*International Journal of Data Envelopment Analysis and *Operations Research*, 2023, Vol. 4, No. 1, 1-32*
Available online at http://pubs.sciepub.com/ijdeaor/4/1/1
Published by Science and Education Publishing
DOI:10.12691/ijdeaor-4-1-1

Predicting Stock Investments Based on Sentiment and Historical Price Data

I. O. Olawale[1], J. Iworiso[1,*], I. A. Amaunam[2]

[1]School of Computing and Digital Media, London Metropolitan University
[2]School of Computer Science and Electronic Engineering, University of Essex
*Corresponding author: J. Iworiso, j.iworiso@londonmet.ac.uk

Received September 02, 2023; Revised October 03, 2023; Accepted October 10, 2023

Abstract This paper examines the impact of integrating sentiment data from Twitter with historical stock prices to enhance stock market prediction accuracy. The study employs a comparative analysis using machine learning and deep learning algorithms—Support Vector Machine (SVM), Recurrent Neural Network (RNN), and Bidirectional Encoder Representations from Transformers (BERT) on a single stock. Among these algorithms, BERT achieved the highest predictive accuracy with a rate of 93.21%. The outperformance of BERT algorithm over the other techniques in this study is an indicative evidence that deep learning classification algorithms are superior to conventional sentiment analysis in stock market predictability, with immense contribution to empirical literature. All computations and graphics in this study are obtained using Python. In an extension to this core analysis, the study simulates two distinct investment strategies using aggregated data from ten different stocks: a passive long-term investment and an active, sentiment-based bot trading strategy. These strategies were evaluated using separate machine learning algorithms—Random Forest and XGBoost classifiers—to inform real-time investment decisions. The results indicate that both versions of the bot trading strategies, regardless of the machine learning or deep learning model employed, consistently outperform the passive, long-term investment strategy. The findings corroborate the utility of incorporating social media sentiment into traditional stock prediction frameworks, thereby providing valuable insights for investors and financial institutions. This study underscores the transformative potential of advanced machine learning algorithms and sentiment analysis in reducing investment risks and enhancing decision-making.

Keywords: *sentiment analysis, supervised learning, classifiers, ML, DL, accuracy*

Cite This Article: I. O. Olawale, J. Iworiso and I. A. Amaunam, "Predicting Stock Investments Based on Sentiment and Historical Price Data." *International Journal of Data Envelopment Analysis and *Operations Research**, vol. 4, no. 1 (2023): 1-32. doi: 10.12691/ijdeaor-4-1-1.

1. Introduction

Predicting stock investment is crucial for investors and financial institutions, which impact their financial performance and decision-making. Accurate prediction can lead to higher returns and better risk management, while inaccurate prediction can result in substantial losses. Precise stock analysis remains challenging despite its importance due to numerous influencing factors, including macroeconomic indicators, company-specific news, and investor sentiment. Researchers seek more accurate and reliable methods for analysing stock trends and movements. Recent studies have demonstrated the potential of incorporating market sentiment indicators and machine learning algorithms for improved forecasting performance [1,2] Consequently, the quest for better stock analysis methods remains a significant area of interest in academia and the financial industry.

As of late, the influence of social media has seen a remarkable increase, particularly Twitter and how it shapes financial market inclinations and tendencies. Twitter serves as a platform for users to express their viewpoints, news, as well as information about companies and their stocks, which enables investors to gauge public sentiment and make informed decisions [3]. Studies have demonstrated that using Twitter data is a viable approach to examining fluctuations within the realm of stock trading. This stems from the insight that what's posted on this social media platform could mirror investor feelings and sentiments, which are known for influencing changes in stock values [4]. Furthermore, studies have found that incorporating sentiment analysis on Twitter data into traditional stock analysis models can enhance their performance and lead to better investment outcomes [5]. With the expansive reach of social media on financial markets, investment firms, and investors must grasp how Twitter's emotional disposition affects market developments.

Sentiment analysis or opinion mining process is utilized in natural language processing to discern the emotions and sentiments conveyed within written material. This includes posts on social media platforms and news articles.

In the context of stock prediction, sentiment analysis has shown significant potential for improving the accuracy of forecasting models by incorporating investor sentiment as an additional input [6]. Studies have demonstrated that changes in public view, and opinion by social media platforms like Twitter, can be used to analyse stock trends and movements, as these changes often reflect investors' expectations and reactions to market events.[4] By incorporating sentiment analysis into traditional stock analysis models, it can help researchers and practitioners better understand market dynamics and improve their forecasting performance [7]. As the impact of communal perception on economic markets escalates, sentiment appraisal is positioned to become an even more invaluable apparatus for scrutinizing stock trends. Analysing stock trend is vital for investors and financial institutions, as precise forecasts can result in improved investment choices and increased returns. Analysing sentiment from Twitter and historical price data have become a compelling method for tackling this challenge. Twitter, a widely used social networking site, is an excellent database for collecting public perspective and investor emotion data. This information can enrich conventional stock analysis models in distinctive ways. By integrating sentiment analysis on Twitter data with historical price data, researchers can create more accurate and dependable forecasting models, allowing investors to make well-informed decisions and enhance their financial performance. Regarding the impact of social media on economic markets continues to expand, utilizing sentiment analysis and historical data is becoming progressively crucial for forecasting stock trends.

Current approaches to stock analysis face several challenges and limitations. Conventional methods, like technical and fundamental analysis, may only partially capture the complex dynamics of financial markets, including the influence of investor sentiment and external events [8]. While promising, sentiment analysis on social media data is subject to issues such as data quality, noise, and the need for reliable sentiment classification models [9]. Additionally, market movements' rapid and often unpredictable nature makes it difficult to achieve consistently accurate analyses [10]. Furthermore, the choice of appropriate features, algorithms, and model selection techniques can significantly impact the performance of forecasting models, necessitating extensive experimentation and validation. Addressing these challenges and limitations will be crucial for developing more effective stock analysis methodologies as the financial landscape evolves.

The aim of this paper is to investigate the potential of Sentiment analysis and historical price data to predict stock investment by simulating two investment strategies. The objective of the study is to compare different sentiment analysis algorithms, identify best among them, simulate two investment strategies and employ relevant metrics to evaluate their performance. We aimed at evaluating and assessing the performance of three distinct sentiment analysis algorithms, namely Support Vector Machine, Recurrent Neural Network, and BERT on a dataset of tweets related to a selected stock. We aimed to determine the optimal algorithm as per the evaluation outcome using evaluation measures like accuracy, precision, recall, F1 score, train, test and validation loss, train, test run time, ROC Curve and confusion matrix. Another objective is to apply the best-performing algorithm to predict the sentiment of the selected stocks and simulate two investment strategies - long-term investment and bot trading - to maximize returns for investors by allocating equal parts of investment across the selected stocks for the long-term approach. We aimed to analyse the effectiveness of the long-term investment and bot trading strategies based on the analyses, thereby evaluating the risk and return trade-offs and the practical implications of the strategies for investors and financial institutions.

This study covers several aspects related to stock investment analysis using sentiment analysis technique on Twitter data and historical price data. These aspects include the following:

Time Frame: The study focuses on one year, explicitly analysing tweets and stock data from Twitter starting from January 1, 2022, to December 31, 2022. This time frame allows for a comprehensive analysis of the selected stocks' performance and the impact of sentiment on their market trends.

Selected Stocks: The study examines 11 stocks from various sectors to ensure a diverse representation of industries, one stock for sentiment analysis, and ten stocks combined with their respective historical price data will be use in analysing the investment strategies. This selection enables the identification of common trends and patterns that may apply to a broader range of supplies and industries.

Algorithms: The study employs three different sentiment analysis algorithms, namely Support Vector Machine (SVM) a mix of traditional machine learning technique, Recurrent Neural Network (RNN) deep learning techniques and transformer models BERT, these algorithms allow for a comprehensive comparison of their performance and suitability for stock analysis using sentiment analysis.

Performance Metrics: The study utilizes various performance metrics to evaluate and assess the effectiveness of the sentiment analysis algorithms. These metrics include accuracy, precision, recall, F1 score, losses, confusion matrix and ROC Curve. Using multiple metrics ensures a robust assessment of the algorithms' effectiveness.

Investment Strategies: The study simulates two investment strategies, long-term investment, and bot trading, to assess their potential returns and risks. This exploration provides insights into the practical implications of sentiment analysis and historical price data for investment decision-making.

The applicability of this study's findings is restricted to the specified time span, stocks, and algorithms; thus, the results may not necessarily generalize to other periods or methods of stock investment analysis.

In addition, it has the potential to set the stage for more extensive investigation and research in this specific area in future years.

1.1. Previous Research on Stock Market Analyses

In recent years, sentiment analysis has gained significant attention in stock prediction. Researchers have

employed various techniques and data sources to analyse sentiment and historical price data in its potential to predict stock trends. Some of the previous are discussed below, along with their results and findings. Wang and his team [11] conducted a study analysing investment sentiment in the stock market by leveraging techniques such as BERT, LSTM, and SVM. They extracted sentiment scores from online content posted by stock investors and employed attention weighting to compute the investor sentiment index. The results indicated that the BERT model surpassed the LSTM and SVM methods, achieving an impressive accuracy rate of 97.35% in investor sentiment analysis. In their 2019 study, [12] Yildirim and his colleagues compared a variety of conventional classifiers and several iterations of RNN, GRU and LSTM models for sentiment analysis of stock-related tweets. Specifically, they evaluated the performance of traditional classifiers like multinomial naive Bayes, random forest (RF), and XGBoost, along with a Deep Learning algorithm (RNN), and a transformer model (BERT) using data collected from Twitter's stock tweets. In their 2021 study, Li and his colleagues [13] used the BERT model to perform sentiment analysis on online reviews from the Chinese Stock Exchange. They compared the effectiveness of BERT with several other algorithms, including TextCNN, TextRNN, Att-BLSTM, and Text-CRNN. Of all the methods they tested, BERT yielded the most superior results. In their 2021 research, Aysun and colleagues [14] utilized a variety of techniques, including conventional classifiers like logistic regression and random forest, deep learning algorithms such as LSTM and GRU, and transformer models like BERT, DistillBERT, RoBERTa, and XLNet. Their goal was to categorize tweets as either "bearish" or "bullish" in relation to five specific stocks: Apple Inc. (AAPL), Walt Disney Co. (DIS), Amazon (AMZN), Boeing Co. (BA), and the SPDR S&P 500 ETF Trust (SPY) during the period from December 2019 to June 2020. The findings from their study indicated that RoBERTa outperformed the conventional classifiers and deep learning algorithms in relation to average F1 scores.

Nguyen and his team [6] introduced a model designed to forecast stock price fluctuations by conducting sentiment analysis on financial news articles and utilizing technical indicators. Support Vector Machine was used to categorize the sentiment in news articles and predict trends in stock prices. The outcomes of their research showed that their sentiment analysis-based model accurately predict the direction of stock prices with an accuracy of 62.3%. However, the study was confined to financial news articles and the SVM may not be the best technique for sentiment analysis. Furthermore, the effectiveness of the model could be influenced by the choice of technical indicators and the specific stocks selected. In their 2014 study, Xiaodong and colleagues [15] utilized a deep learning method involving Recurrent Neural Networks equipped with LSTM to forecast stock price fluctuations. This was done on the sentiment analysis derived from financial news headlines. Their model managed to achieve a prediction accuracy of 70.62% for daily stock price trends. However, the research was limited to financial news headlines, which may not provide a comprehensive view of sentiment information. In addition, the selection of stocks and the period chosen could influence the effectiveness of the model. Zhou and his team [16] employed a BERT-based model for sentiment analysis of financial news articles, aiming to forecast stock market fluctuations. The authors discovered that the BERT-based model demonstrated a prediction accuracy of 73.1%, surpassing conventional machine learning methods like SVM and Naive Bayes. The scope of the study was restricted to financial news articles, and the performance of the model could potentially be affected by the chosen stocks and the period selected. Furthermore, due to the complexity of the model, it might require more computational resources than simpler methods.

1.2. Traditional Classifier (SVM)

Support Vector Machines is a form of supervised learning algorithm suitable for both classification and regression tasks. They are handy for high-dimensional data and are recognise for their robustness and capacity to manage noisy data. In sentiment analysis, SVM can classify text data (such as reviews, tweets, or news articles) into positive, negative, or neutral sentiment categories. It is particularly useful in the financial domain, as sentiment analysis of news articles, social media, or financial reports can help investors make better decisions. Numerous studies have applied SVM for sentiment analysis in stock trend prediction. These studies have used various techniques, including text pre-processing, attribute choice, and hyperparameter fine-tuning, to improve the effectiveness of the SVM models. In their 2011 research, Bollen and colleagues [3] explored the relationship between public mood states, as inferred from Twitter data, and the stock market. They utilized an SVM model for this purpose. Their findings indicated that their model was able to predict daily fluctuations (both increases and decreases) in the closing values of the Dow Jones Industrial Average (DJIA) with an accuracy of 87.6%. Jianfeng and his team [17] examined the application of SVM and various other machine learning methods for predicting stock market movements by conducting sentiment analysis of financial news articles. Their research findings showed that their SVM model managed to attain an accuracy rate of 62.4% when applied to the S&P 500 index. In their 2014 research, Xiaodong and his colleagues [15] investigated the effects of integrating sentiment analysis, derived from financial news articles, into the SVM model with the aim of predicting stock price fluctuations. They utilized a mix of textual and numerical features to boost the accuracy of the model's forecast. Their experimental results from the Chinese stock market indicated that the proposed method surpassed the conventional SVM model that didn't include sentiment analysis, attaining an accuracy rate of 70.7%. Kara and colleagues [18] applied sentiment analysis to financial news articles to forecast the stock market’s direction in the Istanbul Stock Exchange. They used an SVM model for the classification task. Their results showcased that their method successfully predicted the market direction with an accuracy of 79.8%.

In a comprehensive study by Kumar and colleagues 2018, [19]an array of classifiers including KNN, Random Forest, Support Vector Machine (SVM), and Naive Bayes

as per [20] were scrutinized to assess their proficiency in predicting stock market trends. The findings indicated that while the Naive Bayes model surpassed others in accuracy, the SVM, as per [21] and other models like Random Forest exhibited superior overall performance as suggested by their F-measures. Even so, the research did not investigate other models, like deep learning models, which have shown promising results in sentiment analysis. A 2021 investigation by Christina and Christos [22] employed sentiment analysis on Twitter data using two machine learning models: Support Vector Machines and Logistic Regression. The most impressive results were achieved when coupling VADER and SVM for analysing Twitter data, which yielded a maximum F-score of 76.3% and an Area Under Curve of 67%. On the other hand, when analysing StockTwits data using Text Blob in an imbalanced dataset scenario, SVM showcased the most fantastic accuracy, achieving a score of 65.8%. Nonetheless, the study might have seen improved results if it had utilized a different pre-trained algorithm like Roberta (Robustly Optimized BERT Pretraining Approach) for sentiment analysis. TextBlob and VADER occasionally misconstrue positive and negative remarks, resulting in potentially erroneous sentiment scores. Rakhi and Sher [23] undertook the task of evaluating Apple's stock by amalgamating 300,000 tweets and financial data spanning the years 2010 to 2017. To gauge public sentiment and predict whether an individual would opt to buy or sell a stock, they employed Support Vector Machines (SVM). The model incorporated three key factors: date, decision on stock price, and sentiment. The model demonstrated an accuracy of 75.22% on the training set and 76.68% on the test set.

1.3. Deep Learning with RNN

Recurrent Neural Networks (RNNs) are a unique class of artificial neural networks engineered to handle data sequences effectively. Unlike conventional feedforward neural networks, RNNs possess self-looping connections that retain hidden states, capturing information from previous time steps. This attribute makes RNNs highly adept at managing time series data, such as natural language or stock prices, by modelling the temporal dependencies in these sequences. Sentiment analysis, or opinion mining, involves identifying the sentiment or emotion conveyed in a text, like a review or social media post. RNNs are well-suited for sentiment analysis tasks due to their ability to process the sequential nature of textual data. By analysing input sequences element by element, RNNs can grasp the context of words and phrases, associating them with positive or negative sentiment. Previous studies have used RNNs for sentiment analysis in stock prediction by analysing financial news, social media posts, or other textual data sources to determine the prevailing sentiment toward specific stocks or the market. By associating emotion with stock price movements, these models can generate predictions of future trends. One notable study by [24] employed an RNN-based model called LSTM to forecast stock price movement using sentiment analysis on financial news. Their results showed that the LSTM model outperformed traditional models, with an accuracy of 62.03% in forecasting the stock price direction. Another study [25] by utilized an RNN architecture called Gated Recurrent Units (GRUs) to predict stock market movements using sentiment derived from financial news articles. Their model achieved an average directional accuracy of 57.6%, significantly better than the random walk hypothesis (50% accuracy).

1.4. Transformer Model using BERT

BERT is a deep learning and transformer model that was pretrained and introduced by Google in 2019. It has gained significant attention due to its ability to generate highly accurate representations of textual data. BERT is regarded as a Transformer model because it employs self-attention mechanisms to simultaneously process input sequences rather than sequentially, as in RNNs or LSTMs. BERT's bidirectional context helps it capture relationships between words more effectively, making it highly suitable for different natural language processing activities, such as sentiment analysis. In sentiment analysis, BERT can be fine-tuned to classify text based on the underlying sentiment, such as positive, negative, or neutral. Recent findings have focused on the influence of financial news on stock market close price predictions, emphasizing the significance of information in influencing stock prices. A study by [26] examined the Dow Jones Industrial Average (DJIA) and reported an impressive accuracy of 91.12% during the sentiment classification phase, highlighting the importance of the BERT model in this context. Additionally, their results demonstrated that integrating news sentiment into the predictive model resulted in decreased Mean Squared Error and Mean Absolute Error, along with a marginal enhancement in the r2 score. This study underscores the potential benefits of incorporating financial news sentiment, analysed using advanced natural language processing techniques like BERT, in stock market prediction models. Such findings imply that this methodology can enhance the precision and dependability of market forecasts, thereby offering crucial insights beneficial to investors and other interested parties.

Lee and colleagues [27] conducted a study on a BERT-based language model to classify Stocktwits messages as positive or negative sentiments. The authors fine-tuned the BERT model on a dataset with the labelled sentiment, thereby demonstrating the effectiveness of their methodology in determining investor sentiment. The findings from their experiments suggested that the BERT-based model could identify investor sentiment with an accuracy greater than 87.3%. In a study by Matheus and his team [28] BERT was used to execute sentiment analysis on news articles, providing vital result for stock market decision-making. This model underwent pre-training on a vast range of documents from various domains through self-learning. The team manually labelled stock news articles as positive, neutral, or negative to tailor this powerful model for sentiment analysis in the stock market. They compiled a publicly available dataset of 582 documents from different financial news sources and obtained an F-score of 72.5%. The team also executed experiments illustrating how the model's outputs can provide meaningful insights for forecasting subsequent fluctuations in the Dow Jones

Industrial (DJI) Index. The methods presented in the literature have yielded satisfactory results and valuable insights into sentiment analysis and its relationship with the stock market. However, these outcomes and findings vary, which could be attributed to the option of sentiment classifier, the curation and preparation of tweets, and the selection of tweets used for analysis.

1.5. Evaluation Metrics

In previous studies, accuracy has been the preferred metric, playing a pivotal role in aiding researchers and practitioners in appraising the efficacy of their models. Accuracy is determined by the percentage of correctly classified instances relative to the dataset's total instances. It is a widely accepted and intuitive metric for classification problems. However, it is worth noting that accuracy can provide a distorted view when working with imbalanced datasets. When there is a disproportionate number of instances in one class over others, a model that continually predicts the majority class can still achieve high accuracy, even if its performance for the minority classes is subpar. In our study, we have elected to use a suite of performance metrics, including accuracy, F1 score, recall, precision, test and training times, test and training and validation loss, confusion matrix, and ROC Curve. This selection of evaluation metrics allows for a comprehensive comparison of our model's performance.

2. Traditional Classifier (SVM)

Support Vector Machine algorithm for sentiment analysis is a widely used classification and regression tasks supervised learning algorithm under supervised learning. It is based on maximizing the margin between data points of different classes, thereby constructing an optimal separating hyperplane [29] Using other kernel functions, the algorithm can be applied to both linearly and non-linearly separable data.

In its basic form, SVMs are used for binary classification, segregating data points into one of two categories, either 0 or 1. However, for multiclass classification, the same methodology is employed. The multiclass problem is decomposed into several binary classification scenarios, a strategy often referred to as one-vs-one. the number of classifiers required for one-vs-one multiclass classification can be determined using a specific formula, where 'n' represents the total count of classes.

$$\frac{n^{*}(n-1)}{2} \tag{1}$$

The purpose of the hyperplane in a multiclass scenario is to distinguish between multiple class instances, corresponding to a set of points x_i that satisfy the equation $w.x+b=0$, where w is considered the hyperplane's normal vector and $|b|/\|w\|$ is the orthogonal distance from the hyperplane to the origin. $\|w\|$ signifies the Euclidean length or magnitude of w. The objective is to maximize the margin between the classes using the Support Vector Machine algorithm. For example, let's assume $C1$, $C2$ and $C3$ are the distances separating the class samples 1 ,0 and 2 respectively. The equations (1), (2) and (3) for the training data in this study meet the following conditions:

- For the hyperplane separating classes $C1$ and $C2$ ($C1$ being the positive class and $C2$ the negative), the condition is $x_i \cdot w+b>=1$ when corresponds to class $C1$, and $x_i \cdot w+b<=-1$ when y_i corresponds to class $C2$.
- For the hyperplane separating classes $C1$ and $C3$ ($C1$ being the positive class and $C3$ the negative), the condition is $x_i \cdot w+b>=1$ when y_i corresponds to class $C1$, and $x_i \cdot w+b<=-1$ when y_i corresponds to class $C3$.
- For the hyperplane separating classes $C2$ and $C3$ ($C2$ being the positive class and $C3$ the negative), the condition is $x_i \cdot w+b>=1$ when y_i corresponds to class $C2$, and $x_i \cdot w+b<=-1$ when y_i corresponds to class $C3$.

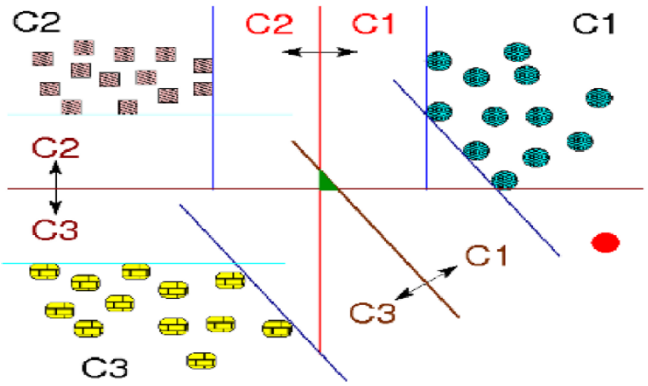

Figure 1. Illustration of the SVM principle and of the one-versus-one multiclass classification method (https://www.researchgate.net/figure/Illustration-of-the-SVM-principle-and-of-the-one-versus-one-multiclass-classification_fig2_220098164)

This process continues until each pair of classes has its own separating hyperplane. During prediction, each classifier votes for a class, and the class with the most votes is chosen as the prediction. This setup allows the SVM to handle the complexity of multiclass problems by breaking them down into multiple binary problems, which can be handled more easily.

2.1. Deep Learning with RNN

Recurrent neural networks (RNNs) have their foundation in the work of David Rumelhart. RNNs belong to the class of artificial neural networks where nodes are connected through directed links, creating a directed graph that follows a chronological sequence. This structure allows them to exhibit material dynamic behavior and process variable-length input sequences, thanks to their memory (internal state) derived from feedforward neural networks [30]. RNNs are particularly well-suited for predicting sequential data, as they perform the exact computation across each sequence unit. The results of

each step in the calculation are contingent on the outcomes of the preceding steps. The term "recurrent" refers to the RNN's ability to use its output as input for subsequent actions in the sequence. The architecture of RNNs allows them to handle variable-length inputs without relying on fixed-sized windows. This capability enables RNNs to model the attributional similarities between words and the relationships between pairs of words. For example, they can capture relationships like Chennai: Tamil: London: English. RNNs have a flexible architecture that can process input sequences of varying lengths and inherently model sequential input data. This architecture allows them to encode both word similarities and relationships between word pairs effectively.

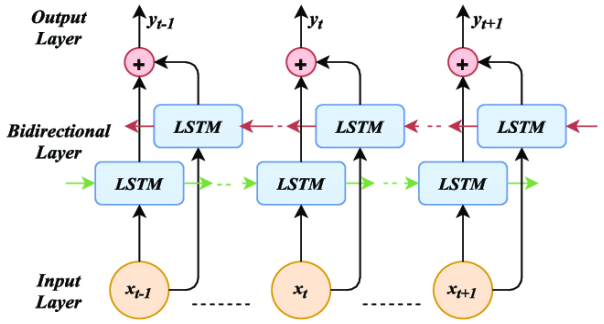

Figure 2. Bidirectional lstm model showing the input and output layers (https://www.researchgate.net/figure/Bidirectional-LSTM-model-showing-the-input-and-output-layers-The-red-arrows-represent_fig3_344554659)

A Recurrent Neural Network (RNN) is used for sentiment analysis. Sentiment analysis is a natural language processing task that aims to classify text based on the sentiment expressed, such as positive, negative, or neutral. RNNs are particularly well-suited for this task because they can process and model data sequences, such as words in a sentence, and capture long-term dependencies.

2.2. Transformer Model using BERT

The BERT model is a highly reliable, state-of-the-art method used in numerous Natural Language Processing (NLP) applications. The transformer, a ground-breaking network architecture which was first proposed by [31] and his team. Google AI Language, led by [32] and his team, introduced BERT in 2019. Since its introduction, BERT has seen widespread adoption and growth in both commercial and academic settings. Notably, BERT is a pre-trained model available in several languages, which can be easily fine-tuned to specific datasets. BERT architecture is a multi-layer bidirectional transformer encoder. Its input representation comprises three types of embeddings: position, segment, and token. In its pre-training phase, BERT deploys two unsupervised tasks, Masked Language Model (MLM) and Next Sentence Prediction (NSP), deviating from traditional sequence modelling approaches. BERT has been pre-trained on a corpus comprising more than 3 billion words.

The data input sequence is tokenized and trimmed to fit the maximum sequence length. This tokenized input is then converted into a tensor format, ready for fine-tuning. Once the model is fine-tuned, it can assess the sentiment of various sequences.

Figure 5 showcases the fundamental architecture of BERT. The network's input is the token representation vector Ei, calculated as the cumulative value of three separate vectors for every token: a distinct word embedding vector, a position embedding vector, and a sentence vector. The role of the position embedding vector is to supply the model with the token's placement data within its sentence, a concept absents in transformer models. The sentence vector comes into play only when the task demands a context wider than a single sentence, which is irrelevant for sentiment analysis (where we regard a document as a single sentence).

The layers based on attention generate a representation (like T (2)i) for each input token (such as Ei for the initial layer T (1) I) as an (adaptive) weighted sum of all token representations within the sentence. This characteristic is the primary advantage of transformer models, where every token representation is established on the representations of all other tokens. Therefore, the context is constrained only by the input sentence. The outcome of one attention-based layer serves as the input for the subsequent layer. The final attention layer's output constitutes the model's output.

Figure 3. A transformer model architecture (https://machinelearningmastery.com/the-transformer-model/)

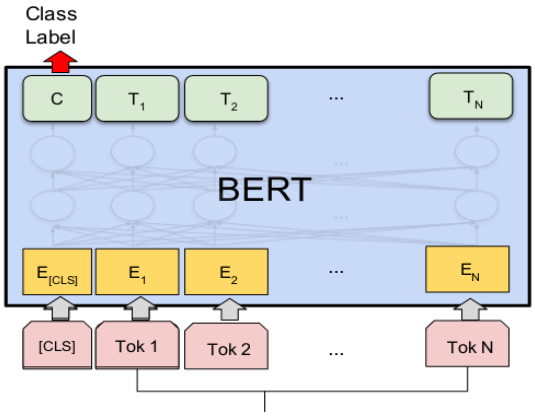

Figure 4. The bert model architecture (https://notebook.community/zhreshold/mxnet/contrib/clojure-package/examples/bert/fine-tune-bert)

2.3. Software Environment

Before diving into any data analysis, the crucial initial step is selecting the appropriate software that meets all the requirements for analysing the project's criteria and sub-criteria. The choice of software significantly influences the project's success in terms of operation and control. It is essential to select a software that can handle all aspects of data preparation, exploration, and modelling processes, including the necessary packages, algorithms, and configurations for accurate analysis. This ensures a clear understanding of the data and simplified results. For the implementation of this project, we will utilize a programming language called Python. Additionally, the project will be implemented using Google Colab Integrated Development Environment, which offers free GPU resources, enabling us to run powerful models more efficiently.

3. Data Understanding

In this phase, we collect data from two sources: stock-related tweets from Twitter (Twitter Inc., 2022) and historical stock prices from Yahoo Finance (Yahoo Finance, 2022). We scraped tweets from one year (2022-01-01 to 2022-12-31) for the following stocks: KO, ADBE, AZN, BP, EBAY, FDX, HPQ, ORCL, PEP, RIO, and TM. The data obtained from Twitter include the following search query parameters: Datetime, Tweet ID, Text, Username, Ticker, Favs, RTs, Followers, Following, and is_RT. The dataset for each stock, with each containing ten attributes, is as follows: KO has 43,313 records, ADBE has 37,161 records, AZN has 17,123 records, BP has 21,927 records, EBAY has 26,073 records, FDX has 32,050 records, HPQ has 15,651 records, ORCL has 24,242 records, PEP has 24,161 records, RIO has 19,103 records, and TM has 15,030 records.

We also collect historical stock price data for each stock from Yahoo Finance, covering the same one-year period (2022-01-01 to 2022-12-31) as the tweet data. The historical stock data consists of data points for each trading day within the specified date range, providing a comprehensive view of the stock's performance throughout the year. The exact number of data points rely on the number of trading days during the period, typically around 252 trading days in a year, excluding weekends and market holidays. For the sentiment analysis, we will only use the KO (Coca-Cola) tweet dataset. In contrast, the other stock tweet datasets will be combined with their respective historical stock price data from Yahoo Finance for stock analysis. This approach allows us to assess the performance of the chosen algorithms on a single stock dataset for sentiment analysis and apply the best-performing algorithm to the remaining stocks for further analyses.

3.1.1. Twitter Data

SunScrape is a Python library that allows users to efficiently scrape tweets from Twitter without needing an API key. It provides an accessible and convenient way to collect many tweets based on specified query parameters. The library can retrieve tweet attributes, such as date, tweet ID, text, and user-related information.

In this study, we use Scrape to collect tweets related to the selected stocks for sentiment analysis. The following query parameters are utilized to gather relevant tweet data (As shown in Table 1).

The Twitter module from the scrape library is imported as "twitter" to efficiently scrape tweets from Twitter. Pandas is a popular open-source library for Python which is used for data manipulation and analysis. It provides robust data structures like DataFrame and Series, simplifying the handling and processing large datasets. In this case, it will store and manipulate the collected tweet data. The library is imported under the alias pd for convenience. A tqdm library is a valuable tool for displaying progress bars during the execution of iterative tasks, such as loops. The progress bar provides a visual representation of the progress and an estimated completion time, which can be helpful when scraping a large volume of tweets. Next step is to define the search criteria, including the stock symbol, language, and date range. Initialize an empty list called "tweets list" for storing tweet data and a set called "processed_tweet_ids" to avoid duplicates. Use the TwitterSearchScraper class to scrape tweet data based on the defined search criteria and display the progress of the scraping process with a tqdm progress bar. Finally, check for and skip duplicates by seeing if the current tweet's ID is already in "processed tweet ids." If it is, skip it to avoid duplicates; otherwise, add the tweet ID to the set of processed IDs.

Table 1. Tweet Variable Description

Variable	Description
Datetime	The timestamp when the tweet was posted, allowing us to filter tweets within our desired time frame (2022-01-01 to 2022-12-31).
Tweet Id	A unique identifier for each tweet, valid for indexing and avoiding duplicates.
Text	The tweet’s content will be the primary input for sentiment analysis.
Username	The Twitter handle of the user who posted the tweet, providing context and potential insights into the tweet’s credibility.
Ticker	The stock symbol associated with the tweet, enabling us to categorize tweets for each stock.
Favs	The number of tweet likes has received, indicating its popularity and potential influence.
RTs	The number of times a tweet has been retweeted, reflecting the tweet's reach and impact.
Followers	The number of followers of the tweet user, suggesting the user's influence in the Twittersphere.
Following	The number of accounts the tweeting user follows, providing context for the user's engagement on the platform.
is_RT	A Boolean variable indicating whether the tweet is a retweet or an original post, allowing us to differentiate between unique and shared content.

	Datetime	Tweet Id	Text	Username	Ticker	Favs	RTs	Followers	Following	is_RT
0	2022-12-30 23:30:01+00:00	1608968680104423426	Nice print for $KO Size: 1968137 Price: 63.61 ...	TradeWithAlerts	KO	0	0	32087	328	False
1	2022-12-30 23:30:00+00:00	1608968678976163843	🔴🔴 You are invited to test drive PREMIUM feat...	Smith28301	KO	0	0	501	0	False
2	2022-12-30 23:20:04+00:00	1608966178818359296	$KO NEW ARTICLE : Coca-Cola: 61st Consecutive ...	StckPro	KO	0	0	5337	19	False
3	2022-12-30 23:19:07+00:00	1608965938690285570	$KO new alert at https://t.co/A7qrDarJHY #st...	mediasentiment	KO	0	0	6937	6237	False
4	2022-12-30 23:17:13+00:00	1608965461320040460	$KO - Coca-Cola: 61st Consecutive Annual Divid...	SeekingAlpha	KO	3	2	225461	139	True
...	...	...	...	...	...	...	...	...	...	...
43180	2022-01-01 03:00:01+00:00	1477112347391049732	$KO Data indicates that the overall analyst s...	invescent	KO	0	0	156	6	False
43181	2022-01-01 01:41:24+00:00	1477092560728375296	Coke vs. Pepsi Stock: Maybe You Can Beat the R...	CabotAnalysts	KO	0	0	4565	1646	False
43182	2022-01-01 01:06:48+00:00	1477083852766171139	@Alishaa41985572 $KO and $GM	omar50to1k	KO	1	0	930	150	False
43183	2022-01-01 00:51:18+00:00	1477079954600960003	RECAP FOR THIS WEEK: $Nio Calls went 800%+ $NI...	omar50to1k	KO	5	0	930	150	False
43184	2022-01-01 00:24:32+00:00	1477073219324723204	$KO @ 59.21 (+0.73%) : Hot Stocks: VORB plunge...	mehabecapital	KO	1	0	2754	35	False

43185 rows × 10 columns

Figure 5. Coca-Cola stock tweet DataFrame.

The tweet data to the tweets list, which includes the datetime, tweet ID, text, username, stock symbol, favourites, retweets, followers, following, and whether it is a retweet. Then, convert the tweets list into a panda DataFrame with the appropriate column names.

3.1.2. Historical Stock Data

The date range of the dataset covers the one year from 2022-01-01 to 2022-12-31, aligning with the time frame used for tweet data collection and the Number of Data Points of which each stock will have a data point for each trading day within the specified date range, providing a comprehensive view of the stock's performance throughout the year. The exact number of data points will rely on the number of trading days during the period, typically around 252 trading days in a year, excluding weekends and market holidays.

Table 2. Historical data Variable Description

Variable	Description
Daily Open	The opening price is the price at which the first stock trade occurs when the market opens for the trading day.
Daily High	The daily high price represents a stock's highest price during a single trading day.
Daily Low	The daily low price represents the lowest price at which a stock is traded during a single trading day.
Daily Close	The closing price refers to the price at which the last stock trade occurs before the market closes for the trading day.
Adjusted Close	The adjusted closing price accounts for corporate actions, such as stock splits, dividends, and new stock offerings, that can affect the stock's price.
Trading Volume	Trading volume represents the total number of shares traded during a single trading day.

Finance library is used to collect historical stock market data for the ten selected stocks, excluding the Coca-cola stock (KO) used for sentiment analysis. It provides an easy-to-use interface for downloading Yahoo Finance data. Next is to extract the date range by getting the minimum and maximum date from the tweet data. In this case, the start_date corresponds to the earliest date in the tweet data, and the end date is set to two days after the latest date in the tweet data. The extra two days are added to ensure we capture the stock price data for the entire range of tweets. Then, the yf. Download function is used to download the historical stock price data for the selected stock within the specified date range. In this step, File. Split function is used to extract the stock symbol from the filename. Finally, call the reset_index () method to reset the index of the resulting DataFrame, making it easier to work with later.

3.1.3. Brief Stock Description

Coca-Cola (KO)

Coca-Cola Company is a top multinational beverage corporation that produces and markets various non-alcoholic beverages, including its iconic Coca-Cola soft drink. Founded in 1892 and headquartered in Atlanta, Georgia, the company operates in over 200 countries and is one of the world's most valuable brands.

Adobe Inc. (ADBE)

Adobe Inc. is a multinational American software company that develops multimedia, creativity, and digital marketing software products. Founded in 1982 and its headquartered situated in San Jose, California, Adobe is best known for its flagship products, such as Photoshop, Illustrator, and Acrobat Reader, which have become industry standards.

AstraZeneca PLC (AZN)

AstraZeneca is a British-Swedish multinational pharmaceutical and biopharmaceutical company that research, develops, and manufactures prescription drugs for various medical conditions. Founded in 1999 and

headquartered in Cambridge, England, the company strongly focuses on oncology, cardiovascular, respiratory, and immunology therapeutic areas.

BP PLC (BP)

A British multinational oil and gas company that functions in all aspects of the energy industry, such as exploration, production, refining, and involved in the marketing of petroleum products. It was Founded in 1909 and it's headquartered is situated in London, England, the company is among the top world's largest energy companies and has a significant presence in renewable energy and low-carbon technologies.

eBay Inc. (EBAY)

eBay is an American multinational e-commerce corporation that operates a global online marketplace for buying and selling various goods and services. Founded in 1995 and headquartered in San Jose, California, eBay has transformed how people trade and connect, offering a platform for individual sellers, small businesses, and large enterprises.

FedEx Corporation (FDX)

FedEx is an American multinational delivery and courier services company that supply a diverse range of services like e-commerce and transportation worldwide. Founded in 1971 and headquartered in Memphis, Tennessee, FedEx has revolutionized the logistics industry with its innovative shipping and tracking solutions.

HP Inc. (HPQ)

HP Inc. is an American multinational IT company that develops and supply a wide range of imaging and printing products as well as personal computing devices. It was founded in 1939 and it headquarter is situated in Palo Alto, California, HP has a long history of innovation and is among the world's leading technology companies.

Oracle Corporation (ORCL)

Oracle is a multinational company in American and a computer technology corporation that focuses on marketing and developing database software, enterprise software products and cloud systems. Founded in 1977 and headquartered in Austin, Texas, Oracle has become a global leader in the IT industry, particularly in database management systems.

PepsiCo, Inc. (PEP)

An American multinational company that specializes in food and beverage and manufactures, markets, and distributes a diverse range of products, such as snacks, soft drinks, and other consumer goods. It was founded in 1965 and it headquarter situated in Purchase, New York, the company is among the world's largest in food and beverage, with a robust global presence and iconic brands.

Rio Tinto Group (RIO)

A British-Australian multinational company that specializes in mining and metals corporation and the exploration, mining, and processing of mineral resources. It was founded in 1873 and it headquarter located in London, England, the company is among the world's largest mining corporation, with operations spanning various continents and commodities.

Toyota Motor Corporation (TM)

Toyota is a Japanese multinational automotive manufacturer that produces a comprehensive range of vehicles, from compact cars to commercial trucks. It was founded in 1937 and it headquarter situated in Toyota City, Japan, Toyota is one of the world's largest automakers.

4. Exploratory Data Analysis

Exploratory Data Analysis is an important step in understanding the dataset and finding patterns, trends, or anomalies within the data. In the context of this tweet dataset, python is used to explore the dataset providing a simplified understanding of the data. During this phase, our main interest is to get enough information regarding the relationship between the explanatory variables(text) and other columns aiming to understand or detect any patterns.

4.1.1. Number of Coca-Cola tweets Over Time

It is seen that November 2022 has the number of Coca-Cola stock-related tweets, followed by May and August. These findings may suggest seasonal patterns or events that lead to increased or decreased social media activity related to Coca-Cola. For instance, the high spike in November could be associated with holiday marketing campaigns or product launches. Similarly, the spikes in May and August might be related to summer promotions or other marketing activities. The low tweet count toward the end of March could be due to a lack of significant events or campaigns.

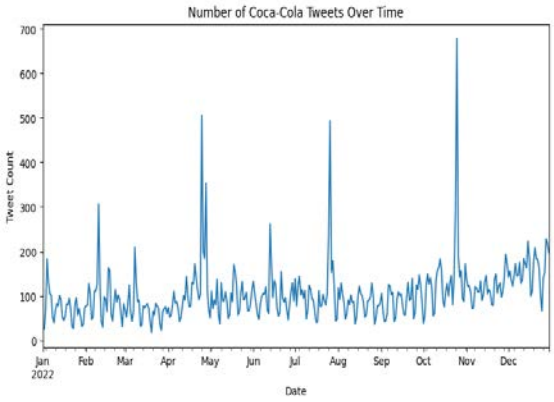

Figure 6. Number of coca-cola tweets over time.

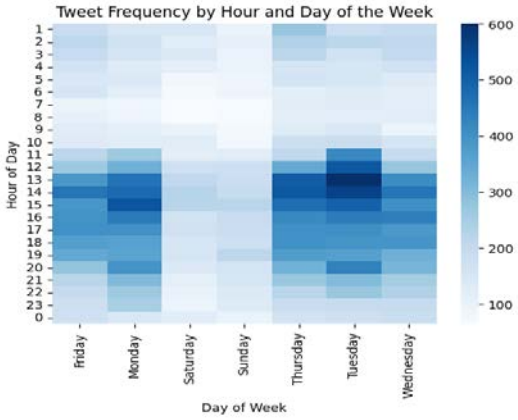

Figure 7. Tweet frequency by hour and day of the week

4.1.2. Tweet Frequency by Hour and Day of the Week

The heatmap shows that the highest tweet frequencies occur on Tuesdays between 11 am and 4 pm and Mondays from 1 pm to 4 pm. This suggests a higher level of social media activity related to Coca-Cola during these periods, which user engagement, marketing campaigns, or other factors might drive. Conversely, the heatmap shows that the weekends, specifically Saturdays and Sundays, have the lowest tweet frequencies throughout 2022. This might indicate that social media conversations related to Coca-Cola are less active on weekends.

4.1.3. Correlation Heatmap of Numerical Columns

Retweets (RTs) and Favourites (Favs) - 87%: This solid positive correlation suggests that when a tweet gets many retweets, it will likely have many favorites. This makes sense as both actions are ways for users to engage with and show appreciation for a tweet. A tweet that resonates with users or is of high quality would likely receive both retweets and favorites. Is_RT and Favourites - 32%: A moderate 32% correlation between retweets and favourites on Twitter. This suggests that retweets, often being popular content, tend to accumulate more favourites due to their wider reach. This insight can be beneficial for users or businesses seeking to enhance their visibility and influence on the platform by strategically retweeting content that aligns with their brand or message. Negative correlation with Tweet ID: The negative correlation between Tweet ID and other variables is likely due to the nature of how Tweet IDs is generated. Tweet IDs are unique identifiers chronologically assigned to each tweet, with newer tweets having higher IDs.

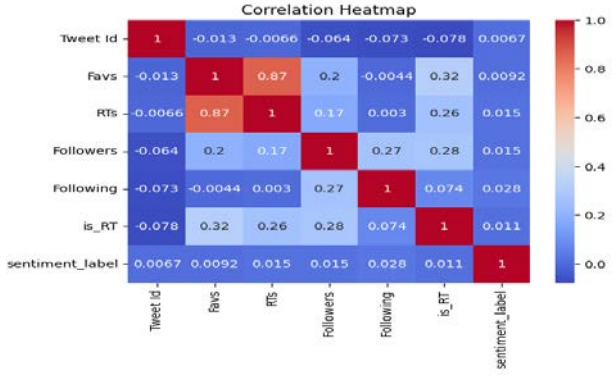

Figure 8. Correlation heatmap of numerical columns

4.1.4. Top Users by Tweet Count

It can be observed from the plot that Username AbhipraGroup has the highest in the top 20 user tweet count, with over 700 tweet count, and the lowest in the top 20 users by tweet count is newsfilterio, with close to 200 tweet count. Some of the top users in the dataset may be influential in the stock market or have a large following that could impact stock prices. By examining their tweets and activity, you can identify market trends or sentiments influencing stock prices. For example, if a popular user frequently tweets bullish opinions about a particular stock, this could increase demand and stock price.

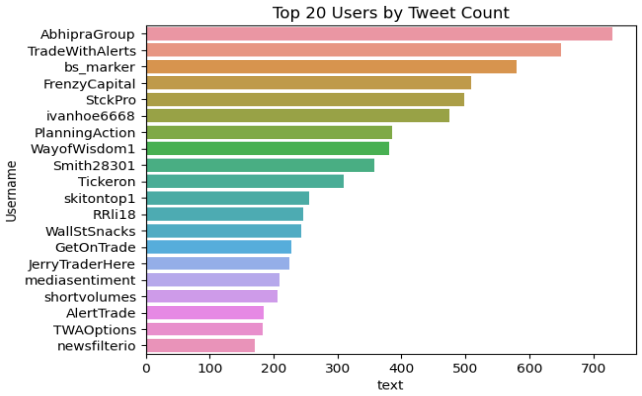

Figure 9. Top Users by Tweet Count

4.1.5. Top Words in Tweet

From the chart, we can observe that the top-used word or hashtag in the tweets is the stock symbol "$KO," which appears close to 35,000 times. This reflects that a large proportion of the tweets in the dataset are directly related to the stock, as the stock symbol is frequently mentioned. The second most common word is 'the,' which appears 15,000 times. This result is expected, as 'the' is a common English word that is often used in various contexts. It is crucial to note that while 'the' is a common word, its frequency might not provide much insight into the stock-specific discussions. The least frequent word among the top 10 words is 'is' and ‘on’ which appears 5,000 times. Like 'the,' 'is' and ‘on’ are also a common English word and may not be particularly informative for understanding stock-related discussions. However, its presence in the top 10 words highlights the general language structure of the tweets.

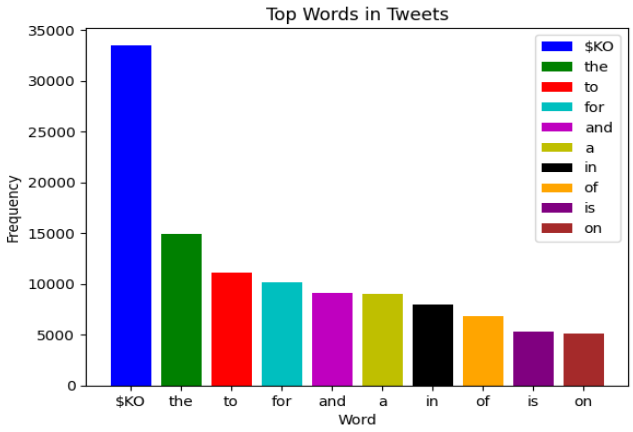

Figure 10. Top words in tweet

4.1.6. Scatter Plot of Retweets and Favorites

The scatter plot between retweets and favorites shows most tweets have low retweets (0-200) and favorites (up to 1250), suggesting more likes than shares. There are outliers with high favorites between 12500, 1500, 1750, and in between 1500 and 1750 indicating viral content. Tweets with 0-100 retweets often have around 500 favorites, further indicating a preference for liking over sharing. This analysis helps understand user engagement trends in the dataset.

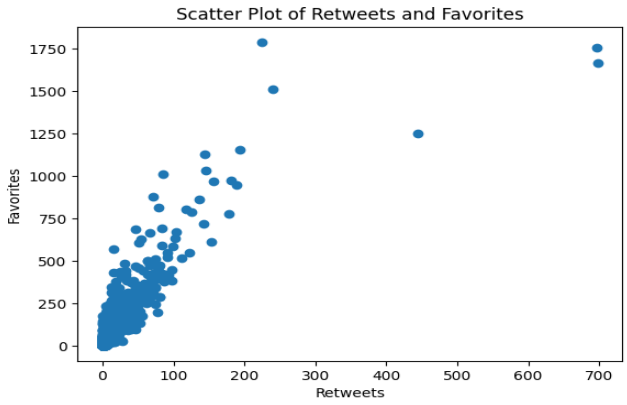

Figure 11. Scatter plot of retweets and favorites

4.1.7. Distribution of Token Counts Within Tweets

From the plot, we can observe that the distribution of token counts is right skewed, which means that there are more tweets with a lower number of tokens than those with a higher number of tokens. Most tweets have around 0 to 20 tokens, as indicated by the peak above 400 in the y-axis. This suggests that most of the tweets in the dataset are relatively short and concise. Another interesting observation is the sharp drop in the number of tokens from around 60 to 100. This could be due to the inherent nature of the platform (Twitter) which, until November 2017, had a 140-character limit per tweet. This character limit might have contributed to users condensing their thoughts into shorter messages, resulting in fewer tweets with a higher number of tokens.

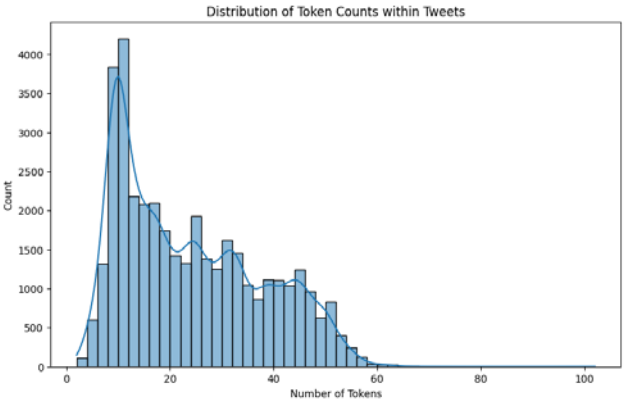

Figure 12. Distribution of token counts within tweets

4.1.8. Word Cloud Plot

From the word cloud plot, it is observed that words like "Coca Cola," "analyst price," "Top analyst," "KO," "Top," "Price target," "https," "next move finance success," "real time," "target today," "right now," "next week," and "success wealth" are among the most visible words. This suggests that these words are frequently mentioned or discussed in the tweets, indicating that they could be significant in understanding the context of the dataset. Based on the visible words, we can infer that the dataset might be related to financial topics, particularly regarding stock prices and price targets. The dataset may include discussions on the performance of specific companies, such as Coca Cola (KO), and the predictions made by top analysts. The references to "real time," "right now," "next week," and "today" imply that the dataset contains time-sensitive information, which is typical for financial markets and stock trading.

Figure 13. Word cloud plot of coca-cola tweet

4.1.9. Distribution of Tweet Lengths

The histogram of tweet lengths indicates that most tweets in our dataset are 400 characters or less, reflecting Twitter's emphasis on brevity. Whereas the maximum length is 800, suggesting more detailed content or usage of features like threading to exceed typical limits. This distribution is important for our analysis, shorter tweets might express sentiment more directly, while longer ones could provide richer information, impacting our sentiment analysis model's approach.

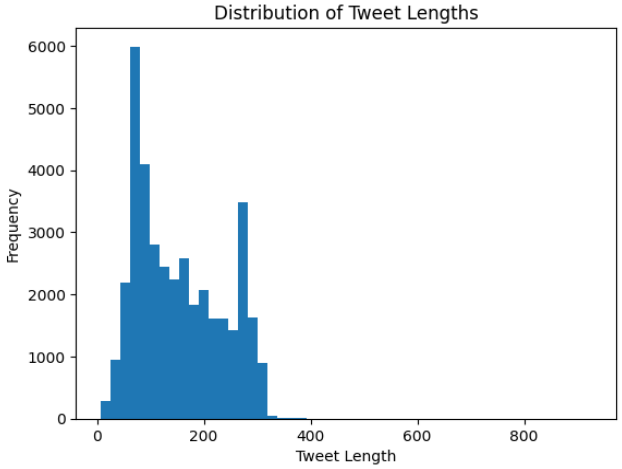

Figure 14. Distribution of Tweet Lengths

5. Data Preprocessing

The vast amount of textual data available offers numerous opportunities to train Natural Language Processing (NLP) models. However, the inherent unstructured form of text data necessitates preprocessing. This involves lowercasing, fixing units, removing punctuation, and eliminating stop words. These preprocessing steps can be efficiently executed using Python programming language, with the help of libraries

such as pandas, NumPy, Regular Expression (re), text blob, or nltk. Subsequently, tokenization, stemming, and lemmatization transform the raw text data into smaller, less redundant units. Tokenization, as per [33], involves breaking down the text into individual words. Stemming techniques which are used to retrieve the root of a word, while lemmatization, like stemming, is used to reduce words to their base or dictionary form.

5.1.1. Text Cleaning

Text preprocessing is a crucial step in NLP that includes cleaning and converting messy text data into a structured format that is prepared for further analysis. This process encompasses several steps like tokenization, stemming, lemmatization, the elimination of stop words, and part-of-speech tagging. In this project, we will use the following text processes like Converting to Lowercase, Removal of Hyperlinks and URLs, Removal of User Mentions and Hashtags, Removal of Emojis, Handling Contractions, Handling Symbols, Remove Special Characters and Punctuations, Fixing Units, Tokenize the Text, Removal of Stop Words, Lemmatization and Part-of-Speech Tagging.

```
# Define the preprocessing steps
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove hyperlinks and URLs
    text = re.sub(r'https?:\/\/.*[\r\n]*', ' ', text)
    text = re.sub(r'www.*[\r\n]*', ' ', text)
    text = re.sub(r'https', '', text)
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    # Remove user mentions and hashtags
    text = re.sub(r'\@\w+','',text)
    text = re.sub(r'\#\w+', '', text)
    text = re.sub(r'@[a-z0-9]+', '', text)
    text = re.sub(r'#[a-z0-9]+', '', text)
    text = re.sub(r'@', '', text)
    text = re.sub(r'#', '', text)
    # Remove emojis
    text = text.encode('ascii', 'ignore').decode('ascii')
    # Handle contractions
    text = contractions.fix(text)
    # Handling symbols
    text = text.replace('%', ' percent')
    text = re.sub(r'&amp', ' and ', text)
    text = re.sub(r'&gt', ' greater than ', text)
    text = re.sub(r'c&h', 'cup and handle', text)
    text = re.sub(r'h&s', 'head and shoulders', text)
    text = re.sub(r'p&f', 'point and figure', text)
    text = re.sub(r's&p', 'SP500', text)
    text = re.sub(r'q&a', 'question and answer', text)
    text = re.sub(r'b/c', ' because ', text)
    text = re.sub(r'b/o', ' break out ', text)
    text = re.sub(r'p/e', ' pe ratio ', text)
    text = re.sub(r'\$[a-z]+', lambda m: 'stock ' + m.group()[1:], text, flags=re.IGNORECASE)
```

Figure 15. Text Preprocessing steps

While index 17 of the data frame is utilized as an example of the text preprocessing, it's crucial to note that not all the steps used in this project were necessary for this specific piece of text.

Convert to lowercase: This step converts all characters in the text to lowercase. It helps ensure consistency and reduce complexity during further text analysis. Lowercasing reduces the chances of having multiple copies of the same word in different cases, which might otherwise be treated as separate entities.

Table 3. Convert to lowercase.

Original Text	Convert to Lowercase
$AAPL $AXP $BAC $CVX $KO - 73% of Warren Buffett's Portfolio Is Invested in These 5 Stocks https://t.co/PMdxlSd51r	$aapl $axp $bac $cvx $ko - 73% of warren buffett's portfolio is invested in these 5 stocks https://t.co/pmdxlsd51r

Remove URLs, and User mentions: URLs and user mentions are removed from the text as they do not contribute to the tweet's sentiment. The re. sub () function

with regular expressions identifies and removes URLs and user mentions.

Table 4. Remove URLs, and User mentions.

Convert to Lowercase	Remove URLs, and User mentions
$aapl $axp $bac $cvx $ko - 73% of warren buffett's portfolio is invested in these 5 stocks https://t.co/pmdxlsd51r	$aapl $axp $bac $cvx $ko - 73% of warren buffett's portfolio is invested in these 5 stocks

Handling Symbols: refers to the process of dealing with various symbols in the text data that may not participate to the overall meaning or context. These symbols could include punctuation marks, mathematical symbols, currency symbols, special characters, etc.

Table 5. Handling Symbols in Text

Remove URLs, and User mentions	Handling Symbols
$aapl $axp $bac $cvx $ko - 73% of warren buffett's portfolio is invested in these 5 stocks	stock aapl stock axp stock bac stock cvx stock ko - 73 percent of warren buffett's portfolio is invested in these 5 stocks

Remove special characters and punctuations: Special characters and punctuations are eliminated from the text to focus on meaningful words. This step simplifies the text data and helps reduce noise during sentiment analysis.

Table 6. Remove Special Characters and Punctuations.

Handling Symbols	Remove special characters and punctuations
stock aapl stock axp stock bac stock cvx stock ko - 73 percent of warren buffett's portfolio is invested in these 5 stocks	stock aapl stock axp stock bac stock cvx stock ko 73 percent of warren buffett s portfolio is invested in these 5 stocks

Tokenize the text: Tokenization breaks down the text into individual words or tokens. This step is important for further text analysis, as it helps us to work with words as separate entities and apply various techniques to understand their meanings and relationships.

Table 7. Tokenize the Text

Remove special characters and punctuations	Tokenize the Text
stock aapl stock axp stock bac stock cvx stock ko 73 percent of warren buffett s portfolio is invested in these 5 stocks	['stock', 'aapl', 'stock', 'axp', 'stock', 'bac', 'stock', 'cvx', 'stock', 'ko', '73', 'percent', 'of', 'warren', 'buffett', 's', 'portfolio', 'is', 'invested', 'in', 'these', '5', 'stocks']

Remove stop words: They are common words (such as 'the,' 'is,' 'and') that do not participate much to the meaning or sentiment of the text. Eliminating stop words helps reduce the magnitude of the text data and focus on the more informative words.

Table 8. Removal of Stop words

Tokenize the Text	Removal of stop words
['stock', 'aapl', 'stock', 'axp', 'stock', 'bac', 'stock', 'cvx', 'stock', 'ko', '73', 'percent', 'of', 'warren', 'buffett', 's', 'portfolio', 'is', 'invested', 'in', 'these', '5', 'stocks']	['stock', 'aapl', 'stock', 'axp', 'stock', 'bac', 'stock', 'cvx', 'stock', 'ko', '73', 'percent', 'warren', 'buffett', 'portfolio', 'invested', '5', 'stocks']

Lemmatization: Lemmatization involves reducing words to their base form or dictionary form, called lemmas. This step helps simplify the text analysis by grouping different inflections of a comment under a single lemma. For example, termers 'running,' 'ran,' and 'runs' would all be reduced to the lemma 'run.'

Table 9. Lemmatization

Removal of stop words	Lemmatization
['stock', 'aapl', 'stock', 'axp', 'stock', 'bac', 'stock', 'cvx', 'stock', 'ko', '73', 'percent', 'warren', 'buffett', 'portfolio', 'invested', '5', 'stocks']	['stock', 'aapl', 'stock', 'axp', 'stock', 'bac', 'stock', 'cvx', 'stock', 'ko', '73', 'percent', 'warren', 'buffett', 'portfolio', 'invested', '5', 'stock']

Perform part-of-speech tagging: This is the procedure of determining the grammatical category of each token, like noun, verb, adjective, adverb, etc. This can be helpful for further text analysis and feature extraction.

Table 10. Part-of-speech tagging

Lemmatization	Part-of-speech tagging
['stock', 'aapl', 'stock', 'axp', 'stock', 'bac', 'stock', 'cvx', 'stock', 'ko', '73', 'percent', 'warren', 'buffett', 'portfolio', 'invested', '5', 'stock']	[('stock', 'NN'), ('aapl', 'NN'), ('stock', 'NN'), ('axp', 'NN'), ('stock', 'NN'), ('bac', 'NN'), ('stock', 'NN'), ('cvx', 'NN'), ('stock', 'NN'), ('ko', 'VBD'), ('73', 'CD'), ('percent', 'NN'), ('warren', 'NN'), ('buffett', 'NN'), ('portfolio', 'NN'), ('invested', 'VBD'), ('5', 'CD'), ('stocks', 'NNS')]

Combine processed features: The final step combines the lemmatized tokens into a single string. This processed text will be used for further sentiment analysis and machine-learning tasks.

Table 11. Combine processed features.

Part-of-speech tagging	Combine processed features
[('stock', 'NN'), ('aapl', 'NN'), ('stock', 'NN'), ('axp', 'NN'), ('stock', 'NN'), ('bac', 'NN'), ('stock', 'NN'), ('cvx', 'NN'), ('stock', 'NN'), ('ko', 'VBD'), ('73', 'CD'), ('percent', 'NN'), ('warren', 'NN'), ('buffett', 'NN'), ('portfolio', 'NN'), ('invested', 'VBD'), ('5', 'CD'), ('stocks', 'NNS')]	stock aapl stock axp stock bac stock cvx stock ko 73 percent warren buffett portfolio invested 5 stock

5.1.2. Cleaning Crawl Data from Twitter

The assessment of the dataset reveals a substantial amount of noise, necessitating preprocessing. This includes transforming various columns into suitable data types to align with the nature of the information they hold. In addition, it is essential to manage any absent or erroneous data entries, as these can distort the analysis or create obstacles to implementing the models. This might involve strategies such as deletion, imputation, or data augmentation. By cleaning the data, conforming it to appropriate formats, and guaranteeing its completeness, we can considerably improve the trustworthiness and precision of our ensuing data examination and predictive modelling tasks.

Missing Values in Data

	features	missing_counts	missing_percent
0	Datetime	52	0.1
1	Tweet Id	75	0.2
2	Text	75	0.2
3	Username	101	0.2
4	Ticker	101	0.2
5	Favs	101	0.2
6	RTs	101	0.2
7	Followers	101	0.2
8	Following	127	0.3
9	is_RT	127	0.3

Figure 16. Missing values in data

Here is a summary of the preprocessing done on each column:

Datetime column:

Converts the column to datetime format and remove rows with invalid dates.

```
# First, convert "Datetime" column to datetime format
data['Datetime'] = pd.to_datetime(data['Datetime'], errors='coerce')

# Use isnull() to create a boolean mask for rows with invalid dates
mask = data['Datetime'].isnull()

# Use dropna() to remove rows with invalid dates
data = data.dropna(subset=['Datetime'])

print(f"Data shape after cleaning Datetime: {data.shape}")
```

Figure 17. Cleaning of the datetime column

Tweet Id column:

Converts the column to integers and remove rows with invalid Tweet Ids

```
# Convert "Tweet Id" column to integers and remove rows with invalid data
data.loc[:, 'Tweet Id'] = pd.to_numeric(data['Tweet Id'], errors='coerce', downcast='integer')

# Use isnull() to create a boolean mask for rows with invalid Tweet Id
mask = data['Tweet Id'].isnull()

# Use dropna() to remove rows with invalid Tweet Id
data = data.dropna(subset=['Tweet Id'])

print(f"Data shape after cleaning Tweet Id: {data.shape}")
```

Figure 18. Cleaning of the tweet id column

Text column:

Replaces empty strings with NaN values and removes rows with invalid missing Text.

```
# Remove rows with empty strings or NaN values in the "Text" column
data = data.assign(Text=data['Text'].replace('', np.nan))

# Use isnull() to create a boolean mask for rows with invalid Tweet Id
mask = data['Text'].isnull()

# Use dropna() to remove rows with invalid Text
data = data.dropna(subset=['Text'])

print(f"Data shape after cleaning Text: {data.shape}")
```

Figure 19. Cleaning of the text column

Username column:

Replaces empty strings with NaN values and removes rows with invalid or missing Usernames.

```
# Remove rows with empty strings or NaN values in the "Username" column
data = data.assign(Username=data['Username'].replace('', np.nan))

# Use isnull() to create a boolean mask for rows with invalid username
mask = data['Username'].isnull()

# Use dropna() to remove rows with invalid Username
data = data.dropna(subset=['Username'])

print(f"Data shape after cleaning Username: {data.shape}")
```

Figure 20. Cleaning of the Username column

Ticker column:

Replaces empty strings with NaN values and remove rows with invalid or missing Tickers.

```
# Remove rows with empty strings or NaN values in the "Ticker" column
data = data.assign(Ticker=data['Ticker'].replace('', np.nan))

# Use isnull() to create a boolean mask for rows with invalid Ticker
mask = data['Ticker'].isnull()

# Use dropna() to remove rows with invalid Ticker
data = data.dropna(subset=['Ticker'])

print(f"Data shape after cleaning Ticker: {data.shape}")
```

Figure 21. Cleaning of the Ticker column

Favs column:

Converts the column to integers and remove rows with invalid Favs value.

```
# Convert "Favs" column to integers and remove rows with invalid data
data.loc[:, 'Favs'] = pd.to_numeric(data['Favs'], errors='coerce', downcast='integer')

# Use isnull() to create a boolean mask for rows with invalid Favs
mask = data['Favs'].isnull()

# Use dropna() to remove rows with invalid Favs
data = data.dropna(subset=['Favs'])

print(f"Data shape after cleaning Favs: {data.shape}")
```

Figure 22. Cleaning of the favourite's column

RTs column:

Converts the column to integers and remove rows with invalid Retweets value.

```
# Convert "RTs" column to integers and remove rows with invalid data
data.loc[:, 'RTs'] = pd.to_numeric(data['RTs'], errors='coerce', downcast='integer')

# Use isnull() to create a boolean mask for rows with invalid RTs
mask = data['RTs'].isnull()

# Use dropna() to remove rows with invalid RTs
data = data.dropna(subset=['RTs'])

print(f"Data shape after cleaning RTs: {data.shape}")
```

Figure 23. Cleaning of the retweet's column.

Followers' column:

Converts the column to integers and remove rows with invalid Followers value.

```
#Convert "Followers" column to integers and remove rows with invalid data
data.loc[:, 'Followers'] = pd.to_numeric(data['Followers'], errors='coerce', downcast='integer')

# Use isnull() to create a boolean mask for rows with invalid Followers
mask = data['Followers'].isnull()

# Use dropna() to remove rows with invalid Followers
data = data.dropna(subset=['Followers'])

print(f"Data shape after cleaning Followers: {data.shape}")
```

Figure 24. Cleaning of the Followers column

Following column:

Converts the column to integers and remove rows with invalid Following value.

```
# Convert "Following" column to integers and remove rows with invalid data
data.loc[:, 'Following'] = pd.to_numeric(data['Following'], errors='coerce', downcast='integer')

# Use isnull() to create a boolean mask for rows with invalid Following
mask = data['Following'].isnull()

# Use dropna() to remove rows with invalid Following
data = data.dropna(subset=['Following'])

print(f"Data shape after cleaning Following: {data.shape}")
```

Figure 25. Cleaning of the following column

Is_RT column:

Converts string values to Boolean values and remove rows with invalid data.

```
# Convert string values to boolean
bool_mapping = {'True': True, 'False': False, None: np.nan}
#bool_mapping = {True: True, False: False, np.nan: np.nan}
data['is_RT'] = data['is_RT'].map(bool_mapping)

# Remove rows with invalid data in the "is_RT" column
data = data[data['is_RT'].isin([True, False])]

print(f"Data shape after cleaning is_RT: {data.shape}")
```

Figure 26. Cleaning of the Is_RT column.

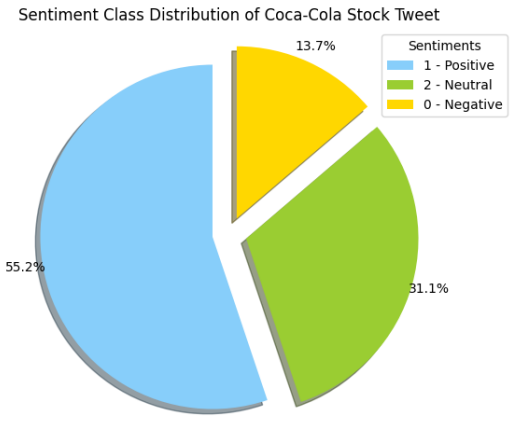

Figure 27. Sentiment class Distribution of coca-cola stock tweet.

5.1.3. Sentiment Class Distribution Using TextBlob

TextBlob is a Python-based library for handling and manipulating text data. It offers an easy API for performing typical natural language processing tasks. It was used to analyse the initial sentiment of the text data and classify it into different sentiment categories, Positive as 1, Negative as 0 and Neutral as 2.

5.1.4. Feature Extraction

After finishing the preprocessing steps, feature extraction is the next stage. This process converts text data, either at the word or character level, into a format that machines can understand and process, a step known as vectorization. This results in corresponding vectors for each word or character, ready for subsequent analytical processing.

5.1.4.1. Frequency-Inverse Document Frequency (TF-IDF) Vectorizer

In the Support Vector Machine (SVM) model, the feature extraction is accomplished using the Term Frequency-Inverse Document Frequency Vectorizer, a component of the sci-kit-learn library [34]. This tool transforms a collection of raw documents into a TF-IDF feature matrix, a technique frequently employed in NLP and text mining. The TF-IDF technique integrates Term Frequency and Inverse Document Frequency. Term Frequency assess how often a term shows up in a document, indicating the word's relevance within the text [35]. IDF assesses the relevance of a word across a collection of documents or a corpus [36]. The product of TF and IDF for each word gives its TF-IDF score, a valuable feature in machine learning algorithms [37]. Notably, the TF-IDF technique minimizes the weight of common words (like "the," "and" etc.) and accentuates words that are unique to a document or a group of documents. The TfidfVectorizer changes the raw text data into a matrix of TF-IDF features, setting it up for additional analysis and processing. The vectorizer is fitted on the training set and subsequently used to transform both the validation and test datasets.

5.1.4.2. Global Vectors for Word Representation (GloVe embeddings)

Feature extraction in the Recurrent Neural Network (RNN) model leverages Word Embeddings, particularly GloVe (Global Vectors for Word Representation) embeddings. Word embeddings are compact vector representations of words that encapsulate semantic and syntactic word properties. They are used as input features for machine learning models. GloVe, developed by [38], is a widely used method for learning word embeddings from extensive text data. It incorporates benefits from global matrix factorization methods (e.g., Latent Semantic Analysis) and local context window methods (like word2vec's skip-gram model). GloVe formulates word embeddings by factorizing the word co-occurrence probability matrix, capturing local and global information. GloVe embeddings are produced by training a neural network on a vast text corpus. This results in a high-dimensional vector for each word that captures its meaning and relations with other words. Using pre-trained

GloVe embeddings, we can use the knowledge gained from extensive text data for our sentiment analysis task. In the RNN model, the tokens in the input text are first converted to indices using the word2index dictionary created from the most common words in the training set. The index is then used to retrieve the corresponding GloVe embeddings from the embedding matrix. These embeddings, serving as the features for the input text, are then fed into the RNN model for sentiment classification.

5.2.1. Data Partition

Models are designed to generalize the knowledge derived from the dataset. However, poor generalization can arise due to overfitting. Data partitioning, or data splitting, is utilized to counter this challenge. Overfitting occurs when a model demonstrates outstanding performance on the training set, which can subsequently result in suboptimal performance when the model is applied to new, unseen datasets. This can result in inaccurate predictions for future observations. The training set aims to establish a model that can accurately predict unseen examples). Before data modeling, the dataset is divided into training, validation, and testing. The training set can be used to fit the data. The validation set evaluates the model's performance and prevents overfitting. Lastly, the testing set is used to assess the model's effectiveness on test data. Various techniques are employed to partition the data, and the choice depends on the type of learning and analysis being conducted. In the Train-Validation-Test split approach, the data is divided into these three distinct sets. This method enables the model to gain insights from the training data and adjust its parameters using the validation data. Ultimately, the test data offers an impartial assessment of the final model's performance, effectively helping to prevent overfitting. During this project, the split ratio for the training, validation, and test sets, relative to the original data size, is approximately 72% for training, 8% for validation, and 20% for testing.

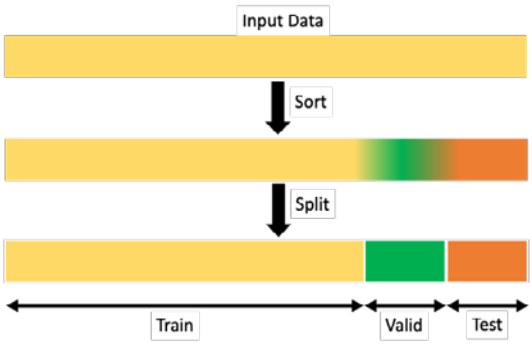

Figure 28. Train test validation split

6. Model Evaluation Metrics

The evaluation metrics used are accuracy, F1 score, precision, recall, and confusion matrix and ROC Curve. Here is an explanation of each metric and its significance:

6.1.1. Accuracy

It is the percentage of correctly classified instances out of the total cases in the dataset. Accuracy is a widely used metric in classification problems as it provides a single measure that can be used to assess different models quickly. However, it can be misleading in cases of class imbalance, where the majority class dominates the classification results.

In a multi-class problem, it can be calculated as:

Accuracy = (Sum of correct predictions for all classes) / (Total instances)

6.1.2 F1 Score

This metric is the harmonic mean of precision and recall. The F1 score is helpful in situations with a class imbalance or when false positives and negatives have different costs. It ranges from 0 (worst) to 1 (best). A high F1 score indicates a better balance between precision and recall, essential in many real-world applications. The F1 score can be calculated for multiclass problems using macro-averaging, micro-averaging, or weighted averaging. The weighted average F1 score computes the F1 score for each class independently, then takes the average considering the number of instances in each class. The formula for the weighted average F1 score is as follows:

Weighted Average F1 Score = Σ (class_weight_c * (2 * Precision_c * Recall_c / (Precision_c + Recall_c)))

Where:

class_weight_c: The weight of class c, which is the proportion of instances in class c concerning the total samples.

Precision_c: The precision for class c.

Recall_c: The recall for class c.

6.1.3. Precision

Precision is the ratio of true positives to the sum of true and false positives. Precision measures how many instances classified as positive by the model are positive. In other words, it represents the correctness of optimistic predictions. It is a valuable metric when the cost of false positives is high. For multiclass problems, precision can be calculated using macro-averaging, micro-averaging, or weighted averaging. Weighted average precision computes the precision for each class independently, and then it takes the average considering the number of instances in each class. The formula for weighted average precision is as follows:

Weighted Average Precision = Σ (class_weight_c * (True Positives_c / (True Positives_c + False Positives_c)))

Where:

class_weight_c: The weight of class c, which is the proportion of instances in class c concerning the total samples.

True Positives_c: The number of actual positive instances in class c.

False Positives_c: The number of false positive instances in class c.

6.1.4. Recall

Sensitivity or actual positive rate is the ratio of true positives to the sum of true positives and false negatives. The recall is a measure of how many of the real positive instances the model can identify. It is a valuable metric when the cost of false negatives is high. For multiclass problems, recall can be calculated using macro-averaging,

micro-averaging, or weighted averaging. Weighted average recall computes the recall for each class independently, then takes the average considering the number of instances in each class. The formula for the weighted average recall is as follows:

Weighted Average Recall = Σ (class_weight_c * (True Positives_c / (True Positives_c + False Negatives_c)))

Where:

class_weight_c: The weight of class c, which is the proportion of instances in class c concerning the total samples. True Positives_c: The number of actual positive instances in class c. False Negatives_c: The number of false negative instances in class c.

6.1.5. Confusion Matrix

The confusion matrix is used to visualize the performance of the models. It shows the number of true positive, true negative, false positive, and wrong pessimistic predictions for each class. In a multi-class problem, the confusion matrix is a square matrix of size C x C, where C is the number of classes. Each row represents the instances in a predicted class, and each column represents the instances in an actual class. The diagonal elements of the matrix represent the correct predictions for each category, while the off-diagonal elements represent misclassifications. The rationale behind choosing these metrics is that they comprehensively evaluate the models, considering different aspects of their performance. In multiclass sentiment analysis, it is essential to consider the models based on their accuracy and ability to identify each class correctly. F1 score, precision, and recall give a deeper understanding of the model's performance regarding true positives, false positives, and false negatives. This is crucial in real-world applications where the costs of misclassifications may vary. Additionally, the confusion matrix is used to visualize the performance of the models. The confusion matrix shows the number of true positive, true negative, false positive, and wrong pessimistic predictions for each class. It helps identify the specific classes the model has difficulties with, providing insights into potential areas for improvement. Examining the confusion matrix makes it possible to understand how well the model generalizes to new data and whether it has any biases toward certain classes.

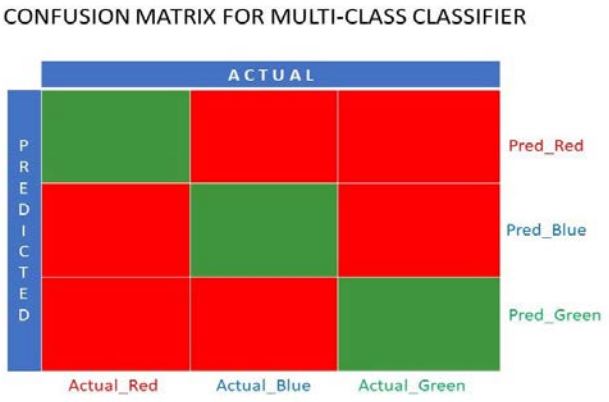

Figure 29. Confusion matrix for multi-class classifier (https://geekflare.com/confusion-matrix-in-machine-learning/)

6.1.6. Receiver Operating Characteristic (ROC curve)

The Receiver Operating Characteristic curve is a graphical representation that evaluates the effectiveness of a binary classifier system across various decision thresholds. By plotting the true positive rate (Sensitivity) against the false positive rate (1-Specificity) for different thresholds, the ROC curve provides insight into the trade-off between sensitivity and specificity. As a result, the model's precision is indicated by the proximity of the curve to the left border and upper border, maximizing the area under the curve. The ROC curve is crucial when examining a classifier's performance over a range of sensitivities and specificities. In this study, the ROC curve is extended to address a multi-class problem. The curve helps evaluate the model's performance by examining the interaction between sensitivity and specificity. This is particularly important when high sensitivity is desired, even at the cost of lower specificity. Unlike the confusion matrix, which evaluates a single classifier with a fixed threshold, the ROC curve's area under the curve (AUC) evaluates the classifier across all possible thresholds. Consequently, the ROC curve enables the assessment of the classifier's performance at various levels of sensitivity and specificity, making it an essential tool for multi-class problems.

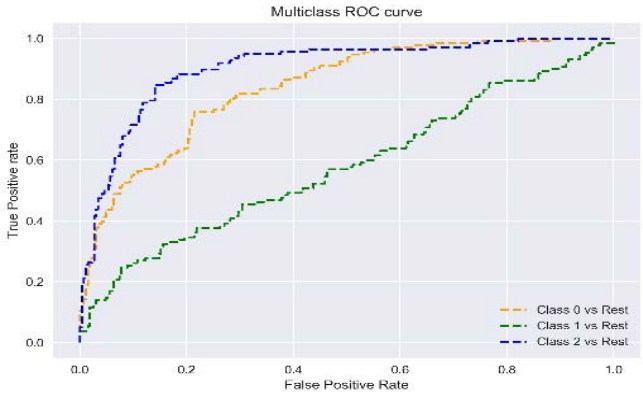

Figure 30. A multiclass roc curve (https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/)

7. Model Implementation

7.1.1. Implementation of the SVM Model

The parameters for the SVM algorithm based on the specific requirements of our sentiment analysis problem. To handle textual data, we used the TfidfVectorizer to convert the text into feature vectors.

```
# Create the TfidfVectorizer
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_val_tfidf = vectorizer.transform(X_val)
X_test_tfidf = vectorizer.transform(X_test)
```

Figure 31. The creation of the tfidfvectorizer for the SVM model

```
class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(y_train), y=y_train)
class_weight_dict = dict(enumerate(class_weights))

svm_clf = SVC(probability=True, class_weight=class_weight_dict)
```

Figure 32. Initializing the class weight to handle class imbalance for the Svm model.

	text	processed_text	sentiment_label	SVM_predicted_sentiment
7632	@xianren88338098 @pradeeepk @MarceloPLima ...bec...	cheap try v stock_unh stock_ko	1	2
7633	@SteveWagsInvest Good examples) I hold $COST, ...	good example hold stock_cost stock_unh stock_j...	1	2
7634	#Coca-ColaCo $KO Last 3 months Daily #StockMov...	colaco stock_ko last 3 month daily line max do...	1	1
7635	$KO "Top analyst price target for next week\nh...	stock_ko top analyst price target next week	1	2
7636	🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯 Congrats to all Team members ...	congrats team member making huge gain trade id...	1	2
7637	@suburbanaces @TheCorp_oration I like $KO insi...	like stock_ko inside week	2	1
7638	Daily chart bullish swing flagged for $KO on 1...	daily chart bullish swing flagged stock_ko 11 ...	0	0
7639	$KO Top Analyst Ratings and Price Targets for ...	stock_ko top analyst rating price target week	1	1
7640	$ko is set and ready for a run up:. ~\nhttps:...	stock_ko set ready run	1	2
7641	#Other\n#MergersAcquisitions #Manufacturing \n...	stock_ko stock_gm stock_wolwf stock_lrlcf 2022...	1	2

Figure 33. SVM model showing the predicted sentiment on test set.

We also used the class_weight parameter in the SVC function to account for class imbalance in the dataset, which can significantly affect the classifier's performance.

The training and testing process involved splitting the dataset into training, validation, and testing sets. The SVM algorithm was trained on the training set, and the model's effectiveness was evaluated using the validation set. Finally, the model's performance on unseen data was assessed using the test set. We reported various evaluation metrics, including accuracy, F1 score, precision, recall, confusion matrix, training and test run times, training, Validation, and test losses, to comprehensively understand the model's performance.

7.1.2. Implementation of the RNN Model

In the implementation, deep learning sentiment analysis classifier using PyTorch, GloVe word embeddings, and a bidirectional LSTM network. A bidirectional LSTM RNN captures both the forward and backward dependencies in the text. LSTM is a variant of RNN that can understand long-range dependencies in the data, making it well-suited for tasks like sentiment analysis, where context plays a crucial role.

The model architecture consists of the following layers:

```
class SentimentNet(nn.Module):
    def __init__(self, embedding_matrix, hidden_dim, num_classes, dropout_rate=0.5):
        super(SentimentNet, self).__init__()

        self.embedding = nn.Embedding.from_pretrained(embedding_matrix)
        embedding_dim = self.embedding.embedding_dim
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, batch_first=True, bidirectional=True)
        self.dropout = nn.Dropout(dropout_rate)
        self.fc = nn.Linear(hidden_dim * 2, num_classes)

    def forward(self, text):
        embedded = self.embedding(text)
        lstm_out, (h_n, c_n) = self.lstm(embedded)
        text_representation = torch.cat((h_n[-1, :, :], h_n[-2, :, :]), dim=1)
        text_representation = self.dropout(text_representation)
        logits = self.fc(text_representation)

        return logits
```

Figure 34. The PyTorch class defining the RNN model layers.

An embedding layer that converts input words into fixed-sized dense vectors using the pre-trained GloVe word embeddings. These embeddings result from training on large text corpora and capture semantic relationships between terms. One layer converts input words into dense vectors. The size of the dense vectors is 300, which matches the dimensions of the pre-trained GloVe word embeddings.

A bidirectional LSTM layer is a type of RNN that addresses the vanishing gradient problem, enabling the model to learn long-term dependencies. The bidirectional LSTM processes the input sequence forward and backward, which helps capture the context from both directions—a bidirectional LSTM layer with 128 hidden units/neurons in each direction (256 in total). The LSTM has a default tanh activation function for the cell state and output gate and a sigmoid activation function for the input and forget gates. A dropout layer is used for regularization, reducing the risk of overfitting by randomly dropping a certain proportion of the neurons during training. The dropout rate is set to 0.5 in this case.A fully connected (linear) layer that maps the LSTM output to the number of classes (unique sentiment labels) and outputs logits. One layer with output neurons equals the number of unique sentiment labels (num_classes).

The training process involves splitting the data into training, validation, and test sets to evaluate the model's performance. They define the class weights, loss function, and optimizer. In this case, the class weights are inversely proportional to the class frequencies, the loss function is the cross-entropy loss, and the optimizer used is Adam, with a learning rate of 0.001.

The model was trained for ten epochs, where a period represents a complete pass through the training dataset. The model's parameters are updated during each epoch using backpropagation and the optimizer. After each epoch, the model's performance is evaluated on various evaluation metrics, including accuracy, F1 score, precision, recall, confusion matrix, training and test run times, training, Validation, and test losses to understand the model's performance comprehensively.

7.1.3. Implementation of the BERT Model

In this implementation using PyTorch, The BERT algorithm is designed to understand the context in the text by considering both the left and right context of each word in a sentence. It leverages a masked language model objective to train on large-scale unsupervised data. The model used in this project is the "Bert-base-uncased" version, which is a pre-trained BERT model with 12 layers, 768 hidden units, and 110 million parameters.

The model architecture consists of two main parts: a pre-trained BERT model and a custom classifier. The classifier is a linear layer that maps the output of the BERT model to the number of unique sentiment labels (num_classes). The classifier is a sequential neural network with one hidden layer and a ReLU activation function. The project begins by preparing the data for training, validation, and testing by splitting the dataset into respective sets. The data is then tokenized and encoded using the BERT tokenizer, which converts the input text into a format the BERT model can understand. The maximum length for encoding (MAX_LEN) is determined based on the most extended encoded sequence in the dataset.

```
# Load GloVe embedding
glove = gensim_api.load("glove-wiki-gigaword-300")

# Build vocabulary and word2index mapping
all_train_tokens = [token for text in X_train for token in tokenize(text)]
counter =Counter(all_train_tokens)
word2index = {word: idx for idx, (word, _) in enumerate(counter.most_common(10000), start=2)}
word2index["<pad>"] = 0
word2index["<unk>"] = 1

# Create an embedding matrix from the GloVe embedding
embedding_matrix = torch.zeros(len(word2index), 300)
for word, idx in word2index.items():
    embedding_matrix[idx] = torch.tensor(glove[word]) if word in glove else torch.randn(300)
```

Figure 35. Preparing the necessary data structures to use pre-trained GloVe word embeddings in the rnn model.

```
# Define the class weights, loss function, and optimizer
class_weights = torch.tensor([1.0 / class_freq for class_freq in np.bincount(y_train)], dtype=torch.float32).to(device)

criterion = nn.CrossEntropyLoss(weight=class_weights)

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

Figure 36. Initializing the class weights, Loss function and optimizer for the Svm model.

	text	processed_text	sentiment_label	RNN_predicted_sentiment
7632	@xianren88338098 @pradeeepk @MarceloPLima ...bec...	cheap try v stock unh stock ko	1	2
7633	@SteveWagsInvest Good examples) I hold $COST, ...	good example hold stock cost stock unh stock j...	1	2
7634	#Coca-ColaCo $KO Last 3 months Daily #StockMov...	colaco stock ko last 3 month daily line max do...	1	1
7635	$KO "Top analyst price target for next week\nh...	stock ko top analyst price target next week	1	2
7636	🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯🕯Congrats to all Team members ...	congrats team member making huge gain trade id...	1	1
7637	@suburbanaces @TheCorp_oration I like $KO insi...	like stock ko inside week	2	1
7638	Daily chart bullish swing flagged for $KO on 1...	daily chart bullish swing flagged stock ko 11 ...	0	0
7639	$KO Top Analyst Ratings and Price Targets for ...	stock ko top analyst rating price target week	1	2
7640	$ko is set and ready for a run up:. ~\nhttps:...	stock ko set ready run	1	2
7641	#Other\n#MergersAcquisitions #Manufacturing \n...	stock ko stock gm stock wolwf stock lrlcf 07 0...	1	2

Figure 37. RNN model showing the predicted sentiment on test set.

```
# Define the Bert NLP Classifier
class BertClassifier(nn.Module):
    def __init__(self, num_classes, freeze=False):
        super(BertClassifier, self).__init__()

        # Define the neurons for the final layer
        input_layer = 768
        hidden_layer = 50
        output_layer = num_classes

        # Use the pretrained Bert model for first section of NN
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        # Define a final layer to attach to the Bert model for custom classification
        self.classifier = nn.Sequential(
            nn.Linear(input_layer, hidden_layer),
            nn.ReLU(),
            nn.Linear(hidden_layer, output_layer))
         # Freeze the model from updating
        if freeze:
            for param in self.bert.parameters():
                param.requires_grad = False

     # Return classification from Bert model
    def forward(self, input_ids, attention_mask):
        outputs = self.bert(input_ids, attention_mask)
        h_cls = outputs[0][:, 0, :]
        logits = self.classifier(h_cls)

        return logits
```

Figure 38. The PyTorch class defining the BERT Model classifier.

```
# Prepare the Bert NLP model tokenizer to encode tweets
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)

# Encode the tweets for Bert model
def preprocessing_for_bert(data):
    input_ids = []
    attention_masks = []
    # For each tweet
    for line in data:
        # encode the data. Return input encoding and attention mask
        encoding = tokenizer.encode_plus(
                text=line, # data to process
                add_special_tokens=True, # adds special chars [CLS] and [SEP] to encoding
                padding='max_length', # pad the tweets with 0s to fit max length
                max_length = MAX_LEN, # assign max length
                truncation=True, # truncate tweets longer than max length
                return_tensors="pt", # return tensor as pytorch tensor
                return_attention_mask=True # return the attention mask
                )
        # add the encodings to the list
        input_ids.append(encoding.get('input_ids'))
        attention_masks.append(encoding.get('attention_mask'))
    # return the lists as tensors
    input_ids = torch.cat(input_ids, dim=0)
    attention_masks = torch.cat(attention_masks, dim=0)

    return input_ids, attention_masks

# Use this to determine max length for encoding
encoded = [tokenizer.encode(sent, add_special_tokens=True) for sent in data['processed_text'].values]
MAX_LEN = max([len(sent) for sent in encoded])
print('Max length: ', MAX_LEN)
```

Figure 39. Function to preprocess the BERT classifier.

```
# Define model hyperparameters
batch_size = 32
epochs = 10
steps = len(train_dataloader) * epochs
learning_rate = 5e-5
epsilon = 1e-8

# Define Adam optimizer
optimizer = AdamW(model.parameters(), lr=learning_rate, eps=epsilon)

# Define scheduler for training the optimizer
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=steps)

# Define weighted cross entropy loss function to account for class imbalance
weights = torch.tensor([1.0, 2.0, 1.0]).to(device)
loss_function = nn.CrossEntropyLoss(weight=weights)
```

Figure 40. Defining the hyperparameter, optimizer, scheduler, and loss function for the BERT model.

	text	processed_text	sentiment_label	BERT_predicted_sentiment
7632	@xianren88338098 @pradeeepk @MarceloPLima ...bec...	cheap try v stock unh stock ko	1	2
7633	@SteveWagsInvest Good examples) I hold $COST, ...	good example hold stock cost stock unh stock j...	1	2
7634	#Coca-ColaCo $KO Last 3 months Daily #StockMov...	colaco stock ko last 3 month daily line max do...	1	1
7635	$KO "Top analyst price target for next week\nh...	stock ko top analyst price target next week	1	2
7636	🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌🙌 Congrats to all Team members ...	congrats team member making huge gain trade id...	1	1
7637	@suburbanaces @TheCorp_oration I like $KO insi...	like stock ko inside week	2	1
7638	Daily chart bullish swing flagged for $KO on 1...	daily chart bullish swing flagged stock ko 11 ...	0	0
7639	$KO Top Analyst Ratings and Price Targets for ...	stock ko top analyst rating price target week	1	1
7640	$ko is set and ready for a run up:. ~\nhttps:...	stock ko set ready run	1	2
7641	#Other\n#MergersAcquisitions #Manufacturing \n...	stock ko stock gm stock wolwf stock lrlcf 07 0...	1	1

Figure 41. BERT model showing the predicted sentiment on test set.

Hyperparameters for training the model include the batch size, which is 32, the learning rate of 5e-5, and epsilon 1e-8. The AdamW optimizer and a linear learning rate scheduler train the model. A weighted cross-entropy loss function accounts for class imbalance in the dataset.

The training and evaluation loop consists of ten epochs. In each generation, the model undergoes training on the training set and its performance is assessed by evaluating on the validation set. The training and validation losses are recorded, as well as the validation accuracy. After training, the model was evaluated on various evaluation metrics, such as accuracy, F1 score, precision, recall, confusion matrix, training and test run times, training, Validation, and test losses to understand the model's performance comprehensively.

8. Results and Discussion

8.1. Algorithm Evaluation

Algorithm evaluation plays a crucial role in understanding the effectiveness and reliability of a model or algorithm. The evaluation process typically involves comparing the result the predicted outputs with that of the actual outputs, assessing various performance metrics, and analysing the results to draw conclusions about the algorithm's performance. This process can be broken down into two main aspects: results and discussion.

8.1.1. Support Vector Machine (SVM)

The performance results of Support Vector Machine algorithm for sentiment analysis based on the chosen metrics is outlined here. The evaluation includes test and validation accuracies, precision, recall, F1 score, training, validation and test loss, training and testing time, confusion matrix and ROC Curve.

```
Training Time: 16m 15s
Validation Accuracy: 88.16% | Validation Loss: 0.308
Test Time: 0m 35s
Test Accuracy: 89.15% | Test Loss: 0.288
Training Loss: 0.071
F1 Score: 89.07% | Precision: 89.74% | Recall: 89.15%
```

Figure 42. SVM Model Result Output.

The SVM algorithm performed well on the sentiment analysis task, achieving an overall test accuracy of 89.15%. These results show that the SVM model performs quite well on both the training data and the unseen validation and test data. The model has high accuracy and F1 scores, and relatively low loss values, suggesting that it is a good fit for the data and effective at predicting sentiment in stock-related tweets.

The confusion matrix gives further insight into the performance of the SVM model. The elements on the diagonal reflect accurate predictions for every class, whereas the elements off the diagonal indicate instances of incorrect classification. The model performed well in distinguishing between classes, with relatively few misclassified instances.

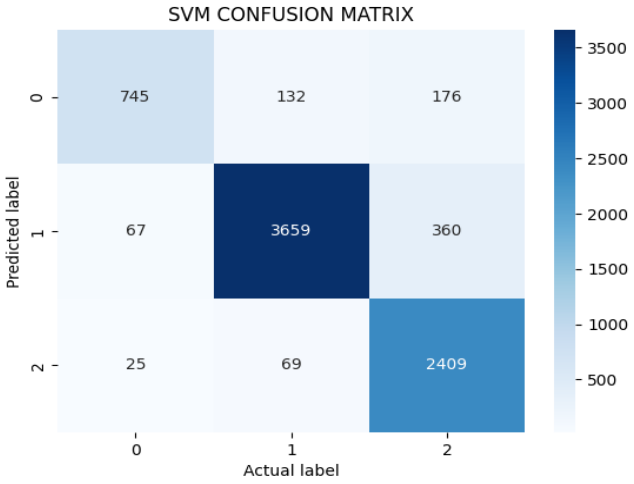

Figure 43. SVM Model Confusion Matrix

Classification Report:

	precision	recall	f1-score	support
0	0.89	0.71	0.79	1053
1	0.95	0.90	0.92	4086
2	0.82	0.96	0.88	2503
accuracy			0.89	7642
macro avg	0.89	0.86	0.86	7642
weighted avg	0.90	0.89	0.89	7642

Figure 44. SVM Model Classification Report

The classification report offers a detailed overview of the model's precision, recall, and F1 score for individual classes, with the overall accuracy, macro average, and weighted average. The classifier demonstrates strong performance in predicting sentiment for the three classes. These metrics indicate that the model effectively predicts tweet sentiments, with a well-balanced precision and recall across the three classes, showcasing its strong performance in sentiment classification.

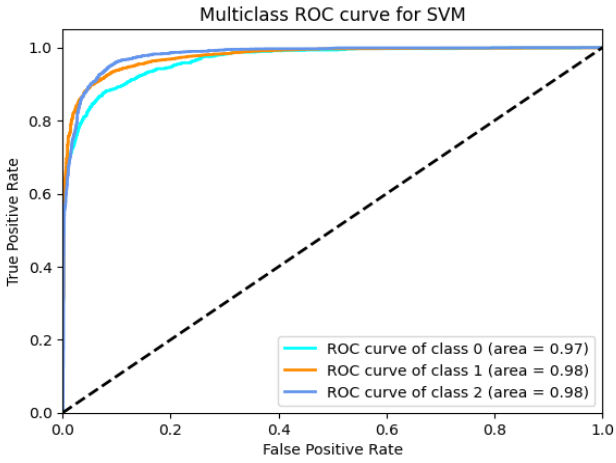

Figure 45. SVM Model Multi Class ROC Curve

The classifier demonstrates a high performance in distinguishing between the three classes. For class 0, the AUC value of 0.97 indicates a very high performance in separating class 0 from the other classes. Similarly, for class 1, the AUC value of 0.98 shows that the classifier is performing very well in distinguishing class 1 from

the other classes. In the case of class 2, the classifier also has an AUC value of 0.98, which is very near to the perfect value of 1.0. Overall, the multiclass ROC curve analysis reveals that the model is well-trained and effective for this classification problem.

8.1.2. Deep Learning with RNN

The performance results of Deep Learning with the RNN algorithm for sentiment analysis based on the chosen metrics is outlined here. The evaluation includes test and validation accuracies, precision, recall, F1 score, training, validation and test loss, training and testing time, confusion matrix and ROC Curve.

```
Epoch [1/20], Train Loss: 0.955, | Train Run Time: 0m 7s, Val Loss: 0.764,| Val Acc: 72.10%
Epoch [2/20], Train Loss: 0.713, | Train Run Time: 0m 5s, Val Loss: 0.656,| Val Acc: 73.67%
Epoch [3/20], Train Loss: 0.641, | Train Run Time: 0m 5s, Val Loss: 0.614,| Val Acc: 76.15%
Epoch [4/20], Train Loss: 0.598, | Train Run Time: 0m 6s, Val Loss: 0.581,| Val Acc: 77.49%
Epoch [5/20], Train Loss: 0.571, | Train Run Time: 0m 5s, Val Loss: 0.570,| Val Acc: 78.28%
Epoch [6/20], Train Loss: 0.551, | Train Run Time: 0m 6s, Val Loss: 0.546,| Val Acc: 77.79%
Epoch [7/20], Train Loss: 0.533, | Train Run Time: 0m 5s, Val Loss: 0.535,| Val Acc: 79.06%
Epoch [8/20], Train Loss: 0.517, | Train Run Time: 0m 6s, Val Loss: 0.529,| Val Acc: 78.48%
Epoch [9/20], Train Loss: 0.507, | Train Run Time: 0m 5s, Val Loss: 0.522,| Val Acc: 79.16%
Epoch [10/20], Train Loss: 0.495, | Train Run Time: 0m 5s, Val Loss: 0.511,| Val Acc: 78.84%
Epoch [11/20], Train Loss: 0.487, | Train Run Time: 0m 6s, Val Loss: 0.507,| Val Acc: 80.34%
Epoch [12/20], Train Loss: 0.477, | Train Run Time: 0m 5s, Val Loss: 0.502,| Val Acc: 79.29%
Epoch [13/20], Train Loss: 0.471, | Train Run Time: 0m 6s, Val Loss: 0.496,| Val Acc: 79.85%
Epoch [14/20], Train Loss: 0.465, | Train Run Time: 0m 5s, Val Loss: 0.506,| Val Acc: 79.59%
Epoch [15/20], Train Loss: 0.457, | Train Run Time: 0m 6s, Val Loss: 0.495,| Val Acc: 80.14%
Epoch [16/20], Train Loss: 0.452, | Train Run Time: 0m 5s, Val Loss: 0.493,| Val Acc: 80.57%
Epoch [17/20], Train Loss: 0.448, | Train Run Time: 0m 5s, Val Loss: 0.491,| Val Acc: 80.01%
Epoch [18/20], Train Loss: 0.445, | Train Run Time: 0m 6s, Val Loss: 0.490,| Val Acc: 80.27%
Epoch [19/20], Train Loss: 0.441, | Train Run Time: 0m 5s, Val Loss: 0.493,| Val Acc: 80.47%
Epoch [20/20], Train Loss: 0.441, | Train Run Time: 0m 6s, Val Loss: 0.493,| Val Acc: 80.44%
Test Loss: 0.499, |Test Acc: 81.39%, |Test Run Time: 0m 1s
F1 Score: 81.76%, |Precision: 82.70%, |Recall: 81.39%
```

Figure 46. RNN Model Epoch Output

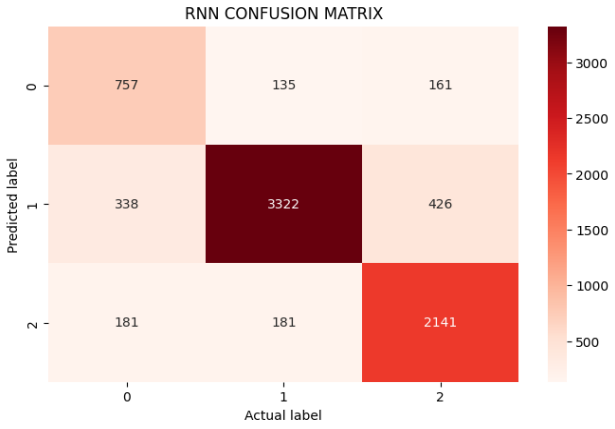

Figure 47. Rnn Model Confusion Matrix

The confusion matrix gives further insight on the value of true positive, false positive, true negative, and wrong predictions for individual class. The elements on the diagonal reflect accurate predictions for every class, whereas the elements off the diagonal indicate instances of incorrect classification.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.59      0.72      0.65      1053
           1       0.91      0.81      0.86      4086
           2       0.78      0.86      0.82      2503

    accuracy                           0.81      7642
   macro avg       0.76      0.80      0.78      7642
weighted avg       0.83      0.81      0.82      7642
```

Figure 48. RNN Model Classification Report

The model performs best when identifying positive sentiments (class 1), with good precision and recall. It performs least well with negative sentiments (class 0), with the lowest precision and recall among the classes. Its performance with positive sentiments (class 2) is relatively good.

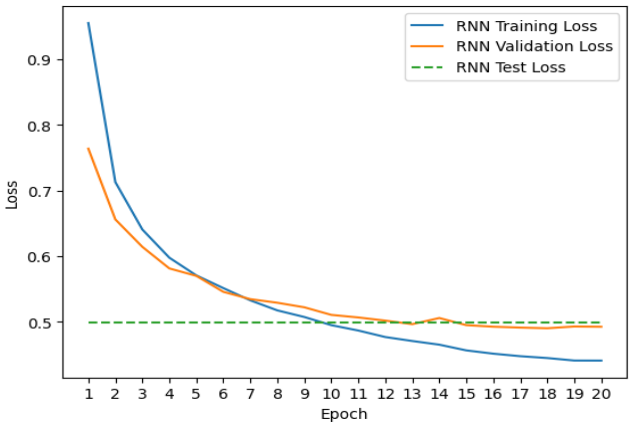

Figure 49. RNN Model Loss Plot

The training, validation, and test losses all show a decreasing trend, indicating that the model is effectively learning from the training data and generalizing well to the unseen validation and test data. Specifically, the training loss starts at 0.955 in epoch 1 and decreases to 0.441 in epoch 20, showing that the model is learning, and the error is decreasing over time. The validation loss starts at 0.764 in epoch 1 and decreases to 0.493 in epoch 20, suggesting that the model is generalizing well to the unseen validation data as the error is decreasing. Finally, the test loss is 0.499, which is relatively near to the final validation loss, showing that the model is also generalizing well to the test data.

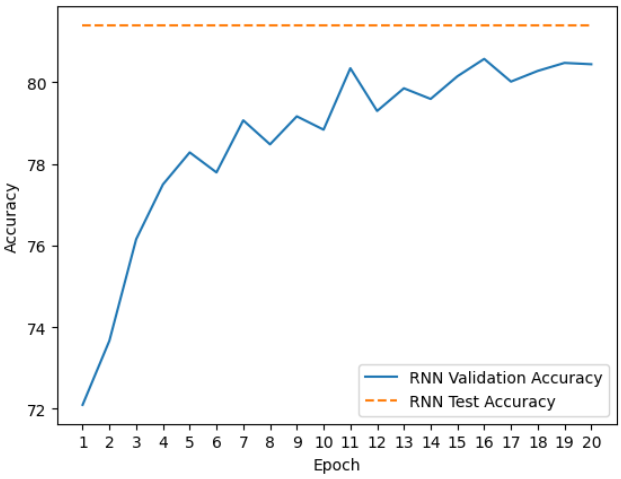

Figure 50. RNN Model Accuracy Plot

The validation and test accuracies both show an increasing trend, indicating that the model is effectively learning from the training data and learning well to the unseen validation and test data. Specifically, the validation accuracy starts at 72.10% in epoch 1 and increases to 80.44% in epoch 20, suggesting that the model is generalizing well to the unseen validation data as its ability to make correct predictions improves. Similarly,

the test accuracy is 81.39%, which is close to the final validation accuracy, indicating that the model is also generalizing well to the test data and effectively making correct predictions.

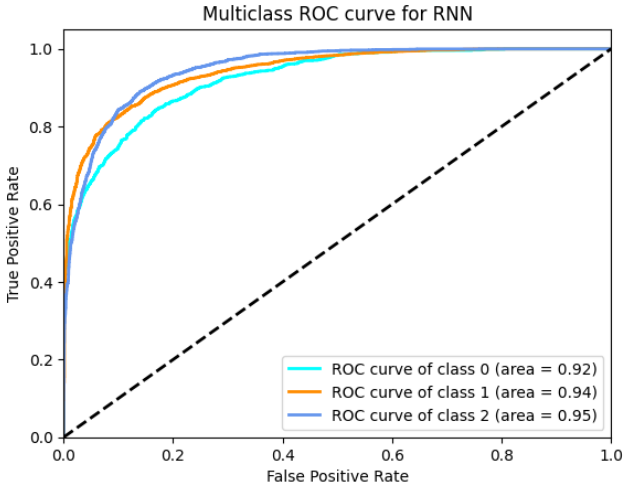

Figure 51. RNN Model Multi Class ROC Curve

The AUC values for each sentiment class in the multi-class classification problem are close to 1, indicating excellent classifier performance across all three classes: Class 0 (Negative sentiment) with an AUC of 0.92, Class 1 (Positive sentiment) with an AUC of 0.94, and Class 2 (Neutral sentiment) with an AUC of 0.95. This suggests that the classifier is doing a great job in sentiment classification for all three classes.

8.1.3. Transformer Model using BERT

The performance results of the Transformer Model using the Bert algorithm utilized for sentiment analysis as per chosen metrics is outlined here. The evaluation includes test and validation accuracies, precision, recall, F1 score, training, validation and test loss, training and testing time, confusion matrix and ROC Curve.

The confusion matrix gives further insight on the value of true positive, false positive, true negative, and wrong predictions for individual class. The elements on the diagonal reflect accurate predictions for every class, whereas the elements off the diagonal indicate instances of incorrect classification.

```
Epoch: 1 | Train Loss: 0.305 | Val Loss: 0.215 | Val Acc: 91.59% | Train Run Time: 11m 47s
Epoch: 2 | Train Loss: 0.181 | Val Loss: 0.206 | Val Acc: 92.35% | Train Run Time: 11m 47s
Epoch: 3 | Train Loss: 0.123 | Val Loss: 0.232 | Val Acc: 92.15% | Train Run Time: 11m 48s
Epoch: 4 | Train Loss: 0.081 | Val Loss: 0.317 | Val Acc: 92.51% | Train Run Time: 11m 47s
Epoch: 5 | Train Loss: 0.054 | Val Loss: 0.334 | Val Acc: 92.38% | Train Run Time: 11m 47s
Epoch: 6 | Train Loss: 0.037 | Val Loss: 0.393 | Val Acc: 92.44% | Train Run Time: 11m 48s
Epoch: 7 | Train Loss: 0.026 | Val Loss: 0.457 | Val Acc: 92.51% | Train Run Time: 11m 47s
Epoch: 8 | Train Loss: 0.020 | Val Loss: 0.489 | Val Acc: 92.51% | Train Run Time: 11m 47s
Epoch: 9 | Train Loss: 0.012 | Val Loss: 0.475 | Val Acc: 92.84% | Train Run Time: 11m 47s
Epoch: 10 | Train Loss: 0.010 | Val Loss: 0.495 | Val Acc: 92.77% | Train Run Time: 11m 47s
Test Loss: 0.443 | Test Acc: 93.21%
F1 Score: 93.15% | Precision: 93.15% | Recall: 93.21%
Test Run Time: 1m 9s
```

Figure 52. BERT Model Epoch Output

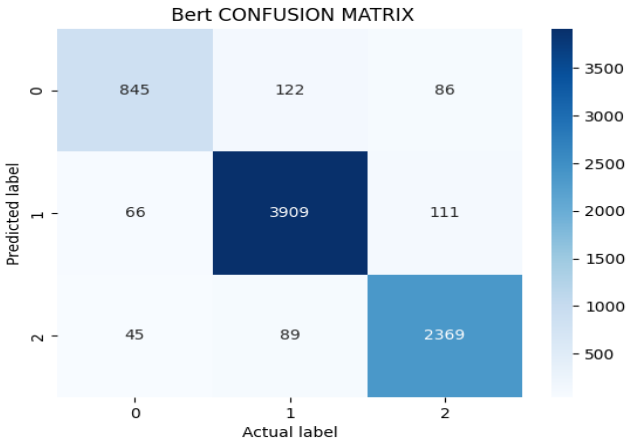

Figure 53. BERT Model Confusion Matrix

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.80      0.84      1053
           1       0.95      0.96      0.95      4086
           2       0.92      0.95      0.93      2503

    accuracy                           0.93      7642
   macro avg       0.92      0.90      0.91      7642
weighted avg       0.93      0.93      0.93      7642
```

Figure 54. BERT Model Classification Report

The classification report demonstrates very strong performance in predicting sentiment for the three classes. It shows good results for all classes.

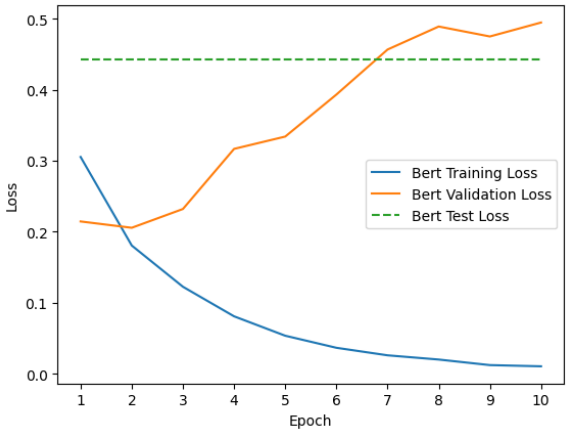

Figure 55. BERT Model Loss Plot

The model's training loss consistently decreases across the epochs, from 0.305 in epoch 1 to 0.010 in epoch 10, demonstrating effective learning from the training data. However, the validation loss, after an initial decrease from 0.215 in epoch 1 to 0.206 in epoch 2, starts to increase, peaking at 0.495 in epoch 10. This increase suggests the model could potentially be overfitting to the training data, as it starts performing worse on the unseen validation data. Despite this, the test loss is at 0.443, which is relatively close to the validation loss in the final epoch, indicating that the model still generalizes reasonably well to completely unseen test data.

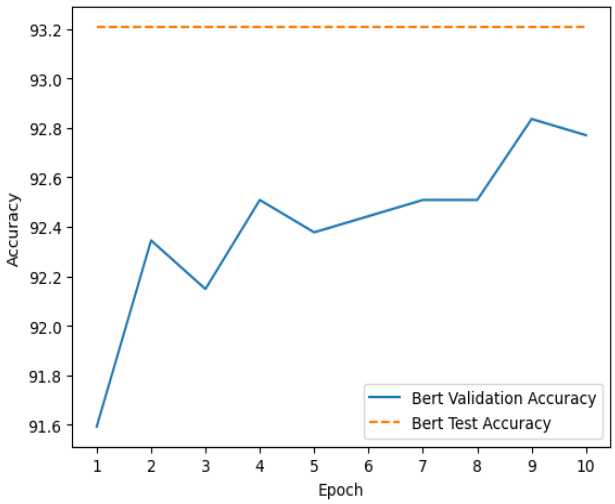

Figure 56. BERT Model Accuracy Plot

The model's validation accuracy demonstrates some fluctuations but generally follows an increasing trend. It starts at 91.59% in epoch 1 and reaches its peak at 92.84% in epoch 9. However, after epoch 2, the improvements in accuracy are relatively minor, suggesting that the model's predictive power on unseen data stabilizes after initial epochs. Despite this, the test accuracy is at a commendable 93.21%, which aligns closely with the validation accuracy in the final epochs, indicating a consistent and reliable model performance when making predictions on new, unseen data.

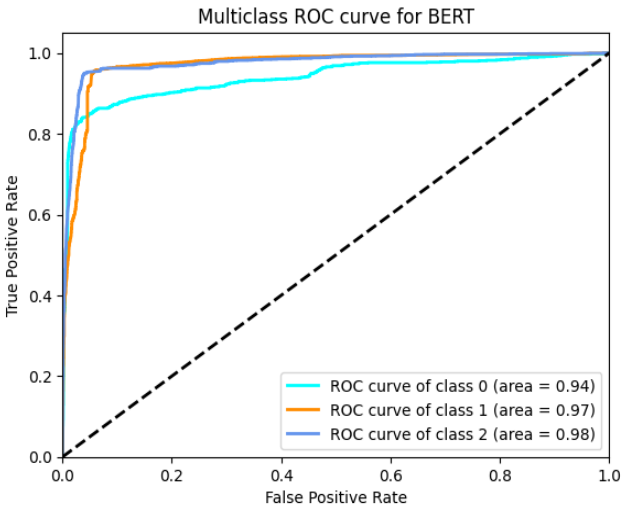

Figure 57. BERT Model Multi Class ROC Curve

The multi-class Bert ROC curve output indicates strong classifier performance in distinguishing each class from the others. The AUC values are 0.94 for class 0, suggesting strong performance, 0.97 for class 1 and 0.98 for class 2, indicating excellent and outstanding performance, respectively. Overall, the high AUC values demonstrate the classifier's effectiveness in separating each class from the rest.

The findings of the three algorithms indicates that the BERT model surpass both the SVM and RNN models regarding test accuracy, F1 score, precision, recall, and AUC values across all classes.

8.2. Model Comparison

Model comparison is a crucial step before deploying any machine learning model. This process evaluates the performance and accuracy of various models using a range of techniques to estimate their generalization accuracy on future predictions. The primary goal is to ensure that the selected model meets the requirements and to identify any flaws if it falls short. Model comparison methods are essential to compare different models and determine the most suitable and accurate option to achieve the desired goals.

In Natural Language Processing, factors such as prediction quality, speed, scalability, robustness, interpretability, and adaptability determine a model's effectiveness and usefulness. For RNN and BERT models, the last epoch is typically used for model comparison. It is crucial to evaluate models using various methods to estimate and select the best-fitted model for the observed dataset, ensuring that users can effectively utilize the outcomes obtained from the chosen model.

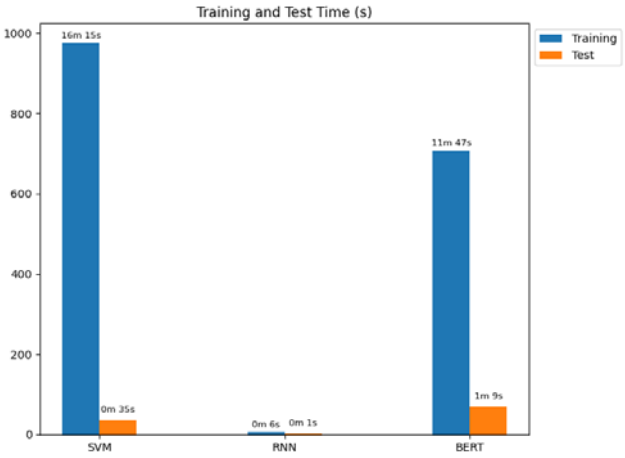

Figure 58. Training and Testing Time Comparison Plot

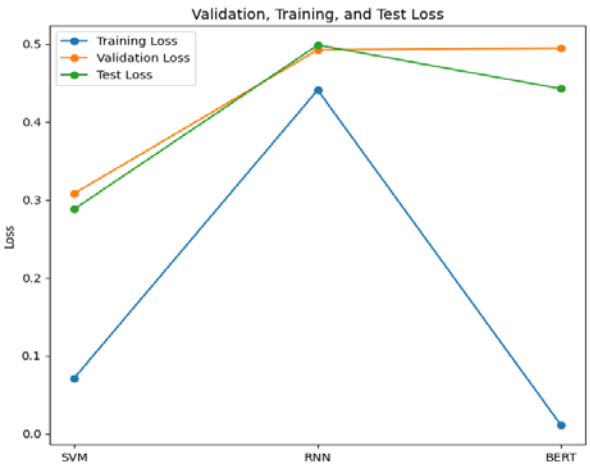

Figure 59. Training, Testing and Validation Loss Comparison Plot

Comparing the SVM, RNN, and BERT models based on training and test run times reveals distinct performance characteristics. The SVM model has the longest training time which only runs for a single epoch compared to RNN and BERT, but relatively short test run time, while the RNN model boasts the shortest training and test run times, making it suitable for real-time applications. The BERT

model has a moderate training time due to pretraining and fine-tuning, but the longest test run time due to its complexity.

When comparing the SVM, RNN, and BERT models based on training, validation, and test losses, the BERT model shows the best performance on training data, followed by the SVM and RNN models. However, all models exhibit some degree of overfitting, as evidenced by the higher validation and test losses.

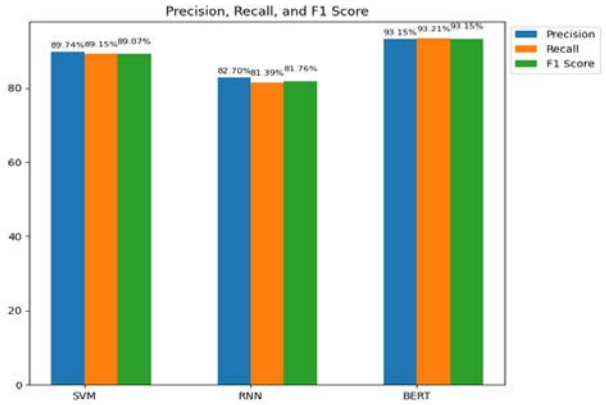

Figure 60. Precision, Recall and F1 Score Comparison Plot

When comparing the SVM, RNN, and BERT models based on precision, recall, and F1 scores, we observe that the BERT model outperforms the other two with the highest scores across all metrics, indicating superior classification accuracy and balance between false positives and false negatives. The SVM model follows with relatively high scores, while the RNN model lags slightly behind.

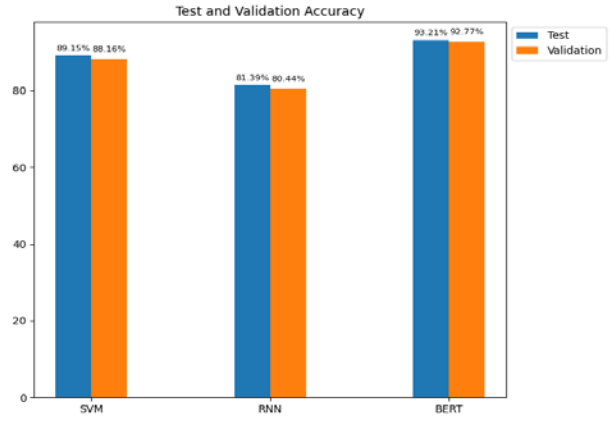

Figure 61. Test and Validation Accuracy Comparison Plot

When comparing the SVM, RNN, and BERT models based on test and validation accuracies, the BERT model outperforms the others with the highest accuracies, indicating superior generalization and prediction capabilities on unseen data. The SVM model follows with relatively high accuracies, while the RNN model lags.

Table 12. Comparison based on Optimizer, Epoch Size, Batch Size, Learning rate for RNN and BERT

Algorithm	Optimizer	Epoch Size	Batch Size	Learning Rate
RNN	Adam	20	16	4e-5
BERT	AdamW	10	32	5e-5

8.3. Insights and Findings

The findings indicate the superior performance of the BERT model over SVM and RNN models in sentiment analysis of financial tweets, scoring higher in test accuracy, F1 score, precision, and recall. This can be attributed to BERT's pre-trained contextual embeddings, which offer an understanding of natural language, crucial in financial analyses where sentiments in tweets can significantly influence market trends.

While SVM and RNN models also perform well, demonstrating their ability to discern some sentiment nuances in text, BERT's superior performance underscores the advantage of transformer models for sentiment analysis tasks.

The BERT model also shows more consistency and accuracy across different class predictions than SVM and RNN, likely due to its better contextual understanding and word interrelationships, enhancing sentiment prediction accuracy.

In this project, among the three models BERT is the best model for sentiment analysis in stock market contexts, thanks to its superior accuracy and ability to capture a nuanced sentiment in natural language. While SVM and RNN models have their merits, BERT offers a more robust and accurate solution for predicting sentiment in financial tweets.

8.4. Limitations and Challenges

In this section, we examine the project's limitations and challenges that may have influenced the models' performance or results interpretation.

The quality and volume of the training and evaluation data can significantly affect the models' performance. Any inconsistencies, noise, or biases in the data could have led to skewed results. Gathering sufficient Twitter stock data was time-consuming, especially for portfolios requiring more analysis data within a year.

Lack of GPU capabilities was also an issue, necessitating the search for free online GPU resources for model running. This limitation could have influenced model choice, training duration, and hyperparameter tuning. BERT, despite outperforming other algorithms, is complex and resource-intensive, requiring significant computational power and extended training times, posing potential challenges for real-time or large dataset applications.

Hyperparameter choices can also impact model performance. Despite some optimization, there might be potential for further refinement. However, exhaustive hyperparameter tuning would demand considerable computational resources and time.

Class imbalance in the dataset could hinder accurate prediction for underrepresented classes, negatively affecting their performance. Weighted loss functions were used to address this issue.

Lastly, while BERT is highly accurate, it may be less interpretable compared to simpler models like SVM, making it harder to understand the decision-making process or determine key features for accurate sentiment prediction.

Addressing these constraints in future work could enhance performance and offer more in-depth insight into the models' sentiment prediction capabilities for stock market applications.

9. Stock Analysis and Investment Strategies

9.1. Using best Model to Classify the Sentiment of Ten Unseen Stock Tweets

In this project phase, the best-performing algorithm, the BERT Classifier, is used to classify the sentiment of tweets related to 10 unseen stocks. The BERT Classifier is a powerful deep learning model with excellent performance in various NLP activities, such as sentiment analysis.

The sentiment classification is combined and added as a new column called 'Sentiment' to the stock data frame and finally, the stock data frame, which now includes the sentiment predictions as it was in the sentiment analysis phase, is saved to a new CSV file with the stock ticker symbol as the file name.

```
Adbe - completed
Fdx - completed
Ebay - completed
Tm - completed
Pep - completed
Orcl - completed
Rio - completed
Hpq - completed
Azn - completed
Bp - completed
```

Figure 62. Output of the Sentiment Classified Ten Stocks

9.2. Combining the stock inputs

The steps to analyse some of the important columns in the dataset and merge the classified stock tweet data and their respective historical price data for all ten stocks. Firstly, stocks tweet data is read and processed, with each tweet's weight calculated based on the number of followers and retweets. This ensures that influential users and tweets with high engagement rates significantly impact the overall sentiment score.

```
# Initialize Weights and Tweet numbers
data['Tweets'] = 1
data['Weight'] = 1

# Determine the mean and standard deviation of the number of followers a given user has for a given tweet
data['Followers_Mean'] = data['Followers'].rolling(10000, min_periods=1).mean()
data['Followers_Std'] = data['Followers'].rolling(10000, min_periods=1).std()
data['Followers_Std'] = data['Followers_Std'].fillna(data['Followers_Std'].values[1])

# Determine the mean and standard deviation of the number of re-tweets a given user has for a given tweet
data['RTs_Mean'] = data['RTs'].rolling(10000, min_periods=1).mean()
data['RTs_Std'] = data['RTs'].rolling(10000, min_periods=1).std()
data['RTs_Std'] = data['RTs_Std'].fillna(data['RTs_Std'].values[1])
```

Figure 63. Determining Follower's and Retweet's Mean and Standard Deviation

Figure 57 shows the computation to determine the mean and standard deviation of the number of followers and retweets a given user has for a given tweet to ensure that tweets with less tweets per day are treated equally. To also ensure that tweets from users with higher follower counts and tweets with more retweets carries more weight.

Processed tweet data is then aggregated monthly and daily, calculating the weighted sentiment and tweet volume. A rolling average of sentiment scores and tweet volume is also computed for smoother data representation.

Next, historical stock price data is fetched using a finance library for the same date range as the aggregated tweet data. Adjusted closing prices and percentage changes in stock prices are determined. Aggregated tweet data is combined with historical price data based on dates, creating a dataset that includes sentiment scores, tweet volume, and stock price information.

Percentage changes in stock prices are categorized into three groups: loss (0), small gain (1), and significant gain (2). This categorical representation simplifies the analysis problem and enables the development of investment strategies based on anticipated stock price movements. Unnecessary columns are removed, and rows with missing values (e.g., weekends without pricing information) are dropped to create a clean dataset for further investigation.

The final dataset contains stock ticker symbols, dates, weighted sentiment scores, rolling averages of sentiment scores, total tweet counts, rolling averages of tweet volumes, adjusted closing prices, percentage changes in stock prices, and categorical representations of percentage changes.

This combined stock input can be used to evaluate the investment strategies based on performance of return or loss.

```
[**********100%**********]  1 of 1 completed
Adbe - Completed
[**********100%**********]  1 of 1 completed
Azn - Completed
[**********100%**********]  1 of 1 completed
Bp - Completed
[**********100%**********]  1 of 1 completed
Ebay - Completed
[**********100%**********]  1 of 1 completed
Fdx - Completed
[**********100%**********]  1 of 1 completed
Hpq - Completed
[**********100%**********]  1 of 1 completed
Orcl - Completed
[**********100%**********]  1 of 1 completed
Pep - Completed
[**********100%**********]  1 of 1 completed
Rio - Completed
[**********100%**********]  1 of 1 completed
Tm - Completed
```

Figure 64. Combined Stock Input

	Ticker	Date	Sentiment_Weighted	Sentiment_MA	Tweets	Tweets_MA	Adj Close	Percent_Change	Percent_Change_B
0	Adbe	2022-01-03	2.187500	2.125000	16	16.000000	564.369995	-1.837446	
1	Adbe	2022-01-04	1.766462	2.012821	52	28.000000	554.000000	-7.142601	
2	Adbe	2022-01-05	1.869159	1.948373	107	58.333333	514.429993	-0.060260	
3	Adbe	2022-01-06	2.044715	1.900779	246	135.000000	514.119995	-0.665211	
4	Adbe	2022-01-07	1.558559	1.824144	111	154.666667	510.700012	2.962601	
...	...	...	...	...	...	...	...	...	
245	Tm	2022-12-22	1.674419	1.743325	43	45.333333	137.000000	0.102189	
246	Tm	2022-12-23	1.769231	1.666402	52	49.666667	137.139999	-0.714595	
247	Tm	2022-12-27	1.419355	1.601906	31	21.666667	136.160004	-1.490892	
248	Tm	2022-12-28	1.703704	1.586474	81	44.666667	134.130005	2.445388	
249	Tm	2022-12-29	1.492754	1.538604	69	60.333333	137.410004	-0.604033	

2500 rows × 9 columns

Figure 65. DataFrame of the Combined Stock Input

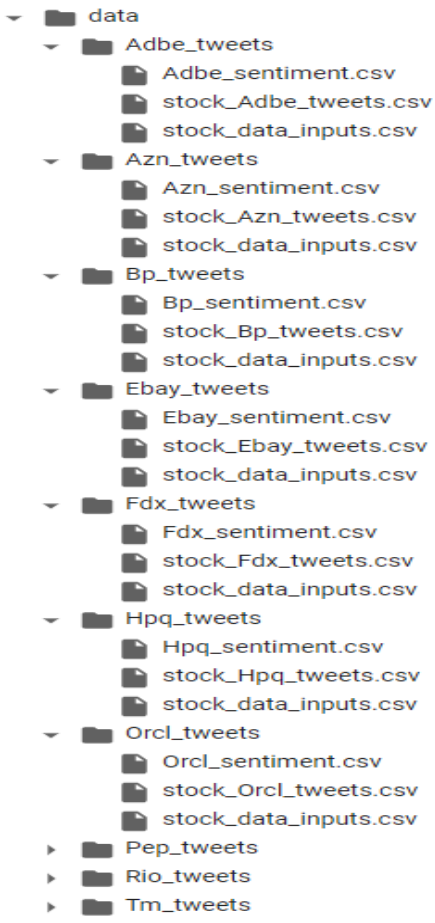

Figure 66. showing the dataset, stock sentiment and the combined stock input for all ten stocks.

9.3.Investment Strategies

Investment strategies are essential in the sense that they provide a roadmap for investors to follow as they navigate the financial markets. An investor can take different approaches to manage their portfolio, aiming to achieve their financial goals and maximize returns while considering their risk tolerance [39] . A well-defined investment strategy helps investors make informed decisions, minimize risks, and align their investments with their financial goals and risk appetite.

9.3.1. Long-term Investment

An investor allocates equal parts of their capital to all the available stocks in the dataset for each date in the test set, regardless of their predicted performance. This approach assumes that, over time, the overall market will grow, and eventual gains will compensate for any temporary losses. It is a passive investment strategy that relies on holding the stocks for extended periods without actively managing the portfolio. The system can be considered a combination of value investing, [40] growth investing [41], and income investing [42]. By investing equal parts in all ten stocks and holding onto them for an extended period, the investor aims to benefit from their potential growth, value appreciation, and dividend income. The long-term investment strategy is focused on capital appreciation and stable returns over time rather than attempting to time the market or capitalize on short-term fluctuations [43].

9.3.2. Bot Trading Investment

This strategy utilizes the predictions generated by the Random Forest and XGBoost Classifier models trained on the sentiment data. The investor only allocates their capital to stocks with a predicted positive price change (small gain or significant gain) for each date in the test set. This approach maximizes returns by actively managing the portfolio and selecting the stocks most likely to generate profits based on the model's predictions. This strategy can be classified as momentum investing [44], using sentiment data derived from Twitter and historical price data to make investment decisions. The bot identifies stocks with positive sentiment and momentum and invests in them, expecting the positive trend to continue. This strategy exploits short-term market trends and fluctuations to generate returns [45], making it a more active approach than the long-term investment strategy.

For each unique date in the test dataset, both investment strategies allocate their current capital to the stocks as follows:

In the long-term investment strategy, the capital is divided equally among all available stocks on that date, regardless of their predicted performance. The profit from investing in each stock is calculated as the allocated capital multiplied by the actual percentage change in the stock price. The total gain for that date is the sum of the profits from all stocks.

In the bot trading strategy, the capital is divided equally among the stocks with only a predicted positive price change. The profit calculation for each stock and the total profit for that date is the same as in the long-term investment strategy.

The capital for each strategy is updated based on the calculated profits, and the account balances for both approaches are plotted over time. By comparing the final returns and percentage returns of both investment strategies, the effectiveness of the sentiment-based prediction model in making profitable investment decisions can be assessed.

9.4. Effectiveness of the Investment Strategies

Two classification models Random Forest and XGBoost are to simulate two investment strategies, A long-term investment approach and a "Twitter bot" approach that invests based on the model's predictions. The initial capital for both strategies is set at £10,000.

9.4.1. Random Forest Classifier

For the first model which is the Random Forest Classifier, the model's performance in predicting losing stocks resulted in an accuracy of about 8.57%, indicating that around 8.57% of the predicted positive return stocks ended up being losing stocks.

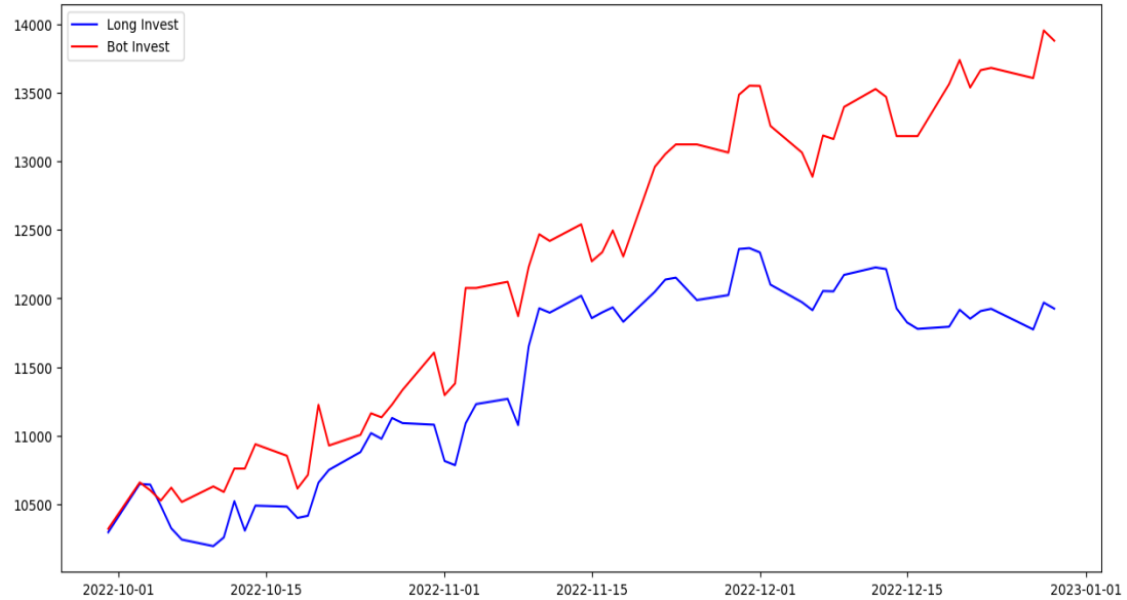

Figure 67. Investment Strategies using Random Forest Classifier

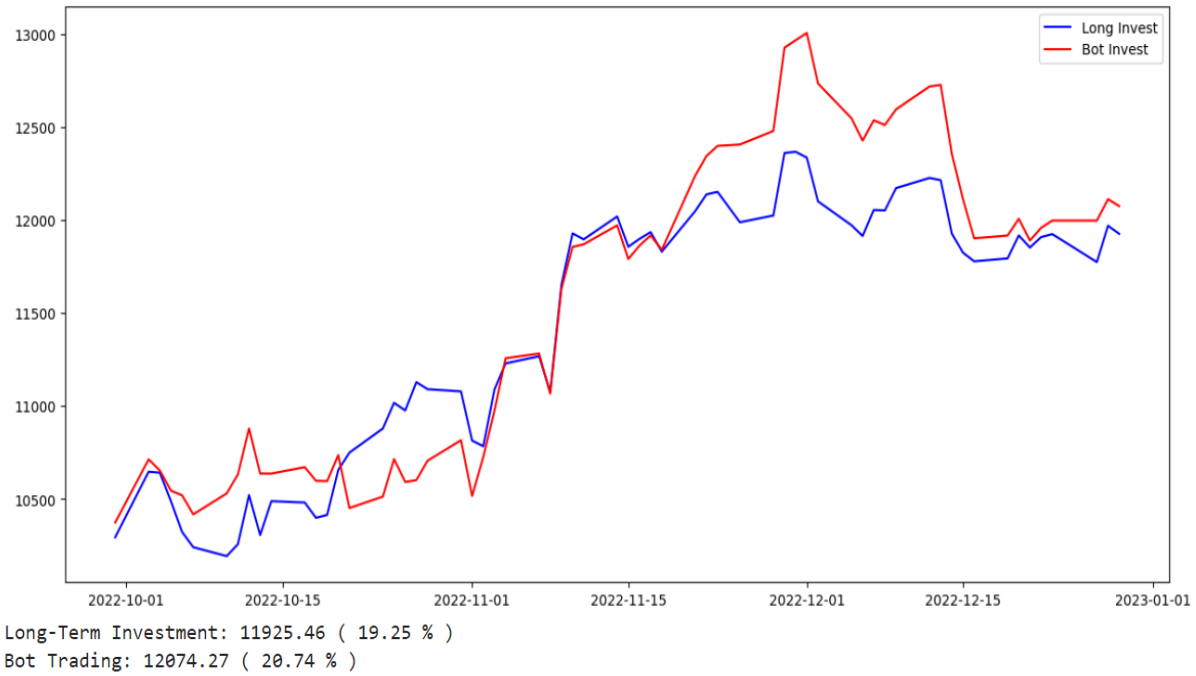

Figure 68. Investment Strategies using XGBoost Classifier

When comparing the simulated investment strategies, the long-term investment approach, which invested equal parts in all stocks for each timeframe and the Twitter bot approach that used the model's predictions to select stocks outperformed the long-term investment approach. The long-term investment strategy yielded a 19.25% return which indicates that an initial capital of £10,000 would have grown to £11,925.46 over the testing period, whereas the Twitter bot strategy provided an 38.79% return, with the initial capital of £10,000 growing to £13,878.8. As a result, the twitter bot approach carries a lower level of risk which rely on the predictive model.

9.4.2. XGBoost Classifier

For the second model, the performance in predicting losing stocks resulted in an accuracy of about 11.90%, indicating that around 11.90% of the predicted positive return stocks ended up being losing stocks.

When comparing the simulated investment strategies, the Twitter bot strategy slightly outperformed the long-term investment approach. The long-term investment strategy yielded a 19.25% return which indicates that an initial capital of £10,000 would have grown to £11,925.46 over the testing period, whereas the Twitter bot strategy provided an 20.74% return, with the initial capital of £10,000 growing to £12,074.27.

10. Conclusion

10.1. Summary of Findings

In this paper, we assess the effectiveness of using sentiment analyses from social media data, specifically Twitter, combined with historical price data to predict the stock price movement.

The key findings are summarised in this section, clearly indicating the contribution to empirical literature. Three algorithms were tested, including a traditional classifier (Support Vector Machine), a deep learning algorithm (Recurrent Neural Network), and a transformer algorithm (BERT). Among these, the best performing algorithm is the BERT, with an accuracy of 93.21%. The superiority of the advanced machine learning techniques in this study seems to agree with the empirical findings in [46,47] and [48] in which sophisticated machine learning and deep learning techniques are evidently superior to the conventional techniques in financial decision making.

The model performed slightly better on the positive and negative sentiment classes than on the neutral sentiment class. This suggests that the model may be better at identifying positive and negative sentiment than neutral sentiment. It is worth noting that the model had a high precision and recall for all three sentiment classes, which indicates that the model was able to accurately identify tweets of each sentiment class. Also, the model had a low number of false positives and false negatives, which implies that it is more accurate in predicting sentiment.

This paper also simulated two investment strategies: long-term investment and bot trading. The long-term investment strategy involved investing equal parts in all ten stocks under consideration. The bot trading strategy relied on sentiment-based predictions to determine which stocks to invest in.

The first result showed that the Twitter bot approach that used the random forest model predictions to select stocks outperformed the long-term investment approach, which invested equal parts in all stocks for each timeframe, while the second result also showed that the Twitter bot strategy slightly outperformed the long-term investment strategy, make the twitter bot investment strategy more less risk on overall evaluation.

These findings have several implications for investors and financial institutions. It demonstrates that sentiment data from social media platforms like Twitter can provide valuable insights when combined with historical price data in predicting the stock movement whether there will be a return on investment or not. Investors and financial institutions can leverage this information to develop more informed investment strategies with the use of machine learning algorithms.

In conclusion, this paper has shown that sentiment-based predictions from social media and historical price data can be valuable tools for investors and financial institutions to make informed decisions about their investment strategies.

10.2. Recommendations for Future Work

Investors and financial institutions can utilize sentiment analysis on social media data alongside traditional financial metrics for well-informed investment strategies. Real-time data from platforms like Twitter offers a dynamic view of market trends and investor sentiment, enabling swift responses to market changes.

Inclusion of sentiment data in investment strategies diversifies research and reduces reliance on conventional financial indicators, potentially reducing market volatility risks and improving portfolio performance. Consideration of macroeconomic factors, political events, and sector-specific news can further enhance stock price prediction models.

A three-fold approach encompassing fundamental analysis, technical analysis, and sentiment analysis can provide a comprehensive market understanding, leading to robust predictions. The deployment of sophisticated AI/ML algorithms can further refine sentiment-based stock predictions.

Customizing sentiment analysis models to align with specific investment objectives can enhance the relevance and accuracy of predictions. Emphasis on data quality and robust preprocessing techniques can mitigate noise and irrelevant information, boosting model performance. Continuous model refinement and updates with new data ensure accurate and timely sentiment predictions. Extending the analysis timeframe, covering diverse stocks, and incorporating novel algorithms can bolster the model's applicability and performance across varied market conditions.

References

[1] W. Chen, T. Chong and X. Duan, "The impact of sentiment on stock market predictability: evidence from the US and China.," *Applied Economics*, vol. 49(45), pp. 4586-4598, 2017.

[2] E. Guresen, G. Kayakutlu and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38(8), pp. 10389-10397, 2011.

[3] J. Bollen, H. Mao and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, vol. 2(1), pp. 1-8, 2011.

[4] G. Ranco, D. Aleksovski, G. Caldarelli, M. Grčar and I. Mozetic, "The effects of Twitter sentiment on stock price returns," *PLoS ONE*, vol. 10(9), p. e0138441, 2015.

[5] X. Zhang, H. Fuehres and P. A. Gloor, "Predicting stock market indicators through Twitter "I hope it is not as bad as I fear"," in *Social and Behavioral Sciences*, 2011.

[6] T. H. Nguyen, K. Shirai and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42(24), pp. 9603-9611, 2015.

[7] T. Loughran and B. McDonald, "Textual analysis in accounting and finance," *A survey. Journal of Accounting Research*, vol. 54(4), pp. 1187-1230, 2016.

[8] B. G. Malkiel, "The efficient market hypothesis and its critics," *Journal of Economic Perspectives*, vol. 17(1), pp. 59-82, 2003.

[9] M. Ghiassi, J. Skinner and D. Zimbra, "Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network," *Expert Systems with Application*, vol. 40(16), pp. 6266-6282, 2013.

[10] C. Krauss, X. A. Do and N. Huck, "Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500.," *European Journal of Operational Research*, vol. 259(2), pp. 689-702, 2017.

[11] F. Wang, M. Li, W. Li, X. Jia and G. Rui, "Sentiment Analysis of StockTwits Using Transformer Models," in *20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Pasadena, CA, USA, 2021.

[12] S. Yıldırım, D. Jothimani, C. Kavaklioğlu and A. Basar, ""Deep learning approaches for sentiment analysis on financial microblog dataset"," in *IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, 2019.

[13] M. Li, L. Chen, J. Zhao and Q. Li, ""Sentiment analysis of chinese stock reviews based on bert model"," *Applied Intelligence*, pp. pp. 1-9, 2021.

[14] B. Aysun, A. Sabrina, C. Mucahit and B. Ayse, "Sentiment Analysis of StockTwits Using Transformer Models," in *20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Pasadena, CA, USA, 2021.

[15] L. Xiaodong, X. Haoran, C. Li, W. Jianping and D. Xiaotie, "News impact on stock price return via sentiment analysis," *Knowledge-Based Systems*, vol. 69, pp. 14-23, 2014.

[16] K. Zhou, C. Zhang and S. Yang, "Fine-tuning BERT for text classification tasks in the financial domain," in *IEEE International Conference on Big Data (Big Data) (pp. 2895-2902)*, 2019.

[17] S. Jianfeng, M. Arjun, L. Bing and L. Qing , "Exploiting topic based twitter sentiment for stock prediction," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL), 24-29*, 2013.

[18] Y. Kara, M. A. Boyacioglu and O. K. Baykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange," *Expert Systems with Applications*, vol. 38(5), pp. 5311-5319., 2011.

[19] I. Kumar, K. Dogra, C. Utreja and P. Yadav, ""A Comparative Study of Supervised Machine Learning Algorithms for Stock Market Trend Prediction"," in *Second International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 100*, 2018.

[20] K. S. Amit, K. M. Pradeep and V. Attar, ""Multiple Kernel Learning for stock price direction prediction"," in *International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), pp. 1-4*, 2104.

[21] Z. Nurulhuda and S. Ali , ""Sentiment analysis using Support Vector Machine"," in *International Conference on Computer Communications and Control Technology Proceedings, pp. 333-337*, 2014.

[22] N. Christina and T. Christos, ""A Methodology for Stock Movement Prediction Using Sentiment Analysis on Twitter and StockTwits Data"," in *Computer Networks and Social Media Conference (SEEDA-CECNSM)*, Greece, 2021.

[23] B. Rakhi and M. D. Sher, ""Integrating StockTwits with sentiment analysis for better prediction of stock price movement"," in *International Conference on Computing Mathematics and Engineering Technologies (iCoMET), pp. 1-5.*, 2018.

[24] W. Bao, J. Yue and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory.," *PLOS ONE*, vol. 12(7), p. e0180944., 2017.

[25] M. Kraus and S. Feuerriegel, "Decision support from financial disclosures with deep neural networks and transfer learning," *Decision Support Systems*, vol. 104, pp. 38-48., 2017.

[26] M. Shruti, C. Anubhav and C. K. Nagpal, "Stock Market Prediction by Incorporating News Sentiments Using Bert," in *Modern Approaches in Machine Learning & Cognitive Science: A Walkthrough*, SpringerLink, 2022, pp. 35-45.

[27] C.-C. Lee, Z. Gao and C.-L. Tsai, "BERT-Based Stock Market Sentiment Analysis," in *IEEE International Conference on Consumer Electronics* , Taoyuan, Taiwan, 2020.

[28] G. S. Matheus, S. Kenzo , d. S. R. Lucas, H. M. Pedro, R. F. Eraldo and T. M. Edson, "BERT for Stock Market Sentiment Analysis," in *IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, Portland, OR, USA, 2019.

[29] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273-297, 1995.

[30] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, 2016.

[31] V. Ashish , S. Noam, P. Niki, U. Jakob, J. Llion, N. G. Aidan, K. Łukasz and P. Illia, ""Attention is all you need", Advances in neural information processing systems," in *NeurIPS Proceedings*, 2017.

[32] J. Devlin, M. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.," in *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[33] S. Vijayarani and R. Janani, "Text mining: open source tokenization tools-an analysis," *Advanced Computing and Intelligence: An International Journal*, vol. 3, no. 1, p. 37–47, 2016.

[34] P. Fabian, V. Gael and G. Alexandre, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[35] C. D. Manning, P. Raghavan and H. Schütze, Introduction to information retrieval., Cambridge University Press, 2008.

[36] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval.," *Information Processing & Management*, vol. 25, no. 5, pp. 513-523, 1988.

[37] J. Ramos, "Using tf-idf to determine word relevance in document queries," 2003.

[38] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation.," Doha, Qatar, 2014.

[39] Z. Bodie, A. Kane and A. J. Marcus, Investments, New York: NY: McGraw-Hill Education., 2017.

[40] B. Graham and D. L. Dodd, Security Analysis, New York: NY: Whittlesey House., 1934.

[41] P. A. Fisher, Common Stocks and Uncommon Profits, New York: NY: Harper & Brothers., 1958.

[42] M. H. Miller and F. Modigliani, "Dividend Policy, Growth, and the Valuation of Shares," *The Journal of Business*, vol. 34, no. 4, pp. 411-433, 1961.

[43] J. J. Siegel, Stocks for the Long Run: The Definitive Guide to Financial Market Returns and Long-term Investment Strategies, New York: NY: McGraw-Hill, 1998.

[44] N. Jegadeesh and S. Titman, "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency," *The Journal of Finance*, vol. 48, no. 1, pp. 65-91., 1993.

[45] T. J. Moskowitz, Y. H. Ooi and L. H. Pedersen, "Time Series Momentum," *Journal of Financial Economics*, vol. 104, no. 2, pp. 228-250, 2012.

[46] J. Iworiso and S. Vrontos, "On the directional predictability of equity premium using machine learning techniques," Journal of Forecasting, vol. 39, no. 3, pp. 449-469, 2020.

[47] J. Iworiso and S. Vrontos, "On the predictability of the equity premium using deep learning techniques," The Journal of Financial Data Science, 2020.

[48] J. Iworiso, "Forecasting stock market out-of-sample with regularised regression training techniques," International Journal of Econometrics and Financial Management, vol. 11, no. 1, pp. 1-12, 2023.