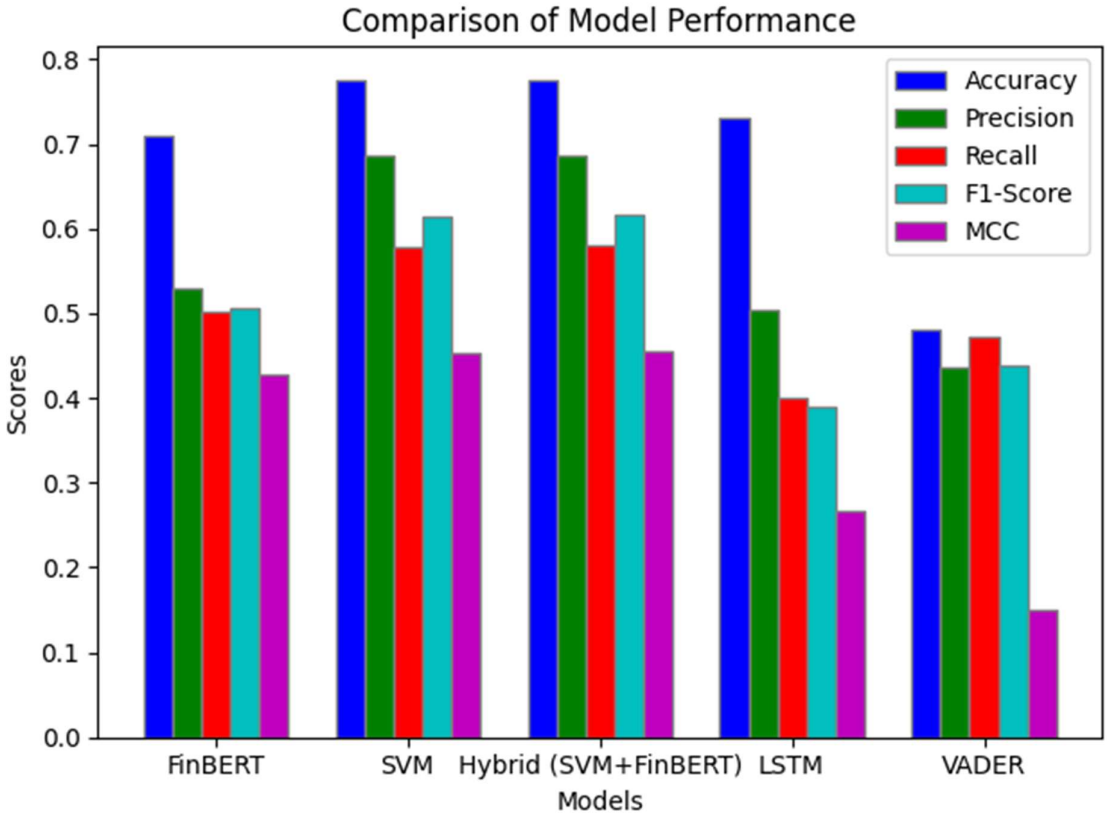
```python
# Evaluate VADER sentiment analysis
accuracy_vader, precision_vader, recall_vader, f1_vader, mcc_vader = evaluate_vader_sentiment(true_labels_mapped,

# Display VADER metrics
print(f'VADER Accuracy: {accuracy_vader}')
print(f'VADER Precision (Macro): {precision_vader}')
print(f'VADER Recall (Macro): {recall_vader}')
print(f'VADER F1 Score (Macro): {f1_vader}')
print(f'VADER Matthews Correlation Coefficient: {mcc_vader}')
```

```
VADER Accuracy: 0.4812951901917636
VADER Precision (Macro): 0.4369539024125119
VADER Recall (Macro): 0.47250805869335594
VADER F1 Score (Macro): 0.4381972916383072
VADER Matthews Correlation Coefficient: 0.14988390394199785
```

```python
# Confusion matrix for VADER

import matplotlib.pyplot as plt
# Create confusion matrix
cm_vader = confusion_matrix(true_labels_mapped, vader_sentiments)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_vader, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Bearish', 'Bullish', 'Neutral'],
            yticklabels=['Bearish', 'Bullish', 'Neutral'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - VADER Sentiment Analysis')
plt.show()
```



```python
import matplotlib.pyplot as plt
import numpy as np
# Data for the bar chart
models = ['FinBERT', 'SVM', 'Hybrid (SVM+FinBERT)', 'LSTM', 'VADER']
accuracy_scores = [accuracy_bert, accuracy_svm, accuracy_hybrid, accuracy_lstm, accuracy_vader]
precision_scores = [precision_bert, precision_svm, precision_hybrid, precision_lstm, precision_vader]
recall_scores = [recall_bert, recall_svm, recall_hybrid, recall_lstm, recall_vader]
f1_scores = [f1_bert, f1_svm, f1_hybrid, f1_lstm, f1_vader]
mcc_scores = [mcc_bert, mcc_svm, mcc_hybrid, mcc_lstm, mcc_vader]

# Set the width of the bars
bar_width = 0.15

# Set the positions of the bars on the x-axis
r1 = np.arange(len(models))
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]
r4 = [x + bar_width for x in r3]
r5 = [x + bar_width for x in r4]

# Create the bar chart
plt.bar(r1, accuracy_scores, color='b', width=bar_width, edgecolor='grey', label='Accuracy')
plt.bar(r2, precision_scores, color='g', width=bar_width, edgecolor='grey', label='Precision')
plt.bar(r3, recall_scores, color='r', width=bar_width, edgecolor='grey', label='Recall')
plt.bar(r4, f1_scores, color='c', width=bar_width, edgecolor='grey', label='F1-Score')
plt.bar(r5, mcc_scores, color='m', width=bar_width, edgecolor='grey', label='MCC')

# Add labels, title, and legend
plt.xlabel('Models')
plt.ylabel('Scores')
plt.title('Comparison of Model Performance')
plt.xticks([r + bar_width*2 for r in range(len(models))], models)
plt.legend()

# Display the chart
plt.tight_layout()
plt.show()
```


Comparison of Model Performance
Accuracy
Precision
Recall
F1-Score
MCC
Scores
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
FinBERT
SVM
Hybrid (SVM+FinBERT)
LSTM
VADER
Models

Appendix VII: GUI Codes

```
import gradio as gr
import pandas as pd
import re
import nltk
import numpy as np
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from transformers import pipeline, AutoTokenizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import io
from PIL import Image

# Download NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
# Download NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

# Initialize the FinBERT pipeline and tokenizer
pipe = pipeline("text-classification", model="ProsusAI/finbert")
tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")

# Get the list of stop words
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Initialize vectorizer and SVM model
vectorizer = TfidfVectorizer()
svm_model = SVC(probability=True)

# Dummy data for SVM model initialization (for demonstration purposes)
# Replace this with actual training data
texts = ["stock market is bullish", "stock market is bearish", "neutral market today"]
labels = ["bullish", "bearish", "neutral"]

# Preprocess the dummy data
preprocessed_texts = [re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE).lower() for text in texts]
X = vectorizer.fit_transform(preprocessed_texts)
y = LabelEncoder().fit_transform(labels)
svm_model.fit(X, y)

# Preprocess text
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    text = re.sub(r'\@\w+|\#', '', text)
    text = re.sub(r'[^A-Za-z0-9 ]+', '', text)
    words = text.split()
    words = [word for word in words if word not in stop_words]
    words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(words)
```

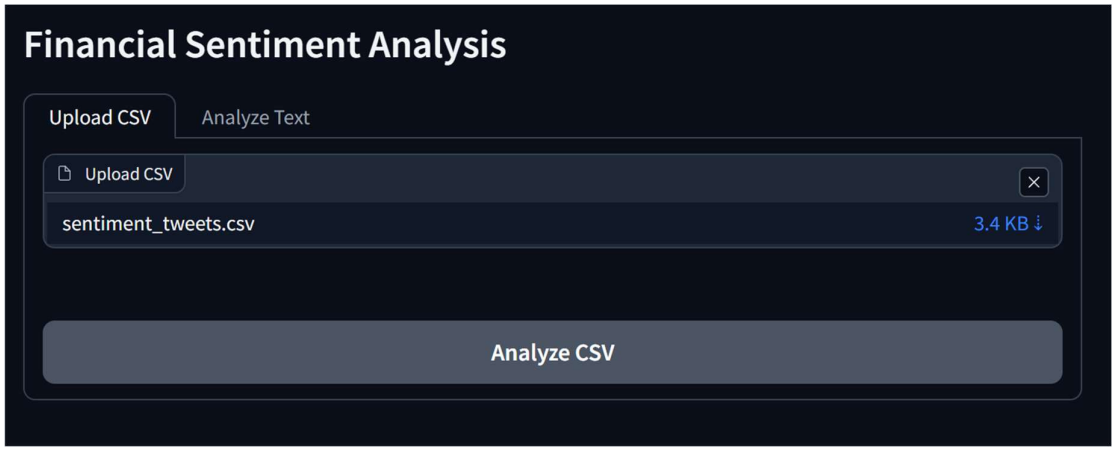
```python
# Hybrid model for sentiment classification
def hybrid_model(text):
    try:
        # Preprocess the text
        preprocessed_text = preprocess_text(text)

        # SVM prediction
        tfidf_text = vectorizer.transform([preprocessed_text])
        preds_svm = svm_model.predict_proba(tfidf_text)[0]

        # FinBERT prediction
        finbert_result = pipe(preprocessed_text)
        preds_finbert = np.zeros(3)
        for res in finbert_result:
            if res['label'] == 'positive':
                preds_finbert[2] += res['score']
            elif res['label'] == 'neutral':
                preds_finbert[1] += res['score']
            else:
                preds_finbert[0] += res['score']

        # Normalize FinBERT scores to sum to 1
        if preds_finbert.sum() > 0:
            preds_finbert /= preds_finbert.sum()

        # Hybrid prediction
        combined_preds = (preds_svm + preds_finbert) / 2
        sentiment_idx = np.argmax(combined_preds)

        # Map the prediction to sentiment labels
        sentiment_map = {0: "bearish", 1: "neutral", 2: "bullish"}
        return sentiment_map[sentiment_idx]
    except Exception as e:
        print(f"Error in hybrid_model: {e}")
        return "Error"
```

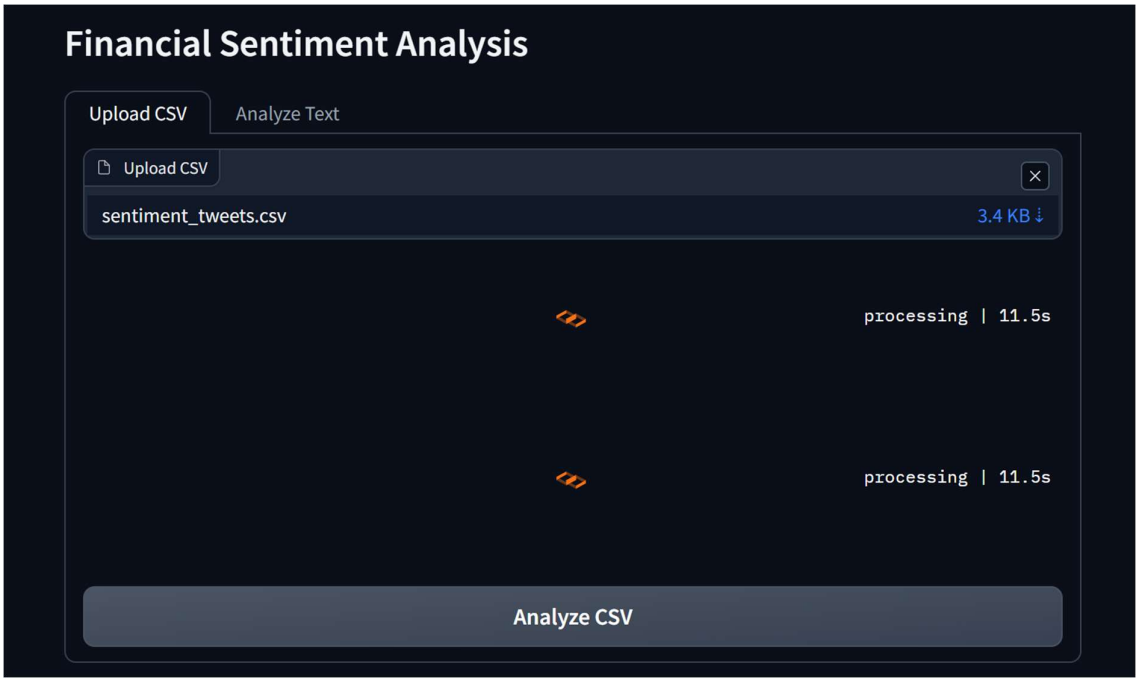
```python
# Process the uploaded CSV file and generate sentiment distribution table and chart
def process_csv(file_path):
    try:
        df = pd.read_csv(file_path)
        if 'text' not in df.columns:
            return "CSV must contain a 'text' column for sentiment analysis.", None, None

        df['predicted_sentiment'] = df['text'].apply(hybrid_model)

        # Generate sentiment distribution table
        sentiment_counts = df['predicted_sentiment'].value_counts()
        table_html = sentiment_counts.to_frame().reset_index()
        table_html.columns = ['Sentiment', 'Count']
        table_html = table_html.to_html(index=False)

        # Generate sentiment distribution chart
        fig, ax = plt.subplots(figsize=(3, 2))  # Further adjusted figure size
        sentiment_counts.plot(kind='bar', ax=ax)
        ax.set_xlabel('Sentiment', fontsize=8)
        ax.set_ylabel('Count', fontsize=8)
        ax.set_title('Sentiment Distribution', fontsize=10)
        ax.tick_params(axis='both', which='major', labelsize=6)
        plt.tight_layout()
        buf = io.BytesIO()
        plt.savefig(buf, format='png')
        buf.seek(0)
        img = Image.open(buf)

        return df.to_html(), table_html, img
    except Exception as e:
        print(f"Error in process_csv: {e}")
        return f"Error processing CSV: {e}", None, None

# Interface for file upload and text input
def upload_page(file):
    df_html, table_html, img = process_csv(file.name)
    return df_html, table_html, img

def analyze_text(text):
    hybrid_result = hybrid_model(text)
    return hybrid_result
```

```python
# Create Gradio interface with tabs for file upload and text input
with gr.Blocks() as demo:
    gr.Markdown("# Financial Sentiment Analysis")
    with gr.Tab("Upload CSV"):
        file_input = gr.File(label="Upload CSV", type="filepath")
        file_output = gr.HTML()
        table_output = gr.HTML()
        img_output = gr.Image()
        file_button = gr.Button("Analyze CSV")
        file_button.click(upload_page, inputs=file_input, outputs=[file_output, table_output, img_output])
    with gr.Tab("Analyze Text"):
        text_input = gr.Textbox(label="Enter text to analyze sentiment")
        text_output = gr.Textbox(label="Sentiment Classification")
        text_button = gr.Button("Analyze Text")
        text_button.click(analyze_text, inputs=text_input, outputs=text_output)

# Launch the interface
demo.launch()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setti

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: https://9e1660a321cd21836b.gradio.live
```

Appendix VIII: Prototype GUI Interface


Financial Sentiment Analysis
Upload CSV
Analyze Text
Upload CSV
sentiment_tweets.csv
3.4 KB
Analyze CSV


Financial Sentiment Analysis
Upload CSV
Analyze Text
Upload CSV
sentiment_tweets.csv
3.4 KB
processing | 11.5s
processing | 11.5s
Analyze CSV

Financial Sentiment Analysis

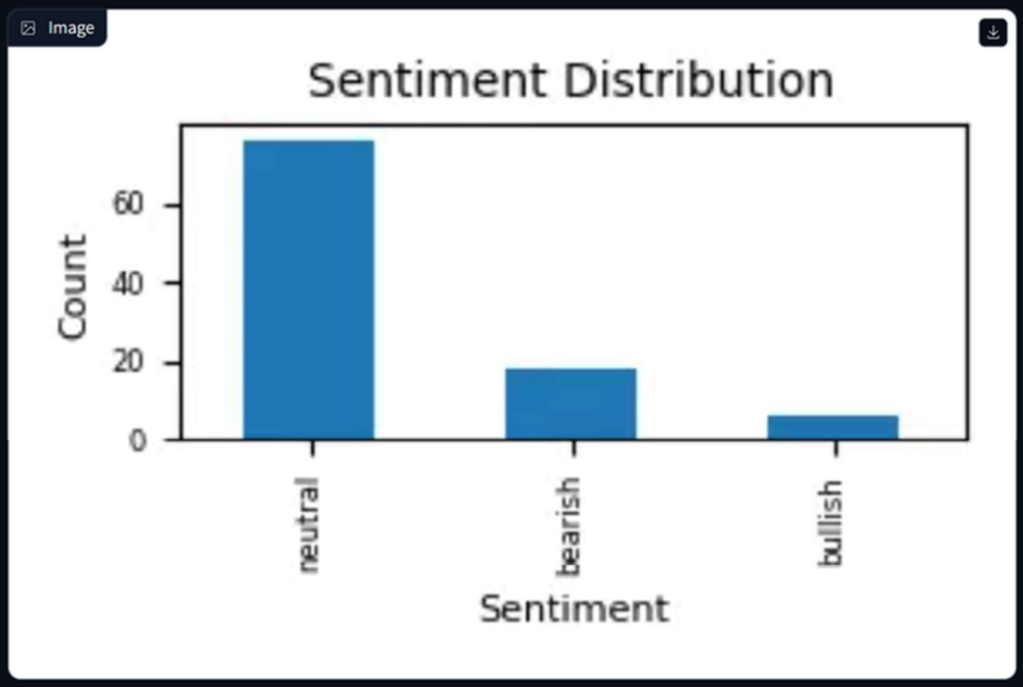
Upload CSV | Analyze Text

Upload CSV

sentiment_tweets.csv 3.4 KB

	text	predicted_sentiment
0	No major movements in the market.	neutral
1	Strong earnings reports are driving stocks up.	bullish
30	Profits are soaring in the market.	bearish
31	The economy is unchanged.	neutral
32	The stock market is stable.	neutral
33	The market remains neutral.	neutral
34	No major movements in the market.	neutral
35	The financial outlook is neutral.	neutral
36	The market is in a holding pattern.	neutral
37	No significant changes in the market today.	neutral
38	The financial outlook is negative.	neutral
39	The stock market is falling.	neutral
40	Weak earnings reports are driving stocks down.	bearish
41	The market is in a holding pattern.	neutral

Sentiment	Count
neutral	76
bearish	18
bullish	6



Analyze CSV

Use via API · Built with Gradio

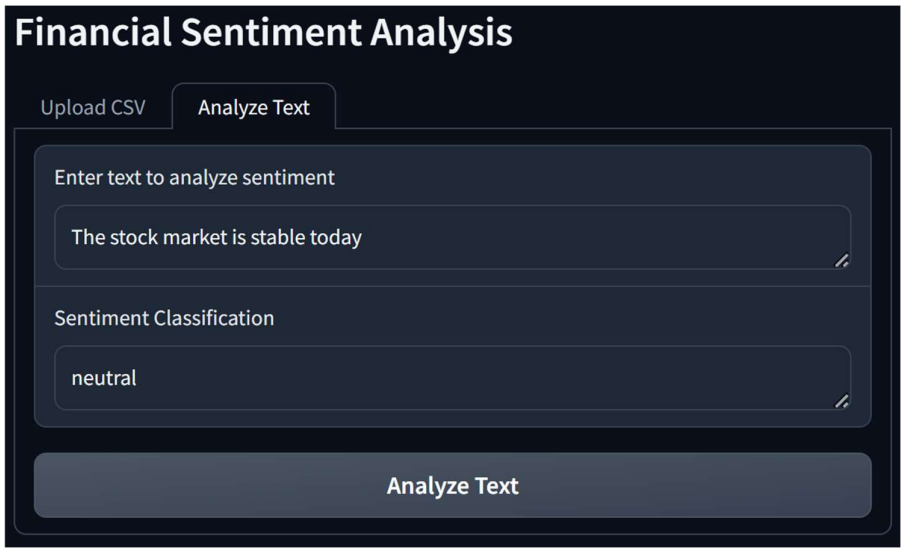
Financial Sentiment Analysis

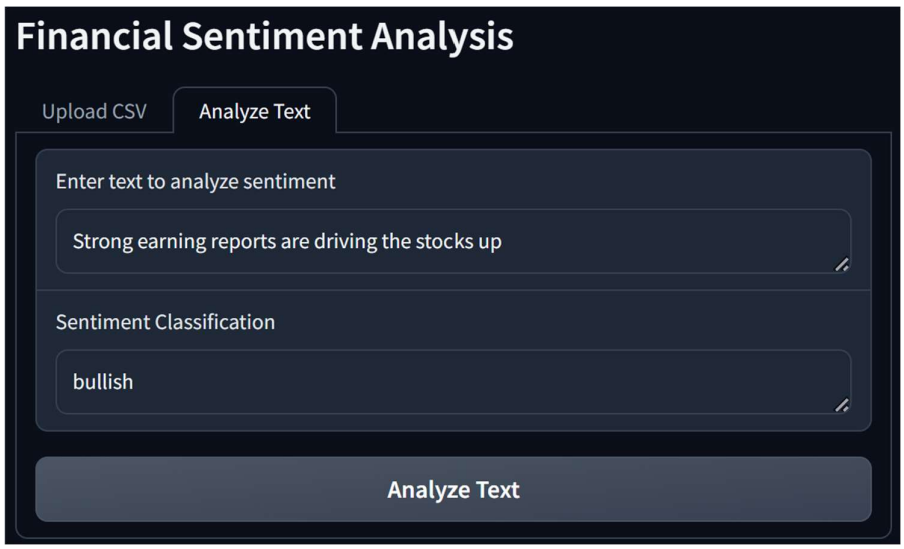
Upload CSV | Analyze Text

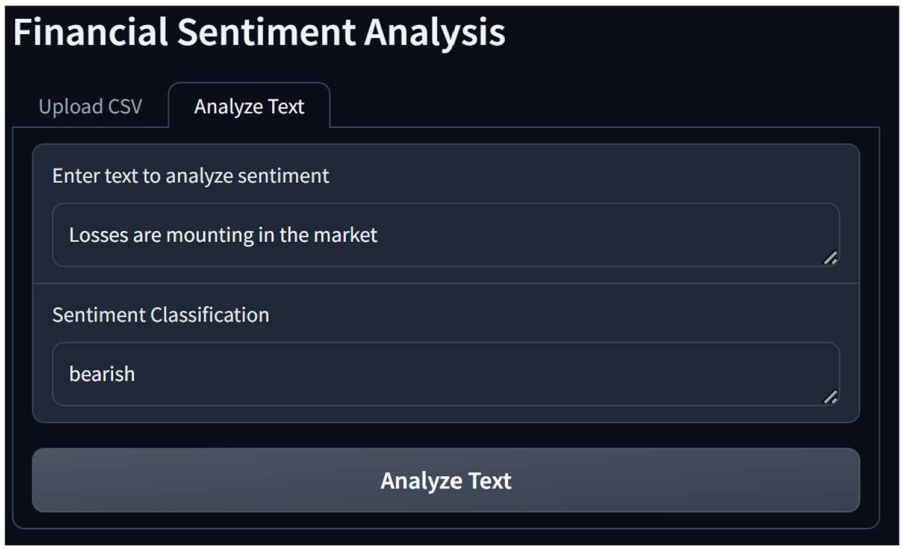
Enter text to analyze sentiment

Sentiment Classification

Analyze Text


Financial Sentiment Analysis
Upload CSV
Analyze Text
Enter text to analyze sentiment
The stock market is stable today
Sentiment Classification
neutral
Analyze Text


Financial Sentiment Analysis
Upload CSV
Analyze Text
Enter text to analyze sentiment
Strong earning reports are driving the stocks up
Sentiment Classification
bullish
Analyze Text


Financial Sentiment Analysis
Upload CSV
Analyze Text
Enter text to analyze sentiment
Losses are mounting in the market
Sentiment Classification
bearish
Analyze Text