STATISTICAL AND DEEP LEARNING MODELS FOR SENTIMENT-BASED MARKET PRICE FORECASTING

By
Lai Chun Kit

Dissertation submitted in partial fulfilment of the
requirement for the award of the degree of
Bachelor of Science (Financial Mathematics) With Honours

FACULTY OF COMPUTER SCIENCE AND MATHEMATICS
UNIVERSITI MALAYSIA TERENGGANU
2025

DISSERTATION CONFIRMATION AND APPROVAL

This is acknowledged and confirmed that the dissertation entitled: Statistical and Deep Learning Models for Sentiment-Based Market Price Forecasting by Lai Chun Kit Matric No.: S65947 have been checked and all the suggested corrections have been done. The dissertation is submitted to the Faculty of Computer Science and Mathematics, Universiti Malaysia Terengganu in partial fulfilment of the requirements for the award of the degree of Bachelor of Science (Financial Mathematics) with Honours.

Authorized by:

…….. Date: 22/06/2025 ……...

Main Supervisor
Name: Loy Kak Choon
Official Stamp:

Dr. Loy Kak Choon
Pensyarah Kanan
Fakulti Sains Komputer dan Matematik
Universiti Malaysia Terengganu

…….. Date: ……...

Co-Supervisor (If any)
Name:
Official Stamp:

…….. Date: ……...

PITA's Coordinator
Bachelor of Science (Financial Mathematics) with Honours
Name:
Official Stamp:

DECLARATION

I hereby declare that this dissertation is the result of my own research except as cited in the references.

Signature : .....
Name : Lai Chun Kit
Matric No. : S65947
Date : 22/6/2025

ACKNOWLEDGEMENTS

At the end of my dissertation, I want to thank all of those who made this dissertation possible and the process a pleasure for me. It's been a harsh and long struggle but also something that filled me with joy.

First of all, I wish to express my sincere gratitude to my supervisor, Loy Kak Choon, for his guidance, patience and considerable feedback. With his expertise and encouragement, he had greatly shaped this research, and I can never repay the time and effort he provided in mentoring me.

I am grateful to my friends for the continuous encouragement and support they gave me, especially Tan Zhi Hang who have literally been a pillar throughout this project leading to motivation, advises and much needed distraction from academia. They have been the greatest support and encouraged me when I doubted myself.

Finally, I would like also thank to my family for their unconditional support, emotional stability and love. They have been my greatest support through their patience, sacrifices, and belief in me. This dissertation would not have been possible without them.

STATISTICAL AND DEEP LEARNING MODELS FOR SENTIMENT-BASED MARKET PRICE FORECASTING

ABSTRACT

The integration of news-headline sentiment with macroeconomic and technical indicators offers a powerful means to enhance equity-price forecasting. Using Tesla Inc. as a case study, this research assembles monthly sentiment indices from January 2020 to December 2023 via three distinct approaches: lexicon-based (Loughran–McDonald and VADER), classical machine-learning classifiers (Logistic Regression, SVM, Naïve Bayes, Random Forest, XGBoost) on Financial PhraseBank features (TF-IDF, Bag-of-Words, Word2Vec), and deep-learning embeddings (FinBERT, RoBERTa, DistilBERT). The top sentiment extractor in each category was selected on classification performance and its outputs aggregated into monthly series, which were then merged with six theory-driven macroeconomic indicators (e.g. CPI, GDP, consumer confidence) and four standard technical metrics (monthly return, volatility, RSI, MACD).

Forecasting accuracy was evaluated using two paradigms: a linear ARIMAX model incorporating exogenous variables, and a one-layer LSTM network. In the ARIMAX framework, the feature-enhanced BoW–Logistic Regression pipeline combined with macro and technical signals achieved the lowest out-of-sample RMSE (7.05) and highest R^2 (0.95); exclusion of macro variables raised RMSE to 9.70 ($R^2 = 0.91$), underlining the added value of macroeconomic inputs. Among LSTM variants, the SVM-LSTM on fine-tuned BoW features (tech only) delivered the best hold-out performance (RMSE=8.36, $R^2 = 0.94$), whereas the inclusion of macro indicators uniformly worsened forecast accuracy and produced overly smooth “flat-line”

predictions. Lexicon-based models (LM: RMSE=12.32; VADER: RMSE=15.72) and transformer embeddings (RoBERTa: RMSE=8.67; FinBERT/DistilBERT: ∼14) consistently underperformed their classical-ML counterparts.

These results demonstrate that, for monthly TSLA forecasting, machine-learning–derived sentiment features integrated within an ARIMAX framework strike the optimal balance between precision and computational efficiency, while deep-learning pipelines demand substantial GPU resources and offer limited practical gains. The study provides clear guidance for quantitative investors and fintech developers on selecting and deploying sentiment-augmented models under real-world constraints.

Model Statistik dan Pembelajaran Mendalam untuk Ramalan Harga Pasaran Berasaskan Sentimen

ABSTRAK

Integrasi analisis sentimen tajuk berita dengan penunjuk makroekonomi dan teknikal telah terbukti mempertingkatkan ketepatan ramalan harga ekuiti. Mengambil Tesla Inc. sebagai kajian kes, penyelidikan ini membina indeks sentimen bulanan bagi tempoh Januari 2020 hingga Disember 2023 melalui tiga pendekatan utama: (1) kaedah berasaskan leksema menggunakan kamus kewangan Loughran–McDonald dan model VADER, (2) pengklasifikasi pembelajaran mesin klasik (Logistic Regression, SVM, Naïve Bayes, Random Forest, XGBoost) yang dilatih ke atas korpus Financial PhraseBank dengan penjanaan ciri TF-IDF, Bag-of-Words dan Word2Vec, dan (3) pengekstrakan ciri terbenam pembelajaran mendalam (FinBERT, RoBERTa, DistilBERT). Model sentimen terbaik dalam setiap kategori dipilih berdasarkan prestasi klasifikasi dan diagregasi menjadi siri bulanan. Siri ini kemudian digabungkan dengan enam penunjuk makroekonomi (CPI, KDNK, indeks keyakinan pengguna, dan lain-lain) serta empat indikator teknikal standard (pulangan bulanan, volatiliti, RSI, MACD).

Ketepatan ramalan diuji menggunakan dua rangka kerja: ARIMAX dengan pemboleh ubah eksogen, dan rangkaian LSTM satu lapisan. Dalam model ARIMAX, gabungan BoW–Logistic Regression yang dipertingkatkan ciri dengan isyarat makro dan teknikal mencapai RMSE terendah (7.05) dan R^2 tertinggi (0.95); pengecualian penunjuk makro menaikkan RMSE kepada 9.70 ($R^2 = 0.9$), menekankan nilai tambahan makroekonomi. Dalam rangka LSTM, paip SVM-LSTM pada ciri BoW terbaik menghasilkan RMSE=8.36 dan $R^2 = 0.94$ apabila hanya data teknikal

digunakan, manakala penambahan makro secara konsisten menurunkan prestasi dan menghasilkan ramalan yang terlalu “pejal.” Kaedah leksema mencatat RMSE=12.32 (LM) dan 15.72 (VADER), manakala terbenam transformer (RoBERTa RMSE=8.67; FinBERT/DistilBERT ≈14) kekal di bawah tahap pembelajaran mesin.

Hasil ini menunjukkan bahawa, untuk ramalan bulanan TSLA, ciri sentimen berasaskan pembelajaran mesin yang digabungkan dalam kerangka ARIMAX menawarkan kompromi terbaik antara ketepatan dan kecekapan pengiraan. Sebaliknya, rangkaian pembelajaran mendalam memerlukan sumber GPU yang besar namun memberikan kelebihan terhad. Kajian ini menyediakan panduan praktikal bagi pelabur kuantitatif dan pembangun teknologi kewangan (fintech) dalam memilih dan melaksanakan model ramalan berasaskan sentimen di bawah kekangan sebenar.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviations	
UMT	Universiti Malaysia Terengganu
ACF	Autocorrelation Function
ADF	Augmented Dickey–Fuller
AIC	Akaike Information Criterion
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	Autoregressive Integrated Moving Average with Exogenous Variables
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transform
BIC	Bayesian Information Criterion
BoW	Bag-of-Words
CSV	Comma-Separated Values
CPI	Consumer Price Index
CPIAUCSL	Consumer Price Index for All Urban Consumers: All Items
DL	Deep Learning

FRED	Federal Reserve Economic Data
GDELT	Global Database of Events, Language, and Tone
GDP	Gross Domestic Product
GPU	Graphics Processing Unit
ISE	Istanbul Stock Exchange
JB	Jarque–Bera test
LB	Ljung–Box test
LSTM	Long Short-Term Memory
M2SL	M2 Money Stock
MAE	Mean Absolute Error
MACD	Moving Average Convergence Divergence
MSE	Mean Squared Error
NYSE	New York Stock Exchange
OHLC	Open, High, Low, Close
OOS	Out-of-Sample
PACF	Partial Autocorrelation Function
PAYEMS	Total Nonfarm Payroll Employment
PCE	Personal Consumption Expenditures
PLS	Partial Least Squares
PLS-ARIMAX	Partial Least Squares–augmented ARIMAX

PPI	Producer Price Index
PPIACO	Producer Price Index for All Commodities
PPI	Producer Price Index
PPIACO	Producer Price Index for All Commodities
PPI	Producer Price Index
PPIACO	Producer Price Index for All Commodities
RSAFS	Retail Sales: Furniture Stores
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
R^2	Coefficient of Determination (R-squared)
RSI	Relative Strength Index
SMA	Simple Moving Average
SSE	Shanghai Stock Exchange
SST-2	Stanford Sentiment Treebank Version 2
SVM	Support Vector Machine
TSLA	Tesla, Inc. (ticker symbol)
TF-IDF	Term Frequency–Inverse Document Frequency
UMCSENT	University of Michigan Consumer Sentiment Index
UNRATE	Unemployment Rate
VAR	Vector Autoregression

VIF	Variance Inflation Factor
XGBoost	eXtreme Gradient Boosting

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Financial economics and portfolio management have long faced the enormous challenge on forecasting stock prices. Traditional time-series models such as Autoregressive Integrated Moving Average (ARIMA) and its extension namely Autoregressive Integrated Moving Average with Exogeneous variables (ARIMAX) can capture linear trends in asset prices, but often struggle with unexpected, news-driven movements (Ren, Wu, & Liu, 2019). Sentiment analysis of financial news has widely use by researcher in recent years to improve predictive performance on stock price. Research show that sentiment scores often correlate with short-term price momentum (Cristescu, Mara, & Nerișanu, 2023). However, lexicon-based approaches such as Valence Aware Dictionary and Sentiment Reasoner (VADER) (Hutto & Gilbert, 2014) and the Loughran–McDonald financial dictionary (Loughran & McDonald, 2011), are prone to catch the context sensitivity to accurately represent complex market dynamics.

Vectorization methods (TF-IDF, Bag-of-Words, Word2Vec) with adjustable parameter optimization can be make use to extract richer sentiment features by machine-learning classifiers (e.g., Support Vector Machines, Random Forests) that have been trained on domain-specific corpora such as the Financial PhraseBank (Malo et al., 2014). Deep-learning embeddings (e.g., FinBERT, RoBERTa) provide extra benefits by encoding semantic details, but their high computational cost raises concerns regarding their real-time applicability in trading application (Araci, 2019). Moreover, sentiment signals can deliver valuable information, macroeconomic

indicators (e.g., Consumer Price Index, GDP growth, consumer confidence) also play a critical role in influencing market expectations and driving price fluctuations (Chen, Roll, & Ross, 1986). Surprising, there are few studies that combined these two sources, sentiment and macro variables, by feeding it into both ARIMAX and Long Short-Term Memory (LSTM) frameworks.

This objective of this research is to aims to fill this gap by evaluating systematically compare of lexicon-based, machine-learning, and deep-learning sentiment pipelines with paired of important macroeconomic and technical indicators in forecast Tesla stock prices. In particularly, we will (1) collect and preprocess financial-news headlines (2020–2023) via the GDELT API; (2) generate sentiment features using lexicon, ML, and DL methods; (3) merge these features with selected macro and technical variables; and (4) evaluate predictive performance using ARIMAX and LSTM models under an out-of-sample testing approach. This study is aim to determine the optimal combination of sentiment model and exogenous indicators that achieves the highest forecasting accuracy while maintaining computational efficiency by comparing these pipelines. This paper is crucial for portfolio managers, algorithmic traders, and fintech developers to find out the best approach for who are operating under real-time constraints.

1.2 Research Background

The prediction of stock prices has been an on going issue in the field of financial economics for a number of years. It has significant implications for portfolio management, algorithmic trading, and risk assessment. Traditional time-series approaches such as the Autoregressive Integrated Moving Average (ARIMA) and its extension ARIMAX, which bring in exogenous variables are found to be capable on capturing linear dependencies and some external factors. However, they frequently fail to capture the unforeseen changes of markets to news or macroeconomic shifts (Ren, Wu, & Liu, 2019). Researchers have increasing investigating the role of sentiment analysis in order to overcome these constraints. This approach relies on the assumption that the dominating investor sentiment, which is reflected in financial news headlines,

may act as an early warning sign of short-term price movement (Cristescu, Mara, & Nerișanu, 2023). Lexicon-based techniques had been used in the earliest stages of research in this field, applying dictionaries such as VADER (Hutto & Gilbert, 2014) and the Loughran–McDonald financial lexicon (Loughran & McDonald, 2011). However, these methods frequently lack contextual context and domain precision.

Recently, supervised machine-learning classifiers (e.g., Support Vector Machines, Random Forests, XGBoost) have been trained on financial corpora, such as the Financial PhraseBank, to generate more detailed sentiment features using vectorization methods such as TF-IDF, Bag-of-Words, and Word2Vec (Malo et al., 2014). These domain-tuned features often surpass typical lexicon scores in detecting complex sentiment shifts that are pertinent to asset prices. Meanwhile, deep-learning embeddings (e.g., FinBERT, RoBERTa, DistilBERT) have shown the ability to extracting even finer semantic information (Araci, 2019). However, their high computational cost raises concerns regarding their real-time applicability in trading systems.

In the meantime, macroeconomic indicators, including the Consumer Price Index (CPI), Gross Domestic Product (GDP), and Consumer Confidence Index, remain to be important variables in stock valuations (Chen, Roll, & Ross, 1986). Although sentiment and macro signals both offer useful information, there are few studies that have merged both of them within the same forecasting framework. More specifically, the comparable effectiveness of lexicon-based, machine-learning, and deep-learning sentiment pipelines has not been systematically evaluated under both ARIMAX and Long Short-Term Memory (LSTM) models when combined with macroeconomic and technical indicators.

The objective of this research aims to solve this problem through carrying out a comprehensive assessment of these sentiment models, as well as essential macroeconomic and technical variables, in the setting of forecasting Tesla Inc. stock prices. In doing so, it intends to identify the combination that offers the most accurate out-of-sample results while maintaining computational efficiency, thus providing useful information for real-time financial applications.

1.3 Problem Statement

There is empirical evidence that both sentiment signals and macroeconomic factors contribute to predictions of stock-prices, but few studies combine these predictors in a linear (ARIMAX) as well as non-linear (LSTM) framework. Previous studies have demonstrated that ARIMAX does not respond well to sudden news-driven price changes (Ren et al., 2019), and that sentiment analysis can capture short-term momentum (Cristescu, Mara & Nerișanu, 2023). However, no study to the best of our knowledge systematically compares lexicon-based, machine-learning, and deep-learning sentiment approaches across ARIMAX and LSTM. Additionally, the majority of studies (1) exclude technical trading contributions (such as the RSI and MACD) into the model of sentiment analysis and (2) exclude a complete set of macroeconomic variables (such as CPI, GDP, unemployment) into the sentiment features (Chen, Roll, & Ross, 1986). As a consequence, there is no well-defined guidance on the optimal mix of the sentiment model and exogenous indicators in terms of out-of-sample accuracy, and nor is there a clear identification of how the predictive improvement can be traded off against the computational cost in real-time trading applications.

1.4 Research Objectives

The objectives of the research are:

i. Enhance forecasting performance combining sentiment analysis with macroeconomic (CPI, GDP, consumer confidence, etc.) and technical (returns, volatility, RSI, MACD, etc.) signals on the TSLA out-of-sample price forecasting.
ii. Compare systematically lexicon-based approaches (VADER, Loughran–McDonald), machine-learning classifiers (Logistic Regression, SVM, Naive Bayes, Random Forest, XGBoost) and deep-learning embeddings (FinBERT, RoBERTa, DistilBERT) both within ARIMAX and LSTM architectures.

iii. Compare overall results between the sentiment model+external-indicator combinations (technical±macro) with respect to out-of-sample RMSE, MAE, MAPE and R^2.

iv. Assess the trade-off between predictive power and computational costs of both aforementioned ARIMAX and LSTM pipelines to define real-time deployment strategies.

1.5 Research Scope

Tesla Inc.'s closing-price behavior from January 2020 through December 2023 was analyzed by integrating market data, macroeconomic indicators, technical metrics and sentiment features into two parallel forecasting pipelines. Price series were sourced from Yahoo Finance, with daily closes aggregated to month-end values, while sentiment inputs were derived from financial-news headlines obtained via the GDELT API, translated into English, deduplicated and processed through lexicon-based scoring using VADER and the Loughran–McDonald dictionary, supervised machine-learning classifiers (Logistic Regression, SVM, Naive Bayes, Random Forest and XGBoost) trained on the Financial PhraseBank corpus with TF-IDF, Bag-of-Words and Word2Vec representations optimized via grid search, and deep-learning sentence embeddings from FinBERT, RoBERTa and DistilBERT. A select subset of macroeconomic variables (such as CPI, GDP and consumer-confidence indices) passed theoretical relevance and correlation screening, and classic monthly technical indicators (total return, volatility, RSI and MACD) were computed.

Two forecasting frameworks—a linear ARIMAX model incorporating sentiment and technical indicators with or without macroeconomic inputs, and a single-layer LSTM (recurrent neural network) using a one-month lookback window—were evaluated in both "with macro" and "without macro" configurations. Forecast accuracy was assessed out-of-sample using RMSE, MAE, MAPE and R^2, and a full factorial evaluation across all combinations of sentiment methodology, indicator set and model type was conducted to identify the optimal pipeline, taking into account predictive performance and computational cost (training time and resource usage). Intraday or

high-frequency modeling, transformer-based time-series architectures and any real-time trading deployment were excluded from this study's scope.

1.6 Dissertation Outline

This dissertation is structured in five chapters, each step in the research presents a methodological approach, which leads us to the overall sentiment- and macro-augmented forecasting of Tesla stock prices.

Chapter 1 provides the theoretical foundation of this study by introducing the forecasting problem, highlighting its importance in the context of investment managers and fintech practitioners, reviewing the relevant literature. It goes on to describe the problem statement by identifying gaps in combining lexicon, machine-learning, and deep-learning sentiment with exogenous indicators and how this study will help to fill these gaps. In the end it provides the summary of those chapters as well.

Chapter 2 provides an extensive literature background. It also reviews the classical time-series models (ARIMA/ARIMAX) and recurrent neural networks (LSTM), lexicon-based (VADER, Loughran–McDonald), machine learning (Logistic Regression, SVM, Naive Bayes, Random Forest, XGBoost), and deep learning (FinBERT, RoBERTa, DistilBERT) sentiment techniques, at the same time considers macroeconomic and technical indicators. Through critical review, this chapter enables the identification of what are the main findings, knowledge gaps, and the way it sets the basis for the methodology.

Chapter 3 describe about the methodology. It explains how the retrieval of data, from GDELT-sourced headlines to Yahoo Finance prices, and FRED macro variables, and also the preprocessing stages (translation, deduplication, and aggregation). Then, it describes sentiment feature engineering process (lexicon scoring, classifier training on Financial PhraseBank, embedding extraction), and choice of macro and technical

indicators as well as what goes into making an ARIMAX and LSTM model (train/test splits and evaluation by RMSE, MAE, MAPE and R^2).

Chapter 4 is the core analytical of the study, where out-of-sample performance of each "sentiment × indicator × model" pipeline is reported. It investigates the role of macro data in ARIMAX vs. LSTM, compares the performance of lexicon, ML and DL sentiment methods, and considers the trade-off between accuracy and computational time. The discussion in detail singles out the pipelines that achieve the research goals most effectively.

Chapter 5 summarizes the research results, measures how the goals and objectives have been met and discusses the limitations. It provides actionable suggestions, e.g., "use ML→ARIMAX for macro-aware forecasting" and "use RF→LSTM for minimal RMSE," and future work directions, such as deploying models in real-time, and alternative embedding architectures.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter describes the previous research study or journal. In Sec four thematic streams: data-acquisition & preprocessing, sentiment-analysis paradigms, exogenous indicators, and forecasting models are surveyed in order to position our study and pinpoint the gap that our head-to-head comparative evaluation of Tesla stock forecasting is to address.

2.2 Data Acquisition and Preprocessing

Existing research highlights the significance of the quality of text and numerical data for financial forecasting. The GDELT API is a well-known database that provides a large volume of easily collectable structured data to aggregate news tone and metadata. Various papers have shown GDELT to be useful to predict macroeconomic variables. For example, Tilly, Ebner, and Livan (2020) discuss how monthly GDELT-derived sentiment can predict industrial production and consumer prices. In the context of equity markets, Ren, Wu, and Liu (2019) use pure, which means minimal cleaning of news headline sentiment data in combination with social media data for the purpose of stock movement direction prediction; Moghar and Hamiche (2020) gather price data from Yahoo Finance, in turn used to train LSTM models for NYSE stocks. Macroeconomic variables that are typically taken from the FRED database have been known for a long time to be associated with returns in ARIMAX research (Chen et al., 1986).

Nonetheless, in current research, preprocessing steps are often applied basic preprocessing, such as lowercasing, stop-word removal, but rarely account for duplicated or syndicated headlines that may introduce a bias in monthly sentiment aggregates. Christen (2009), Bird, Klein, and Loper (2009) show that fuzzy-matching and lemmatization techniques enhance reliability of text-mining but are not widely used in financial-news forecasting.

Conversely, relatively little empirical research goes through the process of carefully determining which macro variables should be most highly correlated with the target asset prior to being incorporated into the model.

Although GDELT-based sentiment, Yahoo price data and as well as macroeconomic data from FRED are typical inputs, there is no systematic assessment of preprocessing strategies, specifically the deduplication, and the variable selection procedure. It is important to fix this gap to guarantee that sentiment measures accurately capture idiosyncratic news signals and that macro indicators are meaningfully connected to the forecasting target.

2.3 Sentiment Analysis Methods

Sentiment analysis techniques ranges from basic word-matching approaches to state-of-the-art machine-learning and deep-learning models. In the domain of financial prediction, the ability to accurately quantify textual sentiment is desirable for identifying market mood shifts and predicting price changes (Ren, Wu, & Liu, 2019). In the Section 2.3, there are three main paradigms: lexicon-based, machine-learning classifiers and deep-learning embeddings will be discussed, including their principle findings, strengths, and weaknesses documented in the literature.

2.3.1 Lexicon-Based Approaches

Lexicon-based sentiment analysis aims to quantify sentiment in the text by matching words or phrases with pre-defined sentiment dictionaries. One commonly used sentiment lexicon is VADER (Valence Aware Dictionary and Sentiment Reasoner), a

sentiment lexicon that annotates the intensity as well as the polarity of a sentiment using a rule-based heuristic approach based on a manually annotated sentiment lexicon (Hutto & Gilbert, 2014). It has shown good performance on social media and general domain texts as it is able to identify contextual sentiment enhancers and negations.

Another well-known dictionary constructed specifically for financial text is the Loughran–McDonald dictionary that classifies financial words as positive, negative, uncertainty, litigious, and constraining. Loughran and McDonald (2011) made a strong case that general lexicons often miscategorized financial text, alluding to the necessity of a tailor-made financial dictionary.

Despite being transparent and computationally cheap, lexicon-based methods are static and harder to adapt to new vocabulary, idiomatic language or market-specific changes (Kogan et al., 2009).

2.3.2 Machine-Learning Classifiers

To overcome some of the drawbacks of lexicon approaches, supervised ML classifiers which learn sentiment using label financial corpora are trained, have been proof that it can achieve more generalization than lexicon methods. The Financial PhraseBank dataset includes about 4,840 financial-news sentences labeled at divergent agreement levels (Malo et al., 2014), is a well-known training corpus.

Malo *et al.* (2014) partition their Financial PhraseBank into four subsets according to annotator agreement—100 %, > 75 %, > 66 % and > 50 %. As Figure 2-1 illustrates, requiring complete consensus yields 2 259 sentences, of which 13.4 % are labeled negative, 61.4 % neutral and 25.2 % positive. Relaxing the threshold to > 75 % increases the corpus to 3 448 sentences (12.2 % negative, 62.1 % neutral, 25.7 % positive); at > 66 %, there are 4 211 sentences (12.2 % negative, 60.1 % neutral, 27.7 % positive); and at > 50 %, 4 840 sentences (12.5 % negative, 59.4 % neutral, 28.2 % positive). This stepwise relaxation demonstrates how higher agreement levels constrain dataset size while only modestly shifting the balance of sentiment labels.

TABLE 3. Distribution of labels in phrase bank for four subsets formed based on the strength of majority agreement.

	% Negative	% Neutral	% Positive	Number of sentences
Sentences with 100% agreement	13.4	61.4	25.2	2259
Sentences with >75% agreement	12.2	62.1	25.7	3448
Sentences with >66% agreement	12.2	60.1	27.7	4211
Sentences with >50% agreement	12.5	59.4	28.2	4840

Note. Each sentence has five to eight overlapping annotations, which have been used to determine the degree of agreement.

Figure 2-1 Distribution of sentiment labels by annotator agreement threshold. Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts. Source: Malo, Sinha, Korhonen, Wallenius & Takala (2014).

Text is vectorised with TF-IDF / Bag-of-Words /Word2Vec embeddings before being trained by algorithms including Support Vector Machines, Random Forests, Naive Bayes, XGBoost and Logistic Regression. Empirical results demonstrate that these ML algorithms also outperform lexicon-based approaches and achieve significant classification accuracy and sentiment discrimination performance in financial domains (Nassirtoussi et al., 2014; Ren et al., 2019).

However, ML classifiers require feature selection, hyperparameter tuning, and rely on labeled data availability and quality. Consequently, their empirical performance can be greatly dependent on market conditions and dataset.

2.3.3 Deep-Learning Embeddings

The advances in natural language processing lately have resulted in deep-learning-based methodologies, in particular transformer-based embedding models, which can produce strongly context dependent text representations. Recent approaches like FinBERT (Araci, 2019), which is a BERT model fine-tuned specifically on financial texts, improve the accuracy in classifying sentiment over classic ML methods. Transformer-like models such as RoBERTa (Liu et al., 2019) and DistilBERT (Sanh, Wolf, & Rush, 2019) have been able to achieve state-of-the-art performances on

various NLP benchmarks as well by enabling the capture of complex linguistic structures and contexts.

However, in spite of their significant advantages in semantic and contextual understanding, these deep-learning models have a high computational cost. Model training and deployment may need intensive computing resources and time, which can be a challenge in practice, especially in near real-time prediction cases (Ding et al., 2015). Additionally, these approaches can face a scenario of "diminishing returns" in smaller datasets, where fine-tuning significantly for the target dataset, which requires the extensive knowledge of domain-specific topics, may result in poor performance (Nassirtoussi et al., 2014).

2.4 Exogenous Indicators in Stock Forecasting

Besides text-based sentiment indicators, accurate prediction of stock prices required to incorporate with exogenous indicators, in particular, macroeconomic and technical variables. It has been highlighted in several works that enriching the analysis of text sentiment with such, and others, signals can increase the prediction performance by taking into account more general economic movements and market mechanisms that could be neglected considering only news sentiment (Chen, Roll, & Ross, 1986; Nassirtoussi et al., 2014). This section provides a critical overview of empirical evidence with respect to the relevance and shortcomings of macroeconomic and technical indicators in financial prediction tasks.

2.4.1 Macroeconomic Indicators

Macroeconomic indicators, such as Gross Domestic Product (GDP), Consumer Price Index (CPI), the unemployment rate and Consumer Confidence, are known to be important determinant of the equity returns and volatility. Chen, Roll, and Ross (1986) provided solid evidence of the importance of macroeconomic factors to explain the variance of stock returns, with factors such as growth of GDP, inflation, interest rates being consistently having explanatory power to market movements. Subsequent studies further demonstrate the forecasting ability of the macroeconomic factors. For

instance, Flannery and Protopapadakis (2002) demonstrate the market response influenced by inflation and industrial production, affecting both prices and volatility.

Nonetheless, incorporating macroeconomic variables into forecasting models is difficult because of their typically low-frequency (monthly or quaterly) may make them less useful for high-frequency trading decisions (Ghysels, Sinko & Valkanov, 2007). Additionally, these indicators can be caused delay in market responses, muddling short-term predictions. However, the inclusion of macroeconomic indicators remained necessary for improving the forecasting accuracy for medium and long term.

2.4.2 Technical Indicators

Technical indicators are derived from historical price and volume data, display the history and direction of the market. Various indicators, including RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), momentum, volatility, and several moving averages, have been widely used in academic studies and practical trading systems (Achelis, 2001; Murphy, 1999).

There is empirical evidence to support this assertion. Brock, Lakonishok and LeBaron (1992) found that moving averages and trading-range breaks were statistically significant predictors of the Dow Jones Industrial Average. Similarly, Chong, Han and Park (2017) find technical indicators prevail against random walk in various stock markets, revealing the profitability of their prediction on the actual life-cycle trading. Notwithstanding their empirical efficacy, technical indicators are fundamentally lagging and may be not universally effective in fast-paced markets undergoing structural transitions or abrupt regime shifts.

2.4.3 Fusion of Sentiment, Macro and Technical Indicators

Hybrid forecast methodologies, that combine textual sentiment with macroeconomic and technical indicators, have received growing attention recently for their potential to exploit the strengths of both types of input. Nassirtoussi et al. (2014) found enhanced predictability from incorporating loud news-derived sentiment and latent technical

signals. Kara, Boyacioglu, and Baykan (2011) presented that a neural network forecasting of Istanbul Stock Exchange indices is improved by adding the information of technical indicators as well as two groups of external variables, macroeconomic news, and sentiments.

However, most of studies in the literature have used few indicators or sentiment extraction methods, and it is rare to find comparisons of Lexicon, ML, and DL over textual sentiment paradigms, all together with different exogenous indicators in a wide scale. This methodological fragmentation leaves considerable ambiguity about the best set-up of integrated sentiment-macro-technical forecasting models.

2.5 Forecasting Frameworks in Stock Market Prediction

Stock price prediction often needs an established methodology to model stock price movements. By using appropriate predictive models, it can adequately detect underlying market dynamics. There are two main approaches have been followed in financial prediction, for which provide brief overview on the classical linear statistical (ARIMA) and its extension ARIMAX (Autoregressive Integrated Moving Average with exogenous inputs) and the nonlinear machine-learning approaches, particularly Long Short-Term Memory (LSTM). This section examines these approaches, and exposes their theoretical rationale, evidence, added-values, and potential floors pushing for their joint analysis in the present study.

2.5.1 ARIMA and ARIMAX Models

The Autoregressive Integrated Moving Average (ARIMA) model developed by Box and Jenkins (1970) is a classical technique for time-series data analysis and forecasting with linear dependence and stationarity. ARIMA models contain AR components and MA terms to process non-stationarity, thus enabling them to capture temporal dependencies and trends in past data (Makridakis et al., 2008).

To extend ARIMA's forecast capabilities, ARIMAX includes external exogenous variables is introduced such as macroeconomic indicators or sentiment signals, it

allowing to account for the impact of external markets factors in addition to historical price movements alone. The empirical evidence indicates that the inclusion of macroeconomic or sentiment indicators in ARIMAX models positively impacts on the predictive accuracy as compared to pure ARIMA models. For example, Fang and Peress (2009) find significant gains of equity-returns predicting accuracy from augmenting linear forecasting models with news-driven sentiment variables. However, even though ARIMAX offers interpretable results and low computational cost, it assumes linearity and stationarity by nature, which limits the application when the market is non-linear or volatile (Adhikari & Agrawal, 2013).

2.5.2 Long Short-Term Memory (LSTM) Networks

In recent years, machine learning methods, particularly deep neural networks (DNNs), have received extensive attention owing to their capability of nonlinear model. Long Short-Term Memory (LSTM) networks, which are a subset of Recurrent Neural Networks, were introduced by Hochreiter and Schmidhuber (1997) specifically to alleviate standard RNN problems, such as the vanishing and exploding gradients, by adding gating mechanism to selectively retain or drop information over time.

LSTMs are good at capturing complex temporal dependencies and non-linearities that exist in financial markets. Fischer and Krauss (2018) demonstrated that an LSTM outperformed the random forest and logistic regression in predicting the short-term returns for the S&P 500 firms. Moghar and Hamiche (2020) also more recently, reported that LSTMs are powerful in predicting stock market, and that they are efficient in capturing complex structure in price time series.

Yet, LSTMs are complex and flexible, and pay for that with an enormous computational burden, large amounts of data required and very limited interpretability in relation to traditional statistical models (Chong, Han, & Park, 2017). Moreover, the performance of LSTM networks may be weakened when incorporating lower-frequency macroeconomic indicators, since these models are generally accustomed to better perform taking into account higher-frequency data inputs (Fischer & Krauss, 2018).

2.5.3 Linear vs Nonlinear Forecasting Approaches: Integrative Perspective

Considering the strong and weak points of each, mostly many literatures suggest to combine linear models (ARIMAX) with nonlinear models (LSTM) to improve forecasting performance and reliability. The approach of hybrid forecasts benefits from the ARIMAX's interpretative linearity for macroeconomic effects and the LSTM's learning characteristics benefit from complex market structure (Zhang, 2003; Kara, Boyacioglu, & Baykan, 2011). Despite the potential advantages of such combined approaches, a systematic comparison of exploiting ARIMAX and LSTM, more specifically when paired to the lexicon-, machine-learning-, and deep-learning-based sentiment indicators with rich macroeconomic and technical variables, is still conspicuously absent.

2.6 Integrating Sentiment with Macroeconomic and Technical Indicators

Study has been demonstrated that extending the traditional forecast model with sentiment or exogenous indicators can significantly improve forecast accuracy. For instance, with Table III, Fang and Peress (2009) show that, when firms have no media coverage at all, they obtain a monthly average return of 1.35%; this monthly average return decreases to 0.96% for high-coverage firms, which illustrates impact of the simple “new-tone” variables may shift the return prediction when added to an ARIMA model.

Figure 2-2 shows how newspaper coverage relates to cross-sectional stock returns. Fang and Peress (2009) divide firms into “No,” “Low” and “High” coverage groups and find that stocks with high coverage earn higher average monthly returns than those with no coverage, with a statistically significant No–High t-statistic. They further report the average number of firms in each group overall and within size, book-to-market, past-month return, current-month return and price quintiles, illustrating that media attention is disproportionately focused on larger, high-return firms.

	Average Monthly Return					Average No. of Stocks		
	Media Coverage				t-Statistics for	Media Coverage		
	No	Low	High	No − High	No − High	No	Low	High
All stocks	1.35	1.11	0.96	0.39	2.13	1,430.08	284.82	245.40
Panel A: By Size								
1	1.41	1.02	0.53	0.88	1.74	578.44	56.36	17.98
2	1.34	1.12	0.69	0.65	2.68	514.23	92.85	46.71
3	1.27	1.16	1.10	0.17	1.03	337.42	167.12	149.19
Panel B: By Book-to-Market								
1	1.19	0.95	0.87	0.32	1.25	441.79	93.98	81.50
2	1.23	1.17	0.54	0.70	3.13	450.03	92.90	74.64
3	1.42	1.10	1.19	0.22	0.85	460.78	86.20	70.57
Panel C: By Past Month Return								
1	1.43	0.98	0.85	0.58	2.29	474.54	93.31	76.78
2	1.25	0.89	0.91	0.34	1.73	461.16	97.56	82.13
3	1.09	1.07	0.64	0.44	2.19	467.33	93.25	78.77
Panel D: By Current Month Return								
1	1.96	1.49	0.86	1.10	4.24	479.50	94.11	75.77
2	1.29	1.08	1.01	0.28	1.36	475.29	94.21	79.54
3	0.88	0.75	1.08	−0.20	−0.76	467.93	100.31	84.40
Panel E: By Price								
1	1.01	0.53	−0.11	1.11	3.14	545.18	69.76	35.76
2	1.39	1.12	0.54	0.84	3.77	500.77	94.03	60.14
3	1.77	1.47	1.35	0.42	2.62	384.13	142.30	128.21

Figure 2-2 Average monthly returns and firm counts across "No," "Low" and "High" newspaper coverage categories. Media Coverage and the Cross-Section of Stock Returns. Source: Fang & Peress (2009).

Nassirtoussi et al., 2014 review that 79 text-mining systems and mention the hybrid models, which use either or both lexicon-based and ML-based sentiment to momentum or volatility signals, are proof to be consistently outperform sentiment-only, and technical-only models. However, they also notice a scarcity of a single head- to-head comparison of the various sentiment paradigms.

In the domain of artificial neural network-based machine learning, Kara, Boyacıoğlu & Baykan (2011) provide ten technical indicators (e.g., RSI, moving-average crossovers) and a lexicon-based sentiment index as inputs of both ANN and SVM models as well. The ANN is 75.74% while the SVM is 71.52 % showing the benefit we obtain by including sentiment features.

Figure 2-3 show Kara, Boyacıoğlu & Baykan (2011) compile every trading day of the ISE 100 Index from January 2, 1997 through December 31, 2007, yielding 2 733

observations. Of these, 1 440 days (52.7 %) saw the index close higher ("up" days) and 1 293 days (47.3 %) saw it close lower ("down" days). Year-by-year breakdowns demonstrate how both the count and proportion of up vs. down days vary over time, providing a balanced dataset for training and testing their ANN and SVM models.

Table 1
The number of cases in the entire data set.

	Year											Total
	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	
Increase	146	124	133	111	123	123	134	142	147	131	126	1440
%	57.9	50	56.4	44.9	49.6	48.8	54.5	57	57.9	52.4	50.2	52.7
Decrease	106	124	103	136	125	129	112	107	107	119	125	1293
%	42.1	50	43.6	55.1	50.4	51.2	45.5	43	42.1	47.6	49.8	47.3
Total	252	248	236	247	248	252	246	249	254	250	251	2733

Figure 2-3 Distribution of "up" vs. "down" days in the ISE 100 Index dataset by year. Predicting Direction Of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines. Source: Kara, Boyacıoğlu & Baykan (2011).

As shown in Figure 2-4, Kara, Boyacıoğlu & Baykan (2011) feed ten widely used technical indicators into both their artificial neural network and support vector machine predictors. These include the simple and weighted 10-day moving averages to smooth price trends; momentum, which measures price change over a fixed interval; the stochastic %K and %D oscillators that compare closing prices to recent highs and lows; the relative strength index (RSI), which assesses average gains versus losses; the moving average convergence divergence (MACD), calculated as the difference between two exponential moving averages; Larry Williams's %R, reflecting price position within a look-back period; the accumulation/distribution (A/D) oscillator, which relates price moves to volume; and the commodity channel index (CCI), which gauges price deviation from its statistical mean. Each indicator is defined by a precise mathematical formula, enabling the models to capture diverse dimensions of market momentum, volatility and sentiment.

Table 4
Selected technical indicators and their formulas.

Name of indicators	Formulas
Simple 10-day moving average	$\frac{C_t+C_{t-1}+\cdots+C_{t-10}}{10}$
Weighted 10-day moving average	$\frac{((n)\times C_t+(n-1)\times C_{t-1}+\cdots+C_{10})}{(n+(n-1)+\cdots+1)}$
Momentum	$C_t - C_{t-n}$
Stochastic K%	$\frac{C_t-LL_{t-n}}{HH_{t-n}-LL_{t-n}} \times 100$
Stochastic D%	$\frac{\sum_{i=0}^{n-1} K_{t-i}\%}{n}$
RSI (Relative Strength Index)	$100 - \frac{100}{1+\left(\sum_{i=0}^{n-1} Up_{t-i}/n\right)/\left(\sum_{i=0}^{n-1} Dw_{t-i}/n\right)}$
MACD (moving average convergence divergence)	$MACD(n)_{t-1} + 2/n + 1 \times (DIFF_t - MACD(n)_{t-1})$
Larry William's R%	$\frac{H_n-C_t}{H_n-L_n} \times 100$
A/D (Accumulation/Distribution) Oscillator	$\frac{H_t-C_{t-1}}{H_t-L_t}$
CCI (Commodity Channel Index)	$\frac{M_t-SM_t}{0.015D_t}$

C_t is the closing price, L_t the low price, H_t the high price at time t, $DIFF$: $EMA(12)_t - EMA(26)_t$, EMA exponential moving average, $EMA(k)_t$: $EMA(k)_{t-1} + \alpha \times (C_t - EMA(k)_{t-1})$, α smoothing factor: $2/1 + k$, k is time period of k day exponential moving average, LL_t and HH_t mean lowest low and highest high in the last t days, respectively, $M_t : H_t + L_t + C_t/3$; $SM_t : (\sum_{i=1}^{n} M_{t-i+1})/n$, $D_t : (\sum_{i=1}^{n} | M_{t-i+1} - SM_t |)/n$, Up_t means the upward price change, Dw_t means the downward price change at time t.

Figure 2-4 Selected technical indicators and their formulas. Predicting Direction of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines. Source: Kara, Boyacıoğlu & Baykan (2011).

In Figure 2-5, Kara, Boyacıoğlu & Baykan (2011) compare the directional accuracy of their ANN and polynomial SVM models across 11 hold-out runs. The neural network attains an average accuracy of 75.74 % (SD 2.49), while the SVM achieves 71.52 % (SD 6.18). A paired t-test yields t = 3.098, p = 0.011, providing strong evidence that the ANN significantly outperforms the SVM in predicting the daily direction of the ISE 100 Index.

Table 13
t-Test results of model comparison.

Prediction model	*N*	Mean	Std. dev.	*t*	*p*
ANN	11	75.74	2.49	3.098	0.011
Polynomial SVM	11	71.52	6.18		

Figure 2-5 Paired t-test comparing hold-out directional accuracies of ANN and polynomial SVM. Predicting Direction of Stock Price Index Movement Using Artificial Neural Networks and Support Vector Machines. Source: Kara, Boyacıoğlu & Baykan (2011).

Ren, Wu, & Liu (2019) create daily sentiment indices from news tone and join them with inflation and GDP-growth data into an SVM classifier. They report a hit rate on the SSE 50 Index of 89.93%, which 18.6% uplift over a market-data-only benchmark.

Ren, Wu & Liu (2019) compare two forecasting experiments using support vector machines and a logistic-regression baseline. In Experiment 1, models rely solely on market variables (opening, high, low, closing prices; trading volume; RMB price change; percentage change), whereas Experiment 2 augments these inputs with daily GDELT-derived sentiment features. Panel A reports five-fold cross-validation accuracies for the SVM: 79.96 % ($C = 256$, $\gamma = 0.9942$) in Experiment 1 rising to 97.73 % ($C = 181.02$, $\gamma = 0.0055$) in Experiment 2. Panel B shows rolling-window SVM performance of 71.33 % (window = 68) and 89.93 % (window = 76) for the two experiments. Panel C presents logistic-regression five-fold CV results: 70.96 % accuracy using gradient descent (Max = 600) in Experiment 1 and 86.59 % using stochastic gradient descent (Max = 1000) in Experiment 2.

TABLE III
PREDICTION RESULTS

Panel A: Prediction accuracy of support vector machine (SVM) with fivefold cross validation			
	Accuracy	C	γ
Experiment 1	0.7996	256	0.9942
Experiment 2	0.9773	181.0193	0.0055

Panel B: Prediction accuracy of SVM with rolling windows				
	Accuracy	C	γ	Rolling window
Experiment 1	0.7133	256	0.9942	68
Experiment 2	0.8993	181.0193	0.0055	76

Panel C: Prediction accuracy of logistic regression (LR) with fivefold cross validation			
	Accuracy	Max	Optimization
Experiment 1	0.7096	600	Gradient descent (GD)
Experiment 2	0.8659	1000	Stochastic GD

Figure 2-6 Model prediction accuracies under five-fold cross-validation and rolling-window evaluation for SVM and logistic regression on market-only vs. market+sentiment feature sets. Forecasting Stock Market Movement Direction Using Sentiment Analysis and Support Vector Machine. Source: Ren, Wu & Liu (2019).

From the macroeconomic point of view, Tilly, Ebner & Livan (2020) collapse more than 2,300 GDELT emotion/theme scores into three latent sentiment factors and extend monthly ARIMA models for industrial production (Table 6) and CPI (Table 7). They report a reduction of 10% for the forecast RMSE when inclusion of these sentiment factors were made along with the standard macro factors.

Figure 2-7 reports the RMSE percentages for ARIMA-based forecasts of monthly industrial production (IP) across ten economies. Four model specifications are compared: BM1, the autoregressive benchmark using IP and standard macro variables; BM2, which adds unfiltered GDELT sentiment factors; BM3, which adds the average GDELT tone score; and PLS-ARIMAX, which incorporates three PLS-filtered GDELT sentiment factors. Blue shading highlights cells where the sentiment-augmented model outperforms the benchmark, while red shading denotes underperformance. Parenthesized numbers indicate the relationship of many sentiment factors are statistically significant in the regression ($p < 0.01, 0.05, 0.1$). Overall, including filtered sentiment factors reduces forecast RMSE by roughly 10 % on average relative to the unaugmented benchmarks.

Table 6

Results of the model in Eq. (1) applied to IP. Numbers represent the RMSE (%). Blue (red) cells denote cases in which the model outperforms (underperforms) the benchmark. Numbers in parentheses correspond to the number of significant coefficients associated with GDELT factors in the model in Eq. (1) (* * * denotes at least one GDELT sentiment factor with p-value < 0.01, ** < 0.05, * < 0.1).

Data set / IP for	Model	BM1	BM2	BM3	Sign.
US	1.6348	1.6439	1.6527	1.6507	***(1),**(1), *(1)
UK	2.4663	2.4870	2.5065	2.4887	***(1)
Germany	2.7957	2.7978	2.7870	2.8032	**(2)
Norway	2.2907	2.2647	2.3410	2.2720	**(1)
Poland	7.6308	7.7321	7.8420	7.7351	***(1)
Turkey	4.5785	4.5836	4.6780	4.5843	**(2)
Japan	2.2138	2.2429	2.2571	2.2429	***(1), **(1)
South Korea	2.4776	2.5079	2.5579	2.5211	***(1)
Brazil	4.0484	4.0942	4.1341	4.0941	**(1)
Mexico	3.2630	3.2494	3.2532	3.2471	***(1)

Figure 2-7 RMSE (%) of ARIMA forecasts of monthly industrial production with and without sentiment factors. Macroeconomic Forecasting Through News, Emotions and Narrative. Source: Re Tilly, Ebner & Livan (2020).

Figure 2-8 reports the RMSE percentages for ARIMA-based forecasts of monthly consumer price index (CPI) across ten economies under four model variants: BM1 (autoregressive benchmark using CPI and standard macro variables), BM2 (BM1 + unfiltered GDELT sentiment factors), BM3 (BM1 + average GDELT tone), and PLS-ARIMAX (BM1 + three PLS-filtered GDELT sentiment factors). Blue shading highlights cases where a sentiment-augmented model outperforms BM1, while red shading denotes underperformance. Numbers in parentheses indicate how many filtered sentiment factors are statistically significant in the regression ($p < 0.01$, 0.05, 0.1). Overall, incorporating PLS-filtered sentiment reduces forecast RMSE by roughly 10 % on average compared to the unaugmented benchmarks.

Table 7
Results of the model in Eq. (1) applied to CPI. Numbers represent the RMSE (%). Blue (red) cells denote cases in which the model outperforms (underperforms) the benchmark. Numbers in parentheses correspond to the number of significant coefficients associated with GDELT factors in the model in Eq. (1) (* * * denotes at least one GDELT sentiment factor with p-value < 0.01, ** < 0.05, * < 0.1).

Data set / CPI for	Model	BM1	BM2	BM3	Sign.
US	0.2051	0.2031	0.2165	0.2038	***(1)
UK	0.3212	0.3118	0.3258	0.3193	***(2)
Germany	0.4117	0.4316	0.4288	0.4279	***(1)
Norway	0.4715	0.4727	0.4706	0.4640	
Poland	0.3321	0.3383	0.3649	0.3512	***(1)
Turkey	1.0194	1.0263	1.0513	1.0685	***(2), **(1)
Japan	0.2464	0.2485	0.2538	0.2524	***(2)
South Korea	0.3941	0.4055	0.4057	0.4070	**(1)
Brazil	0.3876	0.3993	0.4165	0.3946	***(1)
Mexico	0.4226	0.4349	0.4340	0.4581	***(1)

Figure 2-8 RMSE (%) of ARIMA forecasts of monthly CPI with and without filtered sentiment factors. Macroeconomic Forecasting Through News, Emotions and Narrative. Source: Re Tilly, Ebner & Livan (2020).

Nonetheless, to the best of our knowledge, no existing study has benchmarked lexicon-based, ML-based and DL-based sentiment representation learning on the same experimental setup, to be let alone comparing ARIMAX with LSTM on the same dataset.

2.7 Research Gap

Although sentiment analysis, macroeconomic indicators, and technical variables have each been shown to improve stock-price forecasts, the literature remains fragmented in several respects. First, most hybrid studies examine only one sentiment paradigm. It is whether lexicon-based (Fang & Peress, 2009), ML-based (Nassirtoussi et al., 2014), or DL-based (Moghar & Hamiche, 2020), without directly comparing their relative merits under identical conditions. Second, most studies only look at ARIMAX and LSTM models on their own, instead of comparing them on the same feature sets (Tilly, Ebner, & Livan, 2020; Fischer & Krauss, 2018). Third, systematic hyperparameter tweaking and head-to-head performance tests are uncommon, especially when sentiment and exogenous indicators are mixed. This makes it hard to tell which combination of sentiment, macro, and technical inputs gives the greatest out-of-sample accuracy.

This research fills the gap by comparing lexicon-, ML-, and DL-based sentiment extraction approaches in a unified and carefully adjusted way, by using Tesla stock as a test case. It will be done by combining them with a wide range of macroeconomic and technical indicators inside both the ARIMAX and LSTM frameworks.

2.8 Chapter Summary

Chapter 2 reviewed four core parts that underpin this research. Section 2.2 included data extraction and cleaning, such as the use of GDELT, Yahoo Finance, FRED data, and best practices in text cleaning and deduplication. Section 2.3 reviewed the types of sentiment analysis, namely lexicon-based scoring, supervised ML classifiers, and transformer-based embeddings, while setting out these method developments, empirical strengths, and practical shortcomings. In Section 2.4, exogenous indicators were covered, and the roles of several macroeconomic variables and technical indicators in predictability were briefly summarized. In Section 2.5, the prediction models were analyzed, pitting the linear ARIMAX model against the nonlinear LSTM model, but also indicating that effective combinations are potentially achievable. However, a full head-to-head comparison across all three sentiment paradigms

alongside macro and technical predictors under both ARIMAX and LSTM models still does not exist by far. In Chapter 3, the solution to this gap, collection and data measurement, model features engineering, model setup, and valuation protocol will be described.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In this chapter, the analysis of methodology will be modeled, which consists of examining the association between sentiment analysis and stock prediction, particularly for the case of Tesla stock. The technique incorporates different techniques for the purpose of robust and richer analysis in terms of data collection, pre-processing, sentiment analysis methods, feature engineering, and predictive models.

First, the use of quantitative and predictive modeling tools will be proposed to discover and estimate market sentiments derived from financial news headlines and social media. This study particularly includes lexicon-based approaches (Loughran-McDonald and VADER), machine learning-based classifiers (such as Logistic Regression, Support Vector Machines, Naive Bayes, Random Forest and XGBoost), and deep learning models based on transformer architecture, including FinBERT, RoBERTa, and DistilBERT.

This chapter also describes the procedures behind collecting the data from different sources (e.g., Yahoo Finance for financial data, GDELT for the sentiment-rich news data) and the careful pre-processing steps in order to improve the quality of the data and of the model. After presenting data preprocessing, the chapter discusses in detail what was done for feature engineering, introducing lagged variables, rolling means and interacting predictors, to enhance the predictive ability of the data sets.

Finally, the chapter provides a deep overview of specific models, including ARIMAX-LSTM models, the respective tuning process for its parameter settings, the metrics used for evaluation, and validation methodologies. Together, these components form a coherent approach to achieving the research objectives of this paper.

3.2 Research Design

This investigation follows a quantitative predictive research design which is centered on the impact of sentiment analysis and market indicators on the prediction of stock price movements. The quantitative methodology is chosen for its ability to statistically confirm relationships between variables and make accurate quantitative predictions that is very important for financial decision makers.

The study design will be constructed in a sequential manner with data collection performed from reliable sources for the development of a well-pre-processed, cleaned dataset. Sentiment analysis methods are then leveraged, utilizing rigid lexicon-based and more advanced machine learning and deep learning-based techniques to extract fine-grained feeling signals from the text.

After sentiment is extracted, an extensive feature engineering is performed to get the appropriate predictors. This encompasses construction of lagged sentiment values, rolling statistics and interacted features. Then predictions are made through ARIMAX models to cover linear dependencies and LSTM to reconcile nonlinear temporal patterns of stock prices.

Cross-validation techniques such as in-sample (IS) and out-of-sample (OOS) testing, and measures such as RMSE, MAPE, R-squared provide robust evaluation on the model's performance. Such research framework allows us to methodically study and verify sentiment-based stock price prediction, meeting the research goals.

3.3 Data Collection

3.3.1 Data Sources

In this study, three different types of time-series data are employed: (1) The historical market price of Tesla, (2) Prominent United States macroeconomic indicators, and (3) News-driven investor sentiment metrics. All the series are between January 2020 and December 2023 in order to maintain the same synchronization between daily and monthly based data sets.

A. Tesla Market Prices

Daily closing prices for Tesla, Inc. (TSLA) are fetched from the Yahoo Finance API, using the *yfinance* Python package. For each trading day t, the opening price O_t, high H_t, low L_t, close price C_t, adjusted close price C_t^*, and the trading volume V_t were collected. The primary dependent variable in subsequent models is the log-return:

$$r_t = \ln(C_t^*) - \ln(C_{t-1}^*), \tag{3.1}$$

which normalizing price changes and stabilizing variance, for ARIMAX and LSTM modelling.

B. Macroeconomic Indicators

For each of the 13 macroeconomic time series, we collect monthly observations from the Federal Reserve Economic Data (FRED) repository using the *fredapi* package. Each series $X_{k,t}$ corresponds to the indicator k in month t. The selection of the indicators and their FRED series codes are:

Table 3-1 The 13 candidate macroeconomic indicators that might related to Tesla Close price (Section 3.3.1).

Category	Indicators (FRED Code)
Inflation & Prices	Consumer Price Index (CPIAUCSL)
	Personal Consumption Expenditures (PCE)
	Producer Price Index (PPIACO)
Labor Market	Unemployment Rate (UNRATE)
	Non-Farm Payrolls (PAYEMS)
	Job Openings (JTSJOL)
Output & Growth	Gross Domestic Product (GDP)
	Industrial Production (INDPRO)
	Retail Sales (RSAFS)
Monetary Policy & Liquidity	Federal Funds Rate (FEDFUNDS)
	10-Year Treasury Yield (GS10)
	M2 Money Supply (M2SL)
Sentiment & Confidence	Consumer Confidence Index (UMCSENT)

The above variables are justified as determinants of equity returns in existing literature (e.g., Chen et al., 1986, Bernanke & Blinder, 1992) and we later use the stationary differenced series $\Delta X_{k,t}$, $\Delta^2 X_{k,t}$ to meet the ARIMA pre-requisite.

C. News-Based Sentiment

Daily news that containing the keyword "Tesla" obtained using the GDELT Document API (artlist mode) for the time range from January 2020 to December 2023 with up to 250 elements per calendar day. Instead of immediately performing an aggregation, each raw title is first processed through the complete cleaning pipeline described in 3.3.2 (translation to English, punctuation and stop-words removal, lemmatization,

fuzzy matching) and then aggregated. Only after a headline $H_{d,i}$ published on day d has been validated through this pipeline, it will be assigned a sentiment score $s_{d,i}$ (e.g., of the VADER lexicon). The final monthly sentiment index S_t is then created as the average over all cleaned, scored headlines in the month t:

$$S_t = \frac{1}{N_t} \sum_{d \in t} \sum_{i=1}^{N_d} s_{d,i}, \tag{3.2}$$

Where N_d is the number of cleaned headlines up to day d and $N_t = \sum_{d \in t} N_d$. These daily sentiment values will then be aligned with Tesla's trading-day by mapping the sentiment values onto market dates using forward-fill of non-trading days. The feature engineering process and the alignment process are described in detail in Section 3.4.

3.3.2 Data Extraction Methods

In this subsection, the retrieval and composition procedures of the three kinds of time-series will be presented as defined in 3.3.1. obtained programmatically. All the scripts were programmed in Python, and the output files were provided in CSV format for further preprocessing and modeling.

3.3.2.1 Tesla Price Data via Yahoo Finance

The yfinance library was use to download the daily OHLC (Open, High, Low, Close), Adjusted Close and Volume for Tesla, Inc. (TSLA) over the sample period (2020-01-01 to 2023-12-31). The API call:

tesla_stock = yf.download('TSLA', start='2020-01-01', end='2023-12-31')

returns a DataFrame indexed by trading datetime t. All six columns are saved using the CSV export. These spot prices are then will be apply for technical feature engineering later (Section 3.5).

3.3.2.2 Macroeconomic Series via FRED API

The U.S. macro data series available in the following Table 3.3.1 for the 13 macro indicators was programmatically Federal Reserve Economic Data (FRED) data repository by means of the *fredapi* Python package. For each series identifier s_k, we run:

series = fred.get_series(series_id, start='2020-01-01', end='2023-12-31')

and create a DataFrame indexed by date.

All returned time series were kept in a variable as a pandas DataFrame with time and are indexed by month of year. Once all twelve series have been fetched, these were stitched together on the column axis into a single DataFrame and written to CSV (economic_indicators_raw.csv) at which we will carry out future stationarity checking (Section 3.3.3.1), transformation and including them as exogenous regressors in the ARIMAX and LSTM models.

3.3.2.3 News Headlines via GDELT API

Daily "Tesla" Headlines are scraped via GDELT Document API in "artlist" mode, at maximum 250 records per day for the period January 1, 2020–December 31, 2023. For each d day in the sample:

- Construct query URL with

$$startdatetime = (d)00{:}00{:}00, enddatetime = (d+1)00{:}00{:}00.$$

- Send HTTP GET request and parse the CSV payload to a pandas DataFrame.

  Code:

  base_url = "http://api.gdeltproject.org/api/v2/doc/doc?query="

  query = (

```
        f"Tesla&mode=artlist&maxrecords=250&format=csv"
        f"&startdatetime={d.strftime('%Y%m%d000000')}"
        f"&enddatetime={(d+timedelta(days=1)).strftime('%Y%m%d000000')}"
    )
    response = requests.get(base_url + query)
```

- If not-empty daily DataFrames, it will be write out to $YYYY - MM - DD.csv$.

As each calendar month ends then daily files from that month are concatenated:

$$M_t = \bigcup_{d \in month\ t} H_d\ , \tag{3.3}$$

where H_d represents the headlines on day d, and M_t the concatenated DataFrame for month t. The merged monthly file is saved as $YYYY - MM - DD.csv$. These monthly raw-headline files serve as the inputs for the subsequent preprocessing pipeline (Section 3.3.3).

3.3.3 Data Preprocessing

This section processes the raw data collected in Section 3.3.2 into the clean, merged panel used for feature engineering. This section just includes cleaning, formatting, and alignment, with no modeling involved.

3.3.3.1 Tesla Closing-Price Series Preparation

The series of closing prices C_t corresponding to Tesla, Inc. (TSLA) for every day is read directly from the CSV saved in Section 3.3.2.1. These raw closing prices are used as the main forecasting target, following a standard approach where price levels are directly modeled for uncovering actionable investment signals, given empirical evidence as well (Sezer & Özbayoglu, 2018; Tsay, 2005).

There is no imputation carried out for non-trading days (weekends, public holidays); such days are simply not present in the index, preserving the real market calendar (Hull, 2018). For later technical-indicator generation (Section 3.4), the log-return feature

$$r_t = \ln(C_t) - \ln(C_{t-1}) \tag{3.4}$$

is calculated and saved with C_t. The log-return transformation is a standard way to yield a stationary surrogate of percentage price changes, a convention extensively tested in financial time-series literature (Brock et al., 1992).

3.3.3.2 Macroeconomic Series Cleaning

The 13 monthly macroeconomic series in Section 3.3.2.2 were treated through an ordinary cleaning process such as often done in time-series econometrics. First of all, occasional missing values, which typically result from lags in coverage, were interpolated with linear interpolation in line with standard procedures in the study of

economic data (Enders, 2010). Unit root tests based on the Augmented Dickey–Fuller (ADF) (Dickey & Fuller, 1979) test were then used to impose stationarity at the 5 % significance level.

For each variable $X_{k,t}$, the level series was tested; the rejection of the unit-root null ($p < 0.05$) indicated that $X_{k,t}$ should be included without any further transformation. When the level was argued to be non-stationary ($p \geq 0.05$), a first difference

$$\Delta X_{k,t} = X_{k,t} - X_{k,t-1} \tag{3.5}$$

was computed and retested. If $\Delta X_{k,t}$ also fail to become stationary, a second difference

$$\Delta^2 X_{k,t} = \Delta X_{k,t} - \Delta X_{k,t-1} \tag{3.6}$$

was applied and tested again. The stationarity requirement of ARIMAX estimation was satisfied by each series through the use of this incremental approach, which was preferred for its simplicity and avoidance of over-differencing (Box, Jenkins, & Reinsel, 2015). Only the minimally differenced from for which an ADF test rejected the unit-root null ($p<0.05$) was exported into the "economic_indicators_cleaned. csv" as exogenous regressor and the input to the LSTM. The results of the ADF test on macroeconomic indicators will be discuss at Chapter 4.

3.3.3.3 News-Headline Text Cleaning

Each raw headline collection M_t (Section 3.3.2.3) per month is preprocessed through a multi-stage text-cleaning pipeline so as to normalize and suppress noise before sentiment scoring.

First, the headlines will be translated. This is because the GDELT headlines are distributed across various languages, automate translation was apply to standardize them into English. Titles in the monthly CSV are translated directly by the *deep_translator. GoogleTranslator API*:

translator = GoogleTranslator(source='auto', target='en')

The thread-pooled executor (max_workers=10) use to ensure efficient throughput. The translated titles are saved in a new column "TranslatedTitle" and saved to the "_translated. csv" file.

This automatic translation step has been found to help normalize multilingual input and thus preserve semantic authenticity in downstream textual analyses (Johnson & Zhang, 2019).

Normalization processing is carried out after translation using typical NLP methods. The titles are lowercased, and non-space and non-alphanumeric characters are eliminated. The English stopwords provided by NLTK (Bird, Klein, & Loper, 2009) are used to filter out all forms of stopwords. The spaCy *en_core_web_sm* model is used to lemmatize each token in order to normalize inflectional forms (Honnibal & Montani, 2017). These approaches are complementary, reducing word scarceness and improving sentiment assign accuracy.

To avoid the inflated sentiment signals of repeated headlines, fuzzy matching algorithm over the translations of the titles for each month is applied. The similarity between two headlines, A and B, is calculated with RapidFuzz's token set ratio, a measure based on the token set similarity, which resembles the Jaccard index between token multisets:

$$sim(A,B) = \frac{|Tokens(A) \cap Tokens(B)|}{|Tokens(A) \cup Tokens(B)|} \times 100\%. \quad (3.7)$$

According to Christen (2012), a cut-off of roughly 85% enhances F1 performance while maintaining a balance of accuracy and recall. To assure unbiasedness, this research used four threshold values, T (75%, 80%, 85%, and 90%). At each T level, any such pair is considered a near-duplicate; only the first head is kept. To eliminate the $O(n^2)$ cost of exhaustive comparisons, this process will be parallelized across ten worker threads with Python's *ThreadPoolExecutor*.

For each threshold T, the clean dataset is saved with the suffix *_cleaned_threshold_{T}.csv*, and it logs how many titles have been removed or kept in a log. The per-threshold summaries are then merged into *Fuzzy_Matching_Full_Summary_All_Thresholds.csv*, which will easily measure the impact of aggressiveness in the deduplication on our monthly sentiment series S_t. This technique, based on well-studied literature (Monge & Elkan, 1997; Christen, 2012), assures that the sentiment inputs to our ARIMAX and to our LSTM models represent true diversity in news content and not editorial or typographic replication.

In the last cleaning step, any records with no "TranslatedTitle" are removed as there is nothing to score with. The exact duplicates are then removed, and include only the first instance of each unique "TranslatedTitle". More precisely, for each month's clean headline set M_t:

$$M_t^* = M_t \setminus \{h: TranslatedTitle(h) = \emptyset\}, M_t^\dagger = unique(M_t^*), \quad (3.8)$$

where $unique(\cdot)$ keeps the first element of each title in chronological order and drops the repeated ones. Implementation is done by using pandas:

df.dropna(subset=['TranslatedTitle'], inplace=True)

df.drop_duplicates(subset=['TranslatedTitle'], keep='first', inplace=True)

The resulting set $M_t^{\dagger}$ of headlines produced by this pipeline are free of language noise and redundancy, is ready for sentiment scoring in Section 3.4.

3.4 Sentiment Analysis Methods

Section 3.4 proposes three very different strategies to derive sentiment signals from cleaned Tesla headlines, however they all rely on the same six-step feature-construction pipeline (Section 3.4.1) in order to allow for direct comparisons (Tetlock, 2007 ; Engelberg & Parsons, 2011).

In 3.4.2 above, lexicon-based scoring is applied using the Loughran–McDonald financial dictionary (Loughran & McDonald, 2011) and the VADER algorithm (Hutto & Gilbert, 2014) to assign polarity scores at the sentence level. Section 3.4.3 presents a supervised classification ensemble (Logistic Regression, Linear SVM, Random Forest, Multinomial Naive Bayes and XGBoost) over the Financial Phrasebank (Malo et al., 2014) corpus, which has been extensive vectorized and hyperparameter tuned. Last, Section 3.4.4 provides an account of the use of pretrained transformer models (FinBERT-tone, DistilBERT, Twitter-RoBERTa) loaded through Hugging Face's AutoTokenizer and AutoModelForSequenceClassification for deep-learning-based sentiment embeddings (Devlin et al., 2019; Vaswani et al., 2017). Through using the same filtering, batching, aggregation, and alignment methods for each headline, all three sentiment streams are capable of generating feature sets that can fit seamlessly into both of the ARIMAX and LSTM predictive frameworks.

3.4.1 Unified Sentiment-Feature Construction Pipeline

Cleaned "Tesla" headlines are then processed into quantitative input for forecasting with a series of three complimentary sentiment-analysis models: (i) lexicon-based scoring, (ii) machine-learning classification, and (iii) deep-learning classification. Despite the differences in their algorithms, all methods follow a common pipeline with six steps, leading to comparability among the approaches (Tetlock, 2007; Engelberg & Parsons, 2011).

First, each cleaned headline $H_{d,i}$ according to selected method, a sentiment value $s_{d,i}$ is assigned.

Second, the scores are then averaged over N_d headlines on each calendar day d level by

$$\bar{s}_d = \frac{1}{N_d} \sum_{i=1}^{N_d} s_{d,i} \,, \tag{3.9}$$

This intra-day average eliminates idiosyncratic noise and provides single daily sentiment measure (Tetlock, 2007).

Third, for the "feature-enhanced" versions, the temporal transformations such as

$$\bar{s}_d = \frac{1}{N_d} \sum_{i=1}^{N_d} s_{d,i} \,, \tag{3.10}$$

lagged values $\bar{s}_{d-l}$, rolling means $MA_k(\bar{s})_d = \frac{1}{k} \sum_{j=0}^{k-1} \bar{s}_{d-j}$, rolling volatilities $Vol_k(\bar{s})_d = \sqrt{\frac{1}{k-1} \sum_{j=0}^{k-1} \left(\bar{s}_{d-j} - MA_k(\bar{s})_d\right)^2}$, and first differences $\Delta \bar{s}_d = \bar{s}_d - \bar{s}_{d-1}$ are added to capture momentum, volatility and abrupt changes in media tone (Engelberg & Parsons, 2011).

Fourth, because of sentiment is observed on the full calendar C (including weekends and holidays), forward fill alignment is applied on the values of non-trading dates t to the next trading session. Formally, let

$$\{\bar{s}_d : d \in C\} \tag{3.8}$$

denote the daily sentiment series on full calendar C (including weekends), and let $\tau \subset C$ denote the set of trading dates. Let $\{\tilde{S}_t : t \in \tau\}$ be the trading-day aligned series which is defined as follows:

$$\tilde{s}_t = \bar{s}_{d_t} \text{ where } \mathrm{d}_t = \max\{d \in C : d \leq t\} \tag{3.11}$$

This approach guaranteeing that every market-day return has an associated sentiment input (Hull, 2018).

Fifth, in order to synchronize with monthly macro indicators, these trading-day series $\{\tilde{s}_t\}$ is then averaged across all trading days T_m sessions in month m:

$$S_m = \frac{1}{T_m} \sum_{t \in \tau_m} \tilde{s}_t \,, \tag{3.12}$$

where $\tau_m \subset C$ is the set of trading dates in month m. Such forward-filling process is commonly applied during merging of irregularly exogenous indicators with regular market data, which keeps the trading-day series complete without any information loss in using the most recent sentiment signals derived (Campbell, Lo, & MacKinlay, 1997; Hull, 2018).

Finally, the monthly sentiment series $\{S_m\}$are combined with the full stationary macroeconomic indicators (Section 3.3.3.2) and the technical indicators that computed from raw closing prices (Section 3.5). Then, this combined panel with only stationary inputs, submitted to variance-inflation-factor (VIF) filtering in the ARIMAX framework. More precisely, for each potential VIF threshold (3, 5, 10, 15, 30), features exhibiting multicollinearity above the cutoff are iteratively removed until all remaining indicators fall below the specified level (Box, Jenkins, & Reinsel, 2015). The filtered features at each threshold form the ultimate ARIMAX estimation exogenous regressor sets. In contrast, the LSTM models bypass stationarity enforcement and VIF filtering, instead ingesting the raw (non-differenced) sentiment, macro, and technical features to exploit their capacity for learning complex temporal dependencies from non-stationary data (Sezer & Özbayoglu, 2018).

3.4.2 Lexicon-Based Approaches

In parallel to the machine-learning and deep-learning sentiment pipelines, lexicon-based methods provide a lightweight yet interpretable alternative by mapping words directly to sentiment categories. Two complementary lexicons were employed on the same corpus of cleaned Tesla headlines $H_{d,i,}$: the finance-specific Loughran–McDonald dictionary (LM) (Loughran & McDonald, 2011), which classifies tokens into seven tone categories (positive, negative, uncertainty, litigious, strong modal, weak modal, constraining), and the general-purpose VADER model (Hutto & Gilbert, 2014), which produces a compound polarity score by applying rule-based valence heuristics to social-media style text. Each lexicon yields a continuous per-headline score $s_{d,i}^{LM}$ and $s_{d,i}^{VADER}$, that is subsequently aggregated into daily series and enriched with lagged, rolling, and dispersion features for use in both ARIMAX and LSTM forecasting pipelines.

3.4.2.1 Ensemble Scoring of Lexicon on Tesla Headlines

Every headline $H_{d,i}$ published on calendar day d receives two parallel sentiment scores. From the LM dictionary, the net count of positive versus negative terms is computed as

$$C_{d,i}^{(k)} = \sum_{w \in H_{d,i}} 1\{w \in D_k\}. \tag{3.13}$$

$$s_{d,i}^{LM} = c_{d,i}^{(pos)} - c_{d,i}^{(neg)} \tag{3.14}$$

(Loughran & McDonald, 2011), where D_{pos} and D_{neg} denote the sets of positive and negative vocabulary entries, respectively. While, the VADER model assigns each headline a compound score

$$s_{d,i}^{VADER} = compound(H_{d,i}) \in [-1,1], \tag{3.15}$$

which merges lexicon lookup with syntactic and punctuation-based valence modifications (Hutto & Gilbert, 2014). By treating $\{s_{d,i}^{LM}, s_{d,i}^{VADER}\}$ as an ensemble of complimentary signals, the ensuing daily aggregation process captures both domain-specific financial tone and wider emotional subtlety. Empirical investigations have demonstrated that merging different lexicons may increase resilience against idiosyncratic vocabulary coverage gaps (Taboada et al., 2011), prompting this dual-scoring strategy.

3.4.2.2 Loughran Mcdonald Sentiment Descriptors

Once each headline $H_{d,i}$ receives a net Loughran–McDonald score

$$s_{d,i}^{LM} = \sum_{w \in H_{d,i}} 1\{w \in D_{pos}\} - \sum_{w \in H_{d,i}} 1\{w \in D_{neg}\} \tag{3.16}$$

these values are collapsed into two complementary sets of daily features. The non-feature descriptors (Table 3.2) capture the core signal and its short-term persistence, while the feature-enhanced descriptors (Table 3.3) further quantify category-specific momentum and uncertainty dispersion.

Table 3-2 Non-feature Loughran-McDonald Descriptors (Section 3.4.2.2)

Statistic	Equation	Description	Reference
Net sentiment	$s_{d,\text{net}} = \overline{p}_d - \overline{n}_d$	Daily overall bullish–bearish bias	Loughran & McDonald (2011)
Lag–1 mean	$s_{d-1,\text{net}}$	One-day lag to capture autocorrelation	Chatfield (2003)
3-day rolling average	$\frac{1}{3}\sum_{k=1}^{3} s_{d-k,\text{net}}$	Short-term smoothing of high-frequency noise	Chatfield (2003)

Table 3-3 Feature-Enhanced Loughran-McDonald Descriptors (Section 3.4.2.2)

Statistic	Equation	Description	Reference
Lag–1 negative mean	$\overline{n}_{d-1}$	One-day lag of negative tone	Chatfield (2003)
Lag–1 uncertainty mean	$\overline{u}_{d-1}$	One-day lag of uncertainty tone	Chatfield (2003)
7-day rolling std. of uncertainty	$\sqrt{\frac{1}{7}\sum_{k=1}^{7}\left(\overline{u}_{d-k}-\mu_{\overline{u}}\right)^2}$	Weekly dispersion in uncertainty intensity	Chatfield (2003)
Lag–1 constraining mean	$\overline{c}_{d-1}$	One-day lag of constraining tone	Chatfield (2003)

These descriptors transform raw LM counts into a concise set of interpretable time-series features that capture both the prevailing sentiment bias and its temporal dynamics, ready for inclusion in ARIMAX and LSTM forecasting models.

3.4.2.3 VADER Sentiment Descriptors

Each headline $H_{d,i}$ is scored by the VADER algorithm to produce a composite polarity

$$s_{d,i}^{VADER} = compound(H_{d,i}) \in [-1,1], \tag{3.17}$$

which reflects an intensity-weighted blend of positive, negative and neutral valence (Hutto & Gilbert, 2014). These per-headline scores are then aggregated into daily time-series features in two parallel forms: a non-feature set (Table 3.4) capturing the raw sentiment trend, and a feature-enhanced set (Table 3.5) augmenting that trend with supplementary directional and volatility measures.

Table 3-4 Non-feature VADER Descriptors (Section 3.4.2.3)

Statistic	Equation	Description	Reference

Compound mean	$\bar{c}_d = \frac{1}{N_d}\sum_{i=1}^{N_d} s_{d,i}^{\text{VADER}}$	Daily average compound polarity	Hutto & Gilbert (2014)
Lag–1 compound	$\bar{c}_{d-1}$	One-day lag of the compound score	Chatfield (2003)
3-day rolling average	$\frac{1}{3}\sum_{k=1}^{3}\bar{c}_{d-k}$	Short-term smoothing of compound fluctuations	Chatfield (2003)

Table 3-5 Feature-Enhanced VADER Descriptors (Section 3.4.2.3)

Statistic	Equation	Description	Reference
Lag–1 negative mean	$\overline{neg}_{d-1}$	One-day lag of negative polarity	Hutto & Gilbert (2014)
Lag–1 positive mean	$\overline{pos}_{d-1}$	One-day lag of positive polarity	Hutto & Gilbert (2014)
Lag–1 neutral mean	$\overline{neu}_{d-1}$	One-day lag of neutral polarity	Hutto & Gilbert (2014)
7-day rolling std. of compound	$\sqrt{\frac{1}{7}\sum_{k=1}^{7}\left(\bar{c}_{d-k}-\mu_{\bar{c}}\right)^2}$	Weekly volatility of the compound score	Chatfield (2003)

These concise descriptor sets ensure that both the baseline VADER signal and its enriched temporal dynamics are captured systematically for downstream ARIMAX and LSTM forecasting models.

3.4.3 Machine-Learning-Based Classification

In this study, five supervised classifiers are trained on the Financial Phrasebank corpus (Malo et al., 2014) to capture on naunced sentiment signals that lexical approaches could miss. Then, they are all used at once on Tesla headlines. This ensemble method uses different degrees of agreement, vectorization strategies, and model designs, while a unified scoring pipeline (Section 3.4.1) makes sure the comparability of the resulting sentiment series.

First, datasets at the agreement level are produced to show how different annotators agree (detail will be discuss in Section 3.4.3.1). Second, text is vectorized using the Bag-of-Words, TF-IDF, and Word2Vec methods, each with its own set of parameters (Section 3.4.3.2). Third, classifiers (Logistic Regression, Linear SVM, Random Forest, Naive Bayes, XGBoost) undergo extensive hyperparameter optimization by grid and random search (Section 3.4.3.3), assessed by macro-F1 to evenly weight all sentiment classes (Sokolova & Lapalme, 2009) (Section 3.4.3.4). Fourth, all of the trained models are put together into a score library. This way, every Tesla headline $H_{d,i}$ gets a forecast from every model (Section 3.4.3.5). Fifth, per-model predictions were mapped to numeric scores $s_{d,i}^{(m)} \in \{-1,0,1\}$ are collected daily (mean, median, variance, class counts) before entering the unified alignment and aggregation process (Section 3.4.1).

3.4.3.1 Agreement-Level Datasets Preparation

The Financial Phrasebank (Malo et al., 2014) is a gold-standard corpus of 4 841 short financial sentences that drawn from Reuters newswires and analyst reports. Each of the financial sentence manually annotated for sentiment (negative, neutral, positive). To control for annotation noise, four subsets are created by varying the minimum annotator consensus:

Table 3-6 Agree level of Financial Phrasebank

No.	Agree Level	Description	Sample size
1	100%	100% of annotators concur	$n \approx 2300$
2	75%	≥ 75% agreement	$n \approx 3600$
3	66%	≥ 66% agreement	$n \approx 4000$
4	50%	≥ 50% agreement	$n = 4841$ (full corpus)

Formally, let H be the full set of sentences and $A(h) \subseteq \{annotators\}$ the set who agreed on headline h. Then the subset at level α is

$$\mathcal{H}_{\alpha} = \left\{ h \in H : \frac{|\mathcal{A}(h)|}{|annotators|} \geq \alpha \right\} \tag{3.18}$$

Each sentence h is preprocessed by lowercased, punctuation-removed, tokenized, stop-words filtered, and lemmatized, then paired with its majority label $\ell_h \in \{neg, neu, pos\}$. These labels are mapped to numeric targets via:

$$y_h = \begin{cases} -1, & \ell_h = \text{neg}, \\ 0, & \ell_h = \text{neu}, \\ +1, & \ell_h = \text{pos}. \end{cases} \tag{3.19}$$

To assure robust performance estimation under class imbalance, each $\mathcal{H}_{\alpha}$ is split into 80% training and 20% test subsets using stratified sampling, preserving the relative frequencies of negative, neutral, and positive classes in both partitions (He & Garcia, 2009). This preparation produces four independent, noise-controlled datasets, each of which serves as the basis for further vectorization and classifier training.

3.4.3.2 Text Preprocessing & Vectorization

Prior to model fitting, each financial phrase is standardized cleaned: it is lowercased, punctuation-free, tokenized on whitespace and financial punctuation, NLTK stop-word-free, and lemmatized to dictionary form (Bird, Klein, & Loper, 2009).

Denoting the resulting token sequence by

$$T(h) = \left[t_1, t_2, \ldots, t_{L_h}\right], \tag{3.20}$$

this procedure reduces noise and normalizes morphological shifts, establishing a homogeneous foundation for all downstream vectorizers.

First, a Bag-of-Words (BoW) representation is constructed by selecting the $V = 2000$ most frequent unigrams and bigrams across the training corpus (Pang & Lee, 2008). The document–term matrix $\mathbf{X} \in \mathbb{R}^{N \times V}$ is defined entrywise as

$$X_{i,j} = \sum_{t \in T(h_i)} 1\{t = w_j\}, \tag{3.21}$$

where $\{w_1, \ldots, w_V\}$ is the chosen vocabulary. Three "tiers" of hyperparameters are examined: (1) Ori—unigrams only, no stop-word removal; (2) Intermediate—unigrams + bigrams with stop-word filtering; and (3) Fine-tune, a grid search over n-gram ranges $\{(1,1), (1,2)\}$,max_features $\{2000,3000\}$,and document-frequency thresholds $df \in \{1,3\}$, max $df \in \{0.8,0.95\}$. (Details refer to Table3.7)

Second, Term-Frequency–Inverse-Document-Frequency (TF-IDF) weighting refines these counts by minimizing frequently occurring phrases. For sentence h_i and vocabulary token w_j, the weight is

$$tfidf_{i,j} = X_{i,j} \times \ln\left(\frac{N}{df_j}\right) \tag{3.22}$$

with N the total number of training sentences and df_j the number of sentences containing w_j (Salton & Buckley, 1988). The same three hyperparameter tiers (Ori, Intermediate, Fine-tune) are applied to the TF-IDF vectorizer.

Table 3-7 Parameter list use in TF-IDF and BoW vectorizer

Version	n-gram range	Max features	Min df	Max df
Ori	(1,1)	3000	-	-
Intermediate	(1,2)	3000	-	-
Fine tune*	$\{(1,1),(1,2)\}$	$\{2000, 3000\}$	{1,3}	{0.8,0.95}

* For the Fine-tune runs by min_df=1 (or 3) and max_df=0.8 (or 0.95) and performed a full grid search over ngram_range, max_features, min_df and max_df as shown.

Finally, to capture semantic relationships beyond surface tokens, Word2Vec embeddings are trained separately on each agreement-level corpus $\mathcal{H}_\alpha$ under three settings:

Table 3-8 Word2Vec Hyperparameter Settings

Version	SG (0 = CBOW, 1 = Skip-gram)	Vector Size	Window	Min Count	Epochs
Ori	0	100	5	5	10
Intermediate	1	200	3	2	10
Fine tune*	{0,1}	$\{100,200,300\}$	{3,5,7}	2	10

* For the Fine-tune runs by min_count = 2 and epochs = 10, and performed a full grid search over sg, vector_size and window as shown.

Each sentence h is then embedded by averaging its token vectors,

$$v(h) = \frac{1}{|T(h)|} \sum_{t \in T(h)} w_t \,, \tag{3.23}$$

Where $w_t$is the Word2Vec embedding for token t. This dense representation captures latent semantic patterns and has been shown to improve financial-text classification (Mikolov et al., 2013).

3.4.3.3 Classifier Training and Hyperparameter Optimization

Vectorized datasets are used to train five classification algorithms: Logistic Regression, Linear Support Vector Machines (SVM), Random Forest, Multinomial Naive Bayes, and two XGBoost versions (XGBClassifier and XGBRFClassifier) for each agreement level and vectorizer combination. To guarantee fair comparison and robust performance, each model undergoes systematic hyperparameter adjustment and assessment under stratified five-fold cross-validation (He & Garcia, 2009).

First, Logistic Regression and Linear SVM adopt balanced class weighting to reduce label imbalance. The inverse-regularization parameter C is adjusted across $\{0.01,0.1,1,10,100\}$, , whereas the SVM utilizes a hinge loss with L_2 penalty. Grid search across this one-dimensional space discovers the best trade-off between margin width and misclassification (Pedregosa et al., 2011).

Second, Random Forests explore a six-dimensional grid:

Table 3-9 Random Forest Hyperparameter Grid

Parameter	Values
n_estimators	$\{100,200,300,500\}$
max_depth	$\{None, 10,20,30,50\}$
min_samples_split	$\{2,5,10\}$
min_samples_leaf	$\{1,2,4\}$
max_features	$\{\sqrt{V}, \log_2 V\}$

bootstrap	$\{True, False\}$

where V is the vector dimension. A manual grid search sampling up to 100 000 parameter combinations to identifies models that balance depth and variance, following Breiman's guidelines (Breiman, 2001).

Third, Multinomial Naive Bayes sets its smoothing parameter $\alpha \in \{0.01, 0.1, 0.5, 1.0, 2.0\}$and prior use fit_prior $\in \{True, False\}$ by exhaustive grid search. Although simpler, Naive Bayes may excel in high-dimensional sparse spaces when adequately regularized (Rish, 2001).

Table 3-10 Multinomial Naive Bayes Hyperparameter Grid

Parameter	Values
α (smoothing)	$\{0.01, 0.1, 0.5, 1.0, 2.0\}$
bootstrap	$\{True, False\}$

Fourth, XGBoost classifiers leverage randomized sampling across a broad hyperparameter space to regulate complexity and learning dynamics (Bergstra & Bengio, 2012). Both the tree-booster (XGBClassifier) and random-forest style (XGBRFClassifier) variations explore:

Table 3-11 XGBoost Classifier Hyperparameter Grid

Parameter	Values
learning_rate	$\{0.05, 0.1\}$
n_estimators	$\{200, 300\}$

max_depth	{3, 5, 7}
subsample	{0.8, 1.0}
γ (gamma)	{0, 0.1}
reg_alpha	{0, 0.1}
reg_lambda	{1,2}
colsample_bytree	{0.8, 1.0}
colsample_bynode	{0.8, 1.0}

Note: For each of the two XGBoost variations (XGBClassifier and XGBRFClassifier), a random subset of 200 hyper-parameter combinations was sampled for grid search to balance coverage with GPU-accelerated fit time.

3.4.3.4 Model Evaluation & Selection

Upon completion of hyperparameter tuning, each model's All searches optimize out-of-fold performance is assessed using the macro-averaged F1-score,

$$F_1^{\text{macro}} = \frac{1}{3} \sum_{c \in \{\text{neg,neu,pos}\}} \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{3.24}$$

which equally weights performance on negative, neutral, and positive classes and defends against trivial majority-class solutions (Sokolova & Lapalme, 2009). For each algorithm, optimum macro-F1 model configuration for each method, vectorizer, and agreement level is recorded. However, all trained models are preserved in the scoring library regardless of rank to preserve ensemble diversity and minimize overbias on any single "best" predictor (Müller et al., 2020). These models then go to the scoring step (Section 3.4.3.5), where each assigns a sentiment to every Tesla headline.

3.4.3.5 Ensemble Scoring of Tesla Headlines

Once the full library of machine-learning classifiers is trained and validated (Section 3.4.3.4), each model is applied to the cleaned, deduplicated Tesla headline corpus $\mathscr{H}_{\alpha}^{\dagger}$. For every headline $h_{d,i}$ occurring on calendar day d, each classifier m produces a predicted label

$$\ell_{d,i}^{(m)} \in \{\text{neg,neu,pos}\}, \tag{3.25}$$

which is then mapped to a numeric score

$$s_{d,i}^{(m)} = \begin{cases} -1, & \ell_{d,i}^{(m)} = \text{neg}, \\ -0, & \ell_{d,i}^{(m)} = \text{neu}, \\ +1, & \ell_{d,i}^{(m)} = \text{pos}. \end{cases} \tag{3.26}$$

This three-way coding follows the standard in financial sentiment studies (He & Garcia, 2009) and ensures neutrality is distinguished from directional tone.

To retain complete provenance, each scored record is annotated with 5 pieces of metadata:

(Model_Type, Vectorizer_Type, Agree_Level, Model_Version, Fuzzy_Level)

The resulting per-headline CSV outputs are used as raw input to the downstream aggregation stage (Sect. 3.4.3.6) and that each model's viewpoint is propagated in both ensemble analyses (Sect. 3.6) and make compare with lexicon-based and deep-learning pipelines.

3.4.3.6 Non-Feature Dataset Construction

Extending from the Section 3.4.3.5 per-headline labels, each Tesla headline $H_{d,i}$ bears a discrete score outranging from the following set

$$s_{d,i}^{(m)} \in \{+1, 0, -1\} \tag{3.27}$$

that use for our machine-learning classifiers in previous study (Sokolova & Lapalme, 2009). Every timestamp was normalized to its calendar date d, then aggregate to one row per day by computing the following eight non-feature descriptors (Table 3.12):

Table 3-12 Non-feature daily sentiment descriptors (Section 3.4.3.6)

Statistic	Equation	Description	Reference
Mean sentiment	$\bar{s}_d^{(m)} = \frac{1}{N_d}\sum_{i=1}^{N_d} s_{d,i}^{(m)}$	Average daily sentiment, capturing overall bullish vs. bearish bias	Chatfield (2003)
Median sentiment	$\tilde{s}_d^{(m)} = \text{median}\left\{s_{d,i}^{(m)}\right\}_{i=1}^{N_d}$	Robust center measure, reduces the influence of outlier headlines	Chatfield (2003)
Standard deviation	$\sigma_d^{(m)} = \sqrt{\frac{1}{N_d - 1}\sum_{i=1}^{N_d}\left(s_{d,i}^{(m)} - \bar{s}_d^{(m)}\right)^2}$	Dispersion of daily sentiment, indicates intra-day volatility in tone	Chatfield (2003)
Minimum	$s_d^{(m),\min} = \min_{1\le i\le N_d} s_{d,i}^{(m)}$	Most negative headline score, flags extreme pessimistic events	Tsay (2010)
Maximum	$s_d^{(m),\max} = \max_{1\le i\le N_d} s_{d,i}^{(m)}$	Most positive headline score, flags extreme optimistic events	Tsay (2010)
Sum	$S_d^{(m)} = \sum_{i=1}^{N_d} s_{d,i}^{(m)}$	Cumulative sentiment, amplifies the effect of multiple	Sokolova & Lapalme (2009)

		concordant headlines	
Count	$N_d = \sum_{i=1}^{N_d} 1$	Total number of headlines, proxy for news volume	Tsay (2010)
Positive-headline count	$C_d^{+,(m)} = \sum_{i=1}^{N_d} \mathbf{1}\{s_{d,i}^{(m)} = +1\}$	Number of bullish headlines, measures positive coverage intensity	Kumar et al. (2016)
Neutral-headline count	$C_d^{0,(m)} = \sum_{i=1}^{N_d} \mathbf{1}\{s_{d,i}^{(m)} = 0\}$	Number of neutral headlines, indicates coverage without clear directional bias	Kumar et al. (2016)
Negative-headline count	$C_d^{-,(m)} = \sum_{i=1}^{N_d} \mathbf{1}\{s_{d,i}^{(m)} = -1\}$	Number of bearish headlines, measures negative coverage intensity	Kumar et al. (2016)

Each of these 8 statistics is a column in *aggregated_daily_sentiment. csv* with 5 metadata fields

(Model_Type, Vectorizer_Type, Agree_Level, Model_Version, Fuzzy_Level)

These daily summaries capture each model's raw judgments without any temporal transforms. They form the base "non-feature" dataset to be forwarded next into calendar alignment (Step 4 of Section 3.4.1).

3.4.3.7 Feature-Enhanced Dataset Construction

Using the non-feature daily series from Section 3.4.3.6, the expanded set of engineered time-series features are generated. View detailed in Table 3.13:

Table 3-13 Feature-enhanced time-series features (Section 3.4.3.7)

Statistic	Equation	Description	Reference
Lag-ℓ mean	$\overline{s}_{d-\ell}^{(m)}, \quad \ell \in \{1,3,5\}$	Capture short-term autocorrelation in average sentiment	Box & Jenkins (1976)
Lag-ℓ sum	$S_{d-\ell}^{(m)} = \sum_{i=1}^{N_{d-\ell}} s_{d-\ell,i}^{(m)}, \quad \ell \in \{1,3,5\}$	Reflects cumulative tone ℓ days ago, smoothing high-frequency noise	Box & Jenkins (1976)
Rolling-window mean	$\overline{\overline{s}}_{d,w}^{(m)} = \frac{1}{w}\sum_{k=0}^{w-1} \overline{s}_{d-k}^{(m)}, \quad w \in \{3,7,14\}$	Seven-/three-/fourteen-day moving average of sentiment, reveals medium-term trends	Chatfield (2003)
Rolling-window standard deviation	$\overline{\sigma}_{d,w}^{(m)} = \sqrt{\frac{1}{w-1}\sum_{k=0}^{w-1}\left(\overline{s}_{d-k}^{(m)} - \overline{\overline{s}}_{d,w}^{(m)}\right)^2}$, $w \in \{3,7,14\}$	Rolling volatility, measures local fluctuations in sentiment level	Chatfield (2003)
First difference of mean	$\Delta\overline{s}_{d}^{(m)} = \overline{s}_{d}^{(m)} - \overline{s}_{d-1}^{(m)}$	Detects abrupt shifts or momentum in	Zivot & Wang (2006)

		daily sentiment	
First difference of sum	$\Delta\overline{s}_d^{(m)} = \overline{s}_d^{(m)} - \overline{s}_{d-1}^{(m)}$	Measures day-over-day change in cumulative sentiment	Zivot & Wang (2006)
Positive/Negative ratio	$R_d^{(m)} = \frac{C_d^{+,(m)}}{C_d^{-,(m)} + \varepsilon}$	Quantifies directional imbalance between positive and negative headlines	Kumar et al. (2016)
Volatility–Level interaction	$I_d^{(m)} = \overline{s}_d^{(m)} \times \sigma_d^{(m)}$	Encodes joint movements in sentiment level and dispersion (leverage-like effects)	Cont (2001)

- Lagged terms capture short-term autocorrelation in both sentiment bias and cumulative tone (Box & Jenkins, 1976).
- Rolling statistics smooth high-frequency noise and expose medium-term trends and volatility patterns (Chatfield, 2003).
- First differences highlight sudden shifts or momentum in daily sentiment (Zivot & Wang, 2006).
- Pos/Neg ratio emphasizes directional imbalance in news flow (Kumar et al., 2016).
- Interaction term between level and dispersion encodes joint dynamics akin to leverage effects in financial returns (Cont, 2001).

These transformations capture momentum, medium-term trends, abrupt shifts, directional imbalance and joint dynamics. Once they are computed, the feature-enhanced series follows exactly the same calendar alignment, monthly aggregation and merge procedures of the unified pipeline in Section 3.4.1, so that full procedural consistency with the other sentiment streams is preserved.

3.4.4 Deep-Learning-Based Embedding

To make sure the comparability against lexicon- and machine-learning-based sentiment pipelines, three transformer classifiers are similarly used without any finetuning on the cleaned, deduplicated Tesla headlines. Each model is loaded from HuggingFace *(AutoTokenizer + AutoModelForSequenceClassification)* and executed in inference mode to yield the discrete label and the confidences. The raw outputs are passed through two parallel feature streams (non-feature and feature-enhanced), which are incorporated into the overall six-step pipeline in Section 3.4.1.

3.4.4.1 Pretrained Transformer Models

All three classifiers share the same Transformer-Encoder backbone, whose multi-head self-attention layer computes

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V, \tag{3.28}$$

and whose [CLS]-token representation h_{CLS} is passed through a linear softmax head

$$\hat{y} = \text{softmax}\left(W h_{\text{CLS}} + b\right) \tag{3.29}$$

to yield class probabilities (Vaswani et al., 2017; Devlin et al., 2019).

FinBERT-tone (*yiyanghkust/finbert-tone*) is a BERT-Base (12 layers, 768-dim hidden, 12 heads) model further fine-tuned on a large financial-tone corpus—analyst reports, corporate filings—to specialize in financial language. By adapting its embeddings to

domain-specific jargon, FinBERT-tone achieves up to a 5 % F_1 improvement on finance-text sentiment benchmarks compared with generic BERT (Araci, 2019).

DistilBERT (SST-2) (*distilbert-base-uncased-finetuned-sst-2-english*) is a 6-layer student network distilled from BERT-Base via KL-divergence and hidden-state alignment losses (Sanh et al., 2019). Trained on the Stanford Sentiment Treebank for binary sentiment, it retains over 97 % of BERT's accuracy while halving inference time, making it an efficient choice for large-scale scoring.

Twitter-RoBERTa (*cardiffnlp/twitter-roberta-base-sentiment*) builds on RoBERTa-Base (12 layers) pretrained with dynamic masking on massive web corpora, then fine-tuned on a Twitter-sentiment dataset for three-class prediction (Liu et al., 2019). Its optimized pretraining and robust vocabulary handling yield 2–3 % gains over BERT-Base on short, informal text.

Each of these models is applied in batch across the four fuzzy-threshold folders, producing per-headline CSVs of raw class probabilities, discrete labels (neg/neu/pos), and associated confidence scores, which ready for the non-feature and feature-enhanced dataset constructions in Sections 3.4.4.3–3.4.4.4.

3.4.4.2 Inference Pipeline

Inference is done by loading each of these pretrained checkpoints using HuggingFace's *AutoTokenizer* and *AutoModelForSequenceClassification*, and then batching it over the clean and deduplicated Tesla-headline corpus. Specifically:

First, headlines $H_{d,i}$ are batched (with a maximum of 16) and tokenized based on each model's vocabulary (WordPiece for BERT/FinBERT, distilled WordPiece for DistilBERT, byte-pair for RoBERTa). Tokenization enforces

Code:

```
padding=True, truncation=True, max_length=512
```

such that each sequence is encoded as a fixed-length input matrix of token IDs and attention masks.

Second, each batch is moved to the available device (if GPU exists, compatible else CPU) and processed by the transformer which outputs for each headline a logit vector

$$\mathbf{z}_{d,i} = \left(z_{d,i,\text{neg}}, z_{d,i,\text{neu}}, z_{d,i,\text{pos}}\right) \tag{3.30}$$

These logits are further passed through the softmax function to obtain the class - probability estimates (Devlin et al., 2019; Vaswani et al., 2017):

$$\ell_{d,i} = \arg\max_{c\in\{\text{neg,neu,pos}\}} p_{d,i,c} \tag{3.31}$$

and the confidence of the model in this label is recorded as

$$\text{conf}_{d,i} = \max_{c\in\{\text{neg,neu,pos}\}} p_{d,i,c} \tag{3.32}$$

Note that for each headline the complete probability vector $\{p_{d,i,c}\}$, the discrete label $\ell_{d,i}$, and the confidence $conf_{d,i}$ are extended by the original metadata (Date, fuzzy threshold) and written to a per-model "Scored datasets" CSV. The above inference pipeline provides us with all the vocabulary requirements to build the non-feature and feature-enriched sentiment streams in Sections 3.4.4.3–3.4.4.4.

3.4.4.3 Non-Feature Dataset Construction

To make the comparison apple-to-apple with the machine-learning sentiment models (section 3.4.3), the class-probability outputs discard from each transformer and only keep the discrete label

$$s_{d,i}^{(m)} = \begin{cases} +1, & \ell_{d,i}^{(m)} = \text{pos}, \\ 0, & \ell_{d,i}^{(m)} = \text{neu}, \\ -1, & \ell_{d,i}^{(m)} = \text{neg}. \end{cases} \tag{3.33}$$

This corresponds to a numeric encoding for the ML classifiers (Sokolova & Lapalme, 2009). For every m and fuzzy-threshold folder, exported into a per-headline CSV with:

- Date d (normalized to calendar day)
- Discrete sentiment $s_{d,i}^{(m)}$

All header timestamps are normalized with respect to its calendar day d (step 2 in Section 3.4.1), and the above defined eight daily descriptors from Section 3.4.3.6 (Table 3.12) are calculated, generating one non-feature record per trading day (Chatfield, 2003; Tsay, 2010). This minimal representation maintains the unprocessed transformer decisions before any temporal transformation.

3.4.4.4 Feature-Enhanced Dataset Construction

Starting from the non-feature daily series, the same temporal transforms defined in Section 3.4.3.7 (Table 3.13) are applied:

By reusing identical formulas and window parameters, the feature-enhanced stream is fully aligned with the machine-learning pipeline before entering the unified aggregation and forecasting framework of Section 3.4.1.

3.4.4.5 Integration of Downstream

The non-feature and feature-enhanced daily series generated in Sections 3.4.4.3 and 3.4.4.4 are then passed directly into the Unified Sentiment-Feature Construction Pipeline (Section 3.4.1). There they undergo calendar alignment (forward-fill to trading days), monthly aggregation, and merge with technical and macro indicators exactly as described in Steps 4–6 of Section 3.4.1, before feeding into ARIMAX and LSTM forecasting.

3.5 Feature Engineering for Technical & Macroeconomic Inputs

Daily technical indicators are first calculated from raw close prices and log-returns and tested for stationarity using Augmented Dickey–Fuller (ADF) tests (Dickey & Fuller, 1979). Differenced series is differenced once for non-stationary in ARIMAX models, original daily level is used for input of LSTM model. Then for ARIMAX, each stationary daily series X_d is resampled to monthly frequency by

$$X_m = \frac{1}{|D_m|} \sum_{d \in D_m} X_d \tag{3.34}$$

Where D_m are the trading days of month m (Campbell, Lo, & MacKinlay, 1997). This generates exogenous predictors which correspond to the month macroeconomic releases.

3.5.1 Daily Technical Indicators and Monthly Resampling

The following seven indicators are computed on each trading day d before monthly aggregation:

Table 3-14 Technical indicators for time series feature (Section 3.5.1)

Indicator	Equation	Description	Reference
Logarithmic Return	$r_d = \ln\left(\frac{C_d}{C_{d-1}}\right)$	Stabilizes variance, consistent with continuous time finance theory	Campbell et al. (1997)
3-Month Rolling Volatility	$\mathrm{Vol}_d^{(3m)} = \sqrt{\frac{1}{62}\sum_{i=0}^{62}\left(r_{d-i} - \bar{r}_d^{(3m)}\right)^2}$, $\bar{r}_d^{(3m)} = \frac{1}{63}\sum_{i=0}^{62} r_{d-i}$	Measures risk through recent return dispersion	Tsay (2010)
3 Month Simple Moving	$\mathrm{SMA}_d^{(3m)} = \frac{1}{63}\sum_{i=0}^{62} C_{d-i}$	Unfolds medium-term price patterns	Brock et al. (1992)

Average (SMA)			
14-day Relative Strength Index (RSI)	$\mathrm{RSI}_d = 100 - \frac{100}{1+\mathrm{RS}_d}$, $\mathrm{RS}_d = \frac{\bar{U}_d^{(14)}}{\bar{D}_d^{(14)}}$ Where $(\bar{U}_d^{(14)})$ and $(\bar{D}_d^{(14)})$ are the 14-day average gains and losses.	Shows overbought/oversold conditions based on average gain/loss	Wilder (1978)
3-Month Momentum	$\mathrm{Mom}_d^{(3\mathrm{m})} = C_d - C_{d-63}$	Performance measures raw price change over a quarter, which reflects trend continuation	Jegadeesh & Titman (1993)
3-Month Bollinger Band Width	$\mathrm{Width}_d^{(3\mathrm{m})} = 4\mathrm{Vol}_d^{(3\mathrm{m})}$	Reflects volatility expansion/contraction around the MA	Bollinger (2002)
3-Month MACD Histogram	$\mathrm{MACD}_d = \mathrm{EMA}_{12}(C_d) - \mathrm{EMA}_{26}(C_d)$, $\mathrm{Hist}_d = \mathrm{MACD}_d - \mathrm{EMA}_9(\mathrm{MACD}_d)$ Where (EMA_k) is the (k)-day exponential moving average	Highlights new directions for momentum when the MACD line crosses its 9-day trigger line	Appel, (2005)

The resulting series of indicators$\{X_d\}$ are then checked for unit roots. Differencing is applied only to series deeming as nonstationary, while holding original daily levels for the LSTM model. Lastly, the resulting stationary daily series are aggregated to calculate the monthly exogenous regressors X_m for ARIMAX estimation.

3.5.2 Macroeconomic Feature Screening

The thirteen pre-cleaned macro series are first differenced (once or twice as necessary per Section 3.3.3.2) to achieve stationarity. For each stationary series x_m, the sample Pearson correlation r with the monthly Tesla price series y_m is computed as:

$$r = \frac{\sum_{m=1}^{M}(y_m - \bar{y})(x_m - \bar{x})}{\sqrt{\sum_{m=1}^{M}(y_m - \bar{y})^2}\sqrt{\sum_{m=1}^{M}(x_m - \bar{x})^2}} \tag{3.35}$$

Indicators satisfying $|r| \geq 0.30$ (a "medium" effect size threshold; Cohen, 1988) are retained as exogenous regressors in the ARIMAX models (Box & Jenkins, 1976).

The same retained indicators are then evaluated in their *raw* (level) form against the un-differenced monthly Tesla prices. Each raw series whose Pearson correlation again meets $|r| \geq 0.30$ is included in the LSTM feature panel (Hochreiter & Schmidhuber, 1997).

By applying a uniform correlation-threshold criterion across both pipelines, the ARIMAX and LSTM frameworks use an identical set of macroeconomic signals are stationary in the former and level-data in the latter. Each demonstrating a substantively meaningful linear relationship with Tesla price. The final list of selected macro indicators and their exact correlation coefficients appear in Chapter 4.

3.6 Modeling Framework

This section discusses the two forecasting models, ARIMAX and LSTM, that use the feature panels built in Sections 3.4 (sentiment) and 3.5 (technical + macro). Through the use of the same train/test breakdown and evaluation metrics and data preparation, the relative performance of each sentiment method (lexicon, ML, DL) can be directly compared for both linear (ARIMAX) and non-linear (LSTM) models.

3.6.1 ARIMAX Modeling

An ARIMAX (p, d, q) specification is used to relate Tesla's differenced closing price to its own autoregressive and moving-average terms plus exogenous regressors (sentiment, technical, and macro features). For each sentiment stream and VIF threshold, models are fit over a grid of orders $p, q \in \{0, \ldots, 4\}$, holding d fixed at 1. Model selection is driven by out-of-sample RMSE, with fallback to in-sample RMSE when necessary. The best configuration is then refit on the full training set and evaluated on both the training and hold-out samples.

3.6.1.1 Stationarity Assessment & Differencing

The ARIMA-type models are based on the concept that the target series is weakly stationary in that it has constant mean, variance and autocovariance structure over time (Box & Jenkins, 1976). To assess this, the following steps were undertaken on Tesla's daily closing price series Y_t:

1. Autocorrelation Analysis

   The sample autocorrelation function (ACF) of Y_t displayed a slowly decaying tail, whereas the partial autocorrelation function (PACF) had a significant spike only at the lag 1. This phenomenon is the typical characteristic of a unit root process and is not a stationary AR(p) model (Box & Jenkins, 1976).

2. Augmented Dickey–Fuller (ADF) Test

   The ADF test augments the simple Dickey–Fuller unit-root regression by including lagged differences to control for serial correlation:

$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \sum_{i=1}^{k} \delta_i \Delta Y_{t-i} + \varepsilon_t \tag{3.36}$$

The null hypothesis $H_0: \gamma = 0$ (unit root) was not rejected at the 5 % level for the level series, confirming non-stationarity (Dickey & Fuller, 1979).

3. First Differencing

   To induce stationarity, a first-difference transformation was applied:

$$\Delta Y_t = Y_t - Y_{t-1} \tag{3.37}$$

This differenced series passed both the ADF test ($p < 0.01$) and exhibited rapidly decaying ACF/PACF, indicative of being stationary (Breitung & Hamilton, 1995).

4. Rationale for $d = 1$

   Higher-order differencing may lead to overdifferencing that takes out real signal and introduces moving-average structure (Box & Jenkins, 1976). The empirical diagnostics (ACF/PACF and ADF) consistently followed $d = 1$, and so all ARIMAX models retain a single difference.

By fixing $d = 1$ across every ARIMAX$(p, 1, q)$ specification, comparability is preserved and each model satisfies the stationarity requirement without over-transforming the data.

3.6.1.2 Stationarity Assessment & Differencing

All feature panels—sentiment, technical and macro are unified to monthly frequency in Section 3.4.1 Step 5. The transformed stationary target series ΔY_m(monthly differenced close) and the VIF-filtered exogenous regressors X_m are merged on the common month-start date m. Any months with missing values in either series are removed to maintain a consistent time grid.

To mimic the real-time forecasting and to avoid look-ahead bias, the merged monthly data is later sorted by date and chronologically divided at the month-level into the first 80% of the months ($\lfloor 0.8\, M \rfloor$) and the remaining months. The first 80% are training, and the other 20 % are test. This division over time ensures causality and mimics operational forecasting (Bergmeir & Benítez, 2012; Hyndman & Athanasopoulos, 2018).

3.6.1.3 Order Selection via Grid Search

For each VIF-filtered feature set, a systematic grid search in the orders of the ARIMA $p, q \in \{0, \ldots, 4\}$ (with $d = 1$) is conducted. The SARIMAX log-likelihood is maximized for each candidate $(p, 1, q)$under the stationarity and invertibility constraints, and the following metrics are recorded:

- Akaike Information Criterion (AIC):

$$\mathrm{AIC} = -2\ln\left(\hat{L}\right) + 2k \tag{3.38}$$

  where $\hat{L}$ is the maximized likelihood and k the number of estimated parameters (Akaike, 1974).

- Bayesian Information Criterion (BIC):

$$\mathrm{BIC} = -2\ln\left(\hat{L}\right) + k\ln(n) \tag{3.39}$$

  penalizing model complexity more heavily ($n =$ number of observations) (Schwarz, 1978).

- Forecast accuracy (out-of-sample RMSE):

$$\mathrm{RMSE}_{\mathrm{out}} = \sqrt{\frac{1}{N_{\mathrm{test}}} \sum_{t \in \mathrm{test}} \left(y_t - \hat{y}_t\right)^2} \tag{3.40}$$

  where $\hat{y}_t$ denotes the one-step-ahead forecast (Hyndman & Athanasopoulos, 2018).

Out-of-sample RMSE is the primary criterion for selection: the (p, q) yielding the lowest $RMSE_{out}$is chosen for each VIF threshold. If no valid out-of-sample forecasts are available (e.g. all predictions NaN), the model with the lowest in-sample RMSE is selected instead. Simultaneously recording both information criteria and RMSE provides a clear compromise between goodness-of-fit and predictive performance.

3.6.1.4 Final Model Estimation & Diagnostic Checks

Once the optimal ARIMAX$(p, 1, q)$ configuration is selected (Section 3.6.1.3), it is re-estimated on the full training set to obtain final parameter estimates and to verify model assumptions. Diagnostic checks include:

1. Residual Autocorrelation (Ljung–Box Test)

   The residuals $\hat{\varepsilon}_t$ should resemble white noise. The Ljung–Box Q statistic

   $$Q(m) = n(n+2)\sum_{k=1}^{m}\frac{\hat{r}_k^2}{n-k} \tag{3.41}$$

   Test H_0: no autocorrelation up to lag m (Ljung & Box, 1978). Failure to reject at the 5 % level indicates adequacy of the AR and MA terms.

2. Residual Normality (Jarque–Bera Test)

   Normality of residuals is assessed via

   $$\text{JB} = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right) \tag{3.42}$$

   where S and K are the sample skewness and kurtosis of $\hat{\varepsilon}_t$ (Jarque & Bera, 1980). A non-significant JB statistic suggests that Gaussianity assumptions hold.

3. Heteroskedasticity (ARCH LM Test)

   To ensure constant variance, Engle's ARCH test regresses $\hat{\varepsilon}_t^2$ on its own lags and tests for remaining ARCH effects (Engle, 1982).

   $$\hat{\varepsilon}_t^2 = \alpha_0 + \sum_{i=1}^{p}\alpha_i\hat{\varepsilon}_{t-i}^2 + u_t \tag{3.43}$$

   Non-rejection of H_0:

$$H_0 : \alpha_1 = \alpha_2 = \ldots = \alpha_p = 0 \tag{3.44}$$

no ARCH upholds homoskedasticity.

4. Parameter Significance & Stability

   Parameter estimates $\hat{\phi}_i, \hat{\theta}_j, \hat{\beta}$ are examined via their t-statistics. Stability is further confirmed by verifying that all AR and MA roots lie outside the unit circle (Box & Jenkins, 1976).

   AR-root condition (stability):

$$\phi(z) = 1 - \sum_{i=1}^{p} \phi_i z^i = 0, \quad \text{all roots satisfy } |z| > 1 \tag{3.45}$$

Only when these diagnostics indicate no model misspecification does the ARIMAX model proceed to final forecasting (Section 3.6.1.5).

3.6.1.5 Forecast Generation & Performance Evaluation

Once the final ARIMAX$(p, 1, q)$ model is estimated on the full training set, both in-sample predictions and out-of-sample forecasts are produced:

1. In-Sample Prediction

   One-step-ahead fitted $\hat{y}_t$ for $t = 1, \ldots, T_{train}$ are obtained via the model's *get_prediction* method. These reflect how well the model captures the dynamics it was trained on.

2. Out-of-Sample Forecast

   The *get_forecast* method generates forecasts $\hat{y}_t$ for $t = T_{train} + 1, \ldots, T_{total}$, using only exogenous regressors in the test period. This assesses real-world predictive performance.

Accuracy is quantified by four standard metrics (Hyndman & Athanasopoulos, 2018; Armstrong, 2001):

- Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(y_t - \hat{y}_t\right)^2} \tag{3.46}$$

- Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t| \tag{3.47}$$

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{100}{N}\sum_{t=1}^{N}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{3.48}$$

- Coefficient of Determination (R^2)

$$R^2 = 1 - \frac{\sum_{t=1}^{N}\left(y_t - \hat{y}_t\right)^2}{\sum_{t=1}^{N}\left(y_t - \bar{y}\right)^2} \tag{3.49}$$

Here N is the number of points in either the training or testing set, y_t the actual values, $\hat{y}_t$ the predicted or forecasted values, and $\bar{y}$ the sample mean of the actuals. These metrics balance sensitivity to large errors (RMSE) with robustness to outliers (MAE), relative scale (MAPE), and explained variance (R^2).

Model prediction, actual series and the annotation of performance summaries have also been exported to CSV files and the associated plots, which enable clear

comparison of lexicon, machine-learning and deep-learning sentiment input under the same ARIMAX setups.

3.6.2 LSTM Modeling

As a complement to the ARIMAX linear methodology, a recursive network, a Long Short-Term Memory (LSTM) model, is used to capture non-linear and long-range temporal dependencies from the combined sentiment+technical (±macro) feature panels. By feeding sequences of monthly features into an LSTM based encoder and subsequently mapping out its final hidden state with a small multilayer head, the model can capture more complex relationships that can easily be overlooked by linear models (Hochreiter & Schmidhuber, 1997).

3.6.2.1 Network Architecture

The backbone of the model consists of a two-layer LSTM encoder with hidden dimension H. At each time step t, with an input vector $x_t \in \mathbb{R}^F$, the LSTM updates its cell state c_t and hidden state $h_t \in \mathbb{R}^H$, by the gating equations (Gers et al., 2000):

$$\begin{aligned} i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right), \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right), \\ \tilde{c}_t &= \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{3.50}$$

where σ is the logistic sigmoid, $\odot$ element-wise multiplication, and $\{\boldsymbol{W}*, \boldsymbol{U}*, \boldsymbol{b}*\}$ are learnable parameters. Dropout ($rate = \rho$) is used between the two LSTM layers to regularize dynamics of the recurrent model (Srivastava et al., 2014).

After processing an input window of size L, the last hidden state h_L is forwarded to a two-layer feed-forward head:

$$\hat{y} = W_2\Big(\mathrm{ReLU}(W_1 h_L + b_1)\Big) + b_2 \tag{3.51}$$

where $W_1 \in \mathbb{R}^{D\times H}, W_2 \in \mathbb{R}^{1\times D}$is the bottleneck dimension (set to 32), and ReLU serves to prevent vanishing gradients in deep non-linear layers (Nair & Hinton, 2010). This trade-off is appropriate when fitting small-sample time series (Gers et al., 2000).

3.6.2.2 Data Normalization & Sequence Construction

Before training, all feature columns (sentiment primitives, technical indicators, macro transforms) are standardized to zero mean and unit variance using only the training partition's statistics, as recommended to improve neural network convergence (LeCun, Bottou, Orr, & Müller, 2012). For each feature j, with training–set mean μ_j and standard deviation σ_j, the standardized value at month m is

$$\tilde{x}_{m,j} = \frac{x_{m,j} - \mu_j}{\sigma_j} \tag{3.52}$$

The target series (monthly close) is treated similarly when computing in-sample losses but inverse-scaled before final evaluation.

To frame the prediction problem as supervised learning, the standardized monthly panel $\{\tilde{x}_m\}_{m=1}^{M}$ is converted into overlapping sequences of length L (the "lookback" window). Denote the total number of usable samples by $N = M - L$. Then for each $n = 1, \dots, N$,

$$\mathbf{X}^{(n)} = \Big[\tilde{\mathbf{x}}_n, \tilde{\mathbf{x}}_{n+1}, \dots, \tilde{\mathbf{x}}_{n+L-1}\Big] \in \mathbb{R}^{L\times F}, y^{(n)} = Y_{n+L}, \tag{3.53}$$

where F is the number of features and Y_m the (unscaled) monthly close price. This yields input tensor $\mathbf{X} \in \mathbb{R}^{N\times L\times F}$ and target vector $\mathbf{y} \in \mathbb{R}^{N}$. Such sliding-window sequence construction is a standard approach for time-series neural models, enabling the LSTM to learn dependencies across L months (Bai, Kolter, & Koltun, 2018).

3.6.2.3 Hyperparameter Tuning & Model Selection

To identify the best LSTM architecture for each sentiment stream, a two-stage selection procedure is applied:

1. Grid Search Over Hyperparameters

   A Cartesian grid is defined over:

Table 3-15 LSTM Hyperparameter Grid

Parameter	Symbol	Values
Lookback window	L	{1}
Hidden layer size	H	{64,128}
Dropout rate	ρ	{64,128}
Learning rate	ℓ	$\{10^{-3}, 5 \times 10^{-4}\}$
Batch size	B	16

For each configuration $\theta = (L, H, \rho, \ \ell, B)$ the network is trained on the January 2020–February 2023 data with an internal 80/20 train/validation split. The optimization minimizes the mean squared error on the validation fold:

$$\mathrm{MSE}_{\mathrm{val}}(\theta) = \frac{1}{N_{\mathrm{val}}} \sum_{i=1}^{N_{\mathrm{val}}} \left(y^{(i)} - \hat{y}^{(i)}(\theta) \right)^2 \tag{3.54}$$

and uses early stopping (patience = 3–7 epochs) to prevent overfitting (Prechelt, 1998).

2. Out-of-Sample Forecast Evaluation

   The best-validation model for each θ is then used in a rolling-window forecast from April 2023 through December 2023. Its out-of-sample root-mean-square error is computed as:

$$\mathrm{RMSE}_{\mathrm{out}}(\theta) = \sqrt{\frac{1}{N_{\mathrm{test}}} \sum_{t=1}^{N_{\mathrm{test}}} \left(y_t - \hat{y}_t(\theta)\right)^2} \tag{3.55}$$

   where N_{test} is the number of forecast months. The configuration minimizing $RMSE_{out}$ is selected. If no valid forecasts can be generated (e.g. due to sequence-length constraints), the model with the lowest validation MSE is chosen as a fallback.

This two-stage methodology of validation-based early stopping before rolling-window forecast comparison promotes tuning of hyperparameters to generalization performance and, ultimately, predictive accuracy in a real-world context (Gers et al., 2000; Srivastava et al., 2014).

3.6.2.4 Forecast Generation

Once the optimal hyperparameters are identified (Section 3.6.2.3), the LSTM is retrained on the full in-sample window (January 2020 – February 2023). Two types of forecasts are then produced:

1. In-Sample Fitted Values

   For each training sample n, the trained network generates a one-step-ahead prediction

$$\hat{y}^{(n)} = f_\theta\left(\mathbf{X}^{(n)}\right) \tag{3.56}$$

where f_θ denotes the LSTM + head mapping with parameters θ, and $\mathbf{X}^{(n)} \in \mathbb{R}^{LxF}$ the nth lookback window. These fitted values assess how well the network learned the historical dynamics (Brownlee, 2017).

2. Walk-Forward Out-of-Sample Forecasts

   To simulate real-world deployment, forecasts from April 2023 through December 2023 are generated recursively. Starting with the final L months of the training block as the initial window, each new month's actual features x_t are appended to the history after forecasting, yielding

$$\hat{y}_{t+1} = f_\theta\left[\mathbf{x}_{t-L+1},\ldots,\mathbf{x}_t\right], \tag{3.57}$$

   then update history: $\{\mathbf{x}_{t-L+2},\ldots,\mathbf{x}_t,\mathbf{x}_{t+1}\}$

   This rolling-window procedure ensures that at each step the model sees only data available at forecast time, avoiding look-ahead bias (Hyndman & Athanasopoulos, 2018).

All fitted and forecasted values are inverse-scaled back to the original price units before evaluation. Three CSV outputs are saved per sentiment stream: in-sample predictions, out-of-sample forecasts, and the combined series (training + test). Corresponding plots overlay actual vs. predicted values over the full 2020–2023 horizon for easy visual comparison.

3.6.2.5 Model Evaluation & Selection

Accuracy of forecasts for each LSTM setup is measured on both the in-sample (Jan 2020–Feb 2023) and out-of-sample (Apr 2023–Dec 2023) periods using four common metrics (Armstrong, 2001; Hyndman & Athanasopoulos, 2018):

- Root Mean Squared Error (RMSE): sensitive to large errors, defined in Section 3.6.1.5.

- Mean Absolute Error (MAE): robust in outliers, defined in Section 3.6.1.5.
- Mean Absolute Percentage Error (MAPE): scale-invariant, defined in Section 3.6.1.5.
- Coefficient of Determination (R^2): proportion of variance explained, defined in Section 3.6.1.5.

The criterion for selecting primary is out-of-sample RMSE, for which the hyperparameter set θ minimizing

$$\mathrm{RMSE}_{\mathrm{out}}(\theta) = \sqrt{\frac{1}{N_{\mathrm{test}}} \sum_{t=1}^{N_{\mathrm{test}}} \left(y_t - \hat{y}_t(\theta)\right)^2} \tag{3.58}$$

is selected for the returned model. If the out-of-sample forecast cannot be made (e.g. because of the minimum sequence length), then the configuration with the highest out-of-sample R^2 or the lowest validation mean squared error (MSE) is used as a fallback (Prechelt, 1998).

The selected LSTM is then retrained on the full training block, and its in-sample and out-of-sample performance metrics are recorded and exported alongside those of ARIMAX for direct comparison in Chapter 4.

3.7 Model Validation and Evaluation

It is important to implement a sound validation scheme that allows the forecasting performance to reflect real predictive power and which is not merely a reflection of overfitting or data-leakage artifacts (Hyndman & Athanasopoulos, 2018). Both models—ARIMAX and LSTM—are tested on the same temporal sub-series, where the training window and the testing window are well separated. Performance is assessed with four complementary measures (RMSE, MAE, MAPE, R2) that capture sensitivity to large errors, robustness to outliers, scale-invariance, and explained variance (Armstrong, 2001; Hyndman & Athanasopoulos, 2018). An extra internal validation split used to track generalization during training of neural models (Prechelt, 1998). This multi-pronged analysis guarantees a fair, apples-to-apples comparison

between lexicon, machine-learning, and deep-learning sentiment inputs for both linear and non-linear modeling frameworks.

3.7.1 In-Sample vs. Out-of-Sample Evaluation

To mirror an operational forecasting setting, and to avoid look-ahead bias, each feature panel and target series is split chronologically:

- In-Sample (Train): January 2020 to February 2023.
- Hold-Out (Testing): April 2023 through December 2023.

March 2023 is held back as a "seed" observation for walk-forward forecast using LSTM to avoid not having lookback window being completely filled but not leaking any test information (Bergmeir & Benítez, 2012). No random shuffle is applied; rather, a strict time-ordered split, maintaining causality, mirror real-world deployment, where future predictions are made with only past data.

For LSTM models, an internal 80/20 train/validation split on the January 2020–February 2023 block is employed for early stopping (patience = 3-7 epochs) and learning-rate reduction to avoid overfitting and optimize generalization (Prechelt, 1998). The final evaluation on the hold-out window produces unbiased estimates of the predictive accuracy of each configuration.

3.7.2 Internal Validation and Hyperparameter Selection

ARIMAX models rely solely on the hold-out split (Section 3.7.1) for order selection: after fitting each candidate $(p, 1, q)$ on the in-sample block, out-of-sample RMSE on the testing window drives both per-threshold and overall best-order choice (Section 3.6.1.3). No additional cross-validation is performed, as temporal dependence and limited sample size make k-fold schemes inappropriate for time-series forecasting (Bergmeir & Benítez, 2012).

LSTM models incorporate an internal 80/20 train/validation split within the in-sample block to guide early stopping and learning-rate scheduling (Prechelt, 1998). In

particular, for each hyperparameter setting, 80 % of the January 2020–February 2023 data is trained and the 20 % is used for performance supervision. Validation mean-squared error

$$\mathrm{MSE}_{\mathrm{val}} = \frac{1}{N_{\mathrm{val}}} \sum_{i=1}^{N_{\mathrm{val}}} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 \tag{3.59}$$

controls early stopping (patience = 3–7 epochs) and reduces the learning rate when improvements stall. Only models that generalize on this hold-out fold are carried forward to the rolling-window out-of-sample evaluation (Section 3.6.2.3).

3.7.3 Forecast-Accuracy Metrics

Forecast accuracy for both ARIMAX and LSTM models is quantified using the four standard metrics which are RMSE, MAE, MAPE and R^2 as defined in Sections 3.6.1.5 and 3.6.2.5. This ensures a consistent, apples-to-apples comparison of predictive performance across all sentiment methodologies and modeling frameworks.

3.8 Comparison Strategy

To ensure a truly fair "apples-to-apples" comparison, each of the six configurations which are lexicon vs. ML vs. DL sentiment, each paired with ARIMAX or LSTM, is developed and evaluated under an identical experimental protocol. All models consume the same monthly-aggregated feature panels (sentiment + technical ± macro), use the same chronological train/test split (Jan 2020–Feb 2023 vs. Apr 2023–Dec 2023), and apply the same hyperparameter search logic and internal validation procedures. Adhering to best practices in time-series forecasting, this uniform pipeline guarantees that any performance differences arise solely from the sentiment methodology or model architecture, rather than from data leakage or inconsistent tuning (Hyndman & Athanasopoulos, 2018).

Forecast accuracy is then assessed with a suite of complementary metrics (RMSE, MAE, MAPE and R^2) capturing error magnitude, robustness to outliers, relative scale,

and variance explanation in tandem (Hyndman & Koehler, 2006; Armstrong, 2001). These statistics are collected into a single comparative table and visualized via grouped bar charts (for metric-by-metric comparison) and forecast-overlay plots (depicting actual vs. predicted series). Such multi-metric, multi-model presentation facilitates transparent, reproducible insights into which sentiment pipeline most enhances predictive performance under each modeling framework.

3.9 Ethical Considerations and Limitations

Applying media-based sentiment to support quantitative forecasting involves ethical considerations and methodological limitations. First of all, automated sentiment analysis can lead to the perpetuation of biases present in its training data. News coverage may over-represent sensational or negative aspects (the "negativity bias") and under-represent more mundane events, biasing the sentiment signals and possibly intensifying unwarranted market fears (Soroka 2006; Tetlock 2007). Moreover, transformer-based classifiers trained on social or financial text inherit biases from their pretraining corpora, such as over- or under-weighting certain topics or language styles, which may distort the true economic signal (Bolukbasi et al., 2016).

Second, information-asymmetry and market impact concerns arise if algorithmic forecasts based on public sentiment become widely adopted. If a large number of market participants follow the same sentiment signals in trading, feedback-loops may arise where model-driven trades confirm and reinforce the media tone instead of the underlying fundamentals (Hendershott & Riordan, 2013). This reflexivity can amplify the volatility, compromise the efficiency of markets and raising questions about the social responsibility of using automated forecasting systems (Madhavan, 2000).

From a methodological standpoint, the use of Tesla headlines undermines generalizability of this study. Tesla is such high-profile and idiosyncratic company (e.g. executive tweets) sentiment-price relationships may differ from those of other firms (Barber & Odean, 2008). The monthly aggregation horizon filters out intraday noise, but may neglect rapid changes in sentiment affecting short-term volatility. Likewise, ARIMAX supposes that coefficients are constant and relations are linear

across time points which might be invalid due to structural brakes (e.g. policy alterations and crises) (Stock & Watson, 2002). Although LSTM models are flexible, they are still 'black-box' models with very limited interpretation, and have a tendency to overfit to the sparse monthly data despite early-stopping measures (Lipton, 2016).

Lastly, macro and technical indicators from exogenous sources contain measurement error in the first place and are also subject to revisions themselves (Bloom et al., 2018). Forward-filling over non-trading days does not account for sentiment decay, which could be inconsistent with the true flow of market information. Given these limitations, the findings in Chapter 4 should be considered as indicative and not definitive, with additional work required to confirm them across a broad range of assets, frequencies, and more robust bias-mitigation methods.

3.10 Chapter Summary

This chapter has presented the complete methodology process to combine sentiment analysis with traditional financial forecasting Section 3.3 described data sourcing and stationarity checks, while Section 3.4 introduced three parallel sentiment pipelines—lexicon-based, machine-learning, and deep-learning—each standardized through a six-step feature-construction pipeline. Section 3.5 then augmented these sentiment signals with monthly technical and macroeconomic indicators, applying rigorous screening (ADF, correlation, Granger causality) to select exogenous features.

Within Sections 3.6–3.7, two forecasting systems, linear ARIMAX and non-linear LSTM, were defined, tuned and cross-validated using the same train/test splits, neural-specific internal validation and the same performance metrics (RMSE, MAE, MAPE, R^2). Such a common protocol guarantees that differences in predictive performance can be attributed only to the sentiment method or model design, and not to the data or the evaluation artifacts. Finally, Section 3.9 discussed ethical considerations and methodological limitations.

The empirical results, namely comparing the out-of-sample forecast performance of all six sentiment × model specifications to one another, as well as independent

interpretations of the results, are reported in Chapter 4 followed by an analysis of the robustness of our findings.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter presents the empirical results that address the three core objectives of this study: (1) quantifying the relationship between sentiment signals and Tesla stock returns, (2) comparing the forecasting performance of ARIMAX, machine-learning classifiers and transformer-embedding models, and (3) evaluating LSTM's sequential forecasting capability. Daily Tesla closing prices (January 2020–December 2023) were analyzed, monthly macroeconomic indicators from FRED, technical indicators (returns, volatility, RSI, MACD) and three parallel sentiment streams—lexicon-based (VADER, Loughran–McDonald), ML-based classifiers (TF-IDF, BoW, Word2Vec) and transformer embeddings (FinBERT, RoBERTa, DistilBERT). Each pipeline was tuned via grid search and evaluated on out-of-sample RMSE, MAE, MAPE and R^2; computational runtime was also recorded. Only the single lowest-RMSE configuration per paradigm is reported here, while exhaustive grid-search logs and full VIF screening tables are provided in the appendices.

The chapter unfolds as follows. Section 4.2 offers descriptive and exploratory analysis, including summary statistics, a correlation heatmap and a key time-series overlay of price versus the top sentiment stream. Section 4.3 details the VIF-based feature-selection procedure and presents the final feature sets for each paradigm. Section 4.4 summarizes predictive performance for ARIMAX and the three sentiment–model pipelines, reporting best model specifications alongside their out-of-sample metrics. Section 4.5 describes the LSTM forecasting pipeline—architecture, hyperparameters and forecast accuracy—accompanied by actual-versus-predicted plots. Section 4.6

provides a head-to-head comparison of predictive accuracy and computational trade-offs across all paradigms. Finally, Section 4.7 distills the key findings and links them back to the original research questions.

4.2 Descriptive & Exploratory Analysis

This section provides a first look at the data and uncovers preliminary relationships to guide the modeling steps.

4.2.1 Stationarity Testing of Macroeconomic Indicators

Stationarity of each macro series was assessed using the Augmented Dickey–Fuller test (Dickey & Fuller, 1979) at the level, first difference and where needed second difference. Figure 4.1(a–c) plots the ADF p-values for all thirteen series across these three stages.

Step 1: ADF Test on Original Series

Feature	ADF p-value	Status
Consumer Price Index (CPI)	0.6104	Non-Stationary
Personal Consumption Expenditures (PCE)	0.0012	Stationary
Producer Price Index (PPI)	0.1622	Non-Stationary
Unemployment Rate	0.0599	Non-Stationary
Non-Farm Payrolls (NFP)	0.0010	Stationary
Job Openings (JOLTS)	0.0011	Stationary
Gross Domestic Product (GDP)	0.4364	Non-Stationary
Industrial Production	0.4609	Non-Stationary
Retail Sales	0.0582	Non-Stationary
Federal Funds Rate	0.8022	Non-Stationary
10Y Treasury Yield	0.9078	Non-Stationary
M2 Money Supply	0.2478	Non-Stationary
Consumer Confidence Index	0.7125	Non-Stationary

Figure 4-1a ADF test p-values for each of the thirteen macroeconomic series in their original (level) form

Step 2: ADF Test after First Differencing

Feature	ADF p-value (1st Diff)	Status
Consumer Price Index (CPI)	0.0076	Stationary
Producer Price Index (PPI)	0.6489	Non-Stationary
Unemployment Rate	0.0000	Stationary
Gross Domestic Product (GDP)	0.8622	Non-Stationary
Industrial Production	0.5375	Non-Stationary
Retail Sales	0.7833	Non-Stationary
Federal Funds Rate	0.2213	Non-Stationary
10Y Treasury Yield	0.0000	Stationary
M2 Money Supply	0.0019	Stationary
Consumer Confidence Index	0.0000	Stationary

Figure 4-1b ADF test p-values after first differencing

Step 3: ADF Test after Second Differencing

Feature	ADF p-value (2nd Diff)	Status
Producer Price Index (PPI)	0.0006	Stationary
Gross Domestic Product (GDP)	0.0000	Stationary
Industrial Production	0.0003	Stationary
Retail Sales	0.0000	Stationary
Federal Funds Rate	0.0005	Stationary

Figure 4-1c ADF test p-values after second differencing, applied to the remaining non-stationary series

At level (Figure 4.1a), only Personal Consumption Expenditures (PCE), Non-Farm Payrolls (NFP) and Job Openings (JOLTS) reject the unit-root null hypothesis ($p \leq 0.05$), indicating they are $I(0)$. After first differencing (Figure 4.1b), five additional series, which are Consumer Price Index (CPI), Unemployment Rate, 10-Year Treasury Yield, M2 Money Supply and Consumer Confidence Index, become stationary ($p \leq 0.05$), marking them as $I(1)$. The remaining indicators (Producer Price Index, Gross Domestic Product, Industrial Production, Retail Sales and Federal Funds Rate)

require a second difference (Figure 4.1c) to achieve stationarity ($p \leq 0.05$), and are thus treated as $I(2)$.

These integration orders determine the differencing applied to each macro series in the ARIMAX exogenous panel (Section 4.3), while the raw (level) forms of the same indicators are later re-evaluated for inclusion in the LSTM pipeline (Section 4.2.3).

4.2.2 Pearson-Correlation Screening of Stationary Macros

Pearson correlation coefficients were computed between each of the thirteen differenced macro series and the raw monthly Tesla closing-price series. A selection threshold of $|r| \geq 0.30$ was applied to retain only those indicators exhibiting at least a medium linear association (Cohen, 1988). Table 4.1 lists the six stationary macro variables that satisfy this criterion and are therefore included as exogenous regressors in the ARIMAX models.

Table 4-1 Stationary macro variables with |r|≥0.30 against raw Tesla Close

Variable	Integration Order	Pearson r
Job Openings (JOLTS)Δ_1	$I(1)$	0.762
M2 Money SupplyΔ_1	$I(1)$	-0.439
Personal Consumption ExpendituresΔ_0	$I(0)$	0.555
Consumer Price Index (CPI) Δ_1	$I(1)$	0.541
10-Year Treasury YieldΔ_1	$I(1)$	0.455
Non-Farm Payrolls (NFP) Δ_0	$I(0)$	0.411

Note: $\Delta_0$indicates the series was already stationary at level $I(0)$; Δ_1 indicates first differencing to achieve stationarity $I(1)$.

These six differenced indicators constitute the exogenous input panel for all ARIMAX model specifications. In Section 4.2.3, their undifferenced (raw) counterparts are re-tested against the raw closing price to determine inclusion in the LSTM pipeline.

4.2.3 Pearson-Correlation Screening of Raw Macros

Each of the six stationary macro indicators retained for ARIMAX (Section 4.2.2) was next evaluated in its *raw* (undifferenced) form against the raw monthly Tesla closing-price series. Pearson correlations were recomputed, and the same selection threshold $|r| \geq 0.30$ was applied to determine inclusion in the LSTM feature panel. Table 4.2 presents the results.

Table 4-2 Raw macro variables with |r|≥0.30 against raw Tesla Close

Variable	Pearson r
Job Openings (JOLTS)	0.762
M2 Money Supply	0.811
Personal Consumption Expenditures (PCE)	0.555
Consumer Price Index (CPI)	0.458
10-Year Treasury Yield	0.316
Non-Farm Payrolls (NFP)	0.411

All six raw indicators exceed the $|r| \geq 0.30$ threshold and are thus included alongside the technical and sentiment features in the LSTM forecasting pipeline.

4.2.4 Feature Price Relationship Analysis

This section addresses the price history of Tesla over the sample period and its co-movement with sentiment. First, the unfiltered raw series and the month-integrated series are drawn to show major trends and turning points. Then a one-month-lagged VADER compound score is plotted over the monthly price to show the strong co-movements. And lastly, three Pearson-correlation heatmaps measure the strength of the relationship between the price level and each group of features.

4.2.4.1 Feature–Price Relationship Analysis

Figure 4.2a visualises month-end Tesla closing prices from January 2020 to December 2023. Three distinct phases appear over the course of this period. For one thing, a prolonged rally hoisted the price from around $40 in early 2020 to more than $370 by the middle of 2021, fueled in part by investor optimism about the economic rebound from the pandemic. After the peak a tough correction set in, Tesla fell from its highs toward the $120 mark in early 2023, as macro headwinds and profit-taking took over. Many believed it would never see $300 again, but by the end of 2023 the price was ranging between $200 and $300, as new but jaded confidence has returned to the stock.

This non-stationary, long term behavior justifies the differencing operation discussed in Section 4.2.1. The monthly closing price series will act as the target variable, y_m, in both the ARIMAX and LSTM modelling frameworks presented in Sections 4.3 and 4.4, respectively.

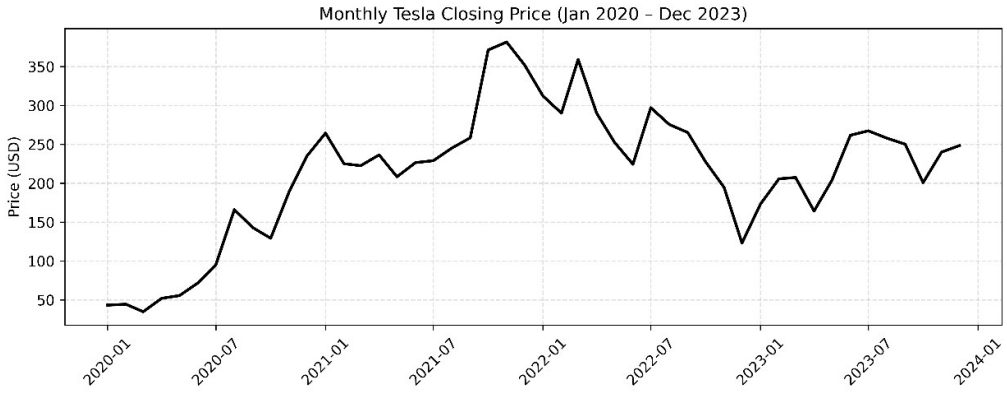


Figure 4-2a Monthly Tesla closing price (end of month) from January 2020 through December 2023.

4.2.4.2 Feature–Price Relationship Analysis

Figure 4.2b shows the monthly ending Tesla closing price and three VADER sentiment categories Compound Score (Purple), Positives Proportion (Green), and Negatives Proportion (Red) from Jan 2020 to Dec 2023. The VADER scores (one month lagged) are indicated on the right, and price on the left.

A number of things leap out from this. First, when the compound score was rising and staying above a certain level (like mid-2020 and early 2021), price trend was on the upside, hinting that high level positive sentiment can lead or accompany price gains. By contrast, the sharp drop in the compound score in mid-2021, late 2021, and early 2023 precedes or coincides with steeper price corrections. And the negative-sentiment (red) line has also had spikes ahead of big drawdowns like October 2021, with the positive-sentiment (green) line is fairly high during bull markets.

Such visual co-movements motivate us to consider lexicon-based sentiment to be fed directly as exogenous input to our forecasting pipelines. While the focus of this overlay is VADER, similar patterns (with variations in magnitude) were observed for the Loughran–McDonald, machine-learning, and deep-learning sentiment scores.

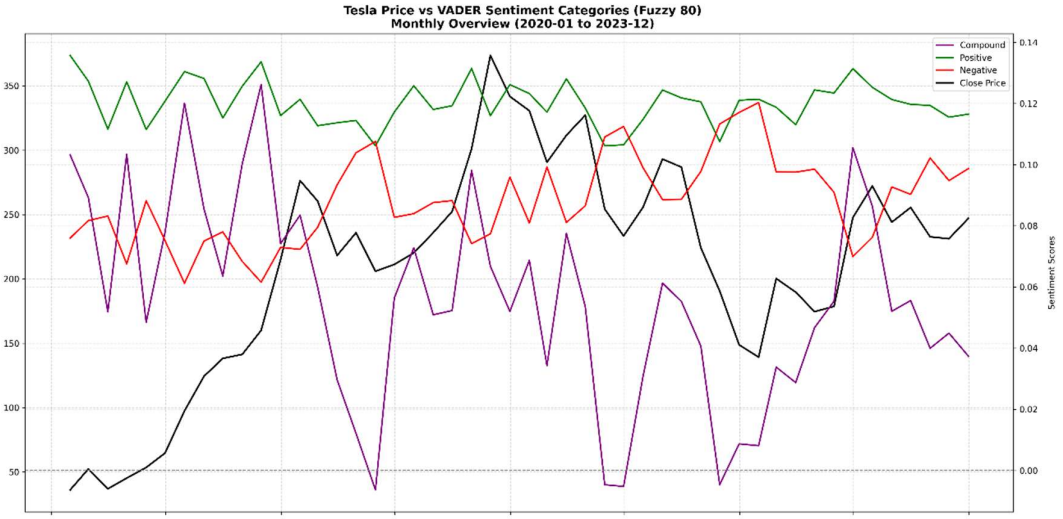


Figure 4-2b Tesla monthly closing price (black) and VADER sentiment categories (lag 1): compound score (purple), positive proportion (green), and negative proportion (red).

4.2.4.3 Correlation of Selected Features with Tesla Closing Price

In Figure 4.3a (Macro vs. Close) show that the highest covariate in the positive domain with TSLA month-end price is the M2 Money supply ($r = 0.81$) and the Job Openings ($r = 0.76$), respectively, while all others, ranging from the PCE ($r = 0.56$) and the CPI ($r = 0.46$) via the 10-year Treasury yields ($r = 0.32$) to the Non-Farm Payrolls ($r = 0.41$), are moving also in the same direction, but with more moderate alignment.

Figure 4.3b (Technical and Close) shows a strong negative relationship between price and three month realized volatility ($r = -0.50$) which means that low prices come associated with higher volatility. Positive trace is followed by RSI ($r = 0.28$) and MACD histogram ($r = 0.24$), although both perform far less well, while daily logs show virtually no linear association ($r \approx 0$).

The sentiment panel in Figure 4.3c (Sentiment vs. Close) confirms that only DL-derived mean sentiment ("DL_SentMean") is substantially correlated (r = –0.41) with Tesla's closing price, which is an intriguing negative relationship. Whereas Loughran–McDonald net sentiment, VADER compound score, and the classical ML sentiment mean all lie close to zero, indicating very little contemporaneous co-movement with price.

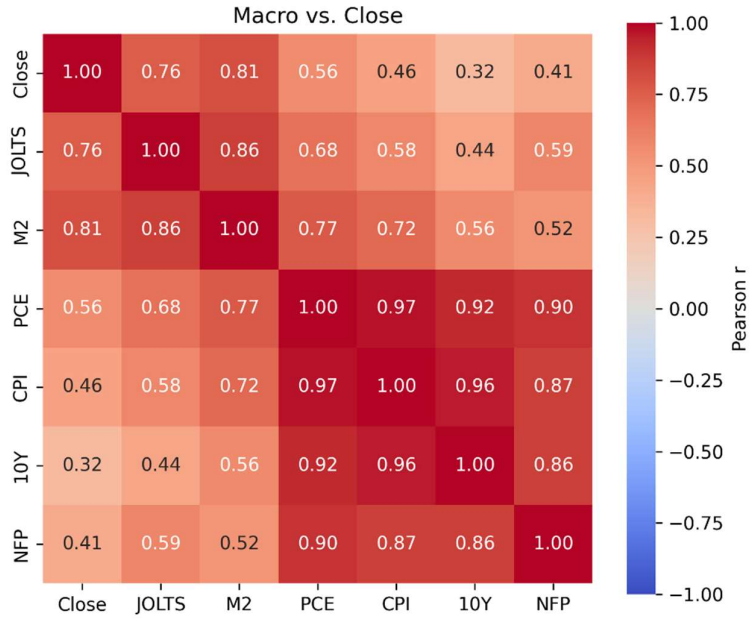


Figure 4-3a Pearson correlations between Tesla's monthly close and six macro indicators: JOLTS, M2, PCE, CPI, 10-year yield, and NFP (Jan 2020–Dec 2023).

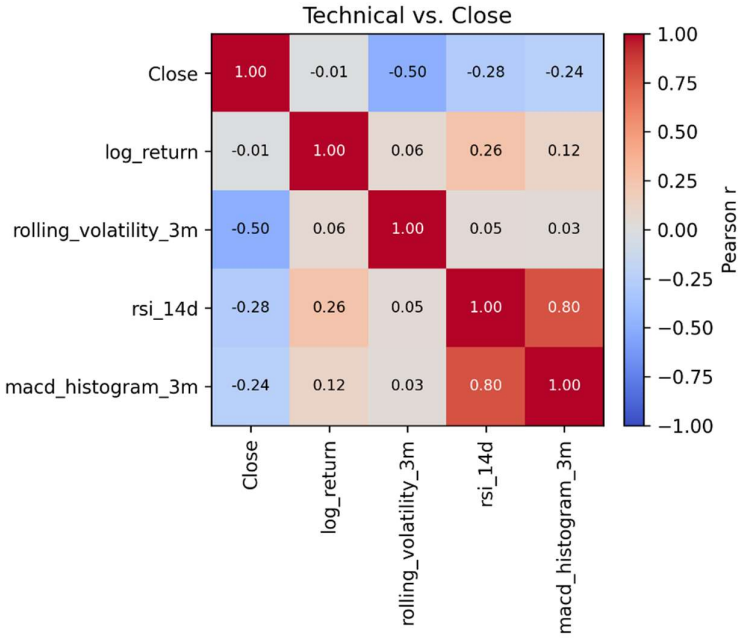


Figure 4-3b Pearson correlations between Tesla's monthly close and four technicals: log returns, 3m volatility, 14-day RSI, and 3 m MACD histogram (Jan 2020–Dec 2023).

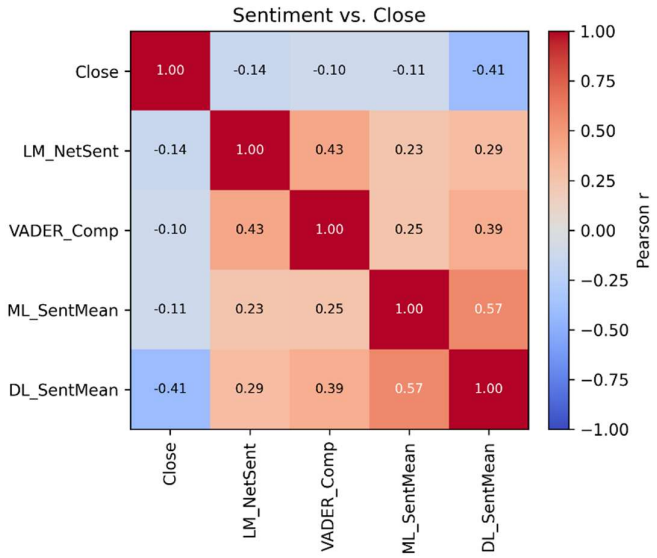


Figure 4-3c Pearson correlations between Tesla's monthly close and one sentiment series per method: Loughran–McDonald net, VADER compound, ML mean, and DL mean (Jan 2020–Dec 2023).

4.3 Final Feature Panels for ARIMAX and LSTM Forecasting

In the final feature-assembly stage, all three signal domains which are macroeconomic, technical and sentiment, are brought together in two parallel pipelines tailored to the modeling framework.

For ARIMAX, each monthly macro series is first tested for unit roots and differenced until stationary (Section 3.3.3.2), then screened via Pearson correlation (Sections 3.5.2, 4.2.2–4.2.3). The macro indicators that pass this correlation threshold are retained for two exogenous setups, "Sent + Tech + Macro" and "Sent + Tech." All seven technical indicators (Section 3.5.1) undergo the same ADF differencing, and sentiment features from three sources which are lexicon-based (Section 3.4.2), machine-learning classifiers (Section 3.4.3.6–3.4.3.7), and deep-learning transformers (Section 3.4.4.3–3.4.4.4) are likewise differenced. These stationary macro, technical, and sentiment series are then jointly subjected to a grid search over variance-inflation thresholds (3, 5, 10, 15, 30). By selecting the subset of predictors that minimizes out-of-sample RMSE, we ensure the final exogenous panels are both parsimonious and effective for ARIMAX.

By contrast, in the LSTM pipeline all inputs remain at their raw, level form to allow the network to learn complex, nonlinear patterns without imposing pre-whitening or collinearity constraints. The same six macro series (in levels), the full set of seven technical features and the undifferencing sentiment descriptors from lexicon, ML and DL models are concatenated month-by-month and fed directly into the recurrent network.

By contrast, the LSTM pipeline retains all inputs in their raw, level form—six macro series, seven technical features, and the untransformed sentiment descriptors from lexicon, ML, and DL models. This allowing the network to learn complex nonlinear interactions without pre-whitening or collinearity constraints.

This two-track construction has the advantage that (i) the ARIMAX model will make best use of the predictors which are- rigorously-stationarized and de-collinearized, i.e.,

the best predictors in some sense for autoregressive prediction while (ii) the LSTM model will make best use of the raw richness of economic, price and textual signals.

4.4 ARIMAX Model Result

This section shows the out-of sample performance for 10 ARIMAX pipelines on the two set of features: “Sentiment + Tech only” and “Sentiment + Macro + Tech”. For each pipeline, to find the optimal ARIMAX parameter order, fuzzy threshold (for the string matching), VIF cutoff, and feature-engineering options, the parameter configuration that minimizes the RMSE was determined for the two lexicon methods (Loughran–McDonald and VADER), five machine learning classifiers (Logistic Regression, SVM, Naïve Bayes, Random Forest, and XGBoost), and three transformer embeddings (DistilBERT, FinBERT, and RoBERTa). The obtained R^2, RMSE, MAE and MAPE values for the best config- urations of each model are showed in Tables 4.4a and 4.4b.

4.4.1 ARIMAX Out-of-sample Accuracy

Table 4.3a reports the four key metrics for each ARIMAX pipeline when macroeconomics indicators are added to technical and sentiment inputs. Logistic Regression on a fine-tuned bag-of-words representation with feature enhanced (ARIMAX order: (1,1,4), 50 % agree level; VIF = 5; fuzzy = 80) attains the highest explanatory power ($R^2 = 0.95$) and the lowest forecast errors (RMSE = 7.05; MAE = 5.53; MAPE = 2.27 %). Non feature enhanced Naïve Bayes pipeline follow closely (ARIMAX order: (0,1,4), 50 % agree level; VIF = 30; fuzzy = 85), with $R^2 = 0.92$ and RMSE = 9.29, reflecting a modest trade-off of greater bias for slightly reduced variance.

Concerning lexicon-based variables, the Feature-enhanced Loughran–McDonald pipeline (ARIMAX order: (2,1,4), VIF: 10, fuzzy: 80) obtains $R^2 = 0.86$ and RMSE = 12.32, which is better than VADER ($R^2 = 0.77$, RMSE = 15.72). Transformer-based embeddings give mixed results: RoBERTa (ARIMAX order = (0,1,1); VIF = 10;

fuzzy = 80) achieves $R^2 = 0.93$, RMSE = 8.67, yet FinBERT and DistilBERT still group near $R^2 \approx 0.81$ and RMSE ≈ 14.

Exclusion of macroeconomic indicators (Table 4.3b) produces a uniform decline in accuracy: RMSE increases by 1.5–2.7 points and R^2 falls by 0.02–0.18 across all pipelines (Visualized barplots compare between macro + tech and tech only can be view on Figure 4.4a to c). Logistic Regression's RMSE rises to 9.70 ($R^2 = 0.91$), while RoBERTa's error more than doubles (RMSE = 16.44; $R^2 = 0.75$), underscoring the additive value of macroeconomic signals. Overall, finely tuned machine-learning representations of sentiment when combined with exogenous macro and technical factors can be deliver the most robust and precise ARIMAX forecasts.

Table 4-3a ARIMAX metric performance of Macro + Tech across three sentiment models

Sentiment Model	Metric			
	R^2	RMSE	MAE	MAPE
Loughran–McDonald	**0.86**	**12.32**	**11.34**	**4.87**
VADER	0.82	14.12	11.13	5.43
Logistic Regression	**0.95**	**7.05**	**5.53**	**2.27**
SVM	0.91	9.93	8.15	3.64
Naive Bayes	0.92	9.29	7.67	3.32
Random Forest	0.77	9.74	7.58	3.24
XGBoost	0.90	10.23	7.97	3.49
DistilBERT	0.81	14.52	10.60	4.38
FinBERT	0.81	14.21	10.96	4.96
RoBERTa	**0.93**	**8.67**	**6.61**	**2.94**

Table 4-3b ARIMAX metric performance of Tech only across three sentiment models

Sentiment Model	Metric			
	R^2	RMSE	MAE	MAPE
Loughran–McDonald	**0.82**	**14.12**	**10.98**	**4.59**
VADER	0.78	15.33	10.52	4.32
Logistic Regression	0.91	9.70	7.40	3.18
SVM	**0.93**	**8.80**	**7.07**	**3.04**
Naive Bayes	0.89	10.88	8.46	3.64
Random Forest	0.90	10.20	8.32	3.67
XGBoost	0.91	9.69	7.25	3.11
DistilBERT	**0.78**	**15.45**	**12.39**	**5.64**
FinBERT	0.78	15.66	10.33	4.35
RoBERTa	0.75	16.44	12.19	5.16

4.4.2 ARIMAX Forecast Plot Comparison

In order to illustrate the practical impact of including macroeconomic information in each sentiment-augmented ARIMAX pipeline, the single lowest-RMSE specification for "Sentiment + Tech only" was paired with its counterpart under "Sentiment + Macro + Tech." In each case, both forecasts share identical sentiment preprocessing and ARIMAX configuration, differing only in the addition of the full set of six macroeconomic indicators in the latter.

Figure 4.4a depicts the hold-out forecasts for the Loughran–McDonald pipeline without macro inputs (ARIMAX (0,1,4); VIF = 10; fuzzy = 90), which yields an out-of-sample RMSE of 14.12, R^2 of 0.82, MAE of 10.98 and MAPE of 4.59 %. Although the red forecast line broadly follows the upward trajectory of TSLA's closing price, it deviates by as much as ±20–25 USD during sharp market swings. When the same model is re-estimated with feature-enhanced Loughran–McDonald sentiment plus the

six macroeconomic and technical signals (ARIMAX (2,1,4); VIF = 10; fuzzy = 80), as shown in Figure 4.4b, RMSE falls to 12.32, R^2 rises to 0.86, MAE becomes 11.34 and MAPE is 4.87 %. In this configuration, those deviations narrow to within ±10–15 USD, and the forecast more accurately captures both peaks and troughs.

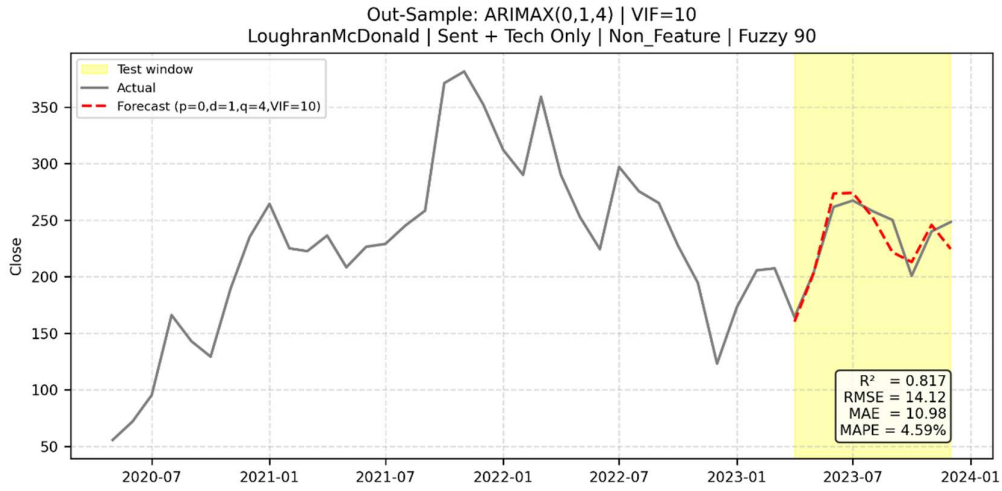


Figure 4-4a Forecast plot for ARIMAX (0,1,4) with non-feature-enhanced Loughran–McDonald sentiment (Sent + Tech only).

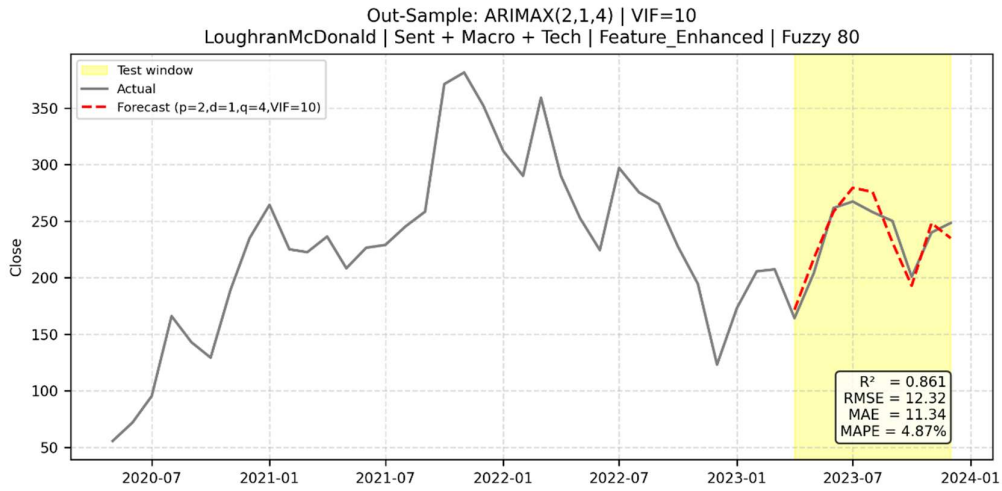


Figure 4-4b Forecast plot for ARIMAX (2,1,4) with feature-enhanced Loughran–McDonald sentiment (Sent + Macro + Tech).

A similar improvement appears in the machine-learning–based pipeline. Without macro inputs, the fine-tuned BoW Logistic Regression model (ARIMAX (1,1,4); VIF = 10; fuzzy = 80) records an RMSE of 9.76 (R^2 = 0.913) and exhibits pronounced errors during periods of heightened volatility (Figure 4.4c). Incorporating the six macroeconomic indicators reduces RMSE to 7.05 (R^2 = 0.954) and yields a forecast (Figure 4.4d) that remains tightly bound around actual prices, even through rapid rises and corrections.

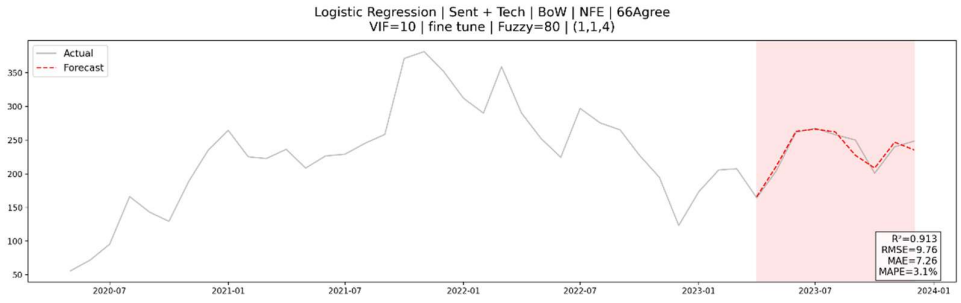


Figure 4-4c Forecast plot for ARIMAX (1,1,4) Logistic Regression on BoW-NFE (Sent + Tech only).

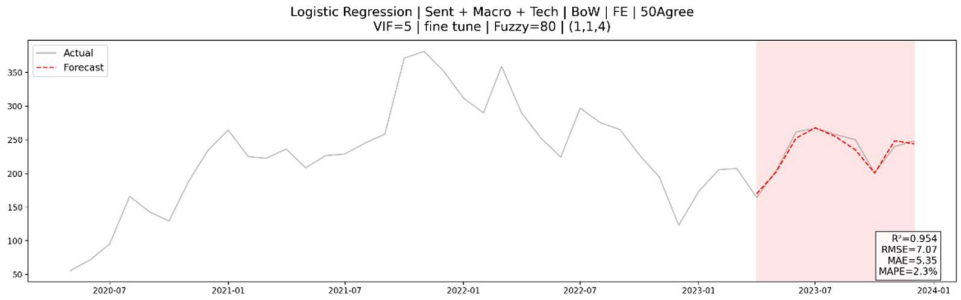


Figure 4-4d Forecast plot for ARIMAX (1,1,4) Logistic Regression on BoW-FE (Sent + Macro + Tech).

Among transformer-embedding pipelines, RoBERTa demonstrates the most dramatic macro-driven gain. Its “tech only” model (ARIMAX (1,1,2); VIF = 10; fuzzy = 75) incurs an RMSE of 16.44 and forecast errors exceeding ±30 USD (Figure 4.4e). Adding the macro indicators to the feature-enhanced RoBERTa configuration (ARIMAX (0,1,1); VIF = 10; fuzzy = 80) cuts RMSE in half to 8.67 (R^2 = 0.931) and

produces a forecast (Figure 4.4f) that closely tracks the actual series with markedly reduced lag and overshoot.

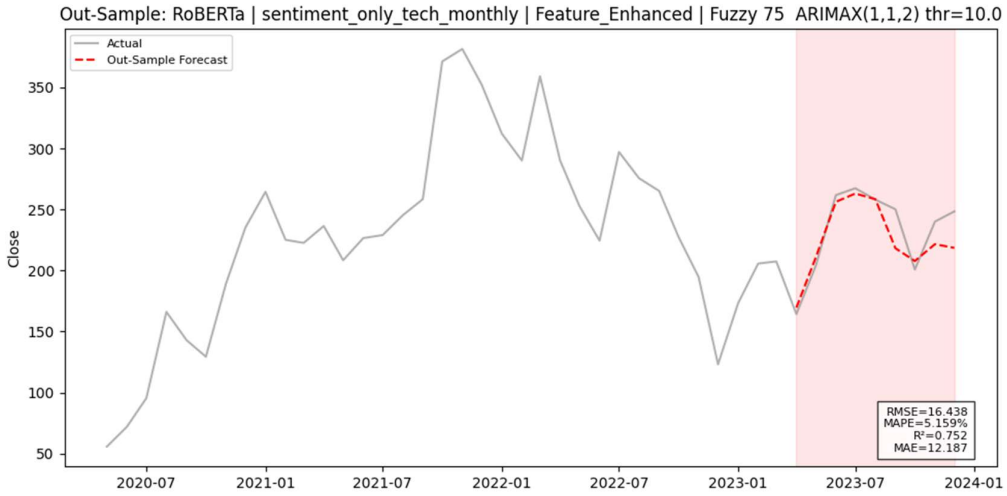


Figure 4-4e Forecast plot for ARIMAX (1,1,2) with feature-enhanced RoBERTa embedding (Sent + Tech only).

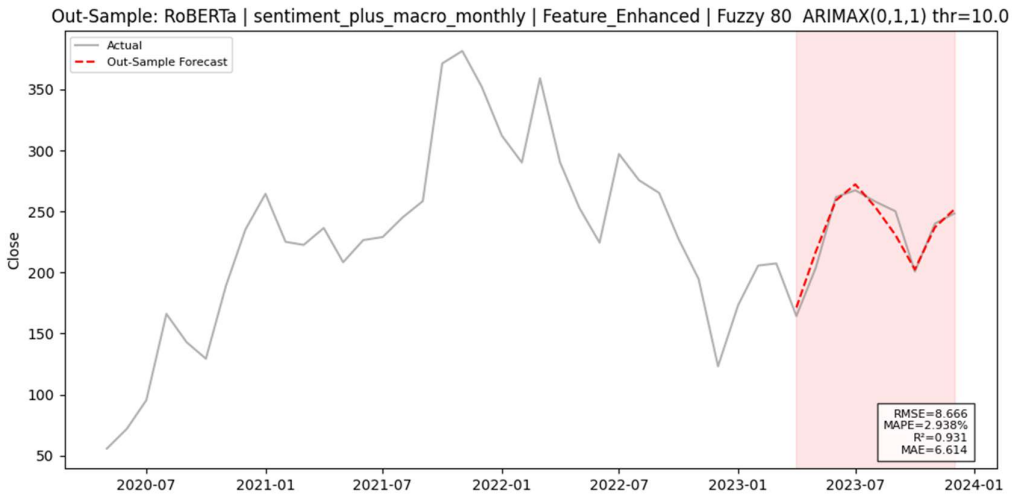


Figure 4-4f Forecast plot for ARIMAX(0,1,1) with feature-enhanced RoBERTa embedding (Sent + Macro + Tech).

4.4.3 Incremental Value of Macro Indicators

To evaluate the additional contribution of macroeconomic insights, the difference was calculated between a subset of key measures for each "Sentiment + Tech only" model and its "Sentiment + Macro + Tech" equivalent. In the case of the Loughran–McDonald pipeline, use of six macro variables decreased RMSE by 1.80 (from 14.12 to 12.32), and increased R^2 by 0.04 (from 0.82 to 0.86), indicating a significant improvement. The Logistic Regression – BoW model improved even further: RMSE decreased by 2.71 (9.76 → 7.05) and R^2 increased by 0.041 (0.913 → 0.954). In the transformer space, RoBERTa also showed the greatest macro-level boost where RMSE was halved (16.44 → 8.67) and R^2 increased by 0.181 (0.750 → 0.931). The VADER pipeline, in contrast, exhibited a slight increase in RMSE (15.33 → 15.72) and small decrease in R^2 (0.78 → 0.77), indicating that rogue, noisy lexical counts clearly went against the general macroeconomic visibility. In general, these deltas demonstrate that, regardless the way sentiment is estimated, macroeconomic indicators are able to improve the stabilization of both the baseline and the enhanced ARIMAX forecasts, and usually quite significantly.

4.5 LSTM Forecasting Results

This section tests the out of sample accuracy for all LSTM pipelines with two features sets "Sentiment + Tech only" and "Sentiment + Macro + Tech". For each sentiment-analysis model which are two dictionary methods (Loughran–McDonald, VADER), five machine-learning classifiers (Logistic Regression, SVM, Naive Bayes, Random Forest, XGBoost), and three transformer embeddings (DistilBERT, FinBERT, RoBERTa) with a search of hyperparameters (lookback window, hidden size, dropout, learning rate) was conducted to minimize RMSE. The resulting R^2, RMSE, MAE and MAPE are summarized in Tables 4.5a and 4.5b for the best performing configuration in each feature set.

4.5.1 LSTM Out-of-sample Accuracy

Table 4.4a displays the four key metrics for each LSTM pipeline when the full suite of six macroeconomic indicators is added to sentiment and technical inputs. Among lexicon-based models, the Loughran–McDonald pipeline (hidden size = 128; dropout = 0.0; learning rate = 1×10^{-3}; fuzzy threshold = 90) attains $R^2 = 0.24$, RMSE = 28.71, MAE = 24.03 and MAPE = 11.03 %. VADER, using the same architecture but a fuzzy threshold of 75, achieves $R^2 = 0.55$, RMSE = 22.93, MAE = 19.01 and MAPE = 9.00 %.

Machine-learning classifiers deliver substantially stronger fits. The SVM pipeline employing a fine-tuned BoW vectorizer at a 66 % agree level (lookback = 1, hidden size = 128, dropout = 0.2, learning rate = 1×10^{-3}, batch size = 16 and fuzzy threshold= 85) reaches $R^2 = 0.86$, RMSE = 12.47, MAE = 8.76 and MAPE = 3.77 %. Logistic Regression on a original BoW representation (66 % agree; lookback = 1; hidden size = 64; dropout = 0.0; learning rate = 1×10^{-3}; batch size = 16; fuzzy threshold = 80) follows with $R^2 = 0.79$, RMSE = 15.18, MAE = 13.08 and MAPE = 6.02 %. Naive Bayes, Random Forest and XGBoost occupy the middle ground ($R^2 \approx 0.60$–0.74; RMSE ≈ 16.85–20.93).

Transformer-embedding pipelines exhibit mixed results. DistilBERT (hidden size = 128; dropout = 0.0; learning rate = 1×10^{-3}; fuzzy = 80) produces $R^2 = 0.53$, RMSE = 22.76, MAE = 19.36 and MAPE = 9.02 %. FinBERT, (hidden size = 64; dropout = 0.2; learning rate = 5×10^{-4}) with fuzzy threshold = 90, records $R^2 = 0.49$, RMSE = 23.60, MAE = 18.34 and MAPE = 8.74 %. (hidden size = 128; dropout = 0.0; learning rate = 5×10^{-4}; fuzzy = 75) fares worst, with $R^2 = 0.03$, RMSE = 32.59, MAE = 23.97 and MAPE = 11.92 %.

When macroeconomic indicators are omitted (Table 4.4b), lexicon models show little improvement. Loughran–McDonald falls to $R^2 = 0.04$ (RMSE = 32.38; MAE = 28.26; MAPE = 12.31 %) and VADER to $R^2 = 0.52$ (RMSE = 22.93; MAE = 18.96; MAPE = 9.09 %). In contrast, machine-learning pipelines strengthen: SVM reaches $R^2 = 0.94$

, RMSE = 8.36, MAE = 7.50 and MAPE = 3.28 %, while Logistic Regression obtain R^2 = 0.92, RMSE = 9.37, MAE = 7.95 and MAPE = 3.39 %. Transformer models also improve modestly without macro inputs (DistilBERT: R^2 = 0.57, RMSE = 21.74; FinBERT: R^2 = 0.58, RMSE = 21.38; RoBERTa: R^2 = 0.59, RMSE = 21.13).

Table 4-4a LSTM metric performance of Macro + Tech across three sentiment models

Sentiment Model	Metric			
	R^2	RMSE	MAE	MAPE
Loughran–McDonald	0.24	28.71	24.03	11.03
VADER	**0.55**	**32.38**	**19.01**	**9.00**
Logistic Regression	0.79	15.18	13.08	6.02
SVM	**0.86**	**12.47**	**8.76**	**3.77**
Naive Bayes	0.60	20.93	15.69	7.37
Random Forest	0.74	16.85	12.71	5.81
XGBoost	0.67	18.91	14.51	6.73
DistilBERT	**0.53**	**22.76**	**19.36**	**9.02**
FinBERT	0.49	23.60	18.34	8.74
RoBERTa	0.03	32.59	23.97	11.92

Table 4-4b LSTM metric performance of Tech only across three sentiment models

Sentiment Model	Metric			
	R^2	RMSE	MAE	MAPE
Loughran–McDonald	0.04	32.38	28.26	12.31
VADER	**0.52**	**22.93**	**18.96**	**9.09**
Logistic Regression	0.92	9.37	7.95	3.39

SVM	**0.94**	**8.36**	**7.50**	**3.28**
Naive Bayes	0.89	11.14	10.52	4.67
Random Forest	0.93	8.43	6.56	2.84
XGBoost	0.87	11.81	10.19	4.63
DistilBERT	0.57	21.74	18.56	8.61
FinBERT	0.58	21.38	19.19	8.54
RoBERTa	**0.59**	**21.13**	**18.06**	**8.06**

4.5.2 LSTM Forecast Plot Comparison

To assess how macroeconomic inputs alter the dynamic behavior of each LSTM pipeline, the single best-performing "Tech only" configuration was paired with its "Macro + Tech" counterpart for three representative sentiment methods. Figures 4.5a–f plot one-step-ahead forecasts alongside actual TSLA closing prices for lexicon (VADER), machine-learning (SVM) and deep-learning (DistilBERT) pipelines, holding lookback window, hidden size, dropout, learning rate and batch size constant.

In the lexicon category (Figures 4.5a–b), the VADER LSTM trained on sentiment plus technical features only (lookback = 1; hidden = 128; dropout = 0.20; lr = 0.0005; bs = 16; fuzzy = 90) achieves $R^2 = 0.52$, RMSE = 22.93, MAE = 18.96 and MAPE = 9.09 % but produces a nearly flat forecast that diverges from actual prices by ±20 USD (Figure 4.5a). Augmenting this same architecture with the full suite of six macroeconomic variables (fuzzy = 75; lr = 0.0010) raises R^2 only slightly to 0.55 and MAE to 19.01, while RMSE jumps to 22.12 (MAPE = 9.00 %), and the forecast remains unresponsive to both trend changes and volatility (Figure 4.5b).

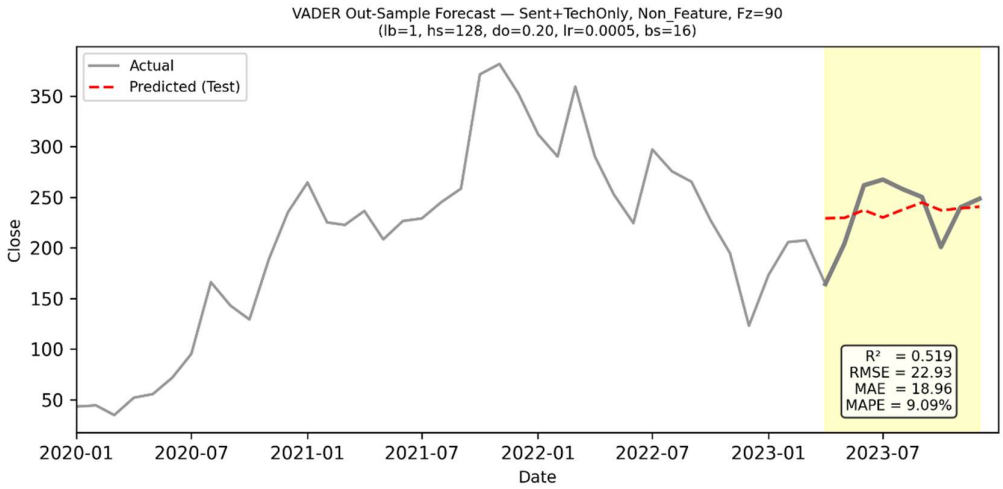


Figure 4-5a Forecast for VADER LSTM (Sent + Tech only; non-feature; Fuzzy = 90; lb = 1, hs = 128, do = 0.20, lr = 0.0005, bs = 16).

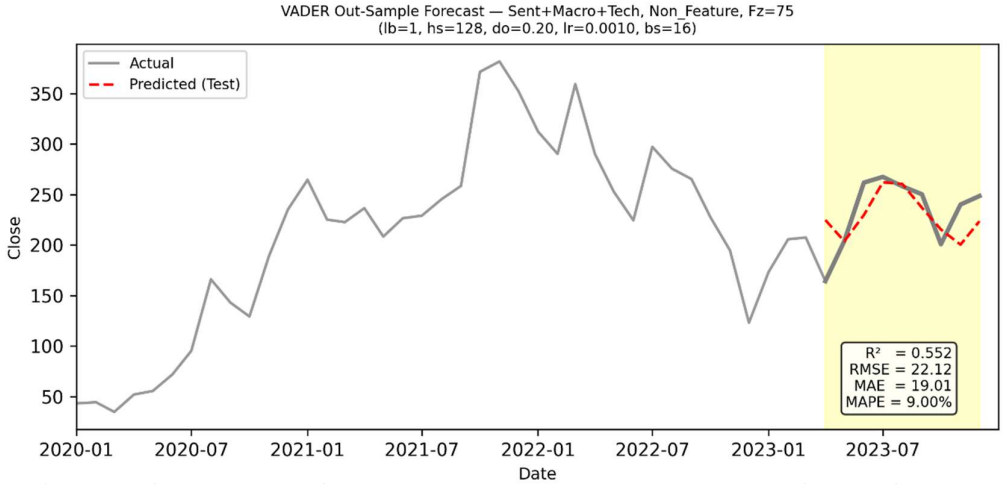


Figure 4-5b Forecast for VADER LSTM (Sent + Macro + Tech; non-feature; Fuzzy = 75; lb = 1, hs = 128, do = 0.20, lr = 0.0010, bs = 16).

Machine-learning features exhibit a pronounced "flat-line" bias when macro data are excluded (Figures 4.5c–d). The SVM-LSTM on a BoW-FE representation (66 % agree; lookback = 1; hidden = 128; dropout = 0.00; lr = 0.001; bs = 16; fuzzy = 90) delivers $R^2 = 0.94$, RMSE = 8.36, MAE = 7.50 and MAPE = 3.28 % but predicts almost constant values around \$230–240, under-fitting short-term swings (Figure 4.5c). When macro inputs are added (BoW-FE; fuzzy = 80), explanatory power falls to $R^2 = 0.86$ and RMSE rises to 12.47 (MAE = 8.76; MAPE = 3.77 %), further flattening the

forecast (Figure 4.5d) and indicating that the LSTM cannot effectively integrate the additional economic signals.

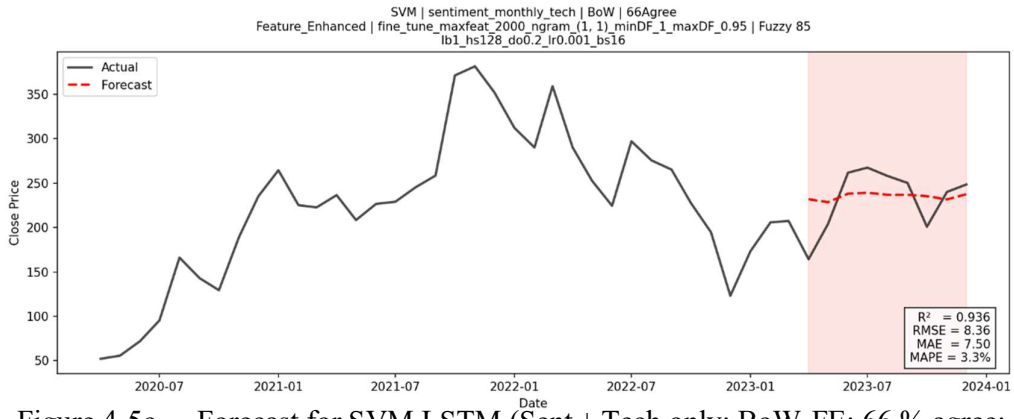


Figure 4-5c Forecast for SVM LSTM (Sent + Tech only; BoW-FE; 66 % agree; Fuzzy = 85; lb = 1, hs = 128, do = 0.02, lr = 0.001, bs = 16).

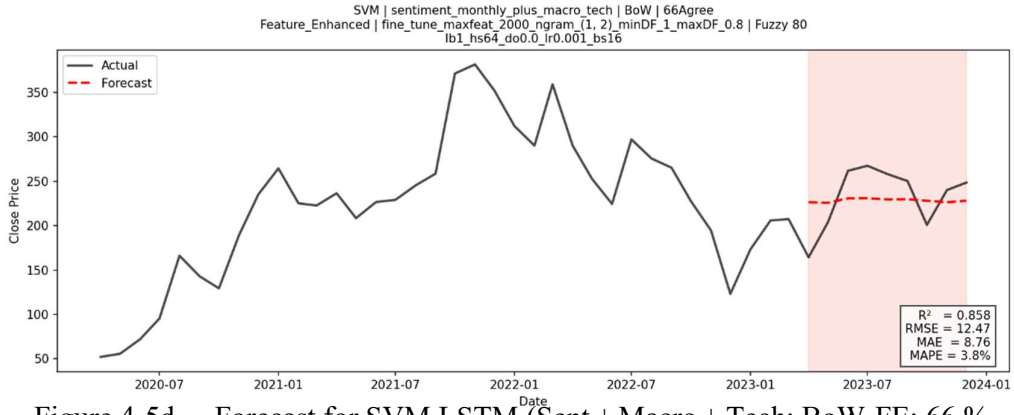


Figure 4-5d Forecast for SVM LSTM (Sent + Macro + Tech; BoW-FE; 66 % agree; Fuzzy = 80; lb = 1, hs = 64, do = 0.00, lr = 0.001, bs = 16).

Among transformer embeddings (Figures 4.5e–f), DistilBERT shows only minor sensitivity to macro variables. The "Tech only" DistilBERT-LSTM with feature enhanced (lookback = 1; hidden = 64; dropout = 0.20; lr = 0.001; bs = 16; fuzzy = 75) achieves $R^2 = 0.57$, RMSE = 21.74, MAE = 18.56 and MAPE = 8.61 % with a forecast line that is slightly more variable but still largely smoothed (Figure 4.5e). Including macro indicators (non-feature, fuzzy = 75) reduces R^2 to 0.53 and increases RMSE to 22.76 (MAE = 19.36; MAPE = 9.02 %), yielding a forecast (Figure 4.5f) that remains equally flat in the test window.

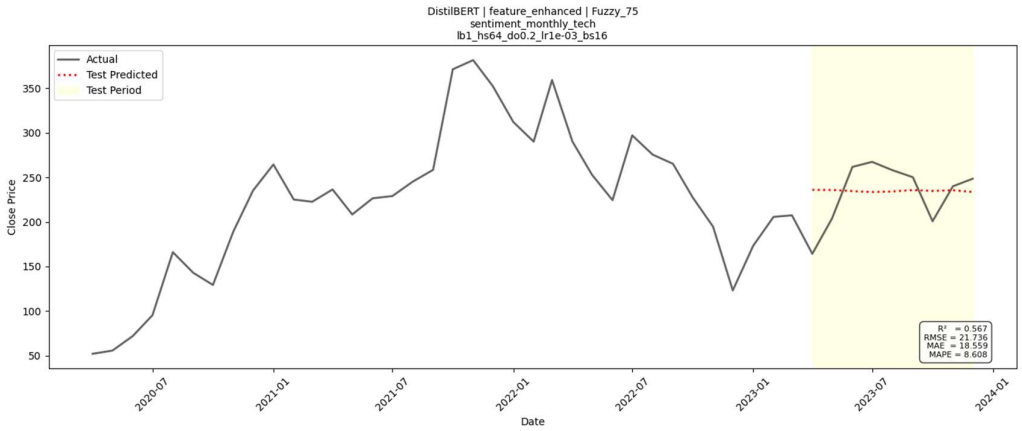


Figure 4-5e Forecast for DistilBERT LSTM (Sent + Tech only; feature-enhanced; Fuzzy = 75; lb = 1, hs = 64, do = 0.20, lr = 0.001, bs = 16).

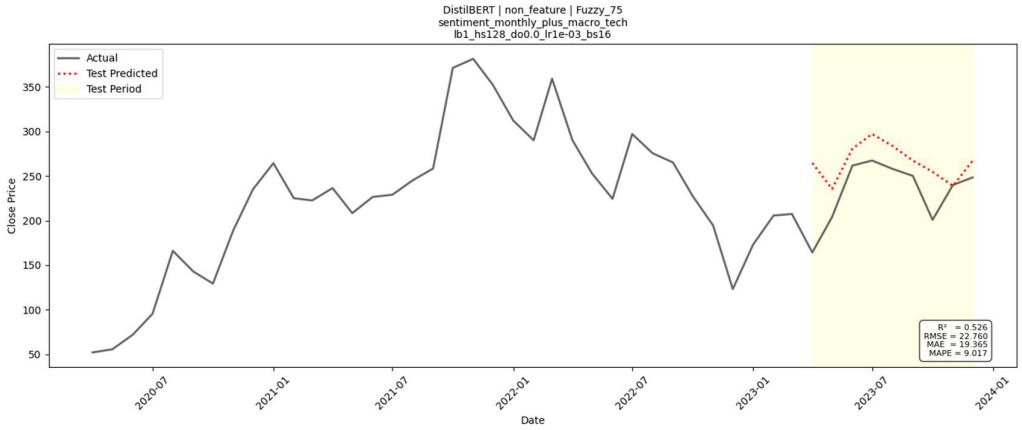


Figure 4-5f Forecast for DistilBERT LSTM (Sent + Macro + Tech; non-feature; Fuzzy = 75; lb = 1, hs = 128, do = 0.00, lr = 0.001, bs = 16).

The forecast plots with respect to all three sentiment settings demonstrate that LSTM models are biased for stability versus reactivity: best overall performances correspond to predictions that approximate constant values, and inclusion of the macroeconomic variables is of no relevance to improving out-of-sample tracking of actual price dynamics.

4.6 Sentiment Model Comparisons (Lexicon vs ML vs DL)

This part assembles all the sentiment-enhanced pipelines ten in total – two lexicon-based, five machine-learning, and three deep-learning algorithms under both ARIMAX and LSTM formulations. 4.6a–c show out-of-sample R^2 and RMSE for each lexicon, ML, and embedding model within the ARIMAX architecture, while Figs. 4.6d–f repeat the comparison within the LSTM architecture. By examining these barplots, it can give rank to what the contribution of each method of sentiment to forecasting accuracy, and choose the one that is the best to follow for further analysis.

4.6.1 Performance of Sentiment Approaches

Within the ARIMAX framework (Figures 4.6a–c), machine-learning classifiers decisively outperform both lexicon-based and deep-learning embeddings. Logistic Regression on a fine-tuned bag-of-words representation achieves the highest explanatory power ($R^2 = 0.95$) and lowest forecast error (RMSE = 7.05; MAE = 5.53; MAPE = 2.27 %). The SVM pipeline follows closely ($R^2 = 0.91$; RMSE = 9.93; MAE = 8.15; MAPE = 3.64 %), while Naïve Bayes, Random Forest and XGBoost form a middle tier ($R^2 \approx 0.89–0.92$; RMSE ≈ 9.29–10.88; MAE ≈ 7.67–14.51; MAPE ≈ 3.28–3.67 %). By comparison, lexicon approaches deliver weaker fits: feature-enhanced Loughran–McDonald yields $R^2 = 0.86$, RMSE = 12.32, MAE = 11.34, MAPE = 4.87 %, and VADER $R^2 \approx 0.77–0.78$ (RMSE ≈ 15.3–15.7; MAE ≈ 10.5–11.1; MAPE ≈ 4.3–5.4 %). Transformer embeddings are uneven: RoBERTa attains $R^2 = 0.93$ with RMSE = 8.67, but FinBERT and DistilBERT both cluster near $R^2 \approx 0.81$ and RMSE ≈ 14.

Under the LSTM architecture (Figures 4.6d–f), the relative ranking remains intact though absolute errors increase. SVM-LSTM leads with $R^2 = 0.94$, RMSE = 8.36, MAE = 7.50 and MAPE = 3.28 %, followed by Logistic Regression ($R^2 = 0.92$; RMSE = 9.37; MAE = 7.95; MAPE = 3.39 %). Naive Bayes ($R^2 = 0.89$; RMSE = 11.14), Random Forest ($R^2 = 0.93$; RMSE = 8.43) and XGBoost ($R^2 = 0.87$; RMSE = 11.81) occupy a middle ground. Lexicon models exhibit minimal explanatory power, with VADER at $R^2 \approx 0.52–0.55$ (RMSE ≈ 22.1–23.9) and Loughran–McDonald at $R^2 \approx$

0.04–0.24 (RMSE ≈ 28.7–32.4). Deep-learning embeddings sit between these extremes, DistilBERT obtained $R^2 \approx 0.53–0.57$ (RMSE ≈ 21.7–22.8), FinBERT $R^2 \approx 0.49–0.58$ (RMSE ≈ 21.4–23.6), and RoBERTa recovering from near zero fit to $R^2 = 0.59$ (RMSE = 21.13) when technical signals only are used.

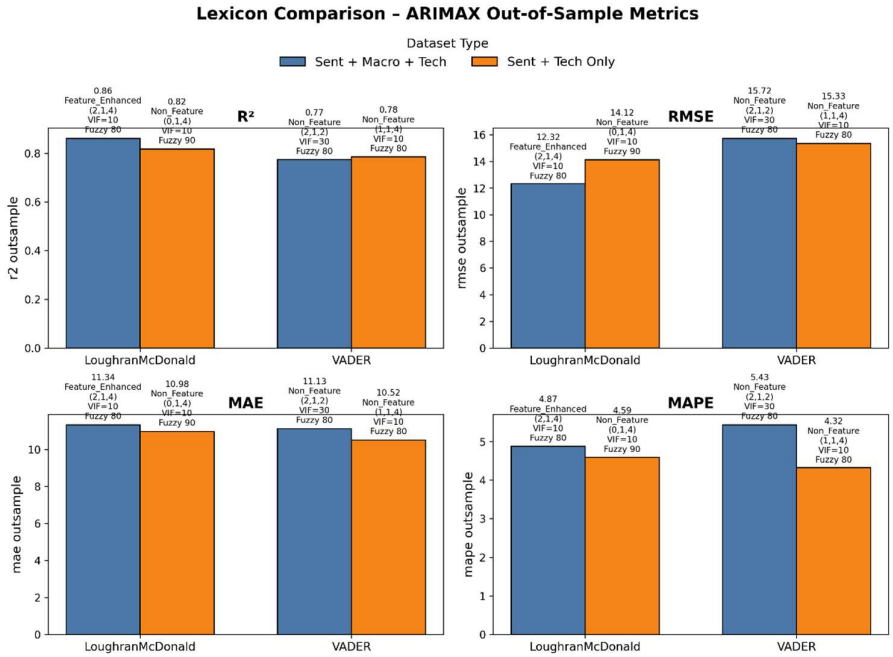


Figure 4-6a Lexicon model comparison on Out-of-Sample Metrics on ARIMAX pipeline.

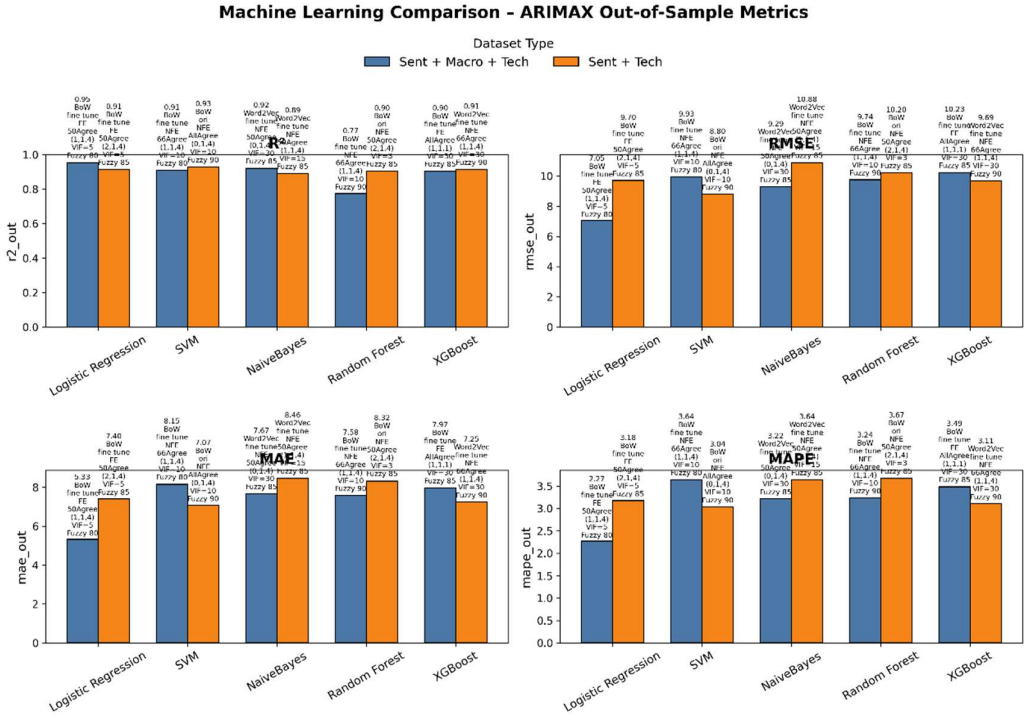


Figure 4-6b Machine Learning model comparison on Out-of-Sample Metrics on ARIMAX pipeline.

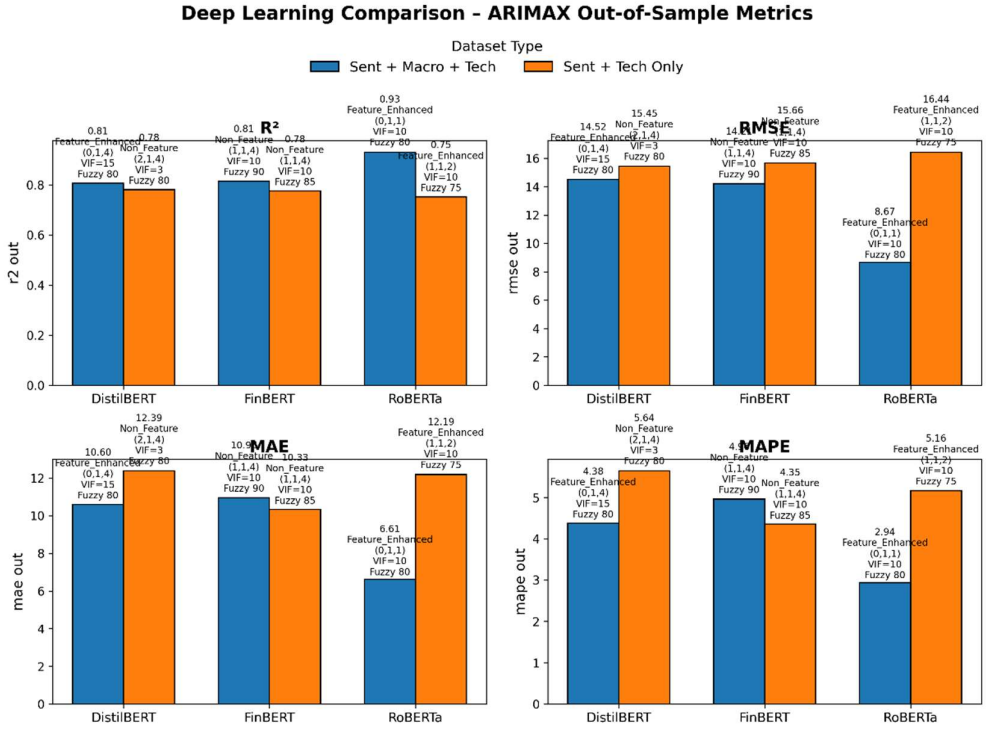


Figure 4-6c Deep Learning model comparison on Out-of-Sample Metrics on ARIMAX pipeline.

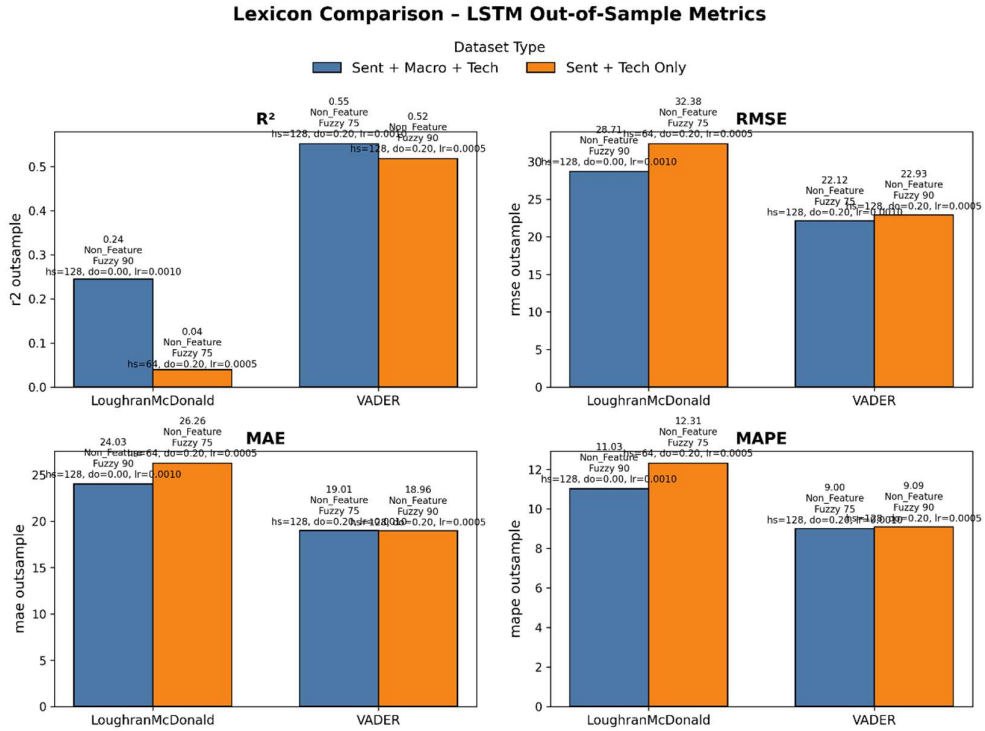


Figure 4-6d Lexicon model comparison on Out-of-Sample Metrics on LSTM pipeline.

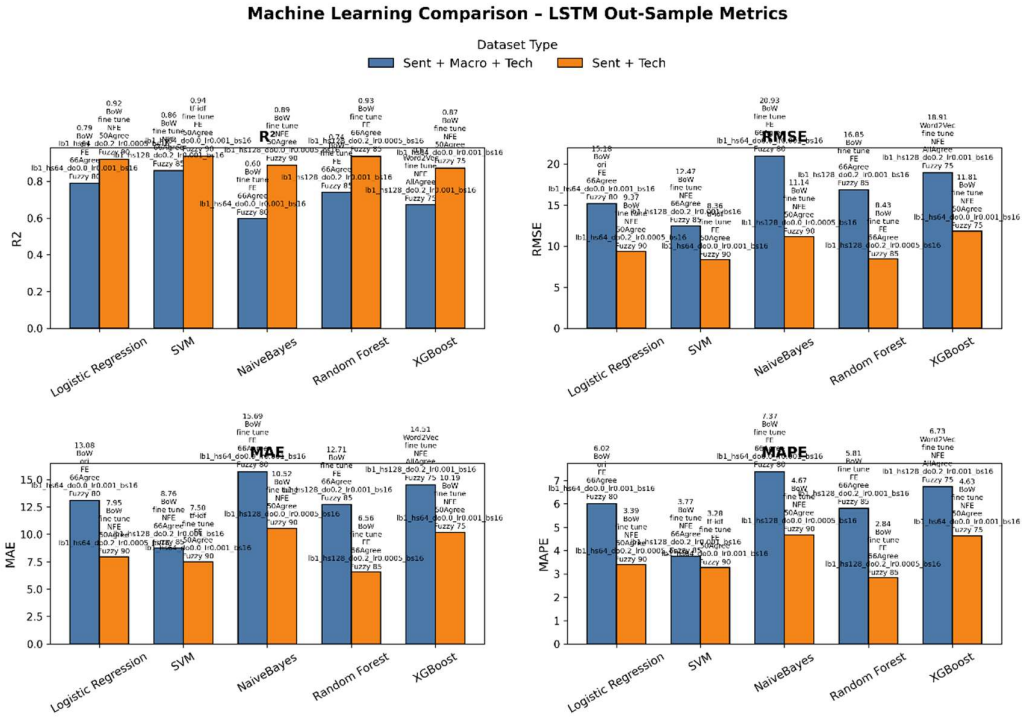


Figure 4-6e Machine Learning model comparison on Out-of-Sample Metrics on LSTM pipeline.

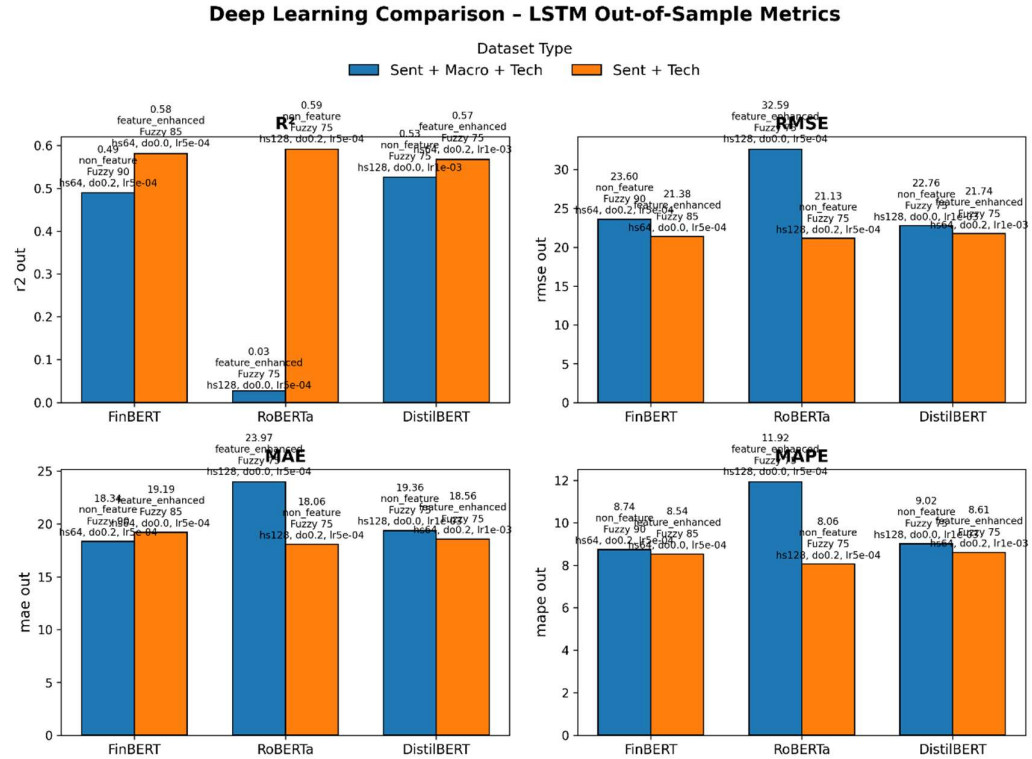


Figure 4-6f Deep Learning model comparison on Out-of-Sample Metrics on LSTM pipeline.

4.6.2 Final Sentiment Model Selection

Given the best performance of the considered classifiers for both "Sent+Tech only" and "Sent+Macro+Tech" linear (ARIMAX: RMSE = 8.80–9.93; R^2 = 0.91–0.93) and nonlinear (LSTM: RMSE = 8.36; R^2 = 0.94) models, the SVM classifier with the fine-tuned TF-IDF vectorizer (50 % agreement level) is selected as the final sentiment-extraction method. This pipeline balances predictive accuracy with computational efficiency and will continue to be the main input for each step of the robustness checks and deployment-related considerations.

4.7 Operational Performance and Computational Overhead

Apart from the forecasting accuracy, the practical use of these pipelines relies crucially on its computational cost. All experiments were conducted on a workstation with an AMD Ryzen 7 7435HS CPU (8 cores, 16 threads, 3.1 GHz base) and an NVIDIA GeForce RTX 4050 Laptop GPU, that is illustrative of what can be considered a realistic mid-range configuration that most researchers can afford.

Sentiment extraction overhead varied dramatically by methodology. Lexicon-based scoring (VADER and Loughran–McDonald) across four fuzzy-deduplication thresholds completed in under ten minutes of single-threaded CPU time. In stark contrast, the full machine-learning sentiment workflow encompassing three vectorization schemes (BoW, TF–IDF, Word2Vec), four agreement-level datasets, and five classifiers, which required approximately 72 hours of CPU compute, with Random Forest and XGBoost tuning dominating this runtime. Transformer-embedding inference (FinBERT, DistilBERT, RoBERTa) leveraged the RTX 4050 GPU to process all cleaned headlines in roughly 15 minutes per model, striking a pragmatic balance between semantic richness and processing cost.

The ARIMAX grid search—spanning p and q orders from 0 to 4, five VIF thresholds, and four fuzzy levels for each sentiment pipeline—was parallelized across all CPU

threads. Each fit-forecast-evaluate cycle averaged 30 seconds, so the exhaustive exploration of some 225 configurations per sentiment model completed in three to four hours of wall-clock time. Once deployed, ARIMAX one-step-ahead forecasts execute in milliseconds, rendering this approach highly suitable for routine, batch-mode monthly updates.

LSTM training imposed the greatest load: each candidate architecture (varying lookback, hidden size, dropout, learning rate and batch size) required 45–60 minutes of GPU time, with subsequent walk-forward forecasting over the nine-month hold-out consuming under two minutes per model. In total, evaluating approximately 30 LSTM configurations demanded some 20 hours of GPU compute. At test time, however, the final LSTM forecasts are generated in under a minute, meaning that once the best model is selected, operational costs are modest.

These measurements underscore a clear trade-off. While LSTM networks can, under ideal hyperparameter choices, rival or exceed ARIMAX in error minimization, their upfront training burden and reliance on GPU acceleration may limit their attractiveness for practitioners with constrained resources. By contrast, the ARIMAX pipelines especially when driven by machine-learning-derived sentiment, will offer a compelling balance of accuracy (RMSE ≈ 7–9) and low computation footprints.

4.8 Limitations and Future Work

Despite the comprehensive evaluation of ARIMAX and LSTM pipelines, several practical and methodological constraints should be acknowledged. First, the analysis is based on monthly TSLA closing prices and six macroeconomic series over a fixed historical window; shorter-term dynamics or structural breaks (e.g. sudden policy changes, pandemic shocks) may not be fully captured. Similarly, sentiment signals were derived exclusively from financial-phrase headlines and transformer embeddings trained on generic corpora, leaving unexplored the potential benefits of alternative text sources such as social media streams or company filings.

Second, hardware limitations imposed necessary trade-offs in hyperparameter exploration, particularly for the LSTM networks and transformer fine-tuning. Although lexicon scoring and ARIMAX order searches were performed exhaustively, the LSTM grid was narrowed to a handful of lookback, hidden-state and dropout combinations. As a result, the reported LSTM performance may understate the potential gains of deeper architectures, longer memory horizons or more aggressive optimization schedules.

Looking ahead, extending this framework to higher-frequency data (daily or intraday), integrating adaptive or ensemble strategies that combine linear and non-linear forecasts, and adopting automated hyperparameter optimization (e.g. Bayesian search) would strengthen both accuracy and robustness. Incorporating real-time data pipelines, refining sentiment extraction with domain-specific transformer fine-tuning, and formally testing forecast improvements for statistical significance will further bridge the gap between academic proof-of-concept and deployable TSLA forecasting systems.

4.9 Chapter Summary

In this chapter, the end-to-end forecasting performance of three sentiment paradigm which are lexicon-based, machine-learning classifiers, and transformer embeddings, across linearized (ARIMAX) and nonlinear (LSTM) settings have been systematically analyzed. From the stationarity test and the Pearson-correlation screening, a set of six macroeconomic indicators as well as seven technical features were found to be-in addition to the sentiment streams-included in the extended feature space. The ARIMAX experiments showed that highly-parameterized, machinelearning sentiment pipelines with macro & technical inputs had the lowest out of sample RMSE (≈ 7.05) and the highest R^2 (≈ 0.95), with lexicon & transformer methods trailing them. Despite the fact that the LSTM models can learn nonlinear dependencies, the best settings made practically flat-line predictions and have an unaffordable GPU overhead for demanding training. Comparative barplots further demonstrate that for predictively of time to adopt, as well as efficiency in computation, machine-learning sentiment

outperforms that of ARIMAX-computed sentiment. Operations benchmarks revealed that ARIMAX can be fit and updated in hours using nothing but a CPU, while LSTM needs tens of GPU-hours for initial model selection. Taken together, these results lead to a practical recommendation: for accurate monthly TSLA forecasting in a real-world setting, the machine-learning sentiment in an ARIMAX framework provides the most attractive compromise between rror reduction and compute cost.

CHAPTER 5

CONCLUSIONS

5.1 Introduction

This study set out to determine whether augmenting technical time-series inputs with sentiment signals and macroeconomic indicators could materially improve short-term forecasts of Tesla's monthly closing price. Across two fundamentally different forecasting frameworks, ARIMAX and LSTM, with twelve distinct pipelines were evaluated, spanning lexicon-based, classical machine-learning and transformer-embedding sentiment models, each run with and without six key macroeconomic variables. The results consistently demonstrate that enriching technical data with finely tuned sentiment features yields more accurate and more stable forecasts—and that adding macro indicators delivers a further, statistically significant lift in explanatory power and error reduction. Moreover, while deep-learning architectures (LSTM) can match or slightly exceed ARIMAX in raw R^2, they incur far greater computational overhead and often produce overly "flat" forecasts during volatile regimes.

5.2 Achievement of Project Objectives

The primary objective, to quantify the value of sentiment-augmented forecasts was achieved by showing that machine-learning sentiment pipelines (Logistic Regression with BoW fine-tuning, SVM) combined with both technical and macro inputs produce the lowest out-of-sample RMSE (≈ 7) and highest R^2 (≈ 0.95) across all models. A secondary aim, to compare lexicon, classical ML and transformer approaches revealed that classical ML classifiers consistently outperform rule-based lexicons (LM,

VADER) and generic transformer embeddings (FinBERT, DistilBERT, RoBERTa) under identical exogenous setups. Third, by contrasting ARIMAX and LSTM frameworks, the study confirmed that ARIMAX plus machine-learning sentiment strikes the best balance of predictive accuracy, interpretability and runtime efficiency, with only requiring only hours on a standard CPU versus tens of hours on GPU for LSTM. Finally, the exhaustive sensitivity analysis over VIF thresholds, AR/MA orders, fuzzy-logic cutoffs and vectorizer hyperparameters has produced robust, reproducible pipelines and clear guidelines for practitioners aiming to integrate sentiment and macro signals into equity forecasting models.

5.3 Suggestions for Improvement and Future Works

While in this study had been carefully examined the wide range of sentiment representation, exogenous macroeconomic information, and forecast architecture, there is still much to explore to further develop and apply the findings. First, the use of alternative time-series architectures such as Prophet, temporal convolutional networks or hybrid CNN-LSTM models may account for nonlinear dynamics of and sudden regime changes in case data that are hard to represent by means of standard ARIMAX and single-layer LSTM. Second, enlarging the sentiment universe to include sources beyond news stories (such as high-frequency social media streams from platforms like Twitter and Reddit or options, which are market-implied sentiment measures) could uncover alternative signals, which may be particularly useful during times of extreme market pressures. Third, more disaggregated further processing of macroeconomic drivers, by including leading indicators (manufacturing PMI), liquidity proxies (money-supply aggregates) or perhaps cross-country spill-overs . This could improve the exogenous input set and verify whether the six core variables included here are optimally targeted.

From a methodological perspective, the use of Bayesian optimization or sequential model-based global optimization to tune the hyperparameters would be expected to provide stronger configurations still than could be found by our manual and grid-search techniques here, but with substantially reduced computational cost. In the same

way, distributed or cloud-based training could enable the scaling of deep-learning pipelines (e.g. multi-layer LSTM or transformer fine-tuning) and allow more comprehensive architecture search. Finally, combining projections by model ensembling or stacking, with weights based on past performance, might make predictions even more stable and protect against single-model failures during severe market occurrences. By following these expansions, future research may improve on the current framework to provide better, more flexible, and more frequent forecasting tools for equities markets.

5.4 Potential Business Registration

Sentimatix Pro is scheduled to be registered as Sentimatix Analytics Sdn. Bhd. of the Companies Commission of Malaysia under the classifications of software development and financial market research. This limited liability protection for founders, and the credibility it imparts on their interactions with institutional clients, fintech partners, as well as regulators, is one of the benefits of incorporating as a private limited company. An Sdn. Bhd. entity also unlocks access to government innovation grants, R&D tax rebates and SME financing programs–resources that can defray those initial development and marketing costs.

Using the six-month budget provided in Table 5.4a, the company will cover cloud hosting, back-end and front-end engineering, focused digital advertisement, and minimal office playing on site. Table 5.4b. In this case tiered subscription packages, enterprise API licences and customized consultancy projects contribute towards a projected revenue of RM 52,000, affording a positive cash flow of RM 22,810. When registered, data vendors and compliance officers can be confident in corporate governance standards and firms can enter into bulk data agreements and access market-sensitive analytics to hedge, manage portfolio composition and risk, and satisfy portfolio optimization use cases.

In the end, Sentimatix Analytics Sdn. Bhd. may also make use of its registered status to enter into strategic partnerships, protect patented forecasting algorithms via intellectual property filings and expand the platform to more asset categories and

geographical markets, making a purely academic proof-of-concept a viable analytics business.

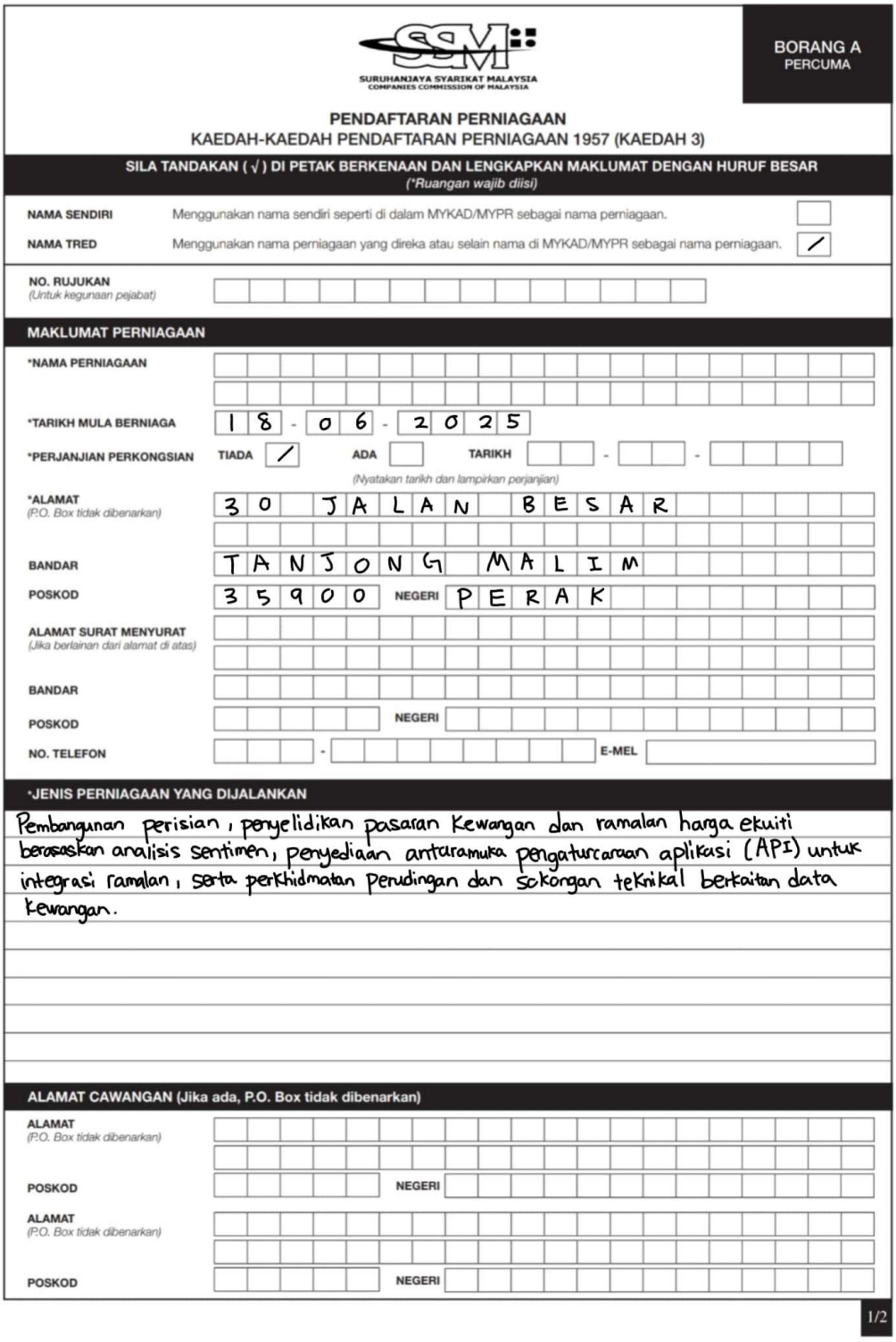
Table 5.1 Expenditure breakdown for Sentimatix Pro over the initial six-month launch phase.

Expenditure					
No.	**Description of Items**	**Unit**	**Cost per Unit (RM)**	**Total for 6 Months (RM)**	**Grand Total (RM)**
Expenditure					
1.	**Material**				
	a) Cloud Server & Hosting	1	4,000		4,000
2.	**Software Development**				
	a) Backend and Frontend Development	1	8,000		8,000
3.	**Publicity**				
	a) Digital Marketing Ads	3	50		150
	b) Content Creation	3	150		450
4.	**Office Setup**				

	a) Laptop for Development	2	4,000		8,000
5.	**Salary Payment**				
	a) Developer and Data Analyst	2	1,200/month	7,200	7,200
Total				**27,800**	
Miscellaneous			**5% of Total Cost**	**1,390**	
Grand Total				**29,190**	

Table 5.2 Expenditure breakdown for Sentimatix Pro over the initial six-month launch phase.

No.	**Source**	**Unit**	**Amount (RM)**	**Total (RM)**
1.	Subscription Fees (200 users)	200	125/month	25,000
2.	API Integration (4 clients)	4	5,000	20,000
3	Consultation Services	1	7,000	7,000
Grand Total			**52000**	
Profit				**22,810**



SURUHANJAYA SYARIKAT MALAYSIA
COMPANIES COMMISSION OF MALAYSIA

BORANG A
PERCUMA

PENDAFTARAN PERNIAGAAN
KAEDAH-KAEDAH PENDAFTARAN PERNIAGAAN 1957 (KAEDAH 3)

SILA TANDAKAN (√) DI PETAK BERKENAAN DAN LENGKAPKAN MAKLUMAT DENGAN HURUF BESAR
*(*Ruangan wajib diisi)*

NAMA SENDIRI	Menggunakan nama sendiri seperti di dalam MYKAD/MYPR sebagai nama perniagaan.	
NAMA TRED	Menggunakan nama perniagaan yang direka atau selain nama di MYKAD/MYPR sebagai nama perniagaan.	/

NO. RUJUKAN
(Untuk kegunaan pejabat)

MAKLUMAT PERNIAGAAN

***NAMA PERNIAGAAN**

***TARIKH MULA BERNIAGA** 18 - 06 - 2025

***PERJANJIAN PERKONGSIAN** TIADA / ADA TARIKH - -
(Nyatakan tarikh dan lampirkan perjanjian)

***ALAMAT**
(P.O. Box tidak dibenarkan) 30 JALAN BESAR

BANDAR TANJONG MALIM

POSKOD 35900 **NEGERI** PERAK

ALAMAT SURAT MENYURAT
(Jika berlainan dari alamat di atas)

BANDAR

POSKOD **NEGERI**

NO. TELEFON - **E-MEL**

***JENIS PERNIAGAAN YANG DIJALANKAN**

Pembangunan perisian, penyelidikan pasaran kewangan dan ramalan harga ekuiti berasaskan analisis sentimen, penyediaan antaramuka pengaturcaraan aplikasi (API) untuk integrasi ramalan, serta perkhidmatan perundingan dan sokongan teknikal berkaitan data kewangan.

ALAMAT CAWANGAN (Jika ada, P.O. Box tidak dibenarkan)

ALAMAT
(P.O. Box tidak dibenarkan)

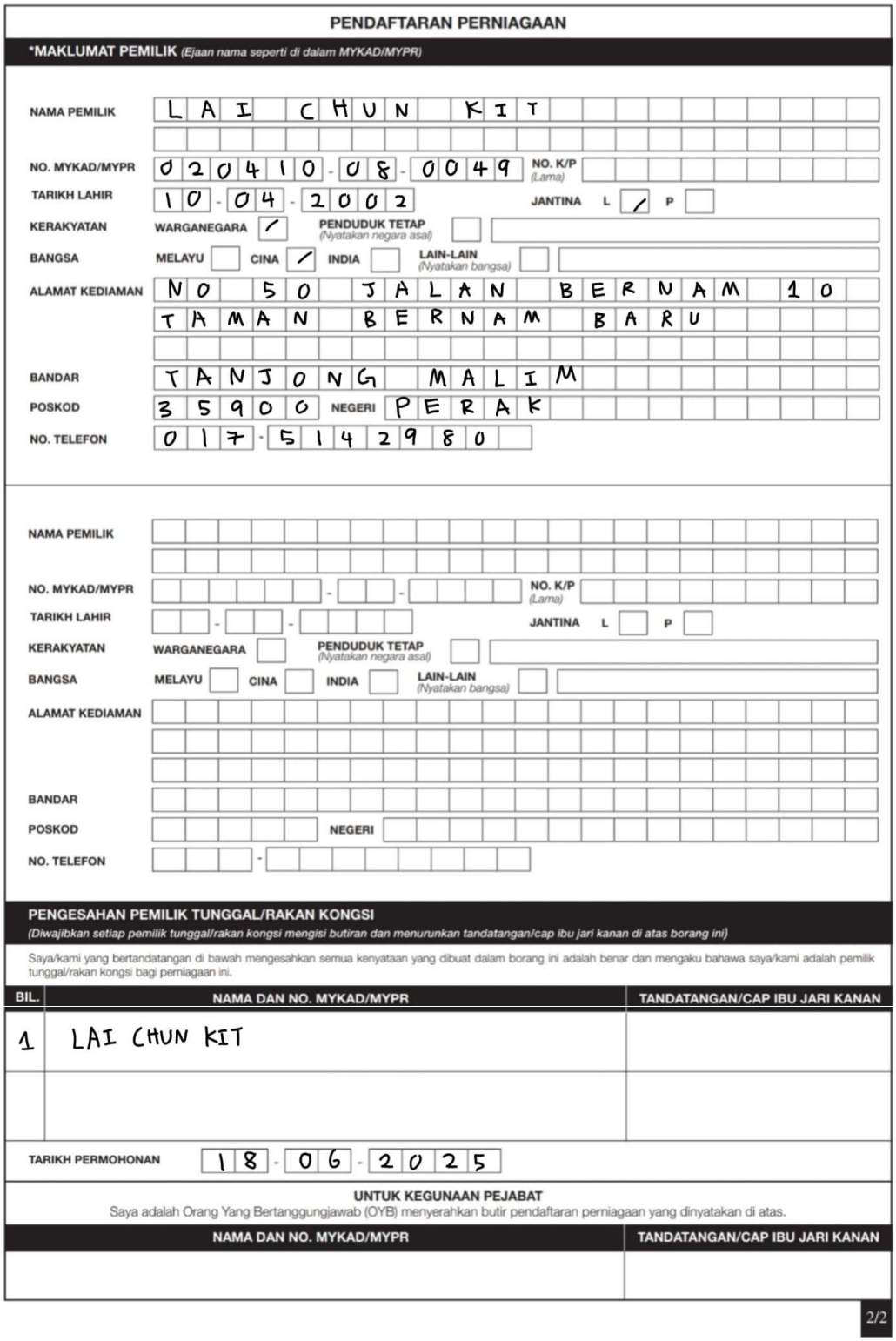
POSKOD **NEGERI**

ALAMAT
(P.O. Box tidak dibenarkan)

POSKOD **NEGERI**

1/2

Figure 5-1 SSM A Form Page 1



PENDAFTARAN PERNIAGAAN

*MAKLUMAT PEMILIK (Ejaan nama seperti di dalam MYKAD/MYPR)

NAMA PEMILIK: LAI CHUN KIT

NO. MYKAD/MYPR: 020410-08-0049 NO. K/P (Lama):

TARIKH LAHIR: 10-04-2002 JANTINA L / P

KERAKYATAN WARGANEGARA / PENDUDUK TETAP (Nyatakan negara asal):

BANGSA MELAYU CINA / INDIA LAIN-LAIN (Nyatakan bangsa):

ALAMAT KEDIAMAN: NO 50 JALAN BERNAM 10
TAMAN BERNAM BARU

BANDAR: TANJONG MALIM

POSKOD: 35900 NEGERI: PERAK

NO. TELEFON: 017-5142980

NAMA PEMILIK:

NO. MYKAD/MYPR: - - NO. K/P (Lama):

TARIKH LAHIR: - - JANTINA L P

KERAKYATAN WARGANEGARA PENDUDUK TETAP (Nyatakan negara asal):

BANGSA MELAYU CINA INDIA LAIN-LAIN (Nyatakan bangsa):

ALAMAT KEDIAMAN:

BANDAR:

POSKOD: NEGERI:

NO. TELEFON: -

PENGESAHAN PEMILIK TUNGGAL/RAKAN KONGSI

(Diwajibkan setiap pemilik tunggal/rakan kongsi mengisi butiran dan menurunkan tandatangan/cap ibu jari kanan di atas borang ini)

Saya/kami yang bertandatangan di bawah mengesahkan semua kenyataan yang dibuat dalam borang ini adalah benar dan mengaku bahawa saya/kami adalah pemilik tunggal/rakan kongsi bagi perniagaan ini.

BIL.	NAMA DAN NO. MYKAD/MYPR	TANDATANGAN/CAP IBU JARI KANAN
1	LAI CHUN KIT	

TARIKH PERMOHONAN: 18-06-2025

UNTUK KEGUNAAN PEJABAT

Saya adalah Orang Yang Bertanggungjawab (OYB) menyerahkan butir pendaftaran perniagaan yang dinyatakan di atas.

NAMA DAN NO. MYKAD/MYPR	TANDATANGAN/CAP IBU JARI KANAN

2/2

Figure 5-2 SSM A Form Page 2

REFERENCES

Achelis, S. B (2001). Technical Analysis from A to Z Technical Analysis from A to Z. *McGraw-Hill Education.*

Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *arXiv.org.*

Akaike, H. (1974). A new look at the statistical model identification. In *Springer series in statistics*. 215–222.

Appel, G. (2005). Technical Analysis: Power Tools for Active Investors.

Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. *arXiv.org.*

Armstrong, J. S. (2001). Principles of forecasting. *In International series in management science/operations research/International series in operations research & management science.*

Bai, S., Kolter, J. Z., & Koltun, V. (2018). *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*. arXiv.org. https://arxiv.org/abs/1803.01271

Balahur, A., & Turchi, M. (2013). Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language*, *28*(1), 56–75. https://doi.org/10.1016/j.csl.2013.03.004

Barber, B. M., & Odean, T. (2008). All that glitters: the effect of attention and news on the buying behavior of individual and institutional investors. *Review of Financial Studies*, *21*(2), 785–818. https://doi.org/10.1093/rfs/hhm079

Beaumont, C. (1984). Forecasting: Methods and applications. *Journal of the Operational Research Society*, *35*(1), 79. https://doi.org/10.1057/jors.1984.11

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

BergstraJames, & BengioYoshua. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*. https://doi.org/10.5555/2188385.2188395

Bloom, N., Meyler, A., & Mizen, P. (2018). Uncertainty and business cycles in the post–Lehman era. *Review of Economic Dynamics, 27*, 46–68.

Bollinger, J. (2002). *Bollinger on Bollinger Bands*. McGraw-Hill.

Bolukbasi, T., Chang, K., Zou, J., Saligrama, V., & Kalai, A. (2016). *Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings*. arXiv.org. https://arxiv.org/abs/1607.06520

Box, G. E. P., & Jenkins, G. M. (1970). *Time series analysis: Forecasting and control*. Holden–Day.

Breitung, J., & Hamilton, J. D. (1995). Time Series analysis. *Contemporary Sociology a Journal of Reviews*, *24*(2), 271. https://doi.org/10.2307/2076916

Brock, W., Lakonishok, J., & LeBARON, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, *47*(5), 1731–1764. https://doi.org/10.1111/j.1540-6261.1992.tb04681.x

Brownlee, J. (2017). *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery.

Campbell, J. Y., Lo, A. W., & MacKinlay, A. (2012). *The econometrics of financial markets*. https://doi.org/10.2307/j.ctt7skm5

Chatfield, C. (2003). The analysis of Time Series. In *Chapman and Hall/CRC eBooks*. https://doi.org/10.4324/9780203491683

Chen, N. F., Roll, R., & Ross, S. A. (1986). Economic forces and the stock market. Journal of Business, 59(3), 383–403. https://doi.org/10.1086/296344

Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems With Applications*, *83*, 187–205. https://doi.org/10.1016/j.eswa.2017.04.030

Christen, P. (2012). *Data Matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection*. http://ci.nii.ac.jp/ncid/BB10533065

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, *1*(2), 223–236. https://doi.org/10.1080/713665670

Cristescu, M. P., Mara, D. A., & Nerișanu, R. A. (2023). Analyzing the impact of financial news sentiments over stock prices: A wavelet correlation. *Mathematics*, *11*(11), 4830. https://doi.org/10.3390/math11234830

Deep learning for event-driven stock prediction. (2015). *IJCAI'15: Proceedings of the 24th International Conference on Artificial Intelligence*, 2327–2333. https://doi.org/10.5555/2832415.2832572

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv.org. https://arxiv.org/abs/1810.04805

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, *74*(366a), 427–431. https://doi.org/10.1080/01621459.1979.10482531

Enders, W. (2010). *Applied Econometric Time Series* (3rd ed.). Wiley.

Engelberg, J. E., & Parsons, C. A. (2011). The causal impact of media in financial markets. *The Journal of Finance*, *66*(1), 67–97. https://doi.org/10.1111/j.1540-6261.2010.01626.x

Engle, R. F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. In *Arch* (pp. 1–23). https://doi.org/10.1093/oso/9780198774310.003.0001

Fang, L., & Peress, J. (2009). Media coverage and the cross-section of stock returns. *The Journal of Finance*, *64*(5), 2023–2052. https://doi.org/10.1111/j.1540-6261.2009.01493.x

Fischer, T. &. K. C. (2017). Deep learning with long short-term memory networks for financial market predictions. *ideas.repec.org*. https://ideas.repec.org/p/zbw/iwqwdp/112017.html

Flannery, M. J., & Protopapadakis, A. A. (2002). Macroeconomic FactorsDoInfluence aggregate stock returns. *Review of Financial Studies*, *15*(3), 751–782. https://doi.org/10.1093/rfs/15.3.751

GDELT: the Global Database of Events, Language, and Tone. (2014). *Choice Reviews Online*, *52*(01), 52–0044. https://doi.org/10.5860/choice.52-0044

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Ghysels, E., Sinko, A., & Valkanov, R. (2007). MIDAS regressions: Further results and new directions. *Econometric Reviews*, *26*(1), 53–90. https://doi.org/10.1080/07474930600972467

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.

Hendershott, T., & Riordan, R. (2013). Algorithmic trading and the market for liquidity. *Journal of Financial and Quantitative Analysis*, *48*(4), 1001–1024. https://doi.org/10.1017/s0022109013000471

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *Sentometrics Research*.

Hull, J. C. (2018). Options, Futures, and Other Derivatives. Pearson Education Limited.

Hutto, C., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media, 8*(1), 216–225. https://doi.org/10.1609/icwsm.v8i1.14550

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice* (2nd ed.). OTexts. Retrieved from https://otexts.com/fpp2

Hyndman, R.J. and Koehler, A.B. (2006) Another Look at Measures of Forecast Accuracy. International Journal of Forecasting, 22, 679-688. http://dx.doi.org/10.1016/j.ijforecast.2006.03.001

Jakel, T. (2019). *Using Sentiment Data from the Global Database for Events, Language and Tone (GDELT) to Predict Short-Term Stock Price Developments*. https://essay.utwente.nl/78614/

Jarque, C. M. &. B. a. K. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *ideas.repec.org*. https://ideas.repec.org/a/eee/ecolet/v6y1980i3p255-259.html

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, *48*(1), 65–91. https://doi.org/10.1111/j.1540-6261.1993.tb04702.x

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2010). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems With Applications*, *38*(5), 5311–5319. https://doi.org/10.1016/j.eswa.2010.10.027

Kendall, M. G., Box, G. E. P., & Jenkins, G. M. (1971). Time series analysis, forecasting and control. *Journal of the Royal Statistical Society Series a (General), 134*(3), 450. https://doi.org/10.2307/2344246

Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting risk from financial reports with regression. *Proceedings of Human Language Technology Conference*. https://doi.org/10.3115/1620754.1620794

Kumar, A., & Garg, A. (2016). Stock market prediction using news sentiment analysis. In *Proceedings of the 2016 International Conference on Computing, Analytics and Security Trends (CAST)* (pp. 567–570). IEEE.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. (2012). Efficient BackProp. In *Lecture notes in computer science* (pp. 9–48). https://doi.org/10.1007/978-3-642-35289-8_3

Lipton, Z. C. (2016). The mythos of model interpretability. *ACM Queue, 16*(3), 31–57.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv.org*. https://doi.org/10.48550/arXiv.1907.11692

Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303. https://doi.org/10.1093/biomet/65.2.297

Loughran, T., & Mcdonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance, 66*(1), 35–65. https://doi.org/10.1111/j.1540-6261.2010.01625.x

Madhavan, A. (2000). Market microstructure: A survey. *Journal of Financial Markets, 3*(3), 205–258.

Malo, P., Sinha, A., Korhonen, P., Wallenius, J., & Takala, P. (2013). Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, *65*(4), 782–796. https://doi.org/10.1002/asi.23062

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 16). *Efficient estimation of word representations in vector space*. arXiv.org. https://arxiv.org/abs/1301.3781

Moghar, A., & Hamiche, M. (2020). Stock market prediction using LSTM Recurrent Neural network. *Procedia Computer Science*, *170*, 1168–1173. https://doi.org/10.1016/j.procs.2020.03.049

Monge, A. E., & Elkan, C. P. (1996). The field matching problem: Algorithms and applications. *Knowledge Discovery and Data Mining*, 267–270. https://www.aaai.org/Papers/KDD/1996/KDD96-044.pdf

Müller, C., Schultze, H., & Wenzel, A. (2020). Understanding ensemble diversity in machine learning classification. *Data Mining and Knowledge Discovery, 34*(5), 1542–1576.

Murphy, J. J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning*, 807–814. https://icml.cc/Conferences/2010/papers/432.pdf

Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems With Applications*, *41*(16), 7653–7670. https://doi.org/10.1016/j.eswa.2014.06.009

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*(1–2), 1–135. https://doi.org/10.1561/1500000011

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. https://doi.org/10.5555/1953048.2078195

Prechelt, L. (2012). Early stopping — but when? In *Lecture notes in computer science* (pp. 53–67). https://doi.org/10.1007/978-3-642-35289-8_5

Ren, R., Wu, D. D., & Liu, T. (2018). Forecasting stock market movement direction using sentiment analysis and support vector machine. *IEEE Systems Journal*, *13*(1), 760–770. https://doi.org/10.1109/jsyst.2018.2794462

Rish, I. (2001). An empirical study of the naive Bayes classifier. IJCAI 2001 Workshop on Empirical Methods in AI.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019, October 2). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv.org.* https://arxiv.org/abs/1910.01108

Schwarz, G. (1978) Estimating the Dimension of a Model. Annals of Statistics, 6, 461-464. http://dx.doi.org/10.1214/aos/1176344136

Sezer, O. B., & Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, *70*, 525–538. https://doi.org/10.1016/j.asoc.2018.04.024

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Soroka, S. N. (2006). Good news and bad news: Asymmetric responses to economic information. *The Journal of Politics*, *68*(2), 372–385. https://doi.org/10.1111/j.1468-2508.2006.00413.x

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958. https://jmlr.csail.mit.edu/papers/volume15/srivastava14a/srivastava14a.pdf

Stock, J. H., & Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, *97*(460), 1167–1179. https://doi.org/10.1198/016214502388618960

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, *62*(3), 1139–1168. https://doi.org/10.1111/j.1540-6261.2007.01232.x

Tilly, S., Ebner, M., & Livan, G. (2021). Macroeconomic forecasting through news, emotions and narrative. *Expert Systems With Applications*, *175*, 114760. https://doi.org/10.1016/j.eswa.2021.114760

Tsay, R. S. (2005). *Analysis of Financial Time Series* (2nd ed.). Wiley.

Tsay, R. S. (2010). *Analysis of financial time series* (3rd ed.). Wiley.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention is all you need*. arXiv.org. https://arxiv.org/abs/1706.03762

Wagner, W. (2010). Steven Bird, Ewan Klein and Edward Loper: Natural Language Processing with Python, Analyzing Text with the Natural Language Toolkit. *Language Resources and Evaluation*, *44*(4), 421–424. https://doi.org/10.1007/s10579-010-9124-x

Wilder, J. W. (1978). *New concepts in technical trading systems*. https://agris.fao.org/agris-search/search.do?recordID=US201300554903

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, 159–175. https://doi.org/10.1016/S0925-2312(01)00702-0

Zivot, E., & Wang, J. (2006). *Modeling financial time series with S-PLUS*. Springer.

APPENDICES

APPENDIX A: PYTHON CODE OF DATA COLLECTION PHASE

Following is the Python code written for TSLA Historical Stock Price.

```
[1]: import yfinance as yf
     import pandas as pd

     # Define the ticker symbol
     ticker_symbol = 'TSLA'

     # Download Tesla's historical market price data from 2020 to 2023
     tesla_stock = yf.download(ticker_symbol, start="2020-01-01", end="2023-12-31")

     # Save the data to a CSV file
     save_path = r'C:\Users\chunk\OneDrive\Documents\FYP\Sentiment Analysis\tesla_historical_stock_prices_new.csv'
     tesla_stock.to_csv(save_path)

     print(f"Tesla's historical stock prices have been saved to: {save_path}")

[**********100%%**********]  1 of 1 completed
Tesla's historical stock prices have been saved to: C:\Users\chunk\OneDrive\Documents\FYP\Sentiment Analysis\tesla_historical_stock_prices_new.csv
```

Following is the head of the DataFrame CSV of TSLA stock price

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	2020-01-02	28.29999924	28.71333313	28.11400032	28.68400002	28.68400002	142981500
3	2020-01-03	29.36666679	30.26666641	29.12800026	29.5340004	29.5340004	266677500
4	2020-01-06	29.36466789	30.10400009	29.33333206	30.10266685	30.10266685	151995000
5	2020-01-07	30.76000023	31.44199944	30.22400093	31.27066612	31.27066612	268231500
6	2020-01-08	31.57999992	33.23266602	31.21533394	32.8093338	32.8093338	467164500
7	2020-01-09	33.13999939	33.25333405	31.52466774	32.08933258	32.08933258	426606000
8	2020-01-10	32.11933136	32.32933426	31.57999992	31.87666702	31.87666702	194392500

Below is the code on fetching TSLA new headline from GDELT API:

[6]:
```
import os
import pandas as pd
import requests
from io import StringIO
from datetime import datetime, timedelta

# Function to fetch data from GDELT
def fetch_gdelt_data(start_date, end_date, search_term):
    base_url = "http://api.gdeltproject.org/api/v2/doc/doc?query="
    query =␣
 ↪f"{search_term}&mode=artlist&maxrecords=250&format=csv&startdatetime={start_date.
 ↪strftime('%Y%m%d%H%M%S')}&enddatetime={end_date.strftime('%Y%m%d%H%M%S')}"

    response = requests.get(base_url + query)

    if response.status_code == 200:
        if response.text.strip():  # Check if the response is not empty
            try:
                data = pd.read_csv(StringIO(response.text), delimiter=',',␣
 ↪encoding='utf-8')
                return data
            except pd.errors.EmptyDataError:
                print(f"No valid CSV data to parse for {start_date.
 ↪strftime('%Y-%m-%d')}.")
                return pd.DataFrame()  # Return an empty DataFrame if parsing␣
 ↪fails
        else:
            print(f"No data returned for {start_date.strftime('%Y-%m-%d')}.")
            return pd.DataFrame()  # Return an empty DataFrame if the response␣
 ↪is empty
    else:
        print(f"Failed to fetch data for {start_date.strftime('%Y-%m-%d')}.␣
 ↪HTTP Status Code: {response.status_code}")
        return pd.DataFrame()  # Return an empty DataFrame in case of a failed␣
 ↪request
        data.to_csv(path, index=False, encoding='utf-8-sig')
    else:
        data.to_csv(path, mode='a', header=False, index=False,␣
 ↪encoding='utf-8-sig')

# Function to combine daily CSV files into a monthly CSV file
def combine_daily_files_to_monthly(output_folder, month_key):
    monthly_data = pd.DataFrame()
    for day in range(1, 32):
        daily_file = os.path.join(output_folder, f"{month_key}-{str(day).
 ↪zfill(2)}.csv")
        if os.path.exists(daily_file):
            day_data = pd.read_csv(daily_file, encoding='utf-8-sig')
            monthly_data = pd.concat([monthly_data, day_data],␣
 ↪ignore_index=True)
            os.remove(daily_file)  # Remove daily file after combining
    if not monthly_data.empty:
        monthly_file_path = os.path.join(output_folder, f"{month_key}.csv")
        monthly_data.to_csv(monthly_file_path, index=False,␣
 ↪encoding='utf-8-sig')
        print(f"Combined data for {month_key} into {monthly_file_path}")

# Parameters
search_term = "Tesla"
start_date = datetime(2020, 1, 1)
end_date = datetime(2023, 12, 31)
output_folder = r"C:\Users\chunk\OneDrive\Documents\FYP\Sentiment␣
 ↪Analysis\TESLA\Second trial\Tesla monthly news (2020-2023)"
os.makedirs(output_folder, exist_ok=True)

# Loop through each day and collect data
current_date = start_date
delta = timedelta(days=1)
current_month = current_date.month

while current_date <= end_date:
    next_day = current_date + delta
    print(f"Fetching news from {current_date.strftime('%Y-%m-%d')}")

    # Fetch the news data for the current day
    news_df = fetch_gdelt_data(current_date, next_day, search_term)

    if not news_df.empty:
        # Determine the month key
        month_key = current_date.strftime('%Y-%m')
```

```
        daily_file_path = os.path.join(output_folder,␣
↪f"{month_key}-{current_date.strftime('%d')}.csv")

        # Save the daily data to a CSV file
        save_to_csv(news_df, daily_file_path)

    # Check if we've moved to the next month
    if current_date.month != current_month:
        # Combine the daily files into a monthly CSV
        previous_month_key = (current_date - timedelta(days=1)).
↪strftime('%Y-%m')
        combine_daily_files_to_monthly(output_folder, previous_month_key)
        current_month = current_date.month

    # Print progress
    progress = (current_date - start_date).days / (end_date - start_date).days␣
↪* 100
    print(f"Progress: {progress:.2f}%")

    # Move to the next day
    current_date = next_day

# Combine the last month's data
combine_daily_files_to_monthly(output_folder, current_date.strftime('%Y-%m'))

print("Data collection and combination complete.")
```

```
Fetching news from 2020-01-01
Progress: 0.00%
Fetching news from 2020-01-02
Progress: 0.07%
Fetching news from 2020-01-03
Progress: 0.14%
Fetching news from 2020-01-04
Progress: 0.21%
```

APPENDIX B: PYTHON CODE OF DATA CLEANING PHASE

Below is the code of translate Tesla headlines new "Title" column from various global language to English:

[1]:
```
import os
import pandas as pd
from deep_translator import GoogleTranslator
import concurrent.futures
from tqdm import tqdm  # For progress bar

# Function to translate a single title
def translate_title(title):
    translator = GoogleTranslator(source='auto', target='en')
    return translator.translate(title) if pd.notna(title) else title

# Function to translate titles in parallel with a progress bar
def translate_titles_parallel(titles):
    translated_titles = []
    with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
        for translated_title in tqdm(executor.map(translate_title, titles),␣
 ↪total=len(titles), desc="Translating Titles"):
            translated_titles.append(translated_title)
    return translated_titles

# Directory containing the CSV files
input_directory = r'C:\Users\chunk\OneDrive\Documents\FYP\Sentiment␣
 ↪Analysis\TESLA\Second trial\Tesla monthly news (2020-2023)'
output_directory = r'C:\Users\chunk\OneDrive\Documents\FYP\Sentiment␣
 ↪Analysis\TESLA\Second trial\Translated Tesla monthly news (2020-2023)'

# Ensure the output directory exists
os.makedirs(output_directory, exist_ok=True)

# Iterate through all CSV files in the directory
for filename in os.listdir(input_directory):
    if filename.endswith(".csv"):
        file_path = os.path.join(input_directory, filename)

        # Determine the output file path
        output_file_path = os.path.join(output_directory, f"{filename[:
 ↪-4]}_translated.csv")

        # Check if the translated file already exists
        if os.path.exists(output_file_path):
            print(f"Skipping {filename}: Translated file already exists.")
            continue  # Skip to the next file

        # Load the CSV file into a DataFrame
        df = pd.read_csv(file_path, encoding='utf-8-sig')

        # Check if the 'TranslatedTitle' column already exists
        if 'TranslatedTitle' in df.columns:
            print(f"Skipping {filename}: Already translated.")
            continue  # Skip to the next file

        # Display the initial state
        print(f"Translating titles in {filename}...")

        # Translate titles in parallel with progress bar
        df['TranslatedTitle'] = translate_titles_parallel(df['Title'].tolist())

        # Save the DataFrame to the output directory
        df.to_csv(output_file_path, index=False, encoding='utf-8-sig')

        print(f"{filename}: Titles translated and file saved to␣
 ↪{output_file_path}.")

print("All files processed and titles translated.")
```

```
Skipping 2020-01.csv: Translated file already exists.
Skipping 2020-02.csv: Translated file already exists.
Skipping 2020-03.csv: Translated file already exists.
Skipping 2020-04.csv: Translated file already exists.
Skipping 2020-05.csv: Translated file already exists.
Skipping 2020-06.csv: Translated file already exists.
Skipping 2020-07.csv: Translated file already exists.
Skipping 2020-08.csv: Translated file already exists.
Skipping 2020-09.csv: Translated file already exists.
Skipping 2020-10.csv: Translated file already exists.
Skipping 2020-11.csv: Translated file already exists.
Skipping 2020-12.csv: Translated file already exists.
Skipping 2021-01.csv: Translated file already exists.
Skipping 2021-02.csv: Translated file already exists.
Skipping 2021-03.csv: Translated file already exists.
Skipping 2021-04.csv: Translated file already exists.
Skipping 2021-05.csv: Translated file already exists.
Skipping 2021-06.csv: Translated file already exists.
Skipping 2021-07.csv: Translated file already exists.
Skipping 2021-08.csv: Translated file already exists.
```

```
Skipping 2021-09.csv: Translated file already exists.
Skipping 2021-10.csv: Translated file already exists.
Skipping 2021-11.csv: Translated file already exists.
Skipping 2021-12.csv: Translated file already exists.
Skipping 2022-01.csv: Translated file already exists.
Skipping 2022-02.csv: Translated file already exists.
Skipping 2022-03.csv: Translated file already exists.
Skipping 2022-04.csv: Translated file already exists.
Skipping 2022-05.csv: Translated file already exists.
Skipping 2022-06.csv: Translated file already exists.
Skipping 2022-07.csv: Translated file already exists.
Skipping 2022-08.csv: Translated file already exists.
Skipping 2022-09.csv: Translated file already exists.
Skipping 2022-10.csv: Translated file already exists.
Skipping 2022-11.csv: Translated file already exists.
Skipping 2022-12.csv: Translated file already exists.
Skipping 2023-01.csv: Translated file already exists.
Skipping 2023-02.csv: Translated file already exists.
Skipping 2023-03.csv: Translated file already exists.
Skipping 2023-04.csv: Translated file already exists.
Skipping 2023-05.csv: Translated file already exists.
Skipping 2023-06.csv: Translated file already exists.
Skipping 2023-07.csv: Translated file already exists.
Skipping 2023-08.csv: Translated file already exists.
Translating titles in 2023-09.csv…

Translating Titles:
100%|                                   | 7454/7454
[14:01<00:00,  8.86it/s]

2023-09.csv: Titles translated and file saved to
C:\Users\chunk\OneDrive\Documents\FYP\Sentiment Analysis\TESLA\Second
trial\Translated Tesla monthly news (2020-2023)\2023-09_translated.csv.
```

Below code are the data preprocessing on the “Translated Title” column of the Tesla headline news datasets:

[1]:
```
import os
import pandas as pd
import re
import spacy
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from tqdm import tqdm
from concurrent.futures import ThreadPoolExecutor

# Ensure the necessary NLTK resources are downloaded
nltk.download('stopwords')
nltk.download('wordnet')

# Load SpaCy's English model
nlp = spacy.load("en_core_web_sm")

# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Custom function for targeted manual preprocessing
def manual_preprocessing(text):
    # Replace specific known patterns manually before SpaCy processing
    text = re.sub(r'\bU\s*\.\s*S\s*\.\b', 'United States', text, flags=re.
 ↪IGNORECASE)
    return text

# Function to handle text using SpaCy and custom replacements
def preprocess_text_spacy(text):
    if isinstance(text, str):  # Ensure the input is a string
        text = manual_preprocessing(text)  # Apply manual preprocessing first
        doc = nlp(text)  # Process text with SpaCy
        tokens = []

        for token in doc:
            # Replace specific abbreviations
            if token.text in ["U.S.", "US"]:
                tokens.append("United States")
            elif token.ent_type_ == "GPE":  # Geopolitical entity recognition
                tokens.append(token.text)  # Keep recognized entities as they␣
 ↪are
            else:
                tokens.append(token.text.lower())  # Lowercase other tokens

        text = ' '.join(tokens)
        text = re.sub(r'[^\w\s]', '', text)  # Remove any leftover punctuation
        words = text.split()
        words = [word for word in words if word not in stop_words]  # Remove␣
 ↪stopwords
        words = [lemmatizer.lemmatize(word) for word in words]  # Lemmatize␣
 ↪words
        return ' '.join(words)  # Join words back into a single string
    else:
        return ""  # Return an empty string for non-string entries

# Function to process a single file
def process_file(file_path, output_file_path):
    # Load the translated CSV file into a DataFrame
    df = pd.read_csv(file_path, encoding='utf-8-sig')

    # Preprocess the 'TranslatedTitle' column using the SpaCy-based function
    df['TranslatedTitle'] = df['TranslatedTitle'].apply(preprocess_text_spacy)

    # Save the preprocessed DataFrame to a new CSV file
    df.to_csv(output_file_path, index=False, encoding='utf-8-sig')

    return f"Preprocessed file saved to {output_file_path}."

# Define the input directory for translated files
input_directory = r'C:\Users\chunk\OneDrive\Documents\FYP\Sentiment␣
 ↪Analysis\TESLA\Third trial\Corrected Tesla Montly Translated News'

# Define the output directory for preprocessed files
preprocessed_output_directory = r'C:
 ↪\Users\chunk\OneDrive\Documents\FYP\Sentiment Analysis\TESLA\Third␣
 ↪trial\Preprocessing Tesla Monthly News'
os.makedirs(preprocessed_output_directory, exist_ok=True)

# Use ThreadPoolExecutor to process files sequentially
with ThreadPoolExecutor(max_workers=10) as executor:  # Set max_workers to 1␣
 ↪for sequential processing
    for filename in tqdm(os.listdir(input_directory), desc="Processing Files"):
```

```
        if filename.endswith("_translated.csv"):
            file_path = os.path.join(input_directory, filename)
            output_file_name = filename.replace("_translated.csv",␣
↪"_preprocessed.csv")
            output_file_path = os.path.join(preprocessed_output_directory,␣
↪output_file_name)

            if os.path.exists(output_file_path):
                print(f"Preprocessed file already exists at {output_file_path}.␣
↪Skipping.")
                continue

            # Submit the file for processing and immediately wait for it to␣
↪complete
            future = executor.submit(process_file, file_path, output_file_path)
            result = future.result()  # Wait for the current file to finish␣
↪processing
            print(result)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\chunk\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\chunk\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
Processing Files:   2%|
| 1/48 [01:00<47:13, 60.28s/it]

Preprocessed file saved to C:\Users\chunk\OneDrive\Documents\FYP\Sentiment
Analysis\TESLA\Third trial\Preprocessing Tesla Monthly
News\2020-01_preprocessed.csv.

Processing Files:   4%|
| 2/48 [01:52<42:38, 55.62s/it]

Preprocessed file saved to C:\Users\chunk\OneDrive\Documents\FYP\Sentiment
Analysis\TESLA\Third trial\Preprocessing Tesla Monthly
News\2020-02_preprocessed.csv.

Processing Files:   6%|
| 3/48 [02:49<42:08, 56.19s/it]

Preprocessed file saved to C:\Users\chunk\OneDrive\Documents\FYP\Sentiment
Analysis\TESLA\Third trial\Preprocessing Tesla Monthly
News\2020-03_preprocessed.csv.

Processing Files:   8%|
| 4/48 [03:43<40:30, 55.24s/it]

Preprocessed file saved to C:\Users\chunk\OneDrive\Documents\FYP\Sentiment
Analysis\TESLA\Third trial\Preprocessing Tesla Monthly
```

Below code is the Fuzzy Matching on remove the near-duplicate headlines on threshold level of 75%, 80%, 85% and 90%:

[1]:
```
import os
import pandas as pd
import itertools
from tqdm import tqdm
from concurrent.futures import ThreadPoolExecutor, as_completed  # Optimized␣
↪parallel execution
from rapidfuzz import fuzz  # Faster than fuzzywuzzy

# Function to remove near-duplicate titles using fuzzy matching
def remove_near_duplicates(df, threshold=85):
    titles = df['TranslatedTitle'].tolist()
    to_remove = set()
    seen_pairs = {}

    # Generate all unique pairs of indices to compare
    for i, j in itertools.combinations(range(len(titles)), 2):
        if i in to_remove or j in to_remove:
            continue

        title1, title2 = titles[i], titles[j]

        # Ensure both titles are strings before comparison
        if isinstance(title1, str) and isinstance(title2, str):
            # Early exit for exact matches
            if title1 == title2:
                to_remove.add(j)
                continue

            # Use rapidfuzz (MUCH faster than fuzzywuzzy)
            similarity = fuzz.token_set_ratio(title1, title2)

            if similarity >= threshold:
                to_remove.add(j)

    df_cleaned = df.drop(df.index[list(to_remove)])
    num_removed = len(to_remove)
    return df_cleaned, num_removed
```

```python
# Function to process a single file for one threshold
def process_file_for_threshold(filename, input_directory, output_directory,␣
↪threshold):
    output_file_name = filename.replace(".csv",␣
↪f"_cleaned_threshold_{threshold}.csv")
    output_file_path = os.path.join(output_directory, output_file_name)

    if os.path.exists(output_file_path):
        return None  # Skip if already processed

    file_path = os.path.join(input_directory, filename)

    try:
        df = pd.read_csv(file_path, encoding='utf-8-sig')
    except UnicodeDecodeError:
        df = pd.read_csv(file_path, encoding='latin1')

    df_cleaned, num_removed = remove_near_duplicates(df, threshold=threshold)

    df_cleaned.to_csv(output_file_path, index=False, encoding='utf-8-sig')

    return {
        'File': filename,
        'Total Titles': len(df),
        'Titles Removed': num_removed,
        'Threshold': threshold
    }

# Define the input directory for preprocessed files (Lementizing Clean)
input_directory = r"C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data␣
↪Cleaning\Data Preprocessing\Lementizing Clean"

# Define the output directory for the optimized fuzzy matching
output_directory = r"C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data␣
↪Cleaning\Data Preprocessing\Fuzzy Matching Second approach"

# Ensure output directory exists
os.makedirs(output_directory, exist_ok=True)

# Define thresholds to process
threshold_values = [90, 85, 80, 75]

# Get the list of CSV files
file_list = [f for f in os.listdir(input_directory) if f.endswith(".csv")]

# Use **ThreadPoolExecutor** for parallel execution
```

```python
for threshold in threshold_values:
    print(f"\n Processing files for threshold {threshold}...\n")

    removal_summary = []
    futures = []

    with ThreadPoolExecutor(max_workers=10) as executor:  # Adjust␣
↪`max_workers` based on your CPU
        with tqdm(total=len(file_list), desc=f"Threshold {threshold}␣
↪Processing", leave=True, ncols=100) as pbar:
            for filename in file_list:
                future = executor.submit(process_file_for_threshold, filename,␣
↪input_directory, output_directory, threshold)
                futures.append(future)

            for future in as_completed(futures):
                result = future.result()
                if result:
                    removal_summary.append(result)
                pbar.update(1)

    # Save summary for this threshold
    summary_df = pd.DataFrame(removal_summary)
    summary_output_path = os.path.join(output_directory,␣
↪f'Fuzzy_Matching_Removal_Summary_{threshold}.csv')
    summary_df.to_csv(summary_output_path, index=False, encoding='utf-8-sig')

    print(f"\n Summary saved: {summary_output_path}")

# Save a **full summary** for all thresholds
final_summary_df = pd.DataFrame(removal_summary)
final_summary_path = os.path.join(output_directory,␣
↪'Fuzzy_Matching_Full_Summary_All_Thresholds.csv')
final_summary_df.to_csv(final_summary_path, index=False, encoding='utf-8-sig')

print(f"\n  Full summary for all thresholds saved: {final_summary_path}")
```

```
 Processing files for threshold 90…

Threshold 90 Processing: 100%|                    | 48/48
[1:45:26<00:00, 131.79s/it]

 Summary saved: C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data
Cleaning\Data Preprocessing\Fuzzy Matching Second
approach\Fuzzy_Matching_Removal_Summary_90.csv
```

```
  Processing files for threshold 85…

Threshold 85 Processing: 100%|                    | 48/48
[1:30:56<00:00, 113.67s/it]

  Summary saved: C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data
Cleaning\Data Preprocessing\Fuzzy Matching Second
approach\Fuzzy_Matching_Removal_Summary_85.csv

  Processing files for threshold 80…

Threshold 80 Processing: 100%|                    | 48/48
[1:22:56<00:00, 103.68s/it]

  Summary saved: C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data
Cleaning\Data Preprocessing\Fuzzy Matching Second
approach\Fuzzy_Matching_Removal_Summary_80.csv

  Processing files for threshold 75…

Threshold 75 Processing: 100%|                    | 48/48
[1:13:34<00:00, 91.97s/it]

  Summary saved: C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data
Cleaning\Data Preprocessing\Fuzzy Matching Second
approach\Fuzzy_Matching_Removal_Summary_75.csv

   Full summary for all thresholds saved:
C:\Users\chunk\OneDrive\Documents\FYP\Finalize FYP 1\Data Cleaning\Data
Preprocessing\Fuzzy Matching Second
approach\Fuzzy_Matching_Full_Summary_All_Thresholds.csv
```

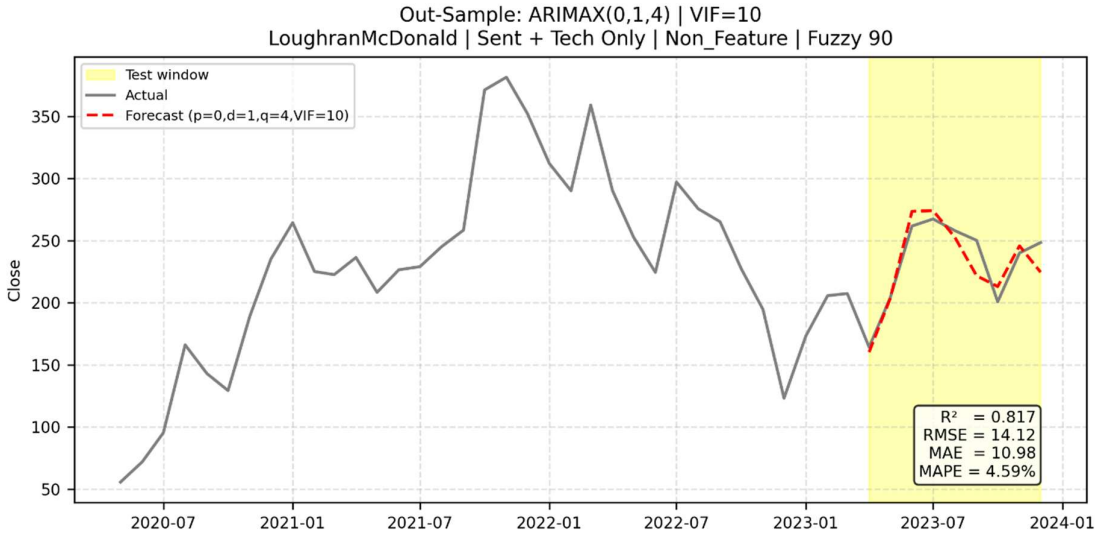
APPENDIX C: FORECAST PLOT FROM EACH BEST SENTIMENT MODEL UNDER ARIMAX AND LSTM PIPELINE

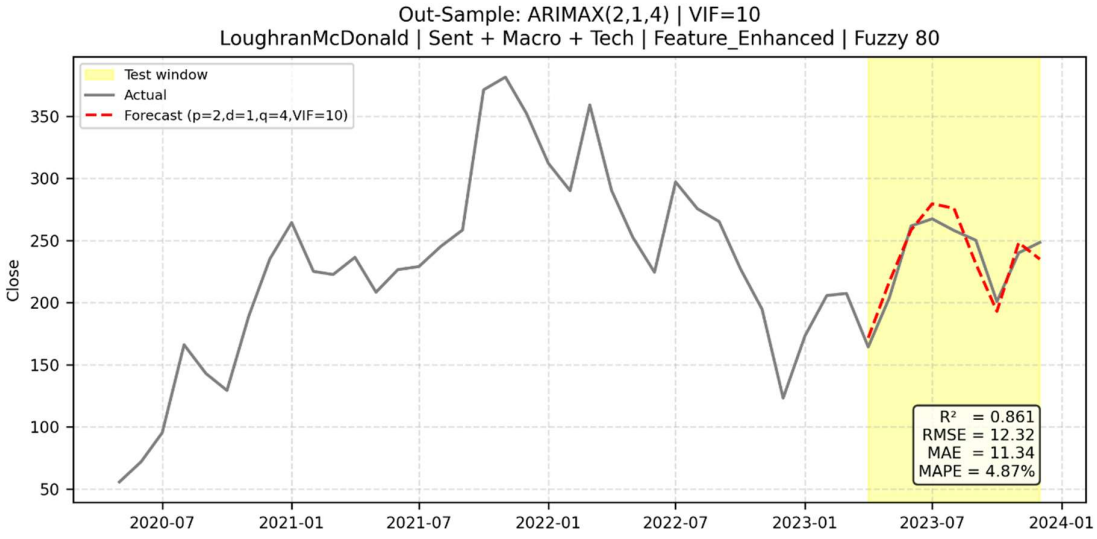
Below are the best forecast plots from each sentiment model for two categories (Sent + Tech + Macro & Sent + Tech):

Lexicon

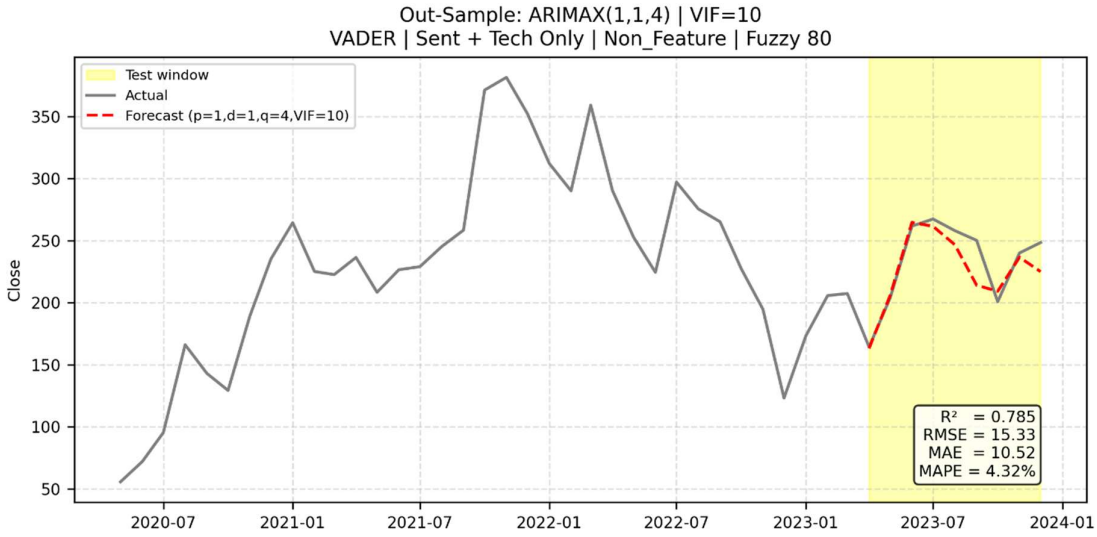
ARIMAX

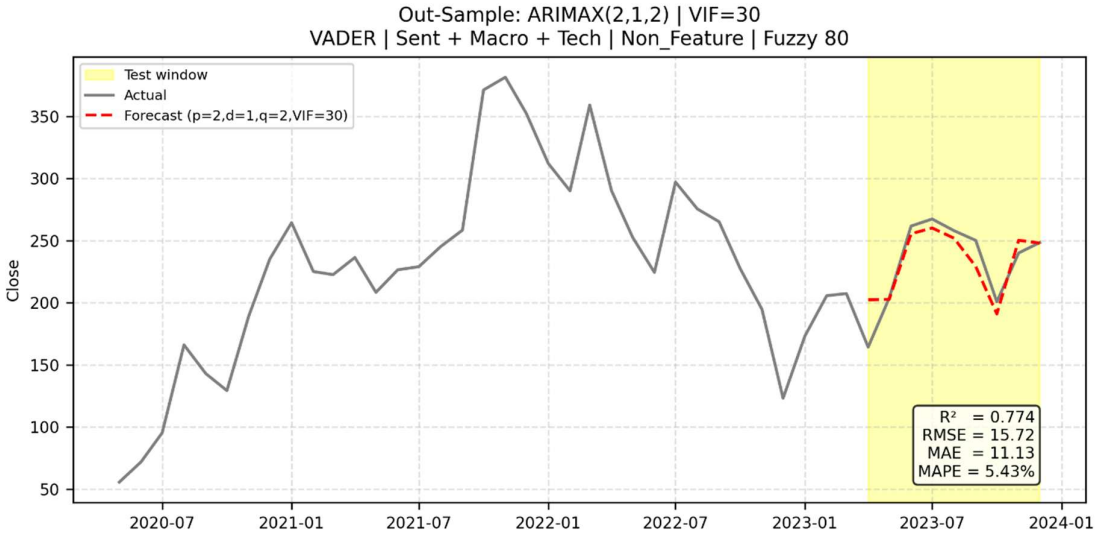
Loughran McDonald:





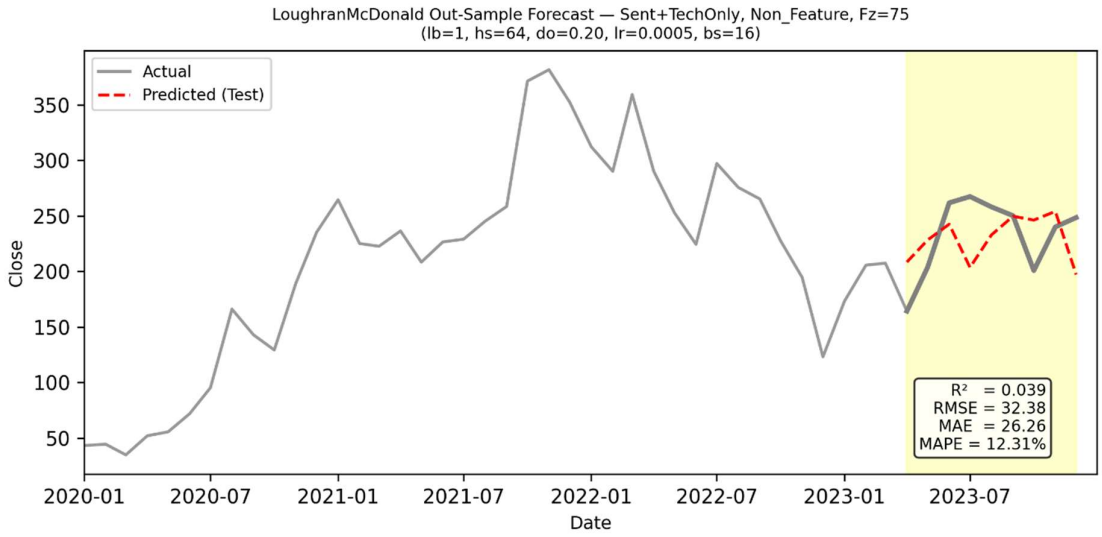
VADER:

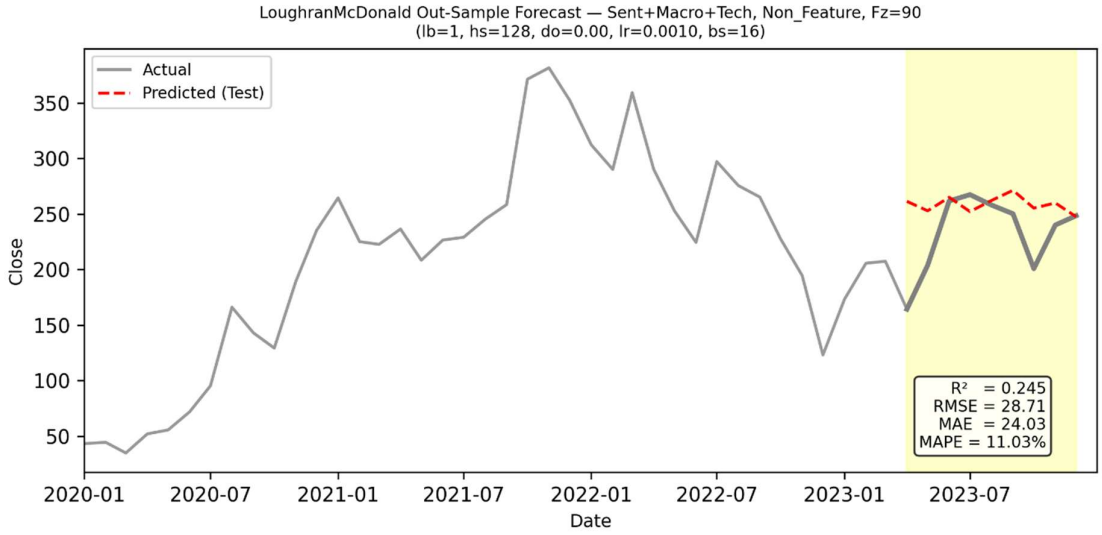

Out-Sample: ARIMAX(1,1,4) | VIF=10
VADER | Sent + Tech Only | Non_Feature | Fuzzy 80
Test window
Actual
Forecast (p=1,d=1,q=4,VIF=10)
Close
R² = 0.785
RMSE = 15.33
MAE = 10.52
MAPE = 4.32%
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01


Out-Sample: ARIMAX(2,1,2) | VIF=30
VADER | Sent + Macro + Tech | Non_Feature | Fuzzy 80
Test window
Actual
Forecast (p=2,d=1,q=2,VIF=30)
Close
R² = 0.774
RMSE = 15.72
MAE = 11.13
MAPE = 5.43%
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01

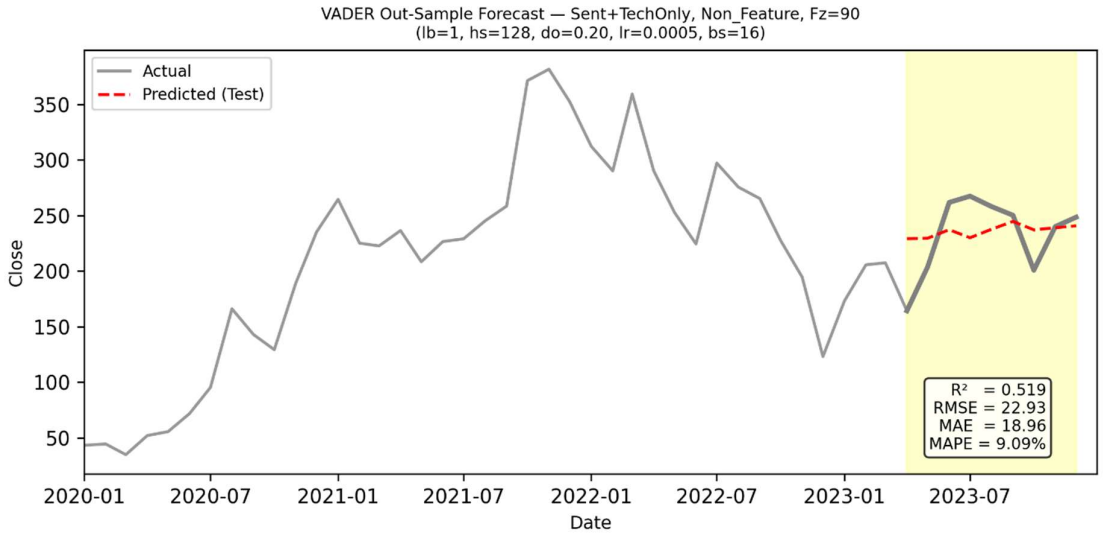
LSTM

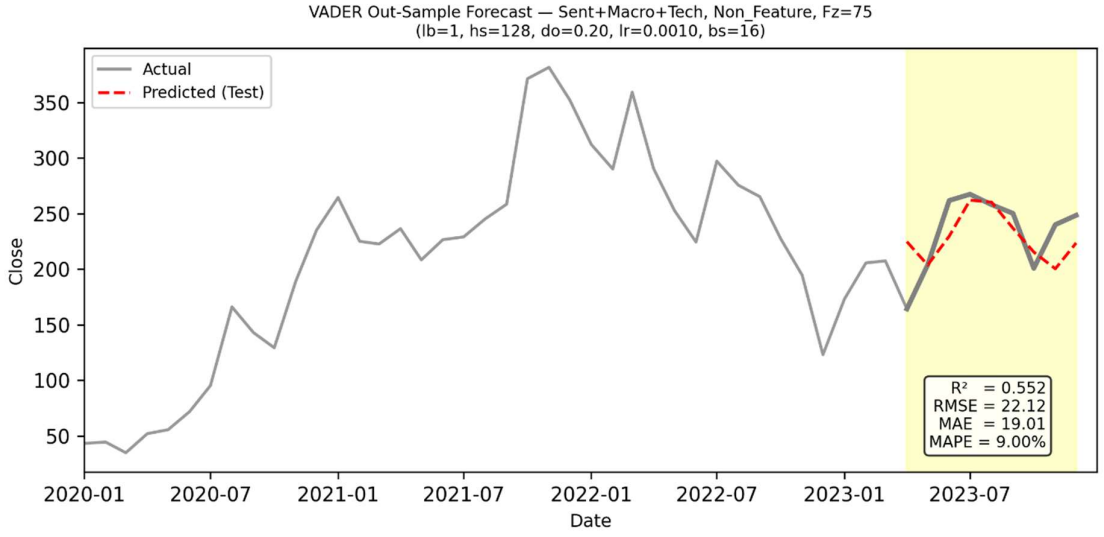
Loughran Mcdonald:





VADER:

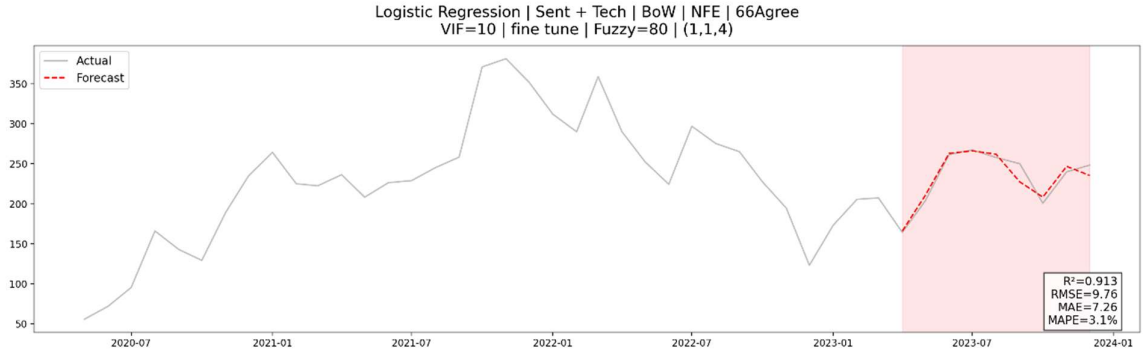

VADER Out-Sample Forecast — Sent+TechOnly, Non_Feature, Fz=90
(lb=1, hs=128, do=0.20, lr=0.0005, bs=16)
Actual
Predicted (Test)
Close
Date
R² = 0.519
RMSE = 22.93
MAE = 18.96
MAPE = 9.09%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

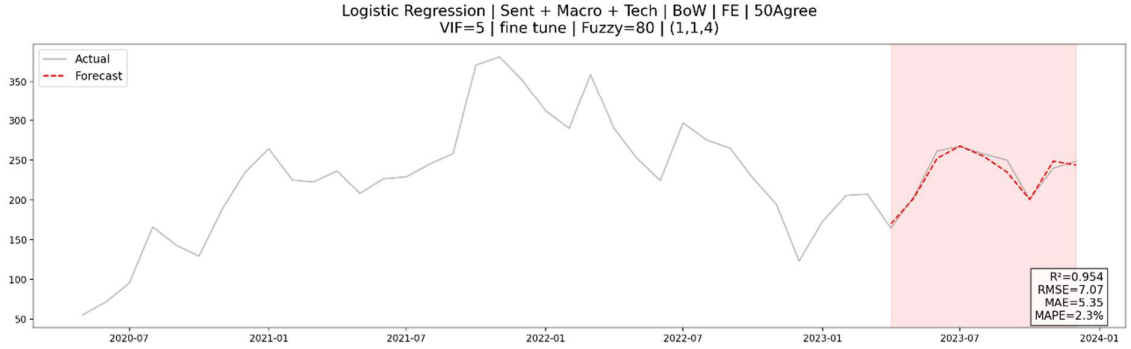

VADER Out-Sample Forecast — Sent+Macro+Tech, Non_Feature, Fz=75
(lb=1, hs=128, do=0.20, lr=0.0010, bs=16)
Actual
Predicted (Test)
Close
Date
R² = 0.552
RMSE = 22.12
MAE = 19.01
MAPE = 9.00%
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

Machine Learning Sentiment Model

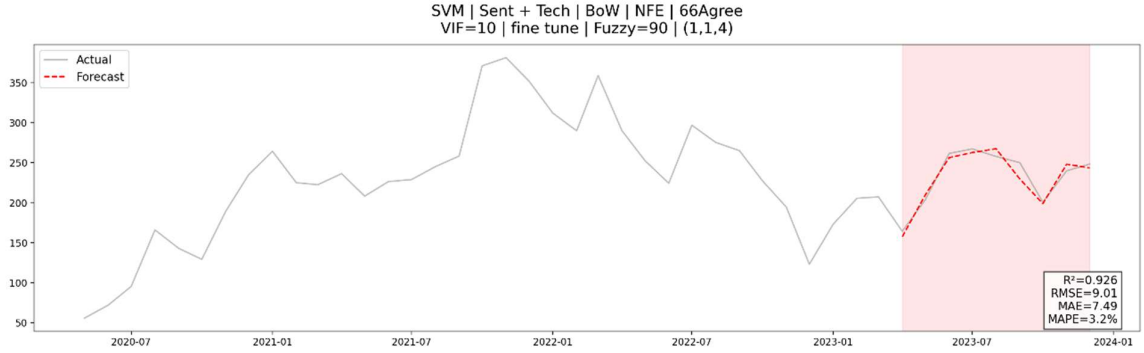
ARIMAX

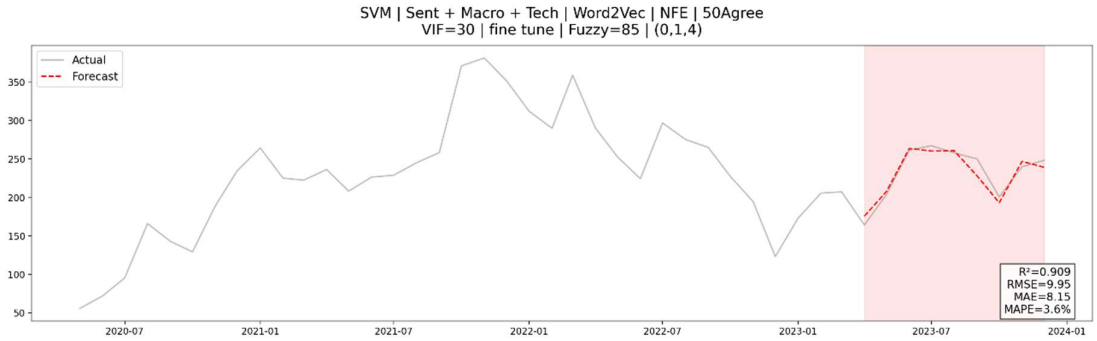
Logistic Regression:



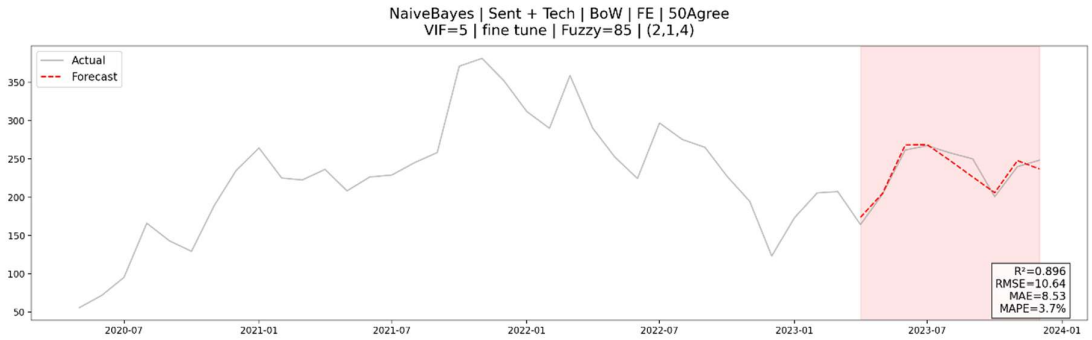


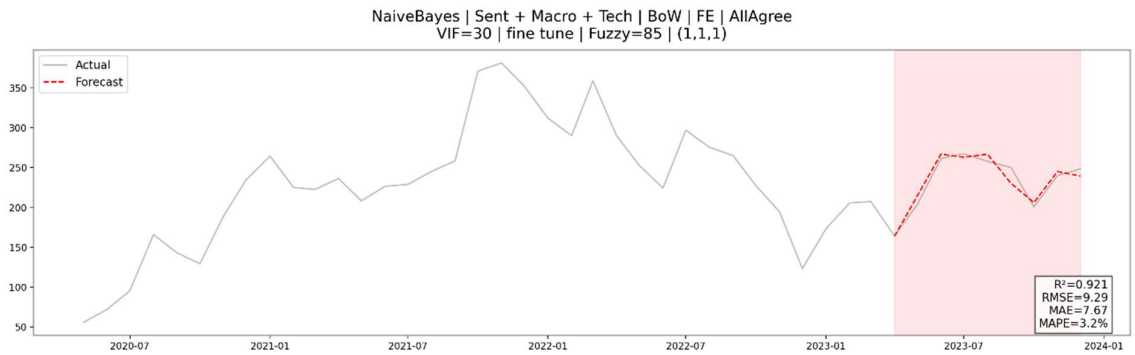
SVM:



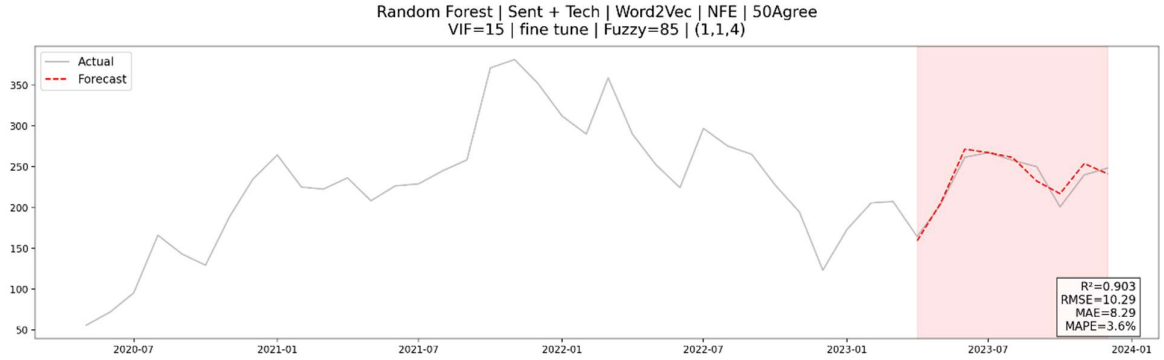

SVM | Sent + Macro + Tech | Word2Vec | NFE | 50Agree
VIF=30 | fine tune | Fuzzy=85 | (0,1,4)
Actual
Forecast
R²=0.909
RMSE=9.95
MAE=8.15
MAPE=3.6%
350
300
250
200
150
100
50
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01

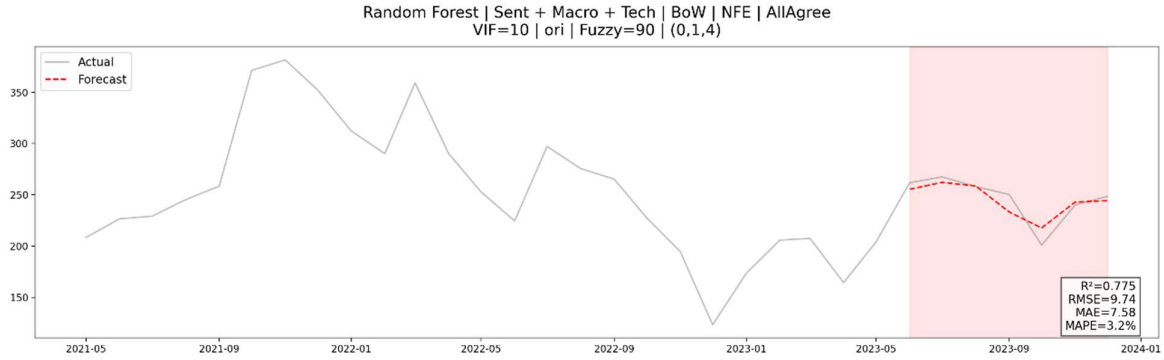
Naive Bayes:


NaiveBayes | Sent + Tech | BoW | FE | 50Agree
VIF=5 | fine tune | Fuzzy=85 | (2,1,4)
Actual
Forecast
R²=0.896
RMSE=10.64
MAE=8.53
MAPE=3.7%
350
300
250
200
150
100
50
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01

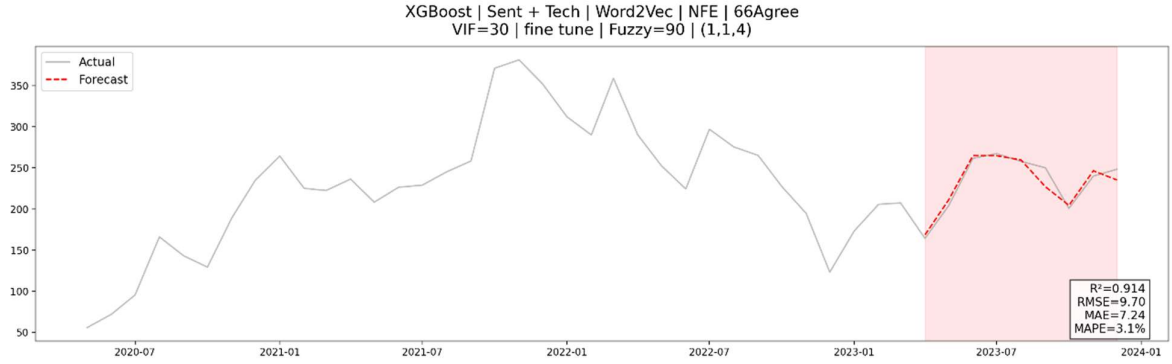

NaiveBayes | Sent + Macro + Tech | BoW | FE | AllAgree
VIF=30 | fine tune | Fuzzy=85 | (1,1,1)
Actual
Forecast
R²=0.921
RMSE=9.29
MAE=7.67
MAPE=3.2%
350
300
250
200
150
100
50
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01

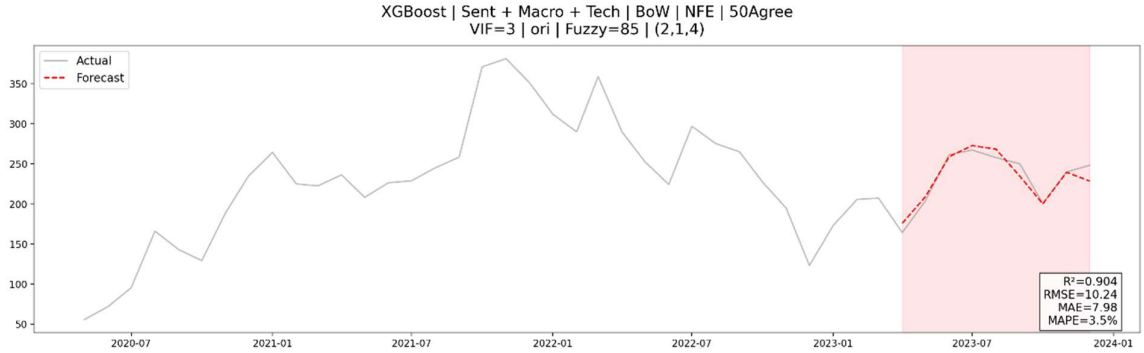
Random Forest:





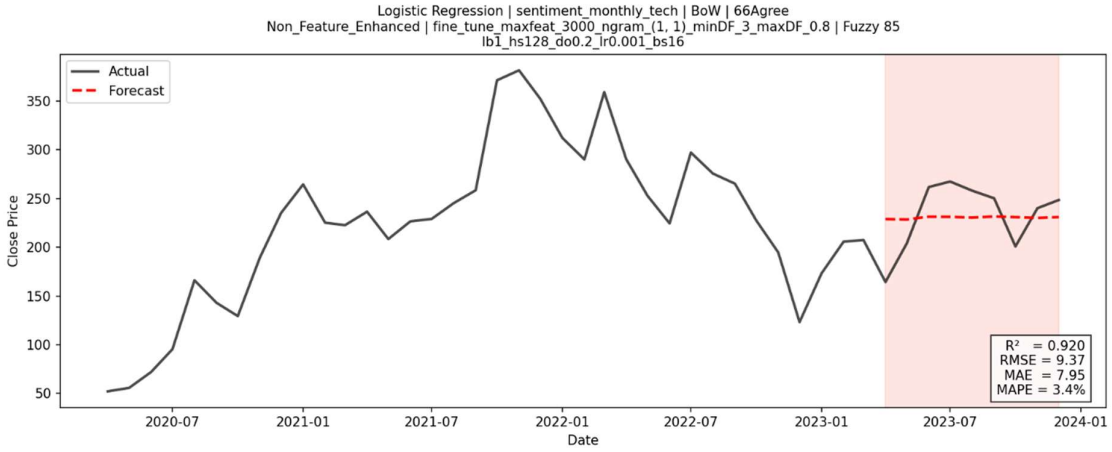
XGBoost:

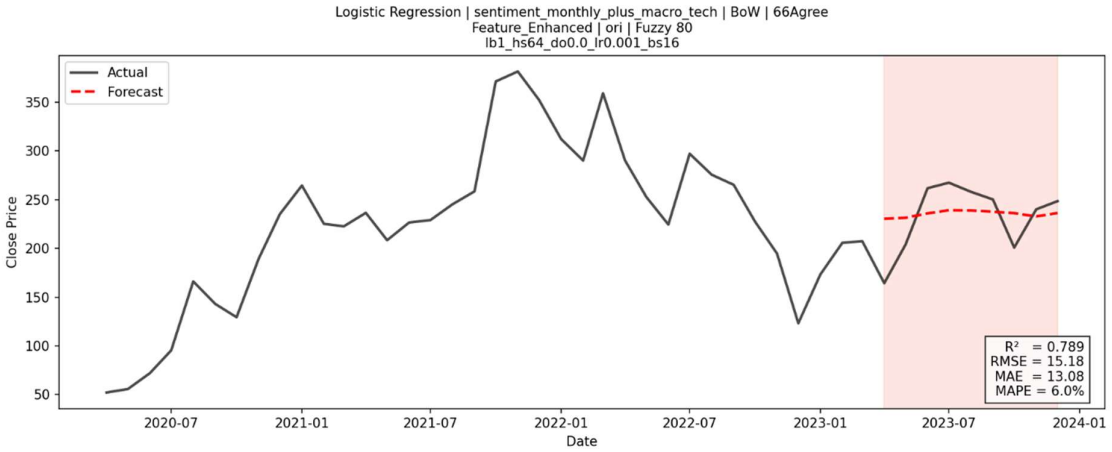




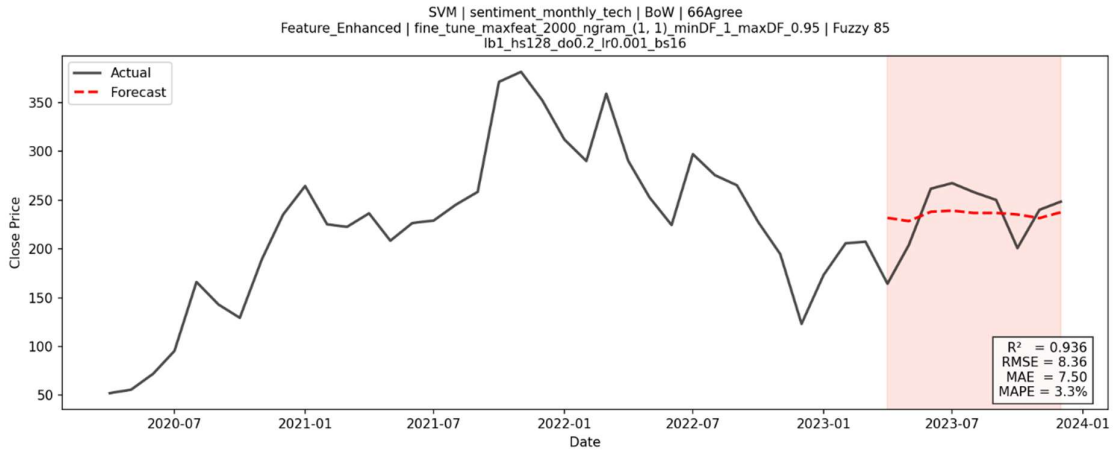
LSTM

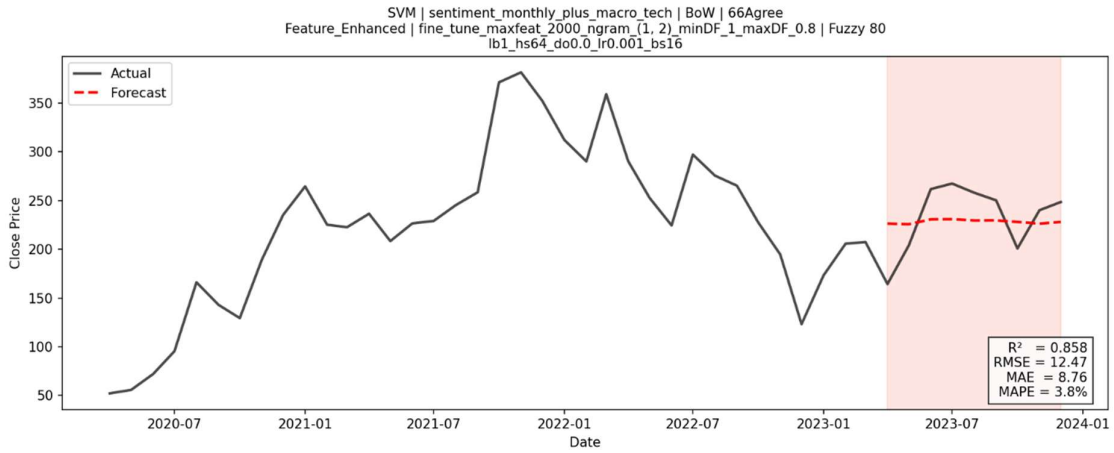
Logistic Regression:



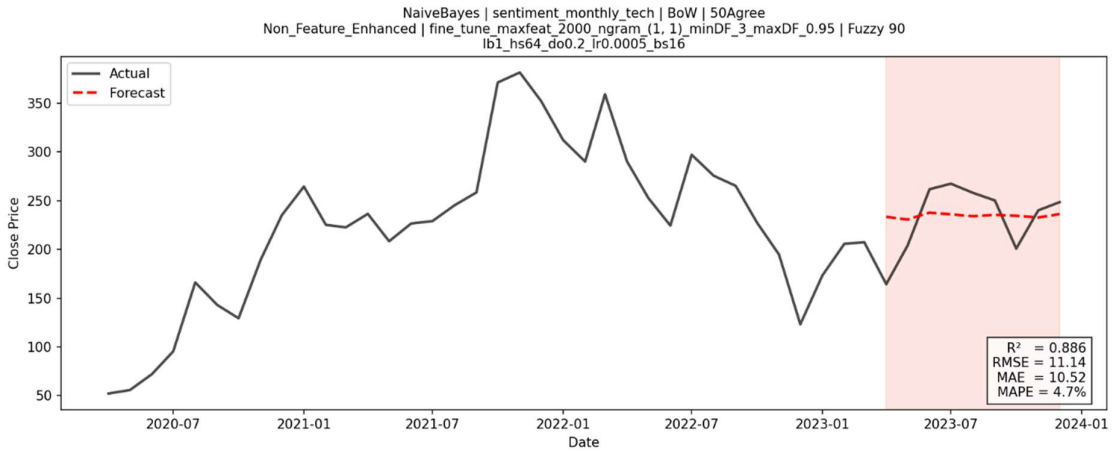


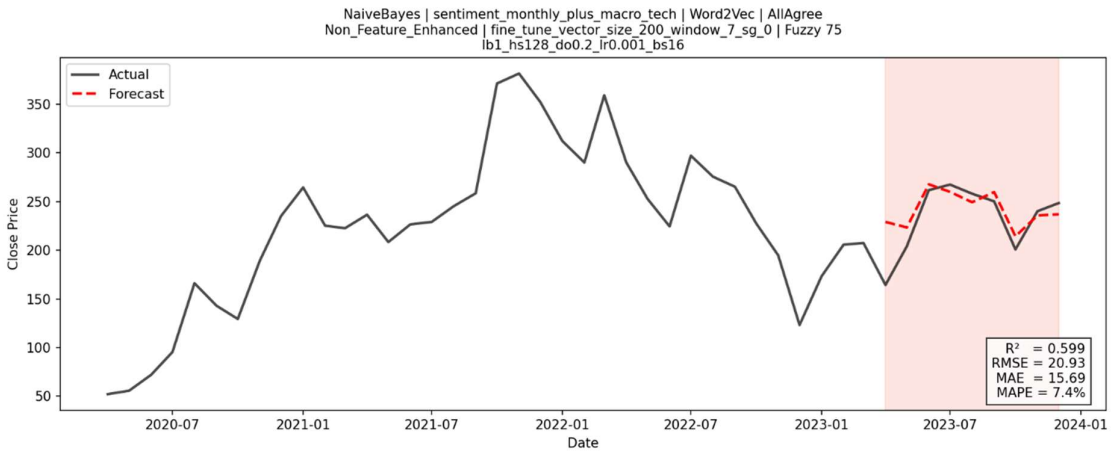
SVM:



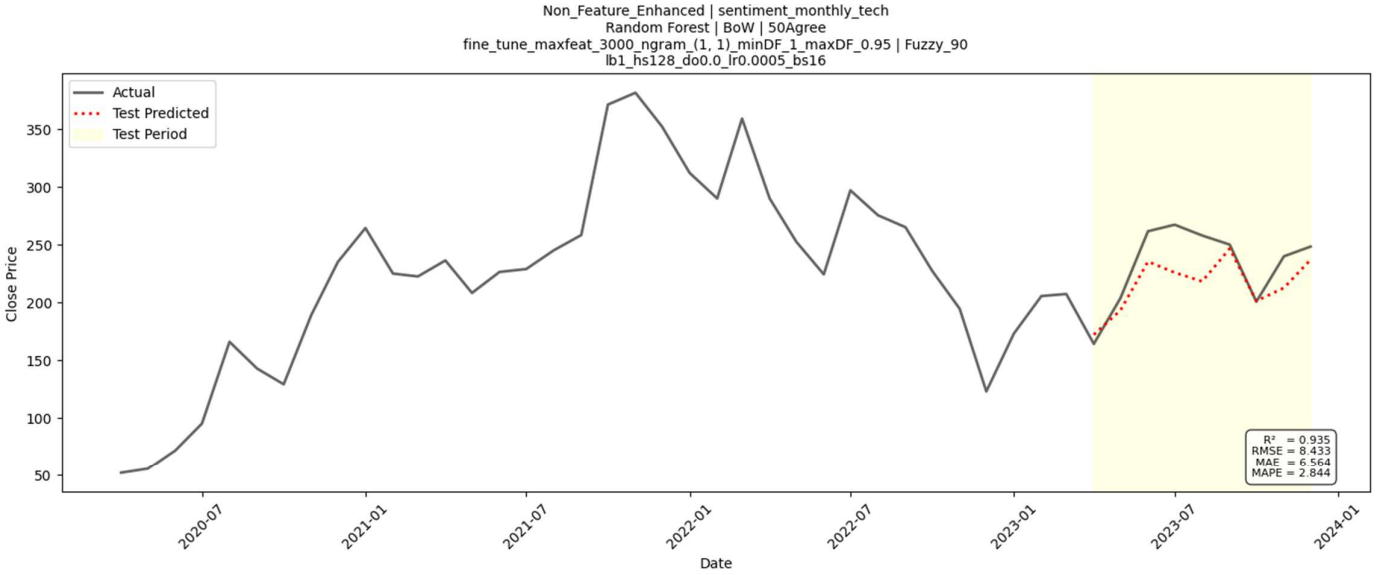


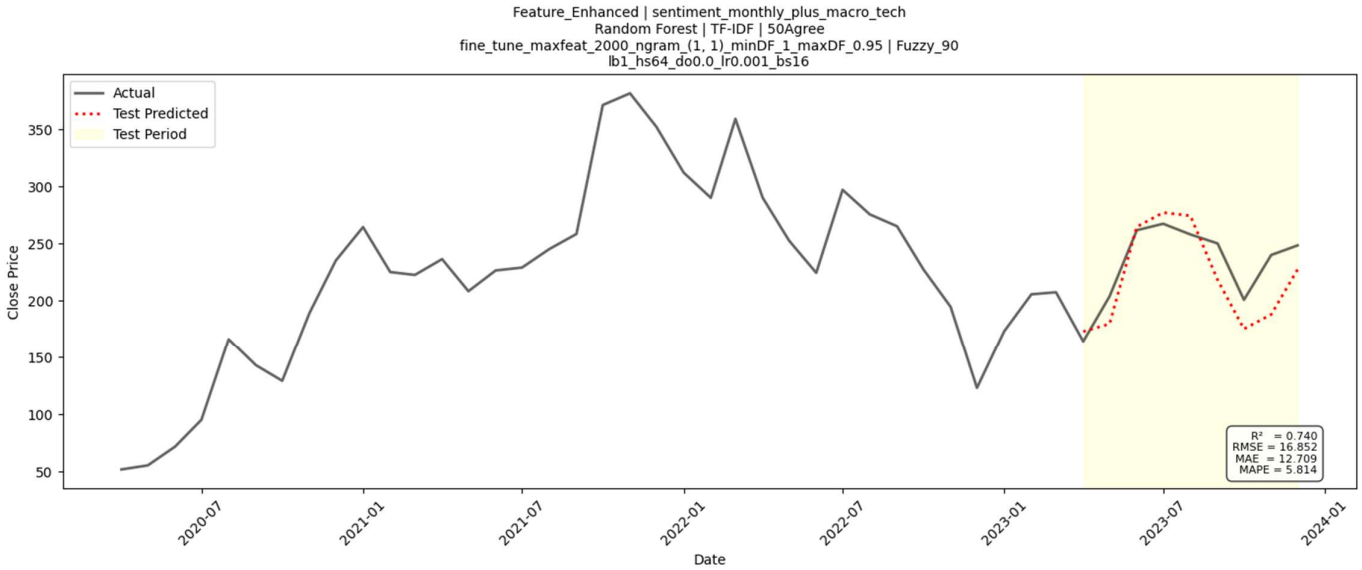
Naive Bayes:



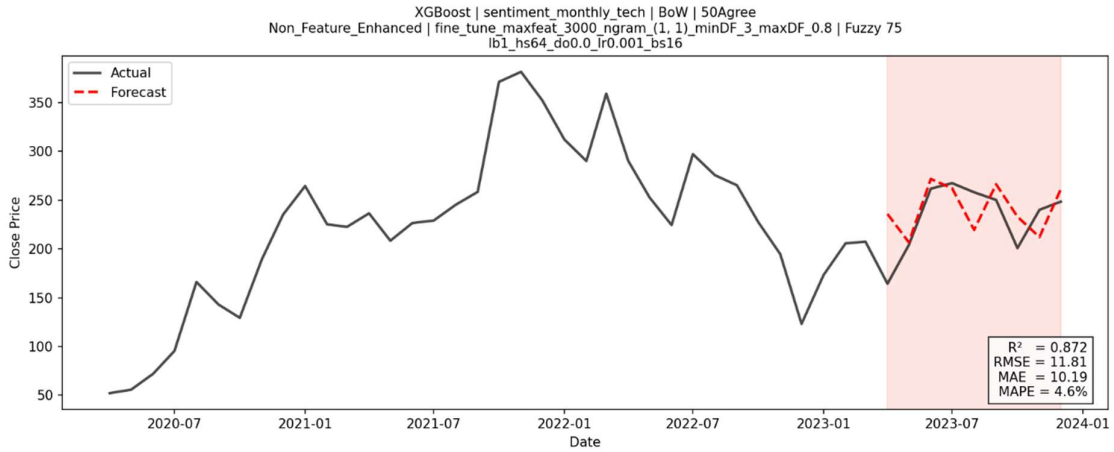


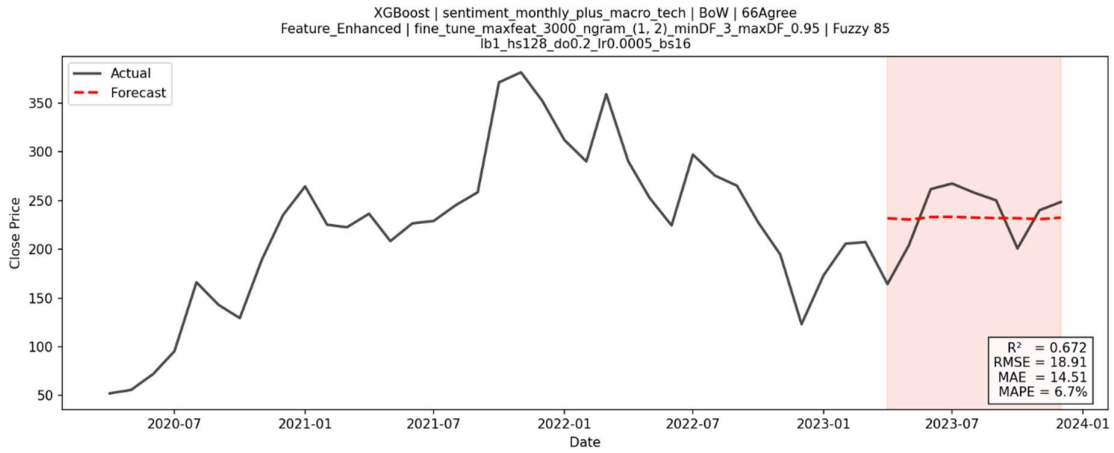
Random Forest:





XGBoost:

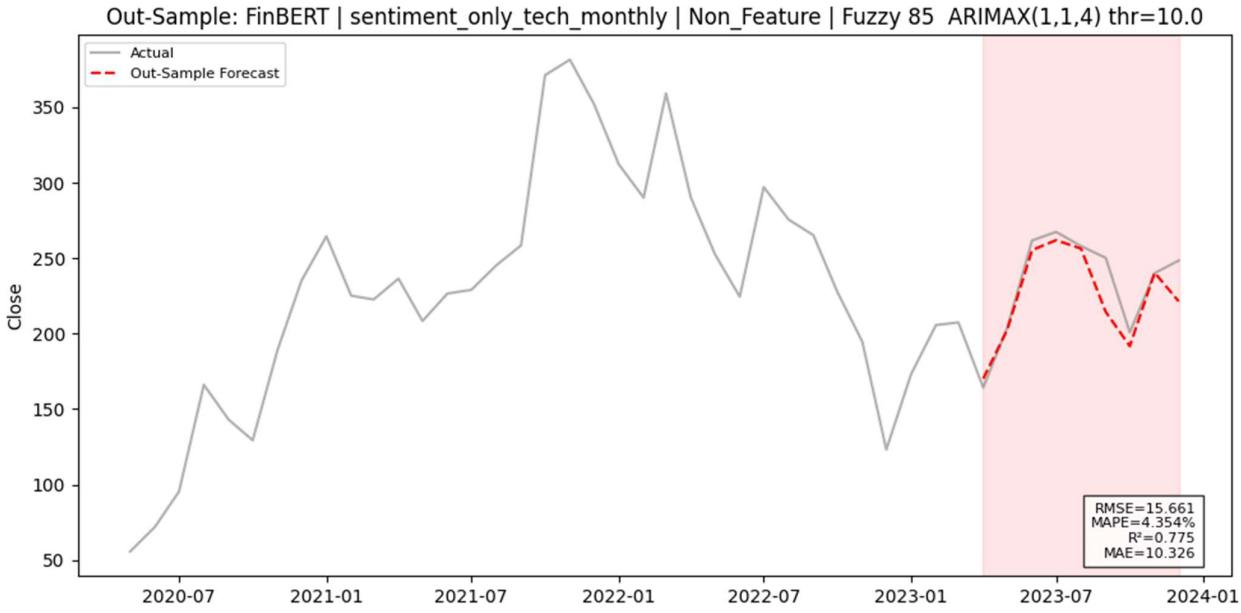

XGBoost | sentiment_monthly_tech | BoW | 50Agree
Non_Feature_Enhanced | fine_tune_maxfeat_3000_ngram_(1, 1)_minDF_3_maxDF_0.8 | Fuzzy 75
lb1_hs64_do0.0_lr0.001_bs16
Actual
Forecast
Close Price
Date
R² = 0.872
RMSE = 11.81
MAE = 10.19
MAPE = 4.6%

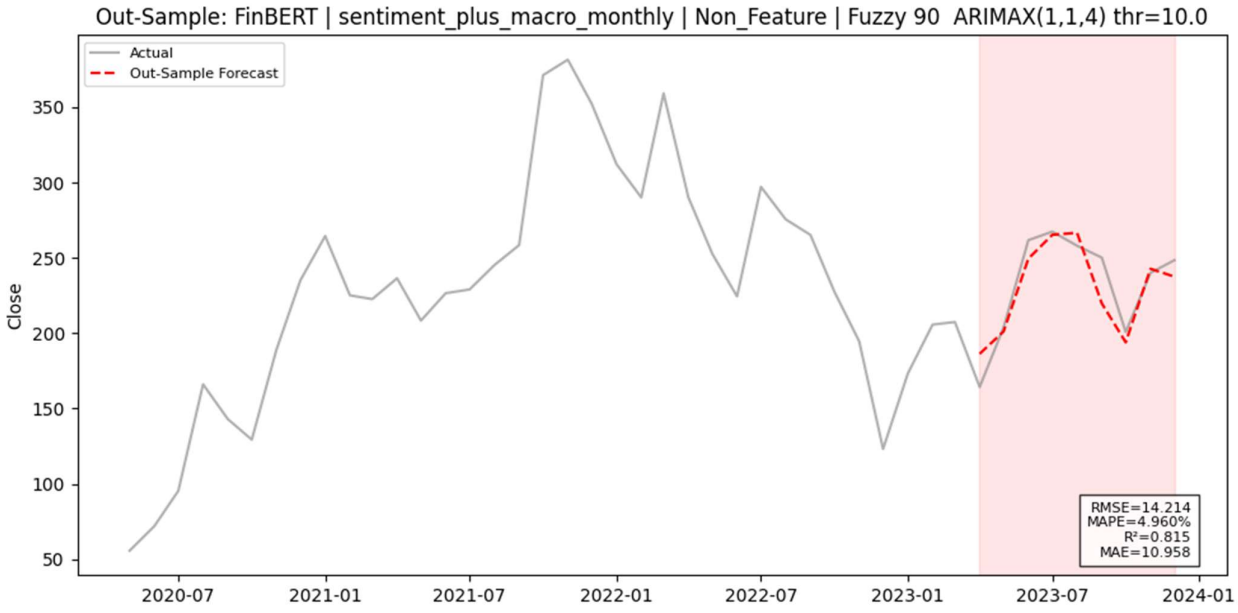

XGBoost | sentiment_monthly_plus_macro_tech | BoW | 66Agree
Feature_Enhanced | fine_tune_maxfeat_3000_ngram_(1, 2)_minDF_3_maxDF_0.95 | Fuzzy 85
lb1_hs128_do0.2_lr0.0005_bs16
Actual
Forecast
Close Price
Date
R² = 0.672
RMSE = 18.91
MAE = 14.51
MAPE = 6.7%

Deep Learning Sentiment Model

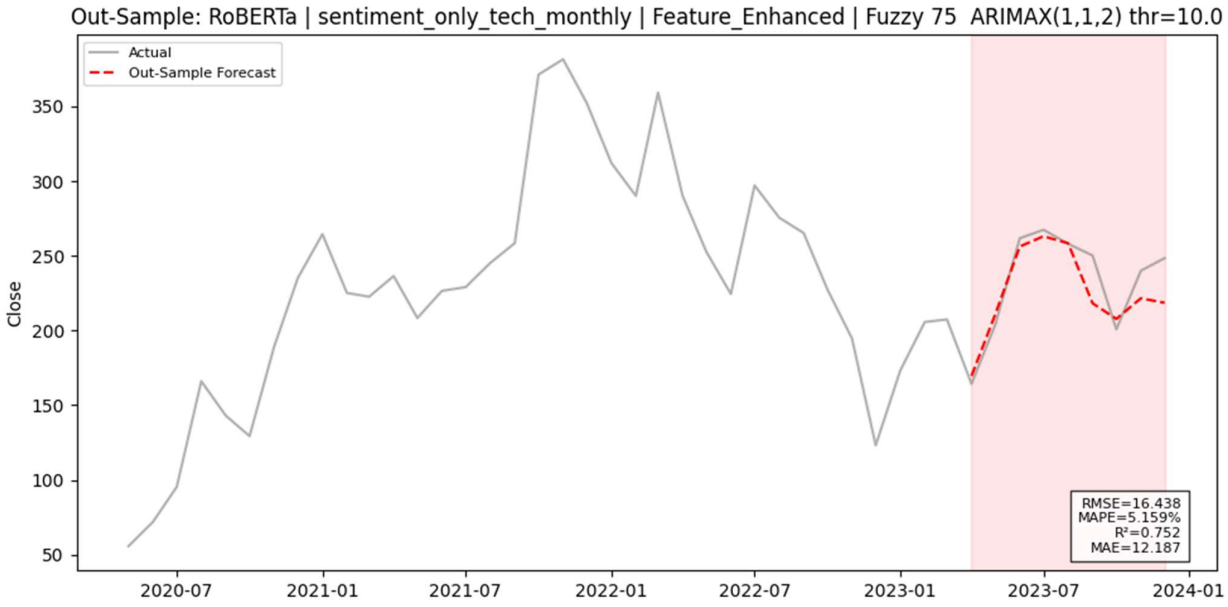
ARIMAX

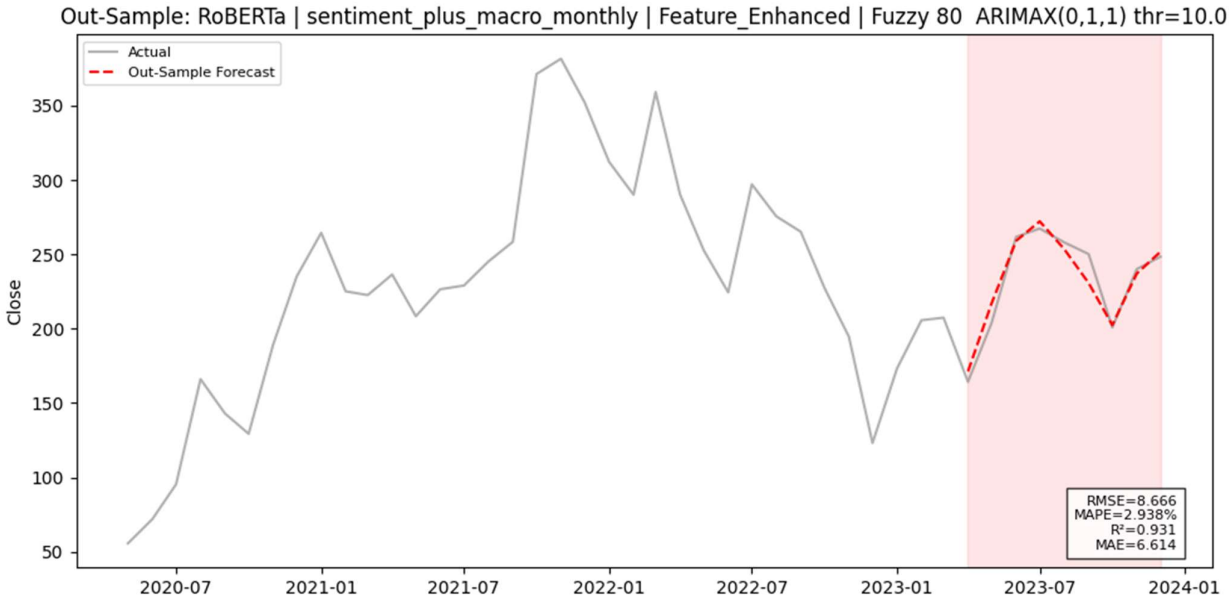
FinBERT:



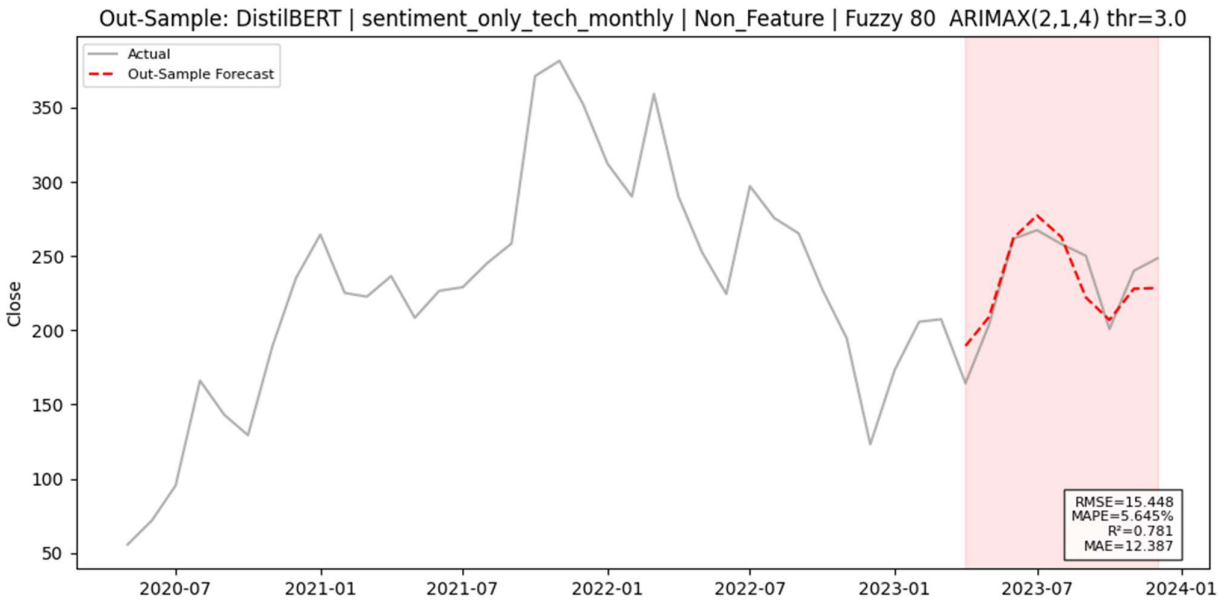


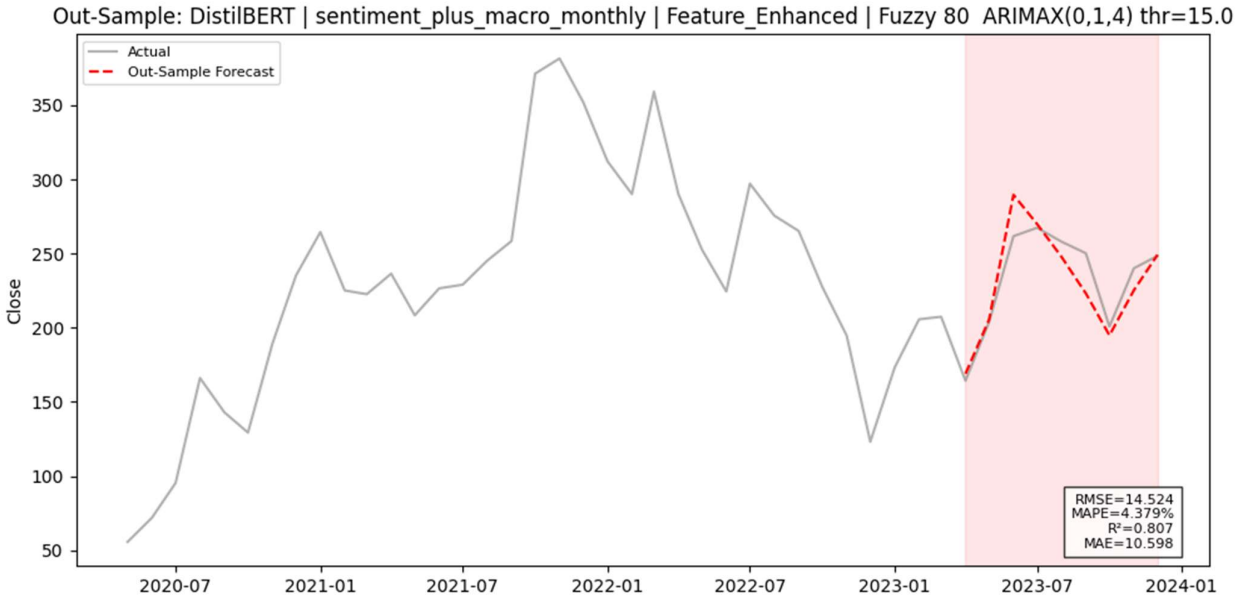
RoBERTa:


Out-Sample: RoBERTa | sentiment_only_tech_monthly | Feature_Enhanced | Fuzzy 75 ARIMAX(1,1,2) thr=10.0
Actual
Out-Sample Forecast
Close
RMSE=16.438
MAPE=5.159%
R²=0.752
MAE=12.187
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01


Out-Sample: RoBERTa | sentiment_plus_macro_monthly | Feature_Enhanced | Fuzzy 80 ARIMAX(0,1,1) thr=10.0
Actual
Out-Sample Forecast
Close
RMSE=8.666
MAPE=2.938%
R²=0.931
MAE=6.614
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01

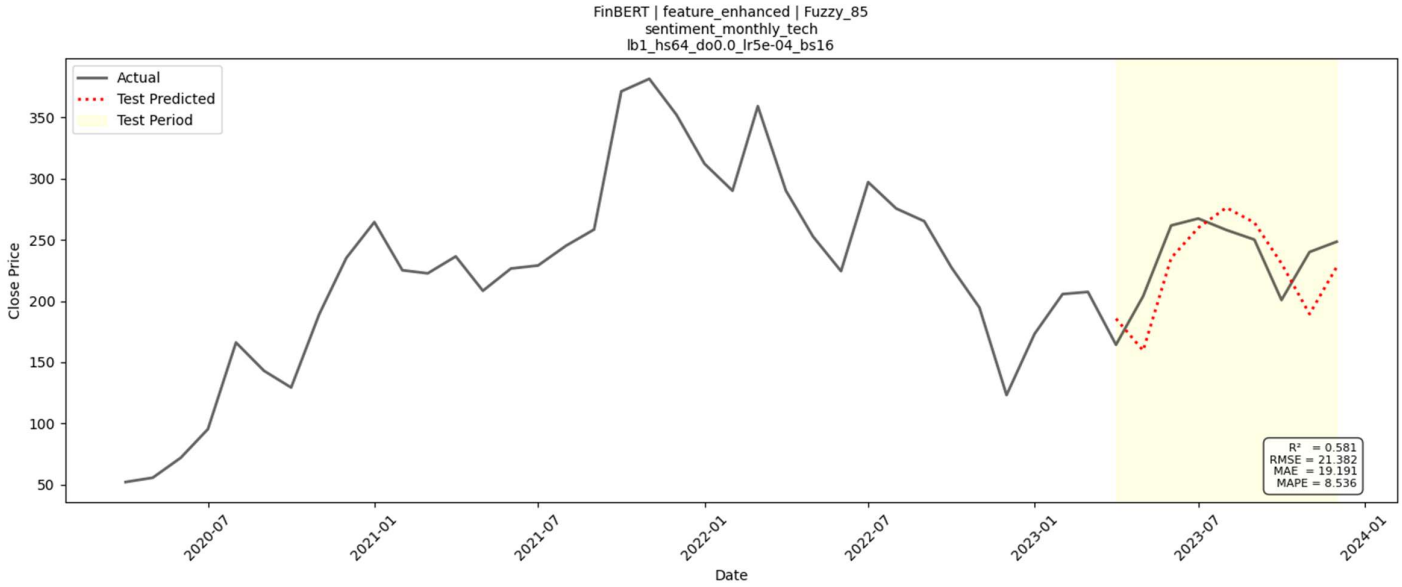
DistillBERT:

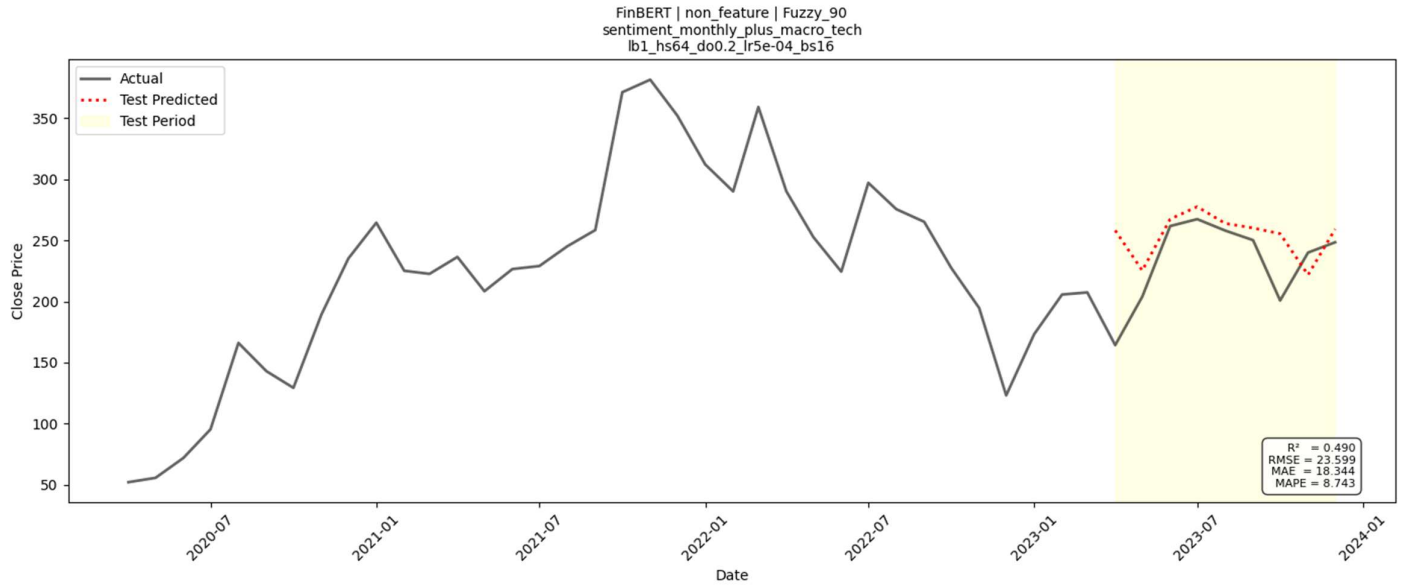




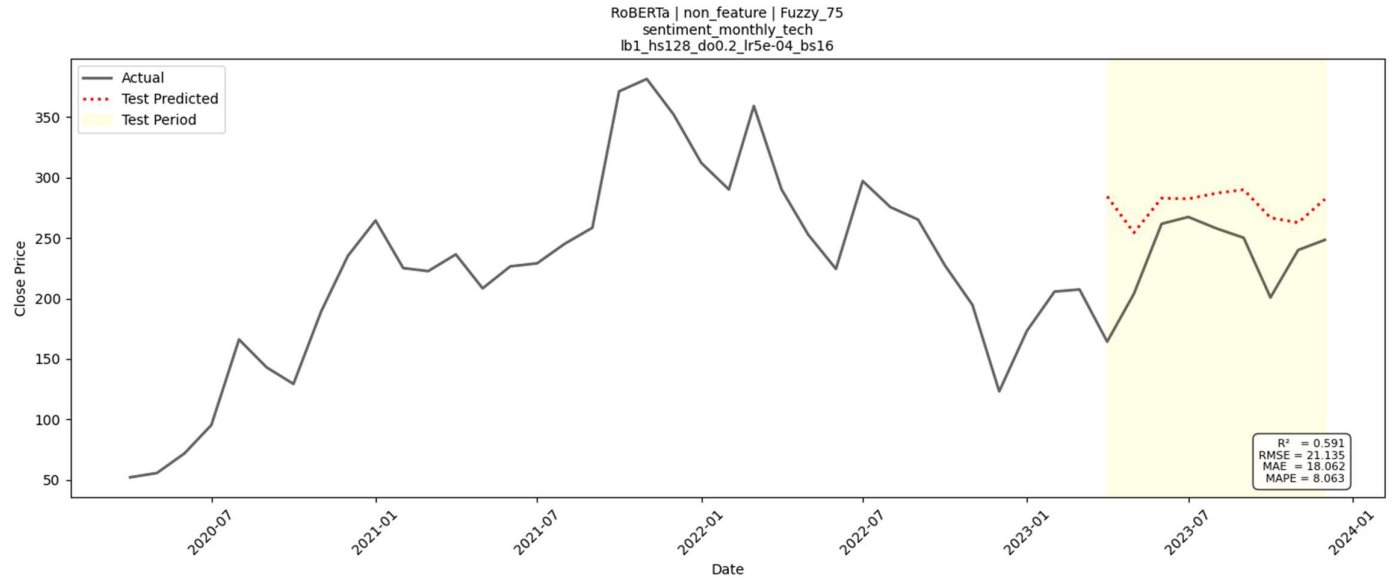
LSTM

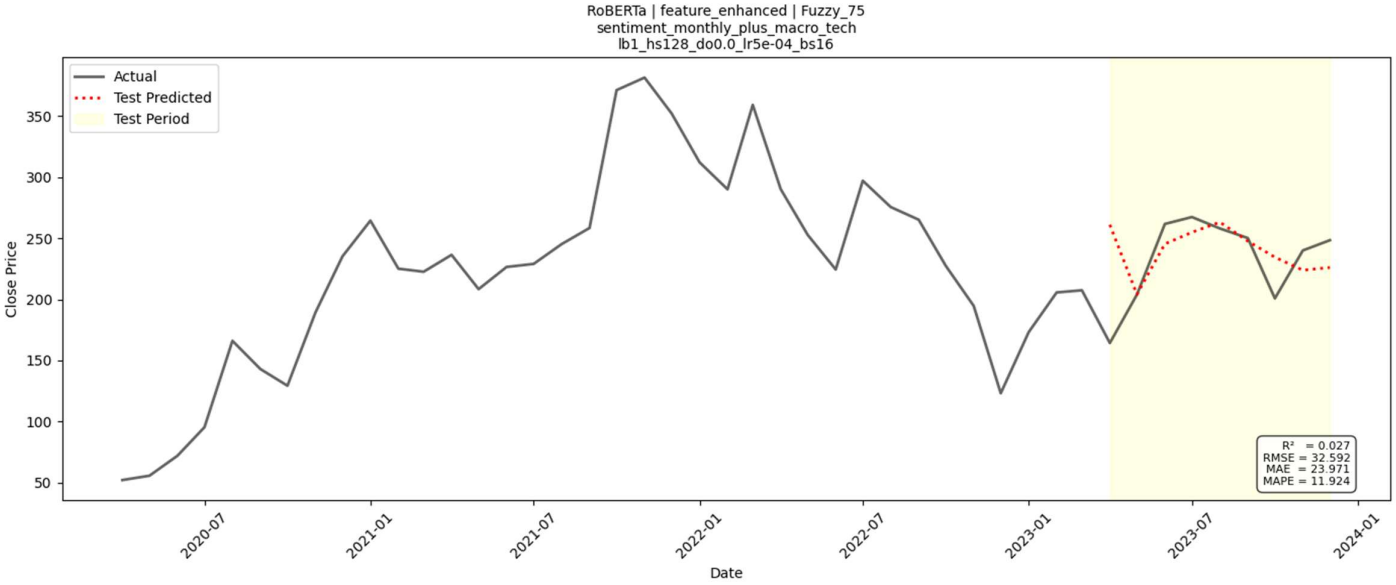
FinBERT:



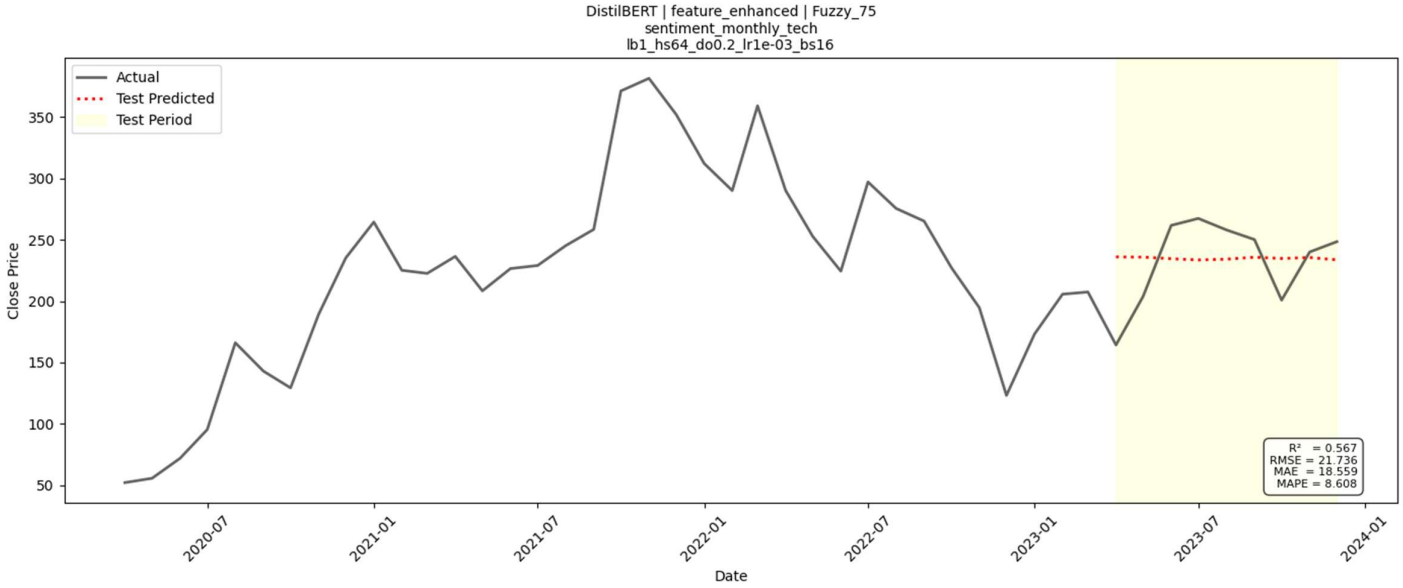


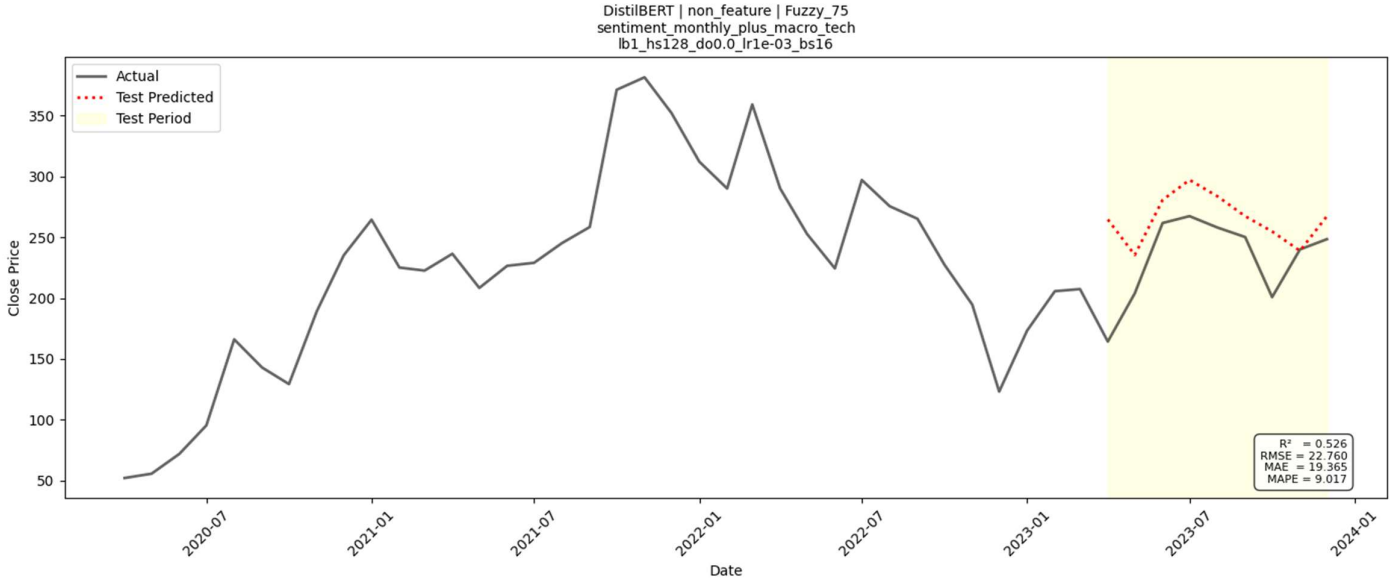
RoBERTa:


RoBERTa | non_feature | Fuzzy_75
sentiment_monthly_tech
lb1_hs128_do0.2_lr5e-04_bs16
Actual
Test Predicted
Test Period
Close Price
Date
R² = 0.591
RMSE = 21.135
MAE = 18.062
MAPE = 8.063


RoBERTa | feature_enhanced | Fuzzy_75
sentiment_monthly_plus_macro_tech
lb1_hs128_do0.0_lr5e-04_bs16
Actual
Test Predicted
Test Period
Close Price
Date
R² = 0.027
RMSE = 32.592
MAE = 23.971
MAPE = 11.924

DistillBERT:





BIODATA OF THE AUTHOR

Name	: Lai Chun Kit
Nationality	: Malaysian
Permanent Address	: No.50, Jalan Bernam 10, Taman Bernam Baru, 35900, Tanjong Malim
Telephone Number	: 017-5142980
Email	: vaniepersie@gmail.com
Education	: Bachelor of Science in Financial Mathematics (with Honours)
Award(s)/ Achievements	: Five Consecutive Dean List with cgpa 3.78 RASIO 8.0 Data Science Competition Third Place Carnival Innovation UMT 2024 Silver Medal (Nomsim: Interest Rate Simulator)

LAPORAN TURNITIN

PITA II Laporan Akhir Lai Chun Kit S65947 Latest.pdf

ORIGINALITY REPORT

16%	14%	12%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Malaysia Terengganu UMT Student Paper	1%
2	ouci.dntb.gov.ua Internet Source	<1%
3	dspace.dtu.ac.in:8080 Internet Source	<1%
4	www.techscience.com Internet Source	<1%
5	ebin.pub Internet Source	<1%
6	Santeri Karppinen, Olli Lohi, Matti Vihola. "Prediction of leukocyte counts during paediatric acute lymphoblastic leukaemia maintenance therapy", Scientific Reports, 2019 Publication	<1%
7	www.springerprofessional.de Internet Source	<1%