SCHOOL OF BUILT ENVIRONMENT, ENGINEERING AND COMPUTING

LEEDS BECKETT UNIVERSITY

Evaluating the Role of Financial News and Social Media Sentiment in Enhancing Technical Indicator-based Stock Price Movement Prediction with XGBoost

By: Terrence Josiah

Supervisor Name: Duncan Mullier

Submitted to Leeds Beckett University in partial fulfilment of the requirements for the degree of MSc Data Science

January 2025

Candidate's Declaration

I, Terrence Josiah, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

I have read and understand the penalties associated with Academic Misconduct.

Signed:

Terrence Josiah

Date: 03/04/2025

Student ID No: 77471560

1

[1] *The author confirms that no AI chatbot was used in the preparation of this dissertation, except for extracting news data from 2015 to 2020 as part of the data collection process. This use is described and justified in Section 3.2.3.*

Acknowledgements

The successful completion of this dissertation would not have been possible without the invaluable assistance of numerous individuals, whose names may not all be listed here. I am sincerely grateful for their contributions, which have been irreplaceable throughout this journey.

Furthermore, I would like to express my profound gratitude to Dr. Duncan Mullier for his continuous support, patience, and understanding throughout the development of this dissertation. His guidance, advice, and encouragement have been essential in shaping the outcome of this work.

To all my relatives, friends, mentors, and others who have offered their support—whether morally, financially, or physically—your contributions have been deeply appreciated. You have made a significant difference in this endeavour, and I am truly thankful for being able to share this memorable journey with you.

Abstract

Stock market prediction has traditionally relied on historical price data and technical indicators such as moving averages, relative strength index (RSI), and volatility. However, market movements are also influenced by external factors such as financial news and investor sentiment. This study explores the impact of incorporating financial news and social media sentiment into an XGBoost-based stock price movement prediction model. Specifically, it evaluates whether these additional features improve predictive performance compared to a model that solely relies on historical stock price data.

To achieve this, historical stock prices are collected using **Yahoo Finance API** alongside the selected stock's related financial news articles that will be web-scraped using **DeepSeek** and tweets gained from a part of the paper published in the 2020 IEEE International Conference on Big Data under the 6th Special Session on Intelligent Data Mining track, which is publicly available on Kaggle. Sentiment analysis is performed using **FinBERT**, a finance-specific natural language processing (NLP) model, to extract vector representations and sentiment scores from news articles and tweets. The extracted sentiment features are then integrated into the existing feature set alongside traditional technical indicators. The XGBoost classifier is trained on two datasets: one with only technical indicators and another with additional sentiment-based features.

The performance of both models is compared using accuracy, precision, recall, F1-score, and AUC-ROC score to determine the contribution of sentiment data in improving prediction accuracy. The findings of this study provide insights into the role of news and social media sentiment in stock market forecasting, offering potential improvements for algorithmic trading and financial decision-making.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full Term
ADX	Average Directional Index
AI	Artificial Intelligence
API	Application Programming Interface
ARCH	Autoregressive Conditionally Heteroscedastic
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
BERT	Bidirectional Encoder Representations from Transformers
BloombergGPT	Large language model trained on financial data by Bloomberg
CART	Classification and Regression Tree
CCI	Commodity Channel Index
DL	Deep Learning
FiLM	Financial Language Model
FinBERT	BERT model fine-tuned for financial sentiment analysis
GBDT	Gradient Boosting Decision Tree
GPT	Generative Pre-trained Transformer
KPCA	Kernel PCA
LLM	Large Language Model
LW	Larry Williams

MACD	Moving Average Convergence Divergence
ML	Machine Learning
MSE	Mean Squared Error
NLP	Natural Language Processing
OBV	On-Balance Volume
PCA	Principal Component Analysis
PCR	Principal Component Regression
P/E Ratio	Price to Earnings Ratio
RoBERTa	Robustly Optimized BERT Pretraining Approach
RoBERTa-Fin	RoBERTa model fine-tuned on financial text data
RF	Random Forest
RNN	Recurrent Neural Networks
RSI	Relative Strength Index
SOTA	State-Of-The-Art
SVM	Support Vector Machine
TA	Technical Analysis
TPU	Tensor Processing Unit
t-SNE	t-distributed stochastic neighbour embedding
UMAP	Uniform Manifold Approximation and Projection
VARMA	Vector Autoregressive Moving Average
XGBoost	eXtreme Gradient Boosting
yfinance	Python API for accessing Yahoo Finance data

Chapter 1: Introduction

1.1. Background

The stock market is a collection of exchanges where investors can buy and sell the shares of publicly listed companies (Billah et al., 2024). It is one of the most important financial marketplaces in a laissez-faire economic system, an economic system with minimal government intervention, which joins people and companies with the flow of money. The stock market has always been incredibly attractive to both investors who seeks profitable returns and companies needing capital to fund their business processes due to its easy accessibility and relatively high profitability. As a result, the stock market experienced significant expansion and development, especially since the popularisation of personal computers and the rapid emergence of electronic trading platforms in the late 20th century (Yun, Yoon and Won, 2021). Unlike the traditional trading systems, which offered limited access to real-time information and requires investors to rely on manual order methods, The development of electronic trading platforms and computing technology has made stock transactions far more efficient by providing access to massive amount of trading data and information.

Although stock investment presents the opportunities for high returns, it also comes with considerable risks. Thus, it is crucial to develop an effective stock price movement prediction method for both investors and companies to be able to take investment opportunities and manage trading risks (Ma et al., 2023). Many data scientists and researchers who are interested in solving the underlying mechanism of stock price movement have been focusing on analysing and processing vast amount of stock data. However, developing a highly accurate system or function to predict the stock price movement is incredibly difficult, as stock prices are volatile, complex, nonlinear, noisy, and sometimes does not make any sense to be predicted (Basak et al., 2019; Chen and Hao, 2017; Chung and Shin, 2018; Henrique, Sobreiro and Kimura, 2019; Wang, Xu and Zheng, 2018). Stock prices are influenced by many external factors, including political, economic, and psychological aspects, which are often reflected by sentiment-based data like financial news and social media (Li et al., 2016). Nevertheless, numerous studies have been conducted on stock prices forecasting by

researchers from different academic backgrounds and various types of investors (Chen and Hao, 2017; Chung and Shin, 2018; Yu and Yan, 2019).

As stated by Shah, Isah and Zulkernine (2019), there are two main traditional approaches in analysing and predicting the stock prices: fundamental analysis and technical analysis. Fundamental analysis helps to assess a company's true value based on factors such as economic, industry and company-specificity. Fundamental analysis includes evaluating the overall economy that may affect a company's future profits (e.g., GDP, CPI), considering the company's financial conditions and growth prospects in the industry, and examining internal factors (e.g., operations and financial health) to estimate intrinsic value. Some valuation techniques to assess a company's value includes P/E Ratio to compare with other stocks with similar growth (Imam et al., 2008), Gordon's Growth Model (Gordon and Shapiro, 1956; Gordon, 1959), which is a dividend discount model, and Financial Ratios, which was used by Dutta, Bandopadhyay and Sengupta (2012) to separate good stocks from the poor ones. Technical analysis involves evaluating and deriving indicators from historical price and volume data to predict future stock price movements. This analysis includes several domains, including analysing investors' strength in anticipating buying or selling power, making use of visual tools like candlestick (K-line) and bar charts, evaluating price trends, cycle, and momentum, and assessing risk and price fluctuations to identify support and resistance levels (Hu et al., 2015). In short, fundamental analysis leverages financial methods to predict future price by determining whether actual stock price is above or below the intrinsic value (Murphy, 1999; Nti, Adekoya and Weyori, 2019), whereas technical analysis leverages statistical methods to predict future price by identifying historical price and volume patterns (Lin, 2011).

There are many techniques that have been developed for studying and predicting stock price movement. From the first Efficient Market Hypothesis (Fama, 1970), generating temporary price components using expected returns (Fama and French, 1988), considering multiple macroeconomic factors (Pesaran and Timmermann, 1995), the development of Box and Jenkins transformations (Box and Jenkins, 2015), linear regression, ARMA model (Hamilton, 1994), and ARCH model (Engle and Mustafa, 1992), all of these popular techniques were proposed and developed to predict future stock price movements. However, these traditional statistical and time series models cannot capture the complexity, nonlinearity, and volatility of the stock

price data (Yun, Yoon and Won, 2021). With the development of Artificial Intelligence (AI) throughout recent decades, many studies have used machine learning (ML) techniques in predicting future stock price movements. ML is a branch of AI that allows computer systems to keep improving their performance on specific tasks by learning the patterns of past data (Chollet, 2021). AdaBoost, support vector machines (SVM), decision trees, k-nearest neighbour (KNN), and many more are some of the examples of ML algorithms (Henrique, Sobreiro and Kimura, 2019; Shah, Isah and Zulkernine, 2019; Strader et al., 2020). eXtreme Gradient Bosting (XGBoost) is also one of the most powerful and widely used in predicting stock prices (Han, Kim and Enke, 2023). These ML algorithms can outperform the traditional statistical model in processing sequential data, including time series, as they do not require the input data to be in a specific format nor following several assumptions such as linearity, stationarity, homoscedasticity, and normality (Chen and Hao, 2017; Chung and Shin, 2018; Wang et al., 2018; Basak et al., 2019; Henrique, Sobreiro and Kimura, 2019).

In the case of automating stock price movement prediction using ML techniques, technical analysis is the one commonly being used as the predictors instead of fundamental analysis due to several reasons. Firstly, the main components of fundamental analysis are heavily derived from qualitative measurements such as management quality, industry outlook, and new business model, which are difficult to quantify and used as variables in data-driven ML models (Murphy, 1999). Secondly, fundamental data such as balance sheets, income statements, cash flow statements, and financial ratios are usually reported on a monthly, quarterly, or annual basis, resulting in the failure upon capturing short-term fluctuations in stock prices, which often changes unpredictably in response to recent events (Yun, Yoon and Won, 2021). Finally, fundamental data often becomes subject to both intentional and non-intentional human bias, errors, or manipulation, which reduces its reliability for short-term predictive tasks (Chen and Hao, 2017; Song, Lee and Lee, 2019). The components of technical analysis, on the other hand, are derived solely from historical price and volume, which are prone to manipulations and have more controllable time frames, therefore become more reliable for predictive tasks.

Despite capturing a lot of insights regarding future stock price movements, technical indicators are still prone to unexpected events that may lead to sudden stock price movements, notably mergers, acquisitions, profit variation, upgrade or downgrade of

financial statements, and changes of corporate systems (Lien Minh et al., 2018). This crucial information is usually reflected in financial news articles posted by providers like Bloomberg or Reuters. Many stock price forecast systems have been reported to function well by putting their base above financial news articles (Matsubara, Akita and Uehara, 2018).

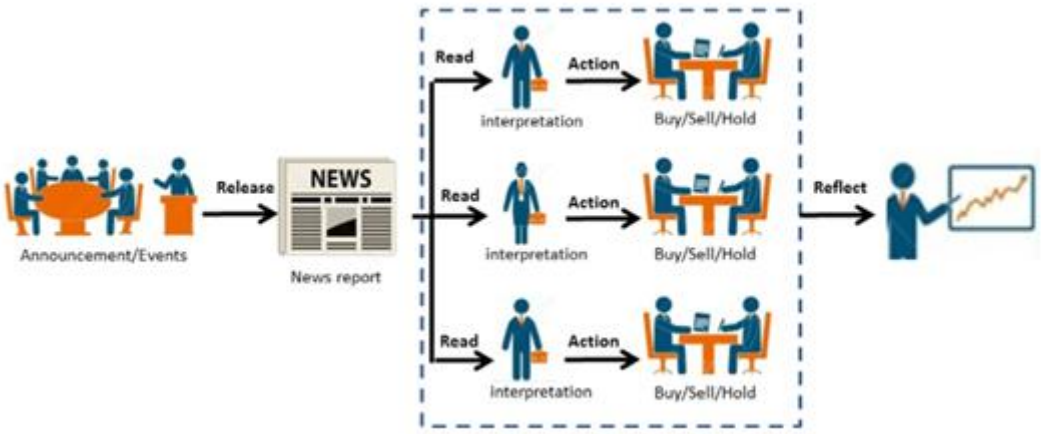


Figure 1. Overview of News-Based Stock Price Decision-Making Process (Source: Hao et al., 2021)

Additionally, Ian Rosen, the CEO of Stockwits Inc., highlights that social media platforms enable investors to communicate, share insights, and thus adding value to capital markets, including the stock market (Hossain, Mammadov and Vakilzadeh, 2021). One of the most popular social media platforms for users to express their thoughts is Twitter. Tweets are proven to give valuable insights on people's sentiment towards certain stocks or companies (Batabyal et al., 2023; Chaudhuri et al., 2018; Kumbure et al., 2022). It is common for analysts to combine public sentiment with historical price data in predicting future stock prices as it has been proven that public opinion has considerable impact towards stock price movements (Raman et al., 2022).

Text data must be converted into numerical representations before being fitted into any ML models. In recent years, the development of some domain-specific language models such as FinBERT have achieved strong performance in financial sentiment and natural language processing (NLP) tasks (Araci, 2019), thus forming the foundation for modern state-of-the-art (SOTA) sentiment-driven prediction systems.

1.2. Rationale

Despite the extensive research and technological advancements in stock price prediction, it is still incredibly difficult to create a highly accurate and reliable model due to the complex and volatile nature of financial markets. While technical indicators provide a strong quantitative foundation, they often fail in capturing real-time sentiment events such as the shift in public opinions as well as some crucial corporate announcements.

With the recent developments in both ML and NLP, particularly with models like XGBoost and FinBERT, it offers opportunity to extract values from sentiment-based data such as financial news articles and social media platforms to improve the predictive model's performances. These data often reflect market information that cannot be seen from technical analysis alone. However, there are lack of unified frameworks that effectively combine both technical indicators with sentiment-derived features to predict stock price movement more accurately.

This study aims to address the gap by developing a classification model that integrates technical indicators with sentiment features extracted from financial news articles published by various sources from the internet and Twitter, one of the most widely known social media platforms by using FinBERT. FinBERT is being used due to its capability in outperforming SOTA ML models in completing financial sentiment analysis tasks. XGBoost classifier is being used due to its proven effectiveness in handling complex data as compared to many other conventional ML models.

By exploring this hybrid approach, this study seeks to improve stock price movement prediction performances and demonstrate the value of integrating as well as effectively engineer diverse data sources in financial forecasting, contributing to both academic literature and real-world trading insights.

1.3. Research Questions

1. Does incorporating financial news and social media sentiment data improve the predictive performance of an XGBoost-based stock price movement classifier compared to using only historical stock price and technical indicators?

2. How do different sentiment sources (news vs tweets) compare in their contribution to predictive performances?
3. How does including sentiment-based features impact key model evaluation metrics (e.g., accuracy, precision, recall, F1-score, AUC-ROC score)?
4. What feature-engineering techniques can be applied to financial news and social-media sentiment features to maximize their contribution to XGBoost stock movement prediction, particularly when raw sentiment features alone fail to improve model performance?
5. Which features contribute the most towards the XGBoost's predictive capabilities?
6. What are the limitations and challenges of integrating sentiment analysis into stock price prediction, and how can they be addressed?

1.4. Research Aim

The aim of this dissertation is to develop and evaluate a sentiment-enhanced stock price movement prediction model by integrating financial news and social media sentiment into an XGBoost classifier. This research seeks to determine whether sentiment-based features improve the performance of stock movement predictions compared to models that rely solely on historical stock price indicators, and to identify the most effective feature engineering techniques to maximize the value of these sentiment-based features. By leveraging FinBERT for financial news and social media sentiment feature extraction and applying various data transformation methods to these predictors, this study will assess the effectiveness of text-based features in improving the predictive performances of the XGBoost classifier model. Ultimately, this research aims to provide insights into how sentiment-driven models can contribute to a more informed investment decision-making.

1.5. Research Objectives

1. To gather historical stock price data, compute relevant technical indicators (TIs), and collect corresponding financial news articles and twitter data.

2. To analyse the whole data, understand the nature of the data, and apply proper transformation to the data.
3. To apply FinBERT to transform news articles and tweets into numerical representations.
4. To compare the performance of the baseline model (only TIs) and the enhanced model (with additional sentiment features) based on the model performances (accuracy, precision, recall, F1-score, AUC-ROC).
5. To apply and evaluate targeted preprocessing (e.g. standardization and dimensionality reduction of the FinBERT-extracted features) when raw sentiment features fail to improve model performances.
6. To analyse and gain insights via SHAP values on which features contribute the most towards XGBoost's predictive performances.
7. To examine practical challenges and propose ways to address them for real-world deployment.

1.6. Outline

This dissertation is organised as follows:

Chapter 1: Introduction introduces the background, rationale, research questions, aim, objectives, and outline of the study.

Chapter 2: Literature Review reviews the past existing studies and research on the topic of stock price predictions using ML and NLP.

Chapter 3: Methodology provides details for the research designs, data collection methods, and technical frameworks used for developing and evaluating XGBoost model in stock price movement prediction.

Chapter 4: Product Design and Implementation explains detailed technical development process, including data preprocessing, calculations, used models, model parameters, and system integration.

Chapter 5: Research Discussion and Evaluation presents the evaluation results, which are the impact of adding sentiment-based features and the application of various data transformation methods to XGBoost model's performance metrics.

Chapter 6: Project Management provides an overview of the timeline of each activity done for the dissertation through a Gantt Chart. This covers the initiation to the end of the implementation.

Chapter 7: Conclusion and Future Work discusses briefly about the result and future actions that can be implemented to develop the project.

Chapter 2: Literature Review

2.1. Introduction

This chapter reviews literatures and past studies in stock price predictions using machine learning (ML) and natural language processing (NLP) methods. This chapter will review the traditional approaches to stock price prediction, followed by machine learning approaches to stock price prediction, sentiment analysis in stock price prediction, advances in NLP, hybrid models, limitations in existing research, and lastly the summary and research gap justifications.

2.2. Traditional Approaches to Stock Price Prediction

2.2.1. Fundamental Analysis

Fundamental analysis is analysed by using a top-down approach with the purpose of projecting the actual value of a stock. As shown in Figure 2, the collection of information is conducted across three business layers, which are economic analysis, industry analysis, and finally, the company analysis (Eiamkanitchat, Moontuy and Ramingwong, 2017).

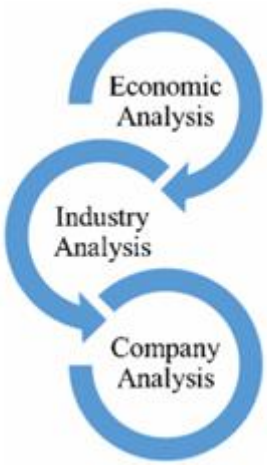


Figure 2. Top-Down Fundamental Analysis (Source: Eiamkanitchat, Moontuy and Ramingwong, 2017)

At the first stage, fundamental analysis involves the investigation of both global and national economy with a focus towards some primary aspects, including gross domestic product (GDP), industrial production, producer price index (PPI), inflation rate, interest rate, and employment rate, which indicate the effect of the macroeconomic environment towards the company's future profit and market attractiveness (Eiamkanitchat, Moontuy and Ramingwong, 2017; Shah, Isah and Zulkernine, 2019). The next stage of the analysis focuses on to the national industry level. This involves collecting information on the business sector's total sales, price levels, and competitor analysis used to locate the most attractive industry. Finally, the last stage of fundamental analysis evaluates a particular company's internal value within a certain industry by focusing on metrics such as revenues, earnings, profit margins, growth potential, company policies, and return on equity (Eiamkanitchat, Moontuy and Ramingwong, 2017).

Within the fundamental analysis exists different valuation approaches to measure the value of stocks. The average growth approximation is one of the most well-known techniques that compares different stocks within the same category to better understand the companies' value. The Price-to-Earnings (P/E) method is often being used in the stock brokerage industry. A company with lower P/E ratio is better than the ones with high P/E ratio, of course, under the assumption that the compared companies have similar growth rate (Shah, Isah and Zulkernine, 2019). Another popular constant growth approximation technique is the Gordon's growth model (Gordon and Shapiro, 1956; Gordon, 1959), which assumes that the dividends of a company, although less than the discount rate, will increase at a constant rate perpetually.

Some studies have suggested that fundamental analysis indeed provides value for future stock price predictions. At least, one of the earliest uses of fundamental analysis can be seen from Benjamin and Dodd (1934) paper, which proven its ability to provide robust information regarding future stock returns across companies from different time periods and countries. Many researchers such as Sloan (1996), Xie (2001), Fairfield, Whisenant and Yohn (2003), O'Shaughnessy (2011), Asness, Frazzini and Pedersen (2019), and Blitz and Vidojevic (2019) further acknowledged fundamental analysis' value by showcasing an approach specifically designed to demonstrate its predictive ability to detect abnormal future returns.

Despite its acknowledged value, fundamental analysis only works for long-term stock price prediction instead of short-term (Chen, Chen and Lu, 2017). Fundamental analysis relies on the analysis of financial statement, which are commonly published monthly, quarterly, or annually (Yun, Yoon and Won, 2021). This creates a time constraint to the point where fundamental analysis fails to capture short-term price fluctuations which are often caused by events within a short timeframe. Additionally, the use of fundamental analysis as a stock price predictor requires complex approaches and only work in specific market environment (Hanauer, Kononova and Rapp, 2022). For instance, Bartram and Grinblatt (2018, 2021) once suggested a new agnostic approach towards fundamental analysis. While this approach succeeded in reliably predict future returns in the United States of America (Bartram and Grinblatt, 2018) and some other regions around the world (Bartram and Grinblatt, 2021), it has been proven to fail in the case of predicting future returns of the European stock markets (Walkshäusl, 2021). Due to the complexity of fundamental analysis, some researchers have also decided to only use it for recommendation purposes instead of price predictor. For instance, Eiamkanitchat, Moontuy and Ramingwong, (2017) applied clustering and classification methods to 10 features derived from various companies' financial statements, filtering the stocks into three categories and determine the most interesting stocks, which will be recommended to the users and thus helping the users to make informed decisions.

Unlike technical analysis, which focuses on the patterns of price and market behaviour within a shorter timeframe, fundamental analysis provides a longer-term perspective based on the true value of a company.

2.2.2. Technical Analysis

In contrast to fundamental analysis, technical analysis, also known as 'Chartism', considers historical market data, primarily share price, to forecast how these prices will move in the future (Becket, 2021). For some dedicated chartist, it does not even matter whether a price chart is related or not related to stock, all relevant information is assumed to have movements pattern. The main thing that makes technical analysis completely different from fundamental analysis is that technical analysis totally ignores the companies' business worth. Technical analysis is not concerned with how the

company is being managed, but with how and when the market price is likely to change (Becket, 2021). Instead of telling which share to buy, technical analysis implies more towards which share prices due for a turn, in either direction, thus providing actionable insights for active traders (Becket, 2021).

Authors	Dataset	Output	Input Features	Feature Engineering	Feature Dimension	Prediction Methods	Evaluation Metrics
(Patel et al., 2015)	4 Stock prices & Indices	Price, Direction	Technical	Generation	(2474, 10)	ANN, RF, SVM, NB	ACC, F1
(Ballings et al., 2015)	European stock prices	Direction	Financial, Economic, Technical	Generation	(5767, 87)	RF, KF, AdaBoost,	AUC
(Qiu & Song, 2016)	Nikkei stock index	Direction	Technical	Generation	(1707, 13), (1707, 9)	GA + ANN	ACC
(Chen & Hao, 2017)	2 Chinese stock indices	Price	Historical, Technical	Generation, Selection	(1500, 13)	FW-SVM, FW-KNN	MAPE, RMSE
(Chung & Shin, 2018)	Korean stock price index	Price	Historical, Technical	Generation	(4203,10)	GA + LSTM	MSE, MAE, MAPE
(Song et al., 2019)	Korean stock price index	Price increase rate	Historical, Technical	Filtering, Generation	(12783,715)	DNN	ACC
(Wang et al., 2018)	Chinese stock index	Direction	Historical, Technical	Generation	(970, 62)	Deep Learning, EC	ACC, AUC, Recall, F1
(Basak et al., 2019)	10 US & Indian stocks	Direction	Historical, Technical	Generation	(10700, 8)	RF, XGBoost	ACC, AUC, Recall, F1
(Naik & Mohan, 2019)	Indian stock index	Price	Historical, Technical	Generation, Selection	(10 years, 22)	ANN	MAE, RMSE
(Ntakaris et al., 2019)	5 Nordic & 2 US stocks	Direction	Economic, Technical LOB features	Generation, Extraction	(13 Million events, 87)	MLP, CNN, LSTM	F1, RMSE
(Yu & Yan, 2020)	6 Stock indices	Price	Historical	Extension	(2518, m)	LSTM	DA, MRSE, MAPE, CORR
(Chung & Shin, 2020)	Korean stock price index	Price	Historical, Technical	Generation	(4203, 12)	GA + CNN	ACC
(Ampomah et al., 2020)	22 Stock prices	Direction	Historical, Technical	Generation, Extraction	(3774, 45)	EC	ACC, AUC, Recall, F1
(Ding & Qin, 2020)	2 Chinese stock prices	Price	Historical, Technical	Generation	(6112, 7)	LSTM + DRNN	MSE, MAE
(Nabipour et al., 2020a)	4 Iranian stock prices	Price	Technical	Generation	(10 years, 10)	Tree models, NN	MAPE, MAE, RMSE, MSE

Figure 3. Recent Stock Price Prediction Studies using Technical Indicators (Source: Yun, Yoon and Won, 2021)

There are several key evaluation metrics as components of technical analysis, namely technical indicators. As shown in Figure 3, numerous studies have indicated that technical indicators can serve as useful variables in stock price forecasting tasks (Thesia, Oza and Thakkar, 2021). For instance, Yun, Yoon and Won (2021) leverages vast amounts of technical indicators, mainly derived from historical price data, including momentum indicator, relative strength index (RSI), commodity channel index (CCI), Larry Williams (LW), and stochastic to predict future stock price movements. These indicators supported the model in achieving above 90% across all performance metrics, in which these indicators' impacts to the model prediction performances were all evidenced through the generated feature importance graph. Additionally, Baek and Lee (2024) stated that employing technical indicators improved the deep learning (DL)

model performance as compared to when only using basic price and volume indicators.

However, there are several challenges in maximising the value of technical indicators, and one of them is choosing the right set of technical indicators (Thesia, Oza and Thakkar, 2021). This have been proven by the difference in the number of technical indicators used in past studies to forecast future stock price. For instance, Yun, Yoon and Won (2021) used 7 technical indicators while Baek and Lee (2024) used 6 to obtain their respective optimal results. Note that each of them used different periods for each indicator, thus resulting in different number of generated features. Another evidence is when Das et al. (2024) used 5 technical indicators whereas Chandar (2024) used 10 technical indicators, with different time periods, to achieve their respective optimal results. Of course, these researchers were using different sets of data, timeframes, and ML or DL models. Different ML or DL models may raise challenges as well, where for instance, one model is subject to multicollinearity and overfitting, while the others are not (Masis, 2021). Therefore, these papers further imply the importance of extensive research to optimise the types and periods of technical indicators used for different cases and methods.

2.3. Machine Learning Approaches to Stock Price Prediction

2.3.1. Limitations of Statistical Models

In the realm of stock price forecasting, the development of statistical models remains one of the most groundbreaking revolutions as it reshapes how analysts understand and predict market movements. Various statistical models such as the Holt Winter's model, autoregressive (AR) model, autoregressive moving average (ARMA) model (Hamilton, 1994), autoregressive integrated moving average (ARIMA), as well as its variations, including autoregressive conditionally heteroscedastic (ARCH) (Engle and Mustafa, 1992) and generalized autoregressive conditional heteroskedasticity (GARCH) (Franses and Ghijsels, 1999) model were popular in solving stock price prediction tasks (Durgapal and Vimal, 2021; Yun, Yoon and Won, 2021; Zhou et al., 2019). Although these models were popular for laying the groundwork for quantitative forecasting, they exhibit some fundamental limitations when applied to modern, high-frequency financial markets.

A primary concern with classical statistical approaches is their reliance on very strict statistical assumptions, namely stationarity, linearity, and homoscedasticity (Lv et al., 2022; Zhou et al., 2019). ARMA and ARIMA models assume that some statistical properties in the data, such as mean and variance, remain constant overtime and among normally distributed variables (Durgapal and Vimal, 2021; Zhou et al., 2019). However, financial markets time series data are rarely stationary as they often exhibit changing volatility and structural breaks due to macroeconomic shocks or regulatory changes (Hagenau, Liebmann and Neumann, 2013; Zhou et al., 2019). As a result, it is highly likely that the predictions of these models will deteriorate rapidly once the market experiences unpredictable fluctuations.

Beyond stationarity, these models also suffer from linearity assumption, which assumes that the current values of the series is a linear function of past values plus a random noise term, thus restricting these models' capability to capture complex interactions between the changing returns within the nonlinear stock market data (Dudek et al., 2024; Lv et al., 2022). Although some improvised ARCH and GARCH models address time-varying volatility, they remain subject to linear dynamics (Francq and Sucarrat, 2023). Some evidence shows that the changes in stock returns and the relationship between different significant predictors are not linearly correlated (Kumar and Shrivastav, 2021), which the GARCH variants only partially accommodate. As a result, large number of unexpected events are poorly forecasted, leading to unreliable model performance during crisis or where there are jumps in volatility (Charles and Darné, 2019).

Another critical limitation is many of these models are only capable of having one predictor to predict stock returns, also known as univariate (Anderson, 1991). Standard ARIMA-type frameworks are only able use the historical stock price to predict future prices, thus neglecting the importance of many other extremely important variables, such as corporate news, macroeconomic indicators, investor sentiment, smoothed quantitative features, and relationship between other variables. This will not only increase the prediction errors but also reduces the models' ability to predict out-of-sample data. While some extension models such as vector autoregressive moving average (VARMA) and multivariate GARCH exist, they require heavy hyperparameter tuning, which may not generalise well across different time frames and stocks variants as different data characteristics require different specific model suitability (Düker,

2025; Li and Maheu, 2024). Additionally, these models also require large computational resources and costs (Düker, 2025; Li and Maheu, 2024).). Since various variables are needed to predict stock price movement accurately, the prediction process will be time consuming, limiting their utility in systems that require millisecond-level updates.

Taken together, these limitations underscore why pure statistical models, albeit foundational, are not sufficient to perform accurately in the environment of modern stock market. Their restrictive assumptions, inability to incorporate rich information, and some practical constraints motivate the development of a more flexible and reliable approaches, namely some ML and NLP approaches that are widely used in this modern times.

2.3.2. Rise of Machine Learning Models

The limitations of traditional statistical models, coupled with advancements in computational power and data availability, have driven the development of machine learning (ML) in the realm of stock price forecasting. ML models have been proven to be capable of capturing nonlinear patterns, efficiently process high dimensional data, and adapt to the volatility of dynamic market conditions, thus being superior alternatives to classical approaches (Henrique, Sobreiro and Kimura, 2019; Yu and Yan, 2019).

Unlike ARIMA or GARCH, ML models do not have some strict assumptions such as linearity and stationarity (Bhattacharjee and Bhattacharja, 2019). For instance, support vector machines (SVM) map data into higher dimensional spaces to capture nonlinear relationships by using its kernels such as polynomial or radial basis function (RBF) (Wu, Zhang and Chiu, 2023), while random forests (RF) and gradient boosting decision trees (GBDT) leverage multiple decision trees to model the complex relationships between existing variables, thus reducing bias and the chance of overfitting (Basak et al., 2019). ML models also excel at incorporating multiple, diverse predictors, such as technical indicators (e.g., RSI, MACD), macroeconomic data, and sentiment features (Wang, Xu and Zheng, 2018). This capability aligns with some evidence stating that stock returns are influenced by multiple mutual factors (Wang, Xu and Zheng, 2018). For instance, Hagenau, Liebmann and Neumann (2013) were

able to increase the stock price prediction accuracy of an SVM model to 76% by incorporating financial news. Additionally, Long, Song and Tian (2019) were able to prove that the semantic and structural relevance between news indeed positively contribute to stock price prediction using an SVM-based model. Finally, modern ML models like eXtreme Gradient Boosting (XGBoost) are also optimised to efficiently process high dimensional data (Basak et al., 2019), and thus able to provide actionable insights within a short period of time.

Some studies have proven modern ML models' predictive strength over the traditional statistical models. ML models consistently outperform statistical models in terms of model performance. For instance, Durgapal and Vimal (2021) stated that XGBoost is approximately 15.34% better than ARIMA in terms of root mean squared error (RMSE) and 87.2% better in terms of mean absolute percentage error (MAPE). Additionally, although not specifically in the case of stock price prediction, Schmid et al. (2024) suggested that ML models such as RF and XGBoost are more robust in predicting nonlinear and complex data, thus outperforming statistical approaches such as ARIMA in terms of MAPE and mean squared error (MSE) under multiple simulations.

Despite their strengths, ML models such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) often lack in terms of interpretability. In the case of stock investment, it is crucial to make a human-friendly interpretable model for the end users to understand the reasons for each prediction outcome (Carvalho, Pereira and Cardoso, 2019; Ghorbani et al., 2021). By discovering the reasons of the samples' learning outcome, investors and model developers can further study and design investment plans and strategies in real-world scenarios. However, some ML models' prediction results are very hard to understand and have a black-box nature, which confuse human investors even further (Yun, Yoon and Won, 2023). Thus, it is important to use some tools such as SHapley Additive Explanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) to enhance ML models' interpretability. Another weakness is that many ML models cannot directly process raw, unstructured textual data like financial news and tweets. These textual data must be transformed into numerical representations before being fitted into the before-mentioned ML models (Long, Song and Tian, 2019). In an environment where market sentiment and news are driving price movements, these shortcomings may lead to missed signals and reduced profits. This gap has led to growing interest in combining

ML models with natural language processing (NLP) models to enrich the existing data and improve ML models' predictive power.

2.3.3. eXtreme Gradient Boosting (XGBoost)

XGBoost has emerged as one of the most popular and powerful ML models. This is due to both its performance and efficiency in solving complex predictive tasks, which includes stock price forecasting. XGBoost is built upon the gradient boosting foundation, where it is an ensemble learning method that combines multiple weak learners, such as decision trees (DT), into a strong predictive model (Chen and Guestrin, 2016). Some methodologies such as the out-of-core computation, cache-aware, and sparsity-aware learning are some of the aspects that made XGBoost extremely scalable in wide-range of scenarios and faster than other existing popular solutions (Chen and Guestrin, 2016).

One of the key reasons why XGBoost has sparked interests of researchers in solving stock price prediction tasks is because XGBoost does not require any extensive feature engineering and hyperparameter tuning, meaning that the model can perform generally and consistently well in almost every possible scenario. For instance, Han, Kim and Enke (2023), states that XGBoost, with a prefixed standard hyperparameter, were able to measure important variables (in this case, technical indicators) on its own and leverage them to build a powerful model that perform consistently well. Additionally, Chen and Guestrin (2016), which are the original developers of XGBoost, stated that XGBoost have been winning competitions with various cases, from store sales prediction, high energy physics event classification, and web text classification to customer behaviour prediction and many more, even against other popular solutions like DL models. Furthermore, XGBoost is known for its ability to efficiently handle high-dimensional data, which translates to its high computational efficiency and speed without sacrificing its predictive performance. Yun, Yoon and Won (2021) stated that one of the challenges in stock price movement prediction is the expensive computational cost of numerical simulation, in which XGBoost performed more efficiently as compared to other deep learning (DL) algorithms. Additionally, Chen and Guestrin (2016) stated that XGBoost, notably in many different scenarios, were able to perform 10 times faster than any other existing solutions.

While XGBoost has demonstrated superior performance in stock price prediction, it has its own limitations. One of them is its lack of interpretability. Many ML classification algorithms have complex internal workings and thus considered as low-interpretation "black-box" models, in which the decision-making processes of these models cannot be transparently visualised and demonstrated in most cases, making it difficult to be understood (Zhang et al., 2022). XGBoost is one of these low-interpretation "black-box" models (Zhang et al., 2022). In the context of stock market prediction, this may pose some significant issues towards investors and analysts as they require transparency in their decision-making processes to justify their strategies. As a result, some research has been focused on interpretable machine learning using several methods such as SHAP and LIME, to provide clearer and more transparent model decisions.

2.4. The Role of Financial Sentiment

Financial sentiment refers to market participants' collective mood and attitudes, which are expressed through textual data such as social media posts, financial news articles and reports, and commentaries. Unlike quantitative measures such as technical indicators, sentiment captures both psychological and behavioural drivers of market movements, reflecting investors' reactions to different kinds of ongoing events. Stock market is a highly volatile and information-driven, in which sentiment can represent price swings that are not caused by fundamental or historical price patterns, making it a valuable variable worth being included in predicting future stock price (Li et al., 2014). Many papers have well documented the fact that the stock sentiments mainly come from both social media platforms and financial news articles (Yang, Fernandez-Perez and Indriawan, 2024).

Among many other social media platforms, Twitter is among the most popular, where users express their thoughts on a wide range of topics, including the stock market. Twitter also captures wide range of statements and beliefs from different societal backgrounds, thus ensuring the providence of rich information and avoiding biased expert or journalistic viewpoints, which typically occurs in traditional media platforms like newspaper (Ma, Li and Zhou, 2025). Consequently, these tweets can provide valuable insights regarding public's opinion towards certain stocks (Batabyal et al.,

2023; Chaudhuri et al., 2018; Kumbure et al., 2022). For example, Das et al. (2024) found out that enhanced deep learning (DL) model with twitter sentiment features performs better in terms of all evaluation metrics than traditional models with technical-indicator-only baseline at predicting future prices. Additionally, Ma, Li and Zhou (2025) successfully integrated Twitter-derived market uncertainty to predict daily market volatility and proved its predictive prowess to remain robust against various alternative benchmark approaches, estimation methods, rolling window sizes, and volatility measures, thus further supporting tweets' value in affecting price movements. Despite its potential benefits, irrelevant tweets may be included within the sample and thus introducing noise to the model that can potentially skew the results (Das et al., 2024). Thus, the use of strict filters such as semiautomatic procedure based on hashtags is highly recommended to improve the dataset quality.

Financial news articles, from a long time ago, have been proven to influence financial markets, such as those that were documented by Dougal et al. (2012), Engelberg and Parsons (2011), and Fang and Peress (2009). In contrast to social media, financial news articles from outlets such as Bloomberg, Reuters, and The Wall Street Journal provide a more structured and formal form of information, contains both qualitative narratives and quantitative measurements on company fundamentals that may affect market's expectations and returns (Hao et al., 2021). For instance, Pinheiro and Dras (2017) explored hybrid recurrent neural networks (RNN) and character-level pretrained model incorporating financial news and showed, as a result, a competitive model with other state-of-the-art (SOTA) approaches in terms of stock price directional changes prediction. Furthermore, Li et al. (2021) proposed a tensor-based Long Short-Term Memory (LSTM) model to predict stock price movements by considering multimodal market information, putting news on top of fundamental information, and demonstrated the model's superiority by at least 8.49% in terms of Sharpe ratio over other SOTA approaches like TeSIA and eMAQT. This further strengthens the positive impact of integrating news on top of other significant variables in predicting stock price movements. However, financial news emerges randomly and news about a stock might impact other correlated stocks as well (Li et al., 2021). Thus, it is crucial to continuously monitor and filter relevant news across the market to capture indirect effects and enhance the overall performance of stock price predictions.

In conclusion, despite the documented benefits, integrating sentiment data poses some challenges but also solutions as mentioned in the above paragraphs. In further effectively addressing the available problems, researchers have explored some advanced natural language processing (NLP) techniques, particularly domain-specific fine-tuned pretrained models such as FinBERT, that not only can vectorize and classify financial texts but also understand the context of each financial-related terms (Araci, 2019).

2.5. Advances in NLP: FinBERT and Financial Text Modelling

2.5.1. Pretrained Language Models in Finance and Their Applications

The rise of Transformer architectures marked a paradigm shift in natural language processing (NLP). Traditional NLP models such as Recurrent Neural Networks (RNN) and their variants, including LSTM and GRU, can only process text one token at a time and in order. This means these approaches assume a bag-of-words structure and ignore context by treating text as a collection of independent individual words instead of considering the grammar and word order (Huang, Wang and Yang, 2023). Transformers addressed these issues by having self-attention mechanisms, thus replacing the recurrence nature of traditional NLP models and can link words despite their distances, which allows them to consider context. These algorithms are also often referred to as large language models (LLMs) as they have large numbers of parameters (up to billions) and can learn both semantic and syntactic relationships among words in which they are trained by (Huang, Wang and Yang, 2023). Building on this foundation, the development of Bidirectional Encoder Representations from Transformers (BERT) introduced two pretraining objectives, namely Masked Language Model (MLM) and Next Sentence Prediction (NSP), to learn deep context from sentence-level tasks. MLM forces the model to predict masked tokens representations using both left and right (bidirectional) context, while NSP trains it to understand relationships between text pairs (Devlin et al., 2019). These advances, particularly BERT, achieved state-of-the-art (SOTA) performance on a large set of both sentence and token-level tasks, and was able to outperform many other task-specific architectures such as ELMo by Peters et al. (2018) and Generative Pre-trained Transformer (OpenAI GPT) by Radford et al. (2018) (Devlin et al., 2019).

Despite its success, the original BERT model was pretrained with only general texts as the focus was not about domain-specific optimisation, but only to achieve SOTA results in various NLP tasks, including natural language inference and question answering (Araci, 2019). Thus, when being faced with financial texts written by professional investors or some financial domain-specific terms, BERT model may not be able to process it well (Huang, Wang and Yang, 2023). For context, the BERT model's vectorizer output for the word "bank" may be derived from multiple related meanings, such as "fish" or "river", whereas in financial context, it is more closely related to "lending" or "loan". Thus, researchers have been trying to address this domain shift by continuing to optimise the base BERT model, also known as pretraining, before further fine-tuning the model. In the domain of finance, the BERT model was further trained on large-scale financial corpora, resulting in the creation of FinBERT. As a result, FinBERT demonstrated a 3.2% and 3.6% increase in terms of model accuracy and F1-score as compared to the baseline BERT model when faced with out-of-sample financial sentiment classification tasks under multiple scenarios (Huang, Wang and Yang, 2023).

Following FinBERT, several efforts have also been expanded to develop stronger performing domain-adapted models and covering a wider scope of cases such as the multilingual contexts. AT-FinGPT (Liu et al., 2025), for example, developed an ability to not only process text data but also capture audio and summarise financial texts. Empirical studies discovered that AT-FinGPT consistently outperformed Generative Pre-trained Transformer (GPT) by at least 88.16% in terms of mean squared error (MSE) under multiple scenarios of predicting China's CSI 1000 stock pool (Liu et al., 2025). Note that GPT, when being compared head-to-head with FinBERT in the case of sentiment analysis performance across different sectors (such as business, health, and technology), were able to outperform FinBERT by 13.77% in terms of average F1-score (Kang and Choi, 2025). In non-English contexts, MFinBERT (Nguyen et al., 2022) leverages the architecture of BERT-based-multilingual-cased and was pretrained on additional 15 GB of Vietnamese data and 300 MB of Lithuanian data from uncompressed texts, respectively, to achieve better performance in multilingual question-answering tasks.

However, these enhanced approaches (LLMs) are subject to some limitations compared to simple NLP approaches, namely computational costs and their specific

use cases. The performance and output quality of such large language models (LLMs) such as GPT highly depends on the prompt design, which requires extensive experimentation for different cases and thus, leading to time constraints (Kang and Choi, 2025). The fact that these LLMs are “black box” in nature (Castelvecchi, 2016; Lauriola et al., 2022) may pose more significant problems in the case of testing economic theories (Loughran and McDonald, 2016). Although these algorithms’ general mechanisms are relatively straightforward, the large number of parameters make it extremely difficult to identify how important inputs translates to the outputs (Huang, Wang and Yang, 2023). Furthermore, additional pretraining requires substantial computational resources (e.g., high-capacity processors such as Tensor Processing Unit (TPU)), which when combined with the before-mentioned condition, is extremely costly (Huang, Wang and Yang, 2023; Nguyen et al., 2022).

Ultimately, pretrained language models in finance have been proven to be successful in bridging the gap between general NLP capabilities and the unique demands of financial text, especially in tasks such as sentiment analysis, question answering, and financial forecasting. By leveraging various approaches, models like FinBERT and AT-FinGPT can better capture contexts, sentiment subtleties, and even have multimodal capabilities for enhanced predictions.

2.5.2. Advantages of FinBERT

FinBERT was introduced to address the previous approaches’ limitations in solving domain-specific tasks within the financial sector, particularly through domain-adaptive pretraining, thus offering some critical advantages as a foundational model for financial text analysis.

First and foremost, FinBERT demonstrates better knowledge specific to financial language, thereby enhancing its relevance in the domain of financial projects. Unlike the original BERT, which was pretrained on a general corpus such as BooksCorpus and English Wikipedia (Devlin et al., 2019), FinBERT was pretrained on domain-specific financial texts, including the TRC2-financial corpus (a set of Reuters’ Technological Research Collection 2 (TRC2) filtered for financial content) and was further fine-tuned on sentiment analysis datasets like the Financial PhraseBank and FiQA Sentiment (Araci, 2019). This enhances FinBERT’s ability in understanding

financial jargons and context-specific nuances, making it a robust tool for financial sentiment analysis-related tasks (Shen and Zhang, 2024).

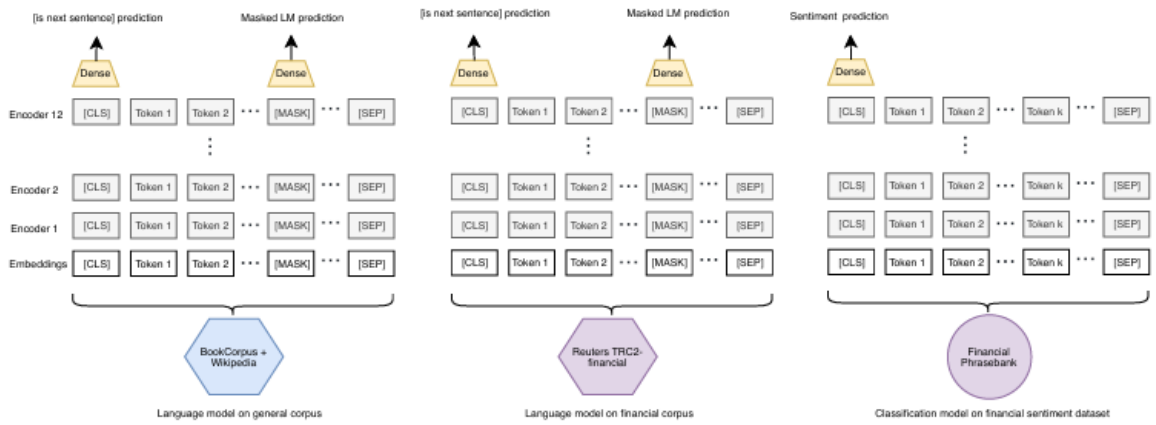


Figure 4. Overview of FinBERT's pre-training and fine-tuning processes (Source: Araci, 2019)

FinBERT proves its superior sentiment sensitivity when dealing with financial discourse, where sentiment expressions will have impact on investors. For instance, FinBERT showed a distinct advantage when detecting negative sentiments, providing 89.7% accuracy as compared to non-BERT models' less than 60% accuracies, which is worth noting since negative sentiments tend to influence investors more strongly than the positive ones (Huang, Wang and Yang, 2023; Huang, Zang and Zheng, 2014; Loughran and McDonald, 2011). In addition, Huang, Wang and Yang (2023) stated that FinBERT's advantages over base BERT models come from its reliance on financial terminologies, specifically, when classifying sentiment.

Another major advantage of FinBERT is its data efficiency. Proven by empirical studies, FinBERT was able to retain a large proportion of its predictive power, even when the training data is substantially reduced. When the training sample was being reduced to only 10% of the full size, FinBERT was still able to retain 92% of its original accuracy whereas the baseline BERT model was only able to retain 73% (Huang, Wang and Yang, 2023). This resilience makes FinBERT valuable in financial applications where datasets are often limited or costly to obtain.

Not only against non-BERT and base BERT models, FinBERT has shown to be able to hold its ground against other large language models (LLMs) as GPT-3.5-turbo and GPT-4o. Although prompt engineering significantly enhanced the performance of these LLMs, allowing GPT-4o to perform as good as FinBERT in finance-specialised fields, its performance may vary based on the complexity and specificity of the prompts. Therefore, in most finance-specific scenarios, a well fine-tuned FinBERT consistently outperformed prompt-engineered general-purpose LLMs (Shen and Zhang, 2024).

In sum, FinBERT represents a major advancement in domain-specific NLP, offering contextual understanding, sentiment sensitivity, and data efficacy while still relevant with the emergence of other LLMs. These advantages have led to its wide adoption in both research and practical applications, such as hybrid models between NLP and ML.

2.6. Hybrid Models: Integrating Technical and Sentiment Features

2.6.1. Justification for Hybrid Models

The complex nature of financial markets requires a more holistic approach to predictive tasks instead of relying solely on a single source of information. Traditionally, the financial forecasting models have only emphasized technical indicators, for example moving averages, relative strength index (RSI), and moving average convergence divergence (MACD), to capture patterns and momentum within stock prices. Although these indicators were effective in detecting market trends and cyclicality, they have some limitations when detecting unpredictable price shifts, often caused by factors such as investor sentiments, macroeconomic shocks, or geopolitical events, which are often being reflected through financial news articles and social media platforms (Li et al., 2014; Yang, Fernandez-Perez and Indriawan, 2024).

Conversely, many NLP-based models like FinBERT could capture the qualitative aspects of the stock market by leveraging textual analysis. However, the use of single-view information, may cause the model to suffer from noise and information deviation (Long et al., 2024). For instance, irrelevant social media and financial news data such as transient public emotions and media exaggeration might unintentionally be included in the data and bias the model's predictions (Das et al., 2024; Li et al., 2021; Lin et al.,

2022). Ultimately, without the support of both market and sentiment data, the model will not be able to capture and justify the prediction pattern.

As illustrated shown in Figure 5, different market scenarios highlight the importance of incorporating both sentiment and technical data. Upon using only sentiment data, in this case news, a “Negative” news would rationally suggest a stock price to “Fall” whereas “Positive” news should signal a “Rise”. However, real market behaviour is often more complex, where the momentum and trend of the price data may conditionally oppose the sentiment direction. For instance, despite the release of “Negative” news, the bullish momentum from technical stock price indicators might still drive the stock upward, which potentially leads to an incorrect prediction.

This also happens when using only historical price as the model predictor without considering the sentiment at all. The model might predict a “Rise” based on upward trend from the indicators, even when there are released “Negative” news, which could fundamentally suggest a “Fall”. The difference between price patterns and sentiment signals may cause the model to fail in explaining the rationale behind market movements when only one type of feature is being considered.

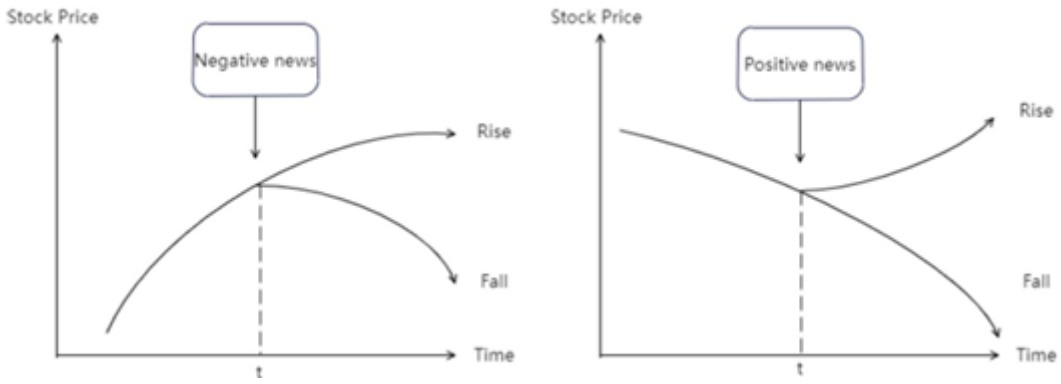


Figure 5. Possible Market Scenario (Source: Long et al., 2024)

Consequently, hybrid models emerge as a necessary development that leverages the richness of both technical and sentiment data of the stock market, allowing for a more robust understanding of the market behaviour. Empirical evidence supports the advantages of hybrid models. Usmani and Shamsi (2023) applied weighting and

categorisation to financial news using Vader and HIV4 models, integrate it with technical indicators, then fitting it into the LSTM model, resulting in the creation of a hybrid WCN-LSTM model. They discovered that this model performed exceptionally well, which was proven by conducting Wilcoxon signed-rank test to various experiments. Long et al. (2024) also leverage the capability of the bag-of-words and TF-IDF model for feature extraction from news data and support vector machine (SVM) for stock price direction classification, resulting in a hybrid model called MVL-SVM that successfully achieved above 70% predictive accuracy in predicting various stocks from the Shanghai Stock Exchange 50 index.

Therefore, hybrid models are not just theoretically justified but also practically validated, offering an excellent solution in adapting to various market conditions.

2.6.2. Feature Engineering and Processing for Sentiment Data

Although some ML models, particularly ensemble methods and deep learning architectures, have robust built-in feature selection and thus not needing extensive feature engineering towards the training data, several studies have proven that targeted feature engineering of sentiment data, particularly the combination of standardisation and dimensionality reduction methods, have been able to increase predictive performances.

One of the issues when working with vectorized text, or embeddings data is that it has high dimensionality and thus, variability and sparsity as well. For instance, the bag-of-words produces *n* number of columns for *n* number of unique words (Zhang, Jin and Zhou, 2010). Another example is in the case of BERT model. Suppose a column in a data frame consist of text data. When BERT-base model is used for extracting numerical representations of that text column, 768 columns will be generated regardless of how many words are inside that column (Huang, Wang and Yang, 2023). High dimensional data have been proven to not only increase computational costs but also having the possibility of degrading machine learning models' performance, also known as the "curse of dimensionality" (Fernández-Martínez and Fernández-Muñiz, 2020; Kim and Lee, 2014). Consequently, some dimensionality reduction methods were proposed to address these issues.

Dimensionality reduction methods can be generally separated into 2 categories: linear and nonlinear. Linear methods such as principal component analysis (PCA) and multidimensional scaling (MDS), although fast and easy to implement, are based on linearity assumptions, which is not applicable in the financial settings, which is almost always nonlinear (Dudek et al., 2024; Lv et al., 2022). Consequently, some nonlinear dimensionality methods emerge to address the implementation limitations of the linear ones. Methods such as Kernel PCA (KPCA), Isomap, and t-distributed stochastic neighbour embedding (t-SNE) are some that can capture low-dimensional structure effectively without sacrificing implementation efficiency (Kim and Lee, 2014).

Although robust, even nonlinear dimensionality reduction methods rely on distance-based calculations, such as Euclidean distance (Kim and Lee, 2014). As a result, these methods are sensitive to the scale of the features. Since vectorizer like BERT and its variants produce dense vectors, applying feature scaling, typically standardisation methods, is crucial to ensure that no single feature heavily influencing the distance computation. Additionally, feature scaling, although not always, generally improves most machine learning model performance (Géron, 2019).

Some studies have combined feature scaling with dimensionality methods to achieve higher predictive results. For instance, Kim and Lee (2014) standardised all text embeddings before applying various dimensionality reduction methods to classify sentiment and some were able to at least retain and mostly increase the models (SVM and KNN) prediction accuracy by up to 25%. Note that the methods were tested for different datasets, namely book, DVD, electronics, and kitchen. Some of the best performing dimensionality reduction methods were able to create clusters based on the similarity of the word embeddings for each category, which were proven by 2D scatter plot visualisations, thus generating better thresholds and therefore, prediction accuracy between the classes.

In summary, while modern ML architectures are capable for feature abstraction, targeted feature engineering techniques such as standardisation and dimensionality reduction remain an indispensable practice that may enhance the efficiency and robustness of hybrid models.

Chapter 3: Methodology

3.1. Ethical Consideration

Research ethics refer to the appropriateness of the author's behaviour in relation to the rights of those who are the subject of the author's work or the work. The author and other researchers involved must make sure that there is no harm (Saunders, Lewis and Thornhill, 2021). Any potential ethical issues must be recognized and considered from the beginning of the research and must be one of the research's criteria (Saunders et al., 2012). This study is conducted for educational purposes only and several key ethical considerations have been recognized at the beginning of the study.

This project has received research ethical approval in line with the Research Ethics Policy and Procedures of Leeds Beckett University. Below are the components of this project that led to the above research ethical approval:

1. The data used for this study are all publicly available. The stock price data are openly provided by Yahoo Finance API and extracted using the yfinance library in Python. The social media data, tweets, which in this case came from twitter, are part of the paper published in the 2020 IEEE International Conference on Big Data under the 6th Special Session on Intelligent Data Mining track, which is publicly available on Kaggle. Finally, the financial news data were obtained by performing prompt engineering with DeepSeek, one of the publicly available chatbots, to extract summaries from publicly accessible financial news headlines such as Bloomberg, Nasdaq, and many more.
2. The extracted tweets data contain only one personal identifier column, which is the account name of the author. However, only the tweets and news content are aggregated with the pricing data to be examined and thus does not include the above-mentioned column at any time within the process. Additionally, all raw tweets and financial news are deleted after the sentiment vectorization and classification process.
3. This study acknowledges that there may be sampling bias from tweets or news data, where all the extracted data may not represent the entire investor

population. Language and regional biases may be present as well due to the data being extracted are all from English-language sources. Thus, there is also some possibilities where the result of this study will not be perfectly applicable in all real-world scenarios. However, some mitigation strategies have been proposed, including using balanced-sampling overtime, which avoids over-weighing any single source. This mitigation strategy has been applied in the collection of financial news data, where the collected data are from several different reliable sources.

4. There may be some potential harms and misuse of the developed model in this study, such as market manipulation risk. Thus, it is clearly stated that this study is for academic insight only, not trading advice. All codes and trained-model details will be provided in a GitHub repository to allow reproducibility.
5. Finally, all open-source components, including any open-source libraries (e.g. HuggingFace's FinBERT) and their licences will be fully listed and acknowledged. All pre-trained models, APIs, and data sources used in this project will be properly cited.

3.2. Data Collection

Saunders, Lewis and Thornhill (2023) stated that secondary information consists of data already collected, may be processed, and analysed to provide knowledges, interpretations, or conclusions. The dataset used for this research consists of three major components:

1. Stock market quantitative historical data
2. Social media data
3. Financial news data

All these data are classified as secondary data, as all of them are publicly available on the internet. The stock that is being researched in this study is the Tesla stock (TSLA), as it was the only company that has the complete data, namely the historical quantitative, social media, and financial news data over the same period from 1st January 2015 to 30th November 2020.

3.2.1. Stock Market Historical Quantitative Data

The stock market historical data consists of price components such as date, open, high, low, and close as well as volume, dividends, and stock splits indicator. This dataset is publicly available and accessible by yfinance application programming interface (API) via Python programming language. This study will use specifically Tesla stock (TSLA) data consisting of 6 components from the historical stock data, namely date, open, high, low, close, and volume. This dataset, as mentioned before, ranges from 1st January 2015 to 30th November 2020, which aligns with the available social media and financial news data.

3.2.2. Social Media Data

The social media data being used in this research comes from Twitter data, also known as tweets. These tweets are part of the paper published in the 2020 IEEE International Conference on Big Data under the 6th Special Session on Intelligent Data Mining track, which is publicly available on Kaggle under the name “Tweets about the Top Companies from 2015 to 2020”. This dataset contains over 3 million unique tweets with information such as tweet id, author of the tweet, postdate, the text body of the tweet, and the number of comments, likes, and retweets of tweets matched with the related company ranging from 2015 to 2020. These tweets are originally collected from Twitter by a parsing script based on Selenium with some motivations, namely researching about tasks in the topics of:

1. Determining the correlation between the market value of company respect to the public opinion of that company
2. Sentiment Analysis of the companies with a time series in a graph and reasoning the possible declines and rises
3. Evaluating troll users who try to occupy the social agenda

These tweets are all about these 5 companies, which are Amazon, Apple, Google, Microsoft, and Tesla. In this research, only the Tesla stock (TSLA) data consisting of 2 components, namely the postdate and text body, are being used.

3.2.3. Financial News Data

As stated by Ahluwalia and Wani (2024), since Large Language Models (LLMs) are trained on massive datasets like text and code with an addition of effective chunking, searching, and ranking algorithms, it possesses the potential to accomplish data extraction from vast textual repositories with proper prompt engineering, thus enabling a faster and more accurate data extraction. The DeepSeek LLM model's training data includes publicly available texts up to 2023 and is sensitive to prompting (Bi et al., 2024) and this research only requires data up to the year 2020. Under these assumptions, the financial news data were prompted from the open source DeepSeek chatbot. Careful prompting was being done, resulting in successful retrieval of the required data from various sources for this research, which consists of 2 variables, namely the date and the summary of the aggregated news per day from 1st January 2015 to 30th November 2020.

3.3. Feature Engineering

3.3.1. Target Feature Creation and Definition

Predicting the stock price direction is considered as a binary classification problem where the target variable in this study is defined as:

$$Y_{t+1} = \begin{cases} 0, & O_{t+1} \leq C_t \\ 1, & O_{t+1} > C_t \end{cases} \quad (1)$$

Y_{t+1} is the predicted stock price direction at day t + 1, O_{t+1} is the opening price of day t + 1, and C_t is the closing price of day t. If the value of O_{t+1} is less than the value of C_t, which indicates a downward direction, the Y_{t+1} will be 0 and Y_{t+1} will be 1 otherwise.

Y'_{t+1} represents the predicted price direction of day t + 1. Y'_{t+1} will be predicted according to the given input set features of day t, which are represented by X_t.

Thus, Y'_{t+1} will be a function of the input features X_t,

$$Y'_{t+1} = f(X_t) \tag{2}$$

f is a nonlinear function that maps all input features represented by X_t at day t and will result in a binary outcome value of $\{0, 1\}$ at day t + 1.

The components of X_t can be broken down as below:

$$Date\ Components \begin{cases} D_t^D \\ D_t^{WD} \\ D_t^W \\ D_t^M \\ D_t^Y \end{cases}$$

$$Current\ and\ Historical\ Values \begin{cases} H_{t-n}^O & H_{t-n+1}^O & \dots\dots\dots & H_t^O \\ H_{t-n}^H & H_{t-n+1}^H & \dots\dots\dots & H_t^H \\ H_{t-n}^L & H_{t-n+1}^H & \dots\dots\dots & H_t^L \\ H_{t-n}^C & H_{t-n+1}^H & \dots\dots\dots & H_t^C \\ H_{t-n}^V & H_{t-n+1}^H & \dots\dots\dots & H_t^V \end{cases}$$

$$Technical\ Indicators \begin{cases} I_{t-n}^1 & I_{t-n+1}^1 & \dots\dots\dots & I_t^1 \\ I_{t-n}^2 & I_{t-n+1}^2 & \dots\dots\dots & I_t^2 \\ & & \dots\dots\dots & \\ & & \dots\dots\dots & \\ & & \dots\dots\dots & \\ I_{t-n}^k & I_{t-n+1}^3 & \dots\dots\dots & I_t^k \end{cases}$$

$$Sentiment\ Features \begin{cases} T_t^E \\ T_t^S \\ N_t^E \\ N_t^S \end{cases}$$

Substituting this into equation 2, the full feature-based prediction formula will be represented by the following equation 3:

$$Y'_{t+1} = f(D_t^D, D_t^{WD}, D_t^W, D_t^M, D_t^Y, H_t^O, H_t^H, H_t^L, H_t^C, H_t^V, I_t^1, \dots\dots\dots, I_t^k, T_t^E, T_t^S, N_t^E, N_t^S) \quad (3)$$

This formula allows the model to learn the complex relationships between provided rich feature set in forecasting the direction of stock prices.

3.3.2. Historical Quantitative Feature Expansion

The first step of the feature expansion is to leverage the date column to generate 5 temporal features, including day, week, weekday, month, and year. This allows the model to capture seasonality, calendar-based patterns, and potential periodic effects in market behaviour.

Next, the feature set will be expanded by generating technical indicators from the open, high, low, close, and volume data. Unlike the classic Markov process that assume the future price is only affected by the current price and not by the sequences previous prices, previous studies have proven that the stock price data often exhibits autocorrelations, volatility clustering, momentum patterns, and reversal phenomenon (Bremer and Sweeney, 1991; Leung and Chen, 2010; Rahman, Ara and Zheng, 2009; Wu, 2011), indicating that past information does effect on future prices and thus, justifying the use of technical indicators and other historical values. The optimal number of technical indicators to predict future stock price directions remains to be researched, however, this study will use some indicators leveraged in most previous studies (Das et al., 2024; Han, Kim and Enke, 2023; Ismail et al., 2020; Yun, Yoon and Won, 2021) and some additional important features as follow:

Category	Indicator	Purpose
Trend-Following	Exponential Moving Averages (EMA)	Capture short-to-medium term price trends.

	Moving Average Convergence Divergence (MACD)	Identify momentum shifts and trend reversals.
	Average Directional Index (ADX)	Measure the strength of a trend.
Momentum Indicators	Relative Strength Index (RSI)	Detect overbought or oversold conditions based on recent gains/losses.
	Stochastic Oscillator	Identify potential reversal points based on momentum.
Volatility Measures	Rolling Standard Deviation	Quantify recent volatility to assess risk or instability.
	Bollinger Bands	Detect volatility-driven price breakouts using upper/lower bounds.
Volume-Based Indicators	Volume Moving Averages (MA)	Track short-term volume trends.
	On-Balance Volume (OBV)	Assess buying/selling pressure through price-volume relationships.

Table 1. Commonly Used Technical Indicators

Below are the formulas of the technical indicators above:

1. **Exponential Moving Average (EMA)**

$$EMA_t = \alpha \times P_t + (1 - \alpha) \times EMA_{t-1} \quad (4)$$

where $\alpha = \frac{2}{N+1}$ and N is the window size.

2. Moving Average Convergence Divergence (MACD)

$$MACD\ Line = MACD_t = EMA_{12}(P_t) - EMA_{26}(P_t) \quad (5)$$

$$Signal\ Line = Signal_t = EMA_9(MACD_t) \quad (6)$$

$$Histogram = Histogram_t = MACD_t - Signal_t \quad (7)$$

3. Average Directional Index (ADX)

$$ADX = \frac{1}{n}\sum DX_t \quad (8)$$

where $DX_t = \frac{|+DI_t - -DI_t|}{|+DI_t + -DI_t|} \times 100$

4. Relative Strength Index (RSI)

$$RSI = 100 - \frac{100}{1+RS} \quad (9)$$

where $RS = \frac{Average\ Gain}{Average\ Loss}$

5. Stochastic Oscillator (%K and %D)

$$\%K = \frac{C - L_n}{H_n - L_n} \times 100 \quad (10)$$

$$\%D = SMA_3(\%K) \quad (11)$$

where C = Close, H_n = High over n days, L_n = Low over n days

6. Rolling Standard Deviation

$$\sigma_t = \sqrt{\frac{1}{n}\sum(P_{t-i} - \mu)^2} \quad (12)$$

where μ is the mean price over n days

7. Bollinger Bands

$$Upper\ Band = MA_{20} + (2 \times \sigma_{20}) \quad (13)$$

$$Lower\ Band = MA_{20} - (2 \times \sigma_{20}) \quad (14)$$

8. Volume Moving Averages (Volume MA)

$$Volume\ MA_t = \frac{1}{n}\sum Volume_{t-i} \quad (15)$$

9. On-Balance Volume (OBV)

$$OBV_t = OBV_{t-1} + Volume_t, \quad if\ Close_t > Close_{t-1} \quad (16)$$

$$OBV_t = OBV_{t-1} - Volume_t, \quad if\ Close_t < Close_{t-1} \quad (17)$$

$$OBV_t = OBV_{t-1}, \quad Otherwise \quad (18)$$

The above features in this study are computed using the functions provided by the ta library by Bukosabino in Python (Bukosabino, 2023). The ta library is a widely used library for performing technical analysis of financial market data and includes around 43 indicators such as the one used by previous studies. The look-back periods for every indicator ranges from 1 day up to 26 days to capture short-term dependencies.

3.3.3. Sentiment Feature Extraction using FinBERT

Following the technical indicators, there are two other variables that must be represented numerically to be able to fit as training data into the model, namely the tweets and financial news body. In this research, FinBERT is used to vectorise both tweets and financial news data separately. FinBERT is a domain-specific model that is pretrained on financial corpora and is specifically designed to accomplish sentiment analysis tasks such as vectorisation (or embedding) and sentiment classification (classifying texts into negative, neutral, or positive), thus capable to becoming both encoder and classifier. The way FinBERT generates embeddings is almost similar as BERT, however, since it is fine-tuned on financial text, it changes the content of the embeddings.

As being shown in Figure 6, BERT does 3 types of embeddings that will be combined into its final embeddings, namely the token, segment, and position embeddings. On the first stage, which is token embeddings, BERT will convert raw text into WordPiece tokens. For instance, the word “playing” will be converted into “play” + “##ing”, in which each token will be mapped to a 768-dimensional vector. The segment embeddings distinguish between sentences. For instance, “my dog is cute” and “he likes playing” will be separated into 2 sentences, namely sentence A and sentence B, instead of treating them as 1 sentence. Finally, the position embeddings will track the order of each word in the sequence, ensuring that BERT understands the word arrangement. Finally, the final embeddings will be the sum of all 3 stages, resulting in a 768-dimensional numerical representation of a set of texts.

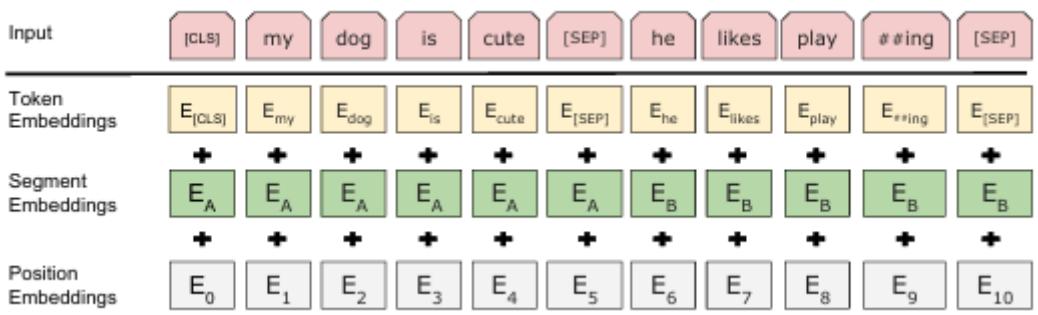


Figure 6. Representation of BERT Embedding (Source: Devlin et al., 2019)

FinBERT, although executes embeddings the same way as BERT, has a considerable difference in the token embeddings stage, especially when processing financial texts. For instance, when there is a word like “EBITDA” or “Bull”, BERT will embed it as “EB” + “##IT” + “##DA” or understand the “Bull” as a type of animal, whereas FinBERT will treat “EBITDA” as a single token and understand “Bull” as a market condition. Thus, in the case of this research, which uses financial tweets and news related to Tesla stock, FinBERT is a superior embedding model.

Additionally, FinBERT was used to classify both the tweets and news as either “negative”, “neutral”, or “positive”, represented numerically as -1, 0, and 1. Since FinBERT understands financial context better and was repeatedly shown to have superior sentiment sensitivity when dealing with financial texts (Huang, Wang and Yang, 2023), FinBERT, on top of being open-source, comes out on top as a sentiment classifier model.

Thereby, for the 2 columns representing tweets and financial news aggregated texts, a total of 1538 columns representations are generated in addition to the above quantitative historical data and technical indicators.

3.3.4. Possible Application of Sentiment Feature Scaling and Dimensionality Reduction

In the case when either or both social media (tweets) and financial news embeddings and classifications fails to improve the model performances, additional preprocessing steps will be explored. Specifically, feature scaling and dimensionality reduction method will be applied to the underperforming representations as several literatures have repeatedly proven the capability of these combination methods in improving model performance.

Of course, before applying any feature scaling and dimensionality reduction methods, the data will be split into training and evaluation set. Then, the training data will be fitted and transformed through both methods and the evaluation data will be transformed based on the fitted training data. This approach prevents data leakage, which occurs when the information of the evaluation data leak into the evaluation data. This will cause several issues, especially overfitting, where the model performs

extremely good on the evaluation but may perform significantly worse in the case of real-world application (Lantz, 2023). When the model already has some information about the evaluation set, it will naturally perform well. However, real-world application will sometimes be completely different than the training data, causing the model to be completely blind. The main purpose of strictly splitting the data is to simulate real-world scenarios and evaluate whether the model can generalise unforeseen data well. In this research, cross-validation approaches will not be applied as the data is time-series, making it impractical to shuffle the data as it will create unrealistic scenarios such as training future data to predict past data.

First, feature standardisation using z-score normalisation will be applied to the FinBERT-generated embeddings. Of course, this feature scaling is not being done to increase the classifier model performance, which is XGBoost, but rather to increase the performance of the dimensionality model. As mentioned in the literature review, almost all dimensionality models, even the nonlinear ones, used distance-based computational methods (Kim and Lee, 2014). The standard scaler function available via the Scikit-Learn library in Python provides feature standardisation using the following equation:

$$z = \frac{x - \mu}{\sigma} \tag{19}$$

Where:

- x is an individual feature value
- μ is the mean of the feature in the training set
- σ is the standard deviation of the feature in the training set
- z is the standardised value

As stated by Lantz (2023), the standardised value of z-score does not have a predefined minimum and maximum value. This allows the extreme values to retain its impacts toward the model, which in the case of embeddings, is highly practical as each token hold an extremely valuable contribution towards the model.

Following standardisation, a dimensionality reduction method called Uniform Manifold Approximation and Projection (UMAP) will be applied to the possible nonperforming sentiment representations. Embedded data, especially by FinBERT, have been known to be highly sparse and variable, making it not only computationally intensive but also nonlinear by nature. The use of UMAP is motivated by the fact that UMAP does not follow linearity assumptions and able to efficiently handle large amount of data. Furthermore, UMAP seeks a low-dimensional representation of the data that preserves the global structure of the original data as closely as possible (Vermeulen et al., 2021), making it a suitable approach to reduce data dimensionality without losing important information.

3.4. Modelling with XGBoost Algorithm

XGBoost stands for extreme gradient boosting, a scalable and highly efficient tree-boosting algorithm that will be used in this study (Chen and Guestrin, 2016). XGBoost prediction process and advantages can be explained as follows.

Like other tree-based ensemble model, for a dataset D with n samples and m features, $D = \{(x_i, y_i)\}\ (|D| = n,\ x_i \in R^m,\ y_i \in R)$, where:

- $x_i \in R^m$ is a data point with m features (for example: closing price, volume, sentiment score, etc.)
- $y_i \in R^m$ is the actual target feature for that data point (e.g., 1 for upward and 0 for downward price direction)

K additive functions will be used to predict the output by adding the outputs from multiple trees as presented in equation 5 below:

$$\hat{y}_i = \emptyset(x_i) = \sum_{k=1}^{K} f_k(x_i),\ f_k \in F \tag{20}$$

Where:

- Each f_k is an element of $F = \{f(x) = w_{q(x)}\}(q : R^m \to T, w \in R^T)$, where F is the space of regression trees (or also known as Classification and Regression Tree (CART)).
- The final prediction $\hat{y}_i$ is the sum of the predictions from all trees $f_1(x_i) + f_2(x_i) + \cdots + f_k(x_i)$.

In this context, each regression tree f_k is a function that assigns a data point to a leaf node and gives a weight from that leaf.

$$f(x) = w_{q(x)} \tag{21}$$

Where:

- $q(x)$ is the function that assigns an input x to a specific leaf index in the tree
- w is the weight at that leaf (a number that contributes to the final prediction)
- T is the total number of leaves in the tree

Therefore, for each data point x_i, it will go down the tree, following the decision rule (e.g., “Is the opening price > 200?”), and lands in one leaf. The tree will give that leaf a value (e.g., +0.55). This will be repeated for all trees and finally, the scores will be added up, as illustrated in Figure 7.

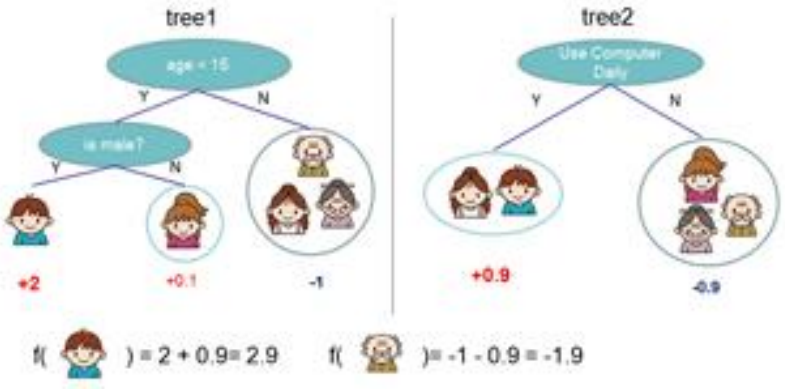


Figure 7. Illustration of Tree Ensemble Model Prediction (Source: Chen and Guestrin, 2016)

What makes XGBoost different from the regular gradient boosting models (GBMs) is that it adds a regularisation term to the above objective function, which is represented as:

$$L(\emptyset) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{22}$$

Where:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{23}$$

In this case,

- $l(\hat{y}_i, y_i)$ is the loss function, which measures how far off the model's prediction $\hat{y}_i$ is from the actual value y_i
- $\Omega(f_k)$ is the regularisation term, which penalizes complexity in each tree f_k
- γ and λ are the regularisation parameters that controls the given penalty (when set to 0, no penalty will be given to the tree complexity)

With the formula in equation 7, XGboost balances its accurate predictions while keeping the model simple and not overly complex. The regularisation also helps to prevent overfitting, which happens when tree-based models memorise the training data too well by building overly complex trees but perform poorly on unseen data (Chen and Guestrin, 2016).

Now, this is the process that makes XGBoost a boosting model. Rather than learning the entire model at once, XGBoost implement additive learning that adds one tree at a time, thus refining the prediction in each round. For instance, at iteration t, the current prediction will be:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{24}$$

The current prediction $\hat{y}_i^{(t)}$ will add a new tree f_t to improve the previous prediction denoted by $\hat{y}_i^{(t-1)}$.

Next, XGBoost minimises the total error at each step so that the model will look like:

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{25}$$

Where:

- l is the loss function (e.g., squared or logistic loss)
- $\Omega(f_t)$ is the previously mentioned regularisation penalty

To fasten the optimisation process, XGBoost uses a second-order Taylor expansion around $\hat{y}_i^{(t-1)}$:

$$L^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{26}$$

Where:

- $g_i = \frac{\partial(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ is the first derivative (gradient)
- $h_i = \frac{\partial^2 l(y_i, \hat{y}_i)}{\partial y_i^2}$ is the second derivative (curvature or Hessian)

Once the gradients and Hessians have been calculated for each data point, XGBoost will group the Data by leaf. For a tree structure with T leaves, each training sample will fall into a leave using a function $q(x_i) = j$, which means that sample i lands in leaf j. Thus, the objective of the tree can be written as:

$$L^{(t)} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2 \right] + \gamma \mathrm{T} \quad (27)$$

Where:

- $G_j = \sum_{i \in I_j} g_i$ is the total gradient in leaf j
- $H_j = \sum_{i \in I_j} h_i$ is the total second-order gradient in leaf j
- $I_j = \{i : q(x_i) = j\}$ represents all samples in leaf j
- w_j is the score or prediction value for that leaf
- λ and γ are the regularisation terms (weight shrinkage and tree complexity penalty)

To find the best score for each leaf j, the computation can be realised using the following function:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \quad (28)$$

And the result of the whole tree becomes:

$$L^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma \mathrm{T} \quad (29)$$

The lower the value goes, the higher the score and thus indicating a better tree structure.

Finally, XGBoost needs to measure the goodness of the tree split. XGboost can do it by letting:

- $I = I_L \cup I_R$ be a parent node split into left and right children

And letting the gain from making the split to be:

$$Gain = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_r^2}{H_R+\lambda} + \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right) - \gamma \quad (30)$$

This equation measures how much overall loss is being reduced from the split. The split with the largest gain will be chosen by XGBoost.

Beside the regularisation methods mentioned in equation 7, XGBoost further prevents overfitting by using two additional techniques, namely shrinkage and column subsampling (Chen and Guestrin, 2016). The shrinkage technique (Friedman, 2002), also known as the learning rate, adds an η factor weights after each tree boosting step, thus reducing the influence of individual tree and leaving a space for model improvements by future trees. This will allow the model to learn more slowly and build more robust trees. Mathematically, the prediction from equation 9 will be improved to:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i) \quad (31)$$

The column subsampling technique allows XGBoost to select a random set of features to train when building each tree. Instead of using all features for every boosting iteration, XGBoost will consider only a random portion (e.g., 80%) of features. This will not only reduce overfitting by introducing randomness but also speeds up XGBoost's complex computations (Chen and Guestrin, 2016).

XGBoost also introduces Sparsity-aware Split Finding method that handles sparse data such as the one introduced by feature engineering (e.g., one-hot encoding, feature embeddings) effectively. By performing the procedure illustrated in Figure 8, XGBoost optimise the split direction by visiting only the non-missing entries and treating non-presence values as missing. XGBoost, unlike other tree-learning algorithms, handle all sparse data uniformly and make the computation complexity linear, thus allowing it to run faster while keeping it robust as compared to other approaches.

Finally, XGBoost is known for its exceptional training speed as compared to other GBMs. This is due to 2 mechanisms, namely the Column Block for Parallel Learning and Cache-Aware Access Patterns. First, XGBoost uses a specialised data layout

called the Column Block structure to accelerate the most time-consuming part of tree learning, which is sorting the data for split finding.

Algorithm 3: Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  // enumerate missing value goto right
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I_k,$ *ascent order by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // enumerate missing value goto left
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*$(I_k,$ *descent order by* $\mathbf{x}_{jk})$ **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

Figure 8. Sparsity-aware Split Finding (Source: Chen and Guestrin, 2016)

This is being done by storing the data in Compressed Sparse Column (CSC) format, pre-sorting each feature column by its value, preparing this layout for training, and reusing the layout in every boosting iteration, thus enabling the data block to be scanned only once within the collection of all tree-leaves information during the splitting process and significantly reducing unimportant computation. Second, XGBoost leverages cache-aware algorithms to minimise memory access latency by fetching gradient statistics needed during split finding into local buffers in mini-batches. This approach avoids simultaneously reading and writing data, which allow the computer to more efficiently uses its memory and speeds up XGBoost's training process.

3.5. Evaluation Metrics

After developing the model, some benchmark must be used to compare how well the model has performed across different feature combinations and data treatment. Since this study involves the use of the classifier model XGBoost, the evaluation process will reflect on few important variables, including actual class values, predicted class values, and the estimated probability of the prediction (Lantz, 2023).

Confusion Matrix is a table that categorise whether the predicted values match the actual value and count them. For an n-class classification, an $n \times n$ confusion matrix will be formed. Confusion matrix is used to evaluate classification models performance for both binary and multiclass levels of the target variable (Kulkarni, Batarseh and Demir, 2020). Since the target variable in this study has only 2 variables, a $2x2$ confusion matrix will be formed. This confusion matrix is the base of most metrics that will be seen and compared in this study, including accuracy, precision, recall, and F1-score.

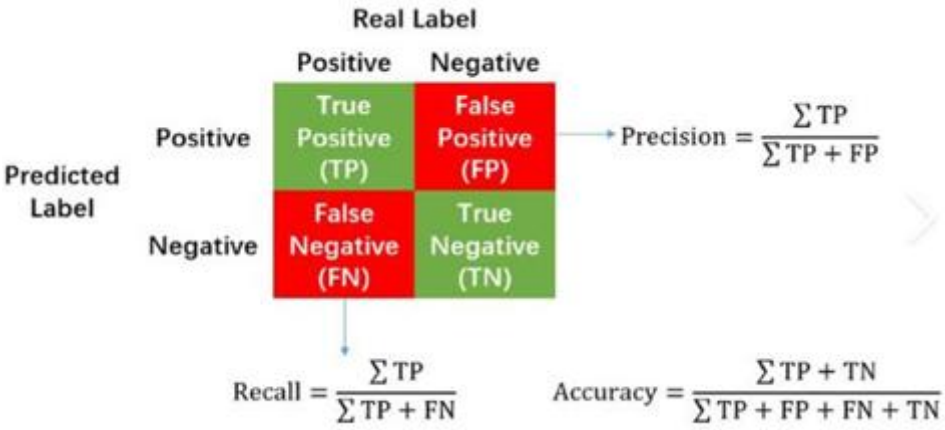


Figure 9. Confusion Matrix

As being shown in Figure 9, True Positive (TP) represents the number of correctly predicted positive instances, True Negative (TN) is the number of correctly predicted negative instances, False Positive (FP) is the number of incorrectly predicted positive instances, and False Negative (FN) is the number of incorrectly predicted negative instances. After generating the confusion matrix values, some major performance metrics can be calculated as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (32)$$

$$Precision = \frac{TP}{TP+FP} \quad (33)$$

$$Recall = \frac{TP}{TP+FN} \quad (34)$$

$$F1 - Score = 2 \times \frac{(Precision \times Recall)}{Precision+Recall} \quad (35)$$

Where:

- Accuracy measures the proportion of instances that are correctly classified
- Precision measures how much positive instances are correctly predicted out of all predicted positive instances
- Recall measures how much positive instances are correctly predicted out of all actual positive instances
- F1-Score measures the balance between precision and recall scores

The higher these metrics values are identified, the better the model is performing. The combination of these values for each class representation will be aggregated, generating a classification report.

Additionally, following the F1-Score, the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) value is also a significant metric to be evaluated in machine learning (ML) applications (Simon Orozco-Arias et al., 2020). The Receiver Operating Characteristics (ROC) graph is usually used to visualise the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold levels, and indicating how well can the model distinguishes the target classes (e.g., upward vs. downward price movement). A model that can perfectly distinguish the two classes would produce a curve that goes to the top-left corner and an AUC score of 1.0, while a value of 0.5 indicates that the model classifies the classes randomly (50-50). In this case, the True Positive Rate (TPR) against the False Positive Rate (FPR) are calculated as:

$$TPR = Recall \tag{36}$$

$$FPR = \frac{FP}{FP+TN} \tag{37}$$

To further calculate the AUC-ROC value, the mathematical formula be used to approximate it as follows:

$$AUC \approx \sum_{i=1}^{n-1}(FPR_{i+1} - FPR_i) \times \frac{TPR_{i+1}+TPR_i}{2} \tag{38}$$

Where n indicates the number of thresholds on the ROC curve.

A higher AUC score indicates a better model. Alongside the F1-Score, these metrics will provide a comprehensive understanding about both the model's performance on the dataset and its ability to justify the predictions.

3.6. Feature Interpretation

To ensure model transparency and interpretability, this study employs Shapley Additive exPlanations (SHAP) for interpreting each feature's contribution towards the XGBoost classifier's prediction. SHAP can decompose gradient-boosted tree-based model such as XGBoost that is perceived as complex, thus providing insight into the model's performance and how it can be improved. SHAP uses the concept of Shapley values, namely the Shapley regression values (Lipovetsky and Conklin, 2001), Shapley sampling values (Štrumbelj and Kononenko, 2014), and Quantitative Input Influence (Datta, Sen and Zick, 2016) to assign an importance value to each feature based on its contribution towards the model's prediction relative to a baseline expectation. In this case, all possible features combinations are considered, and the marginal contribution of each feature is averaged over all permutations (Lundberg and Lee, 2017). The SHAP analysis in this study is carried out in 3 steps:

1. **Feature Importance Summary Plot**: A summary plot will be generated that tanks the features by their average absolute SHAP values, thus quantifying the global feature importance. This plot also gives insight regarding how different ranges of features values will impact the model's predictions, allowing users to

identify the relationship between the most influential features' strength and the predictions' direction.

2. **SHAP Value Distribution and Feature Impact**: Beyond average importance, this plot will examine the SHAP values distribution to understand the variability of each feature's contribution across individual predictions. This plot reveals whether certain features push the predictions direction towards a particular class (e.g., upward price movement) or whether their effects depend on other factors. This is important to understand the nonlinear effects of the features towards the model's prediction.
3. **Local Explanation via SHAP Force Plot**: This plot explains how each feature (e.g., opening price, sentiment score) either increase or decrease the model's confidence on its final prediction. A baseline value, which is what the model would predict without any information, will be presented along with how each feature will push the direction of the prediction to which side of the class. This is important to understand how the model makes a specific choice, justifying its predictions.

Chapter 4: Research Design and Implementation

4.1. Research Design and Flow

This research follows a quantitative, experimental approach, where the performance of XGBoost classifier trained on traditional stock market indicators is compared to an enhanced model with the addition of sentiment analysis features. This study is structured as follows:

1. Data collection (historical stock data, financial news, social media sentiment)
2. Data Preprocessing
3. Feature engineering and sentiment extraction using NLP techniques (FinBERT).
4. Model training and evaluation of both baseline and sentiment-enhanced models
5. Apply feature engineering techniques when either or both tweets and news do not manage to increase the model performance
6. Compare and document all model variations (configurations) and create a conclusions

Figure 10 below showed the chronological step that was taken to achieve the research objectives. Since the study is quantitative, the first step was to gather the datasets needed for the model development. The datasets used in this study, as explained in part 3.2, were the historical quantitative stock price data, tweets data from Kaggle, and Deepseek-prompted news summary data of Tesla stock (TSLA). After the datasets were loaded into the programming environment, each of the individual datasets were lightly cleaned. Feature expansion were being done for all 3 data, including technical analysis for historical quantitative data and FinBERT embedding and sentiment classification for both text data. Since the data was used to train a classifier model, train-test splitting is an essential process to make sure that the model was generalised and avoid underfitting/overfitting. As mentioned in chapter 1, this study used the XGBoost model. After the modelling process was completed, the performance of different model configurations were compared. Necessary feature scaling and engineering processes were also done in the case of underperforming sentiment data. The result of the experiments will be concluded in the last chapter of this study.

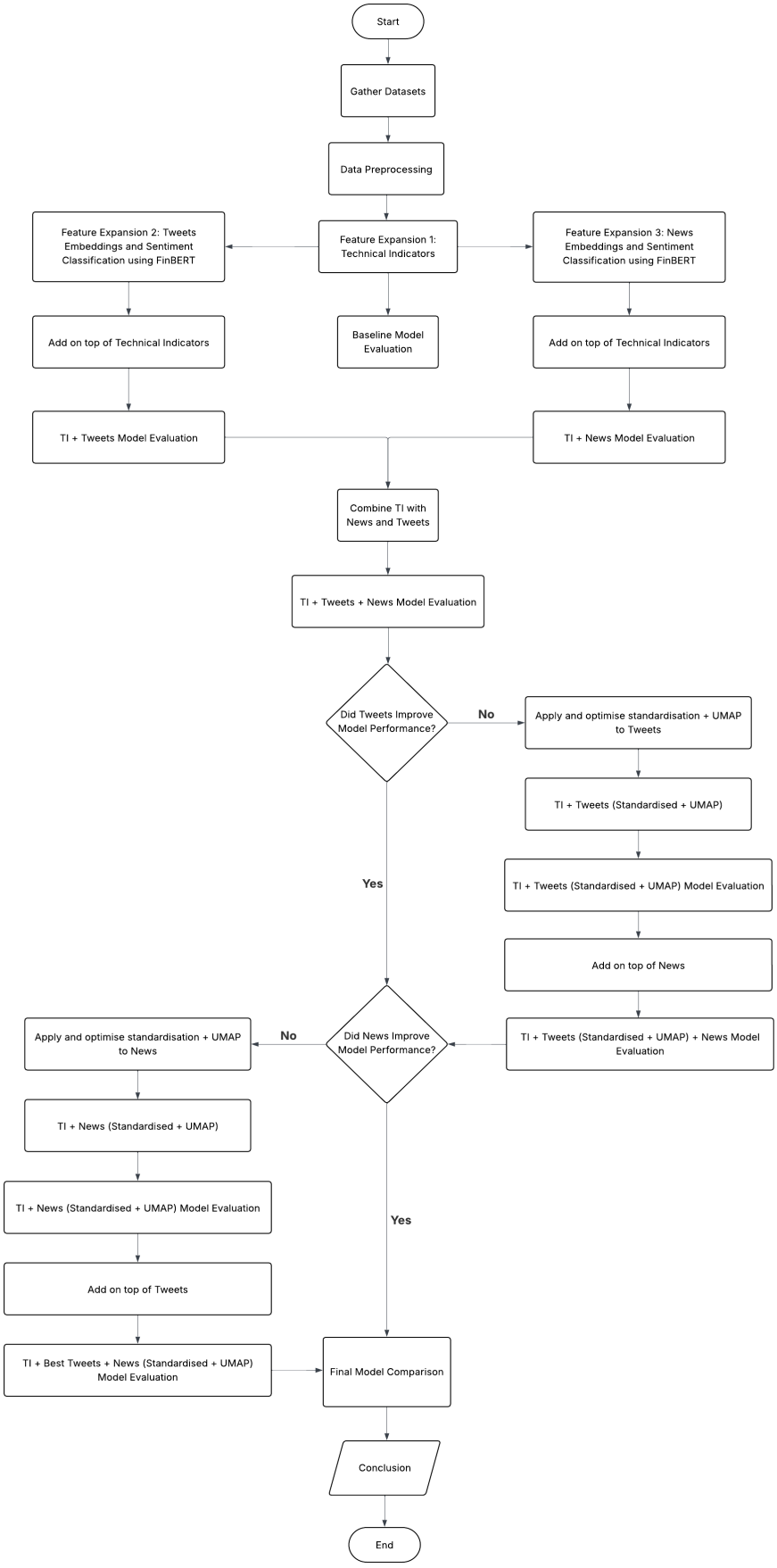


Figure 10. Research Flow Diagram

4.2. Implementation Setup

In this implementation process, some components were considered to complete the research properly. After careful considerations, the below components were the ones that were selected to be utilised throughout this research:

Laptop Specs:

- System Model and Manufacturer: MSI Bravo B5DD
- Processor: AMD Ryzen 7 5800H with Radeon Graphics, 3201 Mhz, 8 Core(s), 16 Logical Processor(s)
- RAM: 16 GB
- GPU: AMD Radeon RX 5500M

Software Environment:

- Operating system: Windows 11
- Python Version: 3.10.0
- Development Environment: VSCode using Jupyter Notebook extension

Main Libraries and Frameworks:

- pandas: 1.5.3
- numpy: 1.23.5
- matplotlib: 3.7.1
- seaborn: 0.12.2
- scikit-learn: 1.2.1
- nltk: 3.8.1
- gensim: 4.3.3
- yfinance: 0.2.55
- torch (PyTorch): 2.1.0+cpu
- transformers: 4.26.0
- tqdm: 4.65.0
- xgboost: 3.0.0
- shap: 0.47.1
- umap-learn: 0.5.7
- ta (technical analysis): 0.11.0

Model Training Details:

- The model training process was done locally
- The training time differ for each model, model parameters, and data size:
    - 19 minutes and 24.1 seconds for tweets embeddings and sentiment classification using FinBERT
    - 6 minutes and 7.9 seconds for news embeddings and sentiment classification using FinBERT
    - Modelling using XGBoost ranges from 1.9 to 12 seconds
    - No RAM and GPU limitations

Additional Information:

- Most of the processes were made into a def function for better reproducibility

4.3. Data preprocessing

Before performing any feature engineering and modelling, some light preprocessing steps were done to the data. For the historical quantitative data, some columns such as the dividends and stock split indicator were removed. After performing feature expansion on the date to create values such as day, week, month, and year, the original date column was removed as well. In the case of tweets data, since originally each row represents one tweet, there were many duplicated dates. Thus, the tweets were aggregated per row, which each row consists of one date and aggregated tweets for that date. As for the news, data, it no preprocessing step was done. All these preprocessing steps were done to ensure that the data are ready to be modelled. In the later stage, all these data will be aggregated as one.

4.4. Feature Engineering

4.4.1. Technical Indicators

As mentioned in section 3.3.2, some technical indicators were computed and added to the historical quantitative dataset. From the initial 5 column consisting of 4 price and 1 volume data, the dataset was expanded into 52 columns. All computed data were derived from the original columns: date, open, high, low, close, and volume variables.

Although there were only 9 types of technical indicators, the specified window periods range up to 26 days, making the feature set richer for better model performance.

4.4.2. Sentiment Feature Construction

This process serves as the foundation to answer research question 2 in chapter 1. As repeatedly mentioned in the before chapters, the text data, namely the tweets and news, need to be transformed into their respective numerical representations for model compatibility. There are 2 main components that will be produced through this step: feature embeddings and feature sentiment classification, in which all were produced by FinBERT. Note that all components explained below are part of the constructed def function called process_financial_tweets.

Def Function:

def process_financial_tweets(final_data, text_column, max_length_clf, max_length_vec, col_prefix, stock_symbol=None, timestamp_column=None)

Pretrained Model and Source:

- Model Name: yiyanghkust/finbert-tone
- Model Source: Hugging Face Transformers Library (https://huggingface.co/yiyanghkust/finbert-tone)
- Product:
    - Feature embeddings (768 (x2 features) additional columns representing each row of text)
    - Sentiment classification: consists of 3 classes: positive, neutral, and negative
- Use cases:
    - Sentiment classification (with AutoModelForSequenceClassification)
    - Text vectorisation/embedding (with AutoModel and output from hidden layers)

Processing Pipeline Overview:

- Model Initialisation
- Sentiment Classification
- Tweet/News Preprocessing
- Vectorisation (Embedding Extraction)
- Final Feature Construction

Model Initialisation and Setup:

- Device Handling: Automatically detects and uses GPU if available (using torch.cuda.is_available())
- Batch Size: Set to 32 for memory-efficient batch processing
- max_length_clf: For sentiment classification input
- max_length_vec: For feature embedding input

Sentiment Classification:

- Tokenizer: AutoTokenizer from Hugging Face for FinBERT
- Model: AutoModelForSequenceClassification
- Postprocessing:
    - Softmax applied to logits
    - Argmax used to determine predicted sentiment
    - Label mapping: 0 for negative, 1 for neutral, 2 for positive
    - Converted to scale: -1 for negative, 0 for neutral, 1 for positive

Text Preprocessing:

This step was done to standardise tweets and news data format

- Cashtags: e.g., $TSLA → CASHTAG_TSLA (tracked, not removed)
- Mentions: e.g., @elonmusk → MENTION
- Hashtags: e.g., #bullish → bullish (symbol removed, word retained)
- URLs: Replaced with token URL

- Emoji Conversion: Financial emojis (📈, 📉, 💰, etc.) mapped to text (BULLISH, BEARISH, etc.)
- Character Cleaning: Removes emojis/symbols but retains currencies like %, $, €, ¥

Embedding Extraction (Vectorisation):

- Model: AutoModel from Hugging Face for FinBERT
- Technique:
    - Extracts the last 4 hidden layers
    - Applies the attention mask
    - Averages across token embeddings per layer
    - Uses learnable softmax weights across the 4 layers (default: 1, 2, 3, 4)
    - Final vector: weighted sum of all 4 mean layer vectors
- Output: One dense vector (size = hidden dimension) per tweets/news

This method leverages the last 4 hidden layers instead of only using the last one (by default) captures a richer data representation. Averaging across layers were done to help smooth out noise and help building a more generalisable embedding. This process was intended to build a more robust embedding for classification task. The softmax-weighted importance gives more influence to layers that are more informative than the others.

Final Output Features:

- Sentiment class column: {prefix}_sentiment_class (e.g., tweet_sentiment_class)
- Embedding Columns: Named as {prefix}_0, {prefix}_1, ..., {prefix}_N
- Merged into final dataset by removing unnecessary columns like the raw and processed text

Caching and Performance Optimisation:

- Embedding function wrapped with @lru_cache(maxsize=10000) to avoid re-computation of repeated tweets/news
- Batch processing was used for sentiment classification for faster execution

In this def function, there are a crucial parameter that affect the output of the FinBERT model, namely the max_length. The max_length parameter controls how many tokens from each input text are fed into the FinBERT model. When max_length is too short, some important information may not be captured. If the setting is too high, it may either capture unnecessary context or slowing down the execution time without adding any significant value. Since there are 2 models used in this def function, namely the vectorizer and sentiment classifier, 2 max_length parameters needed to be optimised. To accommodate this, the def function was designed so that the users could tweak or optimise both max_length parameters, namely the max_length_vec for the vectorizer and max_length_clf for the classifier.

In this study, after numerous manual experimentations, the researcher decided to use the following configurations:

Tweets Data:

- max_length_vec = 64
- max_length_clf = 256

News Data:

- max_length_vec = 64
- max_length_clf = 512

These settings provide the best and most stable results across different feature and model configurations. For vectorisation, a shorter max_length (64) was sufficient as the goal was to extract general representations of the text instead of capturing the full narrative. For sentiment classification, a longer max_length (256 and 512) was required to ensure the FinBERT model understand the full sentiment direction of the entire text documents.

4.4.3. Feature Scaling and Dimensionality Reduction (UMAP)

This section was done to specifically help answering research question 4 in chapter 1. In this research, while the tweets variable managed to improve the baseline technical indicators-only model performance with only FinBERT embeddings and sentiment classification, the news variable failed to do so. Therefore, as mentioned in section 3.3.4, the combination of feature scaling approach, specifically standardisation and UMAP dimensionality reduction method were applied. After numerous manual experimentations on maximising XGBoost model's performances, below are the parameters for these approaches applied to the news variable:

- Standardisation: Used the default StandardScaler() function from sklearn.preprocessing library
- Dimensionality Reduction: Used the UMAP() function from umap library with the following parameters:
    - n_components = 45
    - n_neighbors = 15
    - random_state = 42

These 4 components were 4 of the most influential parameters of UMAP, where n_components determines the dimensionality (number of columns) of the output, n_neighbors determines how much original structure will be preserved, n_jobs controls the number of CPU threads, and random_state ensures consistent results. While both n_components and n_neighbors were manually optimised through various modelling experimentations, the n_jobs parameter was set to 1 to make the algorithm run on a single core, making it more controlled and less resource intensive. The rest of the model parameters were set as default.

4.5. Model Design

4.5.1. Model Choice: XGBoost

The XGBoost algorithm was chosen for the reasons mentioned in the previous chapters. The XGBoost model used in this research have the default XGBClassifier() parameters by the xgboost library, with an additional randome_state parameter being set to 0. This is to ensure reproducibility and preserve consistent results across various

experimentations. No hyperparameter tuning was applied to the model due to some reasons mentioned at section 2.3.1 and 2.3.3.

4.5.2. Binary Classification Setup

As mentioned in section 3.3.1, the target will be created by using the information from the closing price and opening price parameter, where the price movement target is set to 0 when tomorrow's opening price is less than or equal to today's closing price and set to 1 when tomorrow's opening price is higher than today's closing price. This setup sets a stricter condition towards the up movement as compared to the down movement. This is so that the model will predict up more confidently than down, as what matters fundamentally in practice is to leverage the upward direction of price by buying before the market closes and selling by the time the market opens.

There is an assumption being applied to this research. While the training data uses the closing price as a variable, in practice, it is indeed unrealistic to do so as the closing price will be released once the market is already closed. However, the researcher assumed that the price few minutes before the market close will not be much different from the closing price. Additionally, the XGBoost model was proven to be able to at least, using this device, predict the result in less than 12 seconds. While the FinBERT model took some time to embed and classify sentiment data, the sentiment data can be processed first as relevant news and tweets will not come out too often. Thus, in real-world practice, the input will be the price few seconds or minutes before closing and the sentiment data will be extracted as per the predetermined time.

4.6. Evaluation Strategy

Upon evaluating the data and model, some processes were designed thoroughly. First, the data was being split to ensure no data leakage before any feature engineering and modelling process. The testing (or evaluation) set consists of 2.5% of the whole data, which results in around 30 days' worth of data and the training set worth around 4 years of data. This is due to limited training sample. No cross-validation strategies were employed as the data is time-series, making it unrealistic to be applied as explained in section 3.3.4. All evaluation metrics mentioned in section 3.5 were used.

The use of multiple evaluation metrics allows the researcher to examine whether the model can truly distinguish the classes, not overfit/underfit, and perform well under any given circumstances.

4.7. Explainability Integration

In this research, all SHAP capabilities, reasons, and importances of leveraging these plots as mentioned in section 3.6 will be used and further analysed in chapter 5. This section was done to specifically help answering and explaining research question number 5 in chapter 1.

4.8. Summary

Overall, this chapter explains about the overall details regarding this research's technical and fundamental implementation steps. This chapter aligns with both Chapter 2 and 3, and each of the decision is supported by various valid literature review from different sources. The code for this research will be provided under Appendix C: Python Code. Future researchers who are interested in this topic's technical implementation may look at Appendix C at the end of this document.

Chapter 5: Research Outcomes and Evaluation

5.1. Introduction

This chapter will present and evaluate the results of different combinations of features and data preprocessing approaches towards the XGBoost model's performance. There are 6 configurations being tested in this research:

1. Setup 1: The base line model with technical indicators only
2. Setup 2: Technical indicators + tweets (processed via FinBERT)
3. Setup 3: Technical indicators + news (processed via FinBERT)
4. Setup 4: Technical indicators + news (processed via FinBERT + standardised + UMAP-reduced)
5. Setup 5: Technical indicators + tweets (processed via FinBERT) + news (processed via FinBERT)
6. Setup 6: Technical indicators + tweets (processed via FinBERT) + news (processed via FinBERT + standardised + UMAP-reduced)

For all following sections, only the setup type will be mentioned to prevent overwhelming details in each section and point mentioned.

5.2. Model Configurations Overview

Before diving into individual results, the motivation for all 6 model configurations will be briefly discussed below.

Configuration	Order of Completion	Motivation
Setup 1	1	The first setup to be tried as the baseline model comparison with the other setups. This setup is used to determine whether it is necessary to apply further feature engineering (standardisation + UMAP) to either one or both tweets and news variables.

Setup 2	2	This setup was conducted right after trying setup one. This setup purpose was to answer most of the research questions in chapter 1 by identifying the effect of tweets variable (embeddings and sentiment classification) towards the model performance. Since the tweets were able to significantly improve model performance, no other feature engineering techniques were applied.
Setup 3	3	This setup was experimented after setup 2 and before setup 4 as the news data took longer time to be retrieved. The purpose of this setup is the same as setup 2, albeit using news instead of tweets variable. However, the model performance deteriorated, even worse than the baseline model.
Setup 4	6	This setup was the last to be conducted. This is just an experimental setup, where the news that variable that have been feature engineered were combined with the technical indicators. The performance of this setup was below the baseline model.
Setup 5	4	This setup was conducted right after knowing the impact of both tweets and news variables. This setup's purpose was to identify whether the tweets variable can lift the model's performance up despite the news variable dragging it down. As a result, the model's performance was similar as the baseline model.

Setup 6	5	Since the tweets variable was able to lift the model performance up even with the news variable in, further feature engineering was being applied to only the news variable, with the goal of maximising the full hybrid model performance. The feature engineering techniques managed to significantly improve the model performance, creating the best result in this research.

Table 2. Motivation of Various Model Configurations

5.3. Individual Prediction Results and Interpretation

5.3.1. Setup 1: Technical Indicators (TI) Only

Figure 11 to 15 describes the performance of Setup 1. According to Figure 11, the XGBoost model demonstrates poor predictive power, with most metrics ranging around 0.5 to 0.6, although the model is generally better at predicting the 'Up' class (1) than the 'Down' class (0). This indicates that the model is only slightly better than random guessing, suggesting that the features used to predict the price movement are still too weak to capture the movements signals.

Figure 12 indicates that the model achieved 0.5909 AUC score, which is relatively poor in discriminating the target classes. Practically speaking, the model has only 59% chance of ranking the predicted probability of a randomly chosen 'Up" higher than a randomly chosen 'Down'. This level of AUC suggests that the model struggles to capture meaningful patterns of the predictors in determining the stock price movements. Since the curve lies only ever so slightly higher the diagonal baseline, it confirms that the classifier offers minimal improvement over a non-informative model.

To gain deeper insight on the XGBoost model's internal decision-making process for predicting Tesla stock price movement, SHAP values were employed. Figure 13 captures how the magnitude and direction of the features' influence the model's output. For this model, the top 3 features in terms of SHAP impact were *slow_stoch_d*, *Close_8_pct_change*, and *Close_1_pct_change*. Both high (in red) and low (in blue)

slow_stoch_d values contributed inconsistently to the SHAP value, where some high and low values pushing the predictions towards ‘Up’ and some towards ‘Down’. This indicates that *slow_stoch_d* interacts non-linearly with other features and its standalone impact is ambiguous, possibly due to noise or its dependence on other unconsidered features. On the contrary, both *Close_8_pct_change* and *Close_1_pct_change* are more interpretable, where high values of both variables push the model’s predictions towards ‘Down’ and the low values pushes the predictions towards ‘Up’. *Close_8_pct_change* has more extreme high values, indicating that longer period price increase has a strong downward pull on the predictions, whereas *Close_1_pct_change* has more extreme low values, implying that short-term losses are influential in pushing the model’s predictions toward ‘Up’ class.

Following the Beeswarm plot, Figure 14 measures the overall impact of each feature towards the model’s prediction, where similarly*, slow_stoch_d*, *Close_8_pct_change*, and *Close_1_pct_change* provide strong signals towards price movements, suggesting that momentum-based features offer some values in indicating price movements, although not strong enough as being shown from the results.

Finally, Figure 15 describes that the model output is -3.68, which lies below 0, indicating that the model is more confident when predicting a ‘Down’ movement. This suggest that the model requires stronger signals for it to be confident in predicting ‘Up’. From a trading perspective, this model might be advantageous, as when it predicts an upward movement, it is doing so with high certainty, thus minimising false positives and ensuring that the ‘buy’ signals are based on strong evidence as being shown from the model’s performances being better in predicting ‘Up’ than ‘Down’. However, this Setup could not really demonstrate this suggestion as the model’s performance on average are relatively bad.

```
XGBoost Classifier Performance for Tesla:
----------------------------------------
Test Accuracy: 0.5405

Classification Report:
              precision    recall  f1-score   support

           0       0.35      0.64      0.45        11
           1       0.76      0.50      0.60        26

    accuracy                           0.54        37
   macro avg       0.56      0.57      0.53        37
weighted avg       0.64      0.54      0.56        37


ROC AUC Score: 0.5909
========================================

Encoded Values and Class Names:
Encoded Value: 0, Class Name: Down
Encoded Value: 1, Class Name: Up
========================================
```

Figure 11. Setup 1 Classification Report

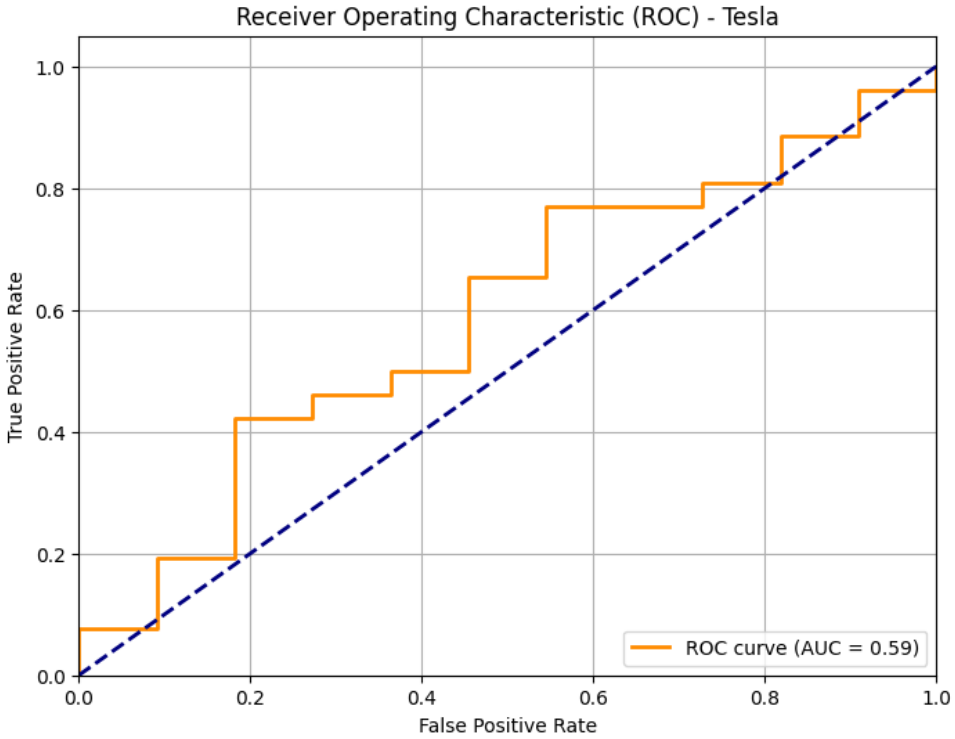


Figure 12. Setup 1 ROC Curve

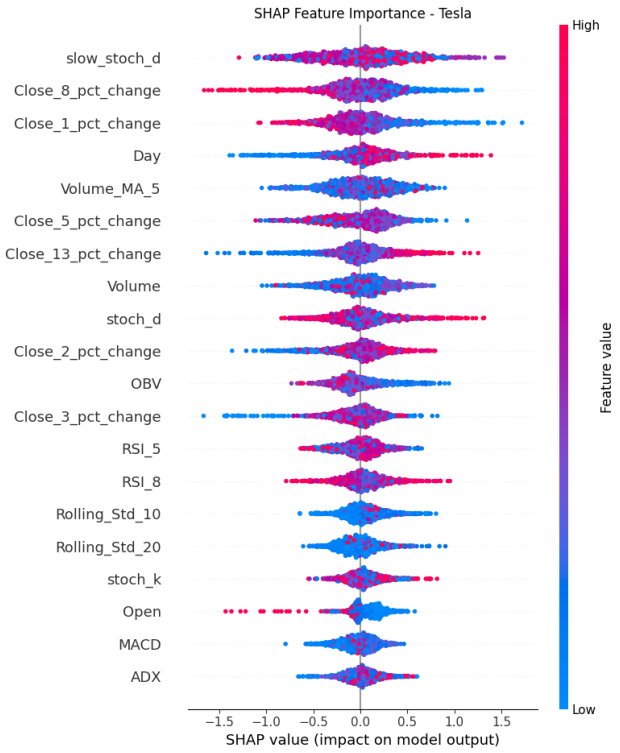


Figure 13. Setup 1 SHAP Summary Plot (Beeswarm)

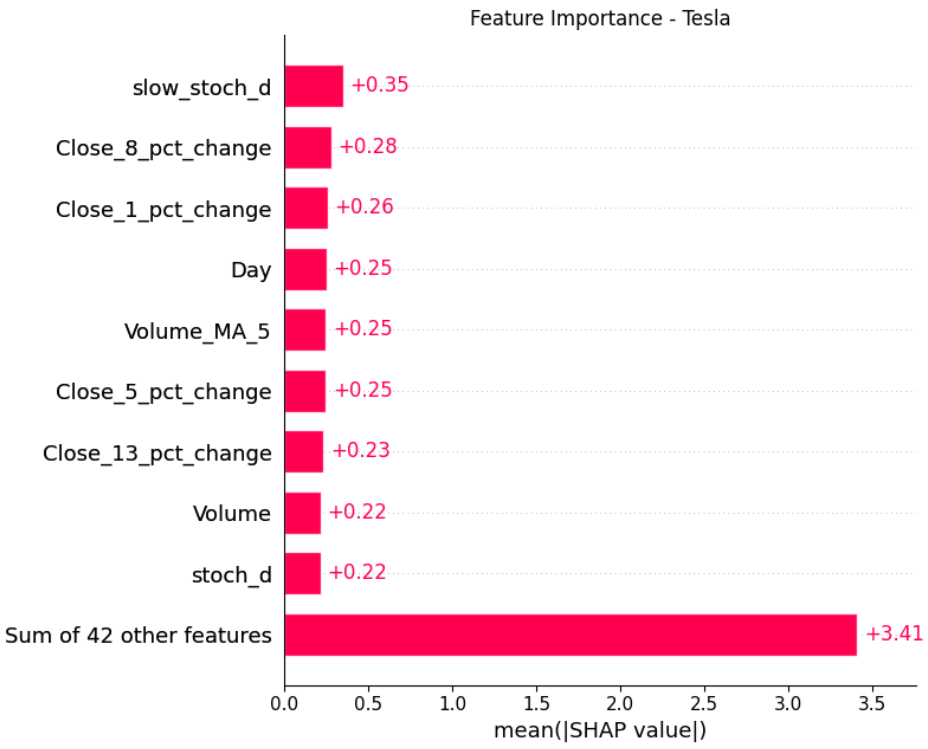


Figure 14. Setup 1 SHAP Feature Importance Plot

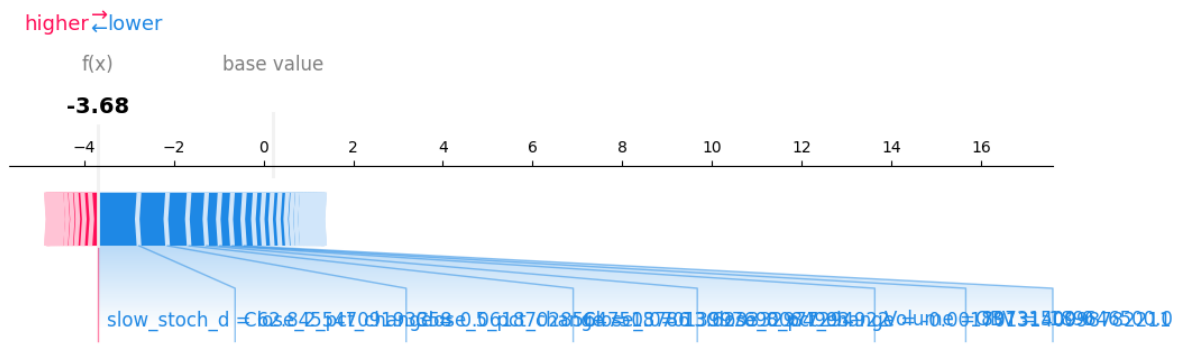


Figure 15. Setup 1 SHAP Force Plot

5.3.2. Setup 2: TI + Tweets

Figure 16 to 20 describes the performance of Setup 2. According to Figure 16, the XGBoost model demonstrates reasonably moderate predictive power. However, the gap in the evaluation metrics results between 'Up' (Class 1) and 'Down' (Class 0) is noticeably large. Figure 16 indicates that the model is significantly better at identifying upward movement than the downward ones, which might happen due to class imbalance or the features not providing enough strong bearish signals. The model might therefore capture distinctive patterns associated with bullish conditions whereas the bearish signals are potentially noisier.

Figure 17 indicates that the model achieved 0.6713 AUC score, which is a moderate performance in discriminating the target classes. Practically speaking, the model has 67% chance of ranking the predicted probability of a randomly chosen 'Up" higher than a randomly chosen 'Down'. This level of AUC suggests that the model can sometimes capture meaningful patterns of the predictors in determining the stock price movements. Since the curve lies mostly higher the diagonal baseline, it confirms that the classifier offers moderate improvement over a non-informative model.

To gain deeper insight on the XGBoost model's internal decision-making process for predicting Tesla stock price movement, SHAP values were employed. Figure 18 captures how the magnitude and direction of the features' influence the model's output. For this model, the top 3 features in terms of SHAP impact were *tweet_519*,

slow_stoch_d, and *OBV*. High *tweet_519* values (in red) generally contributed negatively to the SHAP value, thus pushing the model's prediction toward the 'Down' class (0). On the other hand, low *tweet_519* values (in blue) push the model's prediction toward the 'Up' class (1). This indicates that some words represented by *tweet_519* capture strong signals towards price correction events. The *slow_stoch_d* values' inconsistent impacts towards the model's prediction are similarly present as in Setup 1, indicating that even with tweets variable being considered, there might be other unconsidered features needed to make the *slow_stoch_d* values more clearly and consistently impactful in pushing the model's predictions towards a certain class. *OBV* values impacted the model's predictions similarly as *tweet_519*, with the only difference being *OBV* having slightly more low values and slightly less high values, making it slightly more imbalanced distribution-wise than *tweet_519*. This might indicate that declining volume could lead to bullish reversals whereas higher volumes might indicate overbought, causing downward price movements.

Following the Beeswarm plot, Figure 19 measures the overall impact of each feature towards the model's prediction, where similarly, tweet_519, *slow_stoch_d*, and *OBV* provide strong signals towards price movements, suggesting that the tweets variable offer even higher values when combined with technical indicators in signalling price movements.

Finally, Figure 20 describes that the model output is -4.62, which lies significantly below 0, indicating that the model is much more confident when predicting a 'Down' movement. This suggest that the model requires stronger signals for it to be confident in predicting 'Up'. From a trading perspective, this model might be advantageous, as when it predicts an upward movement, it is doing so with high certainty, thus minimising false positives and ensuring that the 'buy' signals are based on strong evidence as being shown from the model's performances being significantly better in predicting 'Up' than 'Down'.

```
XGBoost Classifier Performance for Tesla:
----------------------------------------
Test Accuracy: 0.6757

Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.27      0.33        11
           1       0.73      0.85      0.79        26

    accuracy                           0.68        37
   macro avg       0.58      0.56      0.56        37
weighted avg       0.64      0.68      0.65        37


ROC AUC Score: 0.6713
========================================

Encoded Values and Class Names:
Encoded Value: 0, Class Name: Down
Encoded Value: 1, Class Name: Up
========================================
```

Figure 16. Setup 2 Classification Report

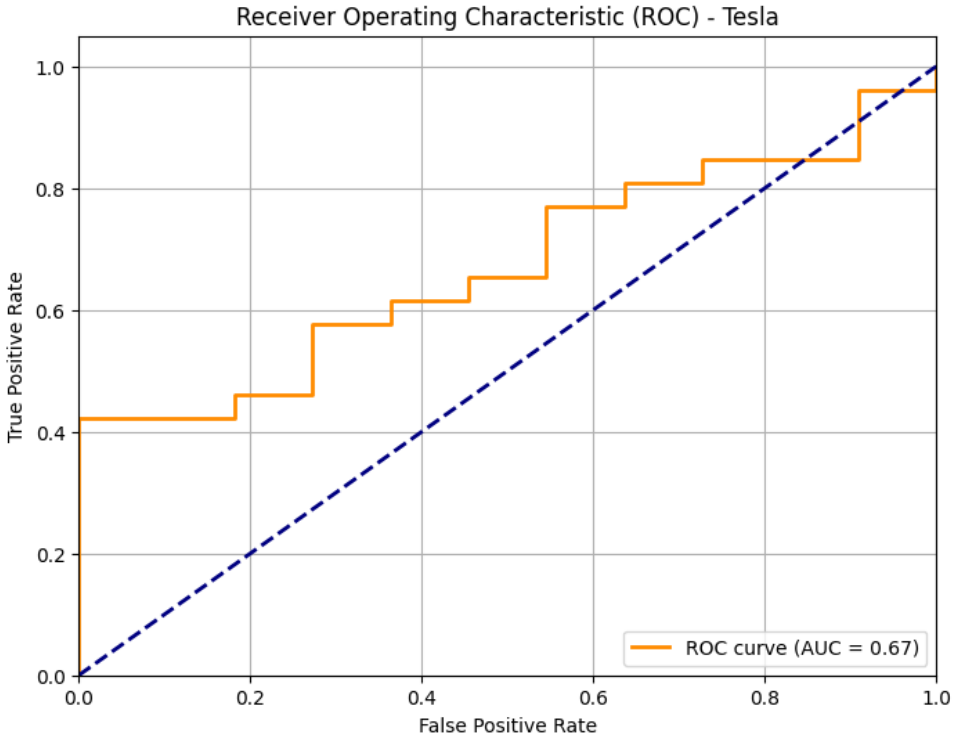


Figure 17. Setup 2 ROC Curve

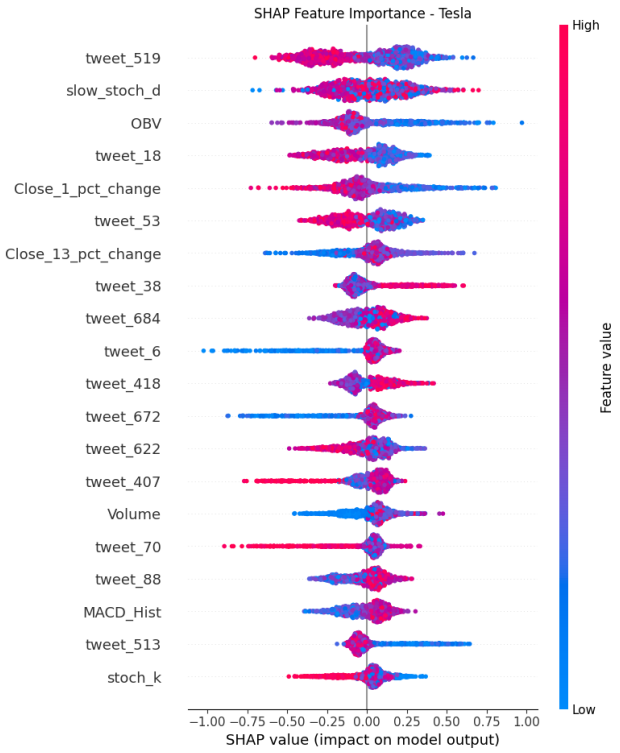


Figure 18. Setup 2 SHAP Summary Plot (Beeswarm)

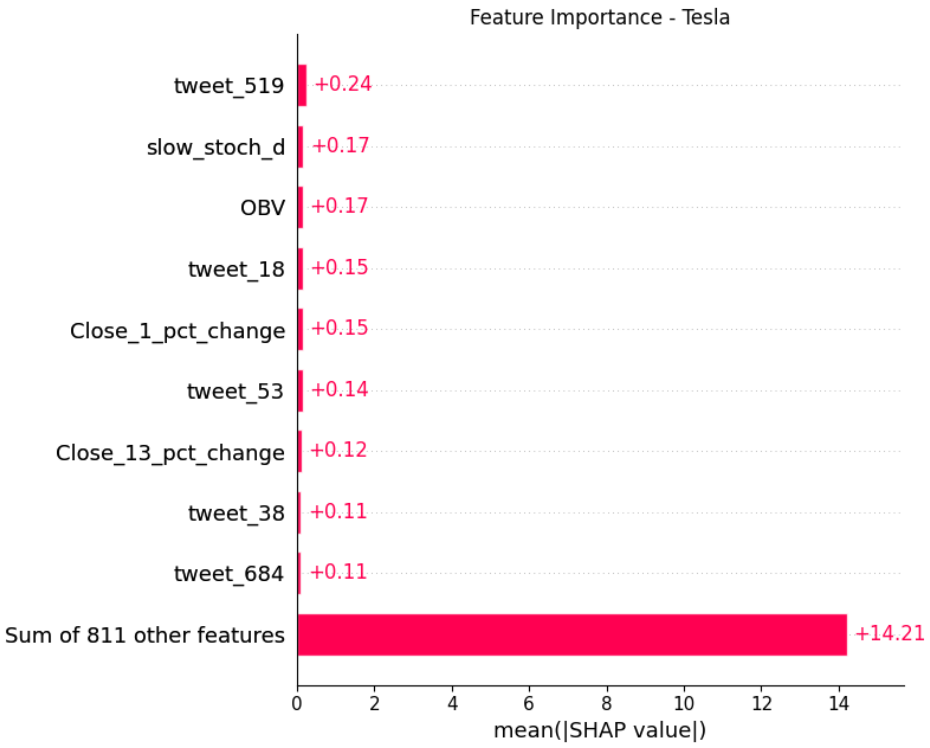


Figure 19. Setup 2 SHAP Feature Importance Plot

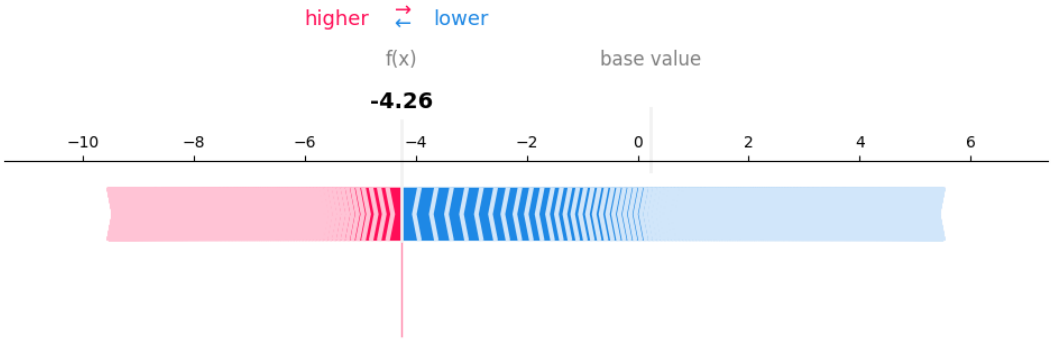


Figure 20. Setup 2 SHAP Force Plot

5.3.3. Setup 3: TI + News

Figure 21 to 25 describes the performance of Setup 3. According to Figure 21, the XGBoost model demonstrates extremely poor predictive performance, with most of the metrics values lie significantly lower than 0.5 and no more than around 0.6 in Class 1. Additionally, the gap in the evaluation metrics results between 'Up' (Class 1) and 'Down' (Class 0) is noticeably large. This indicates that the model is significantly better at identifying upward movement than the downward ones, which might happen due to class imbalance, inadequate feature representations, or the occurrence of features with extreme noises. Thus, the reliability of the model under this Setup is questionable and its ability in practice for stock price movement prediction is extremely limited.

Figure 22 indicates that the model achieved 0.3368 AUC score, which is extremely poor in distinguishing the target classes, even worse than predicting at random. Practically speaking, the model has only 34% chance of ranking the predicted probability of a randomly chosen 'Up" higher than a randomly chosen 'Down'. This level of AUC suggests that the model cannot capture meaningful patterns of the predictors in determining the stock price movements. Since the curve lies almost always under the diagonal baseline, it confirms that the classifier offers worse-than-random performance, where it not only fails to distinguish the price movements but also consistently misclassifying them, which is critically flaw.

To gain deeper insight on the XGBoost model's internal decision-making process for predicting Tesla stock price movement, SHAP values were employed. Figure 23 captures how the magnitude and direction of the features' influence the model's output. For this model, the top features in terms of SHAP impact were majorly dominated by the news variable embeddings and the top features' impacts were relatively inconsistent. Both high (red) and low (blue) feature values are pushing predictions toward both classes without a clear directional trend. This inconsistency indicated that the news embeddings' contributions were unstable and lack interpretable patterns, possibly due to noise or weak signal strength from the news features without any adjustments being made. As a result, the model struggled to form a solid decision boundary, aligning with its overall poor performance in this setup.

Following the Beeswarm plot, Figure 24 measures the overall impact of each feature towards the model's prediction, where it was similarly dominated by the news embeddings, which failed to provide meaningful contributions towards improving the model's performance.

Finally, Figure 25 describes that the model output is -4.62, which lies significantly below 0, indicating that the model is much more confident when predicting a 'Down' movement. This suggest that the model requires stronger signals for it to be confident in predicting 'Up'. Although performing slightly better than random guessing at predicting upward price movements, the model could hardly predict downward trend, making it more likely to result in loss and thus, unworthy to be deployed in real-world scenarios.

```
XGBoost Classifier Performance for Tesla:
----------------------------------------
Test Accuracy: 0.4722

Classification Report:
              precision    recall  f1-score   support

           0       0.11      0.08      0.10        12
           1       0.59      0.67      0.63        24

    accuracy                           0.47        36
   macro avg       0.35      0.38      0.36        36
weighted avg       0.43      0.47      0.45        36


ROC AUC Score: 0.3368
========================================

Encoded Values and Class Names:
Encoded Value: 0, Class Name: Down
Encoded Value: 1, Class Name: Up
========================================
```

Figure 21. Setup 3 Classification Report

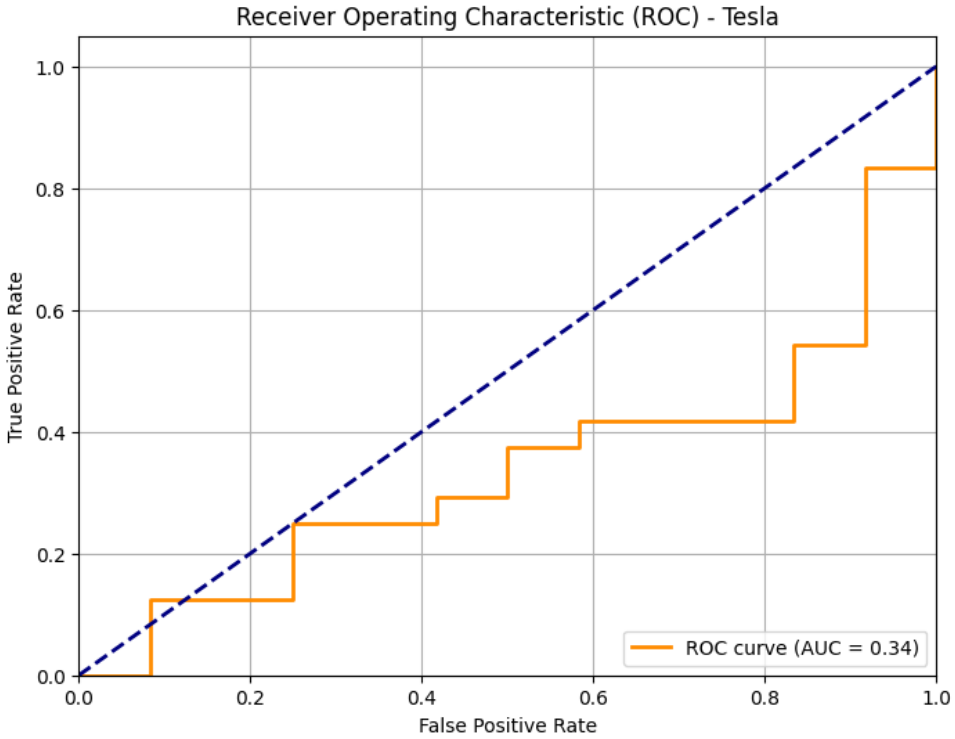


Figure 22. Setup 3 ROC Curve

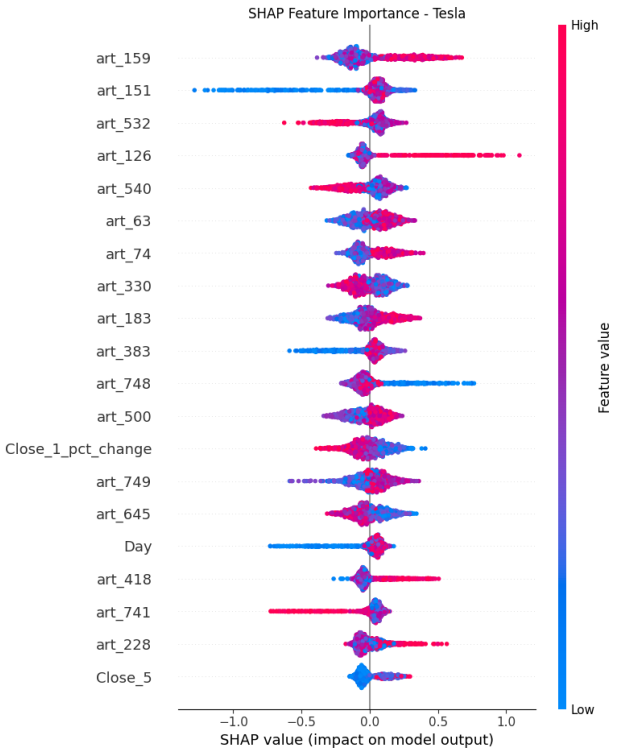


Figure 23. Setup 3 SHAP Summary Plot (Beeswarm)

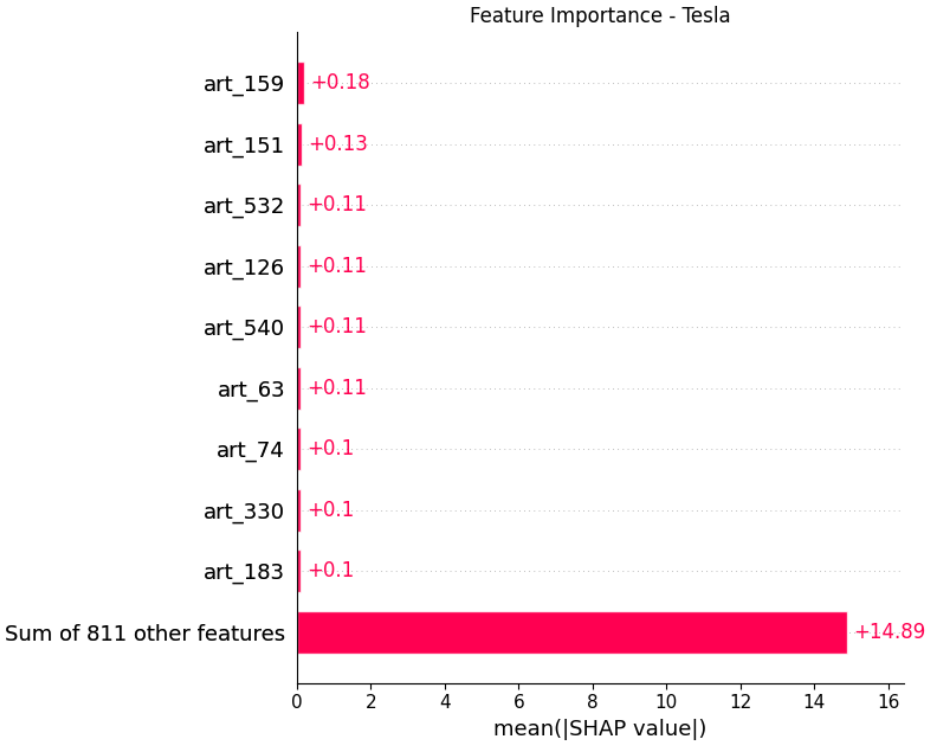


Figure 24. Setup 3 SHAP Feature Importance Plot

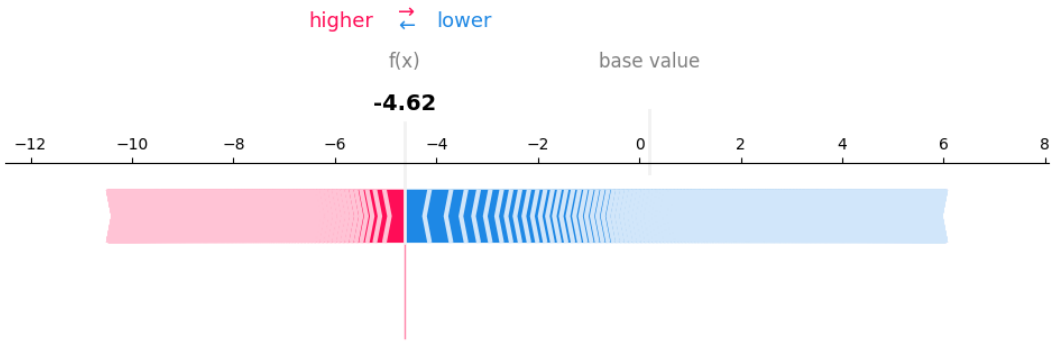


Figure 25. Setup 3 SHAP Force Plot

5.3.4. Setup 4: TI + News (Standardisation + UMAP)

Figure 26 to 30 describes the performance of Setup 4. According to Figure 26, the XGBoost model demonstrates moderate to poor predictive power, with the evaluation metrics of Class 1 ranging up to 0.77 whereas Class 0 evaluation metrics drop as low as 0.18, indicating that the model is better at predicting the 'Up' class (1) as compared to the 'Down' class (0). Such a performance imbalance might indicate some factors such as class imbalance, weaker or noisier feature patterns associated with 'Down' days, or the model learning more towards the majority class. While the model may be beneficial in helping to identify bullish trends, its poor reliability in detecting bearish trends could limit the value of hedging strategies.

Figure 27 indicates that the model achieved 0.521 AUC score, which is relatively poor in discriminating the target classes. Practically speaking, the model has only 52% chance of ranking the predicted probability of a randomly chosen 'Up" higher than a randomly chosen 'Down'. This level of AUC suggests that the model struggles to capture meaningful patterns of the predictors in determining the stock price movements. Since the curve lies only ever so slightly higher the diagonal baseline, it confirms that the classifier offers no better performance over a non-informative model or random guessing.

To gain deeper insight on the XGBoost model's internal decision-making process for predicting Tesla stock price movement, SHAP values were employed. Figure 28

captures how the magnitude and direction of the features' influence the model's output. For this model, the top 3 features in terms of SHAP impact were *Close_1_pct_change*, *slow_stoch_d*, and *Day*. The high *Close_1_pct_change* values (red) push the model's predictions towards 'Down' class (0) whereas the low values push the predictions toward 'Up' class (1), indicating that short-term closing price changes could clearly signal price movements. As for *slow_stoch_d*, the values continue to show inconsistent contributions towards the model's predictions, thus reinforcing the previous observations in Setup 1 and Setup 2 that it may require interactions with other features to improve its impact. Interestingly, the Day variable values provide clear directional influence, where high *Day* values (later in the month) tended to push the predictions towards the 'Up' class, potentially revealing temporal patterns in trading behaviour or new cycle effects.

Following the Beeswarm plot, Figure 29 measures the overall impact of each feature towards the model's prediction, where similarly, *Close_1_pct_change*, *slow_stoch_d*, and *Day* provide strong signals towards price movements, suggesting that integrating standardised and UMAP-reduced news variable is not valuable enough to help XGBoost in capturing meaningful patterns for predicting price direction.

Finally, Figure 30 describes that the model output is -4.01, which lies significantly below 0, indicating that the model is more confident when predicting a 'Down' movement. This suggest that the model requires stronger signals for it to be confident in predicting 'Up'. From a trading perspective, this model might be advantageous, as when it predicts an upward movement, it is doing so with high certainty, thus minimising false positives and ensuring that the 'buy' signals are based on strong evidence. However, like Setup 3, this setup demonstrated an extremely poor performance at predicting the minority class, rendering it ineffective in the case of hedging.

```
XGBoost Classifier Performance for Tesla:
----------------------------------------
Test Accuracy: 0.5946

Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.18      0.21        11
           1       0.69      0.77      0.73        26

    accuracy                           0.59        37
   macro avg       0.47      0.48      0.47        37
weighted avg       0.56      0.59      0.57        37


ROC AUC Score: 0.5210
========================================

Encoded Values and Class Names:
Encoded Value: 0, Class Name: Down
Encoded Value: 1, Class Name: Up
========================================
```

Figure 26. Setup 4 Classification Report

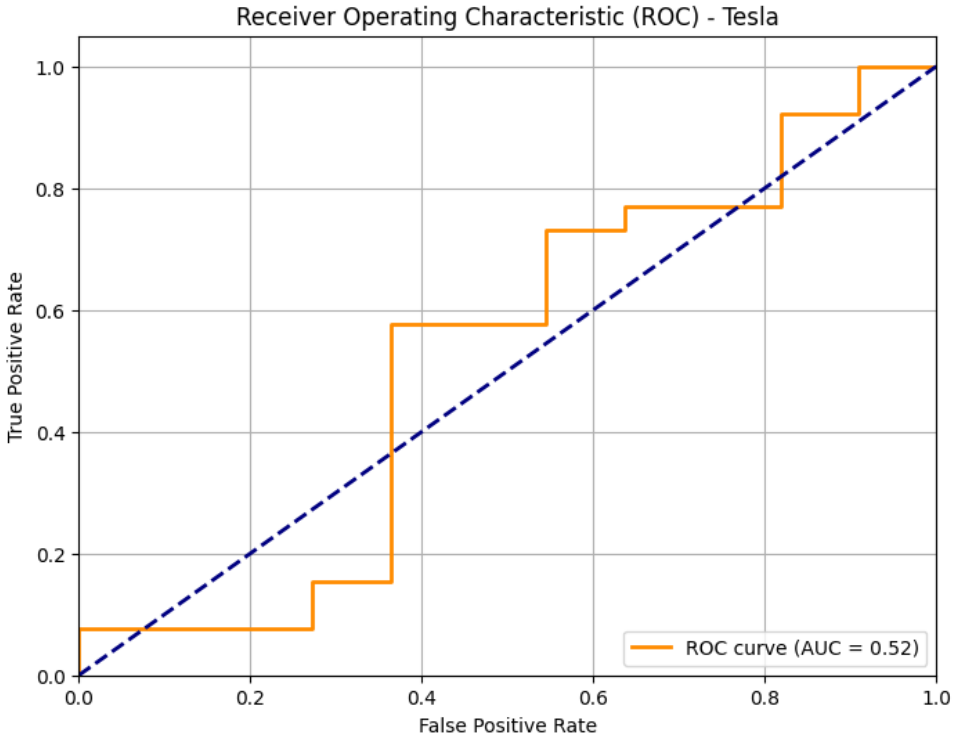


Figure 27. Setup 4 ROC Curve

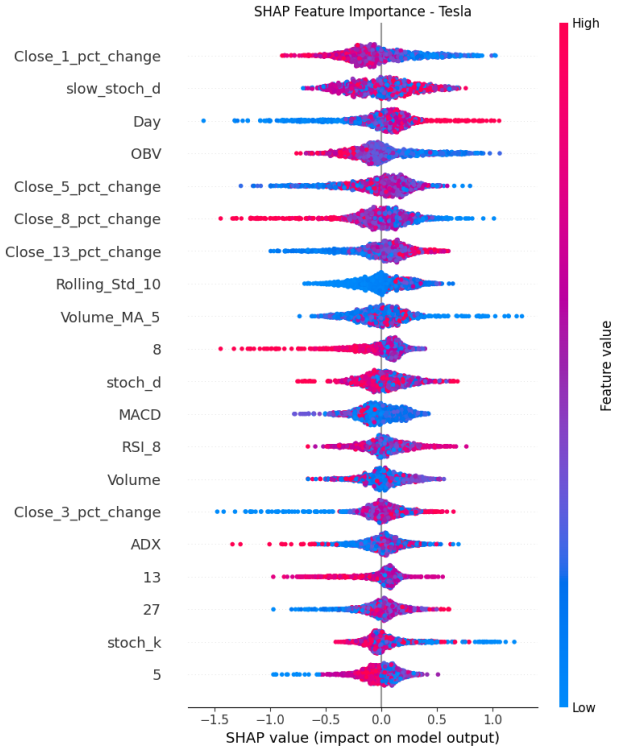


Figure 28. Setup 4 SHAP Summary Plot (Beeswarm)

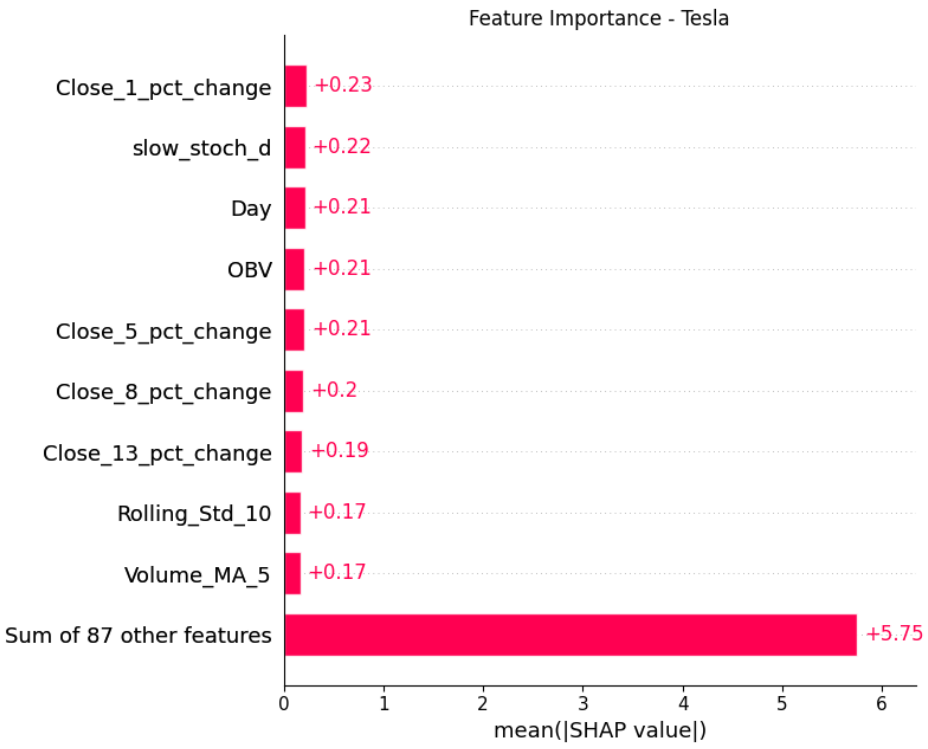


Figure 29. Setup 4 SHAP Feature Importance Plot

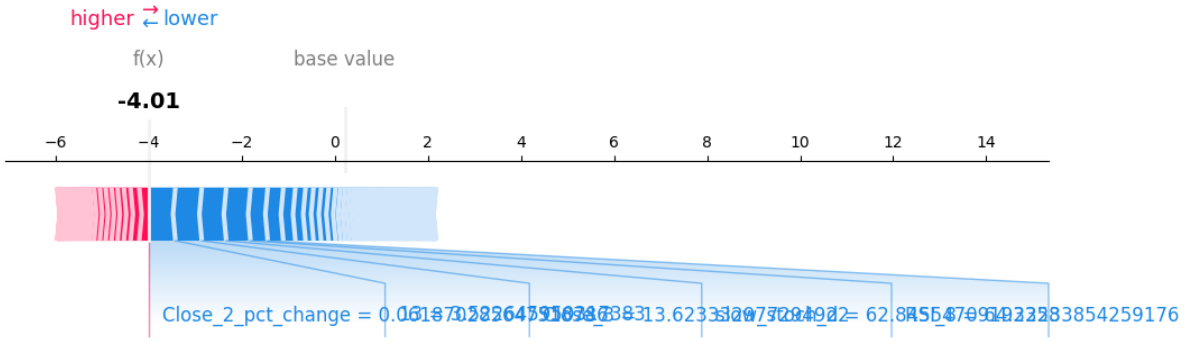


Figure 30. Setup 4 SHAP Force Plot

5.3.5. Setup 5: TI + Tweets + News

Figure 31 to 35 describes the performance of Setup 5. According to Figure 31, the XGBoost model demonstrates moderate to poor predictive power, with the evaluation metrics of Class 1 ranging up to 0.81 whereas Class 0 evaluation metrics drop as low as 0.36, indicating that the model is better at predicting the 'Up' class (1) as compared to the 'Down' class (0). Like Setup 4, such a performance imbalance might indicate some factors such as class imbalance, weaker or noisier feature patterns associated with 'Down' days, or the model learning more towards the majority class. While the model may be beneficial in helping to identify bullish trends, its poor reliability in detecting bearish trends could limit the value of hedging strategies.

Figure 32 indicates that the model achieved 0.5594 AUC score, which is relatively poor in discriminating the target classes. Practically speaking, the model has only 56% chance of ranking the predicted probability of a randomly chosen 'Up" higher than a randomly chosen 'Down'. This level of AUC suggests that the model struggles to capture meaningful patterns of the predictors in determining the stock price movements. Since the curve lies only ever so slightly higher the diagonal baseline, it confirms that the classifier offers slightly better performance over a non-informative model or random guessing.

To gain deeper insight on the XGBoost model's internal decision-making process for predicting Tesla stock price movement, SHAP values were employed. Figure 33

captures how the magnitude and direction of the features' influence the model's output. For this model, the top features in terms of SHAP impact were entirely dominated by the sentiment variables' embeddings, with no technical indicators appearing among the top contributors. Moreover, the top 3 features have more extreme values that pushes the model's predictions strongly and sometimes inconsistently towards both sides, suggesting that the sentiment embeddings capture diverse and potentially conflicting signals, which may reflect the ambiguous nature of the input news and tweet content influencing the stock movement.

Following the Beeswarm plot, Figure 34 measures the overall impact of each feature towards the model's prediction, where similarly, the sentiment embedding variables without any technical indicators provide the strongest signals towards price movements, suggesting that in this setup, the text-based variables alone can dominate the model's decision-making and overshadowing the value of technical indicators. This might be the case where the model might be overfit to sentiment embeddings, making the model fails on the Class 0 as reflected in Figure 31.

Finally, Figure 35 describes that the model output is -4.92, which lies significantly below 0, indicating that the model is more confident when predicting a 'Down' movement. This suggest that the model requires stronger signals for it to be confident in predicting 'Up'. From a trading perspective, this model might be advantageous, as when it predicts an upward movement, it is doing so with high certainty, thus minimising false positives and ensuring that the 'buy' signals are based on strong evidence. However, like Setup 3 and 4, this setup demonstrated an extremely poor performance at predicting the minority class, rendering it ineffective in the case of portfolio hedging.

```
XGBoost Classifier Performance for Tesla:
----------------------------------------
Test Accuracy: 0.6757

Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.36      0.40        11
           1       0.75      0.81      0.78        26

    accuracy                           0.68        37
   macro avg       0.60      0.59      0.59        37
weighted avg       0.66      0.68      0.67        37


ROC AUC Score: 0.5594
========================================

Encoded Values and Class Names:
Encoded Value: 0, Class Name: Down
Encoded Value: 1, Class Name: Up
========================================
```

Figure 31. Setup 5 Classification Report

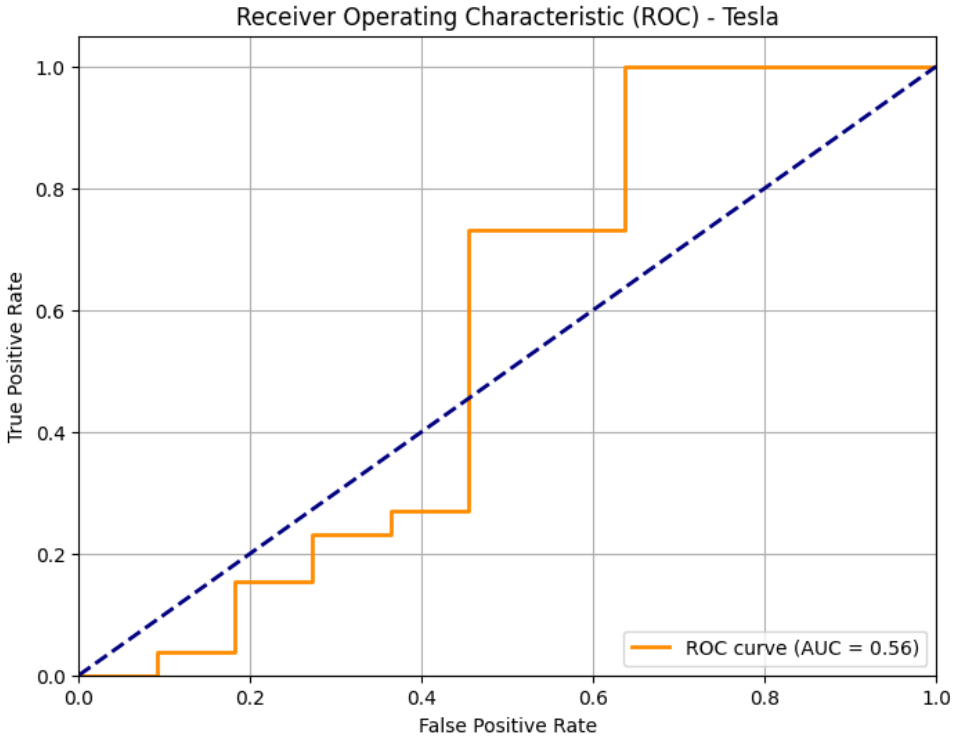


Figure 32. Setup 5 ROC Curve

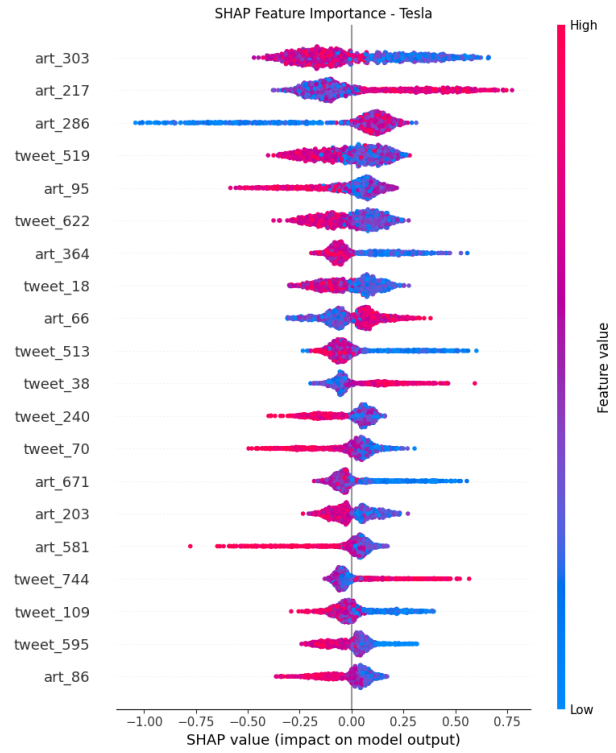


Figure 33. Setup 5 SHAP Summary Plot (Beeswarm)

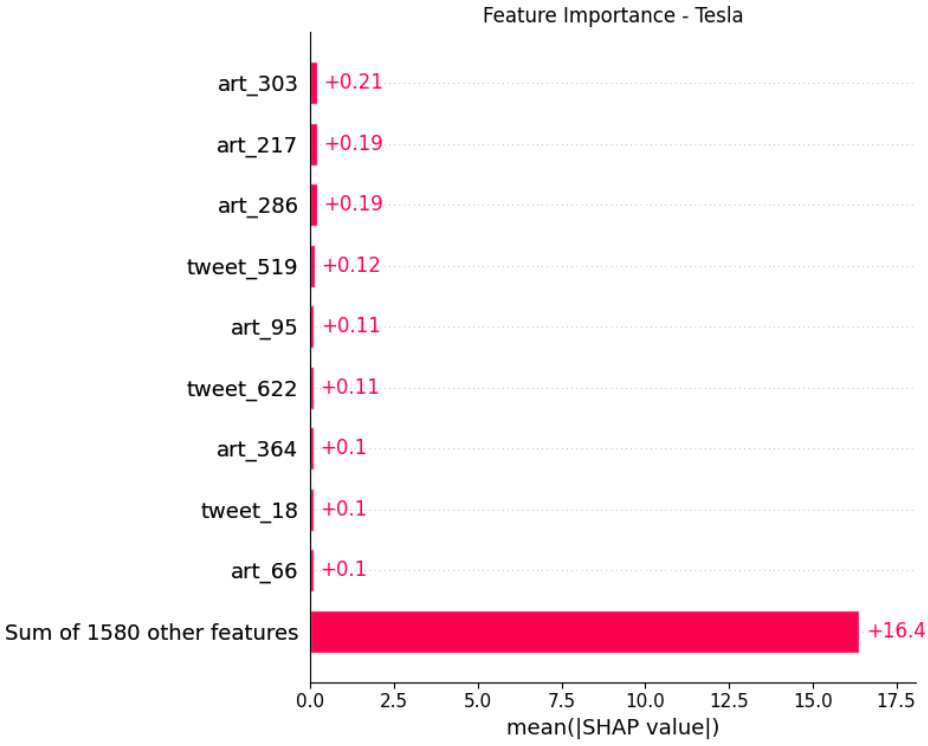


Figure 34. Setup 5 SHAP Feature Importance Plot

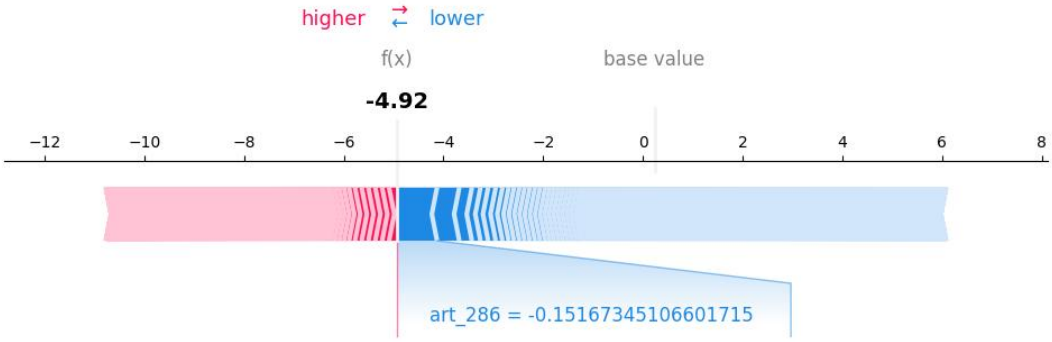


Figure 35. Setup 5 SHAP Force Plot

5.3.6. Setup 6: TI + Tweets + News (Standardisation + UMAP)

Figure 36 to 40 describes the performance of Setup 6. According to Figure 36, 72.97% of the time, the XGBoost model predicted correctly. When the model predicts “Down”, it is correct 56% of the time. Out of all actual “Down”, only 45% were correctly predicted. This results in a 0.5 F1-Score for class “Down”, indicating a low performance on this class. On the other hand, the model shows a strong confidence in predicting “Up” and captures most actual “Up” days, resulting in a high F1-Score of 0.81. From these results, the model can be seen to be biased toward predicting “Up” days, which could happen because of class imbalance or simply because the “Down” days in the testing set contains confusing feature signals. While the model shows exceptional ability in capturing upward trends, the performance of predicting “Down” is relatively poor, which could lead to miss signals for price drops.

Figure 37 indicates that the model successfully achieved 0.73 AUC score, which is moderate but not able to perfectly discriminate between the classes. Practically speaking, the model has a 73% chance of ranking the predicted probability of a randomly chosen ‘Up” higher than a randomly chosen ‘Down’. Although imperfect, this level of AUC suggests that the model could capture the predictors’ meaningful patterns to distinguish between Tesla stock movements. Since the curve lies well above the diagonal baseline, it confirms that the classifier is significantly better than predicting randomly.

To gain deeper insight on the XGBoost model's internal decision-making process for predicting Tesla stock price movement, SHAP values were employed. Figure 38 captures how the magnitude and direction of the features' influence the model's output. For this model, the top 3 features in terms of SHAP impact were *tweet_519*, *OBV*, and *8* (standardised + UMAP news variable). High *tweet_519* values (in red) generally contributed negatively to the SHAP value, thus pushing the model's prediction toward the 'Down' class (0). On the other hand, low *tweet_519* values (in blue) push the model's prediction toward the 'Up' class (1). This indicates that some words represented by tweet_519 capture strong signals towards price correction events. OBV shows a similar pattern, where high *OBV* values tend to push the predictions toward the 'Down' class and low *OBV* values pushes the predictions toward the 'Up' class, although the low values are more extreme and impactful. This might reflect that low-volume trends are followed by bullish momentum or high-volume trends indicate an optimistic market, thus leading to price reversal. Feature *8* (Standardised + UMAP news variable), on the other hand, showed the opposite trend where high feature values push the predictions toward the 'Up' class and low values contributed negatively towards the 'Down' class. This can be interpreted similarly as *tweet_519*, where there are some words that could trigger price movements.

Following the Beeswarm plot, Figure 39 measures the overall impact of each feature towards the model's prediction, where similarly*, tweet_519*, *OBV*, and feature *8* provide strong signals towards price movements, suggesting that multi-modal feature engineering (combining technical indicators, social media, and news variables) is important in determining the stock price movements.

Finally, Figure 40 describes that the model output is -5.01, which lies far below 0, indicating that the model is highly confident when predicting a 'Down' movement. This suggest that the model requires stronger signals for it to be confident in predicting 'Up'. From a trading perspective, this model will be very advantageous, as when it predicts an upward movement, it is doing so with high certainty, thus minimising false positives and ensuring that the 'buy' signals are based on strong evidence, which is supported by the model's strong performance in predicting Class 1.

```
XGBoost Classifier Performance for Tesla:
----------------------------------------
Test Accuracy: 0.7297

Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.45      0.50        11
           1       0.79      0.85      0.81        26

    accuracy                           0.73        37
   macro avg       0.67      0.65      0.66        37
weighted avg       0.72      0.73      0.72        37


ROC AUC Score: 0.7343
========================================

Encoded Values and Class Names:
Encoded Value: 0, Class Name: Down
Encoded Value: 1, Class Name: Up
========================================
```

Figure 36. Setup 6 Classification Report

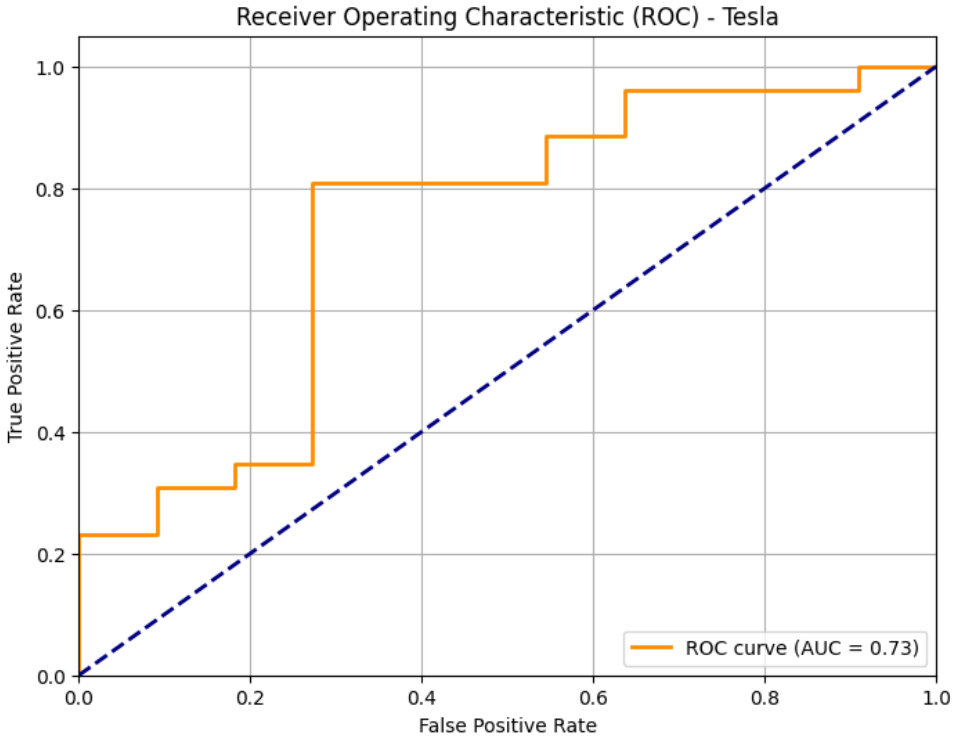


Figure 37. Setup 6 ROC Curve

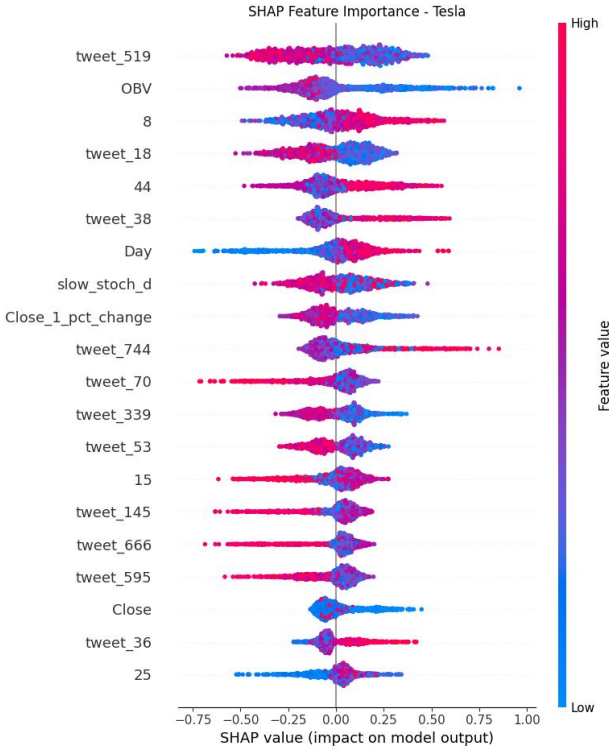


Figure 38. Setup 6 SHAP Summary Plot (Beeswarm)

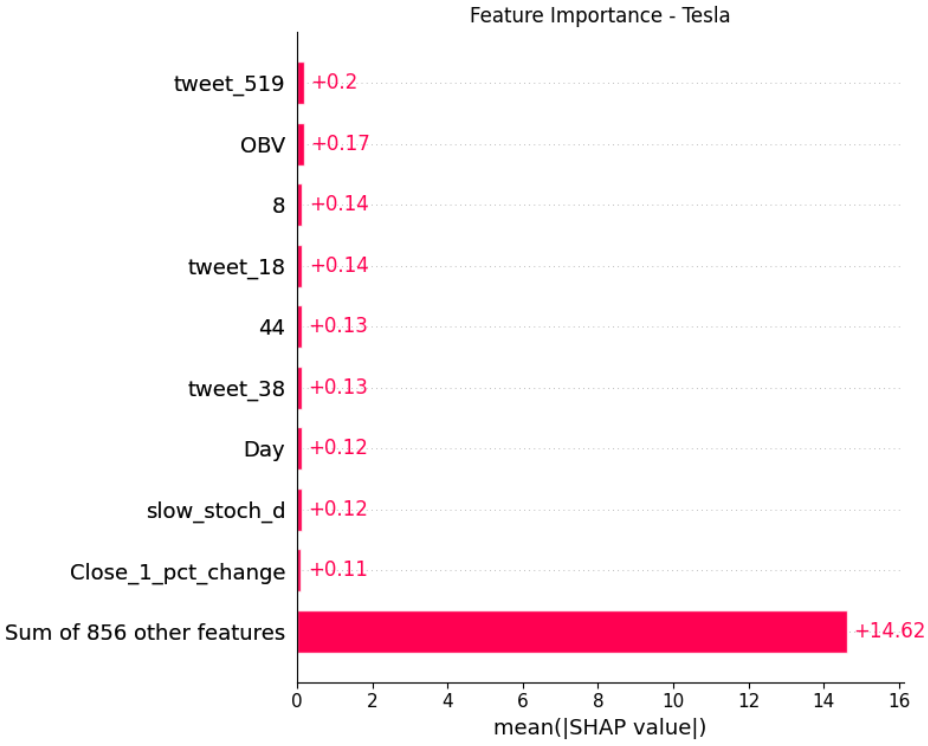


Figure 39. Setup 6 SHAP Feature Importance Plot

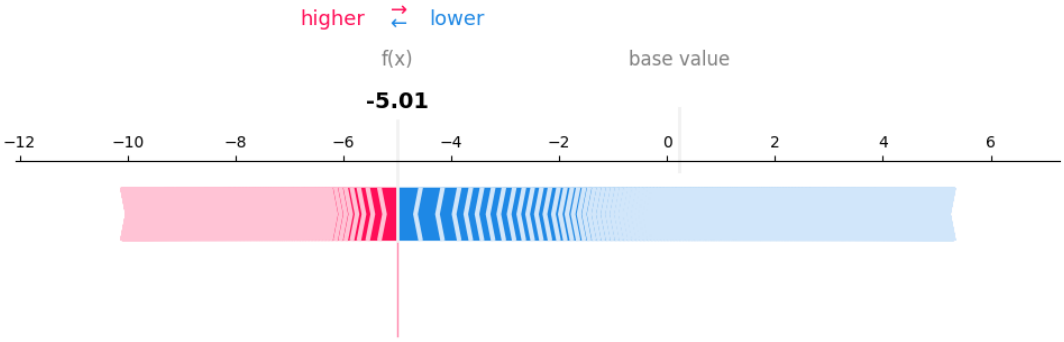


Figure 40. Setup 6 SHAP Force Plot

5.4. Comparative Analysis and Feature Engineering Impact Across All Configurations

This section presents a comprehensive analysis of all model configurations, particularly focusing on the impact of different feature combinations and evaluating the effect of applying standardisation and UMAP dimensionality reduction method to the news variable. The results are summarised in Table 2.

Configuration	Accuracy	Recall	Precision	F1-Score	ROC AUC Score	F1-Score Gap between Classes
Setup 1	54%	54%	64%	56%	59.09%	15%
Setup 2	67.57%	68%	64%	65%	67.13%	46%
Setup 3	47.22%	47%	43%	45%	33.68%	53%
Setup 4	59,46%	59%	56%	57%	52%	52%
Setup 5	67.57%	68%	66%	67%	55.94%	38%
Setup 6	72.97%	73%	72%	72%	73.43%	31%

Table 3. Model Configurations Results

According to table 2, Setup 6 clearly outperforms all other configurations across every evaluation metrics by significant numbers, achieving the highest Accuracy (72.97%), Recall (73%), Precision (72%), and F1-Score (72%). The ROC AUC score of 73.43% also indicates that the model can distinguish between classes well, even approaching the robustness required for production-grade deployment. This performance validates the hypothesis and literatures that state sentiment features with effective preprocessing can significantly enhance short-term stock price movement prediction. As mentioned in section 5.3, the F1-Score gap between classes might exist due to the presence of class imbalance within the testing set. In this case, the baseline model represented by Setup 1 performs the best, indicating that the model can balance the quality of the predictions between classes without significant bias. On the other hand, Setup 3 performs the worst across all metrics, theoretically even worse than manually guessing the price movement by random. This suggests that unprocessed news vectors and sentiment classification may introduce noise, thus degrading the model performance.

To isolate the effect of feature engineering, two direct comparisons are analysed:

Setup 3 Vs. Setup 4 (Both using Technical Indicators + News)

In this comparison, Setup 3 uses raw news feature while Setup 4 applies standardisation and UMAP to the news feature.

Metric	Setup 3	Setup 4	Improvement
Accuracy	47.22%	59.46%	+12.24%
F1-Score	45%	57%	12%
ROC AUC Score	33.68%	52%	+18.32%
F1-Score Gap	53%	52%	+1%

Table 4. Setup 3 Vs. Setup 4 Performance Improvement

According to table 3, after the application of standardisation and UMAP dimensionality reduction, Setup 4 showed a significant improvement across all metrics except for F1-

Score gap. This indicates that UMAP greatly enhances the signal quality of the news data by preserving local and global structures of the original data but reducing the weight of the data which allows the model to separate useful patterns from noise better, leading to higher predictive performance. The slightly reduced gap in F1-Score indicates a modest improvement in balancing the quality of each class's predictions.

Setup 5 vs Setup 6 (Both using Technical Indicators + Tweets + News)

In this comparison, Setup 5 uses raw news feature while Setup 6 applies standardisation and UMAP to the news feature.

Metric	Setup 5	Setup 6	Improvement
Accuracy	67.57%	72.97%	+5.4%
F1-Score	67%	72%	5%
ROC AUC Score	55.94%	73%	+17.49%
F1-Score Gap	38%	31%	7%

Table 5. Setup 5 Vs. Setup 6 Performance Improvement

According to table 4, when tweets are added to the feature set, the model performance significantly increases in both configurations. However, further applying standardisation and UMAP dimensionality reduction to the news variable in Setup 6 elevates the performance even more, especially in terms of ROC AUC score, which improves substantially. The gap between F1-Score also decreases by 7%, indicating better balance in both classes' predictions quality, an important consideration for financial applications where minority class is equally important.

In summary, the comparative analysis indicates that:

- Raw news feature set alone without any transformation degrade model performance due to high dimensionality (weight) and potentially, noise.

- Applying standardisation and UMAP dimensionality reduction method to news feature significantly enhances model performance, particularly when combined with tweets.
- Feature engineering not only improves all performance metrics but also help improving predictions quality across all target classes.
- Tweets variable significantly increases model performance in all configurations.

5.5. Feature Interpretation using SHAP

The following Table 3 shows the top 5 most influential features, ranked, for each configuration by leveraging SHAP values. This is essential for understanding the internal decision-making processes of the model transparently.

Configuration	Top 5 Most Important Features
Setup 1	1. slow_stoch_d 2. Close_8_pct_change 3. Close_1_pct_Change 4. Day 5. Volume_MA_5
Setup 2	1. tweet_519 2. slow_stoch_d 3. OBV 4. tweet_18 5. Close_1_pct_change
Setup 3	1. art_159 2. art_151 3. art_532 4. art_126 5. art_540
Setup 4	1. Close_1_pct_change 2. slow_stoch_d 3. Day 4. OBV 5. Close_5_pct_day
Setup 5	1. art_303 2. art_217 3. art_286 4. tweet_519

	5. art_95
Setup 6	1. tweet_519 2. OBV 3. UMAP_8 4. tweet_18 5. UMAP_44

Table 6. Most Influential Features across All Configurations

Baseline Vs. Tweet Augmentation (Setup 1 Vs. Setup 2)

Setup 1 as the baseline model with technical indicators only identified both momentum and short-term price movements, namely the slow stochastic d and closing price percentage change over 1 day period, to be the dominant features, thus aligning with standard technical analysis assumptions. In setup 2 where tweets sentiment is present, the importance was shifted towards text-derived features (*tweet_519*, *tweet_18*), indicating that the tweet-based sentiment adds predictive information. It is also worth noting that although tweets variable becomes significantly important, some technical indicators such as slow stochastic d and On-Balance Volume remained influential, indicating that the tweets sentiment features are compatible with technical features. As Setup 2 were experiencing model performance improvement, it means that the addition of tweet sentiment successfully enriches the model's understanding without neglecting traditional signals, suggesting that both variables complement each other.

Evaluating Raw Vs. Engineered News Variable (Setup 3 Vs. Setup 4)

Setup 3 integrates the news variable without any additional preprocessing (standardisation and dimensionality reduction). In this setup, the unprocessed article embeddings (*art_159*, *art_151*, etc.) dominated the model, which lack interpretability and offer limited predictive ability as being shown in section 5.4 where Setup 3 had the worst overall model performance. In contrast, Setup 4 that applied standardisation and UMAP on news variable reintroduces meaningful technical indicators such as percentage closing price difference and On-Balance Volume into the top features. Thus, it can be concluded that feature engineering through standardisation and

dimensionality reduction have the capability to improve model performance by combining all complementary features, namely the news variable and technical indicators. This suggests that raw news embeddings and sentiment classification, although provide richer data and additional weight, may introduce noise without proper processing.

Comprehensive Configurations: Tweets + News + Technical Indicators (Setup 5 Vs. Setup 6)

Setup 5 includes all three variables in their raw form without any additional preprocessing. Its top features, as presented in Table 5, are dominated by text-based features, especially from the news variable with only 1 of the tweet embeddings. Setup 6, which applies additional feature engineering to the news variable, shows a much more balanced feature distribution consisting of On-Balance Volume (OBV) from technical indicators, some tweet embeddings (*tweet_519*, *tweet_18*), and some news components (*UMAP_8*, *UMAP_44*). From these facts, it can be concluded that the combination of standardisation and dimensionality reduction via UMAP on news variable supports better feature synergy and model interpretability. The existence of both *OBV* and *tweet_519* across multiple setups proves their reliability as strong technical indicators and sentiment features.

General Observations and Implications

- From the technical indicators side, *OBV* and *Close_1_pct_change* consistently appear across configurations where they are used, indicating that they provide robustness in price movement prediction.
- Tweet-derived features are consistently impactful, especially *tweet_519* which shows its significant contribution towards the model's output even when being used with other variables.
- The raw news-derived features underperform if not processed properly. This can be seen from both Setup 3 and Setup 5 that perform significantly worse with the news-derived features dominating the top features as compared to their

counterparts, Setup 4 and Setup 5, which consider other non-news variables as their most influential features.

- Feature engineering (standardisation + UMAP) allows the news variable to be more effectively combined with other important variables, thus increasing both interpretability and model performance.
- Finally, focusing on the *slow_stoch_d* feature across all setups, the SHAP Summary Plot (Beeswarm) reveals that in Setups 1, 2, 3, and 5, its values contributed inconsistently to the model's predictions, sometimes pushing toward the 'Up' class (1), and other times toward the 'Down' class (0). This inconsistency suggested that *slow_stoch_d* may require interaction with other informative variables to produce a more reliable signal. Interestingly, in Setup 6, where both tweet embeddings and feature-engineered news variables were integrated, the influence of *slow_stoch_d* became noticeably more consistent. High values (red) predominantly pushed the predictions toward the 'Down' class (0), while low values (blue) pushed them toward the 'Up' class (1). This supports the initial assumption that complementary, high-impact features can enhance the effectiveness of individual predictors and, as a result, improve overall model performance, as demonstrated by Setup 6.

5.6. Practical Challenges and Limitations

Throughout this research, there are some challenges and limitations as explained below. This section serves to specifically answer research question number 6 in chapter 1.

1. **Data Accessibility and Validity**

   While it was straightforward to get the historical quantitative stock data such as price and volume, it was very challenging to get both tweets and news data. This is due to the goal of obtaining the most valid, representative, and relevant data for a particular stock. Additionally, some APIs either cannot or too costly to be used to scrape long-term text data, and the researcher must follow certain ethical regulations regarding scraping from reliable websites. Some twitter-scraping APIs are no longer functioning and financial news data were too much to be collected by doing manual copy and paste.

2. **Limitations of Tweets Dataset**

   After days of extensive research and comparisons, the tweets data were found from a valid source that has been used for another research as well. However, the time frame of the data ranges from 2015 to 2020 and the tweets data are only limited to 5 big companies. Therefore, the other data must follow this timeframe as well for valid research.

3. **News Data Acquisition**

   Since the news data could not be found anywhere, the researcher discussed these challenges with fellow data scientists and reading some research papers. After days of research, the researcher found out that some LLM chatbots, with proper prompting, can retrieve some news summary, as explained in section 3.2.3. The researcher then tried to prompt several LLM chatbots, such as Claude, ChatGPT, and DeepSeek. After encountering several prompting and output formatting challenges, and performing experimentations, the researcher found out that only DeepSeek was able to get the correct data format and timeframe as the researcher want, thus completing the news collection process.

4. **Selection of Technical Indicators**

   Searching for the best technical indicators were challenging, since many other research papers are using different indicators and different window settings. Additionally, not all research papers were doing stock movement prediction. Some of them are doing long-term stock price forecasting, making the technical indicators vary a lot. After weeks of reading, consulting with fellow traders, and experimenting with Python, the currently used technical indicators were finalised.

5. **Time Constraints and Scope Management**

   Since the data collection process took a significantly long time to complete, the researcher must find the most practical and time efficient approaches to execute the project. The researcher decided to perform the research using only the Tesla stock (TSLA) and developed 6 model configurations using XGBoost, since the main goal of the research is not model generalisation nor feature engineering methods or model comparisons, but simply proving and maximising stock sentiment data values.

Chapter 6: Project Management

The research project was carefully structured into a detailed and efficient work schedule as represented in a Gantt chart (Figure 41). This research consists of some key phases: proposal, data collection, feature engineering, model development, evaulation, and final presentation. This project was conducted for approximately 6 months, beginning in December 2024 and concluding in May 2025, with a well-defined breakdown of tasks.

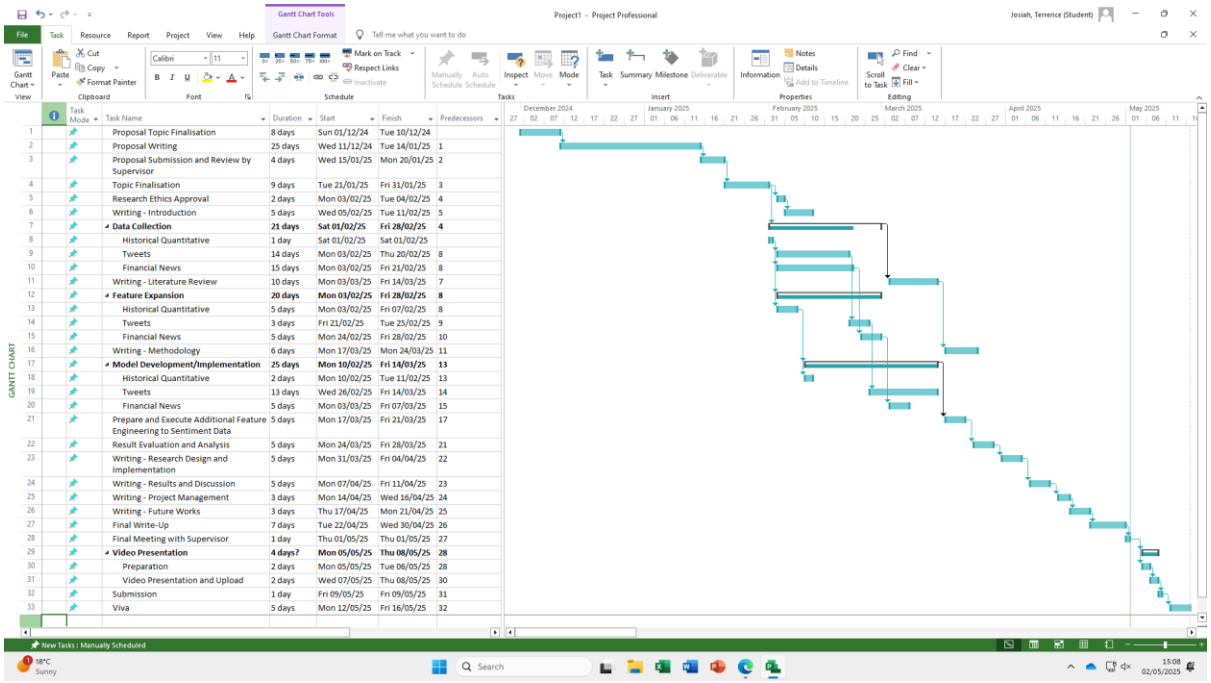


Figure 41. Project Management Timeline

The project began with an 8-day period for proposal topic finalisation and then followed by the supervisor review sessions and ethics approval, which were completed before the data collection process. This early phase ensured that the project adhered to certain ethical standards and the research scope were feasible within institutional guidelines.

The data collection phase, spanning from early to late February 2025, involved the collection of both historical quantitative stock market data and textual data (tweets and

financial news). This phase was followed by literature review and methodology writing, to align theoretical understanding with practical implementation. The literature review was written as the data collection process was still going, as it was far harder to get the textual data.

One of the most critical segments, feature expansion, was allocated 20 days for extracting engineered features from both historical prices and sentiment data. The feature expansion for historical quantitative data were executed while the sentiment data collection process was still ongoing. Right after both sentiment data were obtained, the feature expansion process was further developed and completed. This feature expansion phase was necessary to enhance model performance and enable deeper and richer training data during the classification phase.

The model development and implementation phase took place from 10 February to 14 March, which took the longest time to complete (25 days). This was because there were 6 model configurations and therefore evaluation to be analysed carefully. This stage involves training and evaluating FinBERT for vectorisation and sentiment classification then further integrate all feature combinations into an XGBoost classifier framework.

The next phases focused on refinement, analysis, and validation, including additional feature engineering such as standardisation and dimensionality reduction, and drafting of the results and discussion chapters. These tasks were overlapping with various writing components to reduce bottleneck and keep the project going smoothly.

Finally, video presentation, viva preparation, and submission were crucial to properly end the research project. These phases were scheduled during the final week of the timeline in May 2025. This ensured there was sufficient time for final revisions, formatting, and error handling.

This timeline demonstrates an exceptionally high level of task interdependency and logical sequencing, with extra allocated time for the more crucial phases. This results in a stronger project management application, allowing the researcher to navigate well through tasks complexities such as during the model development process without reducing the quality of the progress.

Chapter 7: Conclusion and Future Work

7.1. Research Summary and Conclusion

This research was conducted with the purpose of developing a hybrid sentiment-aware financial forecasting model by integrating advanced natural language processing (NLP) methods with traditional price-based quantitative indicators. The aim of this study was to assess and maximise the impact of financial sentiment, namely tweets and news, on short-term price movement prediction using a combination of FinBERT for feature vectorisation and sentiment classification and XGBoost for classification.

This study successfully demonstrated that sentiment features, when carefully pre-processed, embedded, and classified using the correct approaches, can serve as impactful features to consider on top of historical quantitative data. The FinBERT model was shown to be effective in capturing contextual financial sentiment, with specific parameters tuning such as separating and optimising max_length for vectorisation and classification, resulting in a significant model performance improvement. Additionally, the combination of standardisation and UMAP dimensionality reduction approach applied to the news variable managed to help the model in adjusting the weight of the news and viewed the news variable from different perspectives, thus improving not only in distinguishing but also balancing the F1-Score gap between the target classes.

The model performance that was evaluated using standard metrics and SHAP explainability tools confirmed that all integrated predictors, namely the historical quantitative, tweets, and news, were important in helping the model decide the future price movement prediction. All in all, this research proposed a realistic and practical real-time pipeline that assumed near-closing prices and sentiment updates can offer relevancy for automated trading applications.

7.2. Critique and Evaluation

While successfully demonstrating how integrating sentiment-aware features could improve predictive performance, it is important to critically evaluate the limitations and assumptions made throughout the process.

First, the use of closing price as a predictor, while have been practically justified through the assumption that prices few seconds or minutes before the market close are almost the same as the closing price, remains an approximation. In real-world trading, even small price fluctuations can affect the decision-making process, and, in some scenarios, sudden extreme price fluctuations may occur, thus rendering the actual deployment of this pipeline ineffective.

In terms of modelling, both FinBERT and UMAP showed sensitivity to their respective parameters. For instance, the max_length parameter of FinBERT and n_components as well as n_neighbors parameters of UMAP, if changed, provided significantly different results. Although this issue was addressed through manual experimentation and tuning, this solution might not generalise across all scenarios, data, and conditions. This mean that reproducibility could be a concern if not tightly controlled. Furthermore, this study obtained both tweets and news data from secondary sources, meaning that sentiment integration, especially for real-time systems, provides challenges in practice due to API limitations and costs, streaming delays, and other noises. Thus, in real-world implementation, critical evaluation needs to be conducted to measure whether integrating the system is worth it compared to the existing operational costs and challenges.

Additionally, although the news data have been manually and carefully prompted and checked, it does not close some possibilities that DeepSeek was hallucinating or used information from future data, leading to partial data leakage and rendering the results partially invalid. Nonetheless, this research still provides strong justification regarding this matter in section 3.2.3 and if some doubts still exist, at the very least, this research still provide some valuable insights into integrating and maximising the benefits of financial sentiment data on top of technical indicators in predicting future stock price movements.

Despite these challenges, this research maintains a high-quality of validity, procedures, and detailed documentation of all modelling steps. Future research may

leverage this research as the baseline procedure and incorporate larger and more fundamentally accurate datasets, alternative transformer-based models, and more thorough deployment methods to improve the project's generalisability and scalability.

7.3. Future Recommendation

There are several future directions that could be pursued to enhance this research methods' effectiveness, scalability, and practical deployment.

Algorithmic Enhancements

Future studies could investigate how deep learning architectures, particularly neural network models such as LSTM, GRU, or Transformer-based classifiers could improve the tree-based classifiers, since they are well-known for their ability to capture long-term dependencies and nonlinear interactions. Additionally, more advanced pretrained financial language models such as BloombergGPT, FiLM, or RoBERTa-Fin may be examined as they offer the possibility to improve sentiment embeddings quality and may provide better contextual understanding over FinBERT.

Additionally, further refinements may be done to the FinBERT vectorisation process by experimenting with dynamic token lengths, batch sizes, and alternative weight aggregation strategies other than softmax such as attention-based layer weighting. When the data is suitable and some requirements are met, hyperparameter tuning could be experimented for better model robustness under varying market conditions. Finally, future work may determine fixed window size (e.g., 4 years) worth of training data and (e.g., 1 month) testing data for each model iteration and evaluation. This will improve the model's robustness and generalisability under various circumstances.

Feature Engineering and Representation

To further improve model performance, future work may explore other advanced feature engineering techniques such as integrating sentiment probabilities (probability for negative, neutral, and positive), exploring feature interactions (polynomial transformation, features ratios), and experimenting different dimensional reduction methods (t-SNE, Kernel PCA). Adding image-based features such as candlestick

charts as predictors and processing them using computer vision models may also enrich the feature space by capturing technical pattern semantics.

Additionally, some unsupervised learning methods such as clustering could further uncover hidden structures in the data. For instance, clustering the sentiment features or even the entire feature space using DBScan or K-Means clustering might add value towards the model's predictive power.

Data Acquisition and Generalisability

Since this research encountered some limitations regarding data availability, particularly for tweets and news, future researchers should prioritise scalable and ethically compliant pipelines to source real-time social and financial news data. This includes finding other reliable sources beyond Twitter/X and general news from the internet such as Reddit, StockTwits, or financial forums.

To enhance model generalisability, the methodology should be applied across multiple stocks and sectors aside from Tesla. Incorporating fundamental data such as financial statements, interest rates, macroeconomic indicators, or company-specific events, especially when these variables are adjusted and engineered, may also lead to more comprehensive predictive frameworks.

Evaluation and Real-Time Deployment

To assess this project's feasibility to be applied on real-world situations, future researchers are encouraged to perform deployment simulations via real-time back testing frameworks. Additionally, it is necessary to analyse the latency across data preprocessing, inference, and decision-making stages to provide insights on the model's capability for live trading applications.

Lastly, some alternative paradigms, such as multi-class sentiment scoring (e.g., low/high confidence up/down), probabilistic outputs, and even treating this problem as regression or unsupervised learning problems, could provide more insightful signals for decision-making systems.

References

Ahluwalia, A. and Wani, S. (2024) “Leveraging Large Language Models for Web Scraping,” *arXiv preprint arXiv:2406.08246*.

Anderson, O.D. (1991) “Interpreting Measured Serial Correlation in Univariate Time Series Analysis, with an Example from the New York Stock Exchange,” *INFOR*, 29(1), p. 44.

Araci, D. (2019) “Finbert: Financial sentiment analysis with pre-trained language models,” *arXiv preprint arXiv:1908.10063*.

Batabyal, D. et al. (2023) “Exploring stationarity and fractality in stock market time-series,” *2023 international conference on intelligent systems, advanced computing and communication (ISACC)* (pp. 1-6). IEEE.

Baek, H. and Lee, E.-B. (2024) “Feature Expansion Effect Approach for Improving Stock Price Prediction Performance,” *Computational Economics*, pp. 1–26. Available at: https://doi.org/10.1007/s10614-024-10787-y.

Bartram, S.M. and Grinblatt, M. (2018) “Agnostic fundamental analysis works,” *Journal of Financial Economics*, 128(1), p. 125.

Bartram, S.M. and Grinblatt, M. (2021) “Global market inefficiencies,” *Journal of Financial Economics*, 139(1), pp. 234–259. Available at: https://doi.org/10.1016/j.jfineco.2020.07.011.

Basak, S. et al. (2019) “Predicting the direction of stock market prices using tree-based classifiers,” *North American Journal of Economics and Finance*, 47, pp. 552–567. Available at: https://doi.org/10.1016/j.najef.2018.06.013.

Becket, M. (2021) *How the stock market works : a beginner’s guide to investment*. Seventh edition. London: Kogan Page.

Bhattacharjee, I. and Bhattacharja, P. (2019) “Stock price prediction: a comparative study between traditional statistical approach and machine learning approach,” *2019 4th international conference on electrical information and communication technology (EICT)* (pp. 1-6). IEEE.

Bi, X. et al. (2024) “Deepseek llm: Scaling open-source language models with longtermism,” *arXiv preprint arXiv:2401.02954*.

Billah, M.M. et al. (2024) “Stock price prediction: comparison of different moving average techniques using deep learning model,” *Neural Computing and Applications*, 36(11), pp. 5861–5871. Available at: https://doi.org/10.1007/s00521-023-09369-0.

Box, G.E. et al. (2015) *Time series analysis: forecasting and control*. John Wiley & Sons.

Bremer, M. and Sweeney, R.J. (1991) “The Reversal of Large Stock-Price Decreases,” *The Journal of Finance*, 46(2), pp. 747–754. Available at: https://doi.org/10.2307/2328846.

Bukosabino, D. (2023) *ta: Technical Analysis Library in Python* (Version 0.10.2) [software]. Available at: https://github.com/bukosabino/ta [Accessed 28 Apr. 2025].

Castelvecchi, D. (2016) “Can we open the black box of AI?,” *Nature*, 538(7623), p. NA. Available at: https://doi.org/10.1038/538020a.

Chandar, S.K. (2024) “Deep learning framework for stock price prediction using long short-term memory,” *Soft Computing : A Fusion of Foundations, Methodologies and Applications*, 28(17), pp. 10557–10567. Available at: https://doi.org/10.1007/s00500-024-09836-3.

Charles, A. and Darné, O. (2019) “Volatility estimation for Bitcoin: Replication and robustness,” *International Economics*, 157, pp. 23–32. Available at: https://doi.org/10.1016/j.inteco.2018.06.004.

Chaudhuri, A. et al. (2018) “Fractality and stationarity analysis on stock market,” *2018 international conference on advances in computing, communication control and networking (ICACCCN)* (pp. 395-398). IEEE.

Chen, Y.-J., Chen, Y.-M. and Lu, C.L. (2017) “Enhancement of stock market forecasting using an improved fundamental analysis-based approach,” *Soft Computing : A Fusion of Foundations, Methodologies and Applications*, 21(13), pp. 3735–3757. Available at: https://doi.org/10.1007/s00500-016-2028-y.

Chen, T. and Guestrin, C. (2016) “Xgboost: A scalable tree boosting system,” *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

Chen, Y. and Hao, Y. (2017) “A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction,” *Expert Systems With Applications*, 80, pp. 340–355. Available at: https://doi.org/10.1016/j.eswa.2017.02.044.

Chollet, F. (2021) *Deep learning with Python*. Second edition. Shelter Island, NY: Manning Publications Co. Available at: https://ieeexplore.ieee.org/book/10280481 (Accessed: April 21, 2025).

Chung, H. and Shin, K.-s. (2018) “Genetic Algorithm-Optimized Long Short-Term Memory Network for Stock Market Prediction,” *Sustainability*, 10(10), p. 3765. Available at: https://doi.org/10.3390/su10103765.

Datta, A., Sen, S. and Zick, Y. (2016) “Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems,” *2016 IEEE symposium on security and privacy (SP)* (pp. 598-617). IEEE.

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2019) “Bert: Pre-training of deep bidirectional transformers for language understanding,” *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171-4186).

Dev Shah, Haruna Isah and Farhana Zulkernine (2019) “Stock Market Analysis: A Review and Taxonomy of Prediction Techniques,” *International Journal of Financial Studies*, 2, p. 26. Available at: https://doi.org/10.3390/ijfs7020026.

Dudek, G. et al. (2024) “Forecasting cryptocurrencies volatility using statistical and machine learning methods: A comparative study,” *Applied Soft Computing Journal*, 151. Available at: https://doi.org/10.1016/j.asoc.2023.111132.

Durgapal, A., Vimal, V. and 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON) Dehradun, India 2021 Nov. 11 - 2021 Nov. 13 (2021) “Prediction of Stock Price Using Statistical

and Ensemble learning Models: A Comparative Study," in *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pp. 1–6. Available at: https://doi.org/10.1109/UPCON52273.2021.9667644.

Dutta, A., Bandopadhyay, G. and Sengupta, S. (2012) "Prediction of stock performance in the Indian stock market using logistic regression," *International Journal of Business and Information*, *7*(1), p.105.

dos Santos Pinheiro, L. and Dras, M. (2017) "Stock market prediction with deep learning: A character-based neural language model for event-based trading," *Proceedings of the Australasian Language Technology Association Workshop 2017* (pp. 6-15).

Düker, M.-C. et al. (2025) "Vector AutoRegressive Moving Average Models: A Review," *Wiley interdisciplinary reviews. Computational statistics*, 17(1), p. e70009. Available at: https://doi.org/10.1002/wics.70009.

Diogo V. Carvalho, Eduardo M. Pereira and Jaime S. Cardoso (2019) "Machine Learning Interpretability: A Survey on Methods and Metrics," *Electronics*, 8(8), p. 832. Available at: https://doi.org/10.3390/electronics8080832.

Dougal, C. et al. (2012) "Journalists and the Stock Market," *The Review of Financial Studies*, 25(3), pp. 639–679.

Engelberg, J.E. and Parsons, C.A. (2011) "The Causal Impact of Media in Financial Markets," *The Journal of Finance*, 66(1), p. 67.

Engle, R.F. and Mustafa, C. (1992) "Implied ARCH Models from Options Prices," *Journal of Econometrics*, 52(1-2), p. 289.

Eiamkanitchat, N., Moontuy, T. and Ramingwong, S. (2017) "Fundamental analysis and technical analysis integrated system for stock filtration," *Cluster Computing : The Journal of Networks, Software Tools and Applications*, 20(1), pp. 883–894. Available at: https://doi.org/10.1007/s10586-016-0694-2.

Fama, E.F. (1970) "Efficient Capital Markets: A Review of Theory and Empirical Work," *The Journal of Finance*, 25(2), pp. 383–417. Available at: https://doi.org/10.2307/2325486.

Fama, E.F. and French, K.R. (1988) “Dividend Yields and Expected Stock Returns,” *Journal of Financial Economics*, 22(1), p. 3.

Fang, L. and Peress, J. (2009) “Media Coverage and the Cross-section of Stock Returns,” *The Journal of Finance*, 64(5), p. 2023.

Franses, P.H. and Ghijsels, H. (1999) “Additive outliers, GARCH and forecasting volatility,” *International Journal of Forecasting*, 15(1), pp. 1–9.

Francq, C. and Sucarrat, G. (2023) “Volatility Estimation When the Zero-Process is Nonstationary,” *Journal of Business & Economic Statistics*, 41(1), pp. 53–66. Available at: https://doi.org/10.1080/07350015.2021.1999821.

Fernández-Martínez, J.L. and Fernández-Muñiz, Z. (2020) “The curse of dimensionality in inverse problems,” *Journal of Computational and Applied Mathematics*, 369. Available at: https://doi.org/10.1016/j.cam.2019.112571.

Friedman, J.H. (2002) “Stochastic gradient boosting,” *Computational Statistics and Data Analysis*, 38(4), pp. 367–378. Available at: https://doi.org/10.1016/S0167-9473(01)00065-2.

Géron, A. (2019) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Second edition. Sebastopol, CA: O’Reilly. Available at: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2245240 (Accessed: April 28, 2025).

Ghorbani, A. et al. (2021) “Beyond Importance Scores: Interpreting Tabular ML by Visualizing Feature Semantics,” *Information*, 13(1), p. 15. Available at: https://doi.org/10.3390/info13010015.

Gordon, M.J. and Shapiro, E. (1956) “Capital equipment analysis: the required rate of profit,” *Management science*, *3*(1), pp.102-110.

Gordon, M.J. (1959) “Dividends, Earnings, and Stock Prices,” *The Review of Economics and Statistics*, 41(2), pp. 99–105. Available at: https://doi.org/10.2307/1927792.

Graham, B. and Dodd, D.L. (1934) *Security analysis*. New York: McGraw-Hill.

Hagenau, M., Liebmann, M. and Neumann, D. (2013) “Automated news reading: Stock price prediction based on financial news using context-capturing features,” *Decision Support Systems*, 55(3), pp. 685–697. Available at: https://doi.org/10.1016/j.dss.2013.02.006.

Hamilton, J.D. (1994) *Time series analysis*. Princeton, N.J.: Princeton University Press.

Han, Y., Kim, J. and Enke, D. (2023) “A machine learning trading system for the stock market based on N-period Min-Max labeling using XGBoost,” *Expert Systems With Applications*, 211. Available at: https://doi.org/10.1016/j.eswa.2022.118581.

Henrique, B.M., Sobreiro, V.A. and Kimura, H. (2019) “Literature review: Machine learning techniques applied to financial market prediction,” *Expert Systems With Applications*, 124, pp. 226–251. Available at: https://doi.org/10.1016/j.eswa.2019.01.012.

Hossain, M.M., Mammadov, B. and Vakilzadeh, H. (2021) “Wisdom of the crowd and stock price crash risk: evidence from social media,” *Review of Quantitative Finance and Accounting*, 58(2), pp. 709–742. Available at: https://doi.org/10.1007/s11156-021-01007-x.

Huang, A.H., Wang, H. and Yang, Y. (2023) “FinBERT: A Large Language Model for Extracting Information from Financial Text,” *Contemporary Accounting Research*, 40(2), pp. 806–841. Available at: https://doi.org/10.1111/1911-3846.12832.

Huang, A.H., Zang, A.Y. and Zheng, R. (2014) “Evidence on the Information Content of Text in Analyst Reports,” *The Accounting Review*, 89(6), p. 2151.

Hu, Y. et al. (2015) “Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review,” *Applied Soft Computing Journal*, 36, pp. 534–551. Available at: https://doi.org/10.1016/j.asoc.2015.07.008.

Imam, S., Barker, R. and Clubb, C. (2008) “The Use of Valuation Models by UK Investment Analysts,” *European Accounting Review*, 17(3), pp. 503–535. Available at: https://doi.org/10.1080/09638180802016650.

Ismail, M.S. et al. (2020) “Predicting next day direction of stock price movement using machine learning methods with persistent homology: Evidence from Kuala Lumpur

Stock Exchange," *Applied Soft Computing Journal*, 93. Available at: https://doi.org/10.1016/j.asoc.2020.106422.

Kang, J.-W. and Choi, S.-Y. (2025) "Comparative Investigation of GPT and FinBERT's Sentiment Analysis Performance in News Across Different Sectors," *Electronics*, 14(6), p. 1090. Available at: https://doi.org/10.3390/electronics14061090.

Kim, K. and Lee, J. (2014) "Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction," *Pattern Recognition*, 47(2), pp. 758–768. Available at: https://doi.org/10.1016/j.patcog.2013.07.022.

Kumbure, M.M. et al. (2022) "Machine learning techniques and data for stock market forecasting: A literature review," *Expert Systems With Applications*, 197. Available at: https://doi.org/10.1016/j.eswa.2022.116659.

Kumar, R. and Shrivastav, L.K. (2021) "An Ensemble of Random Forest Gradient Boosting Machine and Deep Learning Methods for Stock Price Prediction," *Journal of Information Technology Research (JITR)*, 15(1), pp. 1–19. Available at: https://doi.org/10.4018/JITR.2022010102.

Lantz, B. (2023) *Machine learning with R : learn data cleansing to modeling from the tidyverse to neural networks and working with big data*. Fourth edition. Birmingham: Packt Publishing. Available at: https://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9781801076050 (Accessed: April 29, 2025).

Lauriola, I., Lavelli, A. and Aiolli, F. (2022) "An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools," *Neurocomputing*, 470, pp. 443–456. Available at: https://doi.org/10.1016/j.neucom.2021.05.103.

Li, Q. et al. (2016) "A Tensor-Based Information Framework for Predicting the Stock Market," *ACM Transactions on Information Systems (TOIS)*, 34(2), pp. 1–30. Available at: https://doi.org/10.1145/2838731.

Li, C. and Maheu, J.M. (2024) "A multivariate GARCH-jump mixture model," *Journal of Forecasting*, 43(1), pp. 182–207. Available at: https://doi.org/10.1002/for.3019.

Li, X. et al. (2014) “News impact on stock price return via sentiment analysis,” *Knowledge-Based Systems*, 69, pp. 14–23. Available at: https://doi.org/10.1016/j.knosys.2014.04.022.

Li, Q. et al. (2021) “A Multimodal Event-Driven LSTM Model for Stock Prediction Using Online News,” *IEEE Transactions on Knowledge and Data Engineering*, 33(10). Available at: https://doi.org/10.1109/TKDE.2020.2968894.

Liu, Y. et al. (2025) “AT-FinGPT: Financial risk prediction via an audio-text large language model,” *Finance Research Letters*, 77. Available at: https://doi.org/10.1016/j.frl.2025.106967.

Lien Minh, D. et al. (2018) “Deep Learning Approach for Short-Term Stock Trends Prediction Based on Two-Stream Gated Recurrent Unit Network,” *IEEE Access*, 6. Available at: https://doi.org/10.1109/ACCESS.2018.2868970.

Lin, X., Yang, Z. and Song, Y. (2011) “Intelligent stock trading system based on improved technical analysis and Echo State Network,” *Expert Systems With Applications*, 38(9), pp. 11347–11354. Available at: https://doi.org/10.1016/j.eswa.2011.03.001.

Lyle, M.R. and Yohn, T.L. (2021) “Fundamental Analysis and Mean-Variance Optimal Portfolios,” *The Accounting Review*, 96(6), p. 303. Available at: https://doi.org/10.2308/TAR-2019-0622.

Lv, P. et al. (2022) “Modal decomposition-based hybrid model for stock index prediction,” *Expert Systems With Applications*, 202. Available at: https://doi.org/10.1016/j.eswa.2022.117252.

Long, W., Song, L. and Tian, Y. (2019) “A new graphic kernel method of stock price trend prediction based on financial news semantic and structural similarity,” *Expert Systems With Applications*, 118, pp. 411–424. Available at: https://doi.org/10.1016/j.eswa.2018.10.008.

LOUGHRAN, T. and MCDONALD, B. (2016) “Textual Analysis in Accounting and Finance: A Survey,” *Journal of Accounting Research*, 54(4), pp. 1187–1230. Available at: https://doi.org/10.1111/1475-679X.12123.

Loughran, T. and McDonald, B. (2011) “When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks,” *The Journal of Finance*, 66(1), p. 35.

Long, W. et al. (2024) “A hybrid model for stock price prediction based on multi-view heterogeneous data,” *Financial Innovation*, 10(1), p. 48. Available at: https://doi.org/10.1186/s40854-023-00519-w.

Lin, C.-T. et al. (2022) “Spatial-temporal attention-based convolutional network with text and numerical information for stock price prediction,” *Neural Computing and Applications*, 34(17), pp. 14387–14395. Available at: https://doi.org/10.1007/s00521-022-07234-0.

Leung, C.K.Y. and Chen, N.-K. (2010) “STOCK PRICE VOLATILITY, NEGATIVE AUTOCORRELATION AND THE CONSUMPTION-WEALTH RATIO: THE CASE OF CONSTANT FUNDAMENTALS,” *Pacific Economic Review*, 15(2), pp. 224–245. Available at: https://doi.org/10.1111/j.1468-0106.2010.00499.x.

Lipovetsky, S. and Conklin, M. (2001) “Analysis of regression in game theory approach,” *Applied Stochastic Models in Business and Industry*, 17(4), pp. 319–330. Available at: https://doi.org/10.1002/asmb.446.

Lundberg, S.M. and Lee, S.I. (2017) “A unified approach to interpreting model predictions,” *Advances in neural information processing systems*, *30*.

Ma, Y., Li, S. and Zhou, M. (2025) “Twitter-based market uncertainty and global stock volatility predictability,” *North American Journal of Economics and Finance*, 75. Available at: https://doi.org/10.1016/j.najef.2024.102256.

Ma, Y. et al. (2023) “Multi-source aggregated classification for stock price movement prediction,” *Information Fusion*, 91, pp. 515–528. Available at: https://doi.org/10.1016/j.inffus.2022.10.025.

Masís, S. (2021) *Interpretable machine learning with Python : learn to build interpretable high-performance models with hands-on real-world examples*. Birmingham: Packt Publishing, Limited. Available at: http://public.eblib.com/choice/PublicFullRecord.aspx?p=6531443 (Accessed: April 22, 2025).

Murphy, J.J. (1999) *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Nguyen, D. et al. (2022) "MFinBERT: Multilingual Pretrained Language Model For Financial Domain," in *2022 14th International Conference on Knowledge and Systems Engineering (KSE)*, pp. 1–6. Available at: https://doi.org/10.1109/KSE56063.2022.9953749.

Nti, I.K., Adekoya, A.F. and Weyori, B.A. (2019) "A systematic review of fundamental and technical analysis of stock market predictions," *Artificial Intelligence Review : An International Science and Engineering Journal*, 53(4), pp. 3007–3057. Available at: https://doi.org/10.1007/s10462-019-09754-z.

Pesaran, M.H. and Timmermann, A. (1995) "Predictability of stock returns: Robustness and economic significance," *The Journal of Finance*, 50(4), p. 1201.

Peters, M.E. et al. (2018) "Dissecting contextual word embeddings: Architecture and representation," *arXiv preprint arXiv:1808.08949*.

Radford, A. et al. (2018) "Improving language understanding by generative pre-training"

Raman, R. et al. (2022) "Mixed-methods research in the age of analytics, an exemplar leveraging sentiments from news articles to predict firm performance," *International Journal of Information Management*, 64. Available at: https://doi.org/10.1016/j.ijinfomgt.2021.102451.

Rahman, M.M., Ara, L.A. and Zheng, Z. (2009) "JUMP, NON-NORMAL ERROR DISTRIBUTION AND STOCK PRICE VOLATILITY - A NONPARAMETRIC SPECIFICATION TEST," *The Singapore Economic Review*, 54(1), p. 101.

Sarantis, N. (2001) "Nonlinearities, cyclical behaviour and predictability in stock markets: International evidence," *International Journal of Forecasting*, 17(3), pp. 459–482.

Saunders, M.N.K., Lewis, P. and Thornhill, A. (2023) *Research methods for business students*. Ninth edition. Harlow, England: Pearson.

Schmid, L. et al. (2024) “Comparing Statistical and Machine Learning Methods for Time Series Forecasting in Data-Driven Logistics—A Simulation Study,” *Entropy*, 27(1). Available at: https://doi.org/10.3390/e27010025.

Shen, Y., Zhang, P.K. and 2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS) Shenyang, China 2024 July 26 - 2024 July 28 (2024) “Financial Sentiment Analysis on News and Reports Using Large Language Models and FinBERT,” in *2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 717–721. Available at: https://doi.org/10.1109/ICPICS62053.2024.10796670.

Simon Orozco-Arias et al. (2020) “Measuring Performance Metrics of Machine Learning Algorithms for Detecting and Classifying Transposable Elements,” *Processes*, 8(638), p. 638. Available at: https://doi.org/10.3390/pr8060638.

Štrumbelj, E. and Kononenko, I. (2014) “Explaining prediction models and individual predictions with feature contributions,” *Knowledge and Information Systems : An International Journal*, 41(3), pp. 647–665. Available at: https://doi.org/10.1007/s10115-013-0679-x.

Song, Y., Lee, J.W. and Lee, J. (2019) “A study on novel filtering and relationship between input-features and target-vectors in a deep learning model for stock price prediction,” *Applied Intelligence*, 49(3), pp. 897–911. Available at: https://doi.org/10.1007/s10489-018-1308-x.

Takashi MATSUBARA, Ryo AKITA and Kuniaki UEHARA (2018) “Stock Price Prediction by Deep Neural Generative Model of News Articles,” *IEICE Transactions on Information and Systems*, 101.D(4), pp. 901–908. Available at: https://doi.org/10.1587/transinf.2016IIP0016.

Thesia, Y., Oza, V. and Thakkar, P. (2021) “A dynamic scenario-driven technique for stock price prediction and trading,” *Journal of Forecasting*, 41(3), pp. 653–674. Available at: https://doi.org/10.1002/for.2848.

Troy J Strader et al. (2020) “Machine Learning Stock Market Prediction Studies: Review and Research Directions,” *Journal of International Technology and Information Management*, 28(4), pp. 63–83.

Usmani, S. and Shamsi, J.A. (2023) “LSTM based stock prediction using weighted and categorized financial news,” *PLoS ONE*, 18(3), p. e0282234. Available at: https://doi.org/10.1371/journal.pone.0282234.

Vermeulen, M. et al. (2021) “Application of Uniform Manifold Approximation and Projection (UMAP) in spectral imaging of artworks,” *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 252. Available at: https://doi.org/10.1016/j.saa.2021.119547.

Wang, Q., Xu, W. and Zheng, H. (2018) “Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles,” *Neurocomputing*, 299, pp. 51–61. Available at: https://doi.org/10.1016/j.neucom.2018.02.095.

Walkshäusl, C. (2021) “Predicting stock returns from the pricing and mispricing of accounting fundamentals,” *Quarterly Review of Economics and Finance*, 81, pp. 253–260. Available at: https://doi.org/10.1016/j.qref.2021.06.011.

Wu, Q., Zhang, H.-C. and Chiu, Y.-J. (2023) “Application of asymmetric proximal support vector regression based on multitask learning in the stock market,” *Expert Systems With Applications*, 227. Available at: https://doi.org/10.1016/j.eswa.2023.120208.

Wu, Y. (2011) “Momentum trading, mean reversal and overreaction in Chinese stock market,” *Review of Quantitative Finance and Accounting*, 37(3), pp. 301–323. Available at: https://doi.org/10.1007/s11156-010-0206-z.

Yun, K.K., Yoon, S.W. and Won, D. (2021) “Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process,” *Expert Systems With Applications*, 186. Available at: https://doi.org/10.1016/j.eswa.2021.115716.

Yu, P. and Yan, X. (2019) “Stock price prediction based on deep neural networks,” *Neural Computing and Applications*, 32(6), pp. 1609–1628. Available at: https://doi.org/10.1007/s00521-019-04212-x.

Yun, K.K., Yoon, S.W. and Won, D. (2023) “Interpretable stock price forecasting model using genetic algorithm-machine learning regressions and best feature subset

selection," *Expert Systems With Applications*, 213. Available at: https://doi.org/10.1016/j.eswa.2022.118803.

Yang, N., Fernandez-Perez, A. and Indriawan, I. (2024) "Spillover between investor sentiment and volatility: The role of social media," *International Review of Financial Analysis*, 96. Available at: https://doi.org/10.1016/j.irfa.2024.103643.

Zhang, Y. et al. (2022) "Opening the black box: interpretable machine learning for predictor finding of metabolic syndrome," *BMC Endocrine Disorders*, 22(1). Available at: https://doi.org/10.1186/s12902-022-01121-4.

Zhang, Y., Jin, R. and Zhou, Z.-H. (2010) "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, 1(1-4), pp. 43–52. Available at: https://doi.org/10.1007/s13042-010-0001-0.

Zhou, F. et al. (2019) "EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction," *Expert Systems With Applications*, 115, pp. 136–151. Available at: https://doi.org/10.1016/j.eswa.2018.07.065.

Appendices

Appendix A: The “Use of AI Chatbot” Statement

I declare that I have not use any AI chatbot or similar tools to assist me with the writing, analysis, or content development of this research. The only instance in which I used a chatbot was solely and primarily to extract historical news data from the years 2015 to 2020. This is for the purpose of further analysis and the reasoning and details of this data extraction process are explained in Section 3.2.3 of this document.

Appendix B: Prove of Ethical Form Approval

Research Ethics Online

LEEDS BECKETT UNIVERSITY

TERRENCE JOSIAH
ID: 77471560
Course: Data Science (Master of Science)
Email: T.Josiah7327@student.leedsbeckett.ac.uk

Supervisor: Mullier, Duncan
LREC: Chawawa, Margaret

Enhancing Stock Market Prediction: Evaluating the Impact of Financial News and Social Media Sentiment on XGBoost Performance

WILL YOUR RESEARCH STUDY.....?

Please answer the following:

1 Involve direct and/or indirect contact with human participants? No

2 Involve analysis of pre-existing data which contains personal or sensitive information not in the public domain? No

3 Require permission or consent to conduct? No

4 Require permission or consent to publish? No

5 Have a risk of compromising confidentiality? No

6 Have a risk of compromising anonymity? No

7 Collect / contain sensitive personal data? No

8 Contain elements which you OR your supervisor are NOT trained to conduct? No

9 Use any information OTHER than that which is freely available in the public domain? No

10 Involve respondents to the internet or other visual/vocal methods where participants may be identified? No

11 Include a financial incentive to participate in the research? No

12 Involve your own students, colleagues or employees? No

13 Take place outside of the country where you are enrolled as a student, or for staff, outside of the UK? No

14 Involve participants who are particularly vulnerable or at risk? No

15 Involve participants who are unable to give informed consent? No

16 Involve data collection taking place BEFORE informed consent is given? No

17 Involve any deliberate deception or covert data collection? No

18 Involve a risk to the researcher or participants beyond that experienced in everyday life? No

19 Cause (or could cause) physical or psychological harm or negative consequences? No

20 Use intrusive or invasive procedures? No

21 Involve a clinical trial? No

22 Involve the possibility of incidental findings related to health status? No

23 Fit into any of the following security-sensitive categories: concerns terrorist or extreme groups; commissioned by the military; commissioned under an EU security call; involve the acquisition of security clearances? No
If you believe you may need to answer yes to this question, please check here for further guidance before finalising your decision.

RISK CATEGORY 1

Student Applicants

Your study has been provisionally classified as Risk Category 1. This means that your Research Supervisor (or Director of Studies for research students) can normally approve this project.

You must complete the remainder of this application, which will automatically be sent to your Research Supervisor upon submission, who will review your application and decide whether to grant ethical approval, request revisions or reject the application. For security-sensitive research, see the Research Ethics Procedures for details of the approval process.

You will be notified of the outcome by email. You can also view the outcome on the 'My Applications' page of this system.

PROJECT SUMMARY

Start date of project

15th January 2025

Expected completion date of project

1st May 2025

Externally Funding

Is this project externally funded? No

Project Summary

Please give a brief summary of your study (maximum 100 words).

This study explores the impact of incorporating financial news and social media sentiment into an XGBoost-based stock price movement prediction model. By integrating sentiment data from financial news (using FinBERT) and Twitter posts (processed with Word2Vec) alongside traditional technical indicators like moving averages and RSI, the model's predictive performance is evaluated. Historical stock prices are collected via the Yahoo Finance API, and sentiment scores are extracted through sentiment analysis. The study compares models trained on technical indicators alone versus those incorporating sentiment features, using metrics like accuracy, precision, recall, F1-score, and AUC-ROC to assess performance improvements.

Project Group Members

Is this a group project? No

RISK CATEGORY 1: DECLARATION

Comply with Policy and Procedures

Yes : I confirm that I have read the Research Ethics Policy and relevant sections of the Research Ethics Procedures and will adhere to these in the conduct of this project.

Benefits

Yes : The results of the research should benefit society directly or by generally improving knowledge and understanding. Please tick this box to confirm that your study has a potential benefit.

Note: If you cannot identify a benefit you must discuss your project with your Research Supervisor to help identify one or adapt your proposal so the study will have an identifiable benefit.

Confirmation

Yes : I confirm that I will undertake this project as detailed in the application. I understand that I must abide by the terms of this approval and that I may not make any substantial amendments to the project without further approval.

Learned Societies

I have read an appropriate professional or learned society code of ethical practice: Yes

Where applicable, give the name of the professional or learned society:

IEEE (Institute of Electrical and Electronics Engineers)

SUBMISSION CHECKLIST

Please indicate the supporting documents submitted by ticking the appropriate boxes below:

For projects involving human participants, you must submit, where appropriate, the Participant Information Sheet/consent form. You must also submit every communication a participant will see or receive. Failure to do so will cause delays to the application.

N/A : Participant Information Sheet(s)

N/A : Consent Form(s)

N/A : Assent Form (usually for children participants)

N/A : Recruitment documents eg, posters, flyers, advertisements, email invitations, letters, web pages if online research

N/A : Measures to be used eg, questionnaires, surveys, interview schedules, psychological tests

N/A : Screening questionnaire

N/A : Letters/communications to and from gatekeepers/third parties

N/A : Evidence of any other approvals or permissions eg, NHS research ethics approval, in-country approval

N/A : Research proposal/protocol (no more than 2-3 A4 pages): It is not a requirement that this is included, however, if this would help the understanding of a complex project by the reviewer(s), please include

N/A : Risk assessment from: Some projects may require a risk assessment form: see the Procedures document for details (eg, projects involving a physical intervention, collecting data off-campus)

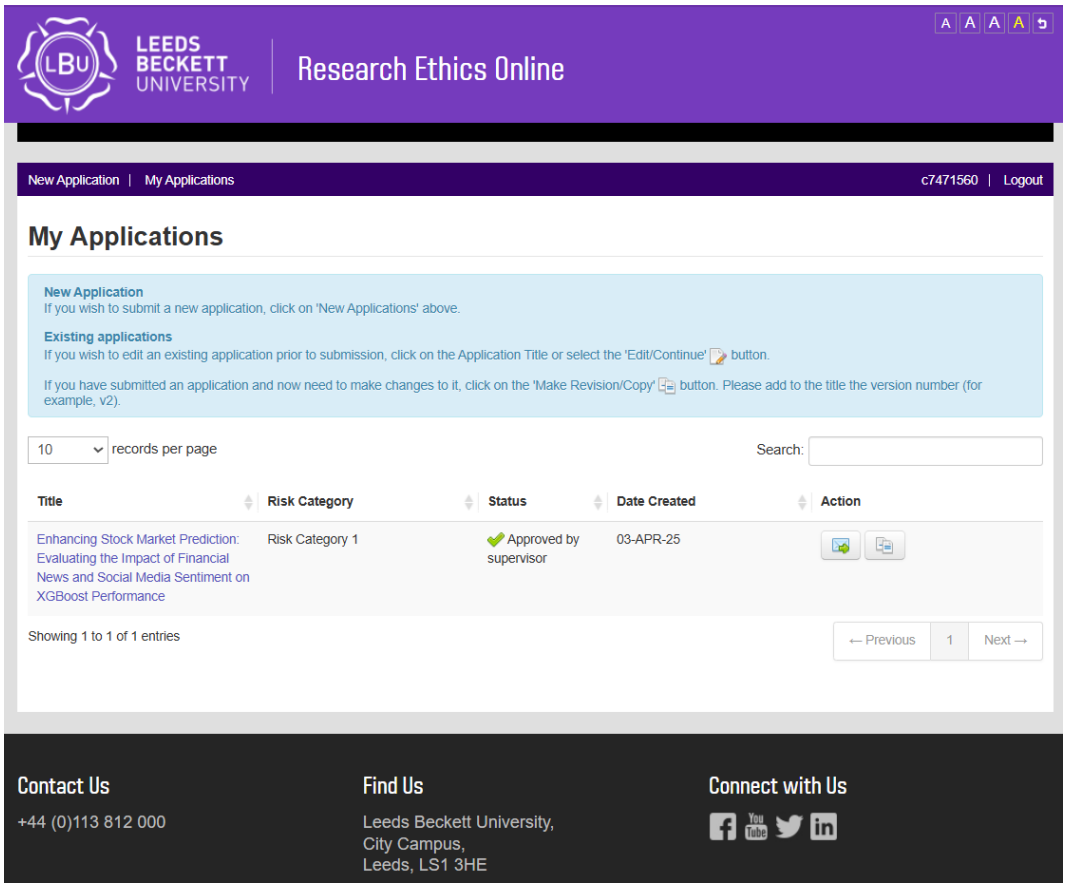
N/A : Approval documentation for projects involving ionising radiation

N/A : Confirmation of insurance and indemnity cover: Some projects need to be referred to the Insurance & Risk Officer: see the Procedures document for details

N/A : Other document/s

File uploads

Please upload your files here:



Appendix C: Python Code

This appendix section provides the code for those who are interested with technical implementations. This code is not ready for production nor specifically optimised for it yet. Many processes that will be repeated later in the implementation were made into a def function, as explained in Chapter 4, for better reproducibility. Although not perfect, this code could serve as sufficient baseline resources for future researchers who want to dive deeper into this topic.

```
# Libraries & Functions
import sys
import importlib.metadata as metadata  # For Python 3.8+

import sklearn
import pandas as pd
import numpy as np
import matplotlib
import seaborn as sns
import nltk
import gensim
import yfinance
import torch
import transformers
import tqdm
import xgboost
import shap
import umap

print("Python:", sys.version)
print("pandas:", pd.__version__)
print("numpy:", np.__version__)
print("matplotlib:", matplotlib.__version__)
print("seaborn:", sns.__version__)
print("scikit-learn:", sklearn.__version__)
print("nltk:", nltk.__version__)
print("gensim:", gensim.__version__)
print("yfinance:", yfinance.__version__)
print("torch (PyTorch):", torch.__version__)
print("transformers:", transformers.__version__)
print("tqdm:", tqdm.__version__)
print("xgboost:", xgboost.__version__)
print("shap:", shap.__version__)
print("umap-learn:", umap.__version__)
```

```
# Handle ta version (fallback if __version__ doesn't exist)
try:
    print("ta (technical analysis):", metadata.version("ta"))
except:
    print("ta (technical analysis): version not found")
# Standard Libraries
import re
import string
import numpy as np
import pandas as pd

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

# Machine Learning & Model Evaluation
from sklearn.model_selection import (
    train_test_split
)
from sklearn.metrics import (
    classification_report,
    accuracy_score,
    roc_auc_score,
    roc_curve,
    roc_auc_score,
    RocCurveDisplay
)
from sklearn.preprocessing import *

from sklearn.preprocessing import StandardScaler
from umap import UMAP

# NLP Processing
import nltk
from nltk.corpus import stopwords
from nltk.stem import *
from nltk.tokenize import word_tokenize
import contractions
from gensim.models import Word2Vec

# Financial Data & Technical Analysis
import yfinance as yf
import ta  # Technical Analysis library

# Transformers & Deep Learning
import torch
from transformers import (
```

```
    AutoTokenizer,
    AutoModel,
    AutoModelForSequenceClassification
)
from tqdm import tqdm

# XGBoost
import xgboost as xgb

# SHAP for Explainability
import shap

from functools import lru_cache

def add_time_features(df):
    """
    Adds time-based features to a DataFrame with a DatetimeIndex.

    Parameters:
    df (pd.DataFrame): DataFrame with DatetimeIndex.

    Returns:
    pd.DataFrame: DataFrame with added time-based features.
    """
    df = df.copy()  # To avoid modifying the original DataFrame

    df['date'] = pd.to_datetime(df['date'])

    # Extract time features
    df['Day'] = df['date'].dt.day
    df['Weekday'] = df['date'].dt.weekday  # Monday=0, Sunday=6
    df['Week'] = df['date'].dt.isocalendar().week.astype(int)
    df['Month'] = df['date'].dt.month
    df['Year'] = df['date'].dt.year

    # Cyclic encoding for Month
    df['Month_sin'] = np.sin(2 * np.pi * df['Month'] / 12)
    df['Month_cos'] = np.cos(2 * np.pi * df['Month'] / 12)

    return df



def create_target(df):
    """
    Creates a 'Target' column indicating whether the stock price moves up or
down the next day.

    Parameters:
```

```
    df (pd.DataFrame): DataFrame containing a 'Close' column.

    Assumption:
    - 1 Minute Close price with actual close price won't be that much
different
    - Neutral = Down

    Returns:
    pd.DataFrame: DataFrame with the added 'Target' column.
    """
    df = df.copy()
    df["Target"] = df["Open"].shift(-1) - df["Close"]
    df["Target"] = df["Target"].apply(lambda x: "Up" if x > 0 else "Down")

    # Remove the last row (since it has NaN in 'Target')
    df = df.iloc[:-1].reset_index(drop=True)

    return df



def encode_target(df, column):
    """
    Encodes a categorical target column using LabelEncoder.

    Parameters:
    df (pd.DataFrame): DataFrame containing the target column.
    column (str): Name of the column to encode.

    Returns:
    pd.DataFrame: DataFrame with the encoded target column.
    """
    df = df.copy()
    encoder = LabelEncoder()
    df[column] = encoder.fit_transform(df[column])
    return df, encoder  # Return encoder in case inverse transformation is
needed



def show_encoded_classes(df, encoder, column):
    """
    Prints the encoded values and their corresponding class names for a given
target column.

    Parameters:
    df (pd.DataFrame): The DataFrame containing the target column.
    encoder (LabelEncoder): The fitted LabelEncoder.
    column (str): The target column that was encoded.
```

```
    Returns:
    None
    """
    print("Encoded Values and Class Names:")
    for encoded_value, classname in enumerate(encoder.classes_):
        print(f"Encoded Value: {encoded_value}, Class Name: {classname}")



def feature_addition(df):
    # ===== 1. Lag Features (Past Prices) FIBONACCI SEQUENCE =====
    df['Close_1'] = df['Close'].shift(1)  # Previous day's close
    df['Close_2'] = df['Close'].shift(2)
    df['Close_3'] = df['Close'].shift(3)
    df['Close_5'] = df['Close'].shift(5)
    df['Close_8'] = df['Close'].shift(8)
    df['Close_13'] = df['Close'].shift(13)

    # ===== 1.1. Lag Features Percentage Change (New Ratio Columns) =====
    df['Close_1_pct_change'] = (df['Close'] - df['Close_1']) / df['Close_1']
    df['Close_2_pct_change'] = (df['Close'] - df['Close_2']) / df['Close_2']
    df['Close_3_pct_change'] = (df['Close'] - df['Close_3']) / df['Close_3']
    df['Close_5_pct_change'] = (df['Close'] - df['Close_5']) / df['Close_5']
    df['Close_8_pct_change'] = (df['Close'] - df['Close_8']) / df['Close_8']
    df['Close_13_pct_change'] = (df['Close'] - df['Close_13']) /
df['Close_13']

    # ===== 2. Exponential Moving Averages (EMA) =====
    df['EMA_5'] = df['Close'].ewm(span=5, adjust=False).mean()
    df['EMA_8'] = df['Close'].ewm(span=8, adjust=False).mean()
    df['EMA_13'] = df['Close'].ewm(span=13, adjust=False).mean()

    # ===== 3. Volatility Features =====
    # Rolling Standard Deviation
    df['Rolling_Std_10'] = df['Close'].rolling(10).std()  # 10-day volatility
    df['Rolling_Std_20'] = df['Close'].rolling(20).std()  # 20-day volatility

    # ===== 4. RSI (Momentum) =====
    df['RSI_5'] = ta.momentum.RSIIndicator(df['Close'], window=5).rsi()
    df['RSI_8'] = ta.momentum.RSIIndicator(df['Close'], window=8).rsi()
    df['RSI_13'] = ta.momentum.RSIIndicator(df['Close'], window=13).rsi()

    # ===== 5. MACD (Trend) =====
    macd = ta.trend.MACD(df['Close'], window_slow=26, window_fast=12,
window_sign=9)
    df['MACD'] = macd.macd()  # MACD line
    df['MACD_Signal'] = macd.macd_signal()  # Signal line
    df['MACD_Hist'] = macd.macd_diff()  # Histogram
```

```
    # ===== 6. Stochastic Oscillator =====
    df['stoch_k'] = ta.momentum.stoch(df['High'], df['Low'], df['Close'],
window=5, smooth_window=3)
    df['stoch_d'] = df['stoch_k'].rolling(3).mean()  # Signal line (%D)
    df['slow_stoch_k'] = df['stoch_k'].rolling(3).mean()  # Slower %K
    df['slow_stoch_d'] = df['slow_stoch_k'].rolling(3).mean()  # Slower %D

    # Label signals as categories
    df['signal'] = 'neutral'
    df.loc[df['stoch_k'] > 80, 'signal'] = 'overbought'
    df.loc[df['stoch_k'] < 20, 'signal'] = 'oversold'

    # One-hot encode the signal column
    signal_one_hot = pd.get_dummies(df['signal'], prefix='signal')
    df = pd.concat([df, signal_one_hot], axis=1)
    df.drop(columns=['signal'], inplace=True)

    # ===== 7. Volume Indicators =====
    df['Volume_MA_5'] = df['Volume'].rolling(5).mean()
    df['Volume_MA_10'] = df['Volume'].rolling(10).mean()
    df['OBV'] = ta.volume.OnBalanceVolumeIndicator(df['Close'],
df['Volume']).on_balance_volume()

    # ===== 8. Price Returns =====
    df['Daily_Return'] = df['Close'].pct_change()  # Daily % return
    df['Cumulative_Return_5'] = (df['Close'] / df['Close'].shift(5)) - 1  # 5-
day return

    # ===== 9. Bollinger Bands =====
    df['MA_20'] = df['Close'].rolling(window=20).mean()
    df['Upper_Band'] = df['MA_20'] + (2 * df['Rolling_Std_20'])
    df['Lower_Band'] = df['MA_20'] - (2 * df['Rolling_Std_20'])

    # ===== 10. ADX (Trend Strength) =====
    df['ADX'] = ta.trend.ADXIndicator(df['High'], df['Low'], df['Close'],
window=14).adx()

    # Drop NaN rows (from rolling calculations)
    df.dropna(inplace=True)

    return df



def data_splitting(df, target_col, test_size):
    """
    Splits the dataset into training and testing sets.
```

```
    Parameters:
    df (pd.DataFrame): The dataset
    target_col (str): The column name of the target variable
    test_size (float): The proportion of data to use for testing

    Returns:
    X_train, X_test, y_train, y_test (tuple): Train-test split datasets
    """
    df = df.sort_index()

    X = df.drop(columns=[target_col])
    y = df[target_col]

    return train_test_split(X, y, test_size=test_size, shuffle=False)



def train_evaluate_and_explain_xgboost(df_name, X_train, y_train, X_test,
y_test, target_encoder):
    """
    Train an XGBoost classifier, evaluate its performance, and generate SHAP
explanations.

    Parameters:
    df_name (str): Name of the dataset
    X_train (pd.DataFrame): Training features
    y_train (pd.Series): Training labels
    X_test (pd.DataFrame): Test features
    y_test (pd.Series): Test labels
    target_encoder: Fitted LabelEncoder for target values

    Returns:
    xgb.XGBClassifier: The trained model
    """

    # Train model with all randomness controlled
    xgboost_model = xgb.XGBClassifier(
        eval_metric='logloss',
        booster='gbtree',
        random_state=0
    )

    xgboost_model.fit(X_train, y_train)

    # Evaluation
    y_pred = xgboost_model.predict(X_test)
    test_accuracy = xgboost_model.score(X_test, y_test)
    y_probs = xgboost_model.predict_proba(X_test)[:, 1]
    auc_score = roc_auc_score(y_test, y_probs)
```

```
    print(f"XGBoost Classifier Performance for {df_name}:")
    print("-" * 40)
    print(f"Test Accuracy: {test_accuracy:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print(f"\nROC AUC Score: {auc_score:.4f}")
    print("=" * 40 + "\n")

    show_encoded_classes(df_name, target_encoder, "Target")
    print("=" * 40 + "\n")

    # ROC Curve Visualization
    plt.figure(figsize=(8, 6))
    fpr, tpr, _ = roc_curve(y_test, y_probs)
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
{auc_score:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'Receiver Operating Characteristic (ROC) - {df_name}')
    plt.legend(loc="lower right")
    plt.grid(True)
    plt.show()

    # SHAP Explanations
    print("Generating SHAP Explanations...")

    # Create explainer and get SHAP values
    explainer = shap.Explainer(xgboost_model)
    shap_values = explainer(X_train)

    # Summary plot
    plt.figure(figsize=(10, 6))
    shap.summary_plot(shap_values, X_train, show=False)
    plt.title(f"SHAP Feature Importance - {df_name}")
    plt.tight_layout()
    plt.show()

    # Bar plot
    plt.figure(figsize=(10, 6))
    shap.plots.bar(shap_values, show=False)
    plt.title(f"Feature Importance - {df_name}")
    plt.tight_layout()
    plt.show()
```

```
    # Force plot - fixed version
    shap.initjs()
    print("\nSHAP Force Plot for First Observation:")

    # Create proper Explanation object for force plot
    expected_value = explainer.expected_value
    if isinstance(expected_value, np.ndarray) and len(expected_value) > 1:
        expected_value = expected_value[1]  # For binary classification

    shap.force_plot(
        expected_value,
        shap_values.values[0],  # Use .values to get numpy array
        X_train.iloc[0],
        matplotlib=True,
        figsize=(12, 4),
        feature_names=X_train.columns.tolist()
    )



def process_financial_tweets(final_data, text_column, max_length_clf,
max_length_vec, col_prefix, stock_symbol=None, timestamp_column=None):
    """
    Combined process for FinBERT sentiment analysis and vectorization of
financial tweets
    1. Sentiment classification
    2. Tweet preprocessing
    3. Embedding/vectorization
    """
    # 1. Setup device and load models
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    # Load tokenizer and models
    tokenizer = AutoTokenizer.from_pretrained('yiyanghkust/finbert-tone')

    # Load classification model
    classification_model =
AutoModelForSequenceClassification.from_pretrained('yiyanghkust/finbert-tone')
    classification_model = classification_model.to(device)
    classification_model.eval()

    # Load embedding model (for vectorization)
    embedding_model = AutoModel.from_pretrained('yiyanghkust/finbert-tone')
    embedding_model = embedding_model.to(device)
    embedding_model.eval()

    # Track relevant stock symbols
    stock_symbols = set([stock_symbol]) if stock_symbol else set()
```

```
    # 2. Define classification function for batch processing
    def batch_classify(texts, batch_size=32):
        all_classes = []
        for i in range(0, len(texts), batch_size):
            batch = texts[i:i+batch_size]
            inputs = tokenizer(batch, return_tensors='pt', truncation=True,
                               padding=True,
max_length=max_length_clf).to(device)
            with torch.no_grad():
                outputs = classification_model(**inputs)
                probs = torch.nn.functional.softmax(outputs.logits, dim=1)
                pred_classes = torch.argmax(probs, dim=1).cpu().numpy() - 1  #
Convert to -1/0/1
            all_classes.extend(pred_classes)
        return all_classes

    # 3. Define tweet-specific preprocessing
    def preprocess_tweet(text):
        text = str(text)

        # Extract and track cashtags instead of removing them
        cashtags = re.findall(r'\$([A-Za-z]+)', text)
        for symbol in cashtags:
            stock_symbols.add(symbol)

        # Keep cashtags but standardize format
        text = re.sub(r'\$([A-Za-z]+)', r'CASHTAG_\1', text)

        # Handle mentions
        text = re.sub(r'@\w+', 'MENTION', text)

        # Process hashtags - keep the content but remove the symbol
        text = re.sub(r'#(\w+)', r'\1', text)

        # Basic URL removal
        text = re.sub(r'https?://\S+|www\.\S+', 'URL', text)

        # Convert common financial emojis to text
        emoji_map = {
            '📈': 'BULLISH',
            '📉': 'BEARISH',
            '🚀': 'ROCKET',
            '💰': 'MONEY',
            '🐂': 'BULL',
            '🐻': 'BEAR'
        }
        for emoji, replacement in emoji_map.items():
            text = text.replace(emoji, replacement)
```

```
        # Remove other emojis and special characters while keeping % $ € £ ¢ ¥
        text = re.sub(r'[^\w\s%$€£¢¥]', ' ', text)
        return text.strip()

    # 4. Define vectorization function with caching
    @lru_cache(maxsize=10000)  # Cache results for identical texts
    def vectorize_single_text(text):
        inputs = tokenizer(
            text,
            return_tensors='pt',
            padding=True,
            truncation=True,
            max_length=max_length_vec,  # Shorter for tweets
            add_special_tokens=True
        ).to(device)

        with torch.no_grad():
            outputs = embedding_model(**inputs, output_hidden_states=True)

            # Get hidden states
            hidden_states = outputs.hidden_states

            # Stack the last 4 layers
            last_four_layers = torch.stack(hidden_states[-4:])

            # Get attention mask and expand dimensions
            attention_mask = inputs['attention_mask'].unsqueeze(0).unsqueeze(-
1)

            # Apply attention mask and calculate mean
            masked_embeddings = last_four_layers * attention_mask
            summed_embeddings = masked_embeddings.sum(dim=2)
            mean_embeddings = summed_embeddings /
attention_mask.sum(dim=2).clamp(min=1)

            # Apply learnable weights to layers (could be trained for specific
task)
            layer_weights = torch.nn.functional.softmax(torch.tensor([1.0,
2.0, 3.0, 4.0], device=device), dim=0)
            weighted_embeddings = mean_embeddings * layer_weights.view(4, 1,
1)

            # Get the final embedding
            final_embedding = weighted_embeddings.sum(dim=0).squeeze(0)

            return final_embedding.cpu().numpy()
```

```
    def finbert_vectorize_batch(text_list, batch_size=32):
        all_embeddings = []
        for text in text_list:
            all_embeddings.append(vectorize_single_text(text))
        return np.vstack(all_embeddings)

    # 5. Start execution process

    # STEP 1: Run sentiment classification first
    texts = final_data[text_column].fillna('').tolist()
    sentiment_classes = batch_classify(texts)
    final_data[f'{col_prefix}_sentiment_class'] = sentiment_classes

    # STEP 2: Apply preprocessing to tweets
    final_data['processed_text'] =
final_data[text_column].fillna('').apply(preprocess_tweet)

    # Add time-based features if timestamp column is provided
    if timestamp_column and timestamp_column in final_data.columns:
        # Convert to datetime if not already
        if not
pd.api.types.is_datetime64_any_dtype(final_data[timestamp_column]):
            final_data['timestamp'] =
pd.to_datetime(final_data[timestamp_column])
        else:
            final_data['timestamp'] = final_data[timestamp_column]

        # Extract temporal features
        final_data['hour_of_day'] = final_data['timestamp'].dt.hour
        final_data['day_of_week'] = final_data['timestamp'].dt.dayofweek
        final_data['weekend'] = final_data['day_of_week'].isin([5,
6]).astype(int)

        # Calculate recency - assuming the most recent date is the reference
        max_date = final_data['timestamp'].max()
        final_data['days_from_recent'] = (max_date -
final_data['timestamp']).dt.total_seconds() / (60*60*24)

        # Exponential recency weight (more recent = higher weight)
        final_data['recency_weight'] = np.exp(-0.1 *
final_data['days_from_recent'])

    # STEP 3: Generate vectors from processed text
    body_vectors =
finbert_vectorize_batch(final_data['processed_text'].tolist())

    # Apply recency weighting if available
    if 'recency_weight' in final_data.columns:
```

```
        body_vectors = body_vectors *
final_data['recency_weight'].values.reshape(-1, 1)

    # Create column names for vectors
    body_cols = [f'{col_prefix}_{i}' for i in range(body_vectors.shape[1])]

    # Merge with original data
    features_to_drop = [text_column, 'processed_text']
    if timestamp_column:
        features_to_drop.extend(['timestamp', 'days_from_recent'])

    final_data_processed = pd.concat([
        final_data.drop(features_to_drop, axis=1),
        pd.DataFrame(body_vectors, columns=body_cols)
    ], axis=1)

    return final_data_processed, list(stock_symbols)

# Sentiment Data
- Tweet
# Load datasets
company_tweet = pd.read_csv('Company_Tweet.csv')
tweet = pd.read_csv('Tweet.csv')
tweet_18_20 = pd.read_csv('tesla_tweet_2018_to_2020.csv')

# Process tweet_18_20 data
tweet_18_20 = (
    tweet_18_20.assign(date=pd.to_datetime(tweet_18_20['created_at']).dt.date)
    [['date', 'full_text']]
    .rename(columns={'full_text': 'body'})
)

# Process and merge company tweets
comb_tweet = (
    pd.merge(tweet, company_tweet, on='tweet_id', how='inner')
    .query("ticker_symbol == 'TSLA'")
    .assign(date=lambda x: pd.to_datetime(x['post_date'],
unit='s').dt.strftime('%Y-%m-%d'))
    [['date', 'body']]
)

# Combine and aggregate all tweets PROPERLY
final_tweet = (
    pd.concat([
        comb_tweet,
        tweet_18_20.assign(date=lambda x:
pd.to_datetime(x['date']).dt.strftime('%Y-%m-%d'))
    ])
```

```
    .groupby('date', as_index=False)
    .agg({'body': ' '.join})  # Explicit aggregation
    .assign(date=lambda x: pd.to_datetime(x['date']))  # Convert to datetime
    .sort_values('date')
    .reset_index(drop=True)
)

final_tweet
- News Articles
ns_df = pd.read_excel('deepseek_tesla_news_summary_2015_2020.xlsx')
ns_df


# TA & Merge Final Data
tsla = yf.Ticker("TSLA")
tsla = tsla.history(start="2015-01-01", end="2020-11-30")

tsla = tsla.drop(['Dividends', 'Stock Splits'], axis=1)

# Move the Date from index to a column and convert to date only
tsla['date'] = tsla.index.date  # Add Date column from index (date only)
tsla.reset_index(drop=True, inplace=True)  # Reset the index

tsla = add_time_features(tsla)
tsla = create_target(tsla)
tsla, tsla_target_encoder = encode_target(tsla, "Target")
tsla = feature_addition(tsla)
tsla_ft = pd.merge(tsla, final_tweet, on='date', how='left')
final_data = pd.merge(tsla_ft, ns_df, on='date', how='left')
final_data = final_data.drop(['date'], axis=1)

final_data


# Text Processing (Sentiment Classification + Vectorization)
- Tweet Body SC & Vec
final_data_processed_body, body_symbols = process_financial_tweets(final_data,
                                                                    text_column
='body',
                                                                    col_prefix=
'tweet',
                                                                    max_length_
clf=256,
                                                                    max_length_
vec=64)

final_data_processed_body
- News Articles SC & Vec
```

```
final_data_processed_all, art_symbols =
process_financial_tweets(final_data_processed_body,
                                                                    text_column='
article',
                                                                    col_prefix='a
rt',
                                                                    max_length_cl
f=512,
                                                                    max_length_ve
c=64)

final_data_processed_all
- Separate Data Components
final_data = final_data.drop(['processed_text'], axis=1)
final_data_processed_body = final_data_processed_body.drop(['processed_text'],
axis=1)

# Technical Analysis Only
ta_only = tsla.drop(['date'], axis=1)

# Tweet Only
cols = final_data_processed_body.columns
adx_pos = cols.get_loc('article')
selected_cols = (
    cols[adx_pos + 1:].tolist()  # All columns after ADX
    + ['Target']                 # Add Target at the end
    if 'Target' in cols
    else cols[adx_pos + 1:]      # Fallback if Target doesn't exist
)
tweet_only = final_data_processed_body[selected_cols]

# News Only
rest_cols = final_data_processed_all.columns
tweet_pos = rest_cols.get_loc('tweet_767')
selected_news_cols = (
    rest_cols[tweet_pos + 1:].tolist()
    + ['Target']
    if 'Target' in cols
    else rest_cols[tweet_pos + 1:]
)
news_only = final_data_processed_all[selected_news_cols]


# Modelling
BEST RESULT: TA + Tweets (Unstransformed) + News (UMAP)
- Data Splitting
X_train_tsla, X_test_tsla, y_train_tsla, y_test_tsla =
data_splitting(final_data_processed_all, "Target", 0.025)
```

```
# Select all columns representing news
tr_art_cols = X_train_tsla.columns.get_loc('tweet_767') + 1
# The rest of the data
rest_columns = X_train_tsla.columns[:X_train_tsla.columns.get_loc('tweet_767')
+ 1]

# Transform X_train
x_tr_news = X_train_tsla.iloc[:, tr_art_cols:]
x_tr_rest = X_train_tsla[rest_columns]

# Transform X_test
x_te_news = X_test_tsla.iloc[:, tr_art_cols:]
x_te_rest = X_test_tsla[rest_columns]
- Dimensionality Reduction Method (only to News): UMAP
# Initialize transformers
scaler_dr = StandardScaler()
dr_algo = UMAP(n_components=45, n_neighbors=15, n_jobs=1, random_state=42)

# Train data pipeline
tr_news_scaled = scaler_dr.fit_transform(x_tr_news)
tr_news_dr = dr_algo.fit_transform(tr_news_scaled)

train_news_dr = x_tr_rest.join(pd.DataFrame(tr_news_dr,
index=x_tr_news.index))
train_news_dr.columns = train_news_dr.columns.astype(str)

# Test data pipeline (Use scaled data for PCA)
te_news_scaled = scaler_dr.transform(x_te_news)
te_news_dr = dr_algo.transform(te_news_scaled)

test_news_dr = x_te_rest.join(pd.DataFrame(te_news_dr, index=x_te_news.index))
test_news_dr.columns = test_news_dr.columns.astype(str)
- Model Evaluation
train_evaluate_and_explain_xgboost("Tesla", train_news_dr, y_train_tsla,
test_news_dr, y_test_tsla, tsla_target_encoder)
SECOND BEST RESULT: TA + Tweets (Untransformed)
- Data Splitting
final_data_processed_body = final_data_processed_body.drop(['article',
'art_sentiment_class'], axis=1)
X_train_tsla, X_test_tsla, y_train_tsla, y_test_tsla =
data_splitting(final_data_processed_body, "Target", 0.025)
- Model Evaluation
train_evaluate_and_explain_xgboost("Tesla", X_train_tsla, y_train_tsla,
X_test_tsla, y_test_tsla, tsla_target_encoder)
THIRD BEST RESULT: TA Only
- Data Splitting
X_train_tsla, X_test_tsla, y_train_tsla, y_test_tsla = data_splitting(ta_only,
"Target", 0.025)
```

```
- Model Evaluation
train_evaluate_and_explain_xgboost("Tesla", X_train_tsla, y_train_tsla,
X_test_tsla, y_test_tsla, tsla_target_encoder)
Fourth Best Result: TA + tweets + News (Untransformed)
- Data Splitting
X_train_tsla, X_test_tsla, y_train_tsla, y_test_tsla =
data_splitting(final_data_processed_all, "Target", 0.025)
- Model Evaluation
train_evaluate_and_explain_xgboost("Tesla", X_train_tsla, y_train_tsla,
X_test_tsla, y_test_tsla, tsla_target_encoder)
Fifth Best Result: TA + News (UMAP)
- Data Splitting
X_train_ta, X_test_ta, y_train_tsla, y_test_tsla = data_splitting(ta_only,
"Target", 0.025)
tr_ta_news_umap = X_train_ta.join(pd.DataFrame(tr_news_dr,
index=x_tr_news.index))
te_ta_news_umap = X_test_ta.join(pd.DataFrame(te_news_dr,
index=x_te_news.index))
- Model Evaluation
train_evaluate_and_explain_xgboost("Tesla", tr_ta_news_umap, y_train_tsla,
te_ta_news_umap, y_test_tsla, tsla_target_encoder)
Worst Result: TA + News (Untransformed)
- Data Splitting
ta_news = ta_only.join(news_only.drop(['Target'], axis=1), how='inner')
X_train_tsla, X_test_tsla, y_train_tsla, y_test_tsla = data_splitting(ta_news,
"Target", 0.025)
- Model Evaluation
train_evaluate_and_explain_xgboost("Tesla", X_train_tsla, y_train_tsla,
X_test_tsla, y_test_tsla, tsla_target_encoder)
```

Appendix D: Data Sample

- **Technical Indicators**

	Open	High	Low	Close	Volume	Day	Weekday	Week	Month	Year	...	Volume_MA_10	OBV	Daily_Return	Cumulative_Return_5	MA_20	Upper_Band
0	14.052000	14.506667	13.987333	14.474000	89731500	20	4	8	2	2015	...	92318850.0	-189646500	0.025507	0.070140	13.996800	14.918865
1	14.377333	14.546667	13.755333	13.822667	127497000	23	0	9	2	2015	...	100202700.0	-317143500	-0.045000	0.017520	14.016967	14.902784
2	13.819333	13.819333	13.446667	13.607333	99054000	24	1	9	2	2015	...	104899500.0	-416197500	-0.015578	-0.001174	14.008833	14.907098
3	13.662667	13.809333	13.505333	13.584000	58642500	25	2	9	2	2015	...	102678000.0	-474840000	-0.001715	-0.003424	14.001433	14.911659

- **Social Media (Tweets)**

	date	body
0	2015-01-01	$GM $TSLA: Volkswagen Pushes 2014 Record Recal...
1	2015-01-02	Will Audi's Electric Q7 Cause $TSLA Model X Ba...
2	2015-01-03	Stock Traders Purchase Large Volume of Tesla M...
3	2015-01-04	"@kirillklip: #China bets on Electric Cars in ...
4	2015-01-05	$TSLA Monthly closed below 13ma, bearish down ...
...	...	...
2157	2020-11-27	The Tesla of boats: Silent Yachts ramps up pro...
2158	2020-11-28	Check out this brilliant brother.. Uncle Rich ...
2159	2020-11-29	Wall Street Braces As Tesla Addition to S&...
2160	2020-11-30	My brother just got hired to tesla lol if they...
2161	2020-12-01	Musk warns Tesla employees the stock will be c...

2162 rows × 2 columns

- **Financial News**

	date	article
0	2015-01-02	TSLA rose 2.3% to $227.50 as Morgan Stanley re...
1	2015-01-05	TSLA dropped 4.1% to $218.20 as oil hit $49/ba...
2	2015-01-06	Shares rebounded 3.2% to $225.10 after Model S...
3	2015-01-07	TSLA fell 1.8% as Consumer Reports documented ...
4	2015-01-08	Tesla rose 2.5% to $230.60 after setting 517-m...
...	...	...
1481	2019-05-31	Stock closed at $185.16, down 22.4% amid equit...
1482	2019-10-11	Shares rose 2.7% to $256.97 as Q3 delivery bea...
1483	2019-10-14	Closed at $258.71, up 0.7% in quiet trading. M...
1484	2019-10-15	Jumped 3.7% to $268.21 after Goldman upgrade. ...
1485	2020-05-29	Finished at $167.78, down 5.1% amid Fremont re...

1486 rows × 2 columns