

COMPARATIVE ANALYSIS OF DEEP LEARNING MODELS FOR FORECASTING VOLATILITY OF S&P 500 USING SENTIMENT ANALYSIS AND EXPLAINABILITY.

TEJA VAMSI DEVALLA

Supervisor Mark Huntly

SEPTEMBER 2026

STUDENT ID: 2505091

COURSE: MSc Data and Artificial Intelligence

Declaration of Originality

I, Teja Vamsi Devalla, hereby certify that I am the sole author of this project. All sources of information have been properly acknowledged, and the work presented is entirely my own. Furthermore, I confirm that the completed dissertation adheres to the prescribed word limit.

This work has not been submitted previously in fulfilment of the requirements for any other degree or qualification.



Signed:

07-09-2026

Date:

CONSENT FORM

I, Teja Vamsi Devalla, authorize that my MSc project in Data and Artificial Intelligence may be available for inter-library loan and photocopying (in compliance with copyright regulations), and that the title abstract may be shared with external organizations.



Signed:

07-09-2026

Date:

Acknowledgement of Copyright

I confirm that the copyright of this project report, as well as any outputs produced during the course of the project, is vest in the University of Wales Trinity Saint David, Swansea.

Abstract

Forecasting volatility is highly important for controlling financial risk, but it is a complex task because financial markets are dynamic and uncertain conditions. This study compares nine deep learning models to predict the S&P 500 21-day realised volatility. Financial price data and VIX were collected from Yahoo Finance from 2008 to 2024. sentiment was derived by using FinBERT and augmented to financial market data. Models evaluated include CNN, GRU, LSTM, Bidirectional LSTM, PatchTST, Temporal Fusion Transformer (TFT), CNN-LSTM, GARCH-LSTM, and GARCH-GRU. Each model was trained with financial market data and retrained after augmented sentiment and then evaluated using MAE, RMSE, and QLIKE metrics. Results show that attention-based transformer architectures outperformed recurrent and convolutional baselines consistently across all metrics. PatchTST with sentiment achieved the best overall performance with MAE: 0.0140, RMSE: 0.0192, QLIKE: 0.0330, with sentiment augmentation improved only three of the nine architectures. Importantly, GARCH-hybrid models didn't perform well than standalone deep learning models, suggesting GARCH volatility brings unnecessary information. Explainability techniques SHAP and LIME identified sentiment contributed 3-6% of total feature importance. Key limitations include depend only on formal news headlines without any social media data. The results may not apply to other stock market index markets or companies. These findings suggest the value of combining transformer architecture and sentiment analysis for forecasting volatility.

Table of Content

Abstract	2
1. Introduction	3
1.1 Background and Context	3
1.2 Problem Statement and Motivation	4
1.3 Research Gap.....	4
1.4 Research Aim and Objectives	4
1.5 Research Question	5
2. Literature Review	5
2.1 Introduction	5
2.2 Financial Market Volatility and S&P 500	5
2.2.1 Definition and nature of volatility:.....	5
2.2.2 Historical Volatility VS Implied Volatility:	6
2.2.3 Importance of volatility in Finance:.....	6

2.2.4 Characteristics of S&P 500:	6
2.2.5 Facts of Financial volatility:.....	7
2.3 Traditional Volatility Forecasting Methods	7
2.3.1 ARCH model:.....	7
2.3.2 GARCH family:.....	7
2.3.3 EGARCH (Exponential GARCH):	8
2.3.4 TGARCH (Threshold GARCH):	8
2.3.5 Stochastic Volatility models:.....	8
2.3.6 Strengths and limitations of traditional econometric models:	8
2.4 Machine Learning for Financial Forecasting	9
2.4.1 Overview of ML approaches in financial forecasting.....	9
2.4.2 Comparison with econometric (GARCH family) models:	9
2.4.3 Limitations of Machine Learning:.....	9
2.5 Deep Learning Models for Volatility Forecasting	9
2.6 Sentiment Analysis in Financial Markets	11
2.7 Explainable Artificial Intelligence (XAI) in Finance	11
2.8 Evaluation Metrics for Volatility Forecasting	12
2.9 Research Gaps and Critical Synthesis	12
2.10 Summary	13
3. Research Methodology.....	13
3.1 Introduction and Research Design	13
3.2 Research Design.....	13
3.3 Data source and collection.....	14
3.4 Feature Engineering	14
3.5 Preprocessing the data.....	15
3.6 Model Architectures.....	15
3.6.1 Convolutional Neural Network (CNN)	15
3.6.2 Gated Recurrent Unit (GRU).....	16
3.6.3 Long-Short Term Memory (LSTM)	16
3.6.4 Bidirectional-LSTM.....	17
3.6.5 Patch Time Series Transformer (PatchTST)	17
3.6.6 Temporal Fusion Transformer (TFT)	18

3.6.7 Hybrid CNN-LSTM.....	18
3.6.8 Hybrid GARCH-LSTM.....	19
3.6.9 Hybrid GARCH-GRU.....	19
3.7 Sentiment Integration.....	20
3.7.1 News Dataset Loading and EDA:	20
3.7.2 News Data Cleaning and FinBERT Sentiment Analysis:	20
3.7.3 Applying FinBERT Sentiment Analysis to the Full News Dataset:	21
3.7.4 News Sentiment Score and Daily Aggregation:	21
3.7.5 Checking Trading days and News Coverage:.....	21
3.7.6 Merging News Sentiment with Financial Data:	21
3.7.7 Sentiment-Based Model Preparation:	22
3.8 Evaluation of the model:.....	22
3.9 Model Comparison	23
3.10 Explainability:	23
4. Results and Analysis	24
4.1 Model's performance without Sentiment:	24
4.2 Impact of Sentiment Analysis:	25
4.2.1 Model comparison before and after sentiment:.....	26
4.2.2 Overall Impact of Sentiment:	26
4.2.3 Best Performing Sentiment Model:	27
4.3 Model Ranking and Overall Comparison:	27
4.3.1 Best overall model	28
4.4 Explainability Results:	29
4.4.1 SHAP results:	30
4.4.2 LIME results:.....	31
4.4.3 SHAP and LIME:	32
5. Discussions	32
5.1 Model performance	33
5.2 Role of News Sentiment in Volatility Forecasting	33
5.3 GARCH hybrid models.....	34
5.4 Limitations	35
6 Conclusion	35

ACKNOWLEDGEMENTS

I, Teja Vamsi Devalla, would like to sincerely thank my supervisor, Mark Huntly of the University of Wales Trinity Saint David, for his outstanding guidance, encouragement, and support throughout the course of this project. His expertise in deep learning, academic writing, and critical evaluation motivation he provided were instrumental in refining my ideas, strengthening the methodology, and bringing this work to completion.

I also want to express my sincere gratitude to my family, who have been a continual source of strength throughout this journey with their unfailing love, tolerance, and encouragement. I was able to remain dedicated and get over the difficulties of this difficult study because of their confidence in me and ongoing assurance.

Finally, I am grateful to the University of Wales Trinity Saint David for providing access to its digital library and online resources. The scholarly articles available through these platforms offered a strong theoretical foundation that informed my exploration of pneumonia detection and model generalization.

1. Introduction

1.1 Background and Context

Volatility is an essential concept in financial markets because it shows price change of an asset over time. Market with big moments and quick price change means high volatility and small moments in price means low volatility, so markets will be sideways. It's important to understand volatility for investors, hedge fund companies, and financial institutions to make better financial decisions [1]. In finance, volatility is used for risk management of investments; it is also used to calculate option pricing. It is used in portfolio allocation and banks to measure value at risk to estimate losses and manage their risks properly. S&P 500 holds top 500 U.S

based companies from multiple industries, and it is the most actively reacting index to world news with more liquidity. Long ago, econometric models like GARCH were used to predict volatility. These models use past financial data to make predictions, but financial markets depend on many factors. Afterwards, machine learning and deep learning models were used because these methods can study large amounts of data which traditional models can't. Additional data like news and social media came into light which holds emotions and opinions of the public. This information can help to understand the market and improve volatility forecasting [2].

1.2 Problem Statement and Motivation

Forecasting volatility is one of the challenging tasks in the stock market. Traditional models like GARCH and its variants were used for this task because they are mathematically strong and easy to understand but face difficulties in some situations. As financial markets are not linear and prices may change suddenly, traditional models fail to capture these sudden changes. In recent studies, deep learning models have shown better results in forecasting volatility [3]. They can find sudden changes by learning different relations from complex data which traditional models can't. Deep learning models can capture complex nonlinear patterns, but it's extremely difficult to interpret the decision. In the financial industry, it's very important to understand the decision made by the model to implement their strategies. An accurate system that can't be explained is difficult to trust and adopt. Another problem is considering the sentiment of investors, shareholders, and public opinion about the market which is available from news and social media can highly influence the price of the financial market. Most of the volatility forecasting studies ignored sentiment data. This study is motivated by these gaps and there is a need for a study that combines deep learning and explainability along with sentiment analysis.

1.3 Research Gap

Most existing studies didn't try different architectures of deep learning for forecasting volatility where it becomes difficult to conclude which architecture is best, studies that compare multiple architectures rarely used same data, same period or evaluation. Sentiment data was explored separately but very few combined with deep learning architectures. Even a few studies applied explainability tools like SHAP and LIME [4]. There is no study with all three elements, comparison of deep learning models with sentiment features and explainability for S&P 500.

1.4 Research Aim and Objectives

The aim is to develop and compare different deep learning models forecasting the next day realised volatility of the S&P 500 index. 1. Training different deep learning architectures on same data from 2008 to 2024, 2. Retrain all models after adding FinBERT Extracted news sentiment features into each model and evaluate all models using evaluation metrics. 3. Then apply explainability techniques to understand what contributes to model prediction and

discuss the impact of sentiment data. Collecting S&P 500 price data and calculating price-based features, technical indicators, realised volatility measures and VIX data and separately collect financial news headlines for sentiment analysis. Build and train deep learning architectures for next day realised volatility forecasting using a 21-day look back window. Then integrating FinBERT extracted sentiment features into each model and retrain, then providing a paired set of results for before and after comparison. All the models evaluated using three consistent metrics. Mean absolute error, root mean squared error and QLIKE – rank the models to find the best model. Finally apply SHAP and LIME techniques on the best model to find key features and amount of contribution of sentiment to forecast.

1.5 Research Question

How sentiment analysis improves the performance of deep learning model and explained by using SHAP and LIME

This study contains 6 chapters; chapter 2 reviews the existing literature on volatility forecasting techniques, deep learning methods, sentiment analysis, and explainable AI then addresses the gaps. Chapter 3 contains a methodology covering data collection, feature engineering, model architectures, integrating sentiment analysis, and evaluation of metrics. Chapter 4 discusses the results, comparing model performance before and after including sentiment and explaining finding through SHAP and LIME. Chapter 5 is a Discussion. Chapter 6 comes with the conclusion of the study by summarising key findings, addressing the main research question, discussing limitations, and recommending directions for future research.

2. Literature Review

2.1 Introduction

In this literature review, volatility forecasting methods of S&P 500 like traditional methods, machine learning methods, and deep learning methods. Additionally, it discussed how sentiment analysis and explainable AI help to increase forecasting performance. The purpose is to explore existing studies and find a research gap to improve volatility forecasting. Machine learning and deep learning techniques use financial data, which is already available, find useful patterns, and predict future values. Sentiment analysis recognises opinions and emotions from textual data like news or social media, and explainable AI brings interpretability among decisions of AI which improves transparency and trustworthiness. It starts by introducing basic ideas and traditional methods followed by machine learning and deep learning techniques. The sentiment analysis and explainable AI, and different ways of evaluations are discussed and proposed a research gap at the end.

2.2 Financial Market Volatility and S&P 500

2.2.1 Definition and nature of volatility:

The change in the volume of a particular asset, stock, or index at a period is known as volatility. If the forecasted volatility is high, then the moment of the Index or stock will be high in either direction. If the forecasted volatility is low, then the moment of the market will be low, so the market will be sideways. Some periods like pandemic, the volatility will be very high and some important news, hearing, wars, shortage of stock may lead to high volatility. Panic selling in the market is one of the major causes of high volatility [5]. Volatility helps to find better opportunities for better profits, can decide how strong the moment will be, we can decide the number of risks in the market at time by using volatility. Traders and investors can calculate their risk and manage their portfolios. Volatility plays a keep role by helping participants to build confidence in making decisions in the market .

2.2.2 Historical Volatility VS Implied Volatility:

Historical volatility is based on the past market price moments; it gives change in realised past prices by calculating the annual standard deviation of historical daily logarithmic return which are additive over time. Taking log return of most recent prices then calculating the standard deviation and annualizing (252 trading days per year) is the formula for calculating Historical volatility [6]. By using the rolling window for recent log returns we can calculate the realised volatility. Calculating realised volatility is easy as it uses historical market data and historical volatility is good at measuring realised risk.

Implied volatility is majorly based on the option pricing; it is not directly calculated from the historical prices. It is implied from the current market prices of the options. There is no formula to calculate Implied volatility; it is a numerical solution of an option pricing model like Black-Scholes which uses current stock price, strike price, time of expiration and interest rates. Newton-Raphson, Bisection, Brent's methods are also included here. Option traders use implied volatility to take positions; they won't just depend on the price and direction: they use implied volatility to find the support and resistance of market for nearest expiry. Traders choose option strategies based on the current implied volatility [7].

2.2.3 Importance of volatility in Finance:

Volatility plays a major role in risk management, traders and investors hedge their positions in high volatile condition to lower their risks. It is the most influenced factor in derivatives and option pricing. In Black-Scholes model, volatility represents the expected change in the Index or stock price on which the premiums depend. Higher volatility may lead to increase in premium of both calls and puts [8]. It also helps to measure the risk; higher volatility means higher risk to investments in portfolio and balancing the portfolio with high and low volatile assets to manage the overall the risk.

Knowing the change in price can assist in measuring the value of risk to take a position in the market. By observing the change in volatility, the market conditions and trends can be assumed to plan trades. Volatility helps investors to choose investments which match their rules in the risk-reward ratio. Volatility helps to make decisions in taking entry and exiting the market with better profits. The most important usage of volatility is in derivatives and option pricing [9].

2.2.4 Characteristics of S&P 500:

The S&P 500 index contains top 500 companies from the United States, it includes different industries like finance, technology, healthcare, services and retail. It shows the overall performance of United States stock market because it holds biggest listed companies in the country. It is also considered as important measure in global stock market. S&P 500 index and stocks in it are actively traded so it has high liquidity in all market conditions . As it holds many large companies it gives better view of the U.S stock market [10]. Moreover, long historical data is available, so S&P 500 is a good choice for Volatility Forecasting Research.

2.2.5 Facts of Financial volatility:

Volatility Cluster: Prices in financial markets often form groups. When there is high volatility, there is more chance of it staying volatile for a while. When prices are calm and stable, they often stay calm. For example, if a company has bad news and its share prices are more volatile, it may continue to be volatile and unstable for some time. Leverage Effect: Bad news creates more volatility than equivalent to good news. If a company's shares fall by 10% because of some fundamental news, then investors become panicked and trade more, and the market becomes more volatile. A 10% rise won't create the same amount of volatility. Fat tails/excess Kurtosis: High-volatility events happen more often than a simple model would expect. A normal bell-curve distribution thinks high-volatility events are very rare, but in financial markets, panic selling and buying happen more often than a model predicts. Mean Reversion of Volatility- Volatility does not stay extremely high or low forever. After a period of chaos, markets often become calmer again [11].

2.3 Traditional Volatility Forecasting Methods

Traditional models are developed to capture the behavior of the market that follows volatility clustering, the leverage effect, which means having more effect from bad news than from good news, and extreme price change happens more often than a normal distribution.

2.3.1 ARCH model:

The ARCH model, the Autoregressive Conditional Heteroscedasticity model, was created by Engle in 1982. It says that volatility is based on changes in recent past prices. If there was a big price change in the recent past, then the ARCH model expects the market to stay volatile for a while. It is the first model which captures volatility clustering and explains that high-volatility markets and calm markets happen in groups [12]. The limitation is that it can't

capture the leverage effect, where bad news creates high volatility more than good news, and it requires many past observations to give accurate predictions and becomes more complicated.

2.3.2 GARCH family:

The GARCH (generalised autoregressive conditional heteroscedasticity) model was introduced by Bollerslav in 1986. It was an improvement of the ARCH model by including past price shocks and past volatility in predicting future volatility. It became the best model for forecasting volatility, as it was simple and worked well with a small amount of recent information. It is good at capturing volatility clustering, but it treats good news and bad news in the same way, which is not true, as financial markets become more volatile for bad news than good news . The standard GARCH model failed to catch this difference [13].

2.3.3 EGARCH (Exponential GARCH):

The EGARCH (Exponential Auto Regressive Conditional Heteroscedasticity) model was introduced by Nelson in 1991. It was an improvement of the GARCH model, which knows that bad news creates more volatility than good news. Additionally, it uses logarithmic variance, so the predicted values will always be positive. Overall, EGARCH is better than GARCH as it can capture volatility clustering and leverage effect [14].

2.3.4 TGARCH (Threshold GARCH):

The TGARCH (Threshold GARCH) is also called GJR GARCH. It was developed by Glosten, Jagannathan, and Runkle in 1993. It was another improvement of the standard GARCH model that helped with the leverage effect. For the leverage effect, it uses a threshold function for the change in price. The threshold was different for positive and negative price change. If the price changes negatively, then the threshold will be higher, and it predicts more volatility in the future. Both GARCH and TGARCH can capture the leverage effect, but TGARCH is directly giving more importance to bad news than good news [15].

2.3.5 Stochastic Volatility models:

Stochastic volatility models are also used for market volatility, where GARCH models use past market price data. SV models explain that volatility can't depend on itself, and it changes over time, which can't be observed directly. It is considered a hidden process. Unlike GARCH models, stochastic volatility models allow volatility to move randomly, so SV models are more realistic because financial markets are unpredictable. ESV models are estimates because they include advanced methods like Bayesian estimation or computer simulations, which is more time-consuming than GARCH models [16].

2.3.6 Strengths and limitations of traditional econometric models:

Our traditional models for financial market volatility are mathematically robust, simple, and easy to evaluate if the results were statistically reliable or not. But these models think that

patterns among variables are simple and fixed; that's why they can't capture the complex patterns that appear in real markets. They also struggle when we include a lot of information, like social media sentiment, news article sentiment, or any other external factors. In addition, one model may not perform well in different market regimes, which means calm periods with less volatility and financial market crisis [17]. These limitations welcomed machine learning and deep learning methods, where they can learn more complex nonlinear relations from the data without depending on fixed mathematical rules and can include additional factors like sentiment from news and social media. These methods can handle large amounts of data and can achieve better predictions. As the financial markets are more complex and data rich, machine learning and deep learning become more popular as they can adapt to different market conditions by making use of more information which traditional econometric models can't handle.

2.4 Machine Learning for Financial Forecasting

2.4.1 Overview of ML approaches in financial forecasting

Unlike traditional econometric models, machine learning models can learn from past data and find complex nonlinear patterns. Machine learning techniques look at past financial data and predict volatility. Different machine learning approaches are; Random Forest: It performs many decision trees to predict and choose the majority prediction as final [18]. It can identify hidden patterns in complex financial data; it can also give important futures for making that prediction. Support vector regression: It is also using historical data to find patterns and forecast future values. Moreover, it is suitable for dataset with less data and contains complicated patterns. So, it is capable of finding both simple and complex patterns. XGBoost: It keeps improving its prediction by learning from past errors, which makes it more powerful. So, it can achieve accurate results by finding complex relations from the training data and simple patterns as well. Light GBM: It is much faster than XGBoost so it is highly useful when both speed and accuracy matter because it can process a large dataset in less time [19].

2.4.2 Comparison with econometric (GARCH family) models:

Although machine learning and GARCH models learn from past data and predict future values, GARCH models use fixed rules and assumptions mainly based on past returns and past volatility, but ML can learn from more complicated data, such as financial information, technical information, sentiment information and more. So, Machine learning can predict well even though market behavior changes suddenly. Many comparing studies stated that ML methods are better than traditional methods [20].

2.4.3 Limitations of Machine Learning:

Machine learning methods have some limitations when they are applied for financial data. When the available data is small and learning patterns which may not repeat in future and end making false predictions. Mainly, ML models treat information like tables, but they should

treat it like a sequence [21]. Which makes ML models harder to understand than traditional econometric models.

2.5 Deep Learning Models for Volatility Forecasting

Artificial Neural Networks (ANN): ANN is a system which is inspired by the human brain, basically it is made up of three layers- input layer which takes the information, hidden layer processes the information and results come from output layer. ANN can learn hidden patterns from raw data without manually selecting the features and without needing every rule to be programmed. But ANN has some limitations which is it can't store past information and fails to understand the sequential patterns [22], so ANN is less used for time-based data.

Convolutional Neural Networks (CNN): CNNs were created for the application of images. It understands the small information from the data, for example, in images it observes the shapes, edges, and uses them to understand bigger things. Afterwards, by using 1D convolutions, CNN is adapted to time-based data like the stock market. CNN is helpful in finding short-term patterns from financial data which also contains long-term patterns [23]. So, CNN can pair with a model which can find long term patterns, so better predictions can be forecasted.

Recurrent Neural Networks (RNN): RNN is a type of AI model, which is designed for sequential data like speech, whether stock market. It has a special thing called hidden state which can carry information from the previous step. So, RNN is suitable for financial data, which is sequential, but it has a problem called vanishing and exploding gradients. Therefore, it can't remember patterns that happened long ago [24].

Long Short-Term Memory (LSTM): LSTM is a model that is designed to fix the older information forgetting problem. LSTM has three special controls that are input gate which decides what new information to store, forget gate which decides what new information to forget and output gate which decides what information to pass to next step. It also has a cell state which stores information that is important from long ago [25]. As financial markets always form volatility clustering, LSTM can capture long-term dependencies from the financial data, It is good at handling long term patterns, but it demands a lot of computational power, needs to adjust many settings, and may not work with unseen patterns [26].

Gated Recurrent Unit (GRU): GRU is a faster model than LSTM, as it uses less gates than LSTM. It trains with fewer parameters on the training data and produces better results. When researchers compared LSTM and GRU, both worked well, but GRU is much faster [27].

Bidirectional LSTM (Bi-LSTM): Bi-LSTM model reads the data from two directions, but normal LSTM reads from past to present where Bi-LSTM reads the sequence forward and backward. It can understand more context from the data as it reads the earlier part and later part of the sequence. Mainly, it is used for speech recognition, language understanding, and performing better than LSTM [28].

Transformer Models: Transformers look at the entire data at a time and find patterns by using an attention mechanism. They assign weights to important parts of data and find connections in the data more easily. The training will be done faster as they can train many parts of the data at the same time. Transformer models are becoming more popular for time series data like financial data. The Temporal Fusion Transformer model uses a self-attention mechanism to train and mainly provides interoperability, which means major contribution of the predictions [29]. PatchTST is a transformer model that creates patches and reads those patches as tokens to a transformer [30]. TFT and PatchTST are two different transformer models that can learn time series financial data.

Hybrid Models: These models contain two different models, where first model learns short-term dependencies and second model learn long term dependencies. Taking the output from the first model and feeding to second can learn more informative patterns from the data. for this hybrid approach traditional GARCH model pairing with LSTM and GRU. Mechanism of EGARCH and TGARCH is also present in second models, so using GARCH as first model will be a correct approach and many studies stated that GARCH-LSTM produced better results [31].

Critical comparison: CNNs capture local patterns efficiently but fail to learn patterns from long back in the window, which allows CNNs to get train on local patterns. Recurrent models directly respond to this by keeping previous information. GRU can learn patterns better than LSTM with less computational resources. Bi-directional variants were greatly used for speech and text. Attention-based models avoid sequential information decay as they treat each data parts separately which works great to capture volatility clustering. Hybrid GARCH models provide statistical information and combines what neural networks learn [32]. Most studies compare models using only a training. Which suggesting 1-2% difference in performance. This makes it hard to reliably rank the deep learning models based on the existing research.

2.6 Sentiment Analysis in Financial Markets

Deep learning models learn patterns from financial prices and making predictions but including sentiment of investors, options of the investors and public will provide much important information to the models. As S&P 500 indexes hold the major listed companies, the market reacts very quickly, and the financial price includes all the available information like (news, reports, events). Investors are not always logical; they become too excited or too scared or just follow the crowd. For example; bad news makes the price decrease [33], then investors may panic and start selling, which drives down more than usual or good news makes the price to increase then investors become more confident and buy which cause the price to rise more than usual. So, knowing the sentiment and understanding the behavior of the market can produce better volatility [34]. News and social media are the major sources where social media is fast but contains a lot of false information, where news is slower but contains direct information which holds true investor options and sentiment of the market's behavior.

Sentiment extraction is a way of finding emotions from text; whether news articles sound positive, negative, or neutral. Lexicon-based methods use a list of words with assigned meanings. VADER and TextBlob identify sentiment by checking words and rules [35]. They are fast and easy to understand but may struggle with complex financial language. FinBERT is a sentiment analysis model which is trained on financial terms, so it can understand the news related to S&P 500. FinBERT categorises the financial text into positive, neutral, and negative categories. FinBERT can capture meaning and context of the word instead of simply counting positive or negative words, so it extracts better sentiment results, but it requires more computational power and time [36]. Adding sentiment information like news, social media opinions or investor emotions can enhance predictions of the volatility when compared with using historical financial prices and technical indicators. Results may not be same always, sentiment-based predictions may vary with source of data. Models become more complex by using deep learning and sentiment analysis; it's difficult to understand the decisions of the models. So explainable AI will help to interpret the decisions of the models.

2.7 Explainable Artificial Intelligence (XAI) in Finance

Models are more complex as they use deep learning and sentiment analysis, and it is hard to understand the decisions. These models are considered black boxes because we can't see how they make decisions. Interpretability means knowing why model make that prediction [37]. In finance it is important to understand the decisions by the models, deep learning models are powerful but difficult to understand and GARCH models are easy to understand but not always powerful. Finance needs a balance between accuracy and understanding, so people can trust the model's prediction and make use of it.

Feature importance is a way to understand the model by ranking the features to make predictions. It is used in tree-based models like random-forest to know the most important features, which is known as intrinsic explainability because the model itself explains. But in the case of complex deep learning models, there are methods called SHAP and LIME to explain decisions, and it is known as post hoc explainability. SHAP (Shapley additive explanation) explains the decision of AI by showing how important each feature is to make the predictions. It is based on Shapley values from game theory that is, taking the contribution of everyone in every possibility to calculate the Shapley values [38]. LIME (local interpretable model agnostic explanation) explains one prediction by approximating one complex model to a simpler interpretable model around that prediction [39]. Attention mechanisms and integrated gradients are also used to explain the decision of AI main for neural networks. All the XAI methods explain how models predict volatility and show factors that affect forecasts such as lagged returns, news sentiment, or important time periods.

2.8 Evaluation Metrics for Volatility Forecasting

Evaluation of the models will be done by using some metrics like accuracy, F1-score, recall, and precision, but here we are dealing with regression problems, so some error-based metrics

are MAE, RMSE, and QLIKE. MAE (Mean Absolute Error): It calculates the average difference between the model predicted values and actual values. Lower MAE value says the model is better on average. RMSE (Root Mean Squared Error): it is like MAE but gives more importance to large error because it squares the difference before averaging. A model with high RMSE value means occasionally making mistakes which are very large, should avoid those models in finance. Qlike: It is an industry standard metric particularly suited for volatility forecasting [40]. It gives more importance to underestimating volatility than overestimating it, which captures real world financial risk.

2.9 Research Gaps and Critical Synthesis

From the literature on deep learning architectures, some studies have reported that LSTM models achieve better results than other models, while others have found that hybrid models perform better than transformer-based models[41] . However, there is no common comparison across different deep learning architectures. Although sentiment information can improve prediction accuracy, earlier sentiment analysis approaches relied on basic word-count methods that fail to capture context and the specific language used in financial texts, rather than using models such as FinBERT. Some studies have also applied explainability techniques to improve model interpretability, but they have not clearly discussed how sentiment information contributes to the models' ability to predict volatility. Therefore, there is a research gap in comparing different deep learning models for forecasting S&P 500 volatility using sentiment analysis, while also applying explainability techniques to understand how sentiment data contributes to the prediction results.

2.10 Summary

This chapter gives information about volatility in the stock market then discusses features of S&P 500, then discusses traditional volatility methods, then how it updated to machine learning and why it moved to deep learning. And then sentiment analysis, explainability and evolution methods helped to judge them models. The main research gap that previous studies didn't compare different deep learning models that include sentiment analysis and explainability in one complete framework for forecasting volatility for S&P 500. This study aims to fill the gap by comparing deep learning models under the same conditions using sentiment analysis and explaining how models make predictions. Next chapter will discuss the methodology used to address this gap. It will describe the data sources, the models used, how sentiment analysis is included, and the methods used to evaluate the results. This will show how the research was carried out.

3. Research Methodology

3.1 Introduction and Research Design

This chapter implements the methodology for the research question, which focuses on comparing different deep learning models under the same conditions for forecasting volatility, incorporating sentiment analysis from news data, evaluating the performance of the models, and understanding which input features drive the model predictions. The study follows a quantitative, comparative experimental approach implemented through a five-stage pipeline.

In the first stage, daily financial data and CBOE volatility (VIX) data for the S&P 500 index are obtained from open-source datasets. In the second stage, the required features are engineered based on the stylised facts of volatility, and fixed-length sequences are created for training the deep learning models. In the third stage, sentiment analysis is extracted from open-source financial news headline data using FinBERT. The sentiment features are then incorporated with the financial data, and the deep learning models are trained using the same hyperparameters to ensure a fair comparison under consistent conditions. Finally, using explainability techniques to understand how each feature drives the predictions of the model, especially sentiment analysis.

3.2 Research Design

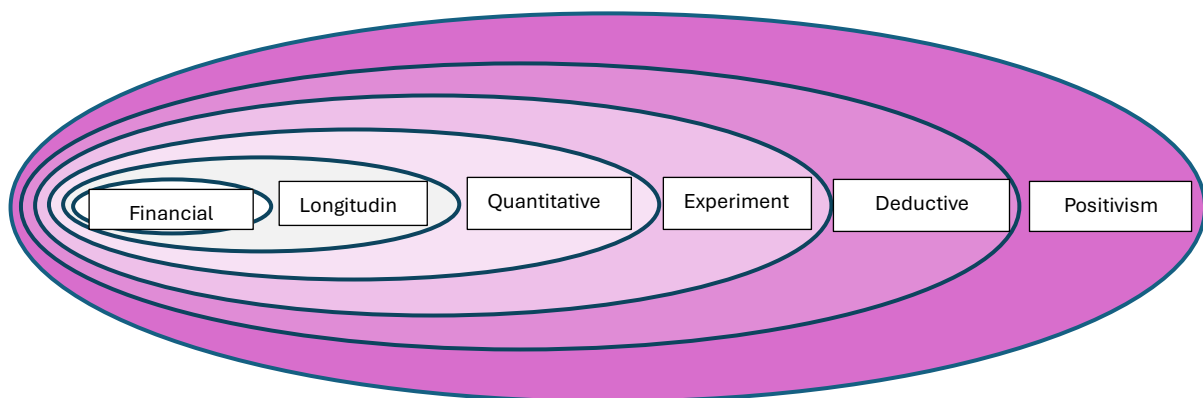


Figure-1: Research Design

This study is a positivist philosophy, as financial data can be calculated by numbers and didn't use any personals to study the financial market. The approach is deductive, as it starts with ideas from previous studies and tests those ideas, for example, sentiment analysis can improve the predictions of models. As it includes comparing models on the same basis, it uses more numbers and statistics, so research is quantitative. And the data used in this study is secondary, as financial data is obtained from the yahoo finance library and sentiment dataset is downloaded from Kaggle. There is no survey conducted in this study for sentiment analysis.

3.3 Data source and collection

This study requires financial data of S&P 500 and sentiment data like news from headlines or social media posts related to S&P 500. Yahoo Finance provides the financial data through yfinance library, values of open, high, low, close and volume by using SPY ticker which contains genuine trader volume and closing prices of VIX is taken which contains volatility from options. Start and end dates based on dates of news dataset. For sentiment data, open-source dataset is downloaded from Kaggle website, “S&P 500 news headlines dataset” contains news headlines related to S&P 500 from 2008 to 2024, which covers major crises like 2008 stock market crash, COVID-19 and 2022 correction driven by inflation. So, comparisons include many high volatile periods.

3.4 Feature Engineering

Open, high, low, close, volume and VIX are already available from the source, and by using those data, some more features are engineered, so models can learn useful patterns to predict volatility. Return based features like log returns tell how big the recent price movements, absolute returns tell same without direction, squared returns highlight the big movements in prices. Negative return flag can be helpful for leverage effect; skewness tells big movements are more likely to be upside or downside. Kurtosis tells how often big movements happen [42]. High-low range, overnight gap holds the information of day range, gaps where average true-range tells the complete movement of the day. Realised volatility is the overall movement of the price for a recent period, which is the target. A simple moving average calculates the average of recent closing prices. Relative strength index is the most popular indicator used to know how strong the movement is, whether the upside or downside. Moving average convergence divergence is short term trend compared to long term trends by using exponential moving average which gives more importance to recent values. Bollinger band is used to know how volatile the market is; more width means high volatile [43]. Unlike realised volatility, Parkinson volatility uses high and low prices instead of closing prices to calculate the volatility [44]. 21 days realised volatility is the target, as 21 trading days in a month.

3.5 Preprocessing the data

Rolling features like realised volatility 63 column contains 63 null values because it needs 63 past values to construct, so 63 rows were removed. In this process, many other null values from rolling features were removed. By exploratory data analysis, univariate summary statistics and distribution plot were produced for all engineered features to observe fat tails. Time series plots of returns and realised volatility were observed for confirming volatility clustering. Correlation between features was observed using heatmap. Realised volatility and vix were plotted to observe how they coincide. Train and test splits were partitioned at the 80:20 ratio. Scaling is done by using min max scalar on training data and used the rules on the test data. The scaling rules were saved, so model predictions can convert back to normal values. Then sequences were created to feed three-dimensional data into deep learning

models with a lookback window of 21 trading days [45]. So, to predict volatility at day- t , models take days $t-21$ to $t-1$, then tensor will be (samples, 21, 26) for predictors and (sample, 1) for target. 21 observations were lost while splitting, and lookback windows were constructed after splitting by ensuring no test data contain in training sequence.

3.6 Model Architectures

Nine different architectures were implemented with identical input tensor of shape (21,26), a single-unit linear output layer, mean squared error loss, and the Adam optimizer, so differences in performance are based on architecture and hyperparameters.

3.6.1 Convolutional Neural Network (CNN)

A one-dimension Convolutional Neural Network was developed to predict volatility by taking sequential input by processing through 3 convolutional blocks. The first block contains 2 conv1D layers with a filter size of 32, a kernel size of 5, ReLU activation, and same padding. Each layer is followed by batch normalisation to stabilize and fasten the training process. A maxpooling1D layer is used to reduce sequence dimensionality [46], and then a dropout layer is used with 0.10 rate to avoid overfitting. The second block contains 2 conv1D layers with a filter size of 64, so network can learn complex patterns and kernel size of 5, ReLU activation, same padding for both layers followed by batch normalisation and maxpooling1D, and higher dropout rate with 0.20. The third block contains a conv1D with 128 filter size and a higher dropout rate of 0.30. Graduale increase in dropout rates helps avoid overfitting . Then, global average pooling is used to compress the sequential features and followed by a fully connected layer with 32 neurons, ReLU activation, and dropout rate of 0.40. Finally, a dense layer with a single neuron produces the output of the network [23]. The CNN model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.6.2 Gated Recurrent Unit (GRU)

A Gated Recurrent Unit neural network was developed for forecasting volatility; it is a recurrent neural network which can capture temporal patterns from sequential data with less parameters. The first GRU layer contains 32 units and returns sequences = true, which allows the complete sequence of hidden states to be passed to the next GRU layer. Then batch normalisation is applied to stabilise and fasten training, and dropout with 0.25 is used to avoid overfitting. The second layer is also similar but with 64 units which allow the network to

capture complex patterns and return sequences is true, so sequential information is passed to the next GRU layer. Then batch normalisation to compress the sequential information and dropout rate with 0.30 were used for more generalisation. The third GRU layer with 128 units was used and return sequences is false which means only final output is passed to next layer which means recurrent layer summaries the sequence into a single representation for the regression layer [47]. GRU architecture gradually increases capacity from 32 to 128 units, which enables networks to learn increasingly complex patterns where batch normalisation and dropout brings stability and generalisation [48]. The GRU model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckpoint saved the model with the best validation performance.

3.6.3 Long-Short Term Memory (LSTM)

A Long Short-Term Memory neural network was developed for forecasting volatility; it is a special neural network which can learn long-term relations through their internal memory mechanisms which makes them suitable for capturing patterns and relation from sequential data. The first LSTM layer with 128 units, return sequences true and dropout rate of 0.25. This configuration learns more patterns then output information at every time step and drops 0.25 of data. The second layer with 64 units and return sequences is true. Here, keeping the sequence output, this layer again processes and improves the temporal patterns learned from the first layer. The third layer with 32 units and return sequences is false, which means only the hidden representations were passed to the next layer, which combines the learned time-based information into a fixed size representation that can be used for final prediction and dropout with 0.40 is also applied. Then a fully connected dense layer with 32 neurons and ReLU activation is used as a regression head. It processes all the patterns and relations from features and target variables. Before the final prediction, a high dropout rate of 0.50 was applied to avoid overfitting. The LSTM model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric [49]. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckpoint saved the model with the best validation performance.

3.6.4 Bidirectional-LSTM

Bidirectional LSTM model takes the input shape and process it by two Bi-LSTM layers first one contains 128 neurons with return sequence true and second one contains 64 neurons with return sequence false, each layer was followed by dropout layer of 0.20 rate. Then three dense layers with 64 neurons, 32 neurons and 16 neurons with relu activation and 0.50 rate of dropout layer. Finally, an output layer. This Bi-LSTM architecture can learn patterns from the sequence in both directions to make predictions [50]. The Bi-LSTM model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.6.5 Patch Time Series Transformer (PatchTST)

PatchTST model designed to take the input shape and process by one-dimensional convolutional layer with 64 neurons and a kernel size of 4 was applied to capture local relations and create useful patterns from nearby observations. The resulting observation is passed to a transformer encoder which uses multi-head self-attention with two attention heads and a key dimension of 32. So, model can learn relations between different positions in the sequence. A residual connection was used to add original output to attention output, then layer normalisation is used for more stability. A feed forward network with two dense layers with 64 relu units, then a layer matching the input feature dimension, then processes the representation. another residual connection and layer normalisation complete the encoder block [51]. Finally, the encoded features are flattened into a single vector and passed through a 32-unit relu dense layer then an output layer with a single neuron produces the prediction. The PatchTST model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 100 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.6.6 Temporal Fusion Transformer (TFT)

Temporal Fusion Transformer architecture combines recurrent processing, gated residual networks, and self-attention for prediction. After getting input shape to input layer, LSTM layer

with 128 neurons, return sequence and 0.20 dropout is used to capture temporal patterns while avoiding overfitting. Now, gated residual network processes the LSTM output using two dense layers with 128 units. The transformed representation was controlled by a separate gating mechanism which is then combined to the original input by a residual connection. Then layer normalisation is used to get more stability while training. Next, a multi-head self-attention layer with four heads and a key dimension of 32 with 0.20 dropout was applied to capture meaningful patterns from the different time steps. Another residual connection and normalisation were applied. Then a second GRN repeats the gated transformation and residual structure [52]. Finally, global average pooling converts the time series features into a vector. Then regression head with two dense layers with 32 and 16 relu units, 0.20 rate of dropout between them, and finally a single neuron output layer predicts the final output. The TFT model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.6.7 Hybrid CNN-LSTM

This architecture combines convolutional and recurrent layers to capture both local patterns and long-term time series relations from sequential data. The model takes the input then two one-dimensional convolutional layers with 64 filters and a kernel size of 2, same padding and relu activation; these layers learn local temporal dependencies [53]. A dropout rate of 0.20 is applied after the convolutional block to avoid overfitting. The extracted patterns then passed to an LSTM block; first LSTM layer contains 64 units and return sequences, allowing the next LSTM layer to process the complete temporal features. The second LSTM layer with 32 units return only the final output, which is a low-dimensional representation of learned features and 0.20 rate of dropout layer was used for regularization. Finally, a fully connected dense layer with 64 relu activated neurons with a 0.40 dropout layer and then a single neuron dense output layer generates the final output [54]. The CNN-LSTM model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.6.8 Hybrid GARCH-LSTM

This is a hybrid architecture which combines traditional volatility models with deep learning to improve volatility predictions. Initially, daily log returns are scaled by 100 and a GARCH (1,1) model with constant mean and normal distribution is fitted using training data from 2008 to 2020. Those fitted GARCH parameters were applied to full return series to generate annualized conditional volatility. This volatility is added as an additional feature to the training and testing data. GARCH residual targets and features were separately scaled using the Minmax Scalar. Then sequences were created by using look back period (21) which allows LSTM to learn time series patterns. Then neural networks with two LSTM layers with 128 and 64 units where first returns sequences to second layer and second layer pass final representation. Both LSTM layers used 0.02 dropouts for regularization. The regression head contains three dense layers with 64, 32, 16 relu neurons, then output layer with a single neuron producing prediction [31]. The GARCH-LSTM model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.6.9 Hybrid GARCH-GRU

This architecture combines GARCH volatility information with a Gated Recurrent Unit (GRU) network to find time series relation from financial temporal data. The model takes input shape and process by GRU layer containing 128 units and returns sequences, enabling the network to maintain sequential information for next layer and dropout rate of 0.2 which avoids overfitting. Then, the sequence is passed to a second GRU layer containing 32 units, but this layer only returns the final form, which is a compact summary of the temporal patterns. Then a 0.2 drop out layer was used for regularization. Finally, the information was passed to fully connected dense layers with 32 and 16 relu activated neurons [55]. These layers again refine the learned features before producing output. Then, a single neuron dense output layer gives prediction [56]. The GARCH-GRU model was compiled with Adam optimiser with learning rate, mean squared error as loss function, and mean absolute error is an evaluation metric. Then the model was trained for 30 epochs with a batch size of 32 and 0.2 validation split. Three callbacks were used during the training- EarlyStopping, ReduceLROnPlateau and ModelCheckpoint. EarlyStopping stop training when validation loss does not improve for 8 consecutive epochs and restores the model weights from best validation loss epoch [57]. ReduceLROnPlateau reduced the learning rate by a factor of 0.5 when the validation loss didn't improve for 3 consecutive epochs. ModelCheckPoint saved the model with the best validation performance.

3.7 Sentiment Integration

3.7.1 News Dataset Loading and EDA:

The news dataset is loaded into the notebook by mounting Google Drive into Google Colab by using drive integration. The file path and required column names were defined, and headlines were stored in the title column, and dates were stored in date column. Initial exploratory data analysis was performed to understand the dataset. The dataset shape was examined to determine the number of observations and variables, by using the `tail()`, `info()` and `describe()` functions we know information about records, datatypes and statistical characteristics; missing values also found. The date column is converted into a standard datetime format, where invalid entries are converted to missing values. Then, check the data by displaying the total number of headlines, unique dates, date range, missing headlines, and duplicate headlines for further processing.

3.7.2 News Data Cleaning and FinBERT Sentiment Analysis:

The news dataset was cleaned to improve data quality before sentiment analysis. Rows with missing headlines or dates were removed, and duplicate headline date entries are removed to avoid redundant data. A light text cleaning function applied to every headline to clean URLs, tags and whitespaces. After cleaning, empty headlines were also deleted and displayed the ready dataset for sentiment analysis. For sentiment classification, FinBERT which is a pre-trained model developed ideally for financial text, was loaded from a hugging face transformers library. The tokenizer converts each headline into a model suitable format and model is transferred to available GPU or CPU and set to evaluation mode. The `explain_finbert_prediction()` function tokenizes each headline and gets its model output without calculating gradients. A SoftMax function converts the output logits into probabilities for the available sentiment classes. The sentiment with the highest probability is selected as predefined classification. It prints a headline with sentiment labels and its probability scores, and final predicted sentiment classification of FinBERT model for one example news headline.

3.7.3 Applying FinBERT Sentiment Analysis to the Full News Dataset:

The model was applied to the completely cleaned news dataset. A text- classification pipeline from the hugging face transformers library was created using the previously loaded tokenizer [58]. It was configured to use the available GPU when CUDA is available; else CPU was used. Here, the cleaned text was converted into a list of batch processing. The `truncation=True` and `max_length = 128` parameters were used to ensure that headlines with more than input length were truncated, and headlines were processed in batches of 64 to reduce memory size and improve processing efficiency. FinBERT produced a probability score for three sentiment classes for every headline. The scores were stored as `prob_positive`, `prob_negative`, `prob_neutral` columns in the news dataset. Probability values show the confidence that each headline belongs to the sentiment category. Then the final sentiment label was assigned to each headline by comparing three probability scores. The sentiment score with highest

probability is selected as the predicted sentiment of that headline. Finally, the distribution of the predicted sentiment categories was visually represented by bar chart.

3.7.4 News Sentiment Score and Daily Aggregation:

From the positive, negative, and neutral probabilities from FinBERT, net sentiment was created by subtracting the probability of negative sentiment from the probability of positive sentiment for every headline. This gives a range of value from -1 to $+1$, which means higher value indicates strongly positive sentiment, and lower value indicates strongly negative sentiment and values near to 0 indicate neutral sentiment [59]. To understand how sentiment was distributed with headlines across the dataset, histogram was used to represent visual distribution, and descriptive statistics were also observed to get more clarity about count, mean, minimum and maximum values. As the financial dataset was organized on a daily basis, the headline sentiment scores were aggregated by date. In the case of multiple headlines per day, the mean sentiment score was calculated to produce a daily sentiment measure. Total number of headlines for each day was calculated and stored in headline count. This gives daily measure of overall sentiment in the news.

3.7.5 Checking Trading days and News Coverage:

To check the overlap between financial data and sentiment data, unique dates from financial price dataset and dates from news were normalized and number of trading days and number of days with new and number of overlapping dates were calculated. Then coverage of news data across trading days was calculated as percentage. Especially, test period coverage was examined by using split data, because test period is more important for evaluation relation between sentiment and financial market data. This important to observe whether the news observations were sufficient to the corresponding trading days before merging.

3.7.6 Merging News Sentiment with Financial Data:

Now, the daily sentiment data was merged with the financial price data using data as the index. Financial data was copied into `df_sent`, and both datasets were normalized, so their dates were in the same format. After merging the datasets, missing daily sentiment values was checked, `has_news` variable was created to separate days with news available, and no news recorded and take values as 1 and 0 respectively. For days without available news, the daily sentiment values were forward filled using the previously available sentiment values. If no daily sentiment hits at the beginning and no previous value, then those observations were replaced by zero [60]. And headline counts for days without news, also replaced by zero. Then a five-day moving average of daily sentiment was calculated and stored in `sentiment_ma5`. Rolling measure smooths short term fluctuations in daily news sentiment and provides a more stable representation of the recent sentiment trend [61]. For initial observations, when complete five days were not available, backward filling was used. Finally, `df_sent` dataset contains financial features with daily news sentiment. This dataset is used for training models and observing how it improves the performance of models.

3.7.7 Sentiment-Based Model Preparation:

After adding sentiment features to financial features, the data was split into training and testing sets using 1 January 2021 as split date. The features and target variable were scaled using minmaxscaler, and then 21 days sequences were created by using lookback period [62]. For hybrid models like GARCH-LSTM and GARCH-GRU require an extra feature which is GARCH volatility. Instead of predicting realised volatility directly, hybrid models predict residual volatility, which is difference from GARCH volatility and realised volatility. Then the residuals and input features were scaled and created sequences for model training and testing [63]. All the models were retrained with sentiment integrated data and tested with evaluation metrics.

3.8 Evaluation of the model:

To evaluate the forecasting performance of each model, three evaluation metrics are used: mean absolute error, root mean squared error, and QLIKE. Each metric captures a different part of prediction performance; that's why these three metrics are used instead of depending on a single metric. After training the models on training data and making predictions on test data, the actual values and predicted values were evaluated using these metrics. But during, all the models use scaled values which are between 0 to 1. These scaled values were not appropriate for evaluation. So, before calculating MAE, RMSE, and QLIKE, the predicted and actual volatility values were converted back to their original values using MinMaxScaler fitted on the training data [64]. All the models were trained on the test dataset. This data was not used during training and validation. For the standard deep learning models, predictions are made directly on the test data. But GARCH-LSTM and GARCH-GRU first predict the residual component, and then residuals add to the GARCH volatility forecast to predict the final volatility. While the models use different methods for predictions, they were all evaluated against the same actual Next_Day_RV_21 values from the test period. This makes sure all the models before and after sentiment were compared fairly [65].

3.9 Model Comparison

Each model was trained twice, first time using only financial market prices and technical features, and second time retrained with two extra sentiment features added. Everything else was the same as different – same train/test split, the same MinMaxScaler, same lookback window (21 days) and same callbacks with earlystopping, learning rate reduction and model checkpointing. So only input features change and any change in performance can be fairly linked to sentiment and not other settings. Improvement is measured as difference between before and after was divided by before and multiplied by 100. A positive percentage tells that error was decreased after integrating sentiment, which means model improved its performance by using sentiment. Compared by a single metric to decide the best model is not fair, because each metric measures different kinds of errors. Each model was ranked individually for MAE, RMSE, and QLIKE with rank 1 given to the model with the lowest error [66]. Then the overall score was calculated by adding three ranks. A lower overall score means

the model performed well across all three measures. This makes sure that a model was not considered best just because it performs very well on one metric but poorly on the other two.

3.10 Explainability:

Deep learning models are treated as black boxes, which means they do predictions but can't explain why. Even strong evaluation metric confirms a model was predicting well they do not reveal the factors that model is using to make prediction. To address this, two explainable techniques were applied. SHAP tells which feature was most important from the data to make prediction [67]. LIME explains the local view what influenced the prediction more.

SHAP explains how model make predictions by using game theory, it treats features as players and give value to each feature according to contribution to make predictions, if the value is high holds strong influence in prediction where low shape value or near to zero holds less influence. The gradient explainer variant of SHAP was used to calculate SHAP values as models were already trained, so there is no need to test every combination of features to create a reference point, 50 samples are randomly chosen from the test data. SHAP values are then calculated for another 50 test samples. This gives a value for each sample, time step, and feature. To find the overall importance of each feature, the absolute SHAP values are averaged across all 50 samples and all 21-time steps [68]. The SHAP values for the two sentiment features, `daily_sentiment` and `sentiment_ma5`, were combined to calculate their share of the model's total feature importance.

LIME makes small changes in the input, tests the model, and observes how predictions change. Then LIME uses a simple linear model to study these results and identify which feature has more effect [69]. It doesn't need how the model works internally; it uses the model's input and outputs. Since the models use input data with 21-time steps and 28 features, the data is first converted into a single list of 588 values before being given to LIME. Each value is given a name, so it is clear which feature and time step it belongs to. A wrapper function then changes the data back into its original 21×28 shape before sending it to the model. This happens in the background, so LIME does not need to know about it. For each explanation, LIME creates 500 slightly modified versions of the input. This provides a good balance between reliable results [70]. To calculate the importance of sentiment, the explanations were collected from 20 test samples and combined.

4. Results and Analysis

This chapter discusses the results of all the deep learning models developed in this study. The results are organised into three parts. The first part evaluates the nine base models trained using only market and technical features. The second part examines the same nine architectures retrained with news sentiment features added, assessing whether sentiment

improves forecasting performance. The third part presents the explainability analysis using SHAP and LIME, revealing which features affect the predictions.

4.1 Model's performance without Sentiment:

	Model	MAE	RMSE	QLIKE
0	CNN	0.047632	0.063190	0.438225
1	GRU	0.023624	0.033975	0.045816
2	LSTM	0.017855	0.024050	0.039493
3	Bi-LSTM	0.019281	0.024755	0.057786
4	PatchTST	0.016398	0.022344	0.036804
5	TFT	0.016607	0.022675	0.038387
6	CNN-LSTM	0.019734	0.023906	0.063701
7	GARCH-LSTM	0.030800	0.033816	0.282385
8	GARCH-GRU	0.032348	0.035925	0.426776

Figure-2 evaluation metrics of non-sentiment models

Individual model results:

The CNN uses filters to find local patterns and relations in the 21-day input data. This works well in the short term, but it struggles to remember information from the long term as it doesn't have any recurrent or attention mechanism. As a result, it was the worst performing model with an MAE of 0.0476 and QLIKE of 0.4382. The GRU was much better because has gates which allow it to learn sequence efficiently and achieve better MAE than CNN but still weaker than advanced recurrent models. The LSTM performed much better as it has dedicated cell state to remember important information from long periods. So, it learned patterns from long term and achieved MAE of 0.0179 and QLIKE of 0.0395. The Bi-LSTM processes the sequence in both forward and backward directions. However, this did not improve performance in this study. It was slightly worse than the standard LSTM on all three measures.

Best Base Model:

PatchTST was the best baseline model. It splits the time series into small patches and uses self-attention to compare these patches. This allows the model to find important patterns that may be far by several days. PatchTST achieved the best results on all three metrics. The TFT was a very close second. It uses attention with other mechanisms that help the model decide which information is important. Its MAE was 0.0166. This tells that attention-based models are particularly effective for forecasting volatility. The CNN-LSTM combines CNNs for finding local patterns with an LSTM for learning patterns over time. It performed reasonably well for RMSE, achieving 0.0239, but its QLIKE score was 0.0637. The two GARCH hybrid models

performed poorly compared with the other deep learning models. Their QLIKE scores were much higher than those of the attention-based models.

PatchTST was the best overall model. It achieved the lowest MAE, RMSE, and QLIKE, so its performance was not dependent on just one evaluation measure. One reason for its success is the way it divides data into patches. It doesn't treat every day separately; it groups nearby days together. Then, use self-attention to compare different patches. This means that information from a volatility period several days earlier can directly influence prediction. Recurrent models such as LSTM must pass information through a sequence of steps, which can make older information weaker over time. This is useful because volatility often shows persistence and clustering. In other words, periods of high volatility tend to be followed by higher volatility, and low-volatility periods tend to continue as well.

The hybrids follow a residual learning design; GARCH (1,1) is fitted to percentage log returns, and its parameters were then fixed and applied to the full sample to generate conditional volatility, then it was added as a 27th input feature. The neural network is not trained to predict volatility directly but to predict the residuals which means the difference of realised volatility that GARCH leaves unexplained, with the final forecast reconstructed by adding the two components. Here both GARCH-LSTM and GARCH-GRU predicted the volatility but comparatively worse than recurrent and attention-based architectures.

4.2 Impact of Sentiment Analysis:

Two sentiment variables were added to the original 26-features input set. Financial news headlines were scored with FinBERT, a transformer pre-trained on financial text and fine-tuned for three-class sentiment classification. Each headline received a net score as $P(\text{positive}) - P(\text{negative})$, between -1 and $+1$, and these were averaged by trading day to produce `daily_sentiment`. A five-day rolling mean, `sentiment_ma5`, was then computed to smooth day-to-day noise and capture the prevailing weekly tone of market news. This expanded the feature set to 28 (29 for the GARCH hybrid models, which also carry conditional volatility). All nine architectures were retrained from scratch on the identical train/test split, lookback window and hyperparameters, so any difference in performance is attributable solely to the sentiment inputs.

4.2.1 Model comparison before and after sentiment:

	Model	MAE Before	MAE After	RMSE Before	RMSE After	QLIKE Before	QLIKE After
0	CNN	0.047632	0.021728	0.063190	0.028964	0.438225	0.062878
1	GRU	0.023624	0.022477	0.033975	0.033524	0.045816	0.043536
2	LSTM	0.017855	0.017814	0.024050	0.023894	0.039493	0.040527
3	Bi-LSTM	0.019281	0.021713	0.024755	0.030948	0.057786	0.044387
4	PatchTST	0.016398	0.013997	0.022344	0.019210	0.036804	0.032972
5	TFT	0.016607	0.023676	0.022675	0.028628	0.038387	0.061432
6	CNN-LSTM	0.019734	0.039249	0.023906	0.052558	0.063701	0.095859
7	GARCH-LSTM	0.030800	0.036877	0.033816	0.040792	0.282385	0.604968
8	GARCH-GRU	0.032348	0.040224	0.035925	0.043343	0.426776	0.801227

Figure-3 Evaluation metrics of models before and after sentiment

The CNN improved significantly in all three metrics when sentiment features were added. MAE and RMSE decreased by 54% while QLIKE decreased by 85%. This means sentiment information provided long term information which CNN missed to learn. the GRU improved small but consistent in MAE, RMSE and QLIKE. the LSTM was unchanged but small improvement in MAE and RMSE, small decrement in QLIKE. This tells that sentiment information was already learned by the LSTM by its own mechanism. the Bi-LSTM gave a mixed result, that MAE and RMSE worsened but QLIKE improved by 23%. PatchTST was the only model that showed a clear and strong improvement across all three performance measures. Even though it was already performing very well, adding sentiment improved its results further: MAE improved by 14.6%, RMSE by 14.0%, and QLIKE by 10.4%. TFT performance got worse on all three measures, with QLIKE becoming 60% worse. CNN-LSTM face the big drop in performance; MAE is 98% worse. This tells that model learned noise from the sentiment data but not useful patterns. The CNN may pick up this noise and pass to LSTM. GARCH-LSTM and GARCH-GRU also became worse after adding sentiment data.

4.2.2 Overall Impact of Sentiment:

Only 3 out of the 9 models improved across all three measures. One of these was the CNN, which had started with particularly poor performance. Therefore, the main conclusion is that FinBERT-based news sentiment does not generally improve volatility forecasting. In fact, when looking at all the models together, sentiment was more likely to hurt performance than improve it. However, sentiment did clearly help PatchTST, which was already the best model. This is important in practice because you would normally use the best-performing model, rather than choosing a model based on the average performance of all models. The different results across models are also important. PatchTST uses attention, so it can decide when sentiment is useful and when it should be ignored. This allows it to filter out some of the noise

in the sentiment data. Finally, the sentiment data itself may provide only limited new information. Sentiment is carried forward on days when there is no new news and is relatively smooth over time. This means that much of the information it contains may already be captured by the existing volatility variables. This tells that sentiment did not help most models, but it helped the strongest model, PatchTST. This suggests that sentiment can be useful when the model is good enough to identify and use the parts of the sentiment signal that matter.

4.2.3 Best Performing Sentiment Model:

From all the models with and without sentiment information, PatchTST with sentiment ranked first, with MAE 0.013997, RMSE 0.019210 and QLIKE 0.032972, achieved the first rank among three metrics. The base patchTST was the second. Compared to the best base model, the improvement was real but moderate; MAE and RMSE gained 14% and QLIKE is 10% improvement. So, sentiment added useful information only when the model has the capacity to weight it selectively. We can say model architecture is more important than simply adding features to the input data.

4.3 Model Ranking and Overall Comparison:

	Model	MAE	RMSE	QLIKE	MAE Rank	RMSE Rank	QLIKE Rank	Overall Rank
0	PatchTST_Sentiment	0.013997	0.019210	0.032972	1.0	1.0	1.0	3.0
1	PatchTST	0.016398	0.022344	0.036804	2.0	2.0	2.0	6.0
2	TFT	0.016607	0.022675	0.038387	3.0	3.0	3.0	9.0
3	LSTM_Sentiment	0.017814	0.023894	0.040527	4.0	4.0	5.0	13.0
4	LSTM	0.017855	0.024050	0.039493	5.0	6.0	4.0	15.0
5	Bi-LSTM	0.019281	0.024755	0.057786	6.0	7.0	9.0	22.0
6	CNN-LSTM	0.019734	0.023906	0.063701	7.0	5.0	12.0	24.0
7	Bi-LSTM_Sentiment	0.021713	0.030948	0.044387	8.0	10.0	7.0	25.0
8	GRU_Sentiment	0.022477	0.033524	0.043536	10.0	11.0	6.0	27.0
9	CNN_Sentiment	0.021728	0.028964	0.062878	9.0	9.0	11.0	29.0
10	TFT_Sentiment	0.023676	0.028628	0.061432	12.0	8.0	10.0	30.0
11	GRU	0.023624	0.033975	0.045816	11.0	13.0	8.0	32.0
12	GARCH-LSTM	0.030800	0.033816	0.282385	13.0	12.0	14.0	39.0
13	GARCH-GRU	0.032348	0.035925	0.426776	14.0	14.0	15.0	43.0
14	CNN-LSTM_Sentiment	0.039249	0.052558	0.095859	16.0	17.0	13.0	46.0
15	GARCH-LSTM_Sentiment	0.036877	0.040792	0.604968	15.0	15.0	17.0	47.0
16	GARCH-GRU_Sentiment	0.040224	0.043343	0.801227	17.0	16.0	18.0	51.0
17	CNN	0.047632	0.063190	0.438225	18.0	18.0	16.0	52.0

Figure-4 All 18 models

In the overall model comparison, attention-based models performed best then followed by LSTM models. The GARCH hybrid models and the basic CNN performed the worst. Adding sentiment did not automatically make a model better. Only three of the top nine models included sentiment. While the sentiment models of TFT, CNN-LSTM, and both GARCH models were in the lower half of the rankings. Overall, the results suggest that the model architecture is more important than adding sentiment features.

4.3.1 Best overall model

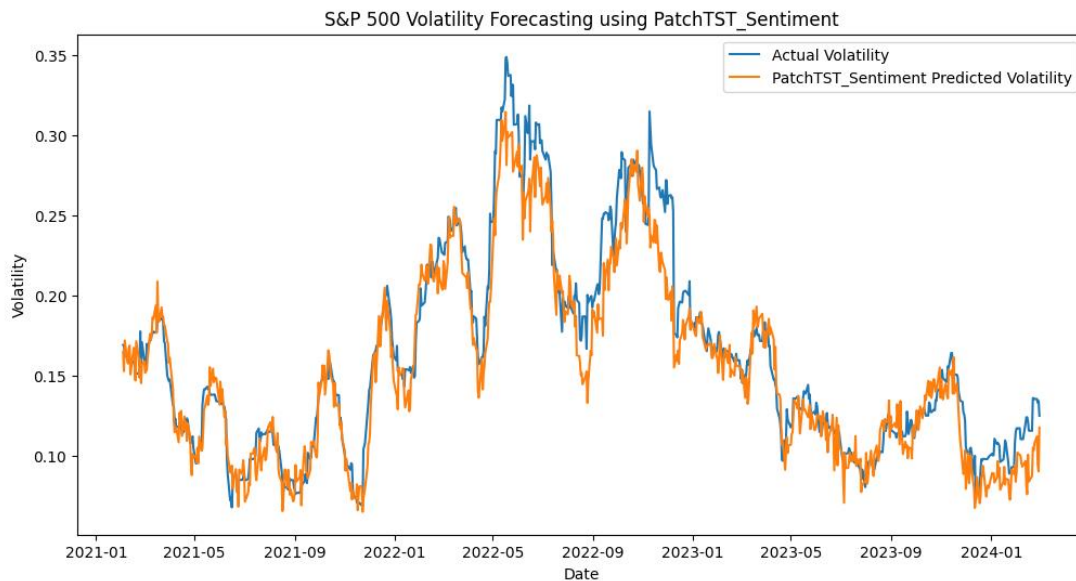


Figure-5 Actual and patchTST predicted volatility (with sentiment) comparison

PatchTST with sentiment was the best model out of all models. It achieved the lowest MAE (0.013997), RMSE (0.19210) and QLIKE (0.032972), giving it the best possible overall rank. Its strong performance is mainly because it combines patching and attention. Patching groups nearby trading days together, which helps reduce daily noise while keeping important short-term patterns. The attention mechanism then allows the model to look at different parts of the sequence and remember important periods of high or low volatility. It can also use sentiment when it is helpful and ignore it when it is not, which explains why patchTST benefited from sentiment while many other models became worse.

All the three metrics agreed for the best models, that patchTST_sentiment, patchTST and TFT ranked 1st, 2nd, and 3rd on MAE, RMSE and QLIKE. This is important as each metric calculates errors in a different way. Some models get different ranks within metrics. CNN-LSTM ranked 5th on RMSE but 12th on QLIKE. This tells that it can make reasonably accurate predictions on average, but it is less reliable when estimated volatility particularly during high volatility periods. GRU_Sentiment showed the opposite pattern: it ranked better on QLIKE than on MAE and RMSE, meaning its daily predictions were less precise, but it was better at avoiding serious underestimation of volatility. Overall, for risk management, QLIKE is particularly important because underestimating volatility can be more dangerous than overestimating it.

4.4 Explainability Results:

Both explainability methods were applied to the best-performing model, PatchTST_Sentiment, using its full 28-feature input over the 21-day lookback window. SHAP values were produced with a Gradient Explainer using 50 randomly drawn background samples and 50 explained samples producing an array of shape (50,21,28). LIME was applied to the same model after flattening each sequence into 588 time-stamped features that are 28 * 21.

4.4.1 SHAP results:

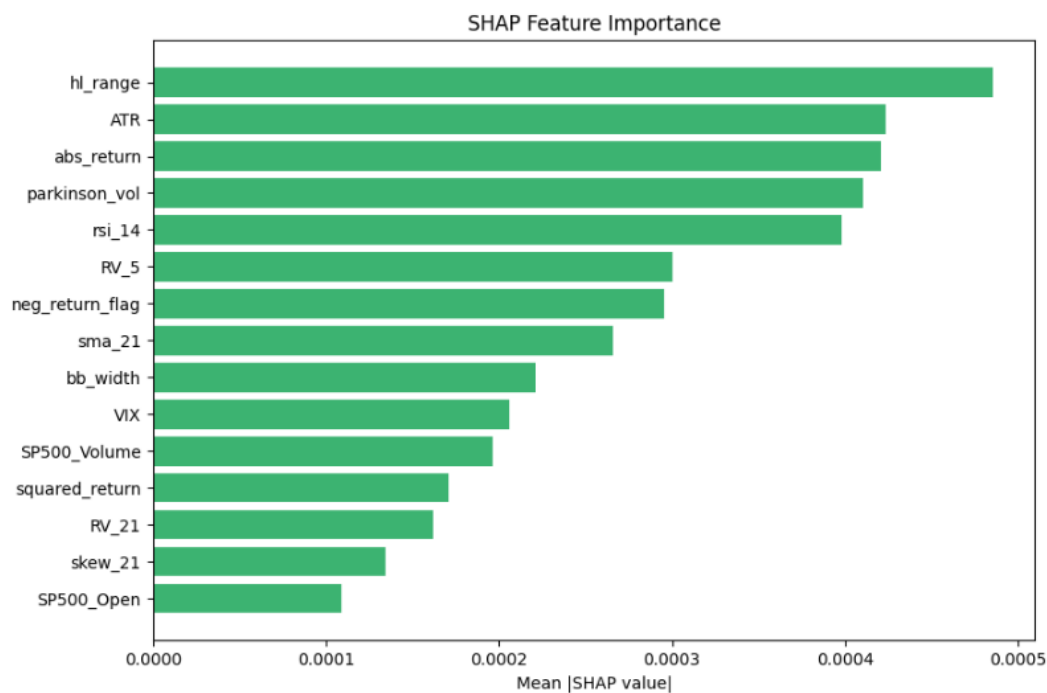


Figure-6 SHAP features importance

Global importance was computed as the mean absolute SHAP value for each feature, averaged across all samples and all timesteps. The top five features are hl_range, ATR, abs_return, parkinson_vol and rsi_14. These are mainly measuring how much prices are moving or how volatile the market is, rather than whether prices are going up or down. This means the model is mainly using information about price movement and volatility to make predictions. The Parkinson volatility and high-low range are especially useful measures of volatility, so their high importance suggests that the model is learning sensible financial patterns. RV_5 is the sixth most important feature, showing that volatility from the last few days is useful for predicting future volatility. neg_return_flag is seventh and captures the leverage effect, meaning that negative returns can lead to higher future volatility than positive returns of the same size.

Share of Model Attention Given to Sentiment (SHAP)

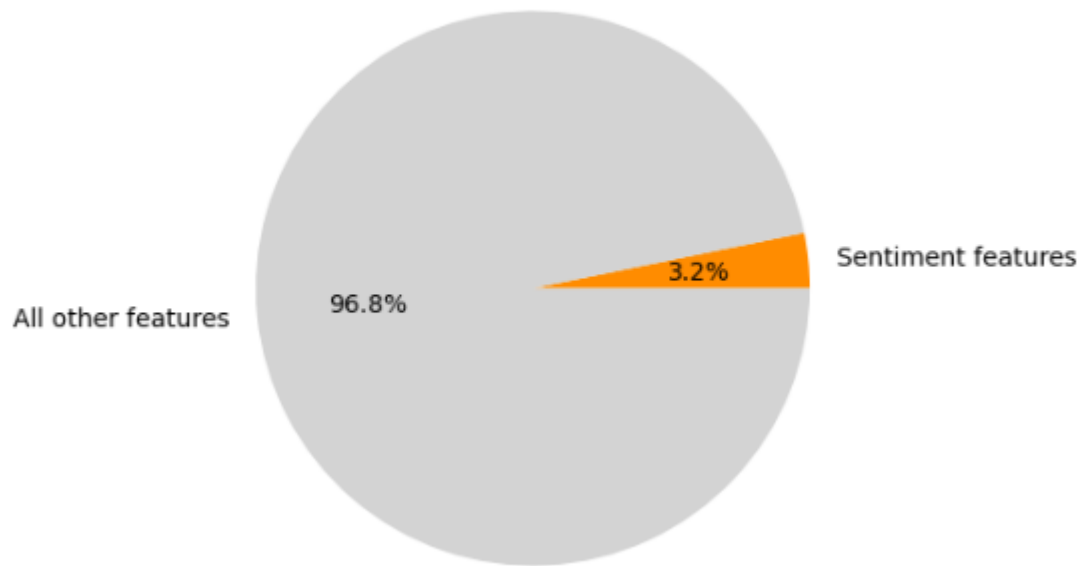


Figure-7 Share of sentiment features

Two sentiment features were not in the top 10, daily sentiment and sentiment_ma5 has 3.21% of model total SHAP importance which is less than average of 28 features (3.57%). It must be more than 7.14% for two sentiment features. It means the model depends on prices and volatility information. Sentiment does help but only small additional information.

4.4.2 LIME results:

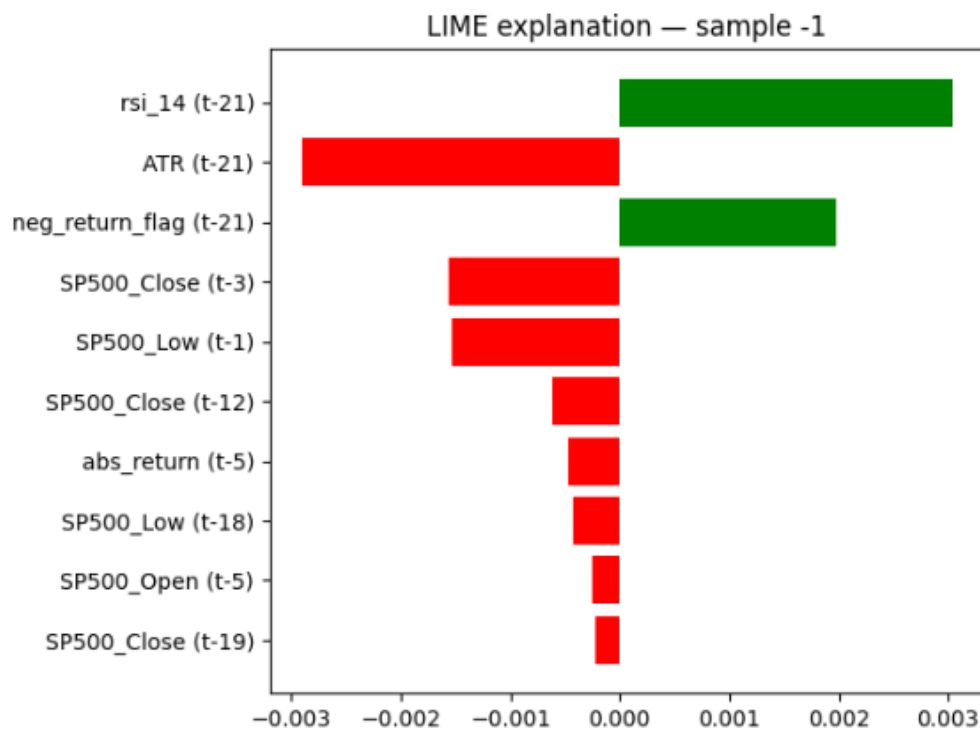


Figure- 8 LIME explanation of one prediction

This forecast was mainly affected by information from 21 days ago instead of the most recent data. The three most important features were RSI_14, ATR, and negative return flag. RSI and the negative return indicator increased the predicted volatility, while ATR reduced the prediction. Recent price information had a smaller effect and reduced prediction. No sentiment feature was in the top 10 important features for this forecast. LIME showed sentiment contributed about 5.96% of the total model explanation. This is lower than the average of 28 features (7.14%). So, this tells that sentiment has a small role while price and volatility features are much more important for model prediction.

4.4.3 SHAP and LIME:

Explainability techniques give the same overall conclusion. SHAP and LIME both show that sentiment is less important than most other features. Sentiment has 3.21% of importance according to SHAP and 5.96% according to LIME, compared with 7.14% if all features were equally important. Both identify similar important features; those were ATR, abs_return, RSI_14 and negative return indicators. Overall, both techniques tell that sentiment analysis is useful but plays a secondary role, influencing roughly 3%-5% of the model's prediction, while recent price and volatility information remains much more important.

5. Discussions

5.1 Model performance

The test data is from 2021 to 2024, which contains three different market conditions. This was a little bit challenging but useful for testing models. 2021 to early 2022 was the recovering from COVID-19, the models like LSTM, Bi-LSTM, and PatchTST performed well. because these models are good at remembering past information from the data. 2022 bear market where market behavior changed a lot, volatility became higher with sudden spikes. Where models like CNN and GARCH-based hybrid models fail to capture. Because CNN focuses on local patterns but doesn't have any memory techniques to carry high volatility information overtime. 2023 to 2024 market recovered in 2023 and volatility became more stable, which was one of the facts discussed in literature review. Models understand these patterns and captured mean reversion better. PatchTST was the strongest overall model, with lowest MAE – 0.0164 without sentiment data and 0.0140 with sentiment data. As patchTST divides 21-day lookback window into patches and uses attention mechanism to decide which parts of the data are more important, this helped to understand every pattern of the market in different conditions. TFT also performed better, as it uses LSTM memory, and attention-based feature selection became an advantage to handle different market conditions.

5.2 Role of News Sentiment in Volatility Forecasting

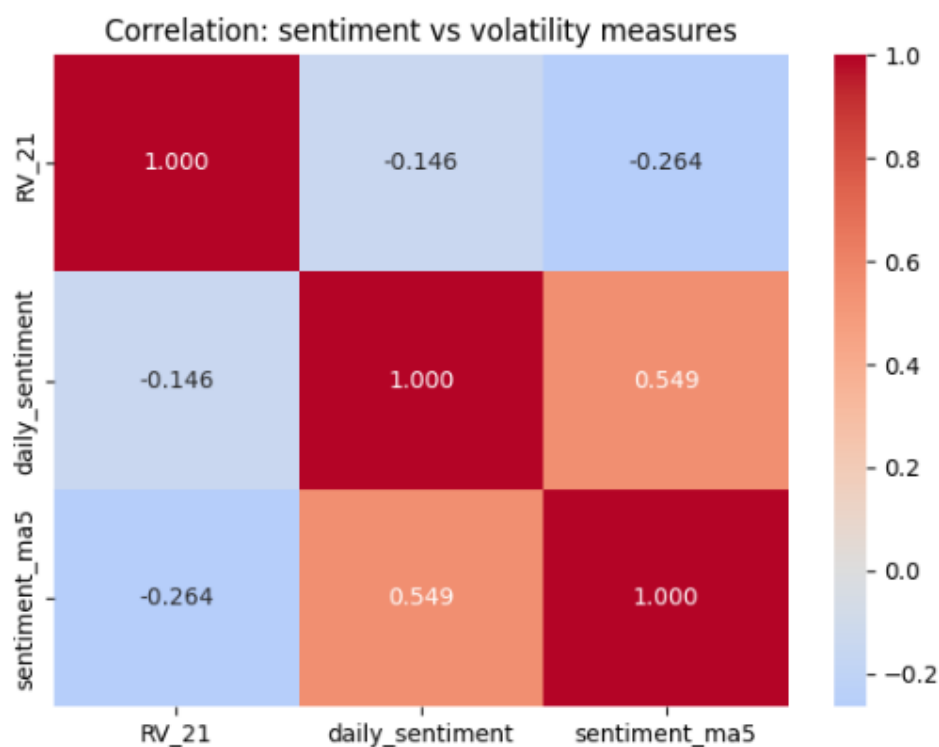


Fig-9 correlation of sentiment and volatility

The EDA showed a negative relation between sentiment and volatility. Sentiment score has -0.145 correlation with 21 day realised volatility and 5-day average sentiment has more negative correlation of -0.264 . It means more negative news linked to high market volatility, and positive news was linked with calm markets with less volatility. This supports the idea that sentiment scores hold some useful information which prices-based indicators don't. 21 days realised volatility is based on past returns, so it reacts to events already happen, but negative news can tell big moments may happen soon before it is completely captured in 21day realised volatility. So, models get information about what may happen in future from sentiment features which indicators like RSI, MACD and Bollinger band width can't provide as they were also calculated from past data. But still sentiment didn't improve the performance of all models. PatchTST benefited more from sentiment because its patching and attention mechanism was able to find import patterns from sentiment within the historical data. Instead of treating every observation equally, the patchTST model focuses more on time periods where sentiment was particularly important. TFT did not benefit from sentiment, which says that TFT decided that existing market features were more useful than sentiment and therefore sentiment got less importance while training. GARCH based model performed worse when sentiment was added, which tells that the information from sentiment may already be captured by GARCH. As a result, additional sentiment features may become unnecessary. There was a limitation in how sentiment data was handled. Some days with no news headlines, the previous sentiment value was carried forward to the nest day. Because previous news was still in trend and people follow the same previous news and execute their trades. During the test period, news data was available for 786 out of 796 days, about 99%, but for training period, coverage was only about 87%. This means the model sometimes received as old sentiment value as if it represented the current day. This may lead to, model to think previous sentiment continues to be relevant. Overall, sentiment can improve volatility forecasting, but the usefulness of sentiment depends on the model and how the model is handling sentiment information. PatchTST benefited the most, but TFT and GARCH-based models did not benefit much. This tells that sentiment is not automatically useful; the model must be able to extract the useful patterns and relations from it.

5.3 GARCH hybrid models

GARCH hybrid models, using both LSTM and GRU, did not perform well compared to other models. adding GARCH volatility as an extra feature appeared to add noise instead of useful information. GARCH (1,1) is good at learning volatility clustering, its statistical results showed that the training data is learned well, but GARCH has weakness, it does not respond well to sudden changes in market conditions. For example, volatility was low in 2021 but became much higher in 2022. GARCH forecasts can be slow to adjust to a new market condition. The hybrid model was that GARCH would find volatility patterns, and LSTM or GRU would learn the remaining patterns. But this did not work well. The

standalone LSTM performed much better than GARCH-LSTM. The reason may be GARCH forecast contains information already in features such as RV_5 and RV_21. Which creates overlap between the features and becomes harder for LSTM and GRU to find useful patterns.

5.4 Limitations

In this study, if there was no news on a day, the previous day's sentiment was used by carrying forward and no social media data like Twitter or Reddit was used, which may provide more immediate public sentiment. These results may not be the same for other indexes or companies. The test set covered 794 days, which may not be long enough to make strong conclusions. This study was conducted using Google Colab which has limited access to T4 GPU. Deep learning and FinBERT models expect more computational resources.

6 Conclusion

This study compared nine deep learning models for forecasting 21-day realised volatility of S&P 500. Each model was evaluated before augmenting FinBERT derived sentiment from news and after augmenting sentiment feature, then applying SHAP and LIME to explain model predictions. Attention-based transformer models consistently outperformed convolutional and recurrent models. PatchTST with sentiment achieved the best results across all three metrics, MAE:0.0140, RMSE: 0.0192, QLIKE: 0.0330. Showing that patching and attention made it easy to learn financial patterns and sentiment as well. GARCH hybrid models underperformed deep learning models, showing that GARCH derived conditional volatility becomes useless, may be information was already captured by financial features. Sentiment features didn't improve the performance of all models, which means that model capacity is more important than including many features. SHAP and LIME confirmed sentiment features hold lower importance than some engineered features like high-low range, ATR and absolute return. Sentiment features contributed 3-6% of total feature importance.

Reflection

This project helped me to understand deep learning model architectures, making use of additional data like sentiment. Before, I expected sentiment to improve most models, but different results challenged that expectation and highlighted the importance of model capacity. Working with FinBERT and different architectures needs careful planning and fair comparison and train-test splits. Applying SHAP and LIME showed that explainability is more than just displaying results; it helps to understand what models actually learn. In the future, I would include sentiment from social media and a longer testing period. Finally, this study showed that integrating strong architecture with interpretable

techniques produces more trust and effective forecasting system.

References

- [1] X. Chen, "Comparing various GARCH-type models in the estimation and forecasts of volatility of S&P 500 returns during Global Finance Crisis of 2008 and COVID-19 financial crisis," vol. 169, pp.1077, 2023, doi: 10.1051/shsconf/202316901077.
- [2] C. Saravanos and A. Kanavos, "Forecasting stock market volatility using social media sentiment analysis," *Neural Comput & Applic*, vol. 37, no. 17, pp.10771–10794, 2025, doi: 10.1007/s00521-024-10807-w.
- [3] N. Siddiraju and R. Hasan, "A Hybrid LSTM-GARCH Network for Stock Market Volatility Prediction", inpp. 427. doi: 10.1109/CCWC62904.2025.10903807.
- [4] Y. Ibrahim, M. S. Sharif and N. D. Moussa, "From Wall Street to Heartbeat: AI Forecasting for Financial Volatility and Investor Well-Being", inpp. 1–7. doi: 10.1109/3ict68299.2025.11442213.
- [5] C. Lento and N. Gradojevic, "S&P 500 index price spillovers around the COVID-19 market meltdown," vol. 14, no. 7, pp.1–13, 2021, doi: 10.3390/jrfm14070330.
- [6] P. Sharma and R. Manchanda, "Historical Volatility and NIFTY Returns: A Multi-timeframe Analysis Using ARDL Models," vol. 33, no. 4, pp.253–262, 2025, doi: 10.1177/09711023251358054.
- [7] E. Boateng, E. Asafo-Adjei, J. G. Gatsi, Ş C. Gherghina and L. N. Simionescu, "Multifrequency-based non-linear approach to analyzing implied volatility transmission across global financial markets," vol. 13, no. 3, pp.699–743, 2022, doi: 10.24136/oc.2022.021.
- [8] S. Wang, M. Chuang, S. Lin and S. Shyu, "Option pricing under stock market cycles with jump risks: evidence from the S&P 500 index," *Rev Quant Finan Acc*, vol. 56, no. 1, pp.25–51, 2021, doi: 10.1007/s11156-020-00885-x.
- [9] J. DUARTE, C. S. JONES and J. L. WANG, "Very Noisy Option Prices and Inference Regarding the Volatility Risk Premium," vol. 79, no. 5, pp.3581–3621, 2024, doi: 10.1111/jofi.13365.
- [10] L. A. Smales, "Stock market liquidity during crisis periods: Australian evidence," vol. 64, no. 2, pp.1849–1878, 2024, doi: 10.1111/acfi.13202.

- [11] K. Kim and J. W. Song, "Analyses on Volatility Clustering in Financial Time-Series Using Clustering Indices, Asymmetry, and Visibility Graph," *Access*, vol. 8, pp.208779–208795, 2020, doi: 10.1109/ACCESS.2020.3037240.
- [12] B. S. Corba, E. Egrioglu and A. Z. Dalar, "AR–ARCH Type Artificial Neural Network for Forecasting," *Neural Process Lett*, vol. 51, no. 1, pp.819–836, 2020, doi: 10.1007/s11063-019-10117-6.
- [13] G. S. Osho, "A Generalized Autoregressive Conditional Heteroscedasticity GARCH for Forecasting and Modeling Crude Oil Price Volatility," vol. 26, no. 6, pp.39–53, 2024, doi: 10.33423/jabe.v26i6.7385.
- [14] C. Peitz, Y. Feng, B. M. Gilroy and N. Stoeckmann, "The Shanghai- Hong Kong Stock Connect: An Application of the Semi-CGARCH and Semi-EGARCH," vol. 10, no. 4, pp.427–438, 2020, doi: 10.18488/journal.aefr.2020.104.427.438.
- [15] H. Sun and B. Yu, "Volatility asymmetry in functional threshold GARCH model," vol. 41, no. 1, pp.95–109, 2020, doi: 10.1111/jtsa.12495.
- [16] M. Nilchi and D. Farid, "Modeling Price Dynamics and Risk Forecasting in Tehran Stock Exchange Market: Nonlinear and Non-gaussian Models of Stochastic Volatility," vol. 25, no. 2, pp.275–299, 2023, doi: 10.22059/frj.2022.350231.1007402.
- [17] B. Adrangi, A. Chatrath and K. Raffiee, "S&P 500 volatility, volatility regimes, and economic uncertainty," vol. 75, no. 4, pp.1362–1387, 2023, doi: 10.1111/boer.12406.
- [18] M. Lolic, "Tree-Based Methods of Volatility Prediction for the S&P 500 Index," vol. 13, no. 4, pp.84, 2025, doi: 10.3390/computation13040084.
- [19] Z. Liu, "Stock Volatility Prediction Using LightGBM Based Algorithm", inpp. 283–286. doi: 10.1109/BDICN55575.2022.00061.
- [20] B. Han, A. Liu, J. Chen and W. Knottenbelt, "Can machine learning models better volatility forecasting? A combined method," vol. 32, no. 13, pp.1543–1564, 2026, doi: 10.1080/1351847X.2025.2553053.
- [21] N. J. C, S. S. G, Y. R. R and N. B. V. P, "Stock Market Prediction Web Service Using Deep Learning", inpp. 1384–1388. doi: 10.1109/I-SMAC65734.2025.11597602.
- [22] M. Sahiner, D. G. McMillan and D. Kambouroudis, "Do artificial neural networks provide improved volatility forecasts: Evidence from Asian markets," *J Econ Finan*, vol. 47, no. 3, pp.723–762, 2023, doi: 10.1007/s12197-023-09629-8.

- [23] P. Ranjan, R. Itani and A. Faccia, "An Interpretable 1D-CNN Framework for Stock Price Forecasting: A Comparative Study with LSTM and ARIMA," vol. 4, no. 4, pp.63, 2025, doi: 10.3390/fintech4040063.
- [24] S. Noh, "Analysis of Gradient Vanishing of RNNs and Performance Comparison," vol. 12, no. 11, pp.442, 2021, doi: 10.3390/info12110442.
- [25] X. Mei and Y. Sun, "Design and Realization of Stock Forecasting Method Based on Long Short-term Memory Model", inpp. 435–439. doi: 10.1109/NetCIT54147.2021.00092.
- [26] K. Al-Utaibi, S. Siddiq and S. M. Sait, "Stock Price Forecasting with LSTM: A Brief Analysis of Mathematics Behind LSTM," vol. 19, no. 4, pp.361–374, 2024, doi: 10.1142/S1793048023410047.
- [27] M. Che Thanh Phuoc, T. Vo Van and H. Che Ngoc, "A Comparative Analysis of GRU and LSTM Networks for Interval-Valued Foreign Exchange Market Time Series Forecasting," vol. 723, pp.1004, 2026, doi: 10.1051/e3sconf/202672301004.
- [28] R. W. A. Dipura, F. I. Maulana and Yulianto, "Enhancing Stock Market Forecasting with LSTM and BiLSTM Models: A Deep Learning Approach", inpp. 50–55. doi: 10.1109/ISRIT64779.2024.10963428.
- [29] S. Ismail, R. V. Pattani and S. C. Chacko, "SMP - TFT: A Transformer-Based AI Framework for Stock Market Prediction", in *2025 IEEE 6th India Council International Subsections Conference (INDISCON)*, pp. 1–6. doi: 10.1109/INDISCON66021.2025.11254107.
- [30] G. Kalaiarasan, B. Premjith and N. M. Manjiparambil, "A Forecast-Driven Deep Learning Framework Using ICEEMDAN and PatchTST for Stock Prediction and Dynamic Portfolio Optimization," *Access*, vol. 13, pp.1, 2025, doi: 10.1109/ACCESS.2025.3648786.
- [31] H. Jlassi and H. Hamdi, "Cryptocurrency Volatility Forecasting Using Garch, LSTM, Hybrid GARCH-LSTM and Hybrid CNN-LSTM with Intraday Frequency", in *2025 IEEE Afro-Mediterranean Conference on Artificial Intelligence (AMCAI)*, pp. 1–7. doi: 10.1109/AMCAI66110.2025.11474386.
- [32] D. K. Mademlis and N. Dritsakis, "VOLATILITY FORECASTING USING HYBRID GARCH NEURAL NETWORK MODELS: THE CASE OF THE ITALIAN STOCK MARKET," vol. 11, no. 1, pp.49–60, 2021, doi: 10.32479/ijefi.10842.
- [33] A. Prasad, "Forecasting spikes in the CBOE VIX Index," ProQuest Dissertations & Theses, 2022.

- [34] W. Zhang, X. Gong, C. Wang and X. Ye, "Predicting stock market volatility based on textual sentiment: A nonlinear analysis," *J.Forecast.*, vol. 40, no. 8, pp.1479–1500, 2021, doi: 10.1002/for.2777.
- [35] M. S. Prasad Guru, P. G. J, M. Mittal, H. R. Kumar Prasanna, V. S. K and A. Jahan, "Rule-Based Sentiment Analysis for Twitter(X) Data", inpp. 1–5. doi: 10.1109/AMATHE61652.2024.10582181.
- [36] J. Kim, H. Kim and S. Choi, "Forecasting the S&P 500 Index Using Mathematical-Based Sentiment Analysis and Deep Learning Models: A FinBERT Transformer Model and LSTM," vol. 12, no. 9, pp.835, 2023, doi: 10.3390/axioms12090835.
- [37] K. Srivastava, A. Sorathiya, J. Mehta and V. Chotaliya, "Enhancing Interpretability, Reliability and Trustworthiness: Applications of Explainable Artificial Intelligence in Medical Imaging, Financial Markets, and Sentiment Analysis", inpp. 1–9. doi: 10.1109/ECAI61503.2024.10607516.
- [38] H. Shalit, "Weighted Shapley values of efficient portfolios," vol. 9, no. 2-4, pp.31–38, 2023, doi: 10.3233/RDA-231507.
- [39] K. Sharma, S. Gupta, S. Gadkari and V. Rathod, "Explainable AI for Temporal Feature Interpretation in Volatility Forecasting", inpp. 1781–1786. doi: 10.1109/CICN67655.2025.11368102.
- [40] G. Taneva-Angelova and D. Granchev, "Deep Learning and Transformer Architectures for Volatility Forecasting: Evidence from U.S. Equity Indices," vol. 18, no. 12, pp.685, 2025, doi: 10.3390/jrfm18120685.
- [41] E. Koo and G. Kim, "A Hybrid Prediction Model Integrating GARCH Models With a Distribution Manipulation Strategy Based on LSTM Networks for Stock Market Volatility," *Access*, vol. 10, pp.34743–34754, 2022, doi: 10.1109/ACCESS.2022.3163723.
- [42] X. Chen, B. Li and A. C. Worthington, "Higher moments and US industry returns: realized skewness and kurtosis," vol. 20, no. 1, pp.1–22, 2021, doi: 10.1108/RAF-06-2020-0171.
- [43] S. Fernandez-Vazquez, R. Rosillo, Juan Martínez Alvarez and Pedro Julián Ramírez-Angulo, "Enhancing cryptocurrency trading with technical analysis: exploring RSI, MACD, momentum, and stochastic oscillator indicators," vol. 36, no. 101, pp.e114957, 2026, doi: 10.15446/innovar.v36n101.114957.
- [44] J. Olbryś, "Entropy of Volatility Changes: Novel Method for Assessment of Regularity in Volatility Time Series," vol. 27, no. 3, pp.318, 2025, doi: 10.3390/e27030318.

- [45] T. Aravanis and A. Kanavos, "Comparative Evaluation of Deep Learning Architectures for Next-Day Stock Price Forecasting Using Technical Indicators," vol. 14, no. 15, pp.2736, 2026, doi: 10.3390/math14152736.
- [46] B. M. Hussein and S. M. Shareef, "An Empirical Study on the Correlation between Early Stopping Patience and Epochs in Deep Learning," vol. 64, pp.1003, 2024, doi: 10.1051/itmconf/20246401003.
- [47] M. Zulqarnain, R. Ghazali, M. G. Ghouse, Y. M. M. Hassim and I. Javid, "Predicting Financial Prices of Stock Market using Recurrent Convolutional Neural Networks," vol. 12, no. 6, pp.21–32, 2020, doi: 10.5815/ijisa.2020.06.02.
- [48] Y. Büyükkör, "Deep Learning in Financial Time Series: A Comparative Analysis of RNN, GRU, LSTM, and Hybrid Models," vol. 28, no. 1, pp.5–38, 2026, doi: 10.15179/ces.28.1.1.
- [49] P. Anand and S. Yadav, "A Time-Series Forecasting Framework for Stock Market Prediction Using LSTM and Technical Indicators", inpp. 773–776. doi: 10.1109/CISES66934.2025.11265263.
- [50] M. A. Istiaque Sunny, M. M. S. Maswood and A. G. Alharbi, "Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model", inpp. 87–92. doi: 10.1109/NILES50944.2020.9257950.
- [51] H. A. Ahmad, S. K. Mortazavi, M. El Bahnasawi, F. Al Machot, W. V. Kambale and K. Kyamakya, "Enhanced Time Series Forecasting: Integrating PatchTST with BERT Layers", in *2024 International Conference on Applied Mathematics & Computer Science (ICAMCS)*, pp. 60–65. doi: 10.1109/ICAMCS62774.2024.00014.
- [52] X. Hu, "Stock Price Prediction Based on Temporal Fusion Transformer", in *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pp. 60–66. doi: 10.1109/MLBDBI54094.2021.00019.
- [53] F. Zhang and H. Wang, "Does CNN-Based Feature Extraction Improve High-Frequency Return Prediction? Evidence from the CSI 300 Index," vol. 19, no. 5, pp.371, 2026, doi: 10.3390/jrfm19050371.
- [54] K. Singh, B. Soni and A. Sharma, "Deep Learning in Financial Markets - A CNN-LSTM Approach to Stock Price Forecasting", inpp. 1–7. doi: 10.1109/GIEST62955.2024.10960084.
- [55] J. Michańków, Ł Kwiatkowski and J. Morajda, "Combining Deep Learning and GARCH Models for Financial Volatility and Risk Forecasting," 2023, doi: 10.48550/arxiv.2310.01063.

- [56] A. Singh, M. Singh and L. R, "Impact of Macroeconomic Integration in Hybrid GARCH-GRU Volatility Modelling on Nifty Bank", inpp. 406–411. doi: 10.1109/ICAAIC64647.2025.11330787.
- [57] A. Moiz and H. Raza, "Predicting volatility of cryptocurrencies: Deep learning and GARCH family models," vol. 4, no. 1, pp.95–113, 2026, doi: 10.61351/mf.v4i1.370.
- [58] S. Baghavathi Priya, M. Kumar, N. Prakash J D and K. N, "Advanced Financial Sentiment Analysis Using FinBERT to Explore Sentiment Dynamics", inpp. 889–897. doi: 10.1109/IDCIOT64235.2025.10915080.
- [59] C. Cherry, W. Mohamed and Y. Brahmbhatt, "Using FinBERT as a refined approach to measuring impression management in corporate reports during a crisis," vol. 42, no. 1, pp.64–80, 2023, doi: 10.36615/jcsa.v42i1.2318.
- [60] R. Pandiri, "Risk-Adjusted Return Forecasting Using Hybrid Financial and News-Driven Models", inpp. 623–629. doi: 10.1109/ICMLAS67792.2026.11483838.
- [61] N. Darraz, I. Karabila, A. El-Ansari, N. Alami and M. El Mallahi, "Detecting trending products through moving average and sentiment analysis," *Multimed Tools Appl*, vol. 84, no. 13, pp.12141–12162, 2025, doi: 10.1007/s11042-024-19430-2.
- [62] S. T. Bodilsen and A. Lunde, "Exploiting News Analytics for Volatility Forecasting," vol. 40, no. 1, pp.18–36, 2025, doi: 10.1002/jae.3095.
- [63] H. T. Araya, J. Aduda, T. Berhane and M. L. Wong, "A Hybrid GARCH and Deep Learning Method for Volatility Prediction," vol. 2024, no. 1, 2024, doi: 10.1155/2024/6305525.
- [64] M. Haroon *et al.*, , "Improving Stock Price Prediction Accuracy Using Multivalued Regression: The Role of Normalization Techniques in Machine Learning Models", inpp. 1–6. doi: 10.1109/IC3TES62412.2024.10877679.
- [65] E. Alreshidi, "Forecasting of Saudi Stock Prices Using Statistical and Machine Learning Models: A Multi-Model Comparative Approach," vol. 16, no. 11, 2025, doi: 10.14569/IJACSA.2025.0161114.
- [66] T. Pang and Y. Zhao, "On GARCH and Autoregressive Stochastic Volatility Approaches for Market Calibration and Option Pricing," vol. 13, no. 2, pp.31, 2025, doi: 10.3390/risks13020031.
- [67] C. Ergenc and R. Aktas, "Sector-specific financial forecasting with machine learning algorithm and SHAP interaction values," vol. 21, no. 1, pp.42–66, 2025, doi: 10.2478/fiqf-2025-0004.

- [68] C. M. Shih and K. C. Yang, "Using LightGBM + SHAP to Analyze the Impact of Factors on Stock Market Investment", inpp. 1392–1396. doi: 10.1109/IEEM62345.2024.10857103.
- [69] D. K. Giri and S. Gangarapu, "Explainable AI Techniques for Transparent Algorithmic Trading Decisions", inpp. 1. doi: 10.1109/SoutheastCon63549.2026.11475945.
- [70] H. Tanveer, S. Arshad, H. Ameer and S. Latif, "Interpretability in Financial Forecasting: The Role of eXplainable AI in Stock Market", inpp. 179–183. doi: 10.1109/ICSTE63875.2024.00039.