Comparative Analysis of Univariate and Multivariate Models in Short-Term Stock Volatility Forecasting

by

Maria Gkavela [687341]

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Quantitative Finance and Actuarial Science

Tilburg School of Economics and Management
Tilburg University

Supervisor: dr. Ramon van den Akker

Second Reader: dr.ir. Erwin Charlier

Date: December 2023

Acknowledgements

I want to express my profound gratitude to my supervisor, dr. Ramon van den Akker, whose guidance and support have been invaluable throughout this journey. His expertise and insights have been fundamental to the successful completion of this work. I am also grateful to my dear friend Kyriaki Roumpi for her support and encouragement, which have strengthened me. Lastly, I must express my appreciation to my family, whose belief in me has been my constant source of courage and motivation.

Abstract

This thesis aims to predict short-term stock volatility, specifically a 5-day ahead forecast, for Apple and Amazon stocks. A comparative analysis assesses the robustness of various models within a univariate framework, including statistical models such as ARIMA, ARCH, and GARCH, as well as machine learning techniques comprising feed forward neural networks (FFNNs) with one to five layers and long short-term memory (LSTM) networks.

In addition to univariate models, the study delved into a multivariate approach by incorporating 63 diverse indicators, among them technical, fundamental, macroeconomic, and sentiment domains. These indicators were applied to LSTM and random forest models to evaluate their impact on predictive performance. Our findings suggest that while the GARCH model displayed consistent robustness on a weekly basis, the univariate LSTM and single-layered FFNN were particularly effective on a monthly scale. Notably, multivariate models outperform their univariate counterparts during periods of high volatility, suggesting an advantage in integrating diverse features to navigate the complexities of 'abnormal' market conditions.

Benchmarked against a naive model, the study indicates that multivariate approaches, specifically the random forest model for Apple and the LSTM for Amazon, showed superior performance during the pandemic. This highlights the role of market sentiment and firm-specific fundamentals in driving stock volatility during crises, illustrating the intricate relationship between various indicators and market behavior.

Contents

1 Introduction **5**
1.1 Literature Review 6
1.2 Purpose and Research Question 7

2 Volatility and Data Analysis **9**
2.1 Introduction to Volatility 9
2.2 Data 10
2.2.1 True Realized Volatility 10
2.2.2 Technical Indicators 13
2.2.3 Fundamental Indicators 16
2.2.4 Macroeconomic Indicators 18
2.2.5 Categorical Variable for the Coronavirus Period 20
2.2.6 Other Indicators 21
2.2.7 News Articles and Sentiment Analysis 21

3 Methodology **25**
3.1 Data Splitting and Hyperparameter Optimization 25
3.2 Data Preprocessing and Transformations 28
3.3 Evaluation Metrics and Feature Interpretation 28

4 Model Theoretical Basis and Implementation Techniques **31**
4.1 Statistical Models 31
4.1.1 ARIMA model 31
4.1.2 ARCH model 33
4.1.3 GARCH model 34
4.2 Artificial Neural Network 36
4.2.1 Feed Forward Neural Network 36
4.2.2 Long Short Term Memory Neural Network 39
4.3 Tree-Based Method: Random Forest 41

5 Results **44**
5.1 Baseline Models 44
5.2 ARIMA Model Analysis 45
5.2.1 Results and Interpretation 45
5.3 ARCH and GARCH Models Analysis 46
5.3.1 Results and Interpretation 46
5.3.2 Comparative Analysis of GARCH and ARCH Models Using AIC and BIC Criteria and Their Combined Forecasting Approach 49
5.4 Comparison of Statistical Models to Baseline Model 50
5.5 Neural Network Analysis 52
5.5.1 Comparison of Neural Network Models to Baseline Model 52
5.6 Multivariate Models 53
5.6.1 Combining dictionaries 53
5.6.2 Comparison of Multivariate Models to Baseline Model 55
5.7 Comparison of all models 55
5.8 Feature Importance 58

6 Conclusion and Discussion 63

7 Appendix 65

8 References 67

1 Introduction

For many years, experts and investors have aimed to understand and predict the behavior of stock market prices and returns. It has been posited that stock prices follow a random walk, meaning that the changes in prices today do not have a predictable relationship with future price changes. In this context, economist Fama (1965) introduced the Efficient Market Hypothesis (EMH), which suggests that stock prices effectively incorporate all available information, making it challenging for investors to outperform the market consistently. This view is supported by Malkiel (2003), who, after evaluating the performance of various funds, concluded that consistent long-term outperformance of the market is elusive. Given the challenges of forecasting returns, recent attention has shifted towards understanding stock return volatility – the magnitude of return variations. Although forecasting stock returns can be inherently challenging, analyzing volatility offers a valuable perspective, providing investors and portfolio managers insights to develop risk mitigation strategies. Hence, over the last few decades, forecasting volatility has gained significant attention in both academic and professional circles, underlining its vital importance.

Given the heightened interest in this field, numerous studies have employed techniques and approaches to achieve robust forecasting results. Beginning with early statistical models like the autoregressive moving average model (ARMA) (Wadhawan & Singh (2019)), autoregressive conditional heteroskedasticity (ARCH) (Alam et al. (2013)), and its generalized version (GARCH) (Costa (2017)), extending to an array of machine learning and deep learning techniques (Christensen et al. (2021) and Xiong et al. (2015)). More recent advancements in neural networks and ensemble methods have also been incorporated into the vast literature (Mademlis & Dritsakis (2021)).

Despite the diverse methodologies, all share a common goal: to understand and predict stock volatility accurately. Recognizing the complex nature of stock prices and the vast information they carry, numerous studies have tried to address this complexity by incorporating a range of features that could potentially impact stock prices and, consequently, their volatility (Filipović & Khalilzadeh (2021)).

The most dominant approaches for both researchers and investors in stock market analysis are Technical Analysis and Fundamental Analysis. According to Murphy (1999): "Technical analysis is the study of market action, primarily through the use of charts, for the purpose of forecasting future price trends. The term market action includes the two principal sources of information, price and volume" (p.1). Technical analysts believe that historical price movements repeat themselves and that these patterns can be used to predict future price movements. Their objective is to leverage price fluctuations over the short term, which usually spans from mere minutes to several months.

Fundamental analysis is concerned with the economic factors that influence price movements (Murphy (1999)). As Krantz (2023) wrote: " Fundamental analysis is the skill of reading through all the information companies provide about themselves to make intelligent decisions" (p.10). Fundamental analysts examine company financial statements, earnings reports, industry trends, competitive analysis, and management performance. The goal is to determine a stock's true worth and identify whether it is undervalued or overvalued. Fundamental analysis helps investors assess a company's long-term prospects and make investment decisions based on its underlying fundamentals.

Although Technical Analysis and Fundamental Analysis are often viewed as opposing approaches, many investors and researchers combine elements from both methodologies to comprehensively understand the market. They aim to make more informed decisions and reduce potential risks by considering both technical indicators and fundamental factors.

In addition to technical and fundamental analysis, macroeconomic variables also play a significant role in stock market analysis. Macroeconomic factors, such as interest rates, inflation, GDP growth, and government policies, can profoundly impact the overall market and individual stocks. Investors often analyze macroeconomic data to assess the economy's health and its potential influence on stock volatility.

Given the accessibility of news and the advancements in natural language processing, researchers and investors have increasingly turned to sentiment analysis in recent years. This approach incorporates an additional, vital factor into examining stock price influences. By assessing public sentiment from various news sources and social media platforms, they can derive insights into the potential impact on stock prices and, by extension, stock volatility.

1.1 Literature Review

Various statistical and machine learning techniques have been employed to forecast stock volatility. Within the domain of statistical models, special attention has been given to the autoregressive conditional heteroskedasticity (ARCH) models and its extension, the generalized autoregressive conditional heteroskedasticity (GARCH). In parallel, auto-regressive moving average (ARMA) and auto-regression integrated moving average (ARIMA) have emerged as commonly used models. Certain researchers have fine-tuned ARIMA models, experimenting with various parameters and subsequently choosing models based on metrics like the Akaike Information Criterion (AIC) or the adjusted R-squared value, with WAHYUDI (2017) serving as a notable example. Additionally, some studies delve into a comparative analysis between ARCH and ARIMA models, as illustrated by Ibrahim (2017).

On the other hand, machine learning methods, especially artificial neural networks (ANNs), have been increasingly popular in predicting stock volatility. Some studies have solely used ANNs, such as Dixit et al. (2013) and Bucci (2020), while others have compared different machine learning techniques, including ANNs. Filipović & Khalilzadeh (2021) compared a linear regression model with an ANN for forecasting stock return volatility. They found that the ANN outperformed the linear model by 22% in out-of-sample R-squared. Gu et al. (2020) compared the performance of three different techniques, namely generalized linear model, regression trees, and ANNs, and found that the ANN had the best performance. They also researched to identify the most relevant features among a large set of predictors using feature selection and dimension reduction methods.

Furthermore, integrating different models has been observed in recent years to achieve more accurate results. For instance, Kim & Won (2018) integrated a long short-term memory (LSTM) model with various types of GARCH models for predicting stock volatility, while Pai & Lin (2005) demonstrated that a hybrid support vector machines and ARIMA model outperformed the single models. They concluded that integrating linear and nonlinear models could better capture the data patterns.

Another research category has focused on comparing machine learning methods with statistical models. Chatterjee et al. (2022) compared three GARCH-based models and a LSTM model by computing the Root Mean Squared Error (RMSE) and concluded that the latter outperformed the GARCH models. An extended generalized version called ARIMAX, which allows the inclusion of exogenous variables as inputs, has been introduced to overcome ARIMA's limitations. Serafini et al. (2020) tested ARIMAX's efficiency for predicting Bitcoin price by comparing it with a recurrent neural network (RNN) with sentiment feature inclusion. They found that ARIMAX had the lowest Mean Squared Error (MSE). Patil (n.d.) also showed similar results, emphasizing the positive impact of adding additional variables, such as text-data features, in the ARIMAX model.

In addition, feature selection plays a crucial role in a model's performance, and various approaches have been used for selecting the most relevant predictors. Technical and fundamental analyses have been the dominant approaches for forecasting stock volatility, while macroeconomic variables have also been included in some analyses. Bettman et al. (2009) found that combining both approaches led to significantly better predictions than using either alone. Ludwig et al. (2015) pursued a feature selection using LASSO and random forest before using the ARMAX model for forecasting electricity prices. Sudhakar & Naganjaneyulu (2020) used the LASSO method to find the most relevant predictors among historical prices, indices yield, and news data and developed an ARIMAX model for predicting stock price. They compared the ARIMAX model with only historical prices to the model with the selected variables from the LASSO technique. They concluded that the latter had the best performance in terms of RMSE.

1.2 Purpose and Research Question

Numerous studies have been conducted on stock volatility prediction, with researchers using multiple approaches and techniques. While many studies have individually delved into comparisons within statistical or machine learning models, only a few have undertaken the challenge of directly contrasting multiple models from both fields. Furthermore, in scenarios where multivariate settings are employed, the integration typically spans just two distinct categories of indicators. This leaves a noticeable research gap: a comprehensive comparison across multiple univariate, both statistical and machine learning domains and the exploration of multivariate models with their univariate counterparts. Addressing this gap can provide profound insights into each model's implications and potential benefits for stock volatility prediction.
Given the issues mentioned above, two main research questions will be addressed:

- How do statistical and machine learning models perform in predicting short-term stock volatility within a univariate setting?
- How do the predictive accuracy of multivariate models, which integrate a diverse range of features, compare to those of univariate models in stock volatility forecasting? Additionally, which specific features contribute most significantly to enhancing predictive performance?

In this study, our primary objective is to forecast stock volatility. We utilize models from both statistical and machine learning fields. Later in our analysis, we incorporate indicators discussed in Section 1.1, along with a specific dummy variable to account for the unique circumstances posed by the pandemic, given that our examination spans from 2013 to 2023. A naive model is introduced for a foundational comparison and to understand the efficacy of our chosen models. We segment our data and examine distinct periods separately to refine our analysis further and ensure its robustness.

The subsequent sections are organized in the following manner. In Section 2, we introduce our target variable and describe the indicators of our interest, complemented by a preliminary explanatory data analysis. In Section 3, we describe the central methodology adopted for our analysis. Section 4 delves into the theoretical framework and details each model's implementation techniques. Our research findings are presented in Section 5. Lastly, Section 6 summarizes our results and offers discussions on potential enhancements and alternative strategies to bolster the robustness of stock volatility forecasts.

2 Volatility and Data Analysis

2.1 Introduction to Volatility

The challenges of forecasting returns have led to an increased focus on understanding stock return volatility. Properly addressing this aspect offers valuable insights for developing risk mitigation strategies for investor and portfolio decisions.

However, a crucial challenge arises when measuring volatility: it is not directly observable. Multiple approaches have been proposed to resolve this matter. Among these strategies is the adoption of stochastic volatility models, characterized by the assumption that volatility behaves in a stochastic (random) manner. A seminal work of stochastic volatility models can be traced back to Taylor (2008)[1]. Additionally, the (generalized) autoregressive conditional heteroskedasticity, or (G)ARCH, introduced by Engle (1982) and Bollerslev (1986), provides another model-based avenue for measuring volatility. Although these models target a well-defined object—volatility— to estimate, they require reliance on latent variables, which adds a layer of complexity to forecasting volatility (McAleer & Medeiros (2008)).

Realized volatility, constructed using high-frequency data, offers a compelling alternative for measuring volatility. Numerous studies have employed this framework to compute past volatility to forecast future volatility. Yet, obtaining intraday returns, crucial for this method, proves challenging due to data availability and limited resources. Therefore, in this research, we are unable to employ this approach.

For this project, we are considering True Realized Volatility (TRV) as a measure of volatility. The TRV was computed for each day as per the method outlined by Roh (2007) and employed by Ramos-Pérez et al. (2019). This method leverages the standard deviation, a common measure for stock return volatility while incorporating a moving average approach to account for recent market behaviors. Through this method, we aim to capture the daily volatility of stock returns over a specific time window, providing a robust basis for our subsequent analyses. The TRV is defined as:

$$TRV_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_{t-i+1} - \widehat{r}_t)^2} \quad \text{where } n = 5, \tag{1}$$

with the stock returns, r_t, defined by:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right), \tag{2}$$

where P_t denotes the stock price at time t, and with:

$$\widehat{r}_t = \frac{1}{n}\sum_{i=1}^{n} r_{t-i+1}, \tag{3}$$

the average return over the previous five days.

It is noteworthy that TRV represents a daily measure based on the volatility over the past five days. Thus, it can be interpreted as daily volatility. The term $\widehat{r}_t$ represents the average return over the previous five days. Returns are determined by taking the natural logarithm of the price ratio at two adjacent time points. Our study adopted a window size of n=5

[1]The first edition was published in 1986.

for calculating the TRV. The methodology for determining the window size draws upon the approach suggested by Ramos-Pérez et al. (2019), which balances the need for stability in TRV calculation against the risk of blending different volatility periods.

For the scope of this research, our forecast targets the 5th day ahead. Details behind the choice of this horizon and its significance in the context of our employed models will be elaborated in Section 4.1.3.

2.2 Data

For this thesis, we focus on two stocks with high market capitalization: Apple and Amazon. Our dataset spans ten years, with daily observations recorded from January 1, 2013, to January 1, 2023. We sourced the daily Open, High, Low, Close, and Volume prices for both stocks from Yahoo Finance. Our primary objective is to predict daily volatility, TRV_t, for both stocks by employing various models, including statistical and nonlinear (machine learning) models, which will be extensively discussed in Section 4. Initially, our analysis will consider univariate models, which predict volatility using a single variable. Subsequently, we will explore two multivariate models considering multiple variables simultaneously. In the latter, various independent variables are examined and tested to assess their contribution to explaining the variation in daily volatility. This transition aims to enhance the accuracy and robustness of our forecasts. Before delving into the detailed analysis, it is essential to introduce the indicators that will be utilized, along with a preliminary analysis of the independent and dependent variables.

2.2.1 True Realized Volatility

Our preliminary analysis focuses on our dependent variable, TRV_t, and the daily returns from which the TRV is computed. Figure 1 illustrates the volatility fluctuations for both stocks over time, along with their corresponding returns. On average, Apple exhibits more stability with an annualized TRV of 22.24%, whereas Amazon demonstrates greater volatility with an annualized TRV of 24.75%. Notably, the significant spikes in TRV for Amazon point to potential short-term extreme volatility events. A particularly intriguing period is 2022, when Apple's TRV returns to its regular values, whereas Amazon's volatility remains elevated, suggesting the presence of these short-term events. These spikes might indicate specific company events or related news affecting stock prices and, consequently, TRV.

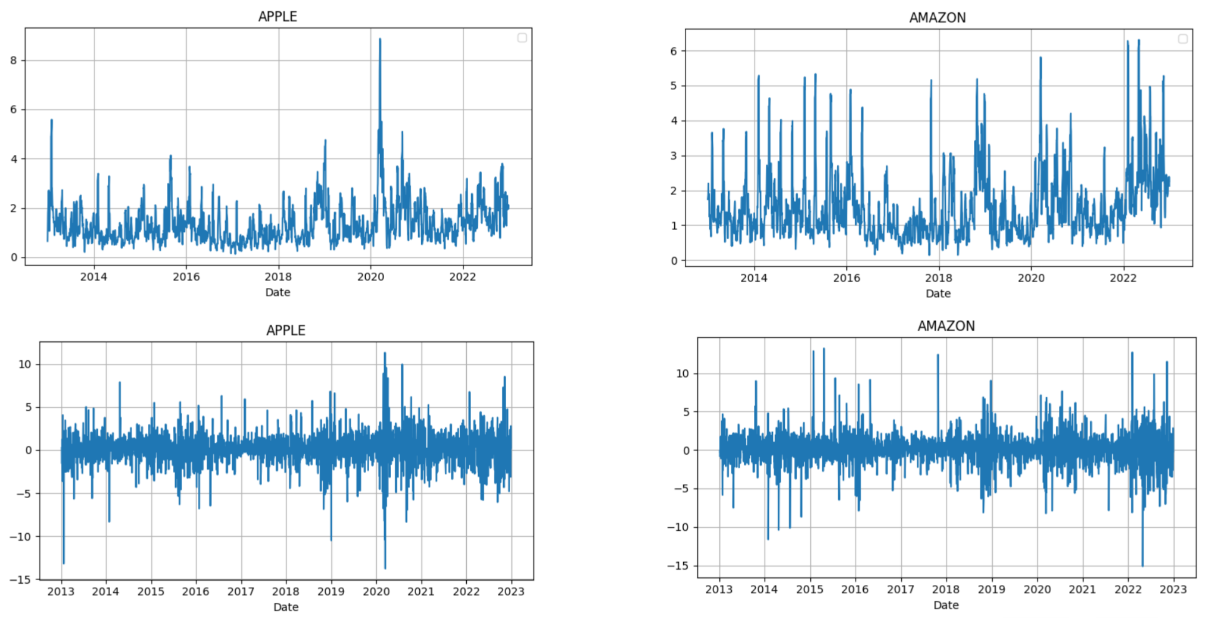


Figure 1: Time plots depicting the True Realized Volatility (TRV) for Apple and Amazon (top row), alongside the Daily Returns for the same (bottom row), over the period January 2013 to January 2023. All values are represented in percentages.

Table 1 presents the descriptive statistics. As depicted, Amazon's variance is higher than Apple's for both returns and TRV. The kurtosis results indicate that both Apple and Amazon have more extreme values/outliers than a normal distribution, with Apple's True Realized Volatility exhibiting higher kurtosis. This observation is further supported by the Jarque-Bera test, where the high values underscore the non-normality of both stocks. The Jarque-Bera test's results necessitate the rejection of the null hypothesis that the data are distributed normally at all usual levels of significance. This suggests that while Amazon might experience more significant short-term fluctuations, Apple's distribution of volatility values has thicker tails, leading to more extreme values for the latter. These patterns become visually clear in Figure 2. While all distributions resemble a bell shape, they are narrower and peak higher, suggesting more extreme values. Given that these outliers can provide valuable insights about its distinct periods, including them can be described as vital.

The autocorrelation plots of the daily volatility, TRV_t of Apple and Amazon stocks, are illustrated in Figure 3. These plots depict the correlation between the current daily volatility and its past values, providing insights into the relationship and patterns in the volatility series. For both stocks, the autocorrelation remains above the significance area for about 28 lags, meaning the daily volatility of the stocks is significantly related to its volatility up to 28 days in the past. Apple shows a steady downward trend, suggesting the further we go into the past, the weaker the relationship with today's volatility becomes. However, Amazon's volatility shows a periodic influence from its distant past.

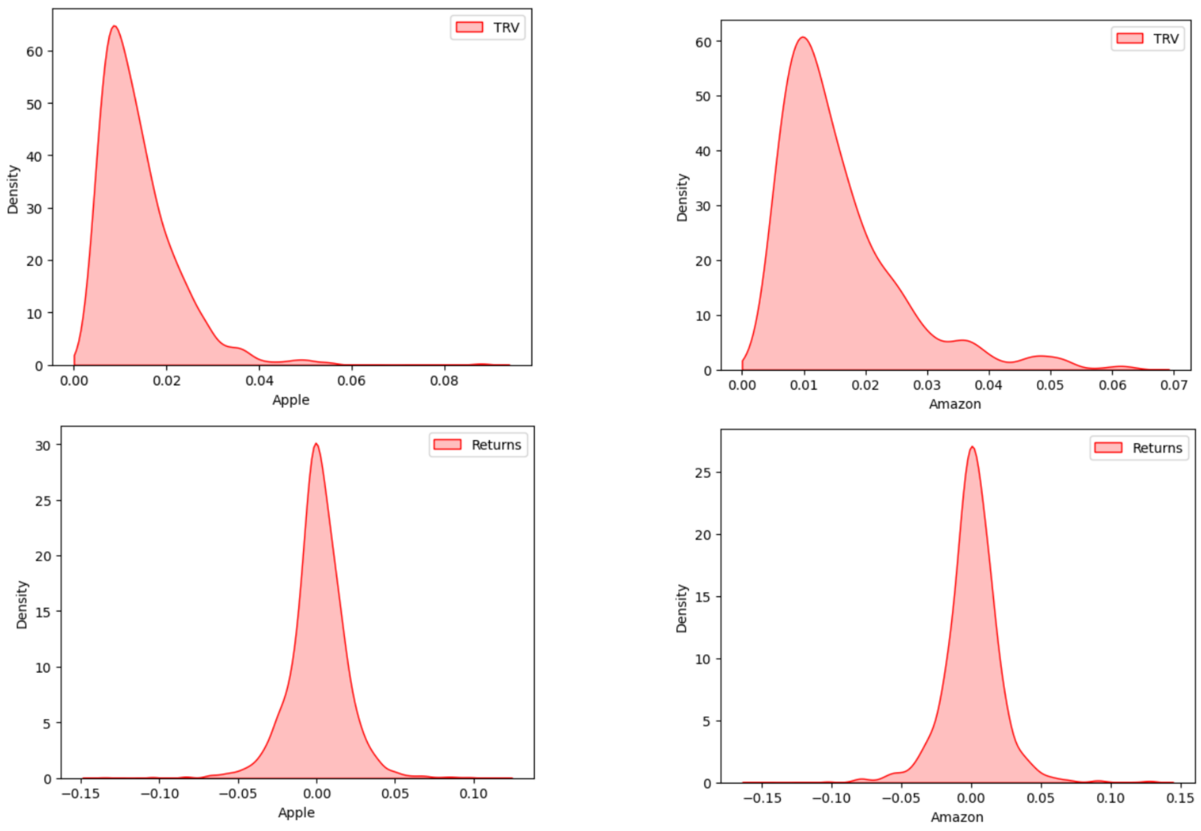


Figure 2: Kernel Density Distributions: True Realized Volatility for Apple and Amazon (top row), and Returns (bottom row).

		Mean	Standard Deviation	Skewness	Kurtosis	Jarque-Bera
Apple	Return	19.43	29.12	-0.326	5.841	3709.43
	TRV	0.01397	0.00871	2.293	10.519	14271.24
Amazon	Return	18.38	32.55	0.011	6.542	4595.31
	TRV	0.01552	0.00981	1.714	3.547	2639.48

Table 1: Descriptive Statistics for TRV_t, and Daily Returns. Note: The Mean and Standard Deviation of the Returns are the annualized percentages.

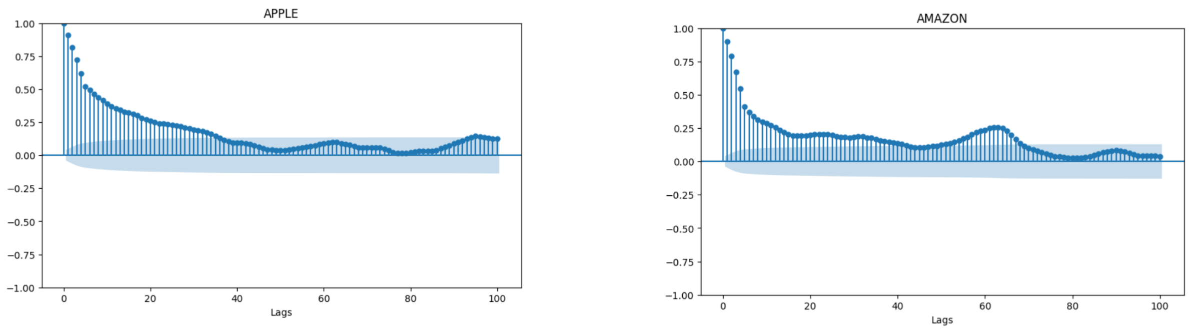


Figure 3: Autocorrelation plot of Daily Stock Volatility, TRV_t, up to 100 Lags.

2.2.2 Technical Indicators

The relationship between technical indicators and stock volatility has been explored by Kyoung-Sook & Hongjoong (2019), who introduced several indicators as potential independent variables. However, there is a noticeable lack of comprehensive research in this field regarding applying these indicators to predict stock volatility. This observed research gap serves as the primary motivation for our study. Our objective is to assess the predictive power of these technical indicators to stock volatility. Our research is informed by studies such as Picasso et al. (2019) and Emir et al. (2012). Additionally, the Technical Analysis Library for Python by Lopez Padial (2018) was instrumental in allowing us to incorporate a broader range of indicators. For our study, we calculated 25 technical indicators from the daily price data and divided them into categories that capture different aspects of Technical Analysis.

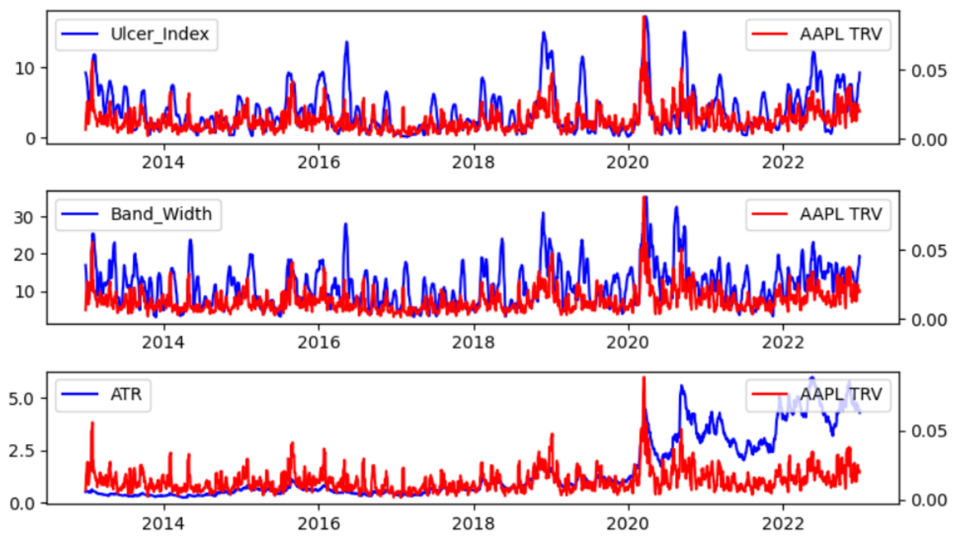
Some categories include momentum indicators, such as the Relative Strength Index (RSI) and True Strength Index (TSI), which measure the strength of the stock's price movement. Volatility factors, like the Average True Range (ATR) and Bollinger Bands, measure a stock's price movement range and help identify potential price reversals. Volume indicators, such as Money Flow Index (MFI) and Chaikin Money Flow (CMF), measure the level of buying or selling movements in the market. Finally, trend indicators, such as the Simple Moving Average (SMA) and Exponential Moving Average (EMA), help identify the direction of a stock's price movement over time.

A brief introduction to some of these indicators can be found in Table 3. Figure 4 displays the three technical indicators$_{t-k}$, (where k=5)[2], with the strongest linear correlation ($\rho > 0.3$) to our target variable. TRV_t. Table 2 provides their descriptive statistics. From the plots, the correlation is especially evident for the first two indicators. While these indicators exhibit strong linear correlations and might be powerful candidates for prediction, it is intriguing to investigate further. Specifically, we aim to discern whether these indicators truly offer significant contributions or if other features might better capture the patterns of the TRV. The Jarque-Bera test shows that all indicators deviate from a normal distribution. Most indicators, except for the Average True Range for Amazon, display positive kurtosis, suggesting thicker tails and indicating the presence of extreme values (outliers) when compared to a normal distribution. On the other hand, the negative kurtosis observed for Amazon's Average True Range implies a distribution with lighter tails, suggesting fewer extreme values.

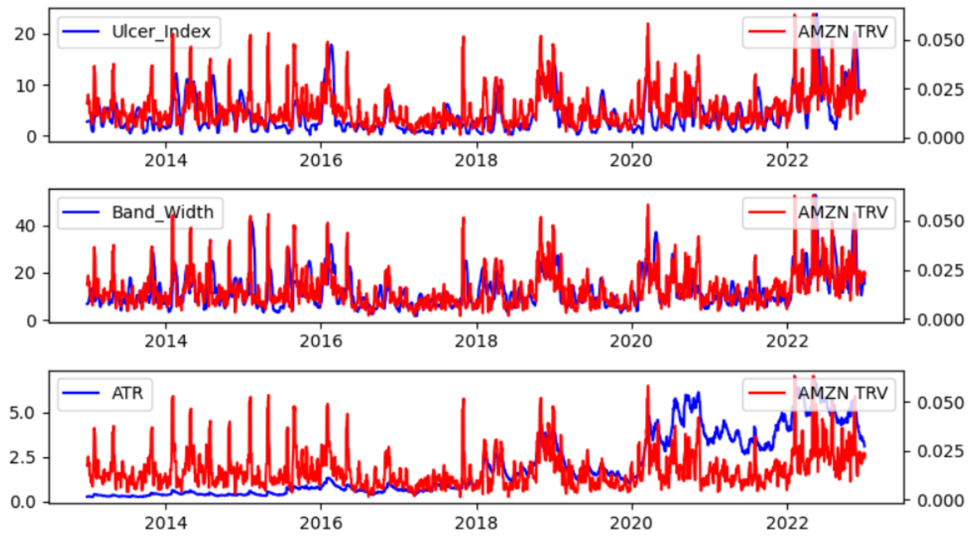
	Apple				
Statistic	Mean	Std	Kurtosis	Skewness	Jarque-Bera
Ulcer Index	3.7084	3.1259	1.9150	1.4129	1220.05
Band Width	11.4538	5.4854	1.3365	1.0147	618.09
ATR	1.5239	1.4868	0.2273	1.2537	664.60
	Amazon				
Ulcer Index	4.1713	3.6628	4.2857	1.8371	3334.62
Band Width	12.5333	7.3557	3.6052	1.6030	2435.53
ATR	1.9550	1.7583	-0.4306	0.9232	377.12

Table 2: Descriptive Statistics of the Top Three Technical Indicators Highly Correlated with Apple and Amazon TRV, Respectively.

[2]For further details, see Section 4.1.3.



(a) Time plots of Technical Indicators alongside Apple's TRV.



(b) Time plots of Technical Indicators alongside Amazon's TRV.

Figure 4: Time plots for both Apple and Amazon. The left y-axis indicates the range for each indicator with lag=5, while the right y-axis represents the respective TRV_t.

Technical Indicators	
Name	Description
Trend Indicators	
Commodity Channel Index (CCI)	Measures the deviation of the stock's price from its average within a given period.
SMA - Simple Moving Average	The average price over a certain period.
EMA - Exponential Moving Average	A weighted moving average.Assigns greater importance to the latest prices, using an exponential weighting approach.
WMA - Weighted Moving Average	A weighted moving average. Assigns different weights based on the data's chronological order.
Moving Average Convergence Divergence (MACD)	Indicates the relationship between two EMA.
Mass Index (MI)	Analyzes the difference between the high and low prices over a certain period.
Trix (TRIX)	Indicates the percentage change in a thrice exponentially smoothed moving average.
Momentum Indicators	
Awesome Oscillator	Compares recent price movements to the movements over a broader period.
Stochastic Oscillator	Evaluates a stock's closing price in relation to its price fluctuations within a specific time frame.
Ultimate Oscillator	Captures momentum across three different time periods(7, 14, 28).
Kaufman's Adaptive Moving Average (KAMA)	A moving average that adds a volatility factor to adjust its sensitivity to price fluctuations.
Williams %R	It measures the relationship between the current closing price and the highest price within a specified period.
Volatility Indicators	
Average True Range (ATR)	Measures price volatility based on the moving average of true ranges.
Ulcer Index	Provides insights into the potential risk by focusing on the extent and length of drawdowns.
Bollinger Channel Percentage Band	Provides insights into the current price's deviation from the average.
Bollinger Channel Band Width	Helps identify periods of increasing or decreasing volatility.
Volume Indicators	
Chaikin Money Flow (CMF)	Measures the amount of Money Flow Volume over a specific period.
Money Flow Index (MFI)	Uses price and volume data to identify overbought or oversold conditions in a stock.
Force Index	Measures how strong the buying or selling is. High values mean prices are increasing, and low values mean the opposite.

Table 3: Description of Technical Indicators

2.2.3 Fundamental Indicators

We sourced our fundamental indicators from Refinitiv Workspace, drawing inspiration from Emir et al. (2012) and Namdari & Li (2018). Another influential paper in our research was Filipović & Khalilzadeh (2021), which delved deep into analyzing a vast array of indicators (172 in total) to pinpoint which ones significantly impact predicting stock price volatility. Interestingly, the bid-ask spread was particularly notable from all the indicators the paper examined. Given its importance, we have decided to include the bid-ask spread among the indicators highlighted by the first two papers, in total, 17 fundamental features. This led us to include metrics like the current and quick ratios, price-to-book ratio, and price-to-cash ratio in our study. We intend to discern whether these slightly varied features might offer a fresh perspective on our research.

It is worth noting that some of the fundamental indicators are only updated quarterly or annually. We delayed non-daily characteristics according to their frequency to ensure our analysis avoids forward-looking biases (Avramov et al. (2023)). By doing so, we ensure that we only use values that were publicly available at that point in time. Subsequently, we converted all variables to a daily frequency using linear interpolation (Nousi (2021) and Koukaras et al. (2022)). Specifically, we are filling in the missing values between two already-known values. By adopting this methodology, we effectively eliminate the risk of looking-forward bias since we only utilize data already known to the public.

To better understand our variables, we introduce a brief description in Table 4. We have provided a breakdown of several fundamental indicators for both Apple and Amazon, and illustrated their progression over time in the charts. As can be observed from Table 5 and Figure 5, Amazon's indicators show a wider range of variation compared to Apple's. Significantly, while some of Amazon's indicators possess a negative kurtosis, indicating rarer extreme values, four of Apple's indicators display a positive kurtosis, suggesting a different distribution of extreme values. It is vital for our research to explore how these differences might influence the predictions of each stock's volatility.

Fundamental Indicators	
Name	Description
Liquidity Ratios	
Quick Ratio, Current Ratio, Cash per Share	Measure a company's ability to cover its short-term liabilities with its short-term assets.
Profitability Ratios	
Earnings per Share, Gross Profit Margin, Return on Equity, Net Margin, Cash Earn Return on Equity, Net Sales, Net Sales/Gross fixed Asset	They are used to evaluate a company's ability to generate income relative to its revenue, operating costs, balance sheet assets, or shareholders' equity over time.
Solvency Ratios	
Debt/Equity, Total Liabilities/Total Assets, Debt/Capital	Evaluate the capability of a company to satisfy its long-term financial liabilities.

Table 4: Fundamental Indicators based on `http://www.investopedia.com`

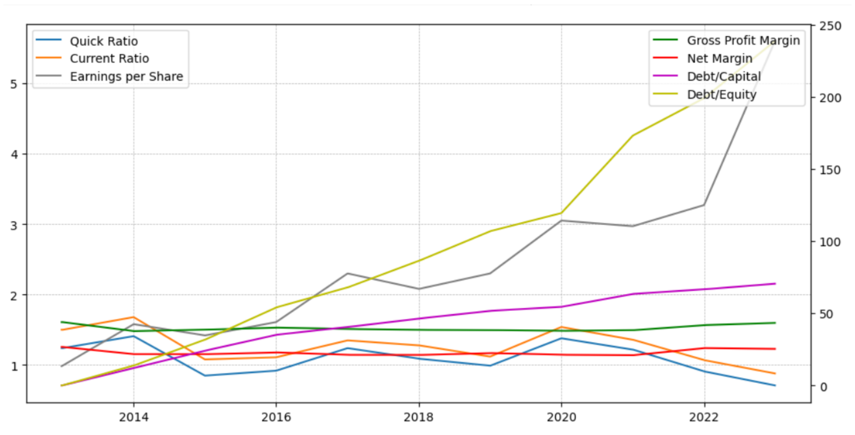
Indicator	Mean	Std	Kurtosis	Skewness	Jarque-Bera
Quick Ratio	1.0987	0.1743	-0.9432	-0.1680	105.359
Current Ratio	1.2783	0.1860	-0.6334	0.1837	56.44
Earnings per Share	2.3867	0.9403	1.2299	1.1019	667.035
Gross Profit Margin	39.3620	1.5414	0.4092	1.2307	652.763
Net Margin	22.4876	1.5486	0.0612	1.2017	606.186
Debt/Capital	42.8834	19.4553	-0.7632	-0.5106	170.689
Debt/Equity	97.0531	66.6465	-0.8835	0.4555	169.137

(a) Apple

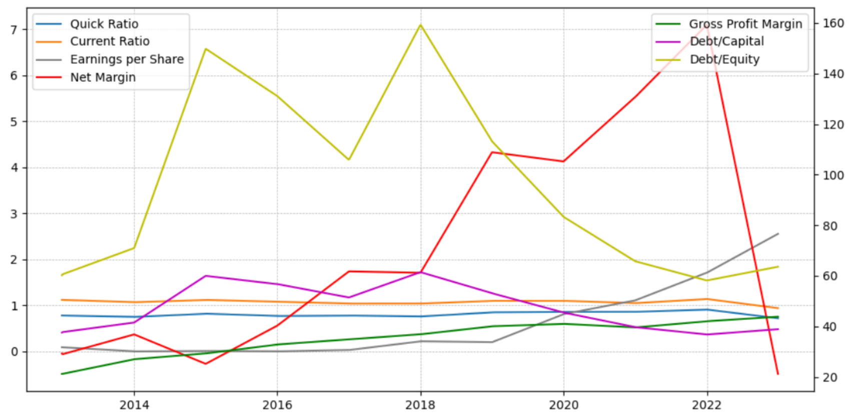
Indicator	Mean	Std	Kurtosis	Skewness	Jarque-Bera
Quick Ratio	0.8110	0.0453	-1.1656	0.4222	217.439
Current Ratio	1.0771	0.0320	2.2595	-1.1455	1083.298
Earnings per Share	0.5412	0.7058	0.4089	1.2662	689.795
Gross Profit Margin	35.5867	5.9539	-0.7163	-0.6450	228.559
Net Margin	2.4892	2.2221	-1.1789	0.4254	221.874
Debt/Capital	48.4223	8.1410	-1.5686	-0.0491	259.124
Debt/Equity	99.9004	32.2254	-1.4822	0.1393	238.638

(b) Amazon

Table 5: Descriptive Statistics for Selected Fundamental Indicators for both Apple and Amazon.



(a) Time plot for Fundamental Indicators for Apple.



(b) Time plot for Fundamental Indicators for Amazon

Figure 5: Time plots with different scales for both Apple and Amazon.

2.2.4 Macroeconomic Indicators

The macroeconomic indicators were sourced from Refinitiv Workspace and consist of 11 measures. This approach is informed by the study in Kirui et al. (2014), which explored the impact of macroeconomic variables on stock volatility. The study highlighted the significance of variables such as Gross Domestic Product, Money Supply M2, Treasury Bill Rate, Exchange Rate, and Inflation. It was revealed that these variables influence stock volatility, which is more pronounced when the variables indicate "bad news". Our research has incorporated a wider array of these variables to gain more comprehensive insights. We have incorporated variables like Visible Balance/GDP, Federal Consumption, and Effective Federal Funds Rate to deepen our analysis. Additionally, insights from Lam (2004) guided our inclusion of more macroeconomic variables, aiming for a comprehensive understanding of their potential influence on stock volatility.

Similarly to the fundamental variables, some macroeconomic indicators are updated less frequently than daily. We have employed a delay according to their frequency and converted all variables to daily frequency using linear interpolation.

Table 7 encompasses a brief description of these variables. In Table 6 and Figure 6, we present a set of macroeconomic variables, both in terms of their descriptive statistics and temporal evolution. Table 6 reveals that all indicators, except Current Account Balance/GDP, show modest fluctuations around their mean values. The relatively low positive kurtosis for these indicators suggests a higher likelihood of extreme values, particularly given the inclusion of the 2020 pandemic period. This observation is visualized in Figure 6, where a sharp decrease is evident around 2020. Prior to this year, several indicators achieved peak values. Our objective in this research is to investigate if these macroeconomic "anomalies" can be captured by our models and eventually influence the predicted stock volatility.

Indicator	**Mean**	**Std**	**Kurtosis**	**Skewness**	**Jarque-Bera**
Visible Balance/GDP	-4.359619	0.297766	1.841766	-0.752610	591.455
Current Account Balance/GDP	-6.924608	2.421669	0.205845	-1.266217	676.916
Effective Federal Funds Rate	0.784333	0.927674	0.875702	1.265083	751.149
10 Year Treasury Bond Yield	2.153069	0.694452	0.097519	-0.163184	12.117
Middle Rate for Prime Rate	3.919048	0.915302	1.127415	1.325223	869.103
3 Month Libor	1.065518	1.043451	1.226422	1.277477	841.296

Table 6: Descriptive Statistics of Selected Macroeconomic Indicators.

Macroeconomic Indicators	
Name	Description
Monetary Indicators	
Money Supply $1\&2$	The total amount of money, including cash and various types of deposits, in circulation within an economy at a specific moment. They serve as crucial indicators of an economy's overall liquidity and financial health.
Effective Federal Funds Rate	The interest rate at which depository institutions lend funds to other institutions overnight. It serves as a benchmark for short-term interest rates.
Middle Rate for Prime Rate	Banks' average interest rate for their most creditworthy customers. This acts as a standard reference point for short-term interest rates..
3-month LIBOR	It signifies the rate at which financial institutions propose to lend to other banks internationally in the interbank market for a three-month duration.
10-year Treasury Bond Yield	Represents the return on an investment in a 10-year U.S. government debt instrument. Is useful for assessing investor sentiment and future interest rate movements.
Other Indicators	
GDP, Government Consumption, Federal Consumption, Visible Balance/GDP, Current Account Balance/GDP	They provide a holistic understanding of economic conditions (at both national and international levels), and evaluate the sustainability of economic activities.

Table 7: Description of Macroeconomic Indicators.

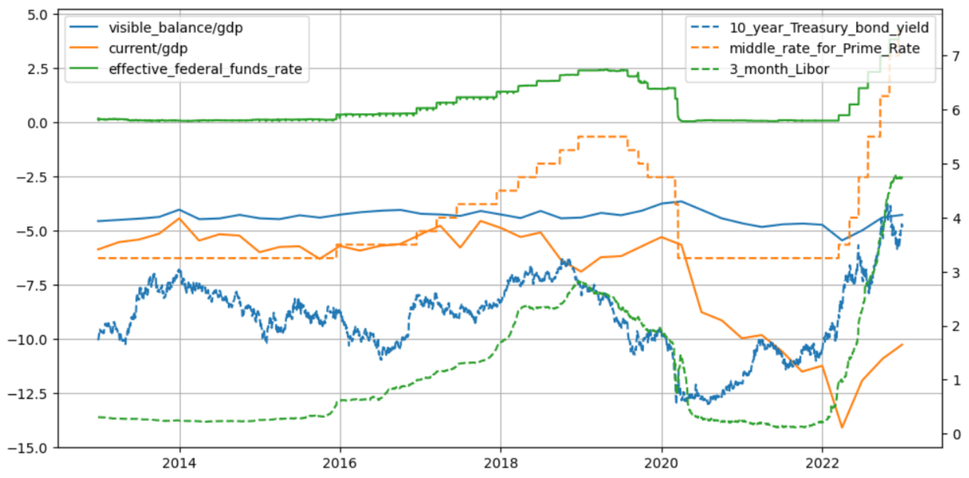


Figure 6: Time plot of the selected Macroeconomic Indicators.

2.2.5 Categorical Variable for the Coronavirus Period

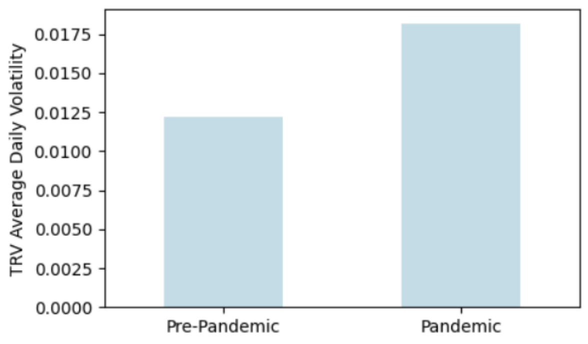
Given the significant impact of the coronavirus on our analysis time frame, we have introduced a dummy variable to capture its effects. This variable is defined as follows[3]:

- 0: Represents the period before the pandemic.
- 1: Represents the period during the pandemic.

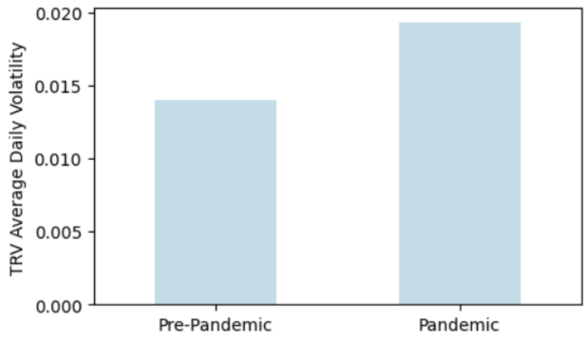
We marked January 30, 2020, as the starting date for this categorization, extending to the end of our interval, i.e., 01/01/2023. On this day, the World Health Organization officially declared the rapidly spreading COVID-19 outbreak a Public Health Emergency of International Concern (Source: News-Medical.Net).

To assess the impact and validate our decision regarding the separation, Figure 7 illustrates the average mean of the True Realized Volatility across the distinct phases: pre-pandemic and during pandemic. The bar plots reveal that the average volatility for both stocks during the pandemic is higher than in the pre-pandemic stage. Specifically, Apple's average daily volatility rose from 0.0125 in the pre-pandemic phase to 0.0175 during the pandemic, reflecting a 40% increase. Similarly, Amazon's volatility experienced a jump from 0.014 to 0.02, a rise of approximately 43%. Furthermore, the slightly more significant proportional rise in Amazon's volatility suggests that the effects of the pandemic may have been more persistent or pronounced for Amazon than for Apple.

[3]In our analysis, we also considered an ordinal variable to differentiate the phases of the pandemic phase. Specifically, we distinguished between the "severe" phase, which covers the initial five months after the onset of COVID-19, and the subsequent "non-severe" phase. However, this variable did not appear to influence the target variable significantly. As a result, we chose a more straightforward approach, employing a dummy variable.



(a) Comparison of Apple's TRV Volatility Across Pandemic Categories.



(b) Comparison of Amazon's TRV Volatility Across Pandemic Categories.

Figure 7: Bar plots showing the average daily volatility of each stock during pre-pandemic and pandemic phases.

2.2.6 Other Indicators

Finally, we have also included the daily volatility, TRV_t, of the Nasdaq stock market index (^IXIC). To determine the most correlated stocks, we computed correlations using a lag of 5 days[4]. Based on this, from the Nasdaq (via Yahoo Finance), we identified Microsoft as highly correlated with Apple and Google with Amazon, using their log returns.

2.2.7 News Articles and Sentiment Analysis

As part of our thorough analysis, we include news articles as an essential data source. We gather 4,845 headlines for Apple and 4,734 headlines for Amazon from the financial website Seeking Alpha, covering the period from January 1, 2013, to January 1, 2023, and using Octoparse for data retrieval.

To extract sentiment information, we utilize two dictionaries: the widely-used Harvard IV-4 Dictionary, which includes an extensive range of words and finds application in diverse fields, such as politics and market research, and the Loughran-McDonald Financial Dictionary, specifically designed for the finance domain (Li et al. (2020)). Both dictionaries have a set of pre-computed sentiment scores for words, where each word has a "positive", "negative", or "neutral" sentiment label.

[4]For further details, see Section 4.1.3.

Sentiment analysis is conducted using the Sentiment Analysis documentation for Python by DeRobertis (2020), involving tokenization and assigning scores to each headline. In particular, we split the headline into individual words ('tokens') and get the overall score of each headline based on the sentiment scores of its constituent words. The overall score is called polarity, and it is computed by the formula: Polarity= (Pos-Neg)/(Pos+Neg), where "pos" is the sum of the positive words of the headline, while "neg" is the sum of the negative expressions. This final score can range from -1 to 1, where -1 indicates negative sentiment, 1 positive sentiment, and 0 neutral.

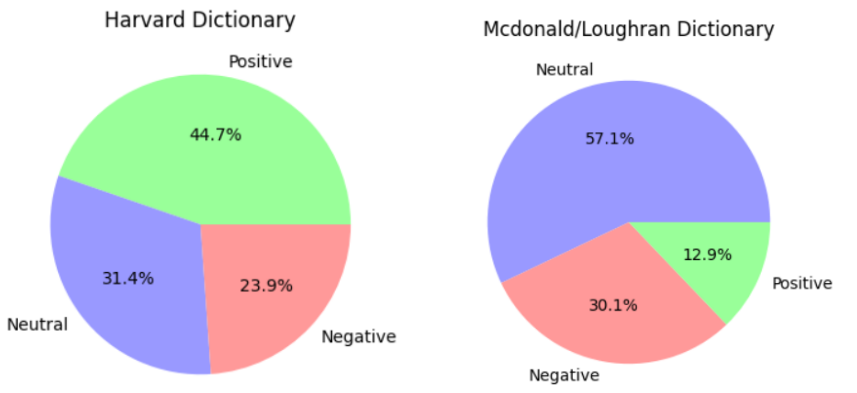
In cases where multiple headlines exist for a given day, the average score of these headlines is computed. We employ the forward fill method to handle missing values, replacing gaps with the most recent non-missing value observed before it.

In addition to the daily sentiment scores, we conduct a rolling mean analysis for both dictionaries over 3, 5, and 7 days. These values are then incorporated into our final dataset. This step aims to explore different time horizons and determine which interval most significantly impacts our target variable.

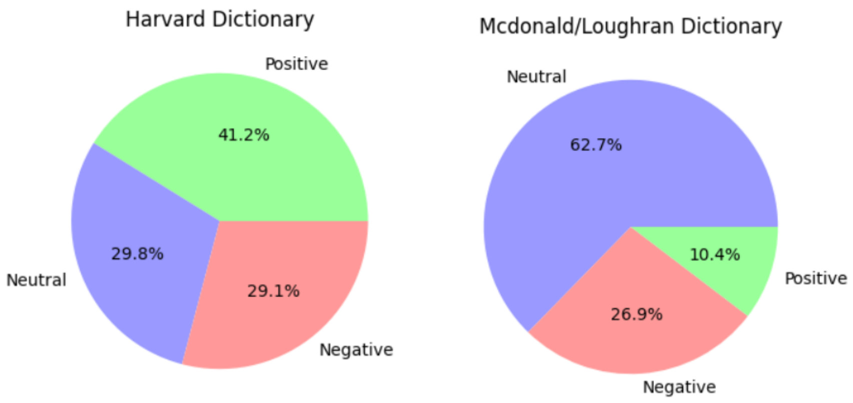
To gain deeper insights into the two dictionaries and better understand their characteristics, we conducted descriptive statistics on the daily sentiment scores for both the Apple and Amazon datasets, as summarized in Table 8. We visualized the sentiment distribution using pie charts, as shown in Figure 8. Additionally, to provide a smoother representation of sentiment trends, we assessed the frequency distribution of the 7-day moving average of daily sentiment scores, depicted in Figure 9.

Stock	Dictionary	Mean	Std	Skewness
Apple	Harvard	0.152177	0.575011	-0.215544
	Loughran/Mcdonald	-0.107653	0.412251	-0.161053
Amazon	Harvard	0.081339	0.590184	-0.151637
	Loughran/Mcdonald	-0.103072	0.370536	-0.466292

Table 8: Descriptive statistics for daily sentiment scores.

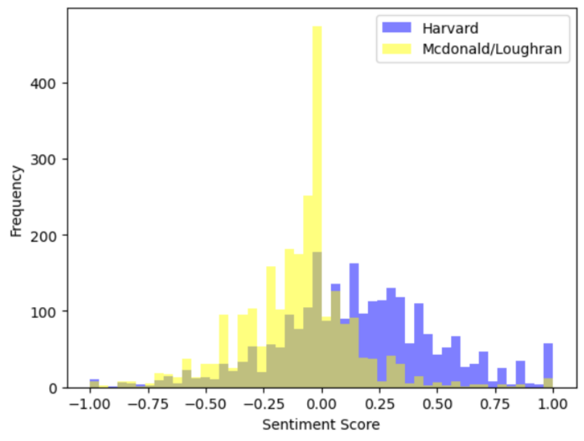


(a) Apple Stock

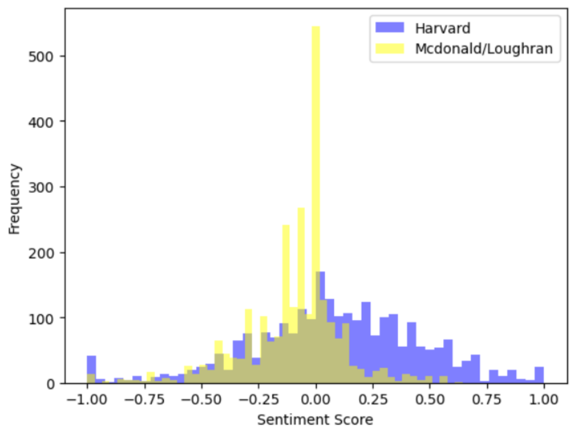


(b) Amazon Stock

Figure 8: Sentiment Distribution Pie Charts for Apple and Amazon Stocks.



(a) Apple Stock Histogram



(b) Amazon Stock Histogram

Figure 9: Frequency Distributions of 7-day Moving Average Sentiment Scores for Apple and Amazon Stocks.

Given the above, it is evident that there are significant differences in sentiment scores generated by the two dictionaries for both stocks. As seen in Table 8, the mean sentiment score from the Harvard dictionary for both stocks is positive, which can be translated to a generally optimistic sentiment for headlines. In contrast, the Loughran/Mcdonald dictionary produces a negative mean sentiment for Apple and Amazon.

A closer look at their standard deviations reveals that the sentiment scores for the former dictionary exhibit higher variability for both companies when compared to Loughran/Mcdonald results. This suggests that the Harvard dictionary encompasses a broader spectrum of sentiment scores. This observation is further supported by Figure 9, where the blue area (representing the Harvard dictionary) is more expansive and extends further along the x-axis than the Loughran/Mcdonald dictionary.

The skewness values provide additional insights. For Amazon, the Loughran/Mcdonald dictionary has a notably higher negative skewness (-0.466) than the Harvard dictionary, suggesting a higher concentration of negative sentiment scores.

These findings are further visualized in the sentiment distribution pie charts (Figure 8). The Harvard dictionary assigns the highest percentage for both companies to positive sentiments, with the pie being divided fairly among the sentiments. In contrast, the Loughran/Mcdonald dictionary allocates only about 10% to positive articles. The remaining 90% is dominated more by neutral sentiments followed by negative ones, indicating a slightly negative bias in the sentiment scores.

Given their differences and to capture the dynamics of both dictionaries; the final dataset is split into two different datasets, each using only one dictionary. Subsequently, we proceed with predictions for both datasets and take the average of their predictions. This approach allows us to gain insights from both dictionaries and their respective sentiment scores in our predictive analysis.

3 Methodology

In our research methodology, we explore a variety of models to predict daily stock volatility. Specifically, our investigation encompasses the ARCH and GARCH models, which utilize lags of daily returns, the ARIMA, feed forward neural network, and long short-term memory (LSTM) models that primarily focus on lags of daily volatility, a naive model serving as a benchmark, and a more advanced analysis with the multivariate LSTM and random forest, designed to delve into the effects of multiple indicators on our target variable. A detailed exposition of each of these models, along with our specific approaches and implementations, will be presented in Section 4.

3.1 Data Splitting and Hyperparameter Optimization

When it comes to predicting stock volatility, two commonly used techniques are the rolling and recursive window approaches. Both cases categorize the data into training, validation, and test sets.

In the rolling window approach, i.e., fixed window, each set consistently shifts forward in time while the sample size remains constant. This efficient method incorporates the most recent data in each window. Additionally, it allows us to explicitly analyze and assess the performance of the model for each distinct period.

In the recursive window approach, i.e., expanding window, the training set gradually increases, keeping previous observations. For example, the initial training set could include the first year of data from the dataset, while the updated training set would cover both the first and second years.

While the recursive window approach can be computationally expensive, an unreported robustness analysis by Filipović & Khalilzadeh (2021) found that "there are no changes in the results if we use an expanding window for the training sample instead of a fixed size window"(p.23). As a result, in this analysis, we will use the rolling window approach (or fixed-size window).

The dataset will be divided into seven subsamples, each with a training set of 3 years and a test set of 1 year. The first subsample will consist of the period from 01-01-2013 to 31-12-2015 as the training set and the period from 01-01- 2016 until 31-12-2016 as the testing set. Subsequent subsamples are obtained by rolling the periods forward by one year while maintaining the same interval lengths. This rolling window approach is illustrated in Figure 10, and the exact intervals for each window are detailed in Table 9.

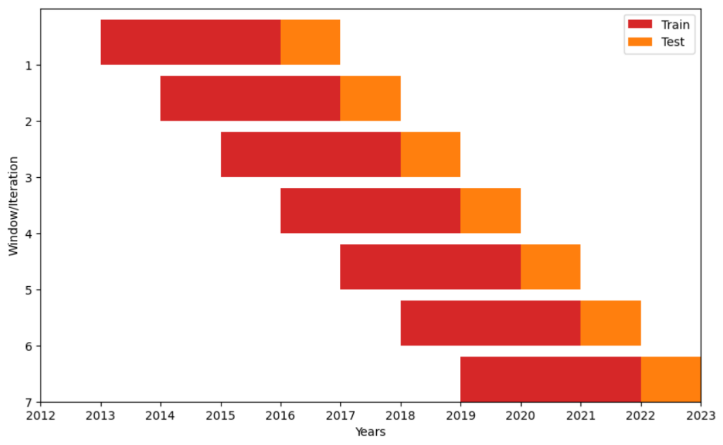


Figure 10: Rolling Window Approach.

Windows	Training Period	Testing Period
1	01-01-2013 to 31-12-2015	01-01-2016 to 31-12-2016
2	01-01-2014 to 31-12-2016	01-01-2017 to 31-12-2017
3	01-01-2015 to 31-12-2017	01-01-2018 to 31-12-2018
4	01-01-2016 to 31-12-2018	01-01-2019 to 31-12-2019
5	01-01-2017 to 31-12-2019	01-01-2020 to 31-12-2020
6	01-01-2018 to 31-12-2020	01-01-2021 to 31-12-2021
7	01-01-2019 to 31-12-2021	01-01-2022 to 31-12-2022

Table 9: Training and Testing Window Intervals.

We carried out hyperparameter tuning for a more nuanced model adjustment, adjusting various parameters across models. This involves searching for the optimal values of multiple parameters that deliver the best model performance based on a specified metric. Hyperparameter tuning is crucial, as it points to the optimal parameters with which we can evaluate the best model on our out-of-sample set.

To effectively perform this tuning, we divided our training data further into a training subset and a validation subset[5]. This division allowed us to test different settings on the validation subset, determining which configurations work optimally before applying them to the final test set. We optimized these settings by selecting the model with the lowest Root Mean Squared Error (RMSE[6]) for the non-linear models. For the statistical models, we utilized different criteria: for the ARIMA model, we considered the lower AIC, while for the ARCH/GARCH models, both the lower AIC and BIC were considered[7]. For the statistical models, along with random forest, a grid search approach was used (Koukaras et al. (2022)), i.e., all combinations of different values of the parameters were examined. Conversely, for the neural network models, the computational demands necessitated using a random search. This means that only a random subsample was tested instead of examining all parameter combinations. These methods are straightforward and intuitive, making them easy to implement and understand.

[5]Further details are being discussed below.

[6]It is defined as the square root of the average of the squared differences between the predicted and actual values. Further details are being discussed in Section 3.3.

[7]We chose these criteria, as AIC and BIC are recognized for their effectiveness in model selection for statistical models, providing a balance between goodness-of-fit and model complexity.

For fine-tuning purposes, we utilized a method known as Rolling-Origin Time Series Cross-Validation (TSCV), Hyndman & Athanasopoulos (2018) (illustrated in Figure 11). This method is particularly suited for time-series data, ensuring the chronological order is maintained during the evaluation. By employing TSCV, we could perform hyperparameter tuning and grid search within the training set's range, preserving the data's time order.

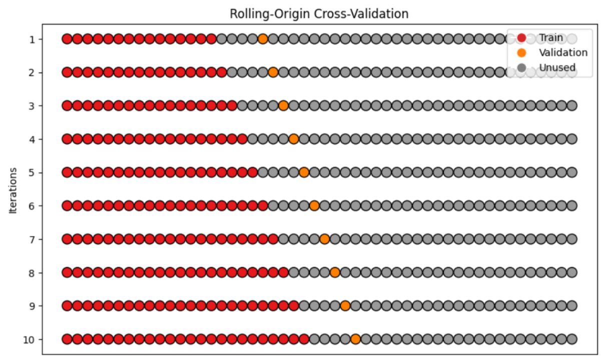
In this arrangement, we primarily dealt with two sets of data within each window: a training set (which was further split for tuning purposes) and a test set (used for final evaluation).

Given the computational intensity of the machine learning models (FFNN, LSTM, and random forest), we adopted a fixed-size TSCV. For these models, within each training set, which consisted of 783 total observations, we identified the initial 'n' observations as the training subset, with 'n' also representing the number of lags used as inputs for the model. That means the model uses previous 'n' data points to make predictions. The optimal value of 'n' was determined through experimentation, where we tested various sizes: 10, 15, 22, 30, 40, 60, and 100 observations, and by examining which value correspond to the lowest RMSE. We identified the optimal size, n=30, and set the (n+5)th observation as our validation point and predicted data point for the test set, maintaining this fixed window size throughout.

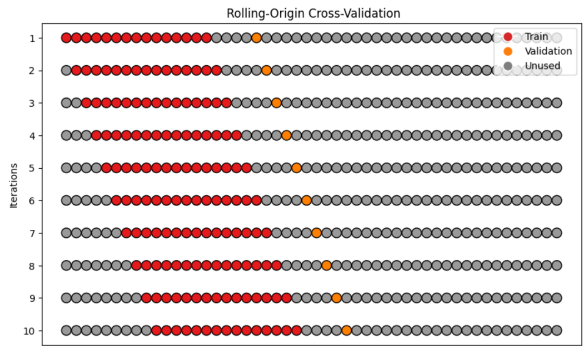
On the other hand, we employed a gradually increasing window approach for the statistical models. For the ARIMA model, using initial 'n' observations as the training subset, we forecasted the volatility for the 5th day ahead and compared it directly with the actual volatility on the (n+5)th day, which acted as our validation point for the training part. For the ARCH and GARCH models, starting with the initial 'n' observations, we utilized the model to forecast the volatility for the next five days. We then took the average of these 5-day forecasts as our single predicted value[8]. This average was contrasted with the actual volatility on the (n+5)th day, serving as the validation point for these models. In both scenarios, with each iteration, we expanded the training window by one observation and shifted our validation point forward, ensuring that we maintained the temporal order of the data. In this context, the 'n' value symbolizes the sample size of the dataset being utilized at each step, while its initial setting was optimized by being n=60. Thus, the 'n' value here does not play the role of the lag values; the lag values used as inputs are fine-tuned parameters.

This process, known as rolling-origin cross-validation, allows for robust validation of our models across different periods (Figure 11).

[8]Further details are presented in Section 4.1.3.



(a) Gradually Increased Rolling-Origin



(b) Fixed Size Rolling-Origin.

Figure 11: Rolling-Origin Cross Validation Techniques.

3.2 Data Preprocessing and Transformations

Data preprocessing is crucial in any machine learning modeling task. It ensures that models receive data in a format optimized for training and prediction. In our analysis, we utilized the following preprocessing techniques based on the requirements of the specific models:

We applied standardization within each window for the feed forward neural network (FFNN). Specifically, we first computed the mean and standard deviation for our target variable, TRV, in the training set. Using these values, we standardized the training and testing sets within each window, following the method outlined by Filipović & Khalilzadeh (2021). This involves subtracting the mean from each training set observation and dividing the result by the standard deviation. We applied an analogous transformation to the testing sets within the same subsample. This approach mitigates the impact of outliers and is expected to enhance model performance.

We experimented with several data transformations for the recurrent neural network (the LSTM model), including standardization and robust scaling. Ultimately, we found normalization — transforming the data into a range of (0, 1) — to be the most suitable for the LSTM.

Unlike the neural network models, the random forest algorithm did not require the input data to be transformed since this model is inherently robust to different scales of the features.

3.3 Evaluation Metrics and Feature Interpretation

As described in Section 3.1, we conducted our model performance evaluation using a train-test split approach. The dataset was divided into two sets: the train set, further split into train and validation subsets, and the test set. Within the train set, we implemented a hyperparameter tuning to obtain the best model in terms of a measure. In this section, we are going to analyze this point further.

For the statistical models, we trained them and tuned the parameters using the validation set. The parameters' combinations, determined through grid search, were evaluated based on two key metrics: the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). Both these criteria are estimators for a model's relative information loss, with lower values indicating superior model performance. It is important to note that the BIC typically favors simpler models than the AIC, due to its more significant penalty for model complexity. More precisely:

$$\text{AIC} = 2k - 2\ln(\hat{L}) \tag{4}$$

where:

k is the total number of parameters in the statistical model,

$\hat{L}$ signifies the highest value achieved by the likelihood function of the model.

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L}) \tag{5}$$

where:

n is the number of observations or data points,

k is the total number of parameters in the statistical model,

$\hat{L}$ signifies the highest value achieved by the likelihood function of the model.

In contrast, for the non-linear models, parameter tuning was executed using a random search, with the primary evaluation metric being the Root Mean Squared Error (RMSE) on the validation set. RMSE, which measures the average squared differences between the predicted and actual values, is particularly beneficial for non-linear models because of its sensitivity to large deviations, ensuring the predictions are close to the actual values.

Once the optimal model parameters were discerned, we applied these models to our test set to evaluate out-of-sample performance. These metrics were also computed for the training set to track potential overfitting or underfitting. Specifically, a model that shows high performance on the training set but underperforms on the test set might be experiencing overfitting, indicating that it has learned the training data too closely and lost generalization capabilities. Conversely, a model that performs poorly on both may be underfitting, suggesting it has not captured the underlying pattern of the data.

The metrics used for these evaluations include:

- **Root Mean Squared Error (RMSE)**: It is defined as the square root of the average squared differences between the predicted and actual values. The formula is given by

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{6}$$

- **Mean Absolute Error (MAE)**: It is the average of the absolute differences between the predicted and actual values. The formula is given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{7}$$

- **Mean Absolute Percentage Error (MAPE)**: It is the average ratio of the absolute differences to the actual values. The formula is given by

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{Y}_i|}{|Y_i|} \tag{8}$$

  Where $\overline{Y}$ is the mean of the observed data, $\hat{Y}$ the predicted value, and Y the actual/observed value (for equations 7 to 9).

- **Skill Score**: It is a measure used to compare the performance of a model to a naive/baseline model. The Skill Score for Root Mean Squared Error (RMSE) is computed as the ratio of the RMSE of the naive model's RMSE to the more complex model. A Skill Score greater than 1 indicates the model outperforms the naive model, while a value less than 1 indicates the opposite. The formula is given by:

$$\text{Skill Score} = \frac{\text{RMSE}_{\text{Naive}}}{\text{RMSE}_{\text{Model}}} \tag{9}$$

Following the performance evaluation, we delved into understanding the underlying factors contributing to our model predictions by performing a feature importance analysis for both the LSTM and random forest models. Our primary objective was to discern the significance of different features in predicting volatility, from which we outlined the top 20. Multiple factors influenced the decision to not include all features. First, limiting the number of features improves computational efficiency. Secondly, we aim to mitigate the risk of overfitting by selecting fewer features—approximately 1/3 of the total. This approach helps exclude potentially noisy indicators and retain only the most informative ones.

We incorporated all available variables and conducted a feature importance analysis using SHAP values (SHapley Additive exPlanations). For the LSTM model, we employed the DeepExplainer tool from the SHAP library. The SHAP method is grounded in game theory, using Shapley values—a concept that assigns a value to each player (or, in this case, feature) based on their contribution to a cooperative game—to quantify the contribution of each feature to every prediction made by the model. DeepExplainer is tailored for deep learning models like LSTM, enabling us to assess the influence of input features on the model's predictions. For this process, we divided the training set into two segments: 80% for model training and hyperparameter tuning, as detailed earlier, and the remaining 20% for computing the SHAP values.

Similarly, we employed SHAP values for our random forest model assessment. Given the ensemble nature of random forests, SHAP values provide an intuitive measure to determine the importance of each feature. The same procedure was followed here. Each SHAP value gives insight into how much each feature contributes to changing the model's prediction for an individual data point compared to the forecast based on the dataset's mean.

By following this comprehensive methodology, we hope to provide robust predictions of daily stock volatility and identify the most influential variables.

4 Model Theoretical Basis and Implementation Techniques

In this section, we explore both the foundational theories underlying the models chosen for our study and the details of their practical implementation. Our discussion builds upon the preliminary introductions presented in Sections 2 and 3, offering a deeper dive into the nuances of statistical and non-linear models.

Three models from the statistical spectrum have been chosen: the autoregressive integrated moving average (ARIMA), autoregressive conditional heteroskedasticity (ARCH), and its extended version, the generalized ARCH (GARCH) models. Firstly, the ARIMA model is notable for its simplicity and robust predictive capabilities. It is often used as a benchmark in papers and research studies when compared with more complex models (for example, Zou & Qu (2020)). Our project selected it due to its simplicity and ease of interpretation. A distinct feature of our utilization of ARIMA is its employment of lags of daily volatility (TRV) as input. Next, we have the ARCH and GARCH models, which are particularly suited for forecasting volatility. They use lag returns to estimate conditional variance, a feature that makes them common picks in financial literature. These models are also relatively straightforward to interpret. Various papers have explored these statistical models for predicting stock volatility, as seen in Miswan et al. (2014), Hansen & Lunde (2005).

Our analysis continues with more complex, non-linear models. From the spectrum of machine learning, three models are selected: two from the area of artificial neural networks (ANN)—namely, the feed forward neural network and the long short-term Memory (LSTM)—and a tree-based model, the random forest. ANNs are widely regarded as one of the most powerful tools in machine learning, mainly known for their capability to recognize complex relationships between target and independent variables. The key strength of neural networks lies in their flexibility, achieved through the interconnection of multiple layers, giving rise to the term "deep learning." One notable advantage of LSTM is its consideration of the chronological order of the inputs, making it favorable for time-series analysis. Multiple papers have examined its efficacy in predicting stock volatility, among them Filipović & Khalilzadeh (2021), Dixit et al. (2013) and Kyoung-Sook & Hongjoong (2019). For both neural network models, the input is lags of daily volatility, while LSTM will also be employed as a multivariate model, inputting multiple indicators to predict daily volatility. This decision stems from its aforementioned advantage concerning time-series data. Random forest was chosen as the second multivariate model due to its robust predictive capabilities, enabled by its aggregating outputs mechanism. This model also benefits from inputting multiple independent variables, providing a comprehensive analysis in predicting daily volatility. Numerous papers have also analyzed the random forest model, including Luong & Dokuchaev (2018), Bouri et al. (2021).

4.1 Statistical Models

4.1.1 ARIMA model

The first statistical model we will investigate is the **autoregressive integrated moving average model, ARIMA(p,d,q)**. This model combines three distinct components: the autoregressive model, AR(p); the integrated part (the number of differencing d, to establish stationarity in the series); and the moving average model, MA(q). Differencing a series involves subtracting the current value from the previous one, helping to remove any trend in the series, thereby working towards achieving stationarity, a crucial assumption in time series analysis. These three parameters are the parameters that were fine-tuned via the auto arima model, while the range of (1,5) was utilized within the grid search. This model is highly useful for estimating future values in a stationary time series data by considering the influence of its past values.

The below equation (10) defines the ARMA part of the ARIMA model:

$$X_t = \omega + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} a_i \varepsilon_{t-i} + \varepsilon_t \tag{10}$$

In this equation, X_t represents the value we aim to predict at time t, ω is the constant, while ε_t denotes the white noise. The ARIMA model consists of two following main components.

The first component, $\sum_{i=1}^{p} \varphi_i X_{t-i}$, represents the autoregressive model, AR(p). It considers the impact of the previous p values of the time series on the current value. The coefficients φ_i indicate the significance of each past value in predicting future values.

The second component, $\sum_{i=1}^{q} a_i \varepsilon_{t-i}$, represents the moving average model, MA(q). It considers the influence of the lagged error terms, ε_{t-i}, on the current value. The coefficients α_i determine the effect of each lagged error term, reflecting their contribution to the prediction accuracy.

In our research, we used lags of the dependent variable, i.e., TRV, up to time t, to predict the value of TRV_{t+5}. The ARIMA model would consider both the most recent lags and their associated error terms from the TRV series up to time t to provide a forecasted value for the TRV five steps ahead.

We employed maximum likelihood for estimating the ARIMA model parameters, assuming Gaussian errors. This assumption is common in time series analysis and is supported both theoretically and empirically. While the consistency of Maximum Likelihood Estimation in the context of ARCH/GARCH models is well established, as demonstrated by Bollerslev & Wooldridge (1992) (Engle & Patton (2001)), a similar logic applies to ARIMA models. Empirically, we validated this assumption in our data by plotting the standardized residuals and confirming their normal distribution (see Figure 20/ Appendix).

Hypothesis testing is crucial in estimating and interpreting the ARIMA model coefficients. The tests are structured as follows:

- **Null Hypothesis** (H_0): For a given coefficient (ϕ_i or α_i), the null hypothesis posits that the coefficient is equal to zero, indicating no significant effect of the corresponding autoregressive or moving average term on the time series. The null hypothesis for the constant ω suggests that the model has captured all the underlying patterns in the time series without any remaining unexplained components.
- **Alternative Hypothesis** (H_A): The alternative hypothesis posits that the coefficient ϕ_i or α_i is different from zero, indicating a significant effect of the corresponding term on the time series. The alternative hypothesis for the constant ω posits that there is a remaining value unexplained from the model.

To test these hypotheses, the t-statistics obtained from the model estimation are utilized. A large absolute value of the t-statistic suggests that the estimated parameter is significantly different from zero, thus providing evidence against the null hypothesis in favor of the alternative hypothesis. The results and interpretations of these tests will be indicated in the subsequent results section.

By combining the autoregressive and moving average components and applying them to a differenced series, the ARIMA model provides a comprehensive framework for capturing the underlying patterns and dynamics in time series data.

4.1.2 ARCH model

The **autoregressive conditional heteroskedasticity, ARCH(q) model**, extends the concept of the autoregressive model by incorporating the conditional heteroskedasticity, which refers to the time-varying volatility of the error term in a time series. In other words, ARCH models aim to capture the changing patterns of volatility observed in many financial and economic time series.

The ARCH(q) model (proposed by Engle (1982)) is a statistical model for time series data, mainly used in financial and econometric contexts to capture these volatility patterns. It models the current error term based on its past errors.

Given r_{t}, the return at time t, and ε_{t} the error term at time t, we can write $r_t = \mu + \varepsilon_t$, where μ represents the mean expected return.

The error term ε_t can be further decomposed into $\varepsilon_t = \sigma_t z_t$, where z_t represents an i.i.d. process with a mean of 0 and a variance of 1, and σ_t signifies the time-dependent standard deviation of the error term.

The time-varying standard deviation σ_t itself can be modeled as an ARCH(q) process:

$$\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2, \quad \text{where} \quad \omega, \alpha_i > 0 \tag{11}$$

For the conditional variance to be a weakly stationary process, it is required that:

$$\sum_{i=1}^{q} \alpha_i < 1. \tag{12}$$

In equation (11), ω is a constant, and the α_i terms are coefficients representing the influence of past error terms on the current volatility. The parameter q indicates the number of past error terms included in the model, which was also fine-tuned within a range of (1,5).

Through this analysis, we considered the lags of log-returns of the stocks up to time t to forecast the conditional variance up to 5 days ahead[9].

The analysis requires the time series data used as input - specifically, the log returns - to be stationary. We use the Augmented Dickey-Fuller (ADF) test, established in 1979 to verify this assumption. If the ADF test produces a p-value exceeding 0.05, this suggests that the series is non-stationary, necessitating transformation. Given that our data met the stationarity

[9]A detailed description is included in Section 4.1.3.

condition without needing transformation, we proceeded with fitting the models.

In estimating the ARCH(q) model, hypothesis testing is crucial in determining the significance and influence of the model's coefficients on the conditional variance. The hypothesis tests are framed as follows:

- **Null Hypothesis** (H_0): For a given coefficient (α_i or ω), the null hypothesis posits that the coefficient is equal to zero. For α_i, this implies no influence of the corresponding past error term (ε^2_{t-i}) on the current conditional variance (σ^2_t). For ω, it suggests no constant long-run variance component in the model. This can indicate that the ARCH model has effectively captured the underlying patterns of the conditional variance without the need for a constant variance component.
- **Alternative Hypothesis** (H_A): The alternative hypothesis posits that the coefficient (α_i or ω) is different from zero. For α_i, this indicates a significant influence of the past error term on the current conditional variance. For ω, it suggests the presence of a constant long-run variance component.

Like in the ARIMA model, the t-statistics obtained from the model estimation are utilized to test these hypotheses.

For the estimation of the parameters, we conducted maximum likelihood as discussed in ARIMA (3.1.1)

The main idea of the ARCH model is that periods of greater volatility in the past, characterized by more significant error terms, generally lead to phases with heightened volatility. In comparison, periods of lower volatility tend to be followed by periods of reduced volatility. This pattern, known as 'volatility clustering,' is a common characteristic observed in financial time series data, hence a suitable candidate for our analysis.

4.1.3 GARCH model

A limitation of the ARCH model is its inability to capture the persistence of conditional variance. This led to the introduction of the **generalized autoregressive conditional heteroskedasticity, GARCH(p,q) model** by Bollerslev (1986). The GARCH model incorporates a moving average component into the ARCH equation for time-varying standard deviation. Specifically, equation (11), becomes:

$$\sigma^2_t = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon^2_{t-i} + \sum_{j=1}^{p} \beta_j \sigma^2_{t-i}, \quad \text{with} \quad \sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j < 1 \quad \text{and} \quad \omega, \alpha_i, \beta_\mathrm{j} > 0 \tag{13}$$

While p refers to the number of past conditional variances[10].

Like the ARCH model, the time series data utilized in the GARCH model should meet the stationarity assumption. Stationarity of the data was confirmed using the Augmented Dickey-Fuller (ADF) test as detailed in the ARCH model, Section 3.1.1.

[10] An identical approach to the one used with the ARCH method, was adopted for generating forecasts with the GARCH models as well.

The estimation of GARCH(p, q) model parameters was conducted using the method of Maximum Likelihood, similar to the procedure outlined in the ARIMA model section. This method operates under the assumption of Gaussian errors, a foundation validated theoretically and empirically, as previously discussed (3.1.1). Estimating GARCH(p, q) model parameters also involves hypothesis testing to evaluate the significance and influence of the coefficients on the conditional variance. The hypothesis tests for the GARCH model coefficients α_i and β_j are framed as follows:

- **Null Hypothesis** (H_0): For a given coefficient (α_i, β_j, or ω), the null hypothesis posits that the coefficient is equal to zero, indicating no influence of the corresponding past error term (ε^2_{t-i}) or past conditional variance (σ^2_{t-j}) on the current conditional variance (σ^2_t). For ω, it suggests no constant long-run variance component.
- **Alternative Hypothesis** (H_A): The alternative hypothesis posits that the coefficient α_i, β_j, or ω is different from zero, indicating a significant influence of the past error term, past conditional variance, or a constant long-run component on the current conditional variance.

Once more, we test these hypotheses by the corresponding t-statistics obtained from the model estimation.

The GARCH model considers both the impact of the lagged error terms and the lagged values of the conditional variance. By including lagged conditional variances, the GARCH model can capture longer-term dependencies and persistence in volatility, making it a more realistic model for financial time series where volatility shocks persist over time.

Given the nature of TRV calculation, which is an average over five days (equation 1), comparing it with a single-day forecast could lead to discrepancies. This is because the forecasted values from the ARCH and GARCH models represent estimates of the conditional variance (σ^2_{t+h}) generated from return values as inputs. To address this, we modified our forecasting horizon to span five days. Instead of forecasting the volatility for one day ahead, we extended our predictions to include the volatility for days t+1, t+2, t+3, t+4, and t+5. More precisely,

$$\bar{\sigma}_{t+5} = \frac{1}{5}\sum_{h=1}^{5} \sigma_{t+h}, \tag{14}$$

Where:

- $\bar{\sigma}_{t+5}$ is the average volatility over the next five days from day t.
- σ_{t+h} is the predicted conditional volatility on day $t+h$.

By doing so, we captured a short-term period of average volatility.

This adjustment allowed us to compare the average forecasted volatility,$\bar{\sigma}_{t+5}$ derived from the ARCH and GARCH models, with the actual volatility (TRV) at t+5 (equation 1).

Given the modification in the forecasting horizon (=5) for the ARCH and GARCH models, it is essential to maintain a consistent forecast horizon for the 5th day ahead ($t+5$) for all remaining models. This consistency will enable a fair comparison across all models regarding their predictive accuracy and other evaluation metrics

4.2 Artificial Neural Network

4.2.1 Feed Forward Neural Network

Our first approach is based on **"feed forward neural network"** (FFNN). This name stems from the fact that data flows through the network in a single pass without any feedback loops or recurrent connections. A FFNN comprises an input layer, one or more hidden layers, and an output layer. These layers are interconnected with weighted connections. In the case of regression tasks like ours, the output layer contains a single neuron representing the final output or prediction, while the input and hidden layers may contain multiple neurons. As discussed earlier, we use the univariate version of FFNN, while the input consists of our TRV time series.

The figure below (Figure 12), illustrates a FFNN with one and two inputs, one hidden layer containing five neurons, and one output layer. Please note that this simplified network serves for illustrative purposes only.

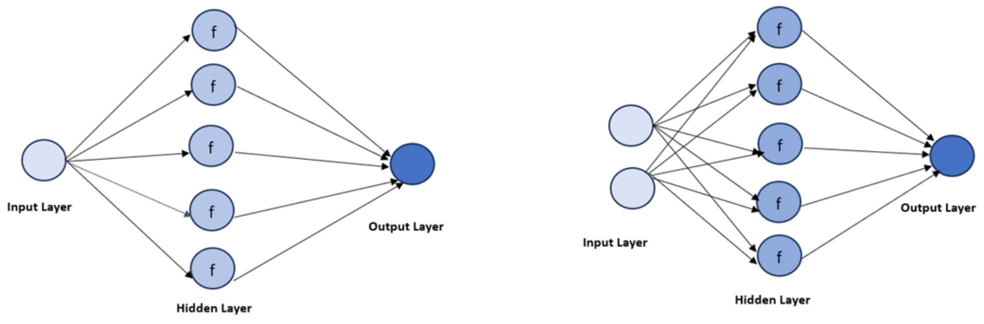


Figure 12: The left Neural Network consists of one input, while the right Network consists of 2 inputs. Each arrow contains a certain weight.

In our research, we consider the lags of the dependent variable, TRV, up to time t, to obtain predictions TRV_{t+5} as input.

In our analysis, we adopted the geometric pyramid structure proposed by Gu et al. (2020), thereby considering networks with up to five hidden layers. The simplest model denoted as NN1, comprises one hidden layer with 32 neurons. The next model, NN2, includes two hidden layers with 32 and 16 neurons, respectively. Subsequently, NN3 consists of three hidden layers with 32, 16, and 8 neurons, respectively. NN4, the fourth model, features four hidden layers with 32, 16, 8, and 4 neurons. Lastly, the most complex model, NN5, incorporates five hidden layers with 32, 16, 8, 4, and 2 neurons, respectively.

In the optimization phase of our models, we employed the Adaptive Moment Estimation (ADAM) algorithm, initially introduced by Kingma & Ba (2014) and effectively used by Ramos-Pérez et al. (2019). Utilizing TensorFlow's ADAM implementation, we set the learning rate at the default value of 0.001, enabling a methodical convergence to the optimal parameters during the training process.

As mentioned earlier, the procedure is straightforward. The input layer receives the initial data or features, with the number of units depending on the variables considered. The data then passes through the hidden layers, where each neuron performs a weighted sum and applies it to an activation function. Following the path covered by previous research, including the work of Gu et al. (2020), we adopted the Rectified Linear Unit (ReLU) function as an activation function, which enables the modeling of non-linear connections between input variables and the output. It is defined as

$$f(x) = \begin{cases} 0, & \text{if } x < 0, \\ x, & \text{otherwise.} \end{cases} \tag{15}$$

The ReLU function serves as a thresholding mechanism in neural networks. For inputs below zero, it outputs 0, effectively rejecting negative values. Meanwhile, for positive inputs, it simply lets them pass through unchanged. This simple yet effective behavior introduces the necessary non-linearity that empowers neural networks to capture underlying data patterns.

To elaborate, let's denote by $x^{(0)}$ the input layer and $x_1^{(0)}$ the unique input. Extending this notation, $\mathrm{x}^{(1)}$ is the hidden layer and $x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)}, x_5^{(1)}$ its neurons. The weights are denoted by w, while the bias term is represented by θ. Each neuron in the hidden layer linearly combines information from the input and applies a nonlinear activation function f. For example, for the third neuron, the procedure would be as follows:

$$x_3^{(1)} = f(\theta_3^{(0)} + w_3^{(0)} x_1^{(0)}), \tag{16}$$

where $\theta_3^{(0)}$ is the bias for the corresponding neuron and $w_3^{(0)}$ is the weight between the input and the third neuron. This function would be more complicated if the input layer contains two units (or more). For the two units, for example, it would be :

$$x_3^{(1)} = f\left(\theta_3^{(0)} + \sum_{l=1}^{2} w_{3,i}^{(0)} x_i^{(0)}\right). \tag{17}$$

Finally, these results are linearly aggregated to produce an output forecast y:

$$y = \theta^{(1)} + \sum_{i=1}^{5} w_i^{(1)} x_i^{(1)}. \tag{18}$$

Given this structure, it is evident that the complexity of FFNNs requires careful consideration of numerous parameters. The model in the figure has $(1+1) * 5 + 6$ parameters $= 16$ parameters in total, where 5 are the neurons and $(1+1)$ are their corresponding weights and biases, while 6 are the last weights (neurons connected with the output) with 1 bias. Similarly, the second Neural Network will have $(2+1) * 5 + 6 = 21$ parameters.

These weights are not predetermined but are assigned initial values randomly. In our implementation, the default weight initialization method provided by the Keras library, specifically the glorot uniform initializer, was used. The network learns to adjust these parameters to their optimal values through backpropagation, aided by our chosen optimization algorithm, ADAM. Backpropagation involves calculating the gradient of the loss function, here specifically the Mean Squared Error, with respect to each parameter and updating the parameters in the direction that minimizes the loss. This iterative process allows the network to gradually optimize its parameters and improve performance on the given task.

During the implementation phase of our multi-layer perceptron model, we aimed to enhance the model's generalization capability and prevent overfitting. To achieve this, we considered various hyperparameters. Firstly, we adopted a two-fold regularization approach known as Elastic Net Regularization, combining both L1 (Lasso) and L2 (Ridge) regularization techniques simultaneously. We maintain a balanced regularization effect by adding a penalty term proportional to the absolute value of the weights (L1) and another proportional to the square of the weights (L2).

Moreover, we integrated dropout regularization within our model. Dropout is employed during training, temporarily omitting randomly selected neurons, which minimizes their contribution to the network, making the model more robust against overfitting. We used batch normalization, as introduced by Ioffe & Szegedy (2015) and implemented by Gu et al. (2020), to normalize the outputs of each layer, helping to reduce variations and improve generalization. We significantly reduced the risk of overfitting into the training data by applying dropout and batch normalization after each layer.

Furthermore, we implemented 'early stopping' to prevent the model from becoming too specialized in the training data. With early stopping, we halt the training process if there is no improvement in predicting the validation set over a certain number of iterations, known as epochs. This strategy is crucial for capturing the optimal performance of the model while avoiding overfitting, ensuring its effectiveness in predicting new data. In addition to the number of epochs, the batch size was another consideration in our implementation. The batch size refers to the subset of the dataset that the network processes in a single iteration, which affects both the speed and the quality of the training process.

Given our models' complexity and numerous parameters, we employed a random search method (discussed in Section 3.1) for optimal parameter values. We chose random search because it is efficient and works well with models with many parameters. We did this process 25 times. The specific values included in the random search are detailed in Table 10. By

Parameter	Values
L1	[0.001, 0.01, 0.1]
L2	[0.001, 0.01, 0.1]
Epochs	[4, 8, 32, 16, 50, 100, 200]
Batch Size	[8, 16, 32, 50]

Table 10: Random Search Parameters for Model Hyperparameter Tuning.

combining the techniques mentioned above, we aimed to create a well-regularized and robust forecasting model that can effectively generalize to unseen data, providing reliable predictions for future scenarios.

4.2.2 Long Short Term Memory Neural Network

Our second approach expands upon the recurrent neural network (RNN) by incorporating the **long short-term memory (LSTM)** architecture. Unlike the feed forward neural network, the RNN utilizes feedback loops, which create a cycle where the output of a neuron influences the input of the next time step or becomes an input to itself. Another critical distinction is that the RNN considers the inputs' chronological order, considering the data's temporal nature.

However, the RNN is fragile to a significant drawback known as the vanishing/exploding gradient problem during the backpropagation procedure. This problem arises when the gradients become extremely small, particularly in RNNs with multiple layers, resulting in ineffective weight adjustments and preventing the learning process. This issue is mitigated with the use of LSTM, an extension of the RNN architecture that addresses the vanishing gradient problem and enables more effective learning and weight adjustment.

The basic unit of the LSTM network is called cell and contains three components: the forget gate, the input, and the output gate. As its name indicates, the LSTM cell can remember values over varying periods, both long and short. To better understand the structure and functionality of the LSTM cell, Figure (13)[11] depicts a single LSTM cell and its process.

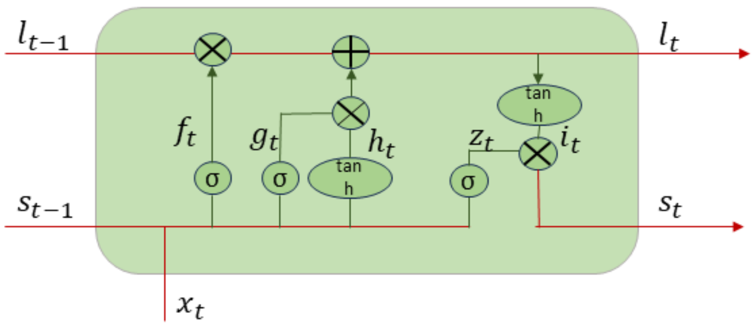


Figure 13: A single LSTM cell

The top red horizontal line represents the Long Term Memory and is called the Cell State, while the bottom red horizontal line represents the Short Term Memories and is called Hidden State. The absence of direct weights on the connections to the Cell State allows the Long Term Memories to flow through the LSTM unit without causing gradients to vanish or explode.

[11]The general approach was adapted from Thu et al. (2021)

The LSTM cell accepts several inputs: x_t the current input vector, which might contain different features; l_{t-1} the previous long-term memory; $\mathrm{s}_{\mathrm{t}-1}$ the previous short term memory. It computes l_t, the current long-term memory, and s_{t} the current short-term memory.

The activation functions here are two, the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$ and the tanh function, $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and capture non-linear dependencies in the data. Consistent with common practices in LSTM design for regression tasks, we utilize a linear activation function in the output layer.

The LSTM cell operation can be divided into stages:
1. **The Forget Gate**: This gate determines what percentage of the $\mathrm{I}_{\mathrm{l}-1}$ is kept. It is defined as:

$$f_t = \sigma\left(w_f\left[s_{t-1}, x_t\right] + b_f\right) \tag{19}$$

2. **The Input Gate**: This gate includes two components (g_t and h_t). They are computed as:

$$\begin{aligned} g_t &= \sigma\left(w_g\left[s_{t-1}, x_t\right] + b_g\right) \\ h_t &= \tanh\left(w_h\left[s_{t-1}, x_t\right] + b_h\right) \end{aligned} \tag{20}$$

3. **The Cell State Update**: The current long-term memory l_t is updated using the outputs from the forget and input gates. More precisely:

$$l_t = f_t * l_{t-1} + g_t * h_t \tag{21}$$

4. **The Output Gate**: This is the final stage, which computes the current short-term memory s_{t} as:

$$s_t = z_t * i_t \tag{22}$$

Where

$$\begin{aligned} z_t &= \sigma\left(w_z\left[s_{t-1}, x_t\right] + b_z\right) \\ i_t &= \tanh\left(l_t\right) \end{aligned} \tag{23}$$

For our research, we initially considered a univariate model. This model takes the lags of the dependent variable, TRV, up to time t to predict TRV_{t+5}. In a later stage, we adopt a multivariate approach, using the lags of multiple features up to time t to forecast TRV_{t+5}.

Our approach included two LSTM layers followed by two dense layers. The first LSTM layer contains several neurons selected via hyperparameter tuning. In contrast, the number of neurons in the second LSTM layer was conditioned to be less than or equal to that in the first layer. To avoid overfitting, the LSTM layers were followed by dropout layers.

Like our feed-forward neural network, we used the Keras library's glorot uniform initializer for random weight initialization. The optimization algorithm adapted for the LSTM was the ADAM optimizer, where the learning rate is maintained at the default setting of 0.001, while the loss function utilized for LSTM was the Mean Absolute Error. This discrepancy between FFNN and LSTM occurred due to the latter's fragility of the extreme values.

The hyperparameters tuned were the number of LSTM units in each layer, the number of training epochs, and the batch size. This random search was repeated 16 times, with each iteration yielding a unique model architecture and a set of hyperparameters. We then compared these models based on their performance on the validation set and selected the model that yielded the lowest Root Mean Squared Error (RMSE).

The hyperparameters for our LSTM model in the random search are listed in Table 11.

Parameter	Values
LSTM Units (Layer 1)	[16, 32, 50, 64,100, 150]
LSTM Units (Layer 2)	[16, 32, 50, 64,100, 150]
Epochs	[8, 16, 32, 50]
Batch Size	[4, 8, 16, 32, 50]

Table 11: Random Search Parameters for LSTM Model Hyperparameter Tuning.

4.3 Tree-Based Method: Random Forest

Random forest is a robust ensemble learning technique applicable to both non-linear regression and classification problems. It operates by constructing multiple decision trees and aggregates their outputs. In the case of regression, it computes the average of the predictions. As a result, it yields a more stable and accurate prediction. Random Forest is also widely recognized for its effectiveness in feature selection, as it helps identify the most significant features contributing to the model's prediction power.

To understand random forest better, let's talk about how decision trees, the building blocks of a random forest, are constructed. Decision trees consist of decision nodes (split nodes) and leaf nodes (or terminal nodes). The figure below (14) illustrates a simple decision tree construction.

Let's denote X as the number of observations in our original data and P as the number of features.

The initiation of a decision tree occurs at the root node, containing the most significant feature among the P features. The "most significant" here signifies the feature that provides the highest data variance explanation. This process continues recursively, generating a tree-like structure of decisions, with each node containing a feature that leads to the best split at that point.

This tree-building procedure continues until specific stopping criteria are reached. One of these stopping criteria is the minimum samples split, a hyperparameter that defines the minimum number of samples required to divide further a node. If the number of samples within a node falls below the specified value, the node is not split further and becomes a leaf node. This approach helps prevent the model from overfitting by not allowing the tree to grow indefinitely.

One flaw of decision trees is that they tend to generate correlated trees since they always start building the structure from the most significant feature. However, this limitation is effectively addressed by the random forest. Random forests vanish this flaw by creating random subsamples from the available features during the tree-building process.

In essence, the random forest algorithm generates multiple decision trees, individually trained on varying subsets of data and distinct sets of features. Incorporating randomness and

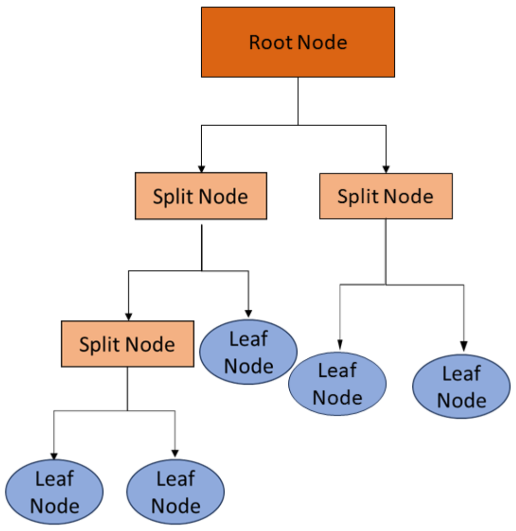


Figure 14: Decision Tree.

diversity into the process ensures that each tree brings its unique perspective to the problem at hand.

The final prediction from the random forest is obtained by computing the average of all individual tree outputs. This ensemble approach results in a more stable and accurate prediction.

More precisely, the procedure is as follows :

Step 1. Set the number of trees B to be created in the Random Forest.

Step 2. For each tree $b = 1, \ldots, B$:

- Draw a bootstrap sample from the original dataset (which allows for sampling of the same observation more than once), creating a new dataset of size X. This new dataset forms the training set for this particular tree.
- Initialize the tree to be built. Identify the number of features, $p < P$, for each split. Typically, p is set to be $P/3$. We generate a tree f_b by considering the most significant features from the p features as the starting point for the decision nodes.

Step 3. After the training process is done, predict an unseen observation, by using all trees and taking the average of their outputs, i.e., $\hat{f} = \frac{1}{B}\sum_{i=1}^{B} f_b(\hat{x})$, where $\hat{x}$ is the new unseen observation.

The random forest in this research, considers the lags of multiple features as input up to time t, to obtain the forecasted TRV_{t+5}.

Among the crucial hyperparameters influencing the random forest model's performance, we adjusted the number of 'estimators,' which determines the number of trees included in the forest.

We also modified the 'max_features' parameter, which determines the maximum number of features evaluated for the optimal split at each node. Another parameter, 'min_samples_split', was fine-tuned to specify the minimum number of samples needed at a node to do a further separation.

The specific values for these hyperparameters that we explored through a grid search are tabulated below:

Parameter	Values
Estimators	[100, 150, 200, 300, 500]
max_features	[3, 4, 7]
min_samples_split	[3, 4, 5]

Table 12: Hyperparameters for Random Forest Model.

5 Results

In this section, we will delve into and analyze our research results. We start by detailing the performance of baseline models for both stocks across various periods or windows. Following this, we contrast these with statistical models, offering a deeper look into their parameters and exploring how we combine GARCH and ARCH model predictions (obtained via AIC and BIC criteria) using the aggregated forecast through exponential reweighting (AFTER) technique. Progressing further, the performance of neural network models is outlined, and these are compared with the baseline models. In Subsection 5.6, our focus shifts to multivariate models, primarily LSTM and random forest. Our approach involves combining forecasts from different news article dictionaries using the Optuna optimization framework and then measuring these combined predictions against the baseline models. For a comprehensive overview, we also present a comparison of all models. We complete our research with a thorough examination of feature importance implemented by the multivariate models.

5.1 Baseline Models

We start our analysis by representing and analyzing the performance of the naive models that we consider as baseline. That is, we assume that the forecasted value for the 5th day ahead, denoted as $\hat{y}_{t+5}$, is the actual value at time t, i.e., $\hat{y}_{t+5} = y_t$. This provides a straightforward baseline for comparing the effectiveness of more complex models.

Table 13 indicates the RMSEs across the different windows for Apple and Amazon. One advantage of considering different windows is that we can examine each period separately and investigate the underlying patterns. Significant findings can be observed for both stocks. At first glance, a variation across windows is observable for both stocks, a range from 0.575 to 1.312 for Apple and from 0.623 to 1.564 for Amazon. This could be due to changing market conditions, events, or other factors that affect the stock prices, thus their volatility. The most interesting period is the 5th window, where the RMSE for both stocks exhibit high values. This is not surprising since COVID-19 impacted the market during our test set. Hence, this abnormal situation could not be captured by the baseline models. An interesting period for Amazon is the 7th window, where we observe the highest RMSE, indicating that the naive approach failed to capture the high variance during this period. Additionally, in most windows, the RMSE of Apple stock is lower than that of Amazon's, suggesting that this simple model struggles to capture the more extreme behavior of Amazon's stock. As seen from the primary statistics and visualizations (Table 1, Figure 1), Amazon exhibits a more volatile trend compared to Apple.

Window	Apple RMSE	Amazon RMSE
1 (01-01-2016 - 31-12-2016)	0.772	0.857
2 (01-01-2017 - 31-12-2017)	0.575	0.851
3 (01-01-2018 - 31-12-2018)	0.761	0.951
4 (01-01-2019 - 31-12-2019)	0.766	0.716
5 (01-01-2020 - 31-12-2020)	1.312	1.178
6 (01-01-2021 - 31-12-2021)	0.619	0.623
7 (01-01-2023 - 31-12-2023)	0.919	1.564

Table 13: Test Root Mean Squared Error (RMSE) for Each Window.

5.2 ARIMA Model Analysis

5.2.1 Results and Interpretation

In Table 14, we present the parameters p, d, and q, representing the order of the models' autoregressive, differencing, and moving average parts, respectively. These parameters were optimized using the *auto arima* function within the training period. For Apple, most windows required the time series to be differenced once (d=1) to achieve stationarity before modeling, except for window 2, where no differencing (d=0) was needed. This suggests that the underlying data patterns for Apple's stock TRV may exhibit non-stationarity. On the contrary, for Amazon, the time series were mostly stationary, with no need for differencing (d=0) in four out of the seven windows. This indicates a fundamental difference in the time series properties between Apple and Amazon.

Window	Apple	Amazon
1 (Jan 2013-Dec 2015)	(2,1,5)	(5,0,2)
2 (Jan 2014-Dec 2016)	(5,0,4)	(4,0,4)
3 (Jan 2015-Dec 2017)	(2,1,2)	(2,1,2)
4 (Jan 2016-Dec 2018)	(5,1,1)	(2,0,2)
5 (Jan 2017-Dec 2019)	(5,1,0)	(2,1,5)
6 (Jan 2018-Dec 2020)	(0,1,0)	(2,0,5)
7 (Jan 2019-Dec 2021)	(0,1,0)	(3,0,5)

Table 14: ARIMA parameters (p,d,q) for each rolling window for Apple and Amazon.

A notable observation for Apple is seen in Windows 6 and 7, where both the autoregressive (AR) parameter (p) and the Moving Average (MA) parameter (q) are zero. This indicates that the model does not identify any dependency on past values or errors to predict the current value for these windows. This peculiarity might be attributed to the abnormal market conditions during COVID-19, which might have disrupted the historical patterns the model relies upon. In contrast, the AR and MA parameters for Windows 6 and 7 are not zero for Amazon, demonstrating a continued dependency on past values and errors in forecasting the current value. This differential behavior in the ARIMA parameters between Apple and Amazon during the same windows could reflect how unique market forces or company-specific factors influence the stock prices differently and, by extension, their volatility.

The estimated coefficients and their respective t-statistics, presented in Table 15, are derived from the ARIMA model fitted to the training dataset for Apple and Amazon stocks. The ω indicates the constant, the ϕ coefficients represent the autoregressive (AR) terms, while the α coefficients signify the model's moving average (MA) terms. A discernible pattern of negative coefficients is evident in specific windows for both companies, such as Window 1 for Apple and Window 5 for Amazon. The coefficients for Apple and Amazon fluctuate between positive and negative across different windows, indicating a complex dynamic between past and current values across different time frames. For Apple precisely, in Window 1, the coefficients for past error terms are all negative and significant, except for α_4, which has a weak t-statistic. This pattern is also observed in Amazon, where the negative coefficients are observed in the past value terms. As discussed earlier, ARIMA could not capture and relate past values and errors to the current value for Windows 6 and 7, where the pandemic is present. At the same time, for Amazon, it seems that mainly the lag error terms (up to 5) correlate negatively with the forecasted value. This finding may suggest that, for Apple, other factors could have contributed to the variance of the TRV. Furthermore, the constants ω are not significantly different from 0, and almost all are statistically significant. This indicates that the ARIMA model for both stocks did not fully capture the underlying patterns of TRV.

Window	$\hat{\omega}$	$\hat{\phi}_1$	$\hat{\phi}_2$	$\hat{\phi}_3$	$\hat{\phi}_4$	$\hat{\phi}_5$	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\alpha}_3$	$\hat{\alpha}_4$	$\hat{\alpha}_5$
Apple											
1	-0.0004	0.2140	0.1313	-	-	-	-0.1928	-0.1079	-0.0305	-0.0044	-0.5911
	(-0.326)	(3.380)	(1.844)	-	-	-	(-3.728)	(-1.974)	(-1.018)	(-0.137)	(-24.865)
2	0.6412	0.1120	0.1044	0.0637	0.0861	0.0969	0.8892	0.7913	0.7182	0.6617	-
	(6.109)	(1.553)	(1.300)	(1.094)	(1.353)	(1.687)	(14.525)	(10.735)	(11.816)	(13.109)	-
3	-0.0011	-1.3750	-0.6755	-	-	-	1.5489	0.9046	-	-	-
	(-0.025)	(-31.365)	(-16.255)	-	-	-	(53.401)	(31.119)	-	-	-
4	0.0027	0.1438	0.0388	0.0491	0.0571	-0.3751	-0.1789	-	-	-	-
	(0.251)	(1.726)	(0.916)	(1.257)	(1.616)	(-17.490)	(-1.836)	-	-	-	-
5	0.0007	-0.0058	-0.0046	0.0279	0.0033	-0.3440	-	-	-	-	-
	(0.059)	(-0.173)	(-0.097)	(0.691)	(0.089)	(-18.160)	-	-	-	-	-
Amazon											
1	0.6883	0.0835	0.4067	0.4174	0.0685	-0.4293	0.8967	0.4597	-	-	-
	(7.276)	(1.063)	(6.289)	(5.995)	(1.805)	(-13.897)	(10.735)	(5.525)	-	-	-
2	0.8244	0.1047	0.1716	0.0798	0.1029	-	0.9077	0.7535	0.6884	0.6457	-
	(5.313)	(1.401)	(1.933)	(1.144)	(1.322)	-	(15.536)	(11.082)	(10.306)	(24.042)	-
3	-0.0024	-1.3444	-0.6188	-	-	-	1.5607	0.8994	-	-	-
	(-0.038)	(-29.102)	(-14.279)	-	-	-	(51.043)	(29.307)	-	-	-
4	0.0155	-1.4681	-0.6896	-	-	-	1.6017	0.8971	-	-	-
	(0.291)	(-29.386)	(-14.519)	-	-	-	(47.386)	(28.019)	-	-	-
5	0.00003	0.3019	0.1479	-	-	-	-0.3558	-0.0864	-0.0186	0.0146	-0.4019
	(0.013)	(4.645)	(1.853)	-	-	-	(-8.393)	(-1.230)	(-0.450)	(0.388)	(-19.664)
6	0.0318	1.3948	-0.4152	-	-	-	-0.4450	0.0801	0.0438	-0.0120	-0.3522
	(1.424)	(15.794)	(-5.348)	-	-	-	(-5.594)	(2.123)	(1.000)	(-0.273)	(-11.908)
7	0.0384	1.1015	-0.0072	-0.1201	-	-	-0.1429	-0.0544	0.0848	-0.0560	-0.4746
	(1.371)	(13.969)	(-0.062)	(-1.704)	-	-	(-1.790)	(-0.648)	(2.757)	(-1.371)	(-16.879)

Table 15: ARIMA Model Coefficients and t-statistics for Apple and Amazon Stock across Different Windows.

5.3 ARCH and GARCH Models Analysis

5.3.1 Results and Interpretation

We employed both the ARCH and GARCH models, determining the best parameters based on the lowest AIC and BIC values. Table 16 presents the selected parameters for both Apple and Amazon. The AIC criterion generally favors models with larger parameter values, indicating a preference for more complex models. In contrast, the BIC criterion chooses models with smaller parameter values, signifying a tendency towards simpler models. Focusing on Apple's parameters, we notice that BIC models exhibit more consistency across various windows. On the other hand, for Amazon, stability is evident across all models, indicating consistent volatility patterns over time. A noticeable observation from the table is that Amazon's stock price volatility is influenced by a broader set of historical data compared to Apple. In the GARCH model, Amazon frequently requires more ARCH (p) and GARCH (q) terms, as suggested by both AIC and BIC. This may imply that Amazon's stock volatility is driven by a broader range of past events or conditions, which is consistent with the observed complex volatility of Amazon.

Window	GARCH		ARCH	
	AIC (p,q)	BIC (p,q)	AIC (q)	BIC (q)
1 (Jan 2013-Dec 2015)	(1, 5)	(1, 1)	3	1
2 (Jan 2014-Dec 2016)	(1, 2)	(1, 1)	3	3
3 (Jan 2015-Dec 2017)	(1, 2)	(1, 2)	5	3
4 (Jan 2016-Dec 2018)	(1, 1)	(1, 1)	5	3
5 (Jan 2017-Dec 2019)	(5, 1)	(1, 1)	3	3
6 (Jan 2018-Dec 2020)	(3, 3)	(1, 1)	4	3
7 (Jan 2019-Dec 2021)	(2, 2)	(1, 1)	5	4

(a) Apple

Window	GARCH		ARCH	
	AIC (p,q)	BIC (p,q)	AIC (q)	BIC (q)
1 (Jan 2013-Dec 2015)	(5, 1)	(1, 1)	5	1
2 (Jan 2014-Dec 2016)	(4, 1)	(1, 3)	4	4
3 (Jan 2015-Dec 2017)	(4, 2)	(4, 2)	4	4
4 (Jan 2016-Dec 2018)	(4, 1)	(4, 1)	5	5
5 (Jan 2017-Dec 2019)	(4, 1)	(4, 1)	5	5
6 (Jan 2018-Dec 2020)	(1, 1)	(1, 1)	5	5
7 (Jan 2019-Dec 2021)	(3, 2)	(1, 2)	5	3

(b) Amazon

Table 16: Parameters selected for the models across different windows, with AIC and BIC used for GARCH and ARCH models.

Table 17 indicates the estimated coefficients from the GARCH model, for both Apple and Amazon, along with their respective t-statistics [12]. The α coefficients represent the past conditional variance terms, the β coefficients represent the lag squared error terms, and the ω coefficient represents the constant long-run variance component. Recall from Section 2 that the null hypothesis is coef $= 0$, indicating no influence on the current value, while the alternative hypothesis posits that the coefficient is different from zero, indicating an effect of this term on the time series.

Before diving into the hypotheses, it is crucial to check if the assumption $\sum_{i=1}^{q} \alpha_i + \sum_{j=1}^{p} \beta_j < 1$ is met from our implementation. Given the coefficients in Table 17, the sum per row is indeed lower than 1. Hence, the conditional variance generated by the model is a weakly stationary process. As we can see from the table, some coefficients are 0, indicating that these terms do not influence the current conditional variance, although their t-statistics are 0; thus, there is no strong evidence for this. A notable fact is that even though these terms do not have an influence, some of the subsequent terms have a significant influence. An example of this pattern is revealed for Apple, in Window 3, where although $\beta_1 = 0$, β_2 has a significant impact, with a high t-statistic (7.797). The same trend can be found on Amazon's Windows 1, 2, and 3. This may suggest that the time series' volatility might be affected by events or conditions from more distant past periods rather than recent periods. Additionally, the impact of past conditional variances for Amazon is more dispersed across

[12]The corresponding table for the ARCH model (Table 27) is presented in the Appendix Section

different lags than Apple. This dispersion could indicate a more complex or nuanced volatility behavior in Amazon's stock prices over the examined windows. For instance, in Window 3 for Amazon, the significance of the coefficients spans from α_1 to β_1, and notably, α_4 has a relatively higher value of 0.5599, suggesting that the lagged conditional variances from further past periods play a notable role in influencing the current conditional variance. On the other hand, Apple's coefficients are more concentrated within the initial lags, especially in Windows 2, 3 and 4, where only α_1, β_1, and β_2 have non-zero values. This concentration may suggest a more immediate or short-term memory in volatility for Apple's stock prices. Interestingly, both Apple and Amazon exhibit positive and statistically significant intercepts. This means that the short-run lagged values used by the model do not fully account for the observed variance. This indicates a long-run variance component that the model doesn't capture.

This disparity in lag significance and coefficient distribution across different lags between companies may reflect their distinct market dynamics, investor behaviors, or reactions to market events. Understanding these nuances could provide insightful perspectives on the volatility modeling of these stocks.

Window	$\hat{\omega}$	$\hat{\alpha_1}$	$\hat{\alpha_2}$	$\hat{\alpha_3}$	$\hat{\alpha_4}$	$\hat{\alpha_5}$	$\hat{\beta_1}$	$\hat{\beta_2}$	$\hat{\beta_3}$	$\hat{\beta_4}$	$\hat{\beta_5}$
Apple											
1	0.4948	0.2753	-	-	-	-	0.0216	0.1795	0.0000	0.0000	0.3564
	(2.014)	(2.875)	-	-	-	-	(0.347)	(1.585)	(0.000)	(0.000)	(2.174)
2	0.4034	0.1638	-	-	-	-	0.000	0.6683	-	-	-
	(2.504)	(2.603)	-	-	-	-	(0.000)	(1.691)	-	-	-
3	0.3086	0.1578	-	-	-	-	0.000	0.6925	-	-	-
	(2.227)	(3.100)	-	-	-	-	(0.000)	(7.797)	-	-	-
4	0.2122	0.1339	-	-	-	-	0.7744	-	-	-	-
	(2.761)	(3.413)	-	-	-	-	(15.245)	-	-	-	-
5	0.3528	0.1138	0.0000	0.1004	0.0000	0.1464	0.5290	-	-	-	-
	(2.611)	(1.903)	(0.000)	(0.887)	(0.000)	(1.192)	(4.764)	-	-	-	-
6	0.4812	0.1454	0.1164	0.2441	-	-	0.0043	0.0000	0.4296	-	-
	(2.919)	(2.306)	(1.991)	(2.806)	-	-	(0.0167)	(0.000)	(1.247)	-	-
7	0.2842	0.0785	0.1804	-	-	-	0.0349	0.6404	-	-	-
	(2.888)	(1.935)	(2.754)	-	-	-	(0.422)	(7.926)	-	-	-
Amazon											
1	2.4398	0.2761	0.000	0.000	0.0769	0.0779	0.000	-	-	-	-
	(1.454)	(1.649)	(0.000)	(0.000)	(0.523)	(0.679)	(0.000)	-	-	-	-
2	2.3652	0.2773	0.0000	0.0000	0.2272	-	0.000	-	-	-	-
	(1.861)	(1.808)	(0.000)	(0.000)	(1.353)	-	(0.000)	-	-	-	-
3	0.7152	0.2186	0.0000	0.000	0.5599	-	0.0000	0.2142	–	-	-
	(2.574)	(2.236)	(0.000)	(0.000)	(3.040)	-	(0.000)	(2.067)	- -	-	-
4	0.4236	0.1161	0.0759	0.0981	0.3776	-	0.3322	-	- -	-	-
	(3.269)	(2.417)	(0.916)	(1.457)	(1.849)	-	(2.735)	-	- -	-	-
5	0.4523	0.1616	0.0331	0.000	0.3645	-	0.3735	-	–	-	-
	(2.962)	(2.275)	(0.460)	(0.000)	(1.549)	-	(2.111)	-	–	-	-
6	0.1867	0.1808	-	-	-	-	0.7834	-	-	-	-
	(2.426)	(4.799)	-	-	-	-	(19.326)	-	-	-	-
7	0.8713	0.1360	0.0111	0.2083	-	-	0.000	0.3896	-	-	-
	(1.627)	(2.026)	(0.360)	(1.673)	-	-	(0.000)	(2.578)	-	-	-

Table 17: GARCH Model Coefficients and t-statistics for Apple and Amazon Stock across Different Windows.

5.3.2 Comparative Analysis of GARCH and ARCH Models Using AIC and BIC Criteria and Their Combined Forecasting Approach

Our next focus is in addressing the question of which criterion, AIC or BIC, yields the most accurate results. As a result, we have two distinct sets of outcomes for each model.

Table 18 presents the out-of-sample RMSE for the models selected by AIC and BIC, both GARCH and ARCH. When combining the insights from both tables, several vital observations emerge. Notably, neither criterion consistently outperforms the other across all windows. Although there are subtle differences between the criteria, in some cases, the superior criterion is apparent. For instance, in Apple's results, in the ARCH model's first window, the AIC's complexity with three lagged values provides better results than the BIC's simpler single-lag model. However, in the third window, the reverse is true. This variability is not surprising given that each window represents a distinct time period with unique characteristics. Turning to Amazon's results, as we expected, given its parameters, the results are more stable among the criteria. Here, we also identify windows where the complexity outperforms simplicity and vice versa.

	GARCH		ARCH	
Window	AIC Test RMSE	BIC Test RMSE	AIC Test RMSE	BIC Test RMSE
1 (Jan 2016-Dec 2016)	0.674	0.692	0.817	0.860
2 (Jan 2017-Dec 2017)	0.495	0.493	0.441	0.441
3 (Jan 2018-Dec 2018)	0.731	0.731	0.832	0.759
4 (Jan 2019-Dec 2019)	0.615	0.615	0.807	0.791
5 (Jan 2020-Dec 2020)	1.195	1.219	1.278	1.278
6 (Jan 2021-Dec 2021)	0.585	0.593	0.676	0.686
7 (Jan 2022-Dec 2022)	0.793	0.791	0.735	0.731

(a) Apple

	GARCH		ARCH	
Window	AIC Test RMSE	BIC Test RMSE	AIC Test RMSE	BIC Test RMSE
1 (Jan 2016-Dec 2016)	0.928	0.906	0.943	1.268
2 (Jan 2017-Dec 2017)	0.771	0.759	0.782	0.782
3 (Jan 2018-Dec 2018)	0.947	0.947	0.940	0.940
4 (Jan 2019-Dec 2019)	0.639	0.639	0.670	0.670
5 (Jan 2020-Dec 2020)	1.085	1.085	1.125	1.125
6 (Jan 2021-Dec 2021)	0.550	0.550	0.577	0.577
7 (Jan 2022-Dec 2022)	1.502	1.483	1.448	1.423

(b) Amazon

Table 18: Comparison of Test RMSE across different windows for GARCH and ARCH models using AIC and BIC criteria.

Given the observations above and noting the absence of a clear "winner" between the models generated by AIC and BIC, we choose to employ the aggregated forecast through the exponential reweighting (AFTER) technique.

The AFTER algorithm, as described by Zou & Yang (2004) and introduced by Yang (2004), combines predicted values to generate new forecasts by allocating weights to individual predictions based on their past performances and estimated variances. By integrating predictions from these distinct models, we aim to capture their unique dynamics and combine them into a potentially more accurate output.

In this study, we combine forecasts derived from AIC and BIC models. Hence, the number of models, J, is 2. Specifically, for each forecasting procedure j, the weight $W_{j,1}$ is initialized to p_j, which is defined as $\frac{1}{J}$, ensuring an equal initial weight for each model.
For subsequent time points $n \geq 2$, the weights are updated as:

$$W_{j,n} = \frac{p_j \times \prod_{i=1}^{n-1} v_{j,i}^{-1/2} \times \exp\left(-\frac{1}{2}\sum_{i=1}^{n-1} \frac{(Y_i - \hat{y}_{j,i})^2}{v_{j,i}}\right)}{\sum_{j'=1}^{J} p_{j'} \times \prod_{i=1}^{n-1} v_{j',i}^{-1/2} \times \exp\left(-\frac{1}{2}\sum_{i=1}^{n-1} \frac{(Y_i - \hat{y}_{j',i})^2}{v_{j',i}}\right)} \tag{24}$$

Where $\hat{y}_{j,i}$ represents the forecast of the i^{th} observation using the j^{th} procedure, $v_{j,i}$ is its conditional variance, and p_j is the initial weight assigned to the j^{th} forecasting procedure. This modification ensures that as prediction errors are computed for past observations, higher weights are assigned to procedures that produce predictions with lower errors.

Table 19 presents the overall RMSE values for both Apple and Amazon, derived from the AIC, BIC, and the combined forecasting approaches.

	Combined RMSE	AIC RMSE	BIC RMSE
	ARCH		
Apple	0.814	0.831	0.826
Amazon	0.964	0.966	1.012
	GARCH		
Apple	0.758	0.757	0.765
Amazon	0.956	0.963	0.955

Table 19: Overall RMSE obtained from AIC, BIC, and combined forecasts.

Several key observations can be made from Table 19. For the ARCH model, Apple's RMSE for the combined forecasts seems to outperform both AIC and BIC scores, while for Amazon, the combination appears to be close to the AIC results. The same pattern is observed in the GARCH results, where for both Apple and Amazon, the combination tends to be closer to the lowest score, with minor differences among all three forecasting approaches. Overall, the combined approach seems to offer a balance between the AIC and BIC, potentially capturing the strengths of both criteria. Hence, from now on, when we refer to the GARCH or ARCH model, the combined forecasts will be used.

5.4 Comparison of Statistical Models to Baseline Model

In this section, we evaluate the performance of the statistical models and compare them with the baseline model (naive approach) we examined in Section 5.1. Table 20 and Table 21 posit the out-of-sample metrics RMSE and the Skill Score for each window, along with the overall performances for Apple and Amazon, respectively. Lower RMSE values indicate better performance, while a Skill Score greater than 1 suggests the outperformance of the specific model in comparison with the naive model.

First, we will examine the models for Apple. In the overall performance, it seems that in terms of Skill Score, GARCH and ARCH outperform the naive model, with the GARCH model having the lowest RMSE of 0.758. ARIMA, on the other hand, has the highest RMSE, indicating that it cannot capture the patterns of the TRV. The robustness of the GARCH model is further demonstrated through its performance across different time windows in this analysis, with two exceptions, Windows 2 and 7, where the ARCH model exhibits the best performance. This indicates that the GARCH model can adapt to changing data patterns and maintain a relatively low error rate, reflecting its robustness in financial time series analysis, especially under varying data distributions. The out-of-sample test set for Window 5 is the period when COVID-19 occurred, thus the relatively too high RMSEs across all models, but even there, ARCH and GARCH models outperform the naive model, with GARCH being the "winner" once more.

In general, for Apple, the GARCH model seems to have a good performance compared to the naive and the ARCH ARIMA models, while ARIMA has the worst performance in an overall view. Considering the linear nature of these models might limit their ability to capture more complex, nonlinear dynamics in the data. Exploring models that can handle non-linearity might provide a way to improve the performance and better model the underlying patterns in Apple's stock prices.

	Naive Model	ARIMA		ARCH		GARCH	
	RMSE	RMSE	Skill Score	RMSE	Skill Score	RMSE	Skill Score
1 (Jan 2016-Dec 2016)	0.773	0.719	1.072	0.829	0.933	0.676	**1.14**
2 (Jan 2017-Dec 2017)	0.574	0.667	0.860	0.443	**1.3**	0.497	1.158
3 (Jan 2018-Dec 2018)	0.761	1.155	0.656	0.778	0.980	0.729	**1.044**
4 (Jan 2019-Dec 2019)	0.766	0.733	1.044	0.729	1.049	0.618	**1.241**
5 (Jan 2020-Dec 2020)	1.312	1.498	0.877	1.281	1.025	1.204	**1.091**
6 (Jan 2021-Dec 2021)	0.620	0.945	0.656	0.674	0.922	0.578	**1.072**
7 (Jan 2022-Dec 2022)	0.919	1.155	0.794	0.729	**1.265**	0.789	1.162
Overall (Jan 2016-Dec 2022)	0.849	1.021	0.831	0.814	1.044	0.758	**1.118**

Table 20: Models Performance per window and overall performance for Apple stock.

A similar trend is observed for Amazon stock, where in the overall performance, GARCH and ARCH outperform the Naive model, with the GARCH model registering the lowest RMSE of 0.958. Once again, ARIMA cannot perform better than the naive approach (Skill Score=0.781), indicating its unsuitability for this dataset. Notably, ARIMA's best performance is illustrated within Window 1, where although it has the lowest RMSE across the statistical models, it cannot outperform the benchmark. A noteworthy observation is the performance during Window 5 for all models, where high RMSEs are recorded. This suggests that the linear approaches of these models cannot capture the abnormal activity during this unique period of the pandemic. Moreover, Window 7 seems to record the worst performances across all models, hinting at other underlying factors that might have driven Amazon's volatility to increase, factors that may not have a linear relationship with our target variable. In subsequent analyses, more sophisticated non-linear and multivariate models will be introduced to unravel these patterns and provide a more nuanced understanding of the factors affecting Amazon's stock performance.

	Naive Model	ARIMA		ARCH		GARCH	
	RMSE	RMSE	Skill Score	RMSE	Skill Score	RMSE	Skill Score
1 (Jan 2016-Dec 2016)	0.857	0.872	**0.985**	0.957	0.894	0.915	0.938
2 (Jan 2017-Dec 2017)	0.851	1.143	0.742	0.785	1.086	0.756	**1.127**
3 (Jan 2018-Dec 2018)	0.951	1.460	0.648	0.939	**1.015**	0.946	1.005
4 (Jan 2019-Dec 2019)	0.716	0.862	0.831	0.663	1.082	0.630	**1.136**
5 (Jan 2020-Dec 2020)	1.178	1.310	0.900	1.126	1.044	1.087	**1.086**
6 (Jan 2021-Dec 2021)	0.623	1.118	0.557	0.575	1.082	0.550	**1.131**
7 (Jan 2022-Dec 2022)	1.564	1.941	0.806	1.435	**1.091**	1.494	1.049
Overall (Jan 2013- Dec 2022)	1.007	1.292	0.781	0.965	1.044	0.958	**1.049**

Table 21: Models Performance per window and overall performance for Amazon stock.

5.5 Neural Network Analysis

In this section, we compare LSTM and FFNN models, benchmarking them against baseline models. A notable point of discussion is the selection of units during hyperparameter tuning for both Apple and Amazon, as depicted in Table 22. An interesting observation from the table is the disparity in the number of units chosen for the two companies. For Amazon, the first layer's unit count peaks at 150, with the smallest being 50. In contrast, Apple's maximum count is 64 units. This difference indicates the complexities present in Amazon's TRV compared to Apple's. Given that Amazon's daily volatility exhibits more pronounced fluctuations than Apple's, it's logical that the model would require a more complex configuration to capture Amazon's underlying patterns accurately.

Windows	Apple		Amazon	
	Units 1	Units 2	Units 1	Units 2
1 (Jan 2013- Dec 2015)	32	32	50	32
2 (Jan 2014- Dec 2016)	32	32	150	32
3 (Jan 2015- Dec 2017)	64	32	100	50
4 (Jan 2016- Dec 2018)	32	32	150	16
5 (Jan 2017- Dec 2019)	50	32	150	16
6 (Jan 2018- Dec 2020)	64	32	150	100
7 (Jan 2019- Dec 2021)	50	32	100	50

Table 22: LSTM units selection via tuning for Apple and Amazon across 7 Windows.

5.5.1 Comparison of Neural Network Models to Baseline Model

We begin our discussion focusing on Apple. Table 23 presents the out-of-sample RMSE and Skill Score for each window, along with their overall performance. Overall, almost all models outperform the benchmark, with a notable exception of NN5. The LSTM model stands out, demonstrating the best performance with an RMSE of 0.776 and a Skill Score of 1.095, while NN1 has the second-best performance with a Skill Score of 1.054.

Delving into specific periods, especially Window 5, we observe that all models exhibit their highest RMSE values, and none of the non-linear models outperform the baseline. It's crucial to recall that the training period spanned from 2017-01-01 to 2019-12-31, with the subsequent test year coinciding with the onset of the COVID-19 pandemic. Given that we utilized a 30-point lag, these results suggest that the model cannot capture this abnormal activity based solely on TRV's past values. Delving deeper with multivariate approaches in subsequent analyses may offer more insights into these underlying factors. A broader overview across all windows suggests that LSTM is dominating, followed closely by NN1.

For Amazon's stock, we observe a different pattern. Here, all models are close to each other, with NN3 illustrating a moderately higher Skill Score. All models outperform the benchmark. NN1 displays a relatively lower performance, with its RMSE being 0.066 lower than the baseline. Among the FFNNs, NN2 and NN3 exhibit better overall results. This again underscores the greater complexity of Amazon's TRV compared to Apple's TRV. Further, increasing the model's complexity by adding more layers does not enhance its performance.

A deeper examination of individual windows offers additional insights into specific periods. In Window 5, for example, even though the RMSEs across models are relatively high, they outperform the baseline, except for NN1 and NN5. A similar trend is evident in Window 7, where NN5 exhibits the lowest RMSE. All models achieved their best performance in Windows 1 and 6, suggesting that during these periods, the models could discern the underlying patterns in the past TRV values. Interestingly, for Window 6, the model was trained on data from the interval (01-01-2018 to 31-12-2020) and was subsequently tested on the interval (01-01-2021 to 31-12-2021). Given that the pandemic began around 30-12-2020, this implies that the models are effectively able to capture the significant shifts in Amazon's market dynamics brought about by the onset of the pandemic.

5.6 Multivariate Models

This section explores two predictive models: LSTM and random forest. Both these models take into account the past 30 days of data from our 20 significant features, identified by SHAP values, as we discussed in Sections 3 and 3.3.

As we have utilized two distinct sentiment analysis techniques, merging their outcomes is crucial. Once integrated, we assess the performance of our predictions against the baseline model. We then draw comparisons between these methods and others previously discussed. Finally, we delve deeper to understand which features played a crucial role in these predictions.

5.6.1 Combining dictionaries

Section 2.2.7 delved into our methodology for sentiment analysis of news articles, employing two distinct sentiment dictionaries: the Loughran/Mcdonald and the Harvard IV-4. Notably, these dictionaries demonstrate significant differences in sentiment categorization. The Harvard IV-4 tends to lean towards positive headline classifications, whereas neutral sentiments dominated the Loughran/Mcdonald analysis. To leverage the unique strengths of both dictionaries, we proceed with the following approach:

- **Individual Predictions:** We initially generated forecasts within each window using two separate datasets, each including exclusively one of the dictionaries.
- **Weighted Combination:** We combined these individual predictions using a weighted average for each window.

To determine the optimal weights for this combination, we used an optimization framework called Optuna. It aims to minimize the RMSE between the actual and predicted values through a set number of iterations. In our case, we conducted 100 trials. Table 24 illustrates the overall

Apple Stock													
	Naive Model	NN1		NN2		NN3		NN4		NN5		LSTM	
	RMSE	RMSE	Skill	RMSE	Skill	RMSE	Skill	RMSE	Skill	RMSE	Skill	RMSE	Skill
1 (Jan 2016- Dec 2016)	0.773	0.592	1.304	0.633	1.221	0.591	**1.308**	0.584	1.323	0.596	1.296	0.646	1.196
2 (Jan 2017- Dec 2017)	0.574	0.451	**1.277**	0.567	1.015	0.474	1.212	0.481	1.196	0.476	1.204	0.511	1.122
3 (Jan 2018- Dec 2018)	0.761	0.729	1.044	0.716	1.063	0.760	1.001	0.776	0.980	0.745	1.020	0.677	**1.122**
4 (Jan 2019- Dec 2019)	0.766	0.547	**1.399**	0.548	1.399	0.553	1.386	0.599	1.281	0.552	1.386	0.550	1.393
5 (Jan 2020- Dec 2020)	1.312	1.491	0.877	1.560	0.843	1.574	0.837	1.544	0.849	1.861	0.707	1.403	**0.933**
6 (Jan 2021- Dec 2021)	0.620	0.523	1.183	0.550	1.127	0.524	1.183	0.521	1.192	0.522	1.187	0.514	**1.204**
7 (Jan 2022- Dec 2022)	0.919	0.825	1.114	0.789	1.162	0.752	1.221	0.749	1.225	0.827	1.109	0.742	**1.237**
Overall (Jan 2016- Dec 2022)	0.849	0.806	1.054	0.835	1.015	0.825	1.025	0.823	1.030	0.914	0.927	0.776	**1.095**

Amazon Stock													
	Naive Model	NN1		NN2		NN3		NN4		NN5		LSTM	
	RMSE	RMSE	Skill	RMSE	Skill	RMSE	Skill	RMSE	Skill	RMSE	Skill	RMSE	Skill
1 (Jan 2016- Dec 2016)	0.857	0.715	1.200	0.654	**1.311**	0.679	1.261	0.687	1.249	0.720	1.192	0.693	1.237
2 (Jan 2017- Dec 2017)	0.851	0.825	1.030	0.850	1.001	0.772	**1.102**	0.860	0.990	0.861	0.980	0.775	1.095
3 (Jan 2018- Dec 2018)	0.951	0.888	**1.072**	0.962	0.990	0.965	0.985	1.022	0.933	0.973	0.980	0.941	1.010
4 (Jan 2019- Dec 2019)	0.716	0.521	1.375	0.572	1.253	0.472	**1.517**	0.483	1.483	0.518	1.382	0.483	1.483
5 (Jan 2020- Dec 2020)	1.178	1.189	0.990	1.103	1.068	1.094	**1.077**	1.151	1.025	1.255	0.938	1.140	1.034
6 (Jan 2021- Dec 2021)	0.623	0.508	1.225	0.526	1.183	0.494	1.261	0.511	1.221	0.528	1.179	0.486	**1.281**
7 (Jan 2022- Dec 2022)	1.564	1.507	1.039	1.279	1.221	1.304	1.196	1.287	1.162	1.173	**1.334**	1.297	1.204
Overall (Jan 2016- Dec 2022)	1.007	0.941	1.072	0.889	1.131	0.875	**1.153**	0.905	1.114	0.903	1.115	0.880	1.145

Table 23: Performance comparison of different models on Apple and Amazon stock prices.

performance for both Apple and Amazon stocks, comparing LSTM and random forest models, along with the RMSEs from their combinations. The advantage of using weighted average predictions is evident across all cases. As can be seen, for LSTM, especially where we observe a more significant difference between dictionaries than in Random Forest, the combination led to a more substantial reduction in RMSE. This indicates that the Optuna algorithm gives more weight to predictions, yielding a lower RMSE. Given the results above, we will compare the combined forecasts with the naive and the other complex models we discussed previously.

	Combined RMSE	RMSE Loughran/Mcdonald	RMSE Harvard
LSTM			
Apple	0.819	0.954	1.003
Amazon	0.894	0.947	1.052
Random Forest			
Apple	0.826	0.839	0.840
Amazon	1.436	1.441	1.449

Table 24: Overall RMSE for LSTM and Random Forest models using different sentiment dictionaries: Loughran/McDonald and Harvard, along with their combination.

5.6.2 Comparison of Multivariate Models to Baseline Model

Table 25 illustrates the performance comparison, within the test set, between the baseline model and the two multivariate models, LSTM and random forest. Notably, the LSTM model consistently outperforms the others in overall performance for both Apple and Amazon. This model exceeds both the naive and Random Forest models, with the latter showing a marked underperformance for Amazon, reflected in a Score Skill of 0.7.

Diving deeper into individual periods, specifically for Apple, there is a discernible balance between the LSTM and random forest models. Both models show an advantage over the naive model during Window 5, with Random Forest achieving a Score Skill of 1.34. This suggests that leveraging multiple features can lead to a more robust performance during abnormal periods, such as the pandemic. Interestingly, Windows 3 and 6, which previously saw superior performance from other complex models, witnessed a fall in performance for the multivariate models. This suggests that integrating many factors may prevent predictive accuracy during more stable periods without significant variance.

For Amazon, the dominance of LSTM is evident across different windows, though it does not outperform the naive model in Windows 3 and 6. Similar to Apple, the pattern in Window 5 is replicated. In Window 7, where the naive model reaches its peak RMSE, the LSTM still performs better. However, its RMSE exceeds 1, signaling that it struggles to capture this period's patterns even with multiple features. It is also worth noting that these sophisticated models could not keep up in windows where prior models shone.

5.7 Comparison of all models

So far, we have examined each category of the models by comparing them with the naive models across different windows/periods, along with their overall performance. In this section, our main concentration is to compare all models we have implemented. We aim to examine how the models perform in the short-term (weekly) and mid-term (monthly). This approach will offer deeper insights into their performance, enabling a straightforward comparison among all models. As depicted in Table 26, we present the results for both Apple and Amazon. It

Apple					
	Naive	LSTM		Random Forest	
Window	RMSE	RMSE	Score Skill	RMSE	Score Skill
1 (Jan 2016- Dec 2016)	0.773	0.588	**1.31**	0.781	0.99
2 (Jan 2017- Dec 2017)	0.574	0.459	**1.25**	0.463	1.24
3 (Jan 2018- Dec 2018)	0.761	0.768	0.99	0.704	**1.08**
4 (Jan 2019- Dec 2019)	0.766	0.546	**1.4**	0.811	0.94
5 (Jan 2020- Dec 2020)	1.312	1.212	1.08	0.978	**1.34**
6 (Jan 2021- Dec 2021)	0.620	0.551	**1.13**	1.126	0.55
7 (Jan 2022- Dec 2022)	0.919	1.224	0.75	0.758	**1.21**
Overall (Jan 2016- Dec 2022)	0.849	0.819	**1.03**	0.826	1.027

Amazon					
	Naive	LSTM		Random Forest	
Window	RMSE	RMSE	Score Skill	RMSE	Score Skill
1 (Jan 2016- Dec 2016)	0.857	0.758	**1.13**	0.886	0.97
2 (Jan 2017- Dec 2017)	0.851	0.799	1.07	0.714	**1.19**
3 (Jan 2018- Dec 2018)	0.951	1.027	**0.93**	1.513	0.63
4 (Jan 2019- Dec 2019)	0.716	0.466	**1.54**	2.592	0.28
5 (Jan 2020- Dec 2020)	1.178	0.931	**1.27**	0.942	1.25
6 (Jan 2021- Dec 2021)	0.623	0.654	0.95	0.488	**1.28**
7 (Jan 2022- Dec 2022)	1.564	1.354	**1.16**	1.737	0.9
Overall (Jan 2016- Dec 2022)	1.007	0.894	**1.13**	1.436	0.7

Table 25: Comparison of RMSE and Score Skill for Apple and Amazon using LSTM and Random Forest with multiple features.

is important to note that 'LSTM-MM' refers to the LSTM model with multiple features, and 'RF' denotes the random forest model.

Upon examining the results for both Apple and Amazon, a consistent trend is evident: across all models, the monthly RMSE consistently registers lower than the weekly one. This pattern provides invaluable insights into the nuances of predictive modeling for stock market volatility. The aggregation of daily data into monthly averages reduces the short-term fluctuations. Moreover, from a monthly perspective, the pronounced effects of daily outliers lent stability to the dataset, making it less vulnerable to erratic shifts. Additionally, the stock market's short-term sentiments, i.e., news or events, are effectively reduced in a monthly aggregation, offering a more comprehensive view of the underlying market trends. These findings highlight the advantages of using longer-term, aggregated data when aiming for enhanced predictive insights in stock market volatility.

Diving deeper, especially into Apple's data, GARCH stands out in the short-term predictions, outperforming other models. This result aligns with expectations since GARCH is designed explicitly for volatility forecasting, making it a strong candidate, even when it stands against more "sophisticated", machine learning models. The reliability of GARCH shines through both in weekly and monthly results, suggesting its robustness across different time horizons. Yet, when looking at monthly metrics, LSTM emerges as the top performer. This makes sense since LSTMs are particularly strong with data where the information order plays a crucial role. Evaluating the overall performance confirms these observations: GARCH exhibits the lowest RMSE, closely followed by LSTM. Interestingly, even though the multivariate models did well during unusual market events, their higher RMSE values here verify our earlier discussion (Section 5.6.2); they tend to perform weakly in more stable periods.

Amazon's results reveal GARCH's consistent strength, especially on a weekly basis, underlining its credibility in forecasting volatility. However, on a monthly scale, NN1 takes the lead with an RMSE of 0.434, with LSTM being closely showcasing its suitability for predicting stock volatility. Interestingly, when considering overall performance, NN3 outperforms all, with NN2 and LSTM not far behind. This superior performance of NN3 aligns with our prior discussions, indicating that, given Amazon's variability in TRV, a neural network with an optimal number of layers is more suitable. Yet, the higher RMSE scores of NN4 and NN5 indicate that adding layers beyond a point does not necessarily enhance accuracy. It is worth noting that random forest consistently registered the highest RMSE values, suggesting challenges in capturing the patterns of such complex stock datasets.

Apple											
RMSE	ARIMA	ARCH	GARCH	NN1	NN2	NN3	NN4	NN5	LSTM	LSTM-MM	RF
Weekly	0.946	0.678	**0.639**	0.691	0.759	0.746	0.747	0.847	0.691	0.752	0.757
Monthly	0.819	0.724	0.637	0.507	0.601	0.589	0.590	0.719	**0.464**	0.639	0.668
Overall	1.02	0.814	**0.758**	0.806	0.835	0.825	0.823	0.914	0.776	0.819	0.826

Amazon											
RMSE	ARIMA	ARCH	GARCH	NN1	NN2	NN3	NN4	NN5	LSTM	LSTM-MM	RF
Weekly	1.199	0.761	**0.746**	0.757	0.765	0.761	0.799	0.806	0.766	0.804	1.38
Monthly	1.016	0.549	0.514	**0.434**	0.527	0.528	0.581	0.577	0.498	0.601	1.29
Overall	1.29	0.965	0.958	0.941	0.889	**0.875**	0.905	0.903	0.880	0.894	1.44

Table 26: Weekly, Monthly, and Overall RMSE for Apple and Amazon.

In Figure 15, we present a comparative analysis of the predictive performance for various models over the test period spanning 2016 to 2022. These time plots illustrate the forecasted values (TRV) from each model against the actual observed values, expressed as a percentage and aggregated monthly to enhance clarity.

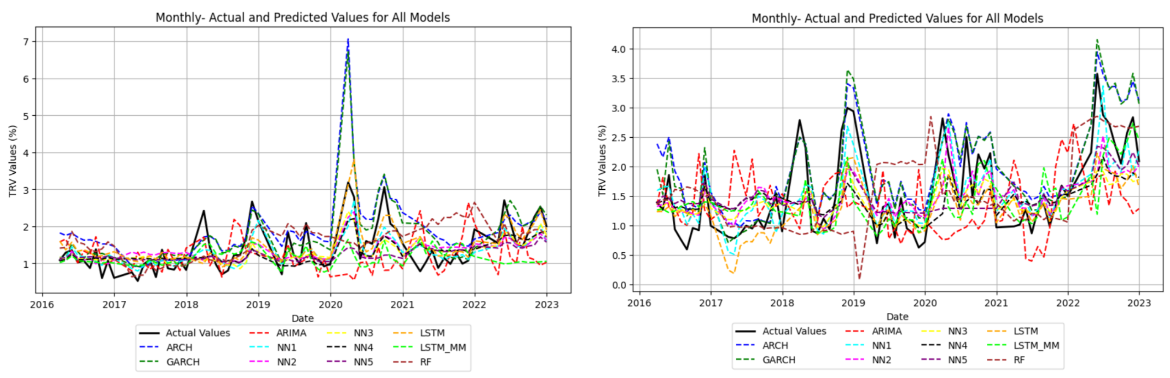


Figure 15: Comparison of (Monthly) Actual and Forecasted Values. The left plot represents the predictive accuracy of models applied to Apple, while the right plot focuses on those used for Amazon.

5.8 Feature Importance

This section discusses and examines the features used as inputs to predict TRV. As a reminder, we collected 63 features categorized into technical, fundamental, macroeconomic, news articles, pandemic dummy variables, and correlated stocks. The importance of these features across all seven windows is depicted in Figure 17, where lighter shades of blue represent higher values and darker shades denote lower values. Since each model was applied using different dictionaries, we calculated the average importance of each feature, factoring in the number of times it was present. Figure 16 displays the distribution of the top 20 features for each model. The categories "Fundamental," "Macroeconomic," and "Technical Indicators" correspond to the ones discussed in Section 2. The "Other" category encompasses the remaining indicators, including correlated stocks, pandemic dummy variable, and sentiment analysis.

An insightful observation from Figure 17 reveals distinct patterns in feature importance between the LSTM and random forest models. For both "Apple/LSTM" and "Amazon/LSTM," a concentrated set of features exhibit lighter shades, signifying that these models allocate higher importance to a select group of features. Conversely, the "Apple/Forest" and "Amazon/Forest" columns display a more dispersed distribution of lighter shades throughout, indicating a broader spread of importance across multiple features.

Figure 16 shows the distribution of the top 20 features' importance across various models. Each of the four models incorporates a mixture of categories, suggesting that an approach considering diverse factors is essential for predicting TRV. At first glance, technical and fundamental indicators dominate the top 20, underscoring their crucial role in providing predictive insights. Delving into specifics, the random forest model prefers technical indicators. When analyzed with random forest, technical indicators stand out for Amazon, suggesting that trading patterns and market sentiment may broadly impact Amazon's volatility. Conversely, for Apple, the LSTM model's emphasis on fundamental indicators implies a substantial influence of the company's financial health and business performance on its TRV.

Given the performance results discussed in Section 5.6.2, where LSTM outperformed random forest, it's evident that fundamental indicators—reflecting the financial health of the companies—are key in forecasting TRV. At the same time, both technical and macroeconomic factors seem to play almost equally significant roles for both companies. This suggests that broader economic trends and specific trading behaviors both have a substantial influence on the predictions.

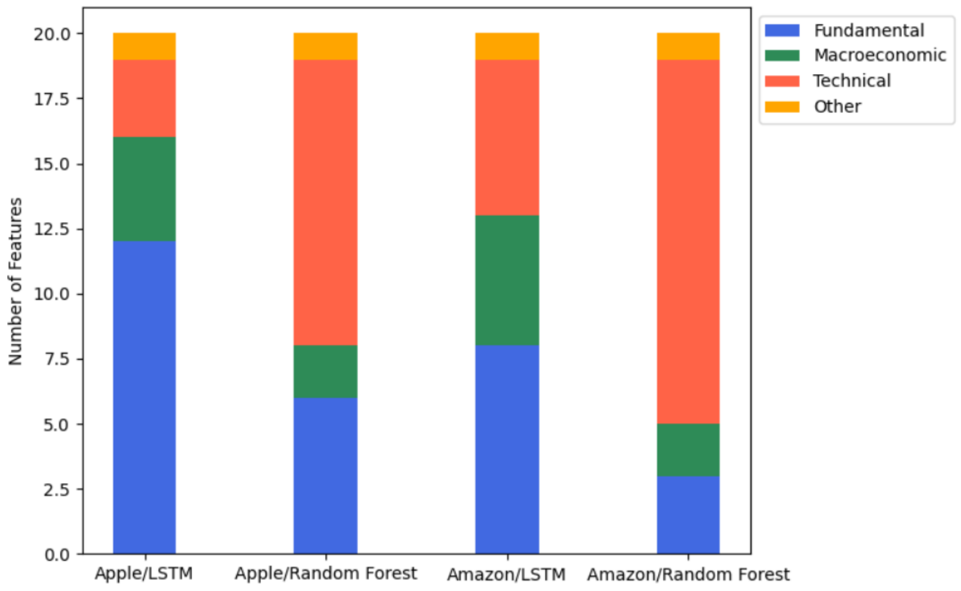


Figure 16: Distribution of Top 20 Features Across Different Categories for Each Model across all windows.

In Figures 18 and 19, we delve into specific periods from our overall timeline. Feature importance across the models is normalized to sum 1. We aim to discuss windows where the multivariate models showed distinct performance patterns compared to the univariate ones. Notably, we will focus on the pandemic period for both stocks: Window 5, where the multivariate models outperformed and which covers the start of the pandemic in the test interval, and Window 6, where the pandemic's onset is included in the training set. Although in Window 6, the multivariate models were not superior to the univariate ones, it remains an important window to analyze, given the significant impact of the pandemic.

First, we will discuss the findings for Apple. For the LSTM model, the most influential features in Window 5 fall within the domain of fundamental indicators. This includes attributes such as Cash Earn Return on Equity, Debts/Equity, and shareholders/equity, to name a few. As we move down the feature importance hierarchy, macroeconomic indicators emerge, followed by technical ones. This trend suggests that, during the period leading up to the pandemic, the company's financial health played a crucial role in predictive modeling. Furthermore, while direct sentiment analysis might not be explicitly represented, we can discern shades of market sentiment via momentum and volume indicators. Notably, the Force Index — a volume-based metric — offers insights into the buying or selling strength. Additionally, Kaufman's Adaptive Moving Average (KAMA), sensitive to price volatility, presents another angle into market momentum.

On the other hand, the random forest model exhibits a more diversified set of influential features in Window 5. Noteworthy is the influence of lag values of TRV, as well as those of the correlated stock, Microsoft. Moreover, while the model does not heavily weigh the direct sentiment indicators derived from news articles, it does emphasize momentum indicators. Indicators such as stochastic RSI (Stochastic Relative Strength Index), which identifies overbought or oversold conditions, and the Ulcer Index and Ultimate Oscillator, underscore the Random Forest model's preference for capturing market sentiment and psychology via technical patterns. This observation posits the potential significance of broad market sentiment rather than sentiment pulled directly from news articles. This diversification can better capture

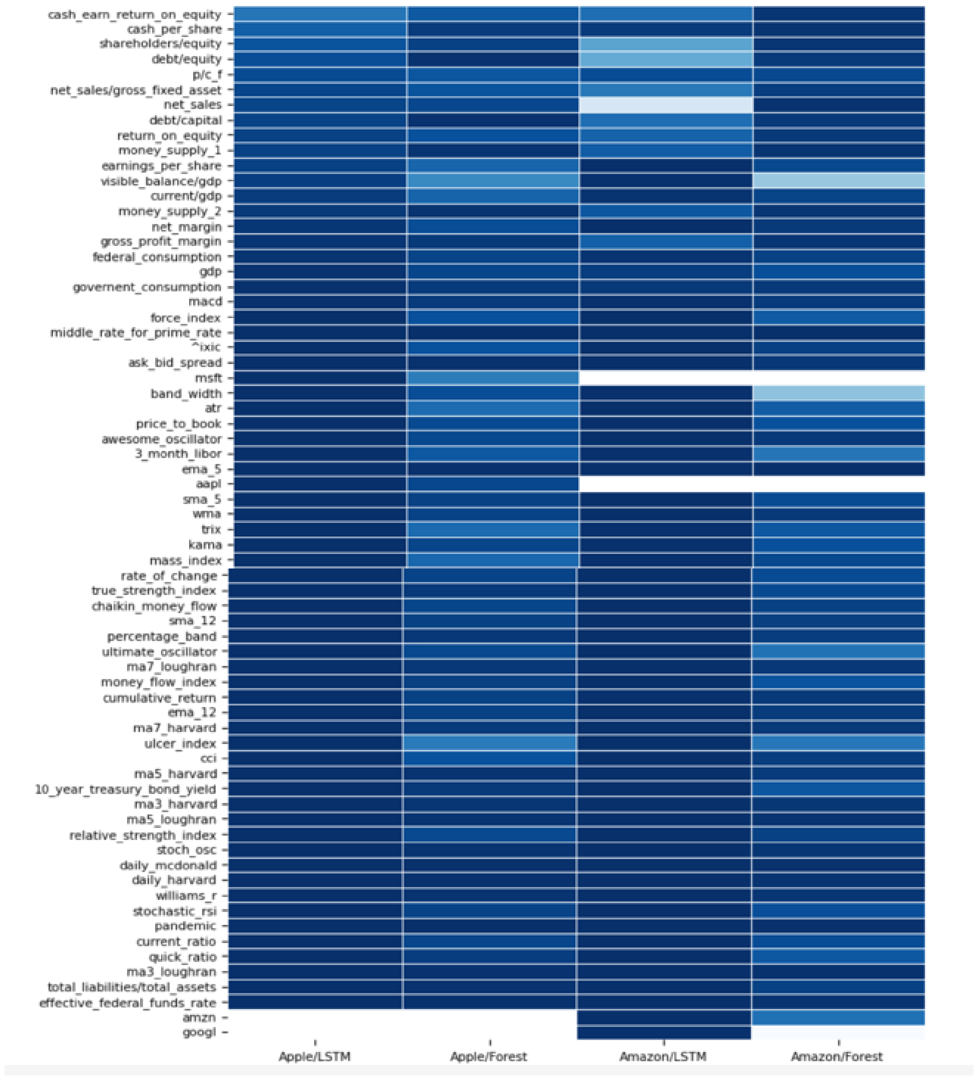


Figure 17: Heatmap illustrating the importance of all features for different models across all windows. Lighter shades of blue indicate higher importance, while darker shades denote lower importance.

the underlying patterns, leading to a better performance of the random forest in Window 5 than LSTM.

In Window 6, where the models were trained during the onset of the pandemic, we observe that the dummy variable for the pandemic is present for both models. Interestingly, while LSTM maintains the fundamental indicators in the first places, random forest gives more space to technical indicators. Again, here we observe the same pattern where the latter spreads out the importance among the 20 features. Considering that the RMSE for the LSTM in Window 6 is lower than the RMSE of the random forest, it can be inferred that the LSTM's feature selection and ranking offer superior predictive insights compared to the random forest.

For Amazon, we observe a pattern similar to Apple's among the two models in Window 5. LSTM assigns more weight to fundamental indicators, while random forest includes a more diversified "basket" of variables, with the technical indicators having a heavier impact. Yet, the significance of the three first indicators is present. Notably, the LSTM model shows a

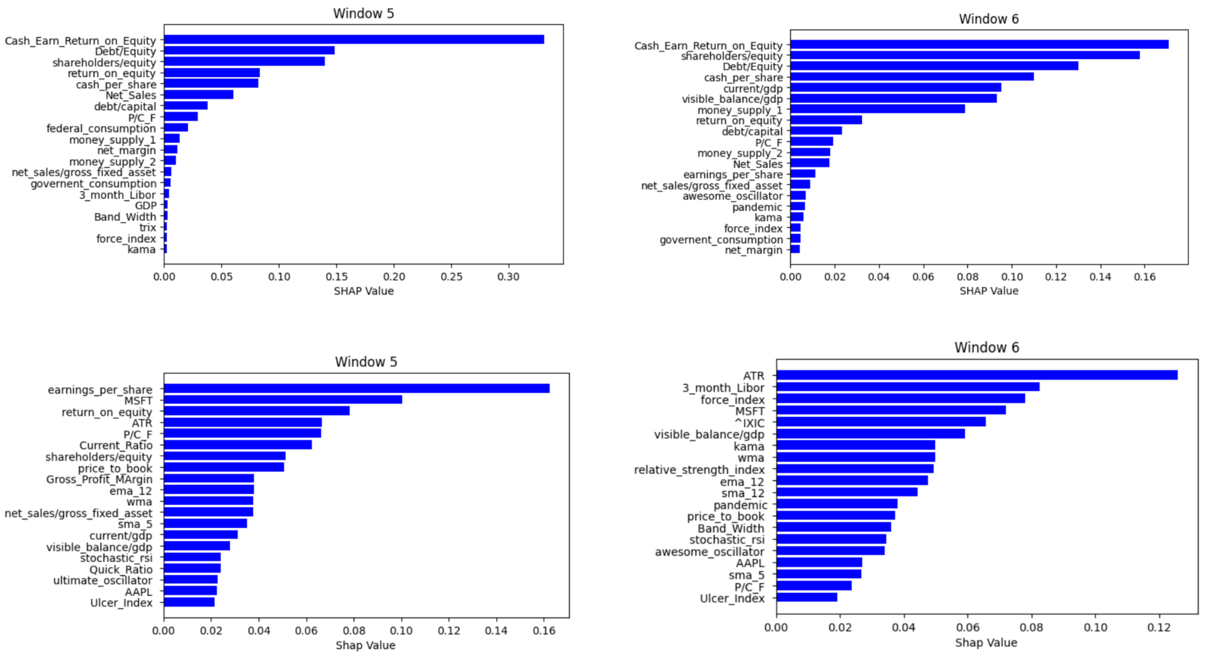


Figure 18: The top 20 features for Apple Stock across Windows 5 and 6. The top row represents the LSTM model, and the bottom row represents the Random Forest model.

pronounced influence from news articles (MA3 Loughran/Mcdonald). Given the superior performance of LSTM for Amazon, it suggests that financial health and business performance metrics offer heightened predictive accuracy.

For Window 6, the LSTM model focuses on Amazon's fundamental attributes and general economic indicators. In contrast, the random forest model integrates a broader range of factors, including technical metrics, sentiment indicators, and even the performance of industry peers like Google. This divergence underscores the different methodologies and priorities of the two models during this period. Given that the RMSE for the random forest was lower, that indicates that during the training within the pandemic period, a broader range of factors can give a better overall view of the movement of the TRV.

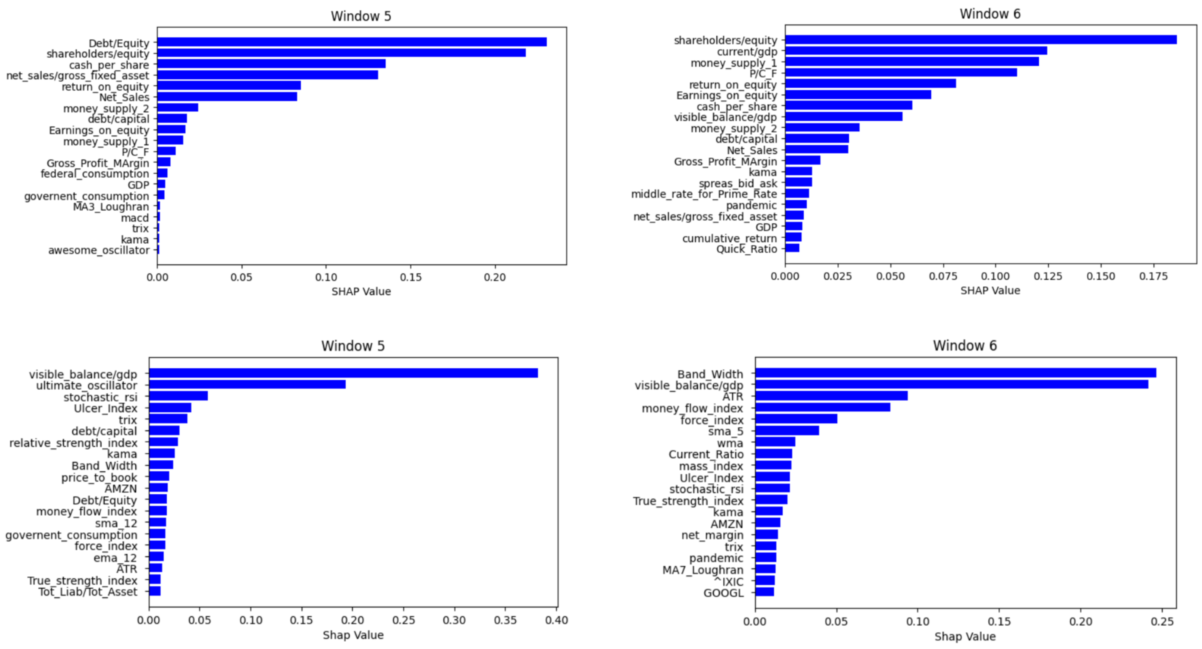


Figure 19: The top 20 features for Amazon Stock across Windows 5 and 6. The top row represents the LSTM model, and the bottom row represents the Random Forest model.

6 Conclusion and Discussion

The primary objective of this thesis was to predict short-term stock volatility (5th day ahead) for Apple and Amazon stocks. Our investigation included a comparative analysis of various models in a univariate setting, spanning both statistical and machine learning methods. We also evaluated the multivariate performance of two models to determine if multiple features enhance predictive accuracy.

Chapter 2 introduced the dependent variable as the True Realized Volatility with a 5-day moving average. We incorporated 63 variables from diverse fields, including technical, fundamental, macroeconomic, and sentiment analysis, plus a pandemic dummy variable. **Chapter 3** described our methodology, emphasizing the rolling window approach. This strategy segmented the 2013-2023 interval into seven windows, facilitating an in-depth analysis of each period. **Chapter 4** detailed the theoretical foundations and implications of the models used, while **Chapter 5** presented our findings.

Benchmarking against a naïve model, our comparative analysis began with statistical models: ARIMA, ARCH, and GARCH. The latter two were implemented using both AIC and BIC criteria. Aggregating their forecasts via the AFTER technique, GARCH consistently outperformed, achieving an overall RMSE of 0.758 for Apple and 0.958 for Amazon. However, ARIMA did not surpass the benchmark.

We evaluated five feed forward neural networks and one LSTM in the machine learning analysis. For Apple, LSTM outperformed, whereas, for Amazon, the three-layered FFNN (NN3) was superior, with an RMSE of 0.875. Interestingly, the highest RMSEs were found within the pandemic's onset (Window 5). Moreover, Amazon exhibited unusual activity in Window 7, 2019-2022.

In the multivariate setting, we focused on LSTM and random forest models. Using the 20 most significant features (out of 63) based on their SHAP values, we found that univariate models typically outperformed multivariate ones, except during abnormal periods. For instance, during the pandemic, Apple's random forest yielded an RMSE of 0.978, better than other models. Similarly, Amazon's multivariate LSTM outshone the rest. This suggests that integrating diverse features boosts model performance during high variability periods.

In the overall comparison, we observed consistent robustness from GARCH on a weekly basis, while from a monthly point of view, LSTM for Apple and NN1 for Amazon showed their superiority.

Regarding feature importance, random forest favored technical indicators, followed by a broader array of significant ones, whereas LSTM leaned towards fundamental ones. Notably, during the pandemic, Apple's performance was influenced by Microsoft stock lags and market sentiment indicators like the Stochastic RSI, Ulcer Index, and Ultimate Oscillator. For Amazon, fundamentals and sentiment from news articles proved crucial, signifying the firm's financial health and business performance as key determinants during the pandemic.

This thesis establishes a foundation for a range of promising directions for future research. One promising direction is the exploration of ensemble methods that blend the robustness of statistical models, such as GARCH, with the adaptability of machine learning techniques, notably LSTM and feed forward neural networks with one to three layers. Their effectiveness, as evidenced by their consistent accuracy in this study, suggests that a hybrid approach could yield substantial benefits.

Our investigation into multivariate models revealed that in 'normal' periods, where volatility is low, including a broad array of features does not substantially improve predictive accuracy.

This raises questions about models' optimal complexity during market stability. Moreover, noise traders—whose decisions are often driven by sentiment rather than fundamental analysis—posit a significant influence on equity pricing, especially in the short term. However, in this thesis, the collection of comprehensive, daily news articles to analyze sentiment presented challenges regarding sourcing and time constraints. Therefore, focused sentiment analysis might unveil subtle patterns not readily detected by conventional features, offering a more nuanced understanding of market behavior.

Building upon these insights, future research could benefit from incorporating sentiment analysis into established financial models. Although this thesis focused on predicting stock volatility, extending the Fama French model to predict stock returns with an added sentiment factor is a promising direction. A cross-sectional approach that assesses the differential impact of positive and negative news across firms and incorporates it as a distinct factor. After all, the evolving field of behavioral finance seems a direction that can offer research opportunities and provide deeper insights into the mechanisms of stock returns and, consequently, stock volatility.

7 Appendix

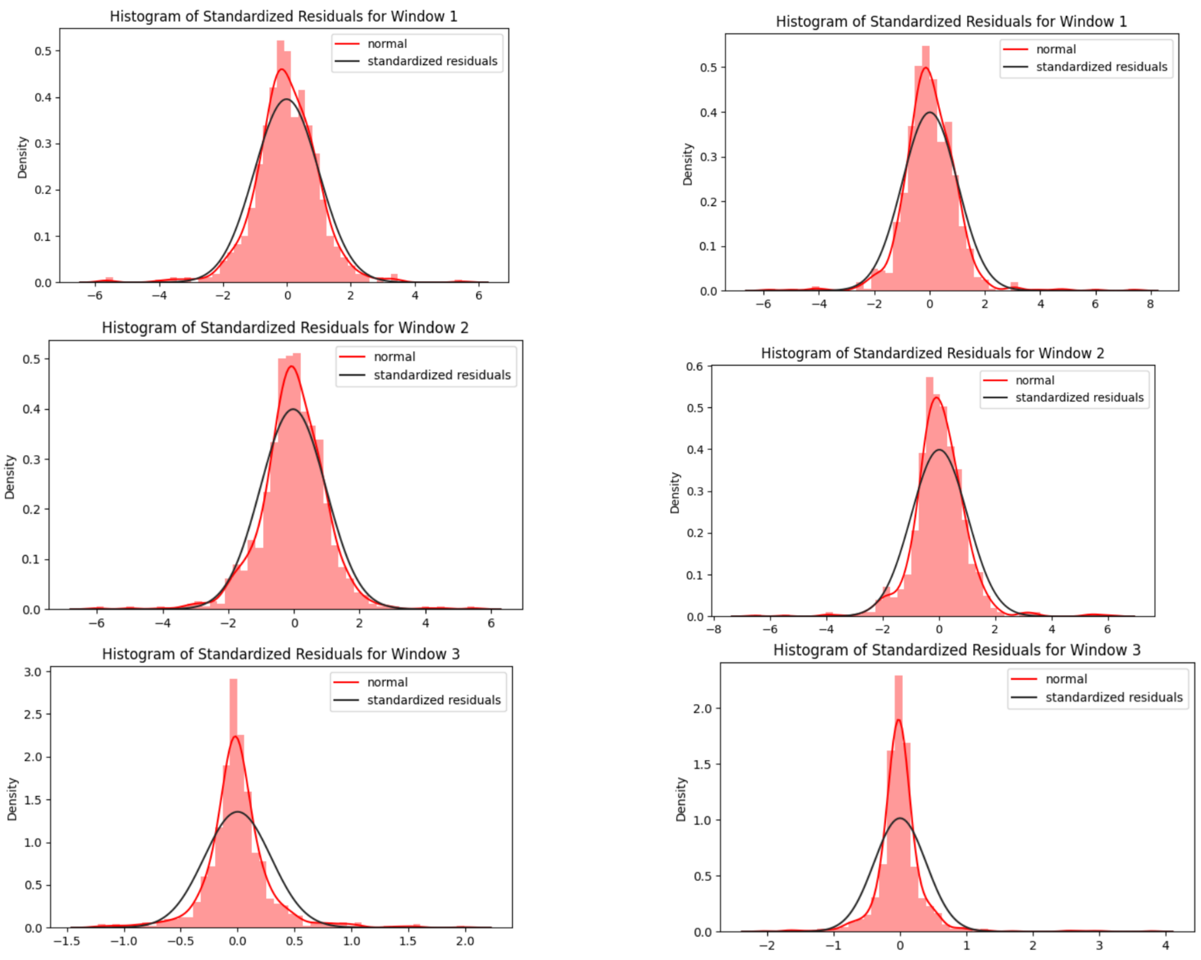


Figure 20: Diagnostic plots verifying the assumption of Gaussian errors for parameter estimation in ARIMA, ARCH, and GARCH models. Top row: Standardized residuals from the GARCH model fitted to Apple's log returns (left) and Amazon's log returns (right). Middle row: Standardized residuals plot from the ARIMA model fitted to Apple's TRV (left) and Amazon's TRV (right). Bottom row: Standardized residuals plot from the ARCH model fitted to Apple's log returns (left) and Amazon's log returns (right). These plots empirically support the assumption of Gaussian errors, consistent with the theoretical underpinnings highlighted by Bollerslev & Wooldridge (1992).

Stock	Window	$\hat{\omega}$	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\alpha}_3$	$\hat{\alpha}_4$	$\hat{\alpha}_5$
Apple	1	1.6213	0.3920	0.0156	0.0931	-	-
		(5.611)	(1.825)	(0.620)	(1.545)	-	-
	2	1.5811	0.1595	0.0615	0.1027	-	-
		(6.164)	(2.041)	(1.012)	(1.730)	-	-
	3	1.2442	0.1573	0.0000	0.1195	0.0365	0.0909
		(4.816)	(2.161)	(0.000)	(2.039)	(0.881)	(0.860)
	4	0.9914	0.1137	0.0995	0.1646	0.1238	0.1151
		(4.416)	(2.338)	(1.308)	(1.939)	(1.078)	(1.192)
	5	1.5301	0.1429	0.0561	0.1996	-	-
		(4.718)	(1.881)	(0.664)	(1.914)	-	-
	6	1.9088	0.1178	0.1937	0.2300	0.0777	-
		(3.857)	(2.030)	(2.521)	(2.606)	(0.840)	-
	7	1.4191	0.0821	0.1591	0.1658	0.2740	0.0483
		(6.109)	(1.865)	(2.387)	(2.422)	(1.630)	(0.920)
Amazon	1	2.4397	0.2761	0.0000	0.0000	0.0769	0.0779
		(4.369)	(1.783)	(0.000)	(0.000)	(0.501)	(0.473)
	2	2.3994	0.2773	0.0000	0.0000	0.2302	-
		(4.684)	(1.822)	(0.000)	(0.000)	(1.378)	-
	3	1.0196	0.3813	0.0063	0.1373	0.4752	-
		3.996	(1.568)	(0.376)	(1.194)	(2.093)	-
	4	0.6618	0.1262	0.1018	0.1303	0.4155	0.2262
		4.570	(2.614)	(1.724)	(2.379)	(2.190)	(2.353)
	5	0.7619	0.1923	0.1166	0.0295	0.3983	0.1610
		5.286	(2.769)	(1.742)	(0.564)	(1.702)	(2.829)
	6	1.0264	0.2199	0.1301	0.1568	0.1241	0.2124
		(4.503)	(3.425)	(2.189)	(2.571)	(1.323)	(3.165)
	7	1.5559	0.1542	0.0189	0.2580	0.0000	0.1194
		(1.781)	(1.274)	(0.603)	(2.460)	(0.000)	(1.618)

Table 27: ARCH Model Coefficients and t-statistics for Amazon and Apple Stocks across Different Windows

8 References

References

Alam, M. Z., Siddikee, M. N. & Masukujjaman, M. (2013), 'Forecasting volatility of stock indices with arch model', *International Journal of Financial Research* **4**(2), 126.

Avramov, D., Cheng, S. & Metzker, L. (2023), 'Machine learning vs. economic restrictions: Evidence from stock return predictability', *Management Science* **69**(5), 2587–2619.

Bettman, J. L., Sault, S. J. & Schultz, E. L. (2009), 'Fundamental and technical analysis: substitutes or complements?', *Accounting & Finance* **49**(1), 21–36.

Bollerslev, T. (1986), 'Generalized autoregressive conditional heteroskedasticity', *Journal of econometrics* **31**(3), 307–327.

Bollerslev, T. & Wooldridge, J. M. (1992), 'Quasi-maximum likelihood estimation and inference in dynamic models with time-varying covariances', *Econometric reviews* **11**(2), 143–172.

Bouri, E., Gkillas, K., Gupta, R. & Pierdzioch, C. (2021), 'Forecasting realized volatility of bitcoin: The role of the trade war', *Computational Economics* **57**, 29–53.

Bucci, A. (2020), 'Cholesky–ann models for predicting multivariate realized volatility', *Journal of Forecasting* **39**(6), 865–876.

Chatterjee, A., Bhowmick, H. & Sen, J. (2022), Stock volatility prediction using time series and deep learning approach, *in* '2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)', IEEE, pp. 1–6.

Christensen, K., Siggaard, M. & Veliyev, B. (2021), 'A machine learning approach to volatility forecasting', *Available at SSRN* .

Costa, F. J. M. (2017), Forecasting volatility using GARCH models, PhD thesis, Universidade do Minho (Portugal).

DeRobertis, N. (2020), 'Pysentiment'.
URL: *https://nickderobertis.github.io/pysentiment*

Dixit, G., Roy, D. & Uppal, N. (2013), 'Predicting india volatility index: An application of artificial neural network', *International Journal of Computer Applications* **70**(4).

Emir, S., Dinçer, H. & Timor, M. (2012), 'A stock selection model based on fundamental and technical analysis variables by using artificial neural networks and support vector machines', *Review of Economics & Finance* **2**(3), 106–122.

Engle, R. F. (1982), 'Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation', *Econometrica: Journal of the econometric society* pp. 987–1007.

Engle, R. F. & Patton, A. J. (2001), 'What good is a volatility model?', *Quantitative finance* **1**(2), 237.

Fama, E. F. (1965), 'The behavior of stock-market prices', *The journal of Business* **38**(1), 34–105.

Filipović, D. & Khalilzadeh, A. (2021), 'Machine learning for predicting stock return volatility', *Swiss Finance Institute Research Paper* (21-95).

Gu, S., Kelly, B. & Xiu, D. (2020), 'Empirical asset pricing via machine learning', *The Review of Financial Studies* **33**(5), 2223–2273.

Hansen, P. R. & Lunde, A. (2005), 'A forecast comparison of volatility models: does anything beat a garch (1, 1)?', *Journal of applied econometrics* **20**(7), 873–889.

Hyndman, R. J. & Athanasopoulos, G. (2018), *Forecasting: principles and practice*, OTexts.

Ibrahim, S. O. (2017), 'Forecasting the volatilities of the nigeria stock market prices', *CBN Journal of Applied Statistics* **8**(2), 23–45.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', pmlr, pp. 448–456.

Kim, H. Y. & Won, C. H. (2018), 'Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models', *Expert Systems with Applications* **103**, 25–37.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kirui, E., Wawire, N. H. & Onono, P. A. (2014), 'Macroeconomic variables, volatility and stock market returns: a case of nairobi securities exchange, kenya'.

Koukaras, P., Nousi, C. & Tjortjis, C. (2022), Stock market prediction using microblogging sentiment analysis and machine learning, *in* 'Telecom', Vol. 3, MDPI, pp. 358–378.

Krantz, M. (2023), *Fundamental analysis for dummies*, John Wiley & Sons.

Kyoung-Sook, M. & Hongjoong, K. (2019), 'Performance of deep learning in prediction of stock market volatility.', *Economic Computation & Economic Cybernetics Studies & Research* **53**(2).

Lam, M. (2004), 'Neural network techniques for financial performance prediction: integrating fundamental and technical analysis', *Decision support systems* **37**(4), 567–581.

Li, X., Wu, P. & Wang, W. (2020), 'Incorporating stock prices and news sentiments for stock market prediction: A case of hong kong', *Information Processing & Management* **57**(5), 102212.

Lopez Padial, D. (2018), 'Technical analysis library in python'.
URL: *https://technical-analysis-library-in-python.readthedocs.io/en/latest/*

Ludwig, N., Feuerriegel, S. & Neumann, D. (2015), 'Putting big data analytics to work: Feature selection for forecasting electricity prices using the lasso and random forests', *Journal of Decision Systems* **24**(1), 19–36.

Luong, C. & Dokuchaev, N. (2018), 'Forecasting of realised volatility with the random forests algorithm', *Journal of Risk and Financial Management* **11**(4), 61.

Mademlis, D. K. & Dritsakis, N. (2021), 'Volatility forecasting using hybrid garch neural network models: The case of the italian stock market', *International Journal of Economics and Financial Issues* **11**, 49–60.
URL: *https://api.semanticscholar.org/CorpusID:231815268*

Malkiel, B. G. (2003), 'The efficient market hypothesis and its critics', *Journal of economic perspectives* **17**(1), 59–82.

McAleer, M. & Medeiros, M. C. (2008), 'Realized volatility: A review', *Econometric reviews* **27**(1-3), 10–45.

Miswan, N. H., Ngatiman, N. A., Hamzah, K. & Zamzamin, Z. Z. (2014), 'Comparative performance of arima and garch models in modelling and forecasting volatility of malaysia market properties and shares', *Applied Mathematical Sciences* **8**(140), 7001–7012.

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

Namdari, A. & Li, Z. S. (2018), Integrating fundamental and technical analysis of stock market through multi-layer perceptron, *in* '2018 IEEE technology and engineering management conference (TEMSCON)', IEEE, pp. 1–6.

Nousi, C. (2021), 'Stock market prediction using sentiment analysis'.

Pai, P.-F. & Lin, C.-S. (2005), 'A hybrid arima and support vector machines model in stock price forecasting', *Omega* **33**(6), 497–505.

Patil, R. (n.d.), 'Time series analysis and stock price forecasting using machine learning techniques'.

Picasso, A., Merello, S., Ma, Y., Oneto, L. & Cambria, E. (2019), 'Technical analysis and sentiment embeddings for market trend prediction', *Expert Systems with Applications* **135**, 60–70.

Ramos-Pérez, E., Alonso-González, P. J. & Núñez-Velázquez, J. J. (2019), 'Forecasting volatility with a stacked model based on a hybridized artificial neural network', *Expert Systems with Applications* **129**, 1–9.

Roh, T. H. (2007), 'Forecasting the volatility of stock price index', *Expert Systems with Applications* **33**(4), 916–922.

Serafini, G., Yi, P., Zhang, Q., Brambilla, M., Wang, J., Hu, Y. & Li, B. (2020), Sentiment-driven price prediction of the bitcoin based on statistical and deep learning approaches, *in* '2020 International Joint Conference on Neural Networks (IJCNN)', IEEE, pp. 1–8.

Sudhakar, K. & Naganjaneyulu, S. (2020), 'Stock price prediction based on finance related news using nlp, lasso and arimax', *i-Manager's Journal on Software Engineering* **14**(4), 11.

Taylor, S. J. (2008), *Modelling financial time series*, world scientific.

Thu, H. G. T., Thanh, T. N. & Le Quy, T. (2021), A neighborhood deep neural network model using sliding window for stock price prediction, *in* '2021 IEEE International Conference on Big Data and Smart Computing (BigComp)', IEEE, pp. 69–74.

Wadhawan, D. & Singh, H. (2019), 'Estimating and forecasting volatility using arima model: A study on nse, india', *Indian Journal of Finance* **13**(5), 37–51.

WAHYUDI, S. T. (2017), 'The arima model for the indonesia stock price.', *International Journal of Economics & Management* **11**.

Xiong, R., Nichols, E. P. & Shen, Y. (2015), 'Deep learning stock volatility with google domestic trends', *arXiv preprint arXiv:1512.04916* .

Yang, Y. (2004), 'Combining forecasting procedures: Some theoretical results', *Econometric Theory* **20**(1), 176–222.

Zou, H. & Yang, Y. (2004), 'Combining time series models for forecasting', *International journal of Forecasting* **20**(1), 69–84.

Zou, Z. & Qu, Z. (2020), 'Using lstm in stock prediction and quantitative trading', *CS230: Deep Learning, Winter* pp. 1–6.