Handelshøyskolen BI

GRA 19703 Master Thesis

Thesis Master of Science 100% - W

Predefinert informasjon

Startdato:	09-01-2023 09:00 CET	**Termin:**	202310
Sluttdato:	03-07-2023 12:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	T		
Flowkode:	202310||11184||IN00||W||T		
Intern sensor:	(Anonymisert)		

Deltaker

Navn:	Adrian Nygård Nielsen og Herman Hansson

Informasjon fra deltaker

Tittel *:	The Power of Sentiment and Attention: Twitter and Google Trends as predictors of stock returns
Navn på veileder *:	Salvatore Miglietta

Inneholder besvarelsen konfidensielt materiale?:	Nei	**Kan besvarelsen offentliggjøres?:**	Ja

Gruppe

Gruppenavn:	(Anonymisert)
Gruppenummer:	159
Andre medlemmer i gruppen:	

Adrian Nielsen Herman Hansson

BI Norwegian Business School

The Power of Sentiment and Attention: Twitter and Google Trends as predictors of stock returns

Supervisor:
Salvatore Miglietta

Hand-in date:
03.06.2022

Campus:
BI Oslo

Examination code and name:
GRA 19701 Master Thesis

Programme:
Master of Science in Finance

Abstract

This thesis examines the impact of Twitter sentiment and Google Trends-derived investor attention on stock returns of Apple, Amazon, Microsoft, and Tesla. Spanning from January 1, 2018, to February 1, 2023, we extracted 0.55 million tweets, constructed a daily Google Trends Search Volume Index (SVI), and gathered adjusted close prices. Sentiment classification is done through a dual approach, integrating lexicon-based classification with machine-learning. By setting up multiple VAR models on first differences, we compare them to a random walk with drift and an AR model based solely on stock lags. We consistently outperform the random walk, challenging the efficient market hypothesis. Compared to the AR model, the results were mixed across the stocks, suggesting that investor behaviour may contribute to market inefficiencies for certain stocks. Our results suggest limited predicting power for Sentiment and SVI. SVI shows slight superiority over sentiment in predictive power, although its impact remains limited, with stock price fluctuations predominantly tied to their historical performance.

Acknowledgments

We would like to express our sincere gratitude to Salvatore Miglietta, our supervisor from the Department of Finance at BI Norwegian Business School, for his invaluable guidance and unwavering support throughout this master's thesis. His expertise and attention have been instrumental in shaping this work. We are truly fortunate to have had the opportunity to work under his mentorship.

We are also grateful to our friends, fellow students, and family for their support and encouragement during this five-year journey. Their belief in us has been a constant source of motivation.

Table of Content

Introduction

The predictability of stock market prices has been a subject of fervent discussion in finance and economics, with the Efficient Market Hypothesis (EMH) proposed by (Fama, 1965) asserting that price fluctuations are random, and forecasting is fundamentally flawed, thus challenging the prospect of long-term abnormal trading profits. However, the emergence of alternative data sources like social media sentiment and search volume, which capture the collective mind – people's mood and attention is disputing this traditional theory, proposing that markets may have a degree of predictability. This notion of predictability is grounded in behavioural finance, a field connecting finance and social sciences to understand the role of human psychology in financial markets, asserting that emotions and attention play a significant role in investment decision-making (Hirshleifer & Hong Teoh, 2003; Shiller, 2003).

Sentiment analysis, a subfield of natural language processing, has emerged as a powerful technique for extracting and quantifying the underlying sentiment in textual data, allowing for the exploration of the potential impact of social media sentiment on financial markets (Tetlock, 2007). We have chosen to utilize Twitter for sentiment analysis due to its real-time information dissemination, large user base, and potential to quickly capture market-influencing sentiments, making it an optimal data source for understanding and potentially predicting financial market dynamics.

Parallel to sentiment analysis, Google Trends provides data on search term frequency for particular keywords or phrases over time, gauging public attention toward specific subjects such as corporations (Da et al., 2011). We utilize Google Trends as it gauges current investor attention and offers potential predictive insights into market movements, providing an additional quantitative perspective to understand and analyse investor behaviour and interest through attention.

This paper investigates the relationship between Twitter sentiment, Google Trends-derived investor attention, and stock returns for four prominent technology companies: Apple, Amazon, Microsoft, and Tesla. Our research is motivated by the growing interest in understanding the role of social media sentiment and

investor attention in shaping stock market dynamics. We hypothesize that Twitter sentiment and investor attention, as measured by the Google Trends Search volume index, significantly impact stock returns in line with behavioural finance theory (BFT), providing valuable insights for investors and financial analysts.

We employ a comprehensive analysis methodology that encompasses the implementation of multiple vector autoregression (VAR) models to execute one-step-ahead forecasts for an out-of-sample period of 30 days, along with conducting various causality tests and variance decompositions. To assess the performance of our VAR models, we compare them against two benchmark models: random walk with drift (RW) and autoregressive (AR) models that include only the lags of the stocks. We utilize a rich dataset of 0,55 million tweets, daily Google Trends Search Volume Index (SVI), and adjusted closing prices for the selected companies, collected over a five-year period from January 1, 2018, to January 1, 2023. Twitter and SVI data are extracted during both trading and non-trading hours. Our approach combines two sentiment analysis techniques, the lexicon-based approach through the Loughran-McDonald dictionary and the Hugging Face sentiment analysis pipeline, to provide a robust and accurate measure of Twitter sentiment.

The VAR model consistently outperformed the RW models, challenging the EMH. Compared to the AR model, the results were mixed across the stocks, suggesting that investor behaviour may contribute to market inefficiencies for certain stocks. Granger causality tests identified minimal causal relationships, with exceptions at both 5% and 10% significance level implying the presence of possible market inefficiencies. Impulse response analysis reveals small effects on stock prices of shocks in sentiment and SVI, reinforcing the limited influence of these variables on stock price movements in line with EMH. Furthermore, the variance decomposition tests confirmed the limited power of sentiment and attention variables in predicting stock price variance. Overall, our findings indicate that while Google Trends Search Volume Index possesses marginally more predictive power than sentiment, its influence remains limited, with the lion's share of stock price variations primarily attributable to the stocks' own historical performance. These findings align with aspects of both the EMH and

BFT underscoring the importance of conducting further research to examine the impact of psychological biases on market behaviour.

Our research enhances the existing body of knowledge by uniquely synthesizing sentiment and attention variables into a comprehensive predictive model, thereby offering an in-depth understanding of the interplay between social media sentiment, investor attention, and stock market dynamics on company level. This approach, which incorporates a broader timeline compared to most studies, employs a novel dual sentiment analysis technique that merges lexicon-based methods with advanced machine learning algorithms. By unveiling the potential of these combined variables and methodologies, our study holds profound implications for investors, financial analysts, and other stakeholders, promising to redefine investment decisions and strategies, and paving the way for novel financial forecasting paradigms.

Literature review

This literature review explores the role of investor sentiment, as conveyed through various media and data sources, in asset pricing and market predictions. It examines research investigating the influence of diverse mediums, from financial news to Twitter, on market volatility and stock returns. The review also considers Google Trends, as a measure of investor attention and its potential predictive value for stock market movements.

Various studies analysed diverse textual content such as media articles and corporate disclosures, demonstrating the influential role of textual content and investor sentiment in asset pricing. Online discussions on finance boards (Antweiler & Frank, 2004), negative media tone (Ahmad et al., 2016) and pessimistic sentiments in financial news, as in a prominent Wall Street Journal column (Tetlock, 2007) can significantly predict market volatility and stock returns. Moreover, mood-altering events like international soccer results have been found to significantly affect stock returns (Edmans et al., 2007), further suggesting that mood shifts may drive market behaviours. A similar predictive capacity is also observable in the linguistic tone of Forward-Looking Statements (FLS) in corporate filings (Li, 2010) and quarterly earnings conference calls (Price et al., 2012). However, potential misclassification issues necessitate caution with word classification schemes (Loughran & Mcdonald, 2011). The significant influence of investor sentiment, especially in the context of securities with subjective valuations and limited arbitrage opportunities, is apparent in stock prices and returns (Baker & Wurgler, 2006). These studies question the Efficient Market Hypothesis by highlighting sentiment-driven market anomalies, thus aligning with behavioural finance theory's emphasis on the role of investor biases in asset pricing.

Investigations into user-generated content reveal that Twitter and other stock microblogs can provide a trove of valuable market information (Sprenger et al., 2014). Pioneer studies by (Bollen et al., 2011) claimed that Twitter sentiment analysis could forecast DJIA closing values with an accuracy of 87.6%, and similar studies highlighted a significant correlation between tweet emotions and U.S. stock indices (Zhang et al., 2011). However, the reliability and

generalizability of these findings have been disputed due to issues of data overfitting and potential limitations in the use of randomized subsamples (Lachanski & Pav, 2017; Sprenger et al., 2014).

The use of social media sentiment analysis for predicting individual stock movements has increasingly been explored, with research moving from aggregate market indices to stock-specific predictions. Researchers have developed sentiment indices for individual stocks (Oh & Sheng, 2011) and demonstrated that Twitter sentiment can significantly impact stock returns, with a trading strategy based on user sentiments potentially yielding annual returns between 11–15% (Sul et al., 2017a). The importance of Twitter sentiment has been further recognized in abnormal returns at Twitter volume peaks (Ranco et al., 2015) and in predicting trade volume and price changes of specific stocks like Apple (Batra & Daudpota, 2018; Mao et al., 2012). However, not all stocks are affected equally, as (Renault, 2020) found social media sentiment extracted from StockTwits failed to predict Apple, Amazon, Facebook, Google, and Microsoft's daily returns.

The utility of Google Trends in reflecting investor attention was pioneered by (Da et al., 2011), finding that increases in the search volume signalled higher prices for Russell 3000 stocks over a two-week span, followed by a price reversal within a year. This was complemented by (Joseph et al., 2011) who, through analysis of S&P 500 firms from 2005–2008, discovered online search intensity could reliably predict abnormal stock returns and trading volumes, especially for stocks that are hard to arbitrage. Following up on these findings, (Huang et al., 2020) further demonstrated the predictive power of Google Trends data. Their study concluded that such data could be effectively used to construct a relatively simple linear model that forecasts the directional movements of the S&P 500 index. This model could potentially generate substantial excess returns in the back testing period, although they also acknowledged certain caveats. In a similar vein, (Preis et al., 2010) found a robust correlation between weekly search volumes for specific companies and their transaction volumes in the S&P 500. (Chen & Lo, 2019) extended this analysis to Taiwan's top 50 firms, revealing a significant positive correlation between the logarithmic variation of SVI and trading volume, turnover ratio, and stock return volatility. In sum, these studies establish a consistent narrative: the intensity of searches for ticker symbols is a valid reflection of

investor attention, serving as a potent forecasting tool for stock returns and volumes.

The review reveals that various data sources, like media articles, corporate disclosures, online chats, Twitter sentiment, and Google Trends, are gradually gaining recognition as tools for predicting market volatility and stock returns. These studies question established norms, including the EMH by pointing at potential market inefficiencies aligning with the BFT. Although the potential of Twitter sentiment and Google Trends for market forecasts has been recognized, their combined use in predicting individual stock returns remains largely unexplored. Despite the engaging narrative presented, this field is still nascent, with a portion of the research appearing in less recognized journals and having fewer citations. This presents an opportune moment for our forthcoming study, aiming to delve deeper into this underexplored interplay between social media sentiment, investor attention, and stock prices of leading tech companies, thereby enriching the existing literature and enhancing our understanding of these innovative market prediction methodologies.

Hypothesis and Methodology

At the core of our study lies the behavioural hypothesis, which suggests that the interaction between investor mood and attention, as measured by Twitter sentiment and Google Trends search volume index (SVI) respectively, has a significant impact on stock returns. With the existing literature yielding diverse findings and the relative novelty of this field, we are captivated by the potential of combining Twitter sentiment and SVI to enhance the prediction of stock price movements. Our particular focus is on examining the intricate interplay between these variables and their influence on the stock prices of four prominent technology companies. Grounded in this context, our methodology sets to answer a set of hypotheses. First, we propose the existence of a causal relationship between Twitter sentiment and the stock price movements of the selected technology companies. Second, we posit a causal relationship between SVI and the stock price movements of the same companies. Lastly, we aim to evaluate the extent to which combining Twitter sentiment and SVI improves the predictability of stock price movements.

Stationarity

Commencing with our analysis, the execution of an Augmented Dickey-Fuller (ADF) test is in order to evaluate the stationarity of the dataset. This is a crucial component of our methodology, given our objective to establish multiple VAR models, which inherently require all associated variables to exhibit stationarity (Brooks, 2019). The primary function of the ADF test is to examine the time series for the existence of a unit root. The null hypothesis propounded by the ADF test posits that the time series incorporates a unit root ($\phi = 1$), signifying its non-stationary nature. Conversely, the alternative hypothesis implies the stationarity of the time series ($\phi < 1$).

To guarantee the stationarity of the time series data, we introduce the first differences to each variable prior to conducting the test. As depicted in Table 1, the stationarity of all variables is corroborated by a p-value that closely approximates zero, reinforcing the graphical depiction of the first differences presented in appendix Section C.

Selecting the optimal number of lags

Selecting the optimal number of lags is a crucial step in building a VAR model for time series analysis. Several information criteria methods have been proposed for this purpose, including Akaike's Information Criterion (AIC) (Akaike, 1969), Final Prediction Error (FPE) (Akaike, 1974), Hannan-Quinn Information Criterion (HQIC) (Hannan & Quinn, 1979), and Schwarz Bayesian Information Criterion (SBIC) (Schwarz, 1978) (Brooks, 2019).

The choice of which information criterion to use for determining the optimal lag number depends on the specific characteristics of the data and the objective of the analysis. The AIC and FPE tends to favour models with more lags, as it penalizes model complexity less severely. This may lead to overfitting, where the model may capture noise in the data rather than true underlying patterns. On the other hand, the HQIC and SBIC tend to favour models with fewer lags, as they impose a stronger penalty on model complexity, thereby mitigating the risk of overfitting.

Table 2 presents a summary of the results obtained calculating all four information criteria to determine the optimal lag number for each model. For all four stocks, we have chosen to use the Hannan-Quinn Information Criterion (HQIC) for setup (1) involving all three variables, along with setup (2) involving adjusted closing price and sentiment score. Additionally, for setup (3) with adjusted closing price and SVI we have found the Akaike's Information Criterion (AIC) to be the most appropriate information criterion. Specifically, all models for Apple will use 9 lags. For Amazon, 6 lags will be used in the first setup, while the latter two setups will use 8 lags. Microsoft's first setup will also use 6 lags, and the following two will employ 9 lags. All Tesla models will include 5 lags.

VAR framework

The VAR approach is useful for handling time-series data, and it allows for multiple endogenous variables, giving us the ability to investigate dynamic effects without rigid limitations and hence is a good fit for our dataset. Another advantage of the VAR models is the allowance for a variable to be dependent on more than just its own lags and combinations of white noise terms. Thus, VARs

are more flexible than univariate AR models and can capture more features of the data as a result of their more complex structure (Brooks, 2019).

For each stock, the first model (1) will include the first differences of; the adj. closing price (S) for the stock, the net sentiment score (NSS), and SVI. In this model, we aim to explore how fluctuations in NSS, and SVI collectively affect the stock price movements. By considering all three variables, we can capture the combined impact of market sentiment and online search behaviour on stock price movements. This model allows us to examine the interconnected dynamics and potential feedback effects between these variables.

The second model (2) will include the first differences of; adj. closing price (S) for the stock and NSS as the endogenous variables. This model focuses on investigating the influence of NSS alone on stock price movements. By isolating the NSS as the primary predictor, we can assess the importance of market sentiment in driving stock price movements. This model helps us understand the extent to which changes in net positivity or negativity impact the stock return over time.

The third model (3) will include the first differences of; adj. closing price and the SVI. In this model, our objective is to examine the relationship between stock returns and SVI. By considering the search behaviour captured by the SVI, we can explore the extent to which online search activity reflects or predicts stock price movements. This model provides insights into the impact of public interest and information-seeking behaviour on stock returns.

Mathematically, the models can be written as follows in matrix form where S_t is the stock return, NSS_t is the change in net sentiment score, SVI_t is the change in Google trends search volume index, t is time, $\beta_{g,0}$ is the constant term, $\beta_{g,k}$ is the coefficient associated with the variables, $\alpha_{g,k}$ are the coefficients associated with the contemporaneous values of the variables, and u is the error terms.

$$\begin{bmatrix} S_t \\ NSS_t \\ SVI_t \end{bmatrix} = \begin{bmatrix} \beta_{10} \\ \beta_{20} \\ \beta_{30} \end{bmatrix} + \sum_{i=1}^{k} \begin{bmatrix} \beta_{11}^i & \alpha_{12}^i & \alpha_{13}^i \\ \alpha_{21}^i & \beta_{22}^i & \alpha_{23}^i \\ \alpha_{31}^i & \alpha_{32}^i & \beta_{33}^i \end{bmatrix} \begin{bmatrix} S_{t-i} \\ NSS_{t-i} \\ SVI_{t-i} \end{bmatrix} + \begin{bmatrix} u_{S_t} \\ u_{NSS_t} \\ u_{SVI_t} \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} S_t \\ NSS_t \end{bmatrix} = \begin{bmatrix} \beta_{10} \\ \beta_{20} \end{bmatrix} + \sum_{i=1}^{k} \begin{bmatrix} \beta_{11}^i & \alpha_{12}^i \\ \alpha_{21}^i & \beta_{21}^i \end{bmatrix} \begin{bmatrix} S_{t-i} \\ NSS_{t-i} \end{bmatrix} + \begin{bmatrix} u_{S_t} \\ u_{Pos_t} \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} S_t \\ SVI_t \end{bmatrix} = \begin{bmatrix} \beta_{10} \\ \beta_{20} \end{bmatrix} + \sum_{i=1}^{k} \begin{bmatrix} \beta_{11}^i & \alpha_{12}^i \\ \alpha_{21}^i & \beta_{21}^i \end{bmatrix} \begin{bmatrix} S_{t-1} \\ SVI_{t-i} \end{bmatrix} + \begin{bmatrix} u_{S_t} \\ u_{Top_t} \end{bmatrix} \quad (3)$$

Forecasting

Our research aims to investigate the predictive relationship between the net positivity of tweets and Google Trends scores, and their influence on stock price movements. Utilizing our estimated VAR model, which was trained on historical data spanning from 01.01.2018 to 01.01.2023, we conducted an out-of-sample one-step-ahead forecast. This forecast allowed us to generate predictions within the period from 01.01.2023 to 01.02.2023, representing the anticipated change in stock return for the subsequent trading day. By examining the impact of Twitter sentiment and Google Trends on stock price fluctuations, we gain valuable insights into the forecasting potential of these factors in the specified time horizon.

Comparing forecasting performance

To evaluate the predictive accuracy of our models, the out-of-sample forecast enables us to derive the accuracy metrics Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

By comparing our model's performance against random processes, we aim to establish its validity, supported by previous research such as the findings of (Dsouza & Mallikarjunappa, 2015), which suggest that stock prices do not follow a random walk and instead exhibit non-random underlying structures in the market. As the stock prices have increased in the sample period, we use a random walk with drift to account for any underlying trends or systematic changes observed in the stock prices. We employ the approach outlined by (Nau, 2014) to estimate the random walk with drift (RW) which assumes that the model takes a random step from its previous value in addition to a drift term.

$$\hat{Y}_{n+k} = Y_n + k\hat{d} \quad (4)$$

To estimate the drift term for each stock, we have computed the average difference between the adjacent price changes over the sample period as outlined by (Nau, 2014).

$$\hat{d} = \frac{Y_n - Y_1}{n - 1} \quad (5)$$

This average change serves as an approximation of the stock's drift, capturing the average increase or decrease in price from one period to the next. By adding the drift term to the most recent observed price, we can generate forecasts using the random walk with drift model.

As a second benchmark an autoregressive model (AR) model was constructed employing solely observations from prior time steps of changes in the adjusted close price as inputs to a regression equation, forecasting the value at the subsequent time step. This benchmark approach is driven by the EMH, allowing for assessing the disparity in accuracy metrics resulting from the inclusion of NSS and SVI variables in the VAR models compared to the AR model. The AIC was applied to all stocks, with 9 lags chosen for Apple and Microsoft, and 1 for Amazon and Tesla (Table 2). Mathematically, the change in adjusted closing price at time t (S_t) in the autoregressive model is a function of a constant term μ, the past (S_{t-i}) changes in adjusted closing prices each weighted by their respective coefficients (ϕ_i) and the error term at time t (u_t) which accounts for other influences not captured by the past prices (Brooks, 2019).

$$S_t = \mu + \sum_{i=1}^{p} \phi_i S_{t-i} + u_t \quad (6)$$

Autocorrelation

Autocorrelation is a statistical measure used to assess the similarity between a time series and its lagged version, revealing potential patterns or dependencies within the data. The presence of autocorrelation in residuals suggests that the model may not adequately capture the underlying data pattern. To evaluate

autocorrelation, we conduct the Durbin-Watson test, as suggested by (Brooks, 2019). The null hypothesis, $H_0: \rho = 0$, states that there is no autocorrelation present in the residuals, indicating that the model adequately captures the underlying data pattern. The alternative hypothesis $H_1: \rho \neq 0$, suggests the presence of autocorrelation, indicating that the model fails to capture certain dependencies within the data. The test statistic used in the Durbin-Watson test is calculated as follows:

$$DW = \frac{\sum_{t=2}^{T}(\hat{u}_t - \hat{u}_{t-1})^2}{\sum_{t=2}^{T}\hat{u}_t^2} \quad (7)$$

The Durbin-Watson analysis generates a test statistic that can range from 0 to 4. The proximity of the test statistic to 2 suggests the non-rejection of the null hypothesis, implying the absence of autocorrelation. Conversely, test statistic nearing the extremities of 0 and 4 suggest the null hypothesis can be refuted, indicating the existence of positive or negative autocorrelation within the residuals (Brooks, 2019).

The Durbin-Watson test results, presented in Table 3, reveal test statistics hovering around 2 for all considered models, thus suggesting the absence of autocorrelation. The range of test statistics spans from 1.9871 (for "Adj. Close Tesla | SVI Tesla") to 2.0244 (for "Adj. Close Tesla | Sentiment Tesla"), with the latter exhibiting the highest deviation from the expected value of 2. However, in this instance, we still don't find adequate evidence to dismiss the null hypothesis, thus we cannot reject the assumption of no autocorrelation across all models.

Causality

Causality tests are employed in our research to examine the interactions between different variables and how they affect each other. In this study, we will be conducting Granger causality tests, impulse response analysis, and variance decompositions. These tests will help us understand the direction, strength, and responsiveness of the relationships between our variables of interest.

Grange Causality

The Granger causality test, as described by (Brooks, 2019), utilizes an F-test framework to assess the direction and strength of causality between variables. By performing this test on all possible combinations of our variables, we can identify

which variables have a significant Granger-causal relationship and gain insights into the potential drivers of stock prices, sentiment, and investor attention.

Impulse Response Analysis

In addition to Granger causality, impulse response analysis is conducted to further explore the relationships between our variables. Impulse response functions provide insights into the responsiveness of the dependent variable in the VAR model to shocks in each of the variables. This analysis helps us understand the potential impact of unanticipated changes in one variable on the others and how the effects of these shocks may evolve over time (Brooks, 2019).

Variance decomposition

The variance decomposition is a method used to measure the contribution of each variable in the VAR model to the forecast error variance of the system over a specified period. The decomposition is carried out by calculating the percentage of the forecast error variance that can be attributed to the innovations of each variable in the system (Lütkepohl, 2005). The formula for the variance decomposition of a variable i at time t is given by:

$$FEV_i(t) = \sum_{j=1}^{p} \sum_{k=1}^{p} \phi_{i,j,k} FEV_j(t-k) + \sigma_i^2(t) \quad (8)$$

The results of variance decomposition provide valuable information regarding the individual impact of variables on the forecast error variance observed within a system. Analysing the percentage contribution of each variable to the overall forecast error variance helps assess its significance in explaining temporal variations within the system. Visualizing the variance decomposition results through a plot facilitates the interpretation of how the contribution of each variable evolves over time in relation to the forecast error variance of the system.

Data

Selection of companies

This research focuses on Apple, Amazon, Microsoft, and Tesla due to their substantial market value, high public visibility, and diverse industry representation. This selection facilitates a nuanced study of Twitter sentiment and investor attention's effects on stock prices across varying sectors (Zeitun et al., 2023). These companies frequently garner media attention and possess robust Twitter presences, enhancing their suitability for sentiment analysis research (Da et al., 2011). Additionally, their wide data availability and consistent growth patterns (Statman, 2010) enable a robust quantitative analysis of sentiment, attention, and stock performance. Previous research (Batra & Daudpota, 2018; Mao et al., 2012) has yielded promising results using some of these companies, underscoring their relevance to this field. Thus, the chosen companies offer a comprehensive exploration of social media sentiment, investor attention, and stock price movements across different industries.

Juridical and moral concepts

The utilization of tweets from personal accounts raises some legal and ethical considerations. Twitter grants users the option to designate their accounts as private or public. The snscrape tool is specifically designed to retrieve tweets from public accounts on Twitter, thereby making them accessible and readable to the public. It can be assumed that the account owners are cognizant of the public nature of their tweets and have not shared sensitive information through those tweets. Moreover, Google Trends provides data that is aggregated and anonymized to safeguard individual privacy. Consequently, the data is presented in a manner that prevents the identification of specific users or their search queries.

Programming language & Data collection

To perform the analysis three datasets for each stock are needed. A dataset containing tweets that include either the company name or ticker, Google trends score extracted using the company ticker, and a dataset containing the adjusted closing price for the stocks. The rationale for employing both the company names and ticker symbols is underpinned by the empirical results presented by (Batra & Daudpota, 2018; Joseph et al., 2011; Preis et al., 2013; Renault, 2020; Sprenger et

al., 2014). The data collection and pre-processing are done in Python, whilst the statistical analysis is done in R. Comprehensive lists of the utilized packages are presented in Table 5 and Table 6.

Tweets

We utilize the snscrape tool, a Python-based web scraping tool, to collect Twitter data, including tweets and Twitter account information, circumventing Twitter's API limitations (JustAnotherArchivist, 2018/2023). With snscrape, we can collect data from the past five years, specifically from January 1, 2018, to February 1, 2023, with the last month as our out-of-sample period. All tweets collected include either the ticker or the company name to ensure relevance to our study on stocks. Additionally, we have set a limitation on the minimum number of followers at 100, to ensure that the accounts we collect data from have some level of influence, while still obtaining a sufficient amount of data for our analysis in line with the findings of (Sul et al., 2017b).

Cleaning Tweets

A rigorous process was employed to clean the Twitter data. This process involved a series of text pre-processing tasks. The tweets underwent a comprehensive cleaning process that included the removal of URLs, user mentions, hashtags, punctuation, numbers, and extra whitespaces. Furthermore, the tweets were converted to lowercase and tokenized to facilitate subsequent analysis (Bird, 2009).

To refine the dataset, stopwords were removed using the Natural Language Toolkit (NLTK) library (Bird, 2009). This step was essential in eliminating common words that do not contribute to the overall sentiment of the text. Moreover, a language detection library, Langdetect (Danilák, 2014/2023) was employed to ensure that the analysis focused exclusively on English-language content, thereby increasing the relevance and accuracy of the findings. Emojis, which have become increasingly prevalent in online communication (Kralj Novak et al., 2015), were also addressed by converting them to textual representations using the Emoji library (Kim, 2014/2023). The inclusion of emojis is consistent with the findings of (Renault, 2020) which found that adding emojis significantly improved sentiment classification performance.

The cleaning process applied in this study aimed to standardize the data, remove irrelevant elements, and ensure the accuracy and reliability of the sentiment analysis.

Sentiment analysis

In our sentiment analysis of the cleaned data, we adopted a hybrid approach blending both lexicon-based methods and machine learning techniques. This combination leverages the strengths of the well-known Loughran-McDonald dictionary and the innovative Hugging Face sentiment analysis pipeline, a synergy proven to enhance the precision of sentiment classifications (Kolchyna et al., 2015).

The Loughran-McDonald dictionary, a lexicon specially crafted for financial and business scenarios (Loughran & Mcdonald, 2011) houses an extensive assortment of words tagged with positive and negative sentiment scores, thereby enabling effective identification of sentiment-bearing words within our textual data. As a lexicon-based approach, its implementation is straightforward and its computational requirements are relatively light, making it an ideal tool for handling large datasets. Its application allowed us to accurately capture sentiment nuances associated with the four companies and their stock performance.

To supplement the Loughran-McDonald dictionary, we tapped into the Hugging Face sentiment analysis pipeline, which utilizes the cutting-edge DistilBERT model (Sanh et al., 2020). This machine learning model offers an advanced understanding of language, taking into account intricate language patterns, sarcasm, and context-dependent sentiment. Unlike the more straightforward lexicon-based methods, the Hugging Face pipeline offers a nuanced sentiment analysis that goes beyond simple keyword identification. This added dimension allowed us to unearth sentiment subtleties potentially overlooked by the Loughran-McDonald dictionary.

Sentiment scores were computed by first tokenizing the cleaned tweets, then calculating the sentiment scores for each method separately. With the Loughran-McDonald dictionary, the difference between the counts of positive and negative tokens was calculated and normalized by the total number of tokens. For the Hugging Face pipeline, the pre-trained model was applied to each tweet, and the

sentiment score was extracted. This blending of two methodologies allowed us to capitalize on the unique strengths of both approaches, enhancing the overall accuracy and dependability of our sentiment analysis.

Google Trends

Given that Google Trends only provides daily data for time frames shorter than 90 days, and weekly data for longer time frames, we encountered limitations in obtaining daily SVI directly for our five-year study period. Therefore, we utilized an approach to create a daily SVI that closely matches the weekly SVI provided by Google Trends. The rationale behind this approach is that higher frequency data may capture nuances and effects that lower frequency data cannot. To achieve this, we utilized the Pytrends library in Python to interact with the Google Trends API. We defined our search term and collected daily data in 90-day intervals throughout our set date range. Subsequently, we resampled the daily data to a daily frequency and filled any missing values using a forward fill method.

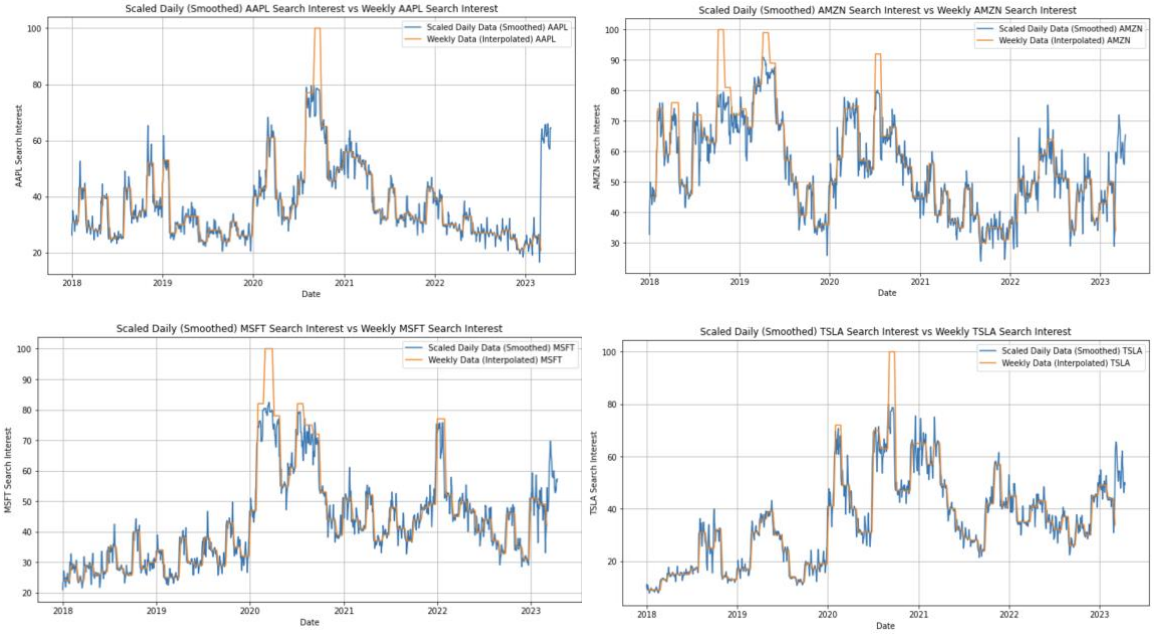
To maintain consistency with the weekly data acquired from Google Trends, we employed a scaling technique. This involved calculating scaling factors by dividing the weekly data by the average of the daily data for that week. Subsequently, these scaling factors were used to rescale the daily data by multiplying each data point with its corresponding scaling factor. To ensure that the rescaled daily data remained within the permissible range of the Google Trends index (which has an upper limit of 100), we implemented a maximum value cap of 100.

Upon this, we introduced a smoothing technique to further refine our daily data. This technique comprised calculating a 7-day moving average, which involved establishing a rolling window of 7 days centred on each date and calculating the average for these windows. The outcome was a smoothed version of our daily data, offering a clearer visualization of the overarching trends and patterns.

In addition, we obtained the weekly data for the entire time period of interest using the Pytrends API and resampled it to a weekly frequency. Any missing values in the weekly data were interpolated to ensure completeness. Plots were

generated to visually inspect the trends and patterns in the data, including the smoothed scaled daily data and the interpolated weekly data.

Figure 1 Constructed daily Google Trends SVI vs Weekly google trends SVI



Integration of Adjusted Closing Prices with Sentiment and SVI Data

The adjusted closing prices employed in our analysis were directly obtained from Yahoo Finance and imported into the R. These prices are widely recognized and commonly utilized in financial analysis because they encompass adjustments for various corporate actions, such as stock splits, dividends, and other factors that may affect historical stock prices. By incorporating adjusted closing prices, our objective was to capture the actual underlying price movements of the stocks, while minimizing the impact of corporate actions on our analysis. This methodology ensured the integrity and precision of our data, allowing us to conduct rigorous analysis and draw reliable conclusions in our research.

Sentiment scores and SVI were synchronized with the adjusted closing prices of the stocks by implementing an 'inner join' strategy during the merging process, retaining only the common dates across all datasets. This strategy automatically aligned the sentiment and SVI data with the adjusted closing price, consequently excluding the weekend data.

Results

To capture the relevant information for the subsequent trading days' adjusted closing price, we consider tweets posted until midnight. This approach allows us to observe the impact over a full trading day after the tweet, ensuring a comprehensive understanding of its effects. It is important to note that any tweets posted while the market is still open may incorporate some of their effect into the price on the same day. If this occurs, it potentially attenuates our results since part of the reaction may have already been incorporated into the prices before our measurement days. Additionally, the SVI is constructed on a day-by-day basis and examined on the adjusted closing price the following trading days. However, we find this to have a potential very limited impact on the results.

Forecasting results

To evaluate the forecasting performance of the VAR models and gain deeper insights into the contribution of different variables, we conducted a comprehensive analysis. In this evaluation, we compared the MSE, RMSE, and MAE of the VAR models to that of a benchmark model, the RW. Additionally, we introduced an autoregressive regression (AR) model that solely incorporates the lags of the stock as a baseline. This AR model allows us to examine the impact of including additional factors, such as Twitter sentiment and Google Trends search volume index, in the forecasting process. The results of this evaluation can be found in Table 6.

Across all stocks and the three VAR model setups, we consistently observed superior performance by the VAR models in comparison to the RW model. This pattern was evident across all accuracy metrics. In every instance, the VAR models exhibited lower values compared to the RW model, underscoring the VAR models consistent outperformance of the RW in terms of accuracy.

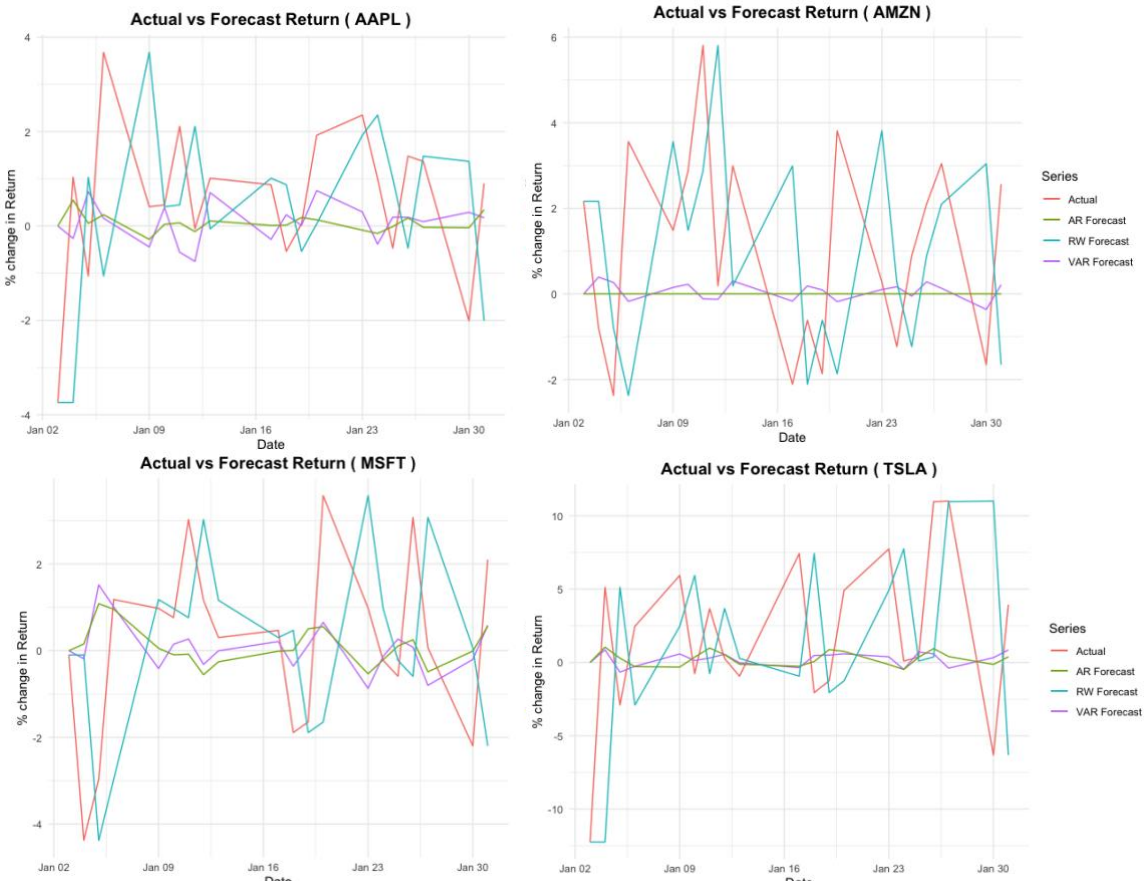
When comparing the performance of the VAR model against the AR model, divergent trends emerge for different stocks. Specifically, the VAR model showcased inferior forecasting accuracy for both Apple and Tesla, in contrast to the AR model. The VAR models consistently exhibited higher accuracy metrics compared to the AR model suggesting that the inclusion of Twitter sentiment and

SVI variables diminishes the model's performance universally for these stocks. Conversely, the VAR modes consistently outperformed the AR model for both Microsoft and Amazon, exhibiting lower accuracy metrics. These consistent findings across both stocks imply that incorporating Twitter sentiment and SVI variables enhances the forecasting accuracy of the VAR model, positioning it as a more effective approach than the AR model for Microsoft and Amazon. Thus, the performance of the VAR model compared to the AR model is contingent upon the specific stocks being analysed, indicating the importance of considering individual stock dynamics when selecting the appropriate forecasting model.

However, upon evaluating the diverse setups within the VAR model for each of the four stocks, we observe some discernible variations in forecasting accuracy. Setup (1) consistently exhibits higher accuracy metrics when compared to both setup (2) and (3). This consistent pattern suggests that the inclusion of both Twitter sentiment and SVI variables, simultaneously, may not improve, but instead might impede the model's performance for these specific stocks.

The visual representation of the VAR setup (1), is clearly outlined in Figure 2 Forecasting performance. Setup (1), alongside actual values, the RW model, and the AR model for all stocks. As noted in tracking the actual values for Apple and Tesla, the AR model manifested superior performance, whereas the VAR model excelled for Amazon and Microsoft. However, it should be further noted that the AR model captured the least volatility in the stock price movements. In comparison, our VAR model demonstrated a marginally superior capability to capture volatility than the AR model. On the other hand, the RW model performed the worst in terms of overall forecasting accuracy but demonstrated the highest capability in capturing volatility. These results indicate that while the AR model is superior in terms of capturing the general trend, our VAR model provides a valuable balance between accuracy and volatility capture. This finding highlights the potential of our VAR model as a reliable forecasting tool for stock price movements, offering insights for investors and decision-makers in the financial industry. Forecast values for the models with setup (2) and (3) are presented in appendix Section E.

Figure 2 Forecasting performance. Setup (1)



Comparative Forecasting Performance of VAR, AR, RW and actual values. Setup (1)

Granger causality

The results of the Granger causality tests conducted for Apple, Amazon, Microsoft, and Tesla stocks provide insights into the relationships between variables. The tests aim to determine whether one variable "Granger causes" another, meaning it provides useful information for predicting the future values of the other variable. The tests are conducted at a 5 percent significance level.

For Apple, the tests indicate that neither sentiment nor SVI significantly Granger cause stock returns. Similarly, the stock returns do not significantly Granger cause sentiment or SVI. These findings suggest that there is no significant causal relationship between these variables for Apple.

For Amazon, the results show that sentiment and SVI do not significantly Granger cause the stock returns, indicating a lack of predictive power in relation to future stock price movements. However, the tests reveal that stock returns significantly

Granger causes sentiment in setup (2) with a p-value of 0.024, implying that changes in stock price provide information that can help predict future sentiment patterns.

In the case of Microsoft, the Granger causality tests demonstrate that sentiment and SVI do not significantly Granger cause the stock returns. Similarly, stock returns do not significantly Granger cause sentiment or SVI. These results suggest a lack of significant causal relationships between these variables for Microsoft.

For Tesla, the findings reveal that sentiment and SVI do not significantly Granger cause stock returns. Likewise, the stock returns do not significantly Granger cause sentiment. The exception is in setup (3), where SVI significantly Granger causes the adjusted closing price (p-value of 0.0138), leading to the rejection of the null hypothesis. implying that search volume trends captured by SVI can provide information for predicting future stock price movements.

By adopting a more lenient significance level of 10%, certain relationships between variables begin to emerge. Notably, for Apple, it becomes apparent that Google Trends SVI Granger causes the stock return (Setup 3). In the case of Microsoft, both the stock return and Google Trends SVI exhibit a significant influence on the sentiment score (Setup 1). Furthermore, in relation to Tesla, it is observed that both sentiment and Google Trends SVI Granger cause the stock return (Setup 1), and that the Adj. Closing price Granger cause Sentiment (Setup 2). These findings suggest that by relaxing the significance level, we gain insight into noteworthy causal relationships among these variables for the respective stocks.

Impulse responses

Impulse response function (IRF) analysis provides valuable insights into the intricate relationships among variables within selected stocks. By applying a one standard deviation shock to each variable and observing its effects on the others over a specified period, we gain a comprehensive understanding of their dynamics and interdependencies. This approach enhances our insights into complex interactions, offering a comprehensive view of their interconnectedness.

It is important to note that the response of a variable to its own shock is typically intense and not of primary interest in this analysis. Therefore, we will primarily focus on examining the effects of shocks transmitted between variables, shedding light on how changes in one variable can influence the others within the selected stocks. Nevertheless, to ensure transparency and completeness of our findings, we also report the results of the variables' responses to their own shocks. These results, along with the other impulse responses, are presented in appendix Section G.

Based on the analysis conducted, it was observed that shocks in the adjusted closing prices of stocks resulted in a subsequent change in sentiment and SVI over a period of 3 to 6 days before gradually declining. The effects of this shock on sentiment and SVI were consistent across all stocks, indicating similarity in their responses.

In contrast, shocks in Twitter sentiment had minimal and negligible effects on the stock returns. These small effects were zero after 7 to 10 days, suggesting that Twitter sentiment has a limited impact on stock prices whilst for the SVI a bit more volatility is observed before it goes to zero after 7 to 8 days.

Sending a shock into the SVI causes some more volatility than Twitter sentiment to stock returns peaking at day 3 to 5, before fading towards the baseline levels. To the same shock, the Twitter sentiment responds with small changes over the next 5 days before exhibiting toward the baseline levels.

Variance decomposition

To explore the relationship between investor sentiment and attention, and the future variance of the four selected stocks, we employ the Forecast Error Variance Decomposition (FEVD) measure and track it for 20 days. FEVD allows us to assess the extent to which exogenous shocks to each variable contribute to the forecast error variance of the other variables. By utilizing the FEVD methodology, we aim to determine the role of investor attention and sentiment in explaining the volatility of the selected stocks (Brooks, 2019).

The results, as presented in the appendix Section H, is very similar across the stocks. Not surprisingly, the majority of the variation in stock prices is explained by the stocks themselves. The lagged values of each stock consistently account for a significant portion of the forecast error variance, reflecting the intrinsic characteristics and market dynamics of the stocks.

When considering the external factors, their combined influence on stock price variance is relatively limited. Collectively, these factors explain no more than 2 percent of the variance in stock prices. Notably, investor attention tends to have slightly higher explanatory power compared to sentiment, albeit still within the limited range.

Moreover, the explanatory power of both investor attention and sentiment tends to increase gradually up to approximately day 10 before stabilizing. This observation suggests that these factors have a modest but time-limited impact on stock price variance.

Robustness test

The reverse ordering test was conducted to examine the impact of variable ordering in the VAR model. The objective was to determine whether the arrangement of variables has any notable effect on the results. The analysis revealed that the ordering of variables had very little impact, and the outcomes remained consistent with the previous findings. This is not surprising as the correlation between variables was found to be low. The low correlation indicates that the relationships between the variables are not strongly influenced by their relative positions in the model. Consequently, the reverse ordering test demonstrates the robustness of the results, confirming that the conclusions drawn hold regardless of the variable ordering.

Discussion and Conclusion

In this study, we conducted an extensive analysis to explore the relationship between stock prices, Twitter sentiment, and investor attention. We collected and cleaned a dataset comprising 0.55 million tweets and calculated a sentiment score for each tweet. Additionally, we constructed a daily Google Trends search volume index (SVI) to capture investor attention. With these data in hand, we aimed to investigate the predictive capabilities and causal relationships among these variables using several Vector Autoregression (VAR) models.

The VAR models' consistent outperformance over the RW models reveals that stock prices do not strictly follow a random walk and that the market does not instantaneously incorporate all available information, hinting at possible inefficiencies. The fact that AR outperformed VAR for Apple and Tesla suggests that the incorporation of sentiment and SVI data did not add significant predictive power beyond historical prices, aligning more with the EMH. However, the superior performance of VAR for Amazon and Microsoft, when sentiment and attention were included, indicates that these additional factors do capture valuable information not entirely reflected in historical prices alone. This supports BFT, which posits that psychological factors and investor sentiment can influence stock prices, suggesting that investor behaviour may contribute to market inefficiencies.

Across all stocks, the Granger causality results generally revealed weak causality links, with Amazon and Tesla being notable exceptions. In Amazon's case, there was a significant Granger causality from the adjusted closing price to sentiment. This suggests a potential one-way causal relationship between fluctuations in the stock price and the prevailing public sentiment. For Tesla, we discovered a significant Granger causality from investor attention to stock return. This implies that search volume trends could serve as a predictive indicator for Tesla's stock price. Interestingly, we observed similar results for Apple, albeit at a lower significance level of 10%. The Theory of Information Cascade lends support to these findings, advocating that investors often imitate perceived trends, which, in turn, further impact the stock return (Hirshleifer & Hong Teoh, 2003). Moreover, sentiment and SVI were found to Granger cause the Tesla stock return at a 10% significance level. On balance, the results related to Tesla suggest that sentiment

plays a relatively subdued role in the model as compared to SVI. Collectively, these findings underscore potential market inefficiencies, thereby questioning the tenets of the Efficient Market Hypothesis.

The variance decomposition analysis showed limited explanatory power of Twitter sentiment and SVI. The results consistently showed that the majority of the variation in stock prices was explained by the stocks themselves, with lagged values accounting for a significant portion of the forecast error variance. The combined influence of Twitter sentiment and SVI on stock price variance was relatively limited, collectively explaining no more than 2 percent of the variance. Investor attention tended to have slightly higher explanatory power compared to sentiment, albeit still within a limited range. This evidence is consistent with the EMH's assumption of market efficiency.

The impulse response analysis shed light on the dynamics and interdependencies among variables. A shock in the adjusted closing prices resulted in subsequent changes in sentiment and SVI, albeit relatively small. A shock in sentiment or SVI had a limited effect on the adjusted closing price. These findings further support the limited influence of Twitter sentiment and SVI on stock price movements in line with the efficient market theory.

The findings challenge the efficient market hypothesis, despite limited support for sentiment and attention as predictors of stock returns. Specific significant findings suggest market inefficiency and potential validity of the behavioural alternative hypothesis. Ongoing research is needed for this complex interplay and tailored, company-specific models. Continual exploration of these relationships using emerging data sources and methodologies is crucial for understanding and predicting market fluctuations.

Limitations and Suggestions for Further Research

Our results suggest modest support for incorporating Twitter sentiment and investor attention as predictors for stock returns. This finding prompts us to critically assess the limitations of our study and highlights the need for further investigation into alternative factors or methodologies to enhance stock price prediction.

Firstly, our focus on Twitter limits the applicability of our results to this specific platform. Twitter users may not fully represent the general investor population, as social media users tend to be younger and potentially less experienced in investment decision-making. Furthermore, the prevalence of bots, fake accounts, and coordinated disinformation campaigns has the potential to skew the data. Lastly, the rapid and often reactionary nature of tweets can create noise and volatility that is not indicative of long-term financial trends. Hence, the sentiment scores we derived may not wholly represent the true sentiment of the investors.

Secondly, the use of Google Trends as a proxy for investor attention also poses limitations. While Google Trends provides a good estimation of general interest, it may not perfectly capture the attention of investors due to its relative data presentation, absence of user intent or background context, and susceptibility to short-term search volume spikes. It does not provide actual search counts but shows the popularity of search terms relative to total volume, complicating precise measurements.

Lastly, we acknowledge the limitations of our sentiment analysis approach, which combines the Loughran-McDonald dictionary with the Hugging Face pipeline. While effective in our study context, the tool might overlook certain sentiment-bearing words due to the dictionary's domain specificity, misconstrue context-dependent phrases or sarcasm due to DistilBERT's limitations, incur computational expenses for larger datasets, or face potential bias when averaging sentiment scores from both methods. Moreover, the quality of text pre-processing profoundly impacts the results, and while our tool capably captures basic sentiment, it may not fully grasp the complexity and nuances of human emotions,

indicating a need for future research to leverage advanced natural language processing techniques for improved accuracy.

Given the limitations of our study, future research should aim to broaden the data sources beyond Twitter, encompassing various media platforms for a more comprehensive understanding of sentiment influences on stock prices. Incorporating macroeconomic variables or market volatility indices may enhance predictive capabilities, while extending the scope of analysis beyond four large-cap US companies can reveal diverse patterns in different enterprise sizes, sectors, or regions. It is worth noting that expanding geographical scope beyond the US and China will necessitate financial lexicons in other languages, demanding extensive machine learning or creation of new lexicons. Lastly, the adoption of advanced machine learning methodologies such as Neural Networks, Support Vector Machines, and Naive Bayesian algorithms can enrich sentiment classification, facilitating a more complex understanding of investor sentiments.

By addressing these limitations and pursuing these avenues of future research, a more comprehensive understanding of the power of sentiment and investor attention on stock market dynamics can be achieved, leading to improved decision-making for investors in various contexts.

Bibliography

Ahmad, K., Han, J., Hutson, E., Kearney, C., & Liu, S. (2016). Media-expressed negative tone and firm-level stock returns. *Journal of Corporate Finance*, *37*, 152–172. https://doi.org/10.1016/j.jcorpfin.2015.12.014

Antweiler, W., & Frank, M. Z. (2004). Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards. *The Journal of Finance*, *59*(3), 1259–1294. https://doi.org/10.1111/j.1540-6261.2004.00662.x

Baker, M., & Wurgler, J. (2006). Investor Sentiment and the Cross-Section of Stock Returns. *The Journal of Finance*, *61*(4), 1645–1680. https://doi.org/10.1111/j.1540-6261.2006.00885.x

Batra, R., & Daudpota, S. M. (2018). Integrating StockTwits with sentiment analysis for better prediction of stock price movement. *2018 International Conference on Computing, Mathematics and Engineering Technologies (ICoMET)*, 1–5. https://doi.org/10.1109/ICOMET.2018.8346382

Bird, S. (2009). *Natural Language Processing with Python*.

Bollen, J., Mao, H., & Zeng, X.-J. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, *2*(1), 1–8. https://doi.org/10.1016/j.jocs.2010.12.007

Brooks, C. (2019). *Inroductory Econometrics for Finance* (Vol. 1–4th Edition). Cambridge University Press. www.cambridge.org/brooks4

Chen, H.-Y., & Lo, T.-C. (2019). Online search activities and investor attention on financial markets. *Asia Pacific Management Review*, *24*(1), 21–26. https://doi.org/10.1016/j.apmrv.2018.11.001

Da, Z., Engelberg, J., & Gao, P. (2011). In Search of Attention. *The Journal of Finance*, *66*(5), 1461–1499. https://doi.org/10.1111/j.1540-6261.2011.01679.x

Danilák, M. (2023). *Langdetect* [Python]. https://github.com/Mimino666/langdetect (Original work published 2014)

Dsouza, J. J., & Mallikarjunappa, T. (2015). Do the Stock Market Indices Follow Random Walk? *Asia-Pacific Journal of Management Research and Innovation*, *11*(4), 251–273. https://doi.org/10.1177/2319510X15602969

Edmans, A., Garcia, D., & Norli, Ø. (2007). Sports Sentiment and Stock Returns. *The Journal of Finance*. https://doi.org/10.1111/j.1540-6261.2007.01262.x

Fama, E. F. (1965). The Behavior of Stock-Market Prices. *The Journal of Business*, *38*(1), 34–105. https://doi.org/10.1086/294743

Hirshleifer, D., & Hong Teoh, S. (2003). Herd Behaviour and Cascading in Capital Markets: A Review and Synthesis. *European Financial Management*, *9*(1), 25–66. https://doi.org/10.1111/1468-036X.00207

Huang, M. Y., Rojas, R. R., & Convery, P. D. (2020). Forecasting stock market movements using Google Trend searches. *Empirical Economics*, *59*(6), 2821–2839. https://doi.org/10.1007/s00181-019-01725-1

Joseph, K., Babajide Wintoki, M., & Zhang, Z. (2011). Forecasting abnormal stock returns and trading volume using investor sentiment: Evidence from online search. *International Journal of Forecasting*, *27*(4), 1116–1127. https://doi.org/10.1016/j.ijforecast.2010.11.001

JustAnotherArchivist. (2023). *Snscrape* [Python]. https://github.com/JustAnotherArchivist/snscrape (Original work published 2018)

Kim, T. (2023). *Emoji* [Python]. https://github.com/carpedm20/emoji (Original work published 2014)

Kolchyna, O., Souza, T. T. P., Treleaven, P., & Aste, T. (2015). *Twitter Sentiment Analysis: Lexicon Method, Machine Learning Method and Their Combination* (arXiv:1507.00955). arXiv. http://arxiv.org/abs/1507.00955

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of Emojis. *PLOS ONE*, *10*(12), e0144296. https://doi.org/10.1371/journal.pone.0144296

Lachanski, M., & Pav, S. (2017). Shy of the character limit: "Twitter mood predicts the stock market" revisited. *Econ Journal Watch*, *14*, 302–345.

Li, F. (2010). The Information Content of Forward-Looking Statements in Corporate Filings-A Naïve Bayesian Machine Learning Approach: The information content of corporate filings. *Journal of Accounting Research*, *48*(5), 1049–1102. https://doi.org/10.1111/j.1475-679X.2010.00382.x

Loughran, T., & Mcdonald, B. (2011). When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *The Journal of Finance*, *66*(1), 35–65. https://doi.org/10.1111/j.1540-6261.2010.01625.x

Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-27752-1

Mao, Y., Wei, W., Wang, B., & Liu, B. (2012). Correlating S&P 500 stocks with Twitter data. *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, 69–72. https://doi.org/10.1145/2392622.2392634

Nau, B. (2014). *Notes on the random walk model*.

Oh, C., & Sheng, O. (2011). *Investigating Predictive Power of Stock Micro Blog Sentiment in Forecasting Future Stock Price Directional Movement. 4*.

Preis, T., Moat, H. S., & Stanley, H. E. (2013). Quantifying Trading Behavior in Financial Markets Using Google Trends. *Scientific Reports*, *3*(1), Article 1. https://doi.org/10.1038/srep01684

Preis, T., Reith, D., & Stanley, H. E. (2010). Complex dynamics of our economic life on different scales: Insights from search engine query data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *368*(1933), 5707–5719. https://doi.org/10.1098/rsta.2010.0284

Price, S. M., Doran, J. S., Peterson, D. R., & Bliss, B. A. (2012). Earnings conference calls and stock returns: The incremental informativeness of textual tone. *Journal of Banking & Finance*, *36*(4), 992–1011. https://doi.org/10.1016/j.jbankfin.2011.10.013

Ranco, G., Aleksovski, D., Caldarelli, G., Grčar, M., & Mozetič, I. (2015). The Effects of Twitter Sentiment on Stock Price Returns. *PLOS ONE*, *10*(9), e0138441. https://doi.org/10.1371/journal.pone.0138441

Renault, T. (2020). Sentiment analysis and machine learning in finance: A comparison of methods and models on one million messages. *Digital Finance*, *2*(1), 1–13. https://doi.org/10.1007/s42521-019-00014-x

Shiller, R. J. (2003). From Efficient Markets Theory to Behavioral Finance. *Journal of Economic Perspectives*, *17*(1), 83–104. https://doi.org/10.1257/089533003321164967

Sprenger, T. O., Tumasjan, A., Sandner, P. G., & Welpe, I. M. (2014). Tweets and Trades: The Information Content of Stock Microblogs: Tweets and Trades. *European Financial Management*, *20*(5), 926–957. https://doi.org/10.1111/j.1468-036X.2013.12007.x

Statman, M. (2010). What Investors *Really* Want. *Financial Analysts Journal*, *66*(2), 8–10. https://doi.org/10.2469/faj.v66.n2.5

Sul, H. K., Dennis, A. R., & Yuan, L. I. (2017a). Trading on Twitter: Using Social Media Sentiment to Predict Stock Returns: Trading on Twitter. *Decision Sciences*, *48*(3), 454–488. https://doi.org/10.1111/deci.12229

Sul, H. K., Dennis, A. R., & Yuan, L. I. (2017b). Trading on Twitter: Using Social Media Sentiment to Predict Stock Returns: Trading on Twitter. *Decision Sciences*, *48*(3), 454–488. https://doi.org/10.1111/deci.12229

Tetlock, P. C. (2007). Giving Content to Investor Sentiment: The Role of Media in the Stock Market. *The Journal of Finance*, *62*(3), 1139–1168. https://doi.org/10.1111/j.1540-6261.2007.01232.x

Zeitun, R., Rehman, M. U., Ahmad, N., & Vo, X. V. (2023). The impact of Twitter-based sentiment on US sectoral returns. *The North American Journal of Economics and Finance*, *64*, 101847. https://doi.org/10.1016/j.najef.2022.101847

Zhang, X., Fuehres, H., & Gloor, P. A. (2011). Predicting Stock Market Indicators Through Twitter “I hope it is not as bad as I fear.” *Procedia - Social and Behavioral Sciences*, *26*, 55–62. https://doi.org/10.1016/j.sbspro.2011.10.562

Appendix

Section A – Statistical tests and optimal lag

Table 1 Augmented Dickey Fuller (ADF) unit root test

Variable	P-value	Conclusion
Adj. Close Tesla	< 0,01	Stationary
Sentiment Tesla	< 0,01	Stationary
SVI Tesla	< 0,01	Stationary
Adj. Close Apple	< 0,01	Stationary
Sentiment Apple	< 0,01	Stationary
SVI Apple	< 0,01	Stationary
Adj. Close Microsoft	< 0,01	Stationary
Sentiment Microsoft	< 0,01	Stationary
SVI Microsoft	< 0,01	Stationary
Adj. Close Amazon	< 0,01	Stationary
Sentiment Amazon	< 0,01	Stationary
SVI Amazon	< 0,01	Stationary

Table 1 presents the results of the Augmented Dickey Fuller (ADF) unit root test conducted for all the variables in this study. The ADF test is used to determine the stationarity of time series data. A variable is considered stationary if its p-value is less than 0.01, indicating a rejection of the null hypothesis of a unit root.

Table 2 Optimal lag length

Model	HQIC	SBIC	FPE	AIC
Adj. Close Apple | Sentiment Apple | SVI Apple	**9***	4	14	14
Adj. Close Apple | Sentiment Apple	**9***	4	14	14
Adj. Close Apple | SVI Apple	1	1	9	**9***
AR Apple	9	1	9	**9***
Adj. Close Amazon | Sentiment Amazon | SVI Amazon	**6***	4	8	8
Adj. Close Amazon | Sentiment Amazon	**8***	4	9	9
Adj. Close Amazon | SVI Amazon	5	1	8	**8***
AR Amazon	1	1	1	**1***
Adj. Close Microsoft | Sentiment Microsoft | SVI Microsoft	**6***	4	9	9
Adj. Close Microsoft | Sentiment Microsoft	**9***	6	16	16
Adj. Close Microsoft | SVI Microsoft	1	1	9	**9***
AR Microsoft	9	1	9	**9***
Adj. Close Tesla | Sentiment Tesla | SVI Tesla	**5***	5	14	14
Adj. Close Tesla | Sentiment Tesla	**5***	5	14	14
Adj. Close Tesla | SVI Tesla	5	1	5	**5***
AR Tesla	1	1	1	**1***

Table 2 displays the results of the lag length selection using several information criteria, including the Hannan-Quinn Information Criterion (HQIC), Schwarz Bayesian Information Criterion (SBIC), Final Prediction Error (FPE), and Akaike Information Criterion (AIC). The optimal lag length is determined based on minimizing these criteria. The selected number of lags is indicated by "" for the different models.*

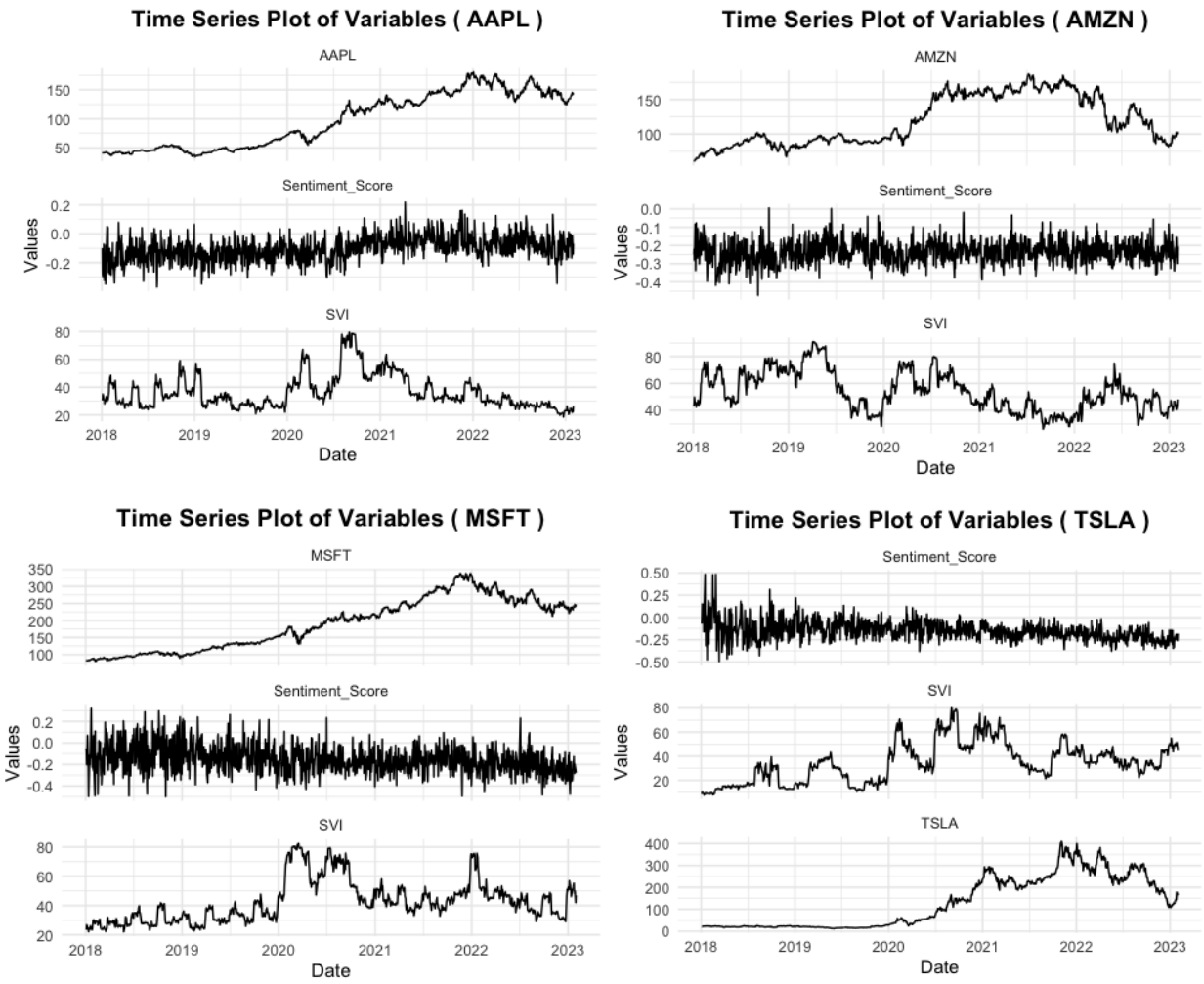
Table 3 Durbin-Watson test for autocorrelation

Model	Test statistic	Conclusion
Adj. Close Tesla | Sentiment Tesla | SVI Tesla	2.0113	No evidence of autocorrelation
Adj. Close Tesla | Sentiment Tesla	2.0244	No evidence of autocorrelation
Adj. Close Tesla | SVI Tesla	1.9871	No evidence of autocorrelation
Adj. Close Apple | Sentiment Apple | SVI Apple	1.9927	No evidence of autocorrelation
Adj. Close Apple | Sentiment Apple	1.9933	No evidence of autocorrelation
Adj. Close Apple | SVI Apple	2.0004	No evidence of autocorrelation
Adj. Close Microsoft | Sentiment Microsoft | SVI Microsoft	2.0184	No evidence of autocorrelation
Adj. Close Microsoft | Sentiment Microsoft	2.0109	No evidence of autocorrelation
Adj. Close Microsoft | SVI Microsoft	1.9968	No evidence of autocorrelation
Adj. Close Amazon | Sentiment Amazon | SVI Amazon	2.0090	No evidence of autocorrelation
Adj. Close Amazon | Sentiment Amazon	2.0047	No evidence of autocorrelation
Adj. Close Amazon | SVI Amazon	1.9945	No evidence of autocorrelation

Table 3 presents the results of the Durbin-Watson test for autocorrelation conducted on the models. A test statistic value close to 2 indicates no evidence of autocorrelation, while values significantly below or above 2 suggest the presence of positive or negative autocorrelation, respectively.

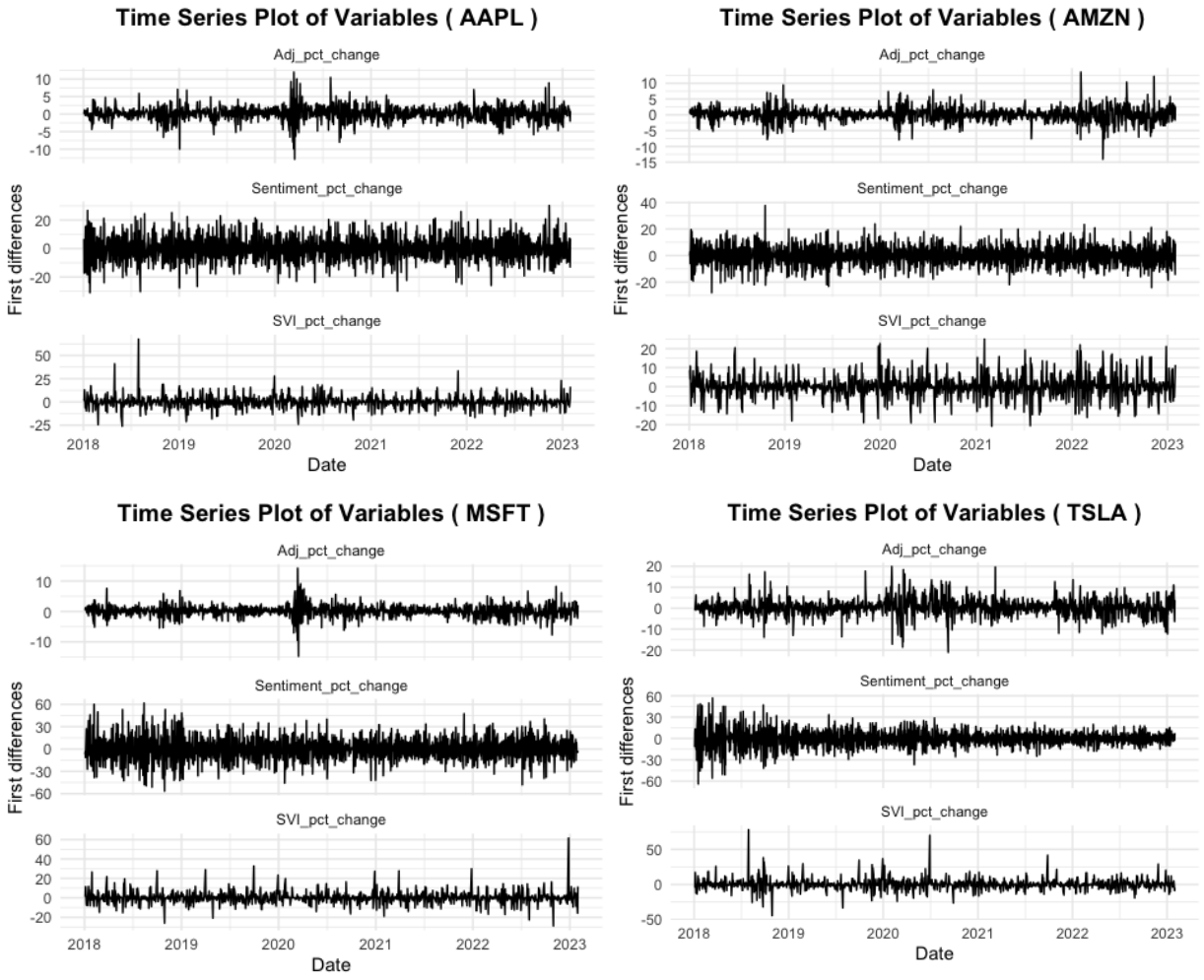
Section B – Time series plot of Variables

Figure 3 Time Series Plot of Variables in actual values



Section C – Time series plot of variables in first differences

Figure 4 Time Series Plot of Variables in first differences



Section D – Forecasting performance

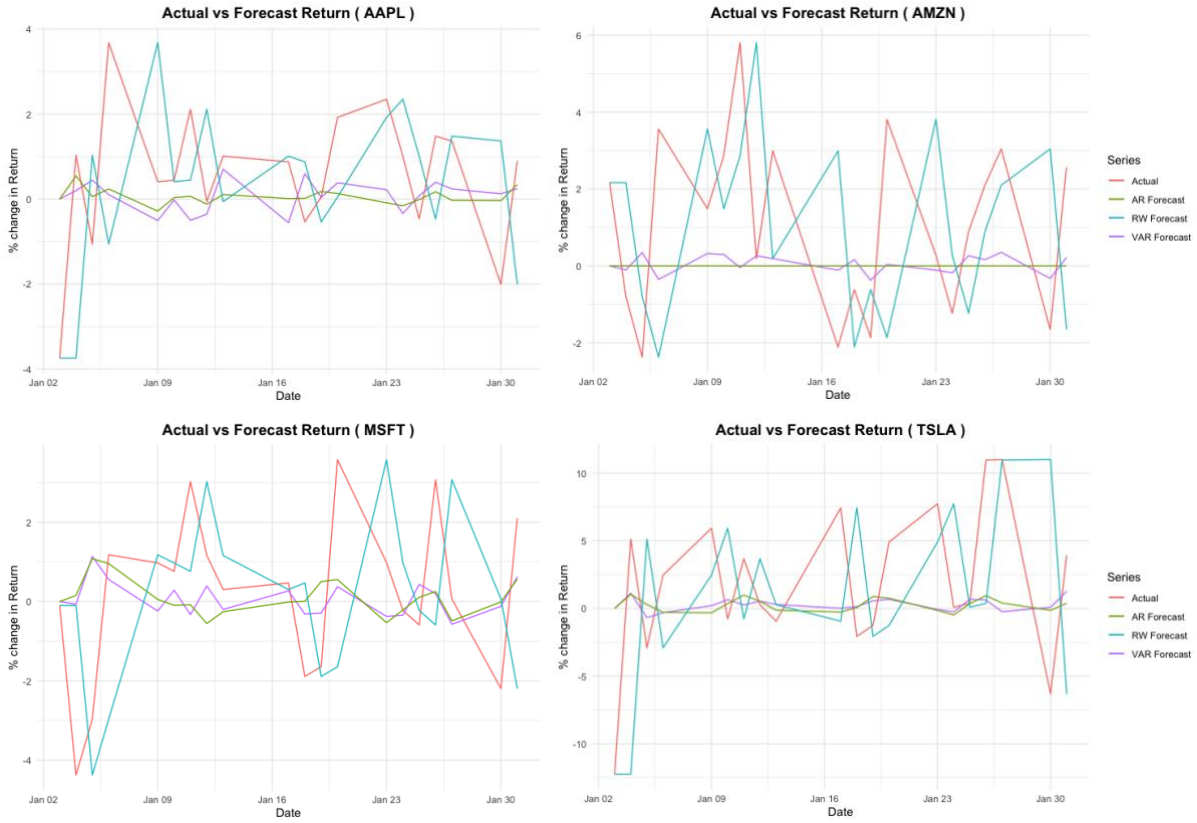
Table 4 Comparative Performance Metrics of the Forecasting Models for Apple, Amazon, Microsoft, and Tesla

	Model	**Variables**	***MSE***	***RMSE***	***MAE***
AAPL	*VAR*	Adj, Close|Sentiment| SVI	2,9323	1,7124	1,3872
	VAR	Adj, Close| Sentiment	2,8764	1,6960	1,3687
	VAR	Adj, Close| SVI	2,8764	1,6960	1,3687
	RW	RW w/drift	5,1338	2,2658	1,7712
	AR	Adj.Close	2,6300	1,6217	1,2765
AMZN	*VAR*	Adj, Close|Sentiment| SVI	6,2093	2,4918	2,1104
	VAR	Adj, Close| Sentiment	5,9435	2,4379	2,0177
	VAR	Adj, Close| SVI	5,9435	2,4379	2,0177
	RW	RW w/drift	11,2116	3,3484	2,8522
	AR	Adj.Close	6,2512	2,5002	2,1202
MSFT	*VAR*	Adj, Close|Sentiment| SVI	4,2029	2,0501	1,6065
	VAR	Adj, Close| Sentiment	4,1091	2,0271	1,5667
	VAR	Adj, Close| SVI	4,1091	2,0271	1,5667
	RW	RW w/drift	6,7028	2,5890	2,0302
	AR	Adj.Close	4,3570	2,0873	1,6457
TSLA	*VAR*	Adj, Close|Sentiment| SVI	32,8331	5,7300	4,4225
	VAR	Adj, Close| Sentiment	32,5098	5,7017	4,4039
	VAR	Adj, Close| SVI	32,5098	5,7017	4,4039
	RW	RW w/drift	63,3531	7,9595	6,1883
	AR	Adj.Close	32,1939	5,6740	4,4162

Table 4 provides comparative performance metrics for the forecasting models employed for Apple (AAPL), Amazon (AMZN), Microsoft (MSFT), and Tesla (TSLA). The metrics evaluated include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE)

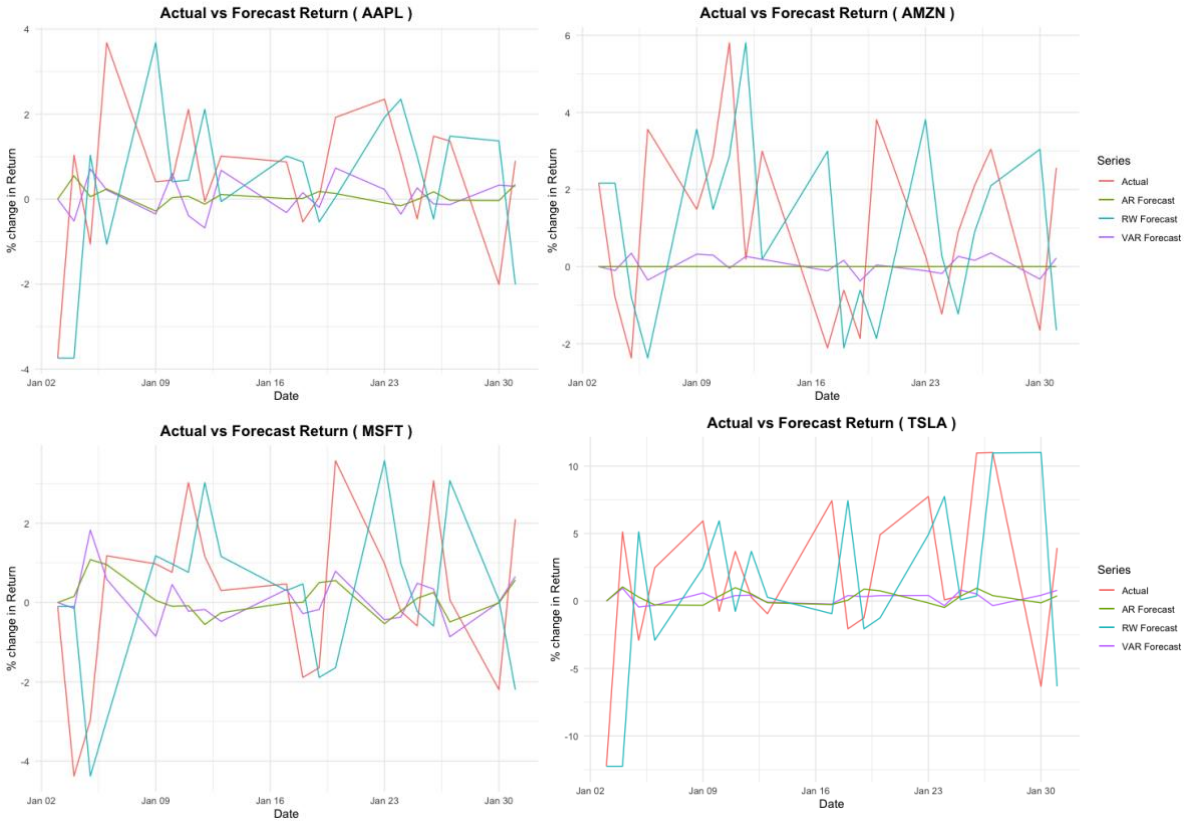
Section E – Forecasting plots

Figure 5 Setup (2): Adj. Close | Sentiment



Comparative Forecasting Performance of VAR, AR, RW, and actual values. Setup (2)

Figure 6 Setup (3): Adj. Close | SVI



Comparative Forecasting Performance of VAR, AR, RW, and actual values. Setup (3)

Section F - Granger Causality tests

Apple

Setup (1)

Null Hypothesis	*P-Value*	*Conclusion*
Sentiment & SVI do not Granger-cause Adj. Closing price	0,1266	Fail to reject H0
Adj. Closing price & SVI do not Granger cause Sentiment	0,4464	Fail to reject H0
Adj. Closing price & Sentiment do not Granger cause SVI	0,1432	Fail to reject H0

Setup (2)

Null Hypothesis	*P-Value*	*Conclusion*
Sentiment do not Granger-cause Adj. Closing price	0,3209	Fail to reject H0
Adj. Closing price do not Granger cause Sentiment	0,8394	Fail to reject H0

Setup (3)

Null Hypothesis	*P-Value*	*Conclusion*
SVI do not Granger-cause Adj. Closing price	0,093	Fail to reject H0
Adj. Closing price do not Granger cause SVI	0,8915	Fail to reject H0

Amazon

Setup (1)

Null Hypothesis	*P-Value*	*Conclusion*
Sentiment & SVI do not Granger-cause Adj. Closing price	0,7616	Fail to reject H_0
Adj. Closing price & SVI do not Granger cause Sentiment	0,1060	Fail to reject H_0
Adj. Closing price & Sentiment do not Granger cause SVI	0,4496	Fail to reject H_0

Setup (2)

Null Hypothesis	*P-Value*	*Conclusion*
Sentiment do not Granger-cause Adj. Closing price	0,1451	Fail to reject H_0
Adj. Closing price do not Granger cause Sentiment	0,0240	Reject H_0

Setup (3)

Null Hypothesis	*P-Value*	*Conclusion*
SVI do not Granger-cause Adj. Closing price	0,6671	Fail to reject H_0
Adj. Closing price do not Granger cause SVI	0,3935	Fail to reject H_0

Microsoft

Setup (1)

Null Hypothesis	***P-Value***	***Conclusion***
Sentiment & SVI do not Granger-cause Adj. Closing price	0,1392	Fail to reject H_0
Adj. Closing price & SVI do not Granger cause Sentiment	0,0809	Fail to reject H_0
Adj. Closing price & Sentiment do not Granger cause SVI	0,4089	Fail to reject H_0

Setup (2)

Null Hypothesis	***P-Value***	***Conclusion***
Sentiment do not Granger-cause Adj. Closing price	0,5759	Fail to reject H_0
Adj. Closing price do not Granger cause Sentiment	0,4536	Fail to reject H_0

Setup (3 **)**

Null Hypothesis	***P-Value***	***Conclusion***
SVI do not Granger-cause Adj. Closing price	0,2700	Fail to reject H_0
Adj. Closing price do not Granger cause SVI	0,6180	Fail to reject H_0

Tesla

Setup (1)

Null Hypothesis	***P-Value***	***Conclusion***
Sentiment & SVI do not Granger-cause Adj. Closing price	0,0792	Fail to reject H_0
Adj. Closing price & SVI do not Granger cause Sentiment	0,2361	Fail to reject H_0
Adj. Closing price & Sentiment do not Granger cause SVI	0,6010	Fail to reject H_0

Setup (2)

Null Hypothesis	***P-Value***	***Conclusion***
Sentiment do not Granger-cause Adj. Closing price	0,6898	Fail to reject H_0
Adj. Closing price do not Granger cause Sentiment	0,0587	Fail to reject H_0

Setup (3 **)**

Null Hypothesis	***P-Value***	***Conclusion***
SVI do not Granger-cause Adj. Closing price	0,0138	Reject H_0
Adj. Closing price do not Granger cause SVI	0,5040	Fail to reject H_0

Section G - Impulse responses

The impulse response functions (IRFs) presented in the figures showcase the dynamic interactions among variables within the system. Each line or curve represents the reaction of a particular variable to a single shock or impulse in another variable. Time, measured in days, is depicted on the horizontal axis, while the magnitude of the response is shown on the vertical axis. Positive or negative spikes indicate the direction and size of the response. The shaded regions or confidence intervals surrounding the curves denote the level of uncertainty associated with the estimated response.

Figure 7 Impulse responses - Apple

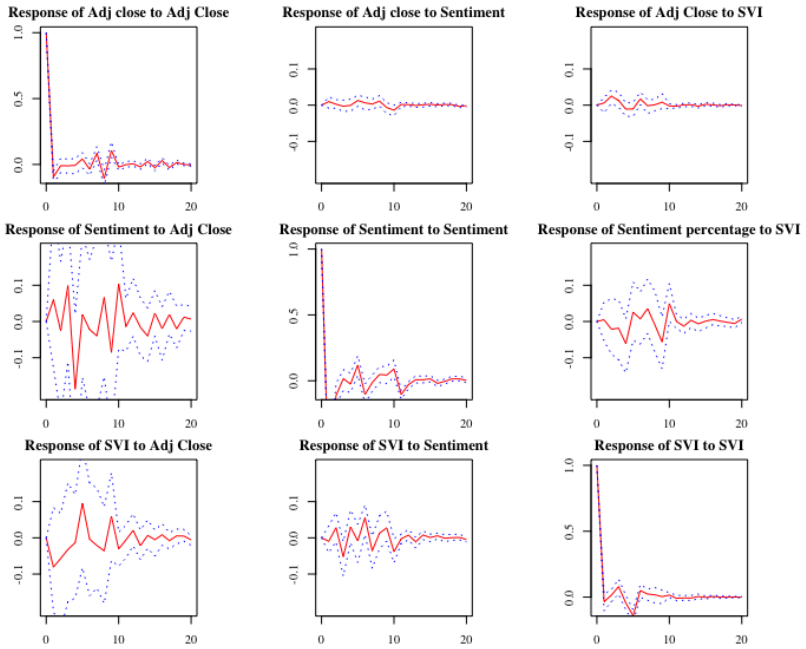


Figure 8 Impulse responses - Amazon

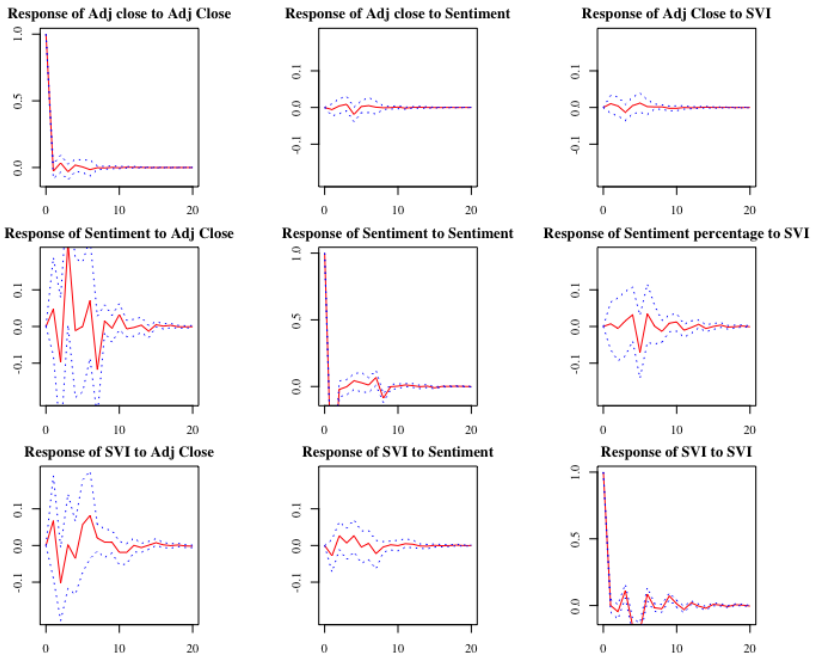


Figure 9 Impulse responses - Microsoft

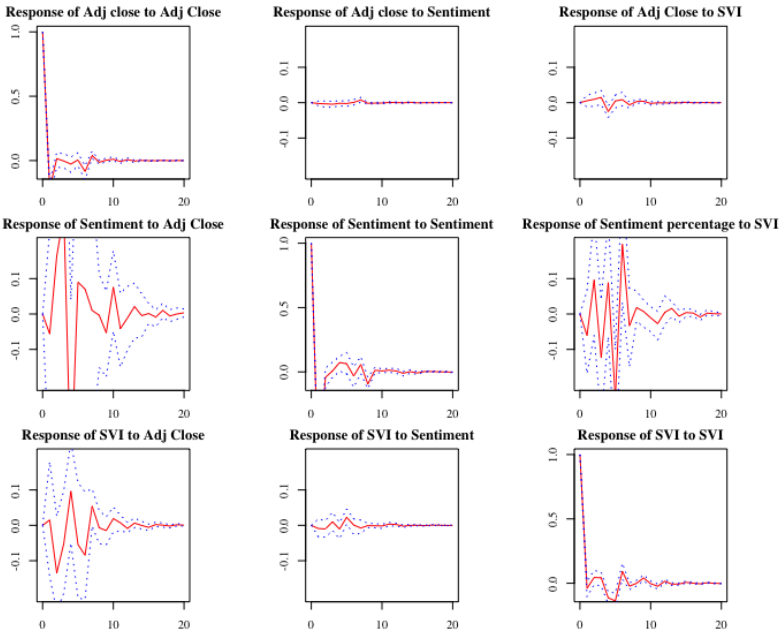
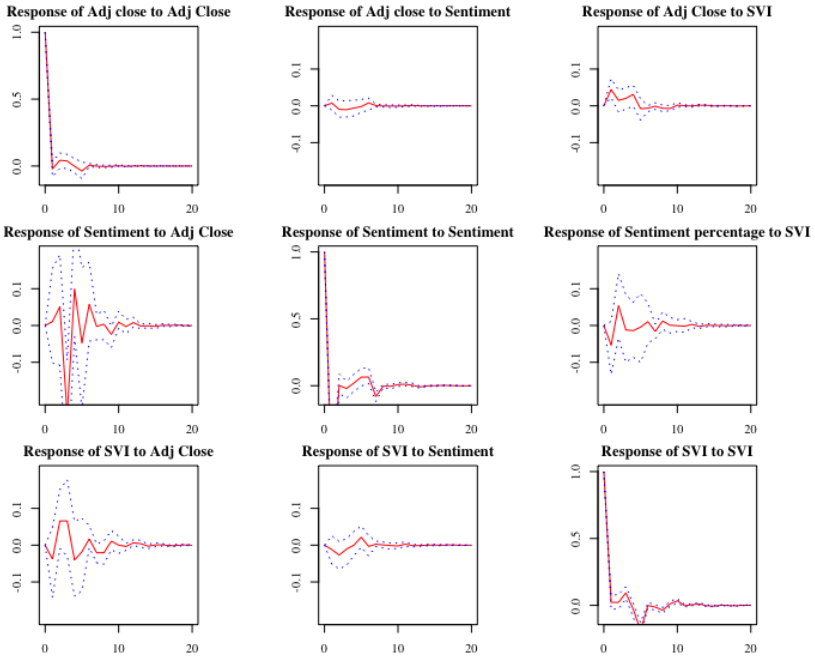


Figure 10 Impulse responses - Tesla



Section H - Variance decomposition

The figures display the variance decomposition results, illustrating the proportion of forecast error variance in each variable attributable to its own shocks (autoregressive component) and the shocks from other variables in the system. The horizontal axis represents time in days, while the vertical axis shows the percentage of variance explained. The variance decomposition plot provides insights into the relative importance and explanatory power of the shocks from different variables, highlighting the main drivers of forecast error variance in the system.

Figure 11 Variance decomposition - Apple

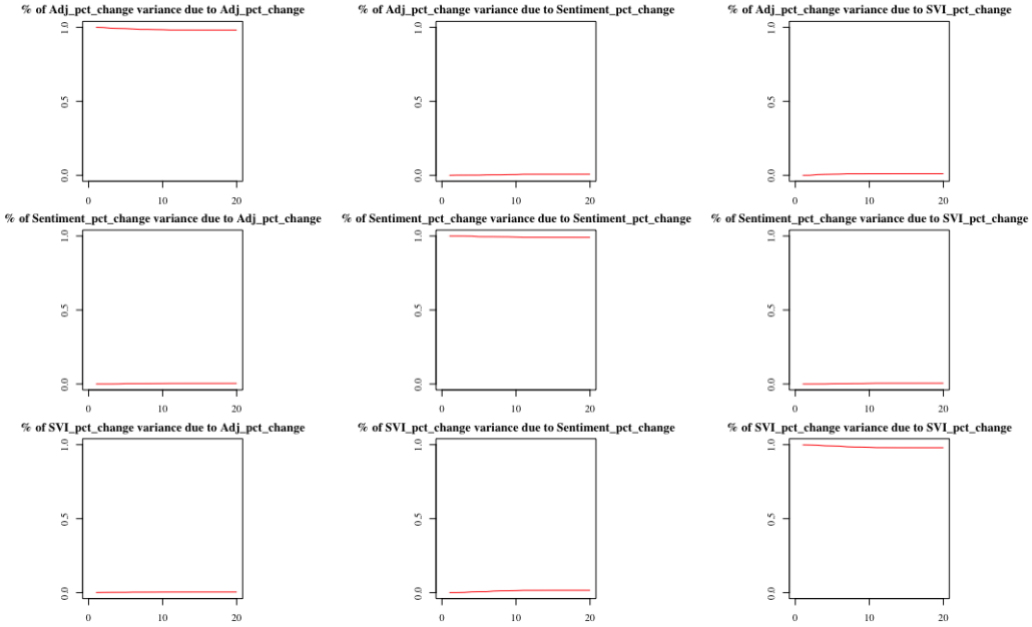


Figure 12 Variance Decomposition - Amazon

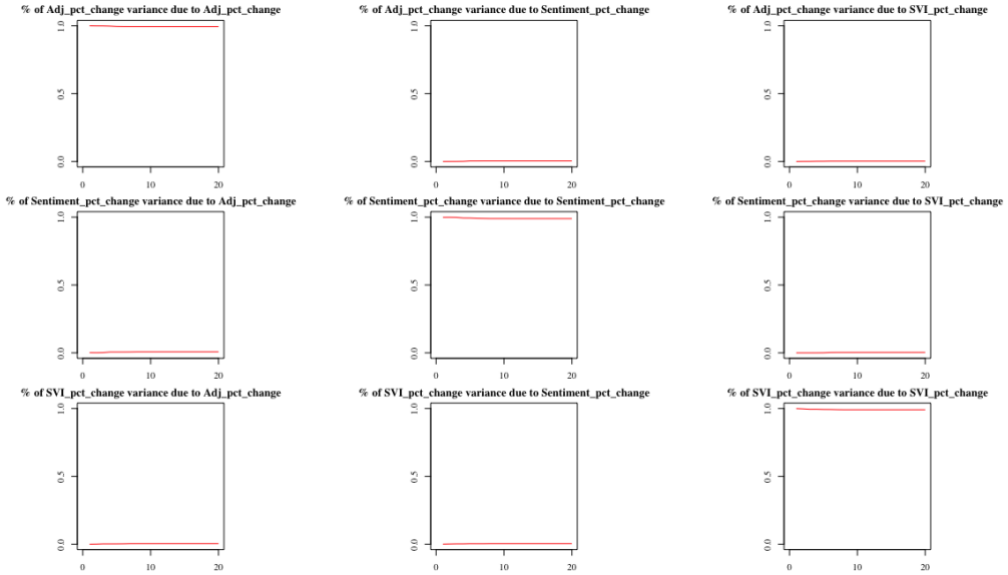


Figure 13 Variance decomposition - Microsoft

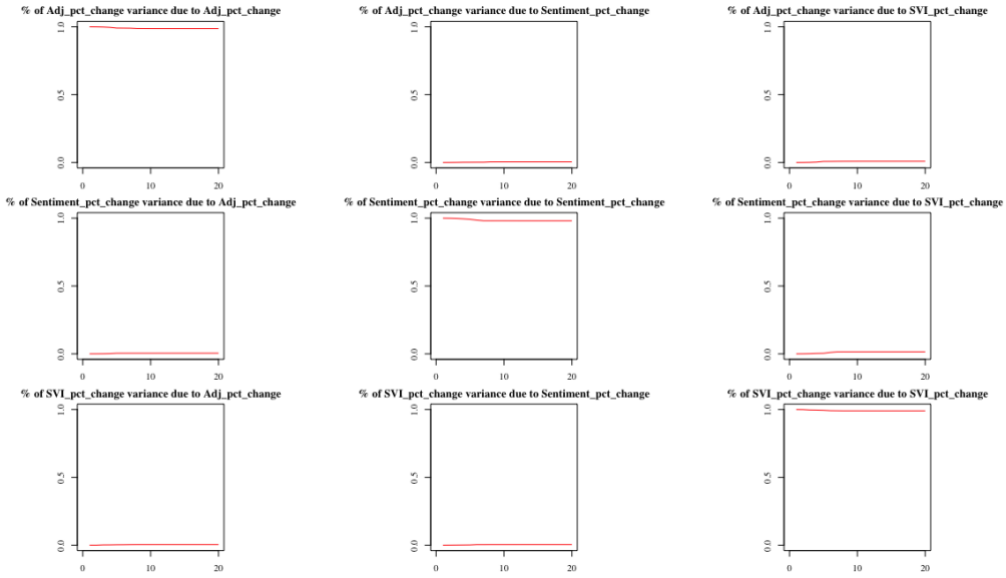
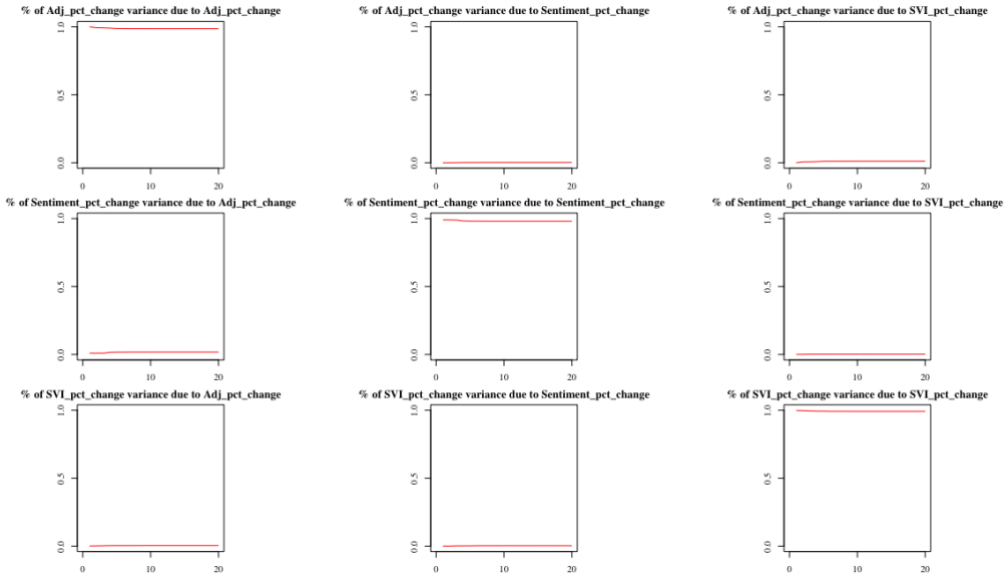


Figure 14 Variance Decomposition – Tesla



Section I – Reverse ordering Robustness tests

The reverse ordering robustness test was conducted to assess the sensitivity of the variance decomposition results to the ordering of variables in the VAR model. The test involved reversing the order of variables in the VAR model and re-estimating the variance decompositions. The purpose was to examine if changing the order of variables would lead to significant differences in the contribution of shocks to forecast error variance. The results of the reverse ordering robustness test indicated that the overall patterns and relative importance of the shocks remained largely consistent, suggesting the robustness of the variance decomposition results. This test provides additional evidence that the identified shocks and their explanatory power are not highly dependent on the specific ordering of variables in the VAR model.

Figure 15 Reverse ordering Robustness test – Apple

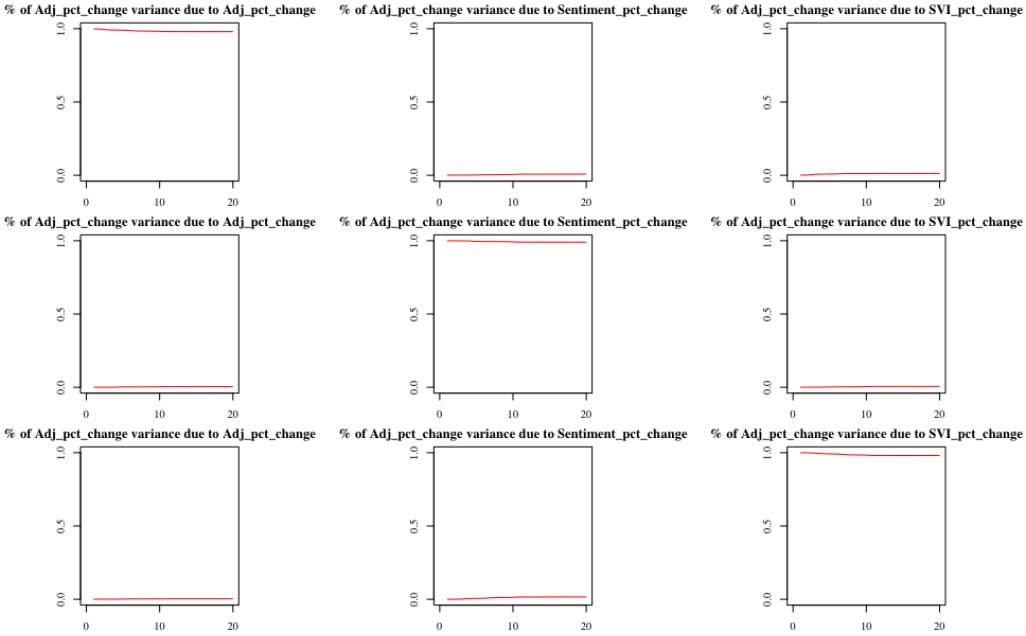


*Figure **16** Reverse ordering Robustness test - Amazon*

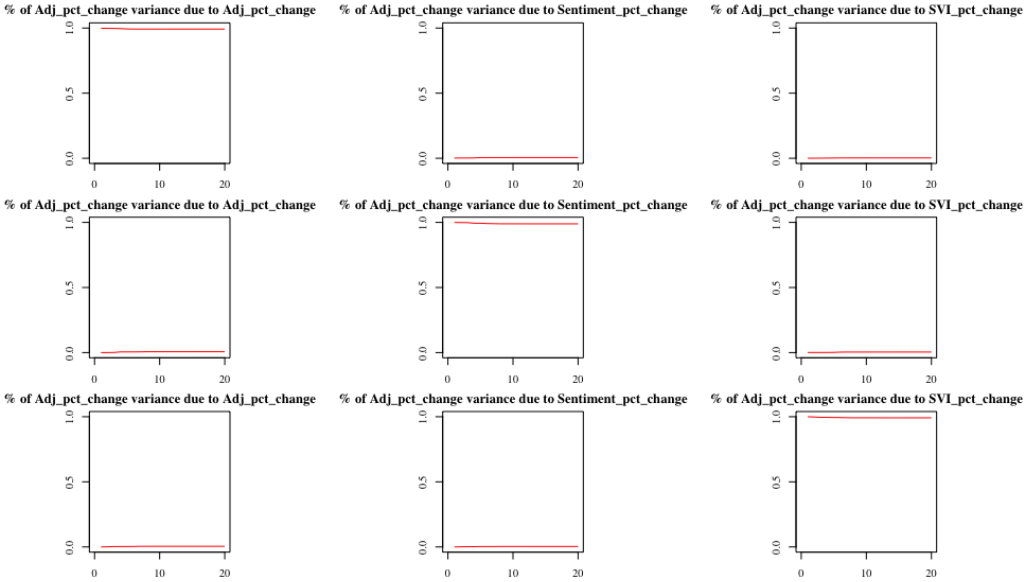


Figure 17 Reverse ordering Robustness test – Microsoft

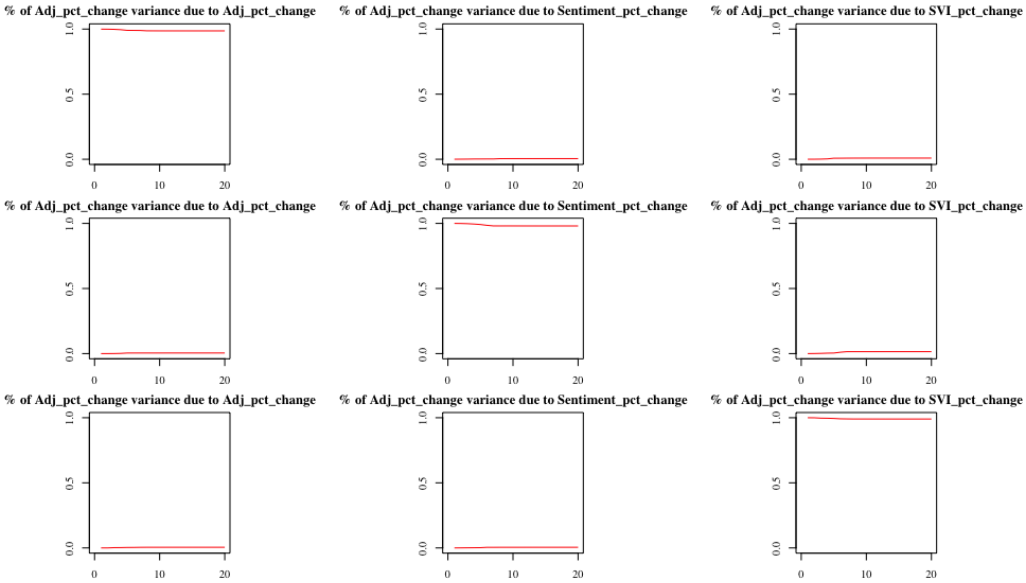
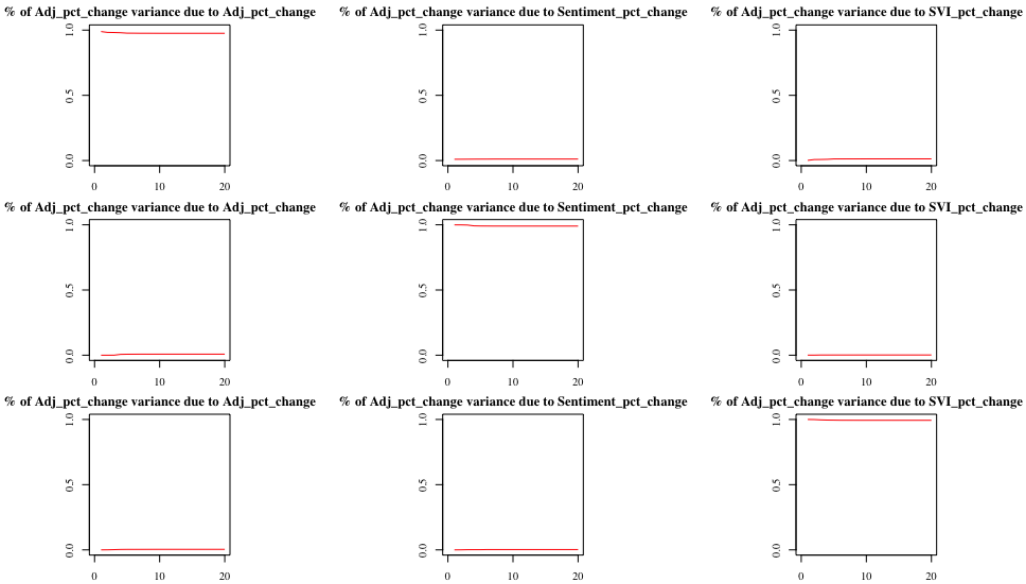


Figure 18 Reverse ordering Robustness test – Tesla



Section J – Packages, modules, and libraries

Table 5 List of Python packages, modules and libraries

Name	Operation
Emoji	Used for handling emojis in tweets
Langdetect	Used for language detection to identify non-English tweets
Matplotlib	Used for data manipulation and array manipulation
NLTK	Used for natural language processing (NLP) tasks, such as tokenization and stopword removal
Numpy	Used for numerical computing and array manipulation
Pandas	Used for data manipulation and analysis, particularly for handling and processing tabular data
Pandarallel	Used for parallel processing with pandas dataframes to optimize performance
Pytrends	Used for making requests to Google Trends API to retrieve search interest data
Re	Used for regular expression operations, used for pattern matching and text manipulation
Snscrape	Used to search, filter and extract date, username, and tweets
String	Used for string manipulation and processing
Transformers	Used for sentiment analysis using pre-trained model from Hugging Face, specifically the "sentiment-analysis" pipeline with "distilbert-base-uncased-finetuned-sst-2-english" model

Table 6 List of R packages, modules, and libraries

Name	Operation
car	Used for Companion to Applied Regression; contains various utility functions for regression analysis

dplyr	Used for data manupilation
forecast	Used for time series forecasting
ggfortify	Used for unifying the plotting output for any R object
ggplot2	Used for creating plots and visualizations
grid	Used for creating grid graphics, which is the basis of the graphics in ggplot2
gridExtra	Used for arranging multiple grid-based plots on a page
latex2exp	Used for converting LaTeX expressions to plot labels in ggplot2
lmtest	Used for performing statistical tests on linear regression models
Metrics	Used for calculating various statistical metrics

quantmod	Used for financial quantitative modelling
readxl	Used for reading Excel files (.xls, xlsx)
rmarkdown	Used for creating dynamic reports in R markdown format
stats	Used for statistical functions
tibble	Used for data organization and manipulation
tidyverse	Used for data manipulation, visualization, and data science
Tinytex	Used for managing LaTeX installations and dependencies for creating PDF reports
tseries	Used for time series analysis
urca	Used for performing unit root and cointegration tests in time series analysis

vars	Used for fitting Vector Autoregression (VAR) models
xts	Used for handling time series data as xts objects
zoo	Used for handling time series data as zoo objects

Codes

Importing tweets

Python code 1 utilizes the snscrape library to extract tweets related to Apple, Amazon, Microsoft, and Tesla, for the purpose of sentiment analysis. The specific search terms, official names, stock tickers, and related hashtags, are used to pull relevant tweets in English from users with at least 100 followers, from January 1, 2018, to March 1, 2023. A limit of 1,500,000 tweets is set for each company. The extracted tweets, including the date, username, and content, are stored in a pandas Data Frame and the date is converted to a time zone-unaware format for compatibility. Finally, the data is saved into separate Excel files for each company for subsequent analysis.

Apple

In [1]:

```
import snscrape.modules.twitter as sntwitter
import pandas as pd

search_words = ["Apple","apple","AAPL","aapl", "#Apple","#apple"]
min_followers = 100

query = " OR ".join([f"({word})" for word in search_words]) + f"
lang:en min_faves:{min_followers} since:2018-01-01 until:2023-03-
01"

tweets_AAPL = []
limit = 1500000

for tweet in sntwitter.TwitterSearchScraper(query).get_items():
    if len(tweets_AAPL) == limit:
        break
    else:
        tweets_AAPL.append([tweet.date, tweet.user.username,
tweet.rawContent])

AAPL = pd.DataFrame(tweets_AAPL, columns=['Date', 'User',
'Tweet'])
```

In [2]:

```
# Convert the datetime values to timezone-unaware datetime objects
AAPL['Date'] = AAPL['Date'].dt.tz_localize(None)

# Save the dataframe as an Excel file
AAPL.to_excel('/Users/****/Desktop/Apple.xlsx', index=False)
```

Amazon

In [3]:

```
search_words = ["Amazon", "amazon", "AMZN", "amzn", "#Amazon",
"#amazon"]
min_followers = 100
```

```
query = " OR ".join([f"({word})" for word in search_words]) + f"
lang:en min_faves:{min_followers} since:2018-01-01 until:2023-03-
01"

tweets_AMZN = []
limit = 1500000

for tweet in sntwitter.TwitterSearchScraper(query).get_items():
    if len(tweets_AMZN) == limit:
        break
    else:
        tweets_AMZN.append([tweet.date, tweet.user.username,
tweet.rawContent])

AMZN = pd.DataFrame(tweets_AMZN, columns=['Date', 'User',
'Tweet'])
```

In [4]:

```
# Convert the datetime values to timezone-unaware datetime objects
AMZN['Date'] = AMZN['Date'].dt.tz_localize(None)

# Save the dataframe as an Excel file
AMZN.to_excel('/Users/****/Desktop/Amazon.xlsx', index=False)
```

Microsoft

In [5]:

```
search_words = ["Microsoft","microsoft","MSFT","msft",
"#Microsoft","#microsoft"]
min_followers = 100

query = " OR ".join([f"({word})" for word in search_words]) + f"
lang:en min_faves:{min_followers} since:2018-01-01 until:2023-03-
01"

tweets_MSFT = []
limit = 1500000

for tweet in sntwitter.TwitterSearchScraper(query).get_items():
    if len(tweets_MSFT) == limit:
        break
    else:
        tweets_MSFT.append([tweet.date, tweet.user.username,
tweet.rawContent])

MSFT = pd.DataFrame(tweets_MSFT, columns=['Date', 'User',
'Tweet'])
```

In [6]:

```
# Convert the datetime values to timezone-unaware datetime objects
MSFT['Date'] = MSFT['Date'].dt.tz_localize(None)

# Save the dataframe as an Excel file
MSFT.to_excel('/Users/****/Desktop/Microsoft.xlsx', index=False)
```

Tesla

In [7]:

```
search_words = ["Tesla", "tesla", "TSLA", "tsla", "#Tesla",
"#tesla"]
min_followers = 100

query = " OR ".join([f"({word})" for word in search_words]) + f"
lang:en min_faves:{min_followers} since:2018-01-01 until:2023-03-
01"

tweets_TSLA = []
limit = 1500000

for tweet in sntwitter.TwitterSearchScraper(query).get_items():
    if len(tweets_TSLA) == limit:
        break
    else:
        tweets_TSLA.append([tweet.date, tweet.user.username,
tweet.rawContent])

TSLA = pd.DataFrame(tweets_TSLA, columns=['Date', 'User',
'Tweet'])
```

In [8]:

```
# Convert the datetime values to timezone-unaware datetime objects
TSLA['Date'] = TSLA['Date'].dt.tz_localize(None)

# Save the dataframe as an Excel file
TSLA.to_excel('/Users/****/Desktop/Tesla.xlsx', index=False)
```

Cleaning tweets

Python code 2 efficiently cleans the collected tweet data for sentiment analysis. It removes URLs, user mentions, hashtags, punctuation, numbers, and stop words, converts text to lowercase, and trims unnecessary spaces. Emojis replaced with their corresponding word meanings for a more accurate interpretation. Empty tweets are discarded post-cleaning. This cleaning process, expedited by the Pandarallel library, is applied to the tweet data of Apple, Amazon, Microsoft, and Tesla, resulting in structured and clean datasets ready sentiment analysis.

In [1]:

```
import pandas as pd
import re
import string
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from langdetect import detect
import emoji
from pandarallel import pandarallel

# Initialize Pandarallel
pandarallel.initialize(progress_bar=True)

def clean_tweet(tweet):
    # Remove URLs
    tweet = re.sub(r'http\S+', '', tweet)
    # Remove user mentions
    tweet = re.sub(r'@\S+', '', tweet)
    # Remove hashtags
    tweet = re.sub(r'#\S+', '', tweet)
    # Remove punctuation
    tweet = re.sub(r'[^\w\s]', '', tweet)
    # Remove numbers
    tweet = re.sub(r'\d+', '', tweet)
    # Convert to lowercase
    tweet = tweet.lower()
    # Remove extra whitespaces
    tweet = re.sub(r'\s+', ' ', tweet).strip()
    # Remove stopwords
    stopwords_list = stopwords.words('english')
    tweet_tokens = word_tokenize(tweet)
    tweet_tokens = [word for word in tweet_tokens if not word in
stopwords_list]
    tweet = ' '.join(tweet_tokens)
    # Detect language and remove non-english tweets
    try:
        lang = detect(tweet)
    except:
        lang = 'unknown'
    if lang != 'en':
        tweet = ''
    # Remove emojis
    tweet = emoji.demojize(tweet)
    return tweet

companies = {
```

```
    'Apple': '/Users/***/Desktop/Apple.xlsx',
    'AMZN': '/Users/***/Desktop/Amazon.xlsx',
    'Microsoft': '/Users/***/Desktop/Microsoft.xlsx',
    'Tesla': '/Users/***/Desktop/Tesla.xlsx',
}

for company, input_file in companies.items():
    # Read the input CSV file into a Pandas dataframe
    tweets_df = pd.read_excel(input_file)

    # Convert the date column to datetime format
    tweets_df['Date'] = pd.to_datetime(tweets_df['Date'])

    # Extract only the date part
    tweets_df['Date'] = tweets_df['Date'].dt.date

    # Apply the clean_tweet function to the 'Tweet' column of the
tweets_df dataframe in parallel
    tweets_df['Tweet'] =
tweets_df['Tweet'].parallel_apply(clean_tweet)

    # Drop rows with empty tweets
    tweets_df = tweets_df[tweets_df['Tweet'] != '']

    # Save the dataframe as a CSV file
    tweets_df.to_csv('/Users/***/Desktop/
{}_cleaned.csv'.format(company), index=False)
```

.

Sentiment analysis. A combination of the Loughran-McDonald dictionary and the Hugging Face sentiment analysis pipeline

Python code 3 represents the implementation of our sentiment analysis process. It harnesses the robustness of the Loughran-McDonald dictionary and the Hugging Face sentiment analysis pipeline. It initially preprocesses the Loughran-McDonald dictionary, making it fit for our task. The code then defines key functions for performing sentiment analysis with both the dictionary and the Hugging Face pipeline, extracting positive and negative sentiments for each text. In the end, the sentiment scores from these methods are averaged for each tweet in our dataset. This hybrid approach is applied to the cleaned tweets of Apple, Amazon, Microsoft, and Tesla, subsequently saving the resultant data frames, replete with sentiment scores, to new CSV files.

Importing the Loughran-McDonald_MasterDictionary_1993-2021

In [1]:

```
import pandas as pd

# Change the file path to the location of the file on your
computer
file_path = "/Users/***/Desktop/Loughran-
McDonald_MasterDictionary_1993-2021.csv"

# Read the CSV file into a pandas dataframe
dictionary = pd.read_csv(file_path)
```

Sentiment Analysis Score Computation

In [2]:

```
# Filtering and preprocessing sentiment dictionary
sentiment_dictionary = dictionary.loc[(dictionary['Negative'] !=
0) | (dictionary['Positive'] != 0), ['Word', 'Negative',
'Positive']]
sentiment_dictionary['Word'] =
sentiment_dictionary['Word'].str.lower()
sentiment_dictionary.loc[:, ['Negative', 'Positive']] =
sentiment_dictionary.loc[:, ['Negative', 'Positive']].apply(lambda
x: x.where(x == 0, 1))

import re
from transformers import pipeline

sentiment_pipeline = pipeline("sentiment-analysis",
model="distilbert-base-uncased-finetuned-sst-2-english")


def get_sentiment(text):
    result = sentiment_pipeline(text)[0]
    if result['label'] == 'POSITIVE':
        return result['score']
    elif result['label'] == 'NEGATIVE':
        return -result['score']
    else:
        return 0

def tokenize(text):
    return re.findall(r'\b\w+\b', text.lower())
```

```
def get_sentiment_score(sentiment_counts, token_count):
    if token_count == 0:
        return 0

    sentiment_score = (sentiment_counts['Positive'] -
sentiment_counts['Negative']) / token_count
    return sentiment_score

def compute_sentiment_scores(df, text_column,
sentiment_dictionary):
    df['Tokenized'] = df[text_column].apply(tokenize)
    df['Token_Count'] = df['Tokenized'].apply(len)

    token_df = df.explode('Tokenized')[['Tokenized',
'Token_Count']]
    token_df = token_df.merge(sentiment_dictionary,
left_on='Tokenized', right_on='Word', how='left').fillna(0)

    sentiment_counts =
token_df.groupby(token_df.index).agg({'Negative': 'sum',
'Positive': 'sum', 'Token_Count': 'first'})
    sentiment_scores =
sentiment_counts.assign(Sentiment_Score=lambda x: (x['Positive'] -
x['Negative']) / x['Token_Count'])

    return sentiment_scores['Sentiment_Score']

def compute_combined_sentiment_scores(df, text_column,
sentiment_dictionary):
    # Compute sentiment scores using the Loughran-McDonald
dictionary
    loughran_mcdonald_scores = compute_sentiment_scores(df,
text_column, sentiment_dictionary)

    # Compute sentiment scores using the Hugging Face sentiment
analysis pipeline
    hugging_face_scores = df[text_column].apply(get_sentiment)

    # Combine the sentiment scores and average them
    combined_sentiment_scores = (loughran_mcdonald_scores +
hugging_face_scores) / 2

    return combined_sentiment_scores
```

Apple

In [3]:

```
# Read the ticker_cleaned.csv file into a pandas dataframe
apple_tweets = pd.read_csv('/Users/***/Desktop/Apple_cleaned.csv')

# Compute sentiment scores for each tweet in the 'Tweet' column
apple_tweets['Sentiment_Score'] =
compute_combined_sentiment_scores(apple_tweets, 'Tweet',
sentiment_dictionary)

# Save the dataframe with sentiment scores to a new CSV file
apple_tweets.to_csv('/Users/***/Desktop/AAPL_LMC.csv',
index=False)
```

Amazon

In [4]:

```
# Read the ticker_cleaned.csv file into a pandas dataframe
amazon_tweets =
pd.read_csv('/Users/***/Desktop/Amazon_cleaned.csv')

# Compute sentiment scores for each tweet in the 'Tweet' column
amazon_tweets['Sentiment_Score'] =
compute_combined_sentiment_scores(amazon_tweets, 'Tweet',
sentiment_dictionary)

# Save the dataframe with sentiment scores to a new CSV file
amazon_tweets.to_csv('/Users/***/Desktop/AMZN_LMC.csv',
index=False)
```

Microsoft

In [5]:

```
# Read the ticker_cleaned.csv file into a pandas dataframe
microsoft_tweets =
pd.read_csv('/Users/***/Desktop/Microsoft_cleaned.csv')

# Compute sentiment scores for each tweet in the 'Tweet' column
microsoft_tweets['Sentiment_Score'] =
compute_combined_sentiment_scores(microsoft_tweets, 'Tweet',
sentiment_dictionary)

# Save the dataframe with sentiment scores to a new CSV file
microsoft_tweets.to_csv('/Users/***/Desktop/MSFT_LMC.csv',
index=False)
```

Tesla

In [6]:

```
# Read the ticker_cleaned.csv file into a pandas dataframe
tesla_tweets = pd.read_csv('/Users/***/Desktop/Tesla_cleaned.csv')

# Compute sentiment scores for each tweet in the 'Tweet' column
tesla_tweets['Sentiment_Score'] =
compute_combined_sentiment_scores(tesla_tweets, 'Tweet',
sentiment_dictionary)

# Save the dataframe with sentiment scores to a new CSV file
tesla_tweets.to_csv('/Users/***/Desktop/TSLA_LMC.csv',
index=False)
```

Collecting and creating the SVI

Python code 4 illustrates our approach to harnessing Google Trends data for the construction of a daily SVI. The Python library 'pytrends' enables us to obtain this data, while we've crafted a custom function to collect and process the search term data on both a daily and weekly basis. Our SVI creation process involves scaling the daily data using weekly data to accommodate fluctuations in overall search volume, followed by a smoothing operation to mitigate potential noise. This procedure ensures a refined, reliable representation of public interest over time. The scaled and smoothed daily SVI is subsequently visualized, comparing it against the interpolated weekly data to validate our methodology. This process is applied for the search terms corresponding to Apple, Amazon, Microsoft, and Tesla, resulting in individual CSV files for each entity's SVI.

Creating formula for collecting Google Trends terms

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pytrends.request import TrendReq

def get_scaled_trend_data(search_term):
    # Create Pytrends object
    pytrends = TrendReq(hl='en-US', tz=360)

    # Define search term and time period
    kw_list = [search_term]
    start_date = '2017-12-29' #Since rolling window
    end_date = '2023-03-5'

    # Collect daily data
    date_range = pd.date_range(start=start_date, end=end_date,
freq='90D')
    daily_df = pd.DataFrame()

    for date in date_range:
        date_str = date.strftime('%Y-%m-%d')
        end_date_str = (date +
pd.DateOffset(days=89)).strftime('%Y-%m-%d')

        pytrends.build_payload(kw_list, cat=0,
timeframe=f'{date_str} {end_date_str}', geo='', gprop='')
        df = pytrends.interest_over_time()

        df = df.resample('D').mean().ffill()
        daily_df = daily_df.append(df)

    daily_df =
daily_df.loc[~daily_df.index.duplicated(keep='first')]

    # Collect weekly data
    pytrends.build_payload(kw_list, cat=0,
timeframe=f'{start_date} {end_date}', geo='', gprop='')
    weekly_df= pytrends.interest_over_time()

    weekly_df = weekly_df.resample('W').mean().ffill()
```

```
    weekly_df =
weekly_df.loc[~weekly_df.index.duplicated(keep='first')]

    # Resample weekly_df to daily frequency
    weekly_df_daily = weekly_df.resample('D').mean().interpolate()

    # Scale daily data
    scaling_factors = (weekly_df[search_term] /
daily_df.resample('W').mean()[search_term]).dropna()

    scaled_daily_data = daily_df.copy()
    scaled_daily_data.drop(columns=['isPartial'], inplace=True)

    for date, scaling_factor in scaling_factors.items():
        start_date = date - pd.DateOffset(days=6)
        end_date = date
        mask = (scaled_daily_data.index >= start_date) &
(scaled_daily_data.index <= end_date)
        scaled_daily_data.loc[mask, search_term] *= scaling_factor

    scaled_daily_data[search_term] =
scaled_daily_data[search_term].clip(upper=100)

    # Smooth the scaled_daily_data
    window_size = 7
    scaled_daily_data_smoothed =
scaled_daily_data.rolling(window=window_size, center=True).mean()

    # Plot the final comparison of smoothed scaled daily data and
weekly data
    plt.figure(figsize=(12, 6))
    plt.plot(scaled_daily_data_smoothed.index,
scaled_daily_data_smoothed[search_term], label=f'Scaled Daily Data
(Smoothed) {search_term}')
    plt.plot(weekly_df_daily.index,
weekly_df_daily[[search_term]], label=f'Weekly Data (Interpolated)
{search_term}')
    plt.xlabel('Date')
    plt.ylabel(f'{search_term} Search Interest')
    plt.title(f'Scaled Daily (Smoothed) {search_term} Search
Interest vs Weekly {search_term} Search Interest')
    plt.legend()
    plt.grid()
    plt.show()

    return scaled_daily_data_smoothed
```

AAPL

In [2]:

```
search_term = 'AAPL'
scaled_trend_data = get_scaled_trend_data(search_term)

output = "/Users/***/Desktop/{}_SVI.csv".format(search_term)
scaled_trend_data.to_csv(output, index=True)
```

AMZN

In [3]:

```
search_term = 'AMZN'
scaled_trend_data = get_scaled_trend_data(search_term)

output = "/Users/***/Desktop/{}_SVI.csv".format(search_term)
scaled_trend_data.to_csv(output, index=True)
```

MSFT

In [4]:

```
search_term = 'MSFT'
scaled_trend_data = get_scaled_trend_data(search_term)

output = "/Users/***/Desktop/{}_SVI.csv".format(search_term)
scaled_trend_data.to_csv(output, index=True)
```

TSLA

In [5]:

```
search_term = 'TSLA'
scaled_trend_data = get_scaled_trend_data(search_term)

output = "/Users/***/Desktop/{}_SVI.csv".format(search_term)
scaled_trend_data.to_csv(output, index=True)
```

Analysis

The R code is used to perform the data analysis workflow using statistical techniques. The code begins with the importation and loading of the datasets, followed by pre-processing steps such as handling missing values and transforming variables. It then proceeds to perform exploratory data analysis, including descriptive statistics and visualizations, to gain insights into the dataset's characteristics ensuring the data is suitable for further analysis. Next, the code employs statistical modelling techniques, such as setting up multiple VAR system, to investigate relationships and make predictions based on the data. Finally, the code evaluates the model's performance using appropriate metrics and presents the results.

```
# Loading libraries and preparing data

# Load libraries
rm(list=ls())

library(car)

library(dplyr)

library(forecast)

library(ggfortify)

library(ggplot2)

library(grid)

library(gridExtra)

library(latex2exp)

library(lmtest)

library(Metrics)

library(quantmod)

library(readxl)

library(rmarkdown)

library(stats)

library(tibble)

library(tidyverse)

library(tinytex)

library(tseries)

library(urca)

library(vars)

library(xts)
```

```r
library(zoo)

# Set Working directory
setwd("/Users/***/Desktop")

# Set the ticker symbol you want to investigate
#Change the ticker name to AAPL, AMZN, MSFT or TSLA to perform the analy
sis on these companies

ticker <- "AAPL"

# Import sentiment data
df <- read.csv(paste0(ticker, "_LMC.csv"), header = TRUE)

# Remove unnecessary columns
df <- df[,c("Date", "Sentiment_Score")]

# Calculate the mean sentiment score for each day
df <- df %>%
  group_by(Date) %>%
  summarize(Sentiment_Score = mean(Sentiment_Score))

# Import Google Trends data
SVI <- read.csv(paste0(ticker, "_SVI.csv"), header = TRUE)

# Remove the first 3 rows
SVI <- SVI[-(1:3), ]

# Remove the last 47 rows
SVI <- head(SVI, n = -47)

# Define the date range of interest
start_date <- as.Date("2018-01-01")
end_date <- as.Date("2023-02-01")

# Use getSymbols() function to download the stock data from Yahoo Financ
e
getSymbols(ticker, from = start_date, to = end_date)

## [1] "AAPL"

# Extract the Adj Close price
adj_close <- as.data.frame(Ad(get(ticker)))
adj_close <- rownames_to_column(adj_close, var = "Date")

# Convert sentiment data to xts time series object
sentiment_xts <- xts(df$Sentiment_Score, order.by = as.Date(df$Date))

# Convert Google Trends data to xts time series object
SVI_xts <- xts(SVI[ticker], order.by = as.Date(SVI$date))

# Convert stock data to xts time series object, using the ticker
variable
stock_data_xts <- xts(adj_close[, paste0(ticker, ".Adjusted")], order.by
= as.Date(adj_close$Date))

# Merge the sentiment scores with adj close data
merged_data <- merge(stock_data_xts, sentiment_xts, SVI_xts, all =
FALSE)

# Change the column name in merged_data to use the ticker variable
colnames(merged_data) <- c(ticker, "Sentiment_Score", "SVI")
```

```r
# Convert the merged_data back to a data frame
merged_data_df <- data.frame(Date=index(merged_data),
coredata(merged_data))

# Make sure the data is numeric
merged_data_df[[ticker]] <- as.numeric(merged_data_df[[ticker]])
merged_data_df$Sentiment_Score <-
as.numeric(merged_data_df$Sentiment_Score)
merged_data_df$SVI <- as.numeric(merged_data_df$SVI)


# Calculate percentage change of the stock's Adj Close, using the ticker
variable
merged_data_df$Adj_pct_change <- c(NA, diff(merged_data_df[[ticker]]) /
merged_data_df[[ticker]][-length(merged_data_df[[ticker]])]) * 100

# Calculate percentage change of mean sentiment score
merged_data_df$Sentiment_pct_change <-c(NA,
diff(merged_data_df$Sentiment_Score))*100

# Calculate percentage change of the SVI
merged_data_df$SVI_pct_change <- c(NA, diff(merged_data_df$SVI) /
merged_data_df$SVI[-length(merged_data_df$SVI)])*100


# Remove the first row with NA in the Adj_pct_change and
Sentiment_pct_change and SVI_pct_change columns
merged_data_df <- merged_data_df[-1, ]


################################################################################
# Summary statistics
summary_stats <- summary(merged_data_df[c("Adj_pct_change",
"Sentiment_pct_change", "SVI_pct_change")])

# Correlation analysis
correlation_matrix <- cor(merged_data_df[, c("Adj_pct_change",
"Sentiment_pct_change", "SVI_pct_change")], use="complete.obs")

# Calculate VIF for each variable
vif_results <- vif(lm(Adj_pct_change ~ Sentiment_pct_change +
SVI_pct_change, data = merged_data_df))


# Histograms
hist_plots <- lapply(names(merged_data_df[, c("Adj_pct_change", "Sentime
nt_pct_change", "SVI_pct_change")]), function(var_name) {
  ggplot(merged_data_df, aes_string(x = var_name)) +
    geom_histogram(bins = 30, fill = 'blue', color = 'black') +
    theme_minimal()
})



# Display the histograms
grid.arrange(grobs = hist_plots, ncol = 3,
             top = grid::textGrob(label = paste("Distribution of values:
","(", ticker,")"),
                                  gp = grid::gpar(fontface = "bold"))
)
```

```r
# Create scatterplot for Adj_pct_change vs Sentiment_pct_change
scatterplot_adj_sentiment <- ggplot(merged_data_df, aes(x = Adj_pct_chan
ge, y = Sentiment_pct_change)) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE, color = 'red', linetype = 'dash
ed') +
  labs(x = "Adj_pct_change", y = "Sentiment_pct_change") +
  theme_minimal()

# Create scatterplot for Adj_pct_change vs SVI_pct_change
scatterplot_adj_svi <- ggplot(merged_data_df, aes(x = Adj_pct_change, y
= SVI_pct_change)) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE, color = 'red', linetype = 'dash
ed') +
  labs(x = "Adj_pct_change", y = "SVI_pct_change") +
  theme_minimal()

# Create scatterplot for Sentiment_pct_change vs SVI_pct_change
scatterplot_sentiment_svi <- ggplot(merged_data_df, aes(x = Sentiment_pc
t_change, y = SVI_pct_change)) +
  geom_point() +
  geom_smooth(method = 'lm', se = FALSE, color = 'red', linetype = 'dash
ed') +
  labs(x = "Sentiment_pct_change", y = "SVI_pct_change") +
  theme_minimal()

# Arrange the scatterplots in a grid
grid.arrange(scatterplot_adj_sentiment, scatterplot_adj_svi,
             scatterplot_sentiment_svi,
             nrow = 2, ncol = 2,
             top = grid::textGrob(label = paste("Scatterplot Relationshi
ps of Variables", "(", ticker, ")"),
                                  gp = grid::gpar(fontface = "bold"))
)



##########################################################################
# Plot the Adj Close, SCI and Sentiment Score
# Create a long-format data frame for ggplot
long_data_df <- merged_data_df %>%
  gather(key = "Variable", value = "Value", ticker, Sentiment_Score, SVI
)



# Create a plot with three panels, one for each variable
multi_panel_plot <- ggplot(long_data_df, aes(x = Date, y = Value)) +
  geom_line() +
  facet_wrap(~ Variable, scales = "free_y", ncol = 1) +
  labs(title = paste("Time Series Plot of Variables","(",ticker,")") , x
= "Date", y = "Values") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 14, fac
e = "bold"))

# Print the plot
print(multi_panel_plot)
```

```
#########################################################################
# Plot the Adj pct change, pct change in sentiment and the pct change in
SVI

# Create a long-format data frame for ggplot
suppressWarnings({
  long_data_df <- merged_data_df %>%
    gather(key = "Variable", value = "Value", Adj_pct_change, Sentiment_
pct_change, SVI_pct_change)
})


# Create a plot with three panels, one for each variable
multi_panel_plot <- ggplot(long_data_df, aes(x = Date, y = Value)) +
  geom_line() +
  facet_wrap(~ Variable, scales = "free_y", ncol = 1) +
  labs(title = paste("Time Series Plot of Variables","(",ticker,")"), x
= "Date", y = "First differences") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 14, fac
e = "bold")) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")

# Print the plot
print(multi_panel_plot)



#########################################################################
# Analyze autocorrelation using ACF and PACF plots
plot_acf_pacf <- function(ts_data, title, lag_length) {
  acf_plot <- ggAcf(ts_data, lag.max = lag_length) + ggtitle(paste0("ACF
: ", title))
  pacf_plot <- ggPacf(ts_data, lag.max = lag_length) + ggtitle(paste0("P
ACF: ", title))
  return(list(acf_plot, pacf_plot))
}

# Define the lag length
lag_length <- 20

# Generate plots
stock_plots <- plot_acf_pacf(merged_data_df$Adj_pct_change, paste("Adj C
lose Percentage Change"), lag_length)
sentiment_plots <- plot_acf_pacf(merged_data_df$Sentiment_pct_change, "S
entiment Percentage Change", lag_length)
svi_plots <- plot_acf_pacf(merged_data_df$SVI_pct_change, "SVI Percentag
e Change", lag_length)

# Combine all plots into one with a title
title_grob <- textGrob(paste("ACF and PACF for", ticker), gp = gpar(font
face = "bold"))
gridExtra::grid.arrange(grobs = c(stock_plots, sentiment_plots, svi_plot
s),
                        ncol = 2,
                        top = title_grob)

#########################################################################
# Split the data frame into the specified periods
in_sample_df <- merged_data_df[ merged_data_df$Date >= "2018-01-01" & me
rged_data_df$Date < "2022-12-31", ]
out_of_sample_df <- merged_data_df[ merged_data_df$Date >= "2023-01-01"
```

```
& merged_data_df$Date <= "2023-02-01", ]

##############################################################################
# Check for stationarity

suppressWarnings({
  adf_adj <- adf.test(in_sample_df$Adj_pct_change)
  adf_sent <- adf.test(in_sample_df$Sentiment_pct_change)
  adf_SVI <- adf.test(in_sample_df$SVI_pct_change)
})

# Print ADF test results
cat("ADF test for Adj_pct_change:", ticker, "\n")

##############################################################################
#########              AR model & RW model                 ##########
##############################################################################
#Fiting and optimal lag selection of the ar model

# Function to fit AR model with different information criteria
fit_AR_model <- function(in_sample_data, max_lag, ic_type) {

  # Create a dataframe to store ICs
  IC_df <- data.frame(lag = 1:max_lag, AIC = rep(0, max_lag), BIC = rep(
0, max_lag),
                      HQIC = rep(0, max_lag), FPE = rep(0, max_lag))

  # Calculate ICs for each lag
  for (i in 1:max_lag) {
    model <- Arima(in_sample_data, order = c(i, 0, 0))
    IC_df$AIC[i] <- model$aic
    IC_df$BIC[i] <- BIC(model)
    IC_df$HQIC[i] <- log(model$sigma2) + 2*i*log(log(nrow(in_sample_df))
)/nrow(in_sample_df)
  }

  # Print all the criteria
  cat("AIC:", IC_df$AIC, "\n")
  cat("BIC:", IC_df$BIC, "\n")
  cat("HQIC:", IC_df$HQIC, "\n")

  # Determine the optimal lag based on the chosen criterion
  if(ic_type == "AIC") {
    optimal_lag <- which.min(IC_df$AIC)
  } else if(ic_type == "BIC") {
    optimal_lag <- which.min(IC_df$BIC)
  } else if(ic_type == "HQIC") {
    optimal_lag <- which.min(IC_df$HQIC)
  }

  # Fit the AR model using the optimal lag
  ar_model <- Arima(in_sample_data, order = c(optimal_lag, 0, 0))

  return(ar_model)
}

# Choose the IC type
ic_type <- "AIC"  # Change to "AIC", "BIC", "HQIC", or "FPE" as needed

# Fit the AR model
ar_model <- fit_AR_model(in_sample_df$Adj_pct_change, 20, ic_type)
```

```
print(ar_model)

# Initialize variables to store the forecast values
ar_forecast_values <- numeric(length(out_of_sample_df$Adj_pct_change))
rw_forecast_values <- numeric(length(out_of_sample_df$Adj_pct_change))
# Add this line

# Perform AR and random walk with drift forecasts
drift_term <- mean(diff(in_sample_df$Adj_pct_change))  # Calculate the d
rift term
rw_forecast_values[1] <- out_of_sample_df$Adj_pct_change[1]

for (i in 2:length(out_of_sample_df$Adj_pct_change)) {
  # Combine in-sample and out-of-sample data up to the current point
  combined_data <- ts(c(in_sample_df$Adj_pct_change, out_of_sample_df$Ad
j_pct_change[1:(i - 1)]))

  # Fit the AR model to the combined data using auto.arima() function
  ar_model <- auto.arima(combined_data, ic = "aic", stationary = TRUE)

  # Generate a one-step-ahead AR forecast
  ar_forecast_values[i] <- forecast(ar_model, h = 1)$mean

  # Use the last observed value as the forecast for the random walk
  rw_forecast_values[i] <- out_of_sample_df$Adj_pct_change[i - 1] + drif
t_term
}

# Combine the actual and forecast values in a data frame
comparison_df <- data.frame(Date = out_of_sample_df$Date, Actual = out_o
f_sample_df$Adj_pct_change, AR_Forecast = ar_forecast_values, RW_Forecas
t = rw_forecast_values)
comparison_df$Date <- as.Date(comparison_df$Date)

# Plot the actual and forecast values
comparison_plot <- ggplot(comparison_df, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_line(aes(y = AR_Forecast, color = "AR Model")) +
  geom_line(aes(y = RW_Forecast, color = "Random Walk")) +
  labs(title = paste("Actual vs Forecast Return","(", ticker,")"), x = "
Date", y = "% change in Return", color = "Series") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 14, fac
e = "bold"))

print(comparison_plot)

##############################################################################
#########          Adj Close, Sentiment and SVI          ############
##############################################################################
# optimal lag selection and VAR-framework

# Define y1,t as the logged values of Sentiment score
y1t <- in_sample_df$Adj_pct_change

# Define y2,t as the logged values of pct change
y2t <- in_sample_df$Sentiment_pct_change

y3t <- in_sample_df$SVI_pct_change

# Construct y matrix
y <- matrix( c(y1t, y2t, y3t), ncol=3 )
```

```r
colnames(y) <- c("Adj_pct_change", "Sentiment_pct_change", "SVI_pct_chan
ge")

# Optimal lags
opt_lag <- VARselect(y, lag.max=20)
cat("Optimal lag selection:", ticker, "\n")

## Optimal lag selection: AAPL

print(opt_lag)

paste0("HQIC: VAR(", opt_lag$selection[2], ")")

# Fit VAR model
model_fit <- VAR(y, p=opt_lag$selection[2])
cat("VAR estimation results:", ticker, "\n")

summary(model_fit)

##############################################################################
# Perform the Granger causality tests
granger_test1 <- causality(model_fit, cause = c("Sentiment_pct_change",
"SVI_pct_change"))$Granger
granger_test2 <- causality(model_fit, cause = c("Adj_pct_change", "SVI_p
ct_change"))$Granger
granger_test3 <- causality(model_fit, cause = c("Adj_pct_change", "Senti
ment_pct_change"))$Granger

# Prepare data for the data frame
cause_vars <- c("Sentiment_pct_change & SVI_pct_change", "Adj_pct_change
& SVI_pct_change",
                "Adj_pct_change & Sentiment_pct_change")
effect_vars <- c("Adj_pct_change", "Sentiment_pct_change", "SVI_pct_chan
ge")
test_results <- list(granger_test1, granger_test2, granger_test3)

# Function to extract p-values from the causality test results
get_test_info <- function(test_result) {
  p_value <- format(round(test_result$p.value, 4), nsmall = 4)
  return(p_value)
}

# Function to assign significance based on p-value
assign_significance <- function(p) {
  p = as.numeric(p)
  if (p <= 0.01) {
    return("***")
  } else if (p <= 0.05) {
    return("**")
  } else if (p <= 0.1) {
    return("*")
  } else {
    return("")
  }
}

# Gather the p-values
p_values <- sapply(test_results, get_test_info)

# Assign significance levels based on p-values
significance <- sapply(p_values, assign_significance)

# Create a data frame to store the p-values and significance
```

```r
result_df <- data.frame(
  Cause = cause_vars,
  Effect = effect_vars,
  P_Values = p_values,
  Significance = significance
)


# Print the title
cat("\nGranger Causality Test Results:", ticker, "\n")

# Print the results
print(result_df,row.names = FALSE)

# Print the legend for significance levels
cat("\np <= 0.1: *, p <= 0.05: **, p <= 0.01: ***")

##############################################################################
# Test for serial correlation of residuals using the Durbin-Watson stati
stic

# Calculate the residuals from the fitted VAR model
residuals <- residuals(model_fit)

# Calculate the Durbin Watson statistic for each variable

dwt <- dwtest(lm(residuals ~ 1))$statistic

cat("Durbin-Watson statistic for", colnames(residuals), ":", dwt , "(",
ticker, ")", "\n") # A value of close to 2 indicate no serial correlatio
n

##############################################################################
# Impulse Response Function (IRF) for all combinations of variables
irf_steps <- 20 # Number of steps ahead for IRF

var_names <- colnames(y)

# Run the irf function
imp_results <- irf(model_fit, n.ahead = irf_steps, ortho = FALSE)

round( imp_results$irf$Sentiment_pct_change, 4)



# Set up the plot parameters
par(mfrow  = c(length(var_names),length(var_names)),
    pty    = "s",
    las    = 0,
    mgp    = c(2.5, 1, 0),
    mar    = c(2, 0, 2, 0),
    family = "serif")

# Upper and lower bounds for confidence intervals (assuming 95%)
upper_bound_Adj <- imp_results$Upper$Adj_pct_change
lower_bound_Adj <- imp_results$Lower$Adj_pct_change

upper_bound_Sentiment <- imp_results$Upper$Sentiment_pct_change
lower_bound_Sentiment <- imp_results$Lower$Sentiment_pct_change

upper_bound_SVI <- imp_results$Upper$SVI_pct_change
lower_bound_SVI <- imp_results$Lower$SVI_pct_change
```

```
# Plot 1: Response of Adj close to Adj Close
plot(0:irf_steps, imp_results$irf$Adj_pct_change[,1], type = "l", col  =
"red",
     xlab = "", ylab = "", main = "Response of Adj close to Adj Close",
     xlim = c(0, irf_steps), ylim = c(-0.1, 1),
     xaxp = c(0, irf_steps, 2), yaxp = c(0, 1, 2))
lines(0:irf_steps, upper_bound_Adj[,1], col = "blue",lty = "dotted")
lines(0:irf_steps, lower_bound_Adj[,1], col = "blue",lty = "dotted")

# Plot 2: Response of Adj close to Sentiment Percentage change
plot(0:irf_steps, imp_results$irf$Sentiment_pct_change[,1], type = "l",
col  = "red",
     xlab = "", ylab = "", main = "Response of Adj close to Sentiment",
     xlim = c(0, irf_steps), ylim = c(-0.2, 0.2),
     xaxp = c(0, irf_steps, 2), yaxp = c(-0.1, 0.1, 2))
lines(0:irf_steps, upper_bound_Sentiment[,1], col = "blue",lty = "dotted
")
lines(0:irf_steps, lower_bound_Sentiment[,1], col = "blue",lty = "dotted
")

# Plot 3: Response of Adj Close to SVI percentage change
plot(0:irf_steps, imp_results$irf$SVI_pct_change[,1], type = "l", col  =
"red",
     xlab = "", ylab = "", main = "Response of Adj Close to SVI",
     xlim = c(0, irf_steps), ylim = c(-0.2, 0.2),
     xaxp = c(0, irf_steps, 2), yaxp = c(-0.1, 0.1, 2))
lines(0:irf_steps, upper_bound_SVI[,1], col = "blue",lty = "dotted")
lines(0:irf_steps, lower_bound_SVI[,1], col = "blue",lty = "dotted")

# Plot 4: Response of Sentiment percentage change to Adj Close
plot(0:irf_steps, imp_results$irf$Adj_pct_change[,2], type = "l", col  =
"red",
     xlab = "", ylab = "", main = "Response of Sentiment to Adj Close",
     xlim = c(0, irf_steps), ylim = c(-0.2, 0.2),
     xaxp = c(0, irf_steps, 2), yaxp = c(-0.1, 0.1, 2))
lines(0:irf_steps, upper_bound_Adj[,2], col = "blue",lty = "dotted")
lines(0:irf_steps, lower_bound_Adj[,2], col = "blue",lty = "dotted")

# Plot 5: Response of Sentiment percentage change to Sentiment percentag
e change
plot(0:irf_steps, imp_results$irf$Sentiment_pct_change[,2], type = "l",
col  = "red",
     xlab = "", ylab = "", main = "Response of Sentiment to Sentiment",
     xlim = c(0, irf_steps), ylim = c(-0.1, 1),
     xaxp = c(0, irf_steps, 2), yaxp = c(0, 1, 2))
lines(0:irf_steps, upper_bound_Sentiment[,2], col = "blue",lty = "dotted
")
lines(0:irf_steps, lower_bound_Sentiment[,2], col = "blue",lty = "dotted
")

# Plot 6: Response of Sentiment percentage change to SVI
plot(0:irf_steps, imp_results$irf$SVI_pct_change[,2], type = "l", col  =
"red",
     xlab = "", ylab = "", main = "Response of Sentiment percentage to S
VI",
     xlim = c(0, irf_steps), ylim = c(-0.2, 0.2),
     xaxp = c(0, irf_steps, 2), yaxp = c(-0.1, 0.1, 2))
lines(0:irf_steps, upper_bound_SVI[,2], col = "blue",lty = "dotted")
lines(0:irf_steps, lower_bound_SVI[,2], col = "blue",lty = "dotted")

# Plot 7: Response of SVI to Adj Close
plot(0:irf_steps, imp_results$irf$Adj_pct_change[,3], type = "l", col  =
```

```
"red",
     xlab = "", ylab = "", main = "Response of SVI to Adj Close",
     xlim = c(0, irf_steps), ylim = c(-0.2, 0.2),
     xaxp = c(0, irf_steps, 2), yaxp = c(-0.1, 0.1, 2))
lines(0:irf_steps, upper_bound_Adj[,3], col = "blue",lty = "dotted")
lines(0:irf_steps, lower_bound_Adj[,3], col = "blue",lty = "dotted")

# Plot 8: Response of SVI to Sentiment Percentage change
plot(0:irf_steps, imp_results$irf$Sentiment_pct_change[,3], type = "l",
col  = "red",
     xlab = "", ylab = "", main = "Response of SVI to Sentiment",
     xlim = c(0, irf_steps), ylim = c(-0.2, 0.2),
     xaxp = c(0, irf_steps, 2), yaxp = c(-0.1, 0.1, 2))
lines(0:irf_steps, upper_bound_Sentiment[,3], col = "blue",lty = "dotted
")
lines(0:irf_steps, lower_bound_Sentiment[,3], col = "blue",lty = "dotted
")

# Plot 9: Response of SVI to SVI
plot(0:irf_steps, imp_results$irf$SVI_pct_change[,3], type = "l", col  =
"red",
     xlab = "", ylab = "", main = "Response of SVI to SVI",
     xlim = c(0, irf_steps), ylim = c(-0.1, 1),
     xaxp = c(0, irf_steps, 2), yaxp = c(0, 1, 2))
lines(0:irf_steps, upper_bound_SVI[,3], col = "blue",lty = "dotted")
lines(0:irf_steps, lower_bound_SVI[,3], col = "blue",lty = "dotted")

##############################################################################
# Variance decomposition

vd_steps <- 20 # Number of steps ahead for variance decomposition

var_dec <- fevd( model_fit, n.ahead=vd_steps )

round( matrix( c( var_dec$Adj_pct_change[,1],
                  var_dec$Adj_pct_change[,2],
                  var_dec$Adj_pct_change[,3],
                  var_dec$Adj_pct_change[,1] + var_dec$Adj_pct_change[,2
] + var_dec$Adj_pct_change[,3] ),
               ncol=4 ),
       4)



round( matrix( c( var_dec$Sentiment_pct_change[,1],
                  var_dec$Sentiment_pct_change[,2],
                  var_dec$Sentiment_pct_change[,3],
                  var_dec$Sentiment_pct_change[,1] + var_dec$Sentiment_p
ct_change[,2] + var_dec$Sentiment_pct_change[,3] ),
               ncol=4 ),
       4)



round( matrix( c( var_dec$SVI_pct_change[,1],
                  var_dec$SVI_pct_chang[,2],
                  var_dec$SVI_pct_chang[,3],
                  var_dec$SVI_pct_chang[,1] + var_dec$SVI_pct_chang[,2]
+ var_dec$SVI_pct_chang[,3] ),
               ncol=4 ),
       4)
```

```
par(mfrow  = c(3,3),
    pty    = "s",
    las    = 0,
    mgp    = c(2.5, 1, 0),
    mar    = c(2, 0, 2, 0),
    family ="serif")
plot(1:vd_steps, var_dec$Adj_pct_change[,1],                type = "l",
col  = "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Adj_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$Adj_pct_change[,2],                type = "l",
col  = "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Sentiment_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$Adj_pct_change[,3],                type = "l",
col  = "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to SVI_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$Sentiment_pct_change[,1],                type
= "l", col  = "red",
     xlab = "",           ylab = "",            main = "% of Sentiment_
pct_change variance due to Adj_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$Sentiment_pct_change[,2],                type
= "l", col  = "red",
     xlab = "",           ylab = "",            main = "% of Sentiment_
pct_change variance due to Sentiment_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$Sentiment_pct_change[,3],                type
= "l", col  = "red",
     xlab = "",           ylab = "",            main = "% of Sentiment_
pct_change variance due to SVI_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$SVI_pct_change[,1],                type = "l",
col  = "red",
     xlab = "",           ylab = "",            main = "% of SVI_pct_ch
ange variance due to Adj_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$SVI_pct_change[,2],                type = "l",
col  = "red",
     xlab = "",           ylab = "",            main = "% of SVI_pct_ch
ange variance due to Sentiment_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
plot(1:vd_steps, var_dec$SVI_pct_change[,3],                type = "l",
col  = "red",
     xlab = "",           ylab = "",            main = "% of SVI_pct_ch
ange variance due to SVI_pct_change",
     xlim = c(0, vd_steps),    ylim = c(0, 1),
     xaxp = c(0, vd_steps, 2), yaxp = c(0, 1, 2))
```

```r
##########################################################################
# Robustness check: Reverse ordering

tau = 20

y_robust <- matrix( c(y3t, y2t, y1t), ncol=3 )
colnames(y_robust) <- c("SVI_pct_change", "Sentiment_pct_change", "Adj_p
ct_change")


# Optimal lags
opt_lag_robust <- VARselect(y_robust, lag.max=20)
cat("Optimal lag selection:", ticker, "\n")

## Optimal lag selection: AAPL

print(opt_lag_robust)

paste0("HQIC: VAR(", opt_lag_robust$selection[2], ")")

# Fit VAR model
model_fit_robust <- VAR(y_robust, p=opt_lag_robust$selection[2])
cat("VAR estimation results:", ticker, "\n")



var_dec <- fevd( model_fit_robust, n.ahead=tau )

par(mfrow  = c(3,3),
    pty    = "s",
    las    = 0,
    mgp    = c(2.5, 1, 0),
    mar    = c(2, 0, 2, 0),
    family ="serif")
plot(1:tau, var_dec$Adj_pct_change[,3],                type = "l", col
= "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Adj_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$Adj_pct_change[,2],                type = "l", col
= "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Sentiment_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$Adj_pct_change[,1],                type = "l", col
= "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to SVI_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$Sentiment_pct_change[,3],              type = "l"
, col  = "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Adj_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$Sentiment_pct_change[,2],              type = "l"
, col  = "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Sentiment_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
```

```
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$Sentiment_pct_change[,1],                type = "l"
, col  = "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to SVI_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$SVI_pct_change[,3],                type = "l", col
= "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Adj_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$SVI_pct_change[,2],                type = "l", col
= "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to Sentiment_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
plot(1:tau, var_dec$SVI_pct_change[,1],                type = "l", col
= "red",
     xlab = "",           ylab = "",            main = "% of Adj_pct_ch
ange variance due to SVI_pct_change",
     xlim = c(0, tau),    ylim = c(0, 1),
     xaxp = c(0, tau, 2), yaxp = c(0, 1, 2))
```

```
##############################################################################
# VAR forecast
n_steps <- 1
y_t_plus_1_forecast <- predict(model_fit, n.ahead = n_steps, ci = 0.95)

# Initialize variables to store the forecast values
forecast_values <- numeric(length(out_of_sample_df$Adj_pct_change))

for (i in 2:length(out_of_sample_df$Adj_pct_change)) {
  # Combine in-sample and out-of-sample data up to the current point
  combined_data <- rbind(in_sample_df[, c("Adj_pct_change", "Sentiment_p
ct_change", "SVI_pct_change")], out_of_sample_df[1:(i - 1), c("Adj_pct_c
hange", "Sentiment_pct_change", "SVI_pct_change")])

  # Fit the VAR model to the combined data
  var_model <- VAR(combined_data, p = opt_lag$selection[2])

  # Generate a one-step-ahead forecast
  forecast_result <- predict(var_model, n.ahead = n_steps)
  forecast_values[i] <- forecast_result$fcst[[1]][1]
}

##############################################################################
# Combine the actual and forecast values in a data frame
comparison_df <- data.frame(Date = out_of_sample_df$Date,
                            Actual = out_of_sample_df$Adj_pct_change,
                            VAR_Forecast = forecast_values,
                            AR_Forecast = ar_forecast_values,
                            RW_Forecast = rw_forecast_values)

comparison_df$Date <- as.Date(comparison_df$Date)
```

```r
# Plot the actual and forecast values
comparison_plot <- ggplot(comparison_df, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_line(aes(y = VAR_Forecast, color = "VAR Forecast")) +
  geom_line(aes(y = AR_Forecast, color = "AR Forecast")) +
  geom_line(aes(y = RW_Forecast, color = "RW Forecast")) +
  labs(title = paste("Actual vs Forecast Return","(", ticker,")"),
       x = "Date", y = "% change in Return", color = "Series") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 14, fac
e = "bold"))


print(comparison_plot)



##############################################################################
# Calculate the mean squared error for VAR, AR, and random walk forecast
s
var_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$VAR_For
ecast)^2)
rw_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$RW_Forec
ast)^2)
ar_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$AR_Forec
ast)^2)

# Calculate the Root Mean Squared Error (RMSE) for VAR, AR, and random w
alk forecasts
var_rmse <- sqrt(var_mse)
rw_rmse <- sqrt(rw_mse)
ar_rmse <- sqrt(ar_mse)

# Calculate the Mean Absolute Error (MAE) for VAR, AR, and random walk f
orecasts
var_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$VAR_
Forecast))
rw_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$RW_Fo
recast))
ar_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$AR_Fo
recast))


# Create a data frame to store the metrics
metrics_df <- data.frame(
  Model = c("VAR Model", "Random Walk", "AR Model"),
  MSE = c(var_mse, rw_mse, ar_mse),
  RMSE = c(var_rmse, rw_rmse, ar_rmse),
  MAE = c(var_mae, rw_mae, ar_mae)
)

# Print the metrics data frame
cat("\nMetrics:", ticker, "For Adj Close, Sentiment and SVI", "\n")



##############################################################################
#########           Adj Close and Sentiment            #########
##############################################################################
# optimal lag selection and VAR-framework

# Define y1,t as the logged values of Sentiment score
```

```r
y1t <- in_sample_df$Adj_pct_change

# Define y2,t as the logged values of pct change
y2t <- in_sample_df$Sentiment_pct_change

# Construct y matrix
y <- matrix( c(y1t, y2t), ncol=2 )
colnames(y) <- c("Adj_pct_change", "Sentiment_pct_change")

# Optimal lags
opt_lag <- VARselect(y, lag.max=20)
print(opt_lag)

paste0("HQIC: VAR(", opt_lag$selection[2], ")")

# Fit VAR model
model_fit <- VAR(y, p=opt_lag$selection[2])
summary(model_fit)

##############################################################################
# Perform the Granger causality tests

granger_test1 <- causality(model_fit, cause = c("Sentiment_pct_change"))
$Granger
granger_test2 <- causality(model_fit, cause = c("Adj_pct_change"))$Grang
er


# Prepare data for the data frame
cause_vars <- c("Sentiment_pct_change", "Adj_pct_change")
effect_vars <- c("Adj_pct_change", "Sentiment_pct_change")
test_results <- list(granger_test1, granger_test2)

# Function to extract p-values from the causality test results
get_test_info <- function(test_result) {
  p_value <- format(round(test_result$p.value, 4), nsmall = 4)
  return(p_value)
}

# Function to assign significance based on p-value
assign_significance <- function(p) {
  p = as.numeric(p)
  if (p <= 0.01) {
    return("***")
  } else if (p <= 0.05) {
    return("**")
  } else if (p <= 0.1) {
    return("*")
  } else {
    return("")
  }
}

# Gather the p-values
p_values <- sapply(test_results, get_test_info)

# Assign significance levels based on p-values
significance <- sapply(p_values, assign_significance)

# Create a data frame to store the p-values and significance
result_df <- data.frame(
  Cause = cause_vars,
```

```
  Effect = effect_vars,
  P_Values = p_values,
  Significance = significance
)


# Print the title
cat("\nGranger Causality Test Results for:", ticker, "\n")

# Print the results
print(result_df,row.names = FALSE)



# Print the legend for significance levels
cat("\np <= 0.1: *, p <= 0.05: **, p <= 0.01: ***")



##############################################################################
# Test for serial correlation of residuals using the Durbin-Watson stati
stic

# Calculate the residuals from the fitted VAR model
residuals <- residuals(model_fit)

# Calculate the Durbin Watson statistic for each variable

dwt <- dwtest(lm(residuals ~ 1))$statistic

cat("Durbin-Watson statistic for", colnames(residuals), ":", dwt , "(",
ticker, ")", "\n") # A value of close to 2 indicate no serial correlatio
n

##############################################################################
# VAR forecast
n_steps <- 1
y_t_plus_1_forecast <- predict(model_fit, n.ahead = n_steps, ci = 0.95)

# Initialize variables to store the forecast values
forecast_values <- numeric(length(out_of_sample_df$Adj_pct_change))

for (i in 2:length(out_of_sample_df$Adj_pct_change)) {
  # Combine in-sample and out-of-sample data up to the current point
  combined_data <- rbind(in_sample_df[, c("Adj_pct_change",
"Sentiment_pct_change")], out_of_sample_df[1:(i - 1),c("Adj_pct_change",
"Sentiment_pct_change")])

  # Fit the VAR model to the combined data
  var_model <- VAR(combined_data, p = opt_lag$selection[2])

  # Generate a one-step-ahead forecast
  forecast_result <- predict(var_model, n.ahead = n_steps)
  forecast_values[i] <- forecast_result$fcst[[1]][1]
}

##############################################################################
# Combine the actual and forecast values in a data frame
comparison_df <- data.frame(Date = out_of_sample_df$Date,
                            Actual = out_of_sample_df$Adj_pct_change,
                            VAR_Forecast = forecast_values,
                            AR_Forecast = ar_forecast_values,
                            RW_Forecast = rw_forecast_values)
```

```r
comparison_df$Date <- as.Date(comparison_df$Date)

# Plot the actual and forecast values
comparison_plot <- ggplot(comparison_df, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_line(aes(y = VAR_Forecast, color = "VAR Forecast")) +
  geom_line(aes(y = AR_Forecast, color = "AR Forecast")) +
  geom_line(aes(y = RW_Forecast, color = "RW Forecast")) +
  labs(title = paste("Actual vs Forecast Return","(", ticker,")"),
       x = "Date", y = "% change in Return", color = "Series") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 14,
face = "bold"))


print(comparison_plot)


################################################################################
# Calculate the mean squared error for VAR, AR, and random walk forecast
s
var_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$VAR_For
ecast)^2)
rw_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$RW_Forec
ast)^2)
ar_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$AR_Forec
ast)^2)

# Calculate the Root Mean Squared Error (RMSE) for VAR, AR, and random w
alk forecasts
var_rmse <- sqrt(var_mse)
rw_rmse <- sqrt(rw_mse)
ar_rmse <- sqrt(ar_mse)

# Calculate the Mean Absolute Error (MAE) for VAR, AR, and random walk f
orecasts
var_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$VAR_
Forecast))
rw_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$RW_Fo
recast))
ar_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$AR_Fo
recast))


# Create a data frame to store the metrics
metrics_df <- data.frame(
  Model = c("VAR Model", "Random Walk", "AR Model"),
  MSE = c(var_mse, rw_mse, ar_mse),
  RMSE = c(var_rmse, rw_rmse, ar_rmse),
  MAE = c(var_mae, rw_mae, ar_mae)
)

# Print the metrics data frame
cat("\nMetrics:", ticker, "For Adj Close and Sentiment", "\n")

################################################################################
#########          Adj Close and Google Trends          #########
################################################################################

# Define y1,t as the logged values of Adj_pct_change
y1t <- in_sample_df$Adj_pct_change
```

```r
# Define y3,t as the logged values of SVI_pct_change
y3t <- in_sample_df$SVI_pct_change

# Construct y matrix
y <- matrix( c(y1t, y3t), ncol=2 )
colnames(y) <- c("Adj_pct_change", "SVI_pct_change")

# Optimal lags
opt_lag <- VARselect(y, lag.max=20)
print(opt_lag)

paste0("AIC: VAR(", opt_lag$selection[1], ")")

# Fit VAR model
model_fit <- VAR(y, p=opt_lag$selection[1])
summary(model_fit)

##############################################################################
# Perform the Granger causality tests
granger_test1 <- causality(model_fit, cause = c("SVI_pct_change"))$Grang
er
granger_test2 <- causality(model_fit, cause = c("Adj_pct_change"))$Grang
er

# Prepare data for the data frame
cause_vars <- c("SVI_pct_change","Adj_pct_change")
effect_vars <- c("Adj_pct_change", "SVI_pct_change")
test_results <- list(granger_test1, granger_test2)

# Function to extract p-values from the causality test results
get_test_info <- function(test_result) {
  p_value <- format(round(test_result$p.value, 4), nsmall = 4)
  return(p_value)
}

# Function to assign significance based on p-value
assign_significance <- function(p) {
  p = as.numeric(p)
  if (p <= 0.01) {
    return("***")
  } else if (p <= 0.05) {
    return("**")
  } else if (p <= 0.1) {
    return("*")
  } else {
    return("")
  }
}

# Gather the p-values
p_values <- sapply(test_results, get_test_info)

# Assign significance levels based on p-values
significance <- sapply(p_values, assign_significance)

# Create a data frame to store the p-values and significance
result_df <- data.frame(
  Cause = cause_vars,
  Effect = effect_vars,
  P_Values = p_values,
  Significance = significance
```

```
)

# Print the title
cat("\nGranger Causality Test Results for:", ticker, "\n")

##
## Granger Causality Test Results for: AAPL

# Print the results
print(result_df,row.names = FALSE)

##           Cause         Effect P_Values Significance
##  SVI_pct_change Adj_pct_change   0.0931            *
##  Adj_pct_change SVI_pct_change   0.8915

#############################################################################
# Test for serial correlation of residuals using the Durbin-Watson stati
stic

# Calculate the residuals from the fitted VAR model
residuals <- residuals(model_fit)

# Calculate the Durbin Watson statistic for each variable

dwt <- dwtest(lm(residuals ~ 1))$statistic

cat("Durbin-Watson statistic for", colnames(residuals), ":", dwt , "\n")
# A value of close to 2 indicate no serial correlation

#############################################################################
# VAR forecast
n_steps <- 1
y_t_plus_1_forecast <- predict(model_fit, n.ahead = n_steps, ci = 0.95)

# Initialize variables to store the forecast values
forecast_values <- numeric(length(out_of_sample_df$Adj_pct_change))

for (i in 2:length(out_of_sample_df$Adj_pct_change)) {
  # Combine in-sample and out-of-sample data up to the current point
  combined_data <- rbind(in_sample_df[, c("Adj_pct_change", "SVI_pct_cha
nge")], out_of_sample_df[1:(i - 1), c("Adj_pct_change", "SVI_pct_change"
)])

  # Fit the VAR model to the combined data
  var_model <- VAR(combined_data, p = opt_lag$selection[1])

  # Generate a one-step-ahead forecast
  forecast_result <- predict(var_model, n.ahead = n_steps)
  forecast_values[i] <- forecast_result$fcst[[1]][1]
}

#############################################################################
# Combine the actual and forecast values in a data frame
comparison_df <- data.frame(Date = out_of_sample_df$Date,
                            Actual = out_of_sample_df$Adj_pct_change,
                            VAR_Forecast = forecast_values,
                            AR_Forecast = ar_forecast_values,
                            RW_Forecast = rw_forecast_values)

comparison_df$Date <- as.Date(comparison_df$Date)

# Plot the actual and forecast values
```

```r
comparison_plot <- ggplot(comparison_df, aes(x = Date)) +
  geom_line(aes(y = Actual, color = "Actual")) +
  geom_line(aes(y = VAR_Forecast, color = "VAR Forecast")) +
  geom_line(aes(y = AR_Forecast, color = "AR Forecast")) +
  geom_line(aes(y = RW_Forecast, color = "RW Forecast")) +
  labs(title = paste("Actual vs Forecast Return","(", ticker,")"),
       x = "Date", y = "% change in Return", color = "Series") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, vjust = 1, size = 14, fac
e = "bold"))


print(comparison_plot)



##############################################################################
# Calculate the mean squared error for VAR, AR, and random walk forecast
s
var_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$VAR_For
ecast)^2)
rw_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$RW_Forec
ast)^2)
ar_mse <- mean((out_of_sample_df$Adj_pct_change - comparison_df$AR_Forec
ast)^2)

# Calculate the Root Mean Squared Error (RMSE) for VAR, AR, and random w
alk forecasts
var_rmse <- sqrt(var_mse)
rw_rmse <- sqrt(rw_mse)
ar_rmse <- sqrt(ar_mse)

# Calculate the Mean Absolute Error (MAE) for VAR, AR, and random walk f
orecasts
var_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$VAR_
Forecast))
rw_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$RW_Fo
recast))
ar_mae <- mean(abs(out_of_sample_df$Adj_pct_change - comparison_df$AR_Fo
recast))


# Create a data frame to store the metrics
metrics_df <- data.frame(
  Model = c("VAR Model", "Random Walk", "AR Model"),
  MSE = c(var_mse, rw_mse, ar_mse),
  RMSE = c(var_rmse, rw_rmse, ar_rmse),
  MAE = c(var_mae, rw_mae, ar_mae)
)

# Print the metrics data frame
cat("\nMetrics:", ticker, "For Adj Close and SVI", "\n")
```