Ca' Foscari
University
of Venice

Master's Degree Programme in
Data Analytics for Business and Society

Time Series, Stocks and News: A Multi-Model Forecasting Approach for the Infotech Sector

Supervisor
Prosdocimi Ilaria

Graduand
Trevisan Manuel 886937

Academic Year
2024-2025

Abstract

This thesis explores the efficiency of different forecasting techniques in predicting financial data in the infotech industry by leveraging five different key assets: NVDIA, Apple, Broadcom and Microsoft stocks, as well as an ETF tracking the whole sector (IUIT.L). Trying to address the sector's strong trend and fast shifts, the analysis includes classical time series models (such as ETS, ARIMA and GARCH), newer trend-seasonality models (such as Prophet) and more advanced machine learning approaches (such as XGBoost). Including a dataset containing over 62,000 news sourced from the New York Times, this analysis tries to address whether a multi-source model can improve forecast accuracy. Methods include sentiment extraction to generate features that can be included in the forecasting process. A full analysis was dedicated to understanding the intricate relationship between sentiment scores and asset's returns. Results highlight a strong growing trend characterizing the sector, stopping during the first period of 2025, while unpredictability increased during the same time window. To address drastic changes in direction, the changing volatility underlined during GARCH implementation, and short term movements, XGBoost was found to be the best model for the scope. When implementing Sentiments, a complex non-monotonic relationship emerged. By injecting different scenarios in the Sentiment data, context specific influences were highlighted. The inclusion of sentiment scores between the model's features (even if theoretically promising) did not outperform the financial data alone. This outcome can be linked to the increased variability brought by the sentiment scoring algorithm and a data source for news that is too generic for the scope.

Contents

List of Figures

List of Tables

Introduction

Accurately forecasting stock prices remains a major challenge in modern finance. The stock market plays a central role not only for investors, who rely on it to manage risk and seize opportunities, but also for policymakers, who depend on reliable predictions to develop effective and responsive economic strategies (Li et al., 2024). However, the inherent unpredictability of financial markets (marked by rapid volatility and complex, non-linear trends) makes forecasting particularly difficult. This challenge is especially pronounced in the U.S. information technology sector, one of the most volatile segments of the market. This volatility is driven by the sector's dynamic nature, characterized by both fast-growing and well-established tech giants, as well as ongoing research and constant innovation. With artificial intelligence emerging as a central focus, recent studies suggest that the intensifying pace of technological advancement has "ushered in an era of heightened competition that is significantly influencing the volatility of technology stocks" (Chessum, 2024). This highlights the importance of studying and understanding which forecasting methods can give the most robust predictions in such rapid context.

This study focuses on five key assets within the information technology sector: the stocks of NVIDIA (NVDA), Apple (AAPL), Broadcom (AVGO), and Microsoft (MSFT), along with an ETF tracking the S&P 500 Information Technology sector (IUIT.L). These assets were chosen according to their percentage contribution in the sector index, taking the four with highest weights. The inclusion of the index itself serves as an aggregate measure, providing a larger view of the sectors trends. As a result, this analysis focuses on market leaders: as displayed in FinanceCharts.com (2025), NVIDIA, Microsoft, and Apple in mid 2025 were declared the top three companies worldwide by market capitalization, with Broadcom ranking only eighth. By including both the broader sector index and some individual tech giants in the examination, the study captures a comprehensive view of the IT market. This focus is particularly relevant given the rapid technological developments, especially in artificial intelligence, which have led changing trends and sharp movements in asset prices. As Chessum (2024) explained in his report, innovation and news-driven developments have triggered significant volatility in large-cap tech valuations,

underscoring the sectors sensitivity to change.

In tackling these forecasting problems, this study will examine both classical time-series models and modern machine learning approaches. Traditional methods like exponential smoothing, ARIMA (autoregressive integrated moving average), and GARCH (generalized autoregressive conditional heteroscedasticity) have long been used in finance. For example, ARIMA is a popular time-series forecasting tool in this domain (Deepan et al., 2024), while GARCH models are widely applied to capture time-varying volatility in asset prices (Marisetty, 2024). These models are valued for their mathematical tractability and ability to model trends and volatility, but they assume relatively simple (often linear) relationships in the data. In contrast, machine learning techniques have shown strong performance on financial data (Sunki et al., 2024). In this thesis a variety of methods across this spectrum will be compared, from statistical time-series models to advanced neural predictors.

Beyond purely numerical price data, this research also incorporates textual sentiment information from financial news. The rationale is that news sentiments often contain predictive clues not captured in past prices alone. Prior work has found that blending market data with news sentiment can improve forecasts (Fu and Zhang, 2024). In fact, recent forecasting models explicitly combine "sentiment analysis with technical indicators to improve the precision of stock market prediction" (Agrawal et al., 2024). In practice, this means transforming financial news text into sentiment scores or indices, and using these alongside traditional inputs. By bringing these sources together, researchers have observed that the model "provides a more comprehensive approach to understanding market sentiments influence on stock prices" (Agrawal et al., 2024).

Moreover, this study investigates how different sentiment scenarios in news data can influence financial asset behaviour. There is substantial empirical evidence supporting the idea that sentiment can significantly impact financial predictions. For example, studies such as the ones conducted by Mohan et al. (2019) and Gupta and Chen (2020) demonstrate a measurable relationship between sentiment (whether extracted from news or social media) and asset price movements. To explore this effect, sentiment data were manipulated manually to simulate various hypothetical scenarios. These datasets with injected scenarios were then used to

generate forecasts, allowing a comparative analysis of how different sentiment configurations might translate into movements of asset prices and returns.

Accordingly, this thesis conducts a comprehensive comparison of forecasting approaches for the IT sector. The goal is to determine whether a hybrid, multi-source approach yields lower forecast error than using prices alone. Improved accuracy would have practical significance: in fast-paced tech markets, even small gains in prediction can aid portfolio managers, traders, and policymakers in making more informed decisions. Ultimately, this research aims to shed light on how diverse data sources can impact forecasts in a dynamic, information-sensitive sector. Moreover, this study tries to address the sensitivity of financial instruments to shifts in sentiment and to better understand the relative impact of different types of news (e.g., political, business, technological) on final price dynamics.

Chapter 1

Data Collection

One of the main challenges in conducting this type of analysis is finding a clean and reliable data source to use during the research. Since polished and readily accessible datasets are not easy to find and are often not up to date, using APIs became a necessary solution. All the APIs used in this study are freely accessible and offer intuitive interfaces in Python, which was the programming language selected for the data collection process.

The financial data were collected using Yahoo Finance (Yahoo Finance, 2025). For each asset, a data table was created to store all records from January 2021 to the end of February 2025, which marks the starting point of the analysis. Each row in the table represents a trading day. The initial features included the trading Date, Opening and Closing prices, the Highest and Lowest prices reached during the day, and the traded Volume. Additional features were subsequently computed and added to the dataset, resulting in extra columns. In particular, a sequential trading day index was introduced (serving as a continuous indicator for days, since the Date field omits weekends and holidays), along with Returns and log(Returns) (calculated as the difference in Closing prices and the logarithm of Closing prices, respectively; the second one is referenced in data as Log_returns), and a Volatility measure (computed as the standard deviation of log(Returns) over the previous 21 trading days). In Figure 1 can be seen a visual representation of the data, taking the ETF as a reference.

Regarding news data, most APIs are either expensive or limited to recent publications. However, this analysis aimed to be fully replicable for any researcher and applicable to any asset. To meet this requirement, the New York Times API was chosen. It is free, user-friendly, and allows bulk retrieval of data extending back several years, more so than many other alternatives. Using this tool, another table was created to store all news articles published during the analysis period, filtering only those potentially related to financial markets. Specifically, all articles categorized

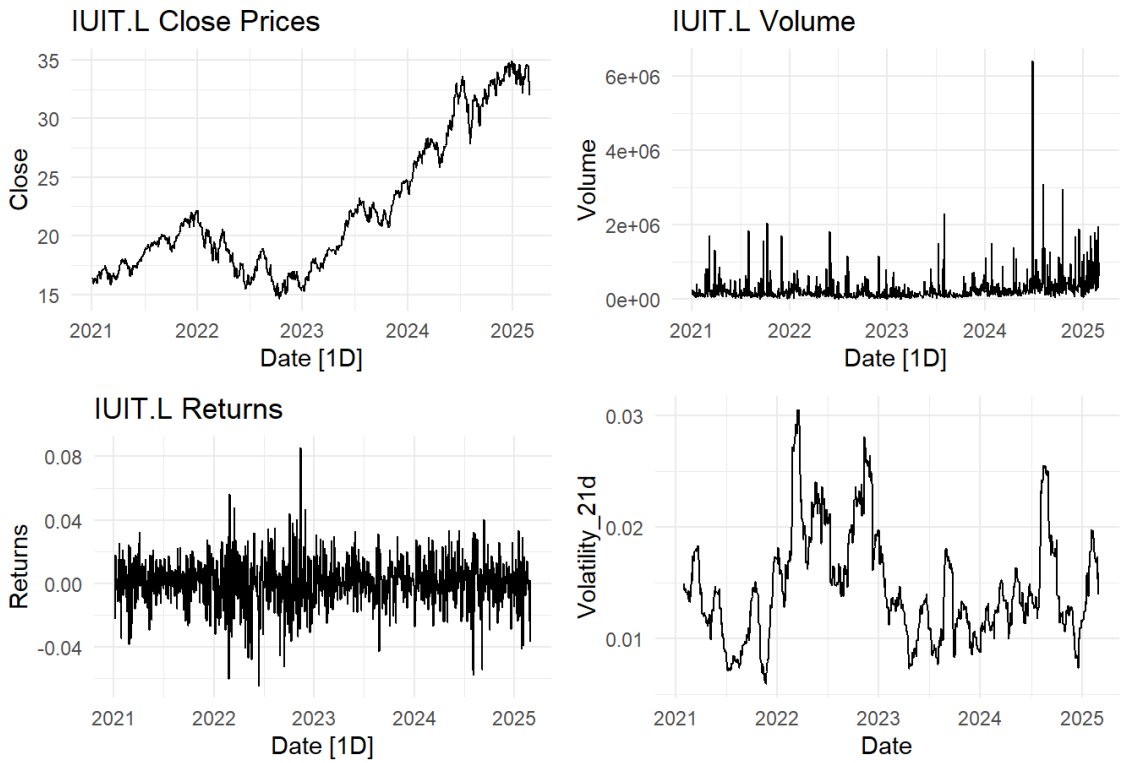

Figure 1: IUIT.L Index Analysis

under "Politics," "Business", and "Technology" were collected. Additionally, all articles mentioning the selected firms were manually filtered and categorized under "Companies". The resulting table includes over 62,000 records, with each article entry containing the Date, Category, Text, Title, and Keywords, but only the first tree were used.

Chapter 2

Financial Methods

The main techniques used during the analysis are time-series decomposition, Exponential Smoothing, ARIMA, GARCH, Prophet, XGBoost, Random Forest, LSTM, as well as some different kinds of Cross Validation for time series. Most of the analysis was done using R, but some parts like the data collection, the Random Forest and the LSTM was done using Python programming language.

2.1 Time Series Decomposition

Before conducting any in-depth analysis, it is essential to understand the context in which the study is being carried out. In time series analysis, this is achieved in a different way if compared to more traditional research settings. Typically, the first step of an analysis involves computing summary statistics, such as the mean, standard deviation, and other descriptive metrics. However, applying this approach to time series can lead to misleading conclusions (Chatfield and Xing, 2019). Therefore, after constructing a basic time plot, the initial step in this study was to identify the main components influencing the evolution of the asset prices. This was achieved using time-series decomposition.

The primary objective of time series decomposition is to break down a series into interpretable components that highlight its key characteristics. As noted by Chatfield and Xing (2019), the standard decomposition separates a time series into three main elements: Trend, Seasonal and/or Cyclic, and Random components. The Trend refers to "the long-term change in the mean value of the series over time" (Chatfield and Xing, 2019). The Seasonal component captures"regular, repeating patterns or cycles that occur over a fixed period". Closely related is the Cyclic component, which also exhibits repeating patterns but lacks fixed periodicity, typically aligning with broader business or economic cycles. Finally, the Random component (also known as Noise or Remainder) includes all unpredictable fluctuations not

explained by the preceding components. The output of this decomposition process should be similar to what is illustrated in Figure 2.

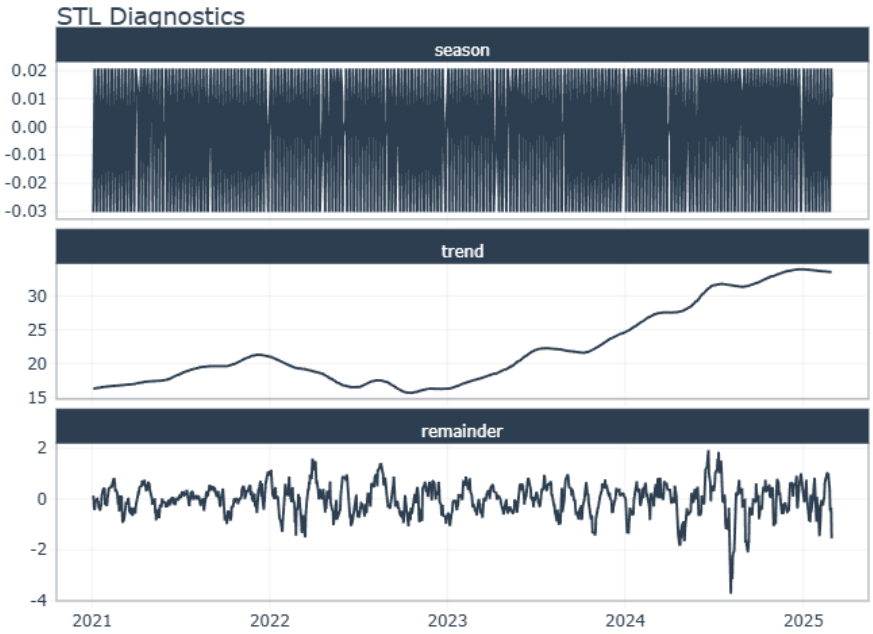

Figure 2: IUIT.L Index Decomposition

In this study, a Seasonal-Trend decomposition using Loess (STL) was employed. As described by Shunway and Stoffer (2019), STL applies Locally Estimated Scatterplot Smoothing (LOESS) to extract the trend, seasonal, and remainder components. Compared to more conventional decomposition methods that rely on moving averages (MA), STL is better suited to this context, particularly in its treatment of the seasonal component.

Since the data involve financial markets, which are not continuously open, and given the rapidly changing market dynamics, a more flexible and adaptive model was necessary. STLs window-based approach allows the seasonal effect (St) to be approximately equal to the seasonal effect from the previous cycle (e.g., $St \approx St-12$ for monthly data). Requiring it to be exactly equal, which occurs using the MA approach specifying a periodic seasonal component, would have brought different results. This flexibility enables STL to more accurately capture slowly evolving seasonal patterns over time (Shunway and Stoffer, 2019).

Another important first exploratory analysis is the Auto Correlation Function (ACF) and Partial Auto Correlation Function (PACF) plot, also called Correlogram.

The ACF is defined as

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}},$$

and measures the linear correlation between observations at different distances apart, known as lags (Shunway and Stoffer, 2019). It leverages the autocovariance function, which mathematically is defined as

$$\gamma_X(s,t) = \text{cov}(X_s, X_t) = E[(X_s - \mu_s)(X_t - \mu_t)]$$

where X_s and X_t are the values of the time series at times s and t respectively, μ_s and μ_t are the expected values (means) at times s and t, $E[\,]$ denotes the expected value operator. It measures the "linear dependence between two points on the same series observed at different times" (Shunway and Stoffer, 2019). The PACF instead is defined as

$$\phi_{11} = \text{corr}(x_1, x_0) = \rho(1)$$

and

$$\phi_{hh} = \text{corr}(x_h - \hat{x}_h, x_0 - \hat{x}_0), \quad h \geq 2,$$

where $\hat{x}_h$ is the regression of x_h on $\{\ x_1, x_2, ..., x_{h-1}\}$ and $\hat{x}_0$ is the regression of x_0 on $\{x_1, x_2, ..., x_{h-1}\}$, and measures the "correlation between x_{t+h} and x_t with the linear dependence of everything between them, namely $\{x_{t+1}, ..., x_{t+h-1}\}$, on each, removed" (Shunway and Stoffer, 2019). In this analysis they where useful to identify some parameters for models like ARIMA and GARCH. Without giving much anticipations, looking at the Correlogram we can understand AR and MA orders: if some lags in the ACF shows a clear cut-off, this indicates an MA model; if the clear cut-off is present in the PACF, this indicates an AR model; if the cut-off can be observed in both the ACF and PACF, this suggests an ARMA model.

2.2 Exponential Smoothing

The first model applied after understanding the scenario was the Simple and Double ETS. Exponential Smoothing (ETS) refers to a "general class of forecasting procedures that rely on simple update equations to calculate forecasts" (Chatfield and Xing, 2019). The core idea is to give more weight to recent observations and less

weight to older observations in the forecast. This is achieved by using geometrically decreasing weights for past observations.

It depends on tree main concepts: the weighted averages, the recursive updates and the smoothing parameters. ETS models consists on the calculation of a smoothed value as a weighted average of past observations, with weights that decline exponentially. This differs from simple moving averages, as they give equal weight to all observations within the window. It can be used to forecast h steps ahead by simply recursively updating the next step using the latest observation and the previous forecast. These methods involve smoothing parameters, typically denoted by α (alpha), γ (gamma), and δ (delta).

To choose which parameters to include in the model, the Correlogram and the STL can be helpful. If the series is not time dependent (so there are no signals of trend or seasonality), a Simple ETS can be applied. This is the more basic model among all the ETS, and it's based on a single smoothing parameter (α). If the trend component is particularly significant, a Double ETS is more appropriate. It's based on two smoothing parameters (α and γ) and it's optimal for non seasonal or deseasonalized trends with leaner growth. Since the growing trend realistically can't always go on exponentially forever, an evolution of the last one is the double dumped ETS, which introduces a new parameter (ϕ) to damp the estimated trend or growth rate towards zero over time.

There are also some other ETSs that include more parameters (like the Triple additive and Triple Multiplicative), but they were not leveraged during this analysis since no assets were showing a seasonality great enough to justify their use. Recalling Figure 2, can be seen what a time series with trend looks like. And by taking a closer look at the y axis of the seasonal component, it's clear that the magnitude is not great enough to be significant.

2.3 ARIMA

The next model that was applied is the ARIMA model. ARIMA stands for Autoregressive Integrated Moving Average, which is a systematic model used for time series forecasting and time-correlated modelling. It's particularly useful when the series are non-stationary (Chatfield and Xing, 2019).

An ARIMA(p,d,q) model is composed of three main parts:

- AR (Autoregressive) component (p): An AR(p) model represents a time series where the current value, X_t, is a linear function of its p past values $(X_{t-1}, X_{t-2}, \ldots, X_{t-p})$ and a white noise error term (Z_t). This implies that forecasting is possible because current values depend on past values (Woodward et al., 2017).
- I (Integrated) component (d): This component addresses non-stationarity in the mean by using differencing. A time series is differenced d times until it becomes stationary (Shunway and Stoffer, 2019) .
- MA (Moving Average) component (q): An MA(q) model expresses the current value of the series as a linear combination of current and past white noise (error) terms (shocks) up to q lags(Chatfield and Xing, 2019). The model can be written as $X_t = \theta(B)Z_t$, where $\theta(B)$ is the moving average operator (Shunway and Stoffer, 2019).

The general function is

$$\phi(B)(1-B)^d X_t = \theta(B)\varepsilon_t$$

where:

- $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ (AR polynomial)
- $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 - \cdots + \theta_q B^q$ (MA polynomial)
- $(1-B)^d$ represents the differencing operator
- B is the backshift operator: $BX_t = X_{t-1}$

To determine the appropriate (p, d, q) parameters for the ARIMA model, the auto.arima() function from the "forecast" R package was used. By fitting the model using different optimizers, an initial estimate of the optimal parameters can be obtained. However, these suggestions are not definitive, and further refinements can be made. Therefore, a few custom models were also fitted starting from the suggestions provided by auto.arima(). To decide on the AR and MA parts, different correlograms can be used. In particular, the PACF of the series and the ACF of the residuals.

In Figure 3 the ACF and PACF of the Close prices of the IUIT.L title are shown. Here, auto.arima() suggested a (0,1,0) model, but by looking at the PACF plot, it's quite clear that a $p = 1$ could also be appropriate, since the first lag shows a strong

break-out.

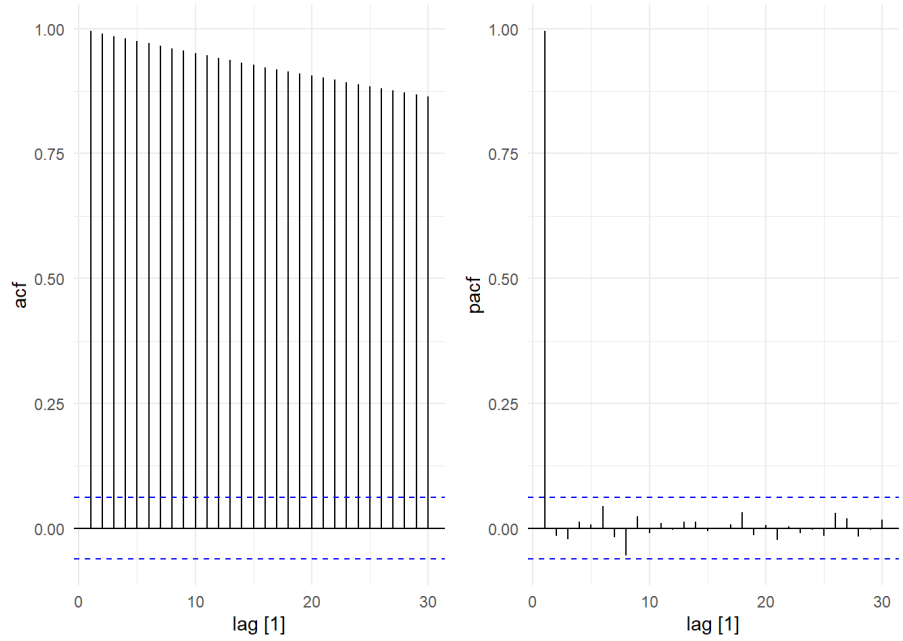

Figure 3: IUIT.L Correlogram for Close Prices

As for the q parameter, this can be evaluated by looking at the ACF of the residuals, shown in Figure 4. In lags 20 and 25 can be observed a breakout, indicating that there are some correlations that would be omitted by using a (0,1,0) model. Using a $q = 25$ instead should allow the model to capture the correlations at both lags.

A basic ARIMA(1,1,1) model was also fitted to every series, as it is the standard default used by most programs.

2.4 Prophet

Another model tested in this research was Prophet, described by Taylor and Letham (2018) as a "time series forecasting model designed to handle the common features of business time series". Is based on an additive model in which non-linear trends are fit with yearly, weekly, and daily seasonality, as well as holiday effects. Its mathematical formulation can be expressed as:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

where:

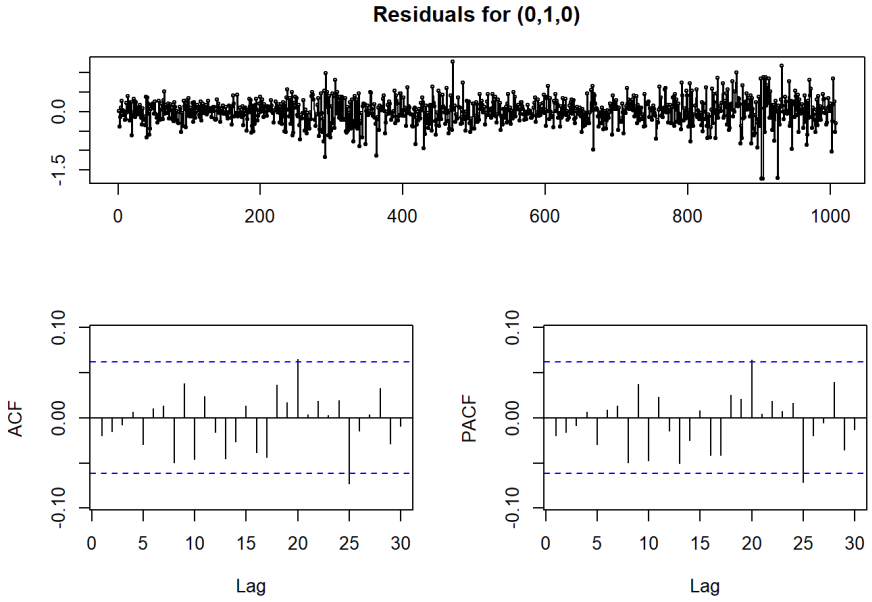

Figure 4: ARIMA(0,1,0) Correlogram for IUIT.L Residuals

- $g(t)$ represents the trend component, which can be modelled using piecewise linear or logistic growth,
- $s(t)$ captures seasonal effects, including yearly, weekly, daily patterns,
- $h(t)$ represents holiday effects,
- $\varepsilon_{\mathbf{t}}$ is the error term

In their paper, Taylor and Letham (2018) lists also some core mechanisms on which the model is founded. The first one is the Trend Modelling: Prophet uses piecewise linear or logistic growth curves to capture non-linear trends in the data. It automatically detects change points where the trend shifts, allowing for flexible modelling of complex growth patterns. Then there is the Seasonality: The model leverages Fourier series to model seasonal patterns, making it particularly effective for data with strong seasonal effects. It can handle different seasonalities simultaneously (e.g., yearly, weekly, and daily). On top of that, it includes also Holiday Effects: Prophet incorporates holiday effects by allowing users to specify custom holidays and their impact windows, which is particularly valuable for business forecasting. The last key mechanisms is the Uncertainty Quantification, by which the model provides uncertainty intervals around its predictions using a Bayesian approach with Monte Carlo sampling.

One key advantage of Prophet, and a major reason for its inclusion in this study,

is its ability to handle missing dates, a frequent occurrence in financial data due to market closures. Unlike other models that require continuous trading days or imputation of missing values, Prophet can work directly with irregular time series.

In addition, Prophet serves as a valuable baseline model for comparison against more complex approaches. It is often recommended for situations involving multiple time series and limited expertise in advanced time-series modelling. Since the aim of this research is to remain accessible and replicable by individuals with a general interest in financial markets, including a model that is both user-friendly and effective adds meaningful value to the analysis.

2.5 GARCH

During the implementation of the previous models, volatility resulted to be an important factor influencing the analysis. Therefore, another model was exploited to study also this aspect of the series: GARCH. The acronym stands for Generalized Autoregressive Conditional Heteroscedasticity, and it represents a technique that was developed appositely to model changes in the volatility of a time series, particularly in financial data. As explained by Shunway and Stoffer (2019), this model was created to address the violation of a core assumption made in traditional ARMA models (constant variance) by allowing the conditional variance to change over different periods of time.

As mentioned by Chatfield and Xing (2019), GARCH models are particularly effective at capturing volatility clustering, a common characteristic in financial data where periods of high volatility tend to be grouped together. And this characteristic can already be observed by looking at the bottom-right corner of Figure 1, where an alternation of low and high volatility periods can be observed.

Other two factors that need to be checked when working with GARCH models are stationarity and the presence of ARCH effects. The fist one was addressed by working with log(returns) instead of close prices. The second was tested for each individual series by using the ArchTest R function from the FinTS package.

Different types of GARCH models were fitted, starting from the basic one that assumes normal distribution of Returns. The parameters were estimated using the "garch" R function from the "tseries" package, while the actual models were fitted

using the "rugarch" library. However, using a normal distribution was proven to be not a good representation of reality. By looking at Figure 5 can be seen that a skewed student-t (sstd) distribution is better overlapping with the real scenario.

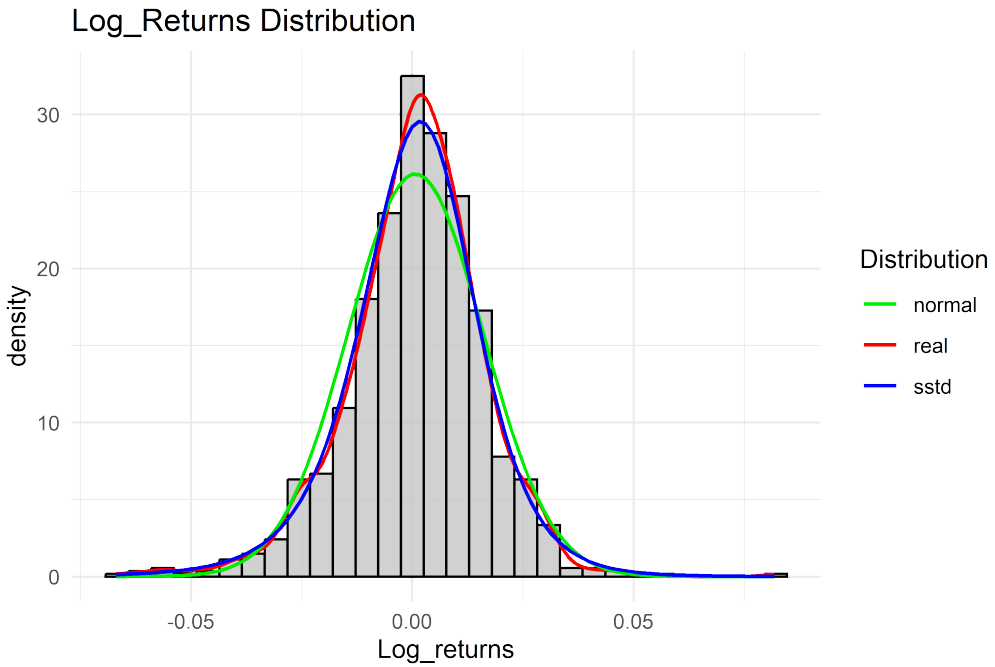

Figure 5: Distribution of Log_Returns

Another way in which the basic GARCH model was tried to improve is by implementing asymmetric responses to returns. One limitation of the basic GARCH model is that it assumes that positive and negative returns have the same effect on volatility. In reality, especially in stocks, this is not true. Positive and negative Returns have different impacts on stocks volatility, and to address this, E-Garch and GJR-Garch models were implemented. A visual representation of the influence of Returns on volatility for the different models applied on the NVDIA stock is displayed in Figure 6.

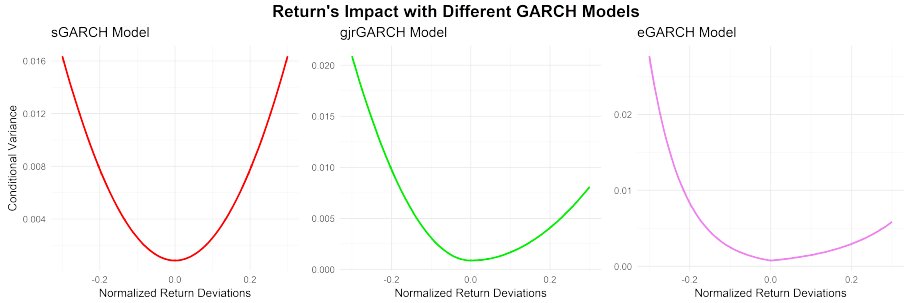

Figure 6: Returns Impact on Volatility with different Garch models

2.6 XGBoost

2.6.1 Theoretical Review

The next model that was applied is the Extreme Gradient Boosting (XGBoost). XGBoost is a scalable, efficient, and high-performance implementation of gradient tree boosting. Its widespread adoption is largely due to its empirical success in machine learning competitions and practical applications across various domains. This model distinguishes itself through algorithmic and system-level innovations that yield both predictive accuracy and computational efficiency (Chen and Guestrin, 2016).

It functions by fitting additive tree models. The core idea is to train the model in an additive manner, where for each iteration a new function (specifically a regression tree) is chosen to most improve a regularized objective and greedily added to the model. As Chen and Guestrin (2016) describe, XGBoost performs "additive optimization in functional space," fitting each new tree to the gradient (first-order derivatives) and curvature (second-order derivatives) of the loss relative to the predictions of the current model. During this study, the one used is the squared error loss ($\ell(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$).

In the paper written by Chen and Guestrin (2016) are also listed some key fundamentals on which the model is based, and most of them can be associated with the Bias-Variance trade-off handling. The first one is the already explained sequential construction, which enables the scaling of each newly added tree by a factor (also called learning rate) in order to reduce overfitting and control the model's complexity.

The second is the regularization. As has already been said, the model runs by minimizing a regularized objective, which can be seen here:

$$\mathcal{L} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

The first part of the equation is the chosen loss function. The second term ($\Omega(f_k)$) serves as a penalization of the complexity of the tree, in order to avoid overfitting. If an overly complex tree is added, the loss reduction must be great enough to overcome

its cost. Another way of controlling complexity is by manually setting a maximum depth of each tree, encouraging weak learners that focus on simple patterns.

Once the tree structure is set, a weight (w_j) is assigned to each leaf (j) using the following equation:

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda},$$

where I_j is the set of instances in leaf j, g_i represents the first-order gradients, h_i represents the second-order gradients (Hessian), and λ is a regularization parameter. Manually setting a minimum sum of instance weights (i.e., the Hessians) that must be present in a child node for a split to be allowed during tree construction is another form of structural regularization that prevents overfitting. This use of second-order derivative information allows XGBoost to achieve better approximation of the loss function compared to methods that rely only on first-order gradients.

The last key characteristic of this model is the use of both columns and rows subsampling. For each tree (or even each split), XGBoost can select a random subset of features to consider, similar to Random Forests. In the XGBoost paper (Chen and Guestrin, 2016), it is noted that "using column subsampling prevents overfitting even more so than the traditional row subsampling". This randomness reduces the correlation between trees and further lowers variance.

2.6.2 Practical Implementation

The first step in fitting an XGBoost model is preparing the dataset for training. A key challenge when working with time series data is that XGBoost does not inherently handle sequential data. However, with appropriate feature engineering, it can be adapted to handle also forecasting tasks that require working on time-dependant data. The training dataset was structured as a table in which the first column was occupied by the dependent variable, while all other columns contained explanatory variables. In time series forecasting, it is crucial that all features are either lagged variables or derived solely from past observations. When predicting the value for day n, all input features (excluding the target variable y) must correspond to informations available up to day $n-1$. This ensures that no look-ahead bias is introduced.

The final model was trained to predict Log_Returns. The explanatory variables included:

- Lagged Log_Returns for the previous 1 to 10 days,
- Log_Returns from the 15th, 20th and 30th day before,
- Rolling means over the past 5, 10, 20, and 30 days,
- Two volatility measures (standard deviations over the past 5 and 10 days),
- Temporal indicators such as day of the week, month, and quarter.

Before forecasting, the model was tuned using a technique called grid search. It consists on fitting multiple models with various combinations of hyperparameters and selecting the one that minimized the prediction error. The parameters explored during grid search included:

- The number of boosting rounds,
- Maximum tree depth,
- Learning rate,
- Subsample ratio of features used to build each tree,
- Minimum sum of instance weights (Hessian) required in a child node,
- Subsample ratio of training instances randomly selected to build each tree.

A subsequent challenge involved long-term forecasting. Unlike models such as Exponential Smoothing, where it is possible to directly specify the number of steps ahead, XGBoost does not natively support multi-step forecasting. To address this, a recursive forecasting approach was employed. In recursive multi-step forecasting, the model first predicts day $n + 1$. That forecast is then treated as observed data and appended to the input set (computing also any derived features such as moving averages and standard deviations) to generate a prediction for day $n + 2$. This process is repeated iteratively for h steps into the future.

While effective, this approach introduces a significant drawback: error propagation. Since each prediction is based partly on previous predictions, inaccuracies can accumulate and compound over time, potentially diverging from the real scenario and producing forecasts that deviate substantially from reality in the long run. This problem was particularly accentuated when testing the models directly on Close Prices.

2.7 Models Validation

Each model used in this research was evaluated using several validation approaches. The most straightforward one is what is commonly denoted as traditional validation. It involves splitting the dataset into a training set and test set and then comparing the forecast (generated using the training data) against the actual values in the test data. During this research, the entire dataset was divided by using the first two months of 2025 as test set, with all earlier data treated as training data. This means that all data from January 2025 onward were treated as unknown at the time of forecasting.

However, in time-series analysis model accuracy can be sensitive to specific validation windows. In other words, a model that performs well in one period might not work in a different context. Therefore, evaluating model performance only against the most recent period is often insufficient. For this reason, additional validation techniques were employed on the training set to asses robustness across different widows of times.

When validating a time series model, the usual validations techniques (like basic k-folds validation) cannot be applied. The main issue here is that randomly splitting the data into train and validation set is not possible, since future data cannot be used to forecast past data. In other words, future looking is something to avoid when validating models. Therefore, time-based validation strategies that preserve temporal order are required.

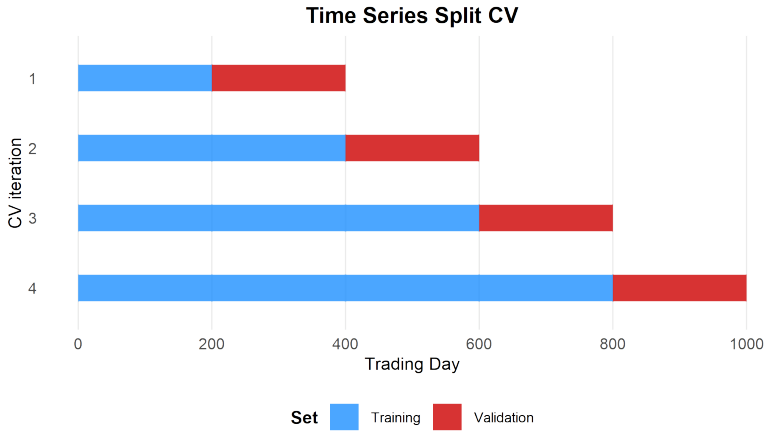

Figure 7: Split Cross Validation Iterations

The first method used is the Split Cross Validation, also called k-folds for time series. In this method the dataset is divided into k sequential parts (folds). The fist fold is used to predict the second, which serves as the validation set. Then, both the first and second folds are combined and used to predict the third one, and so on. This method progressively builds the training set, ensuring that no future data are used in model fitting. A visualization of its functioning is shown in Figure 7.

The next method applied is the Blocked Cross Validation. It operates similarly but with a different objective. In each iteration, only one fold (e.g., fold 1) is used as a training set to predict h steps ahead in the next fold(e.g., fold 2). The model is then re-trained on fold 2 to predict h steps into fold 3, and so on until the end of the training set. This approach is particularly useful when the focus is on multi-step-ahead forecast. An illustration of this method can be seen in Figure 8.

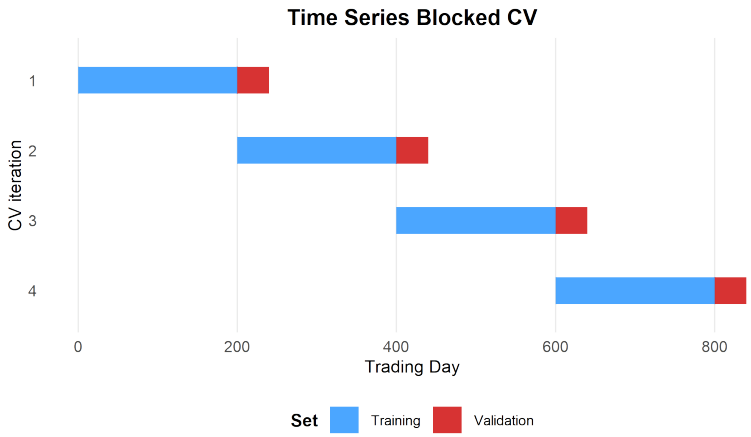

Figure 8: Blocked Cross Validation Iterations

The last method used is the Rolling Blocked Cross Validation. This technique extends the previous one by employing a rolling window approach. Consist on dividing the training set into more folds, and using more of them at once to predict the next one. Next, the first fold is removed from the training set, and the one used as validation is added instead. Practically speaking, folds 1-4, for example, were used to validate fold 5, folds 2-5 to validate fold 6 and so on. This dynamic adjustment of the training set simulates real-world forecasting conditions, where models are continuously updated as new data become available. The visualization of the process can be seen in Figure 9.

During each fold, models have been evaluated using different metrics. In particular:

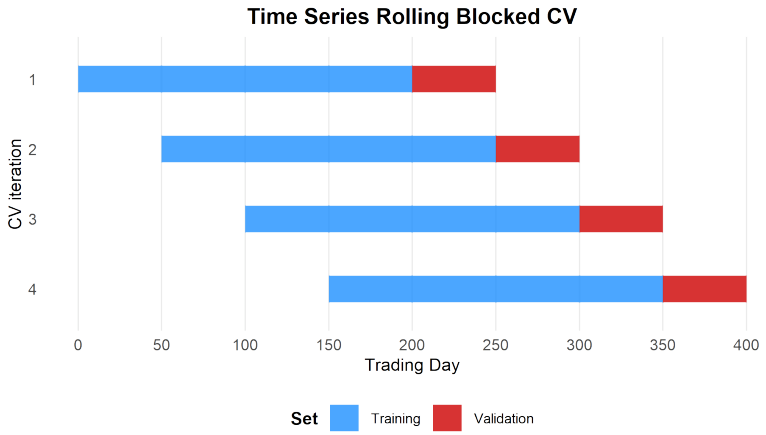

Figure 9: Rolling Blocked Cross Validation Iterations

- Mean Absolute Error (MAE): as Jierula et al. (2021) stated in their article, MAE represents the average magnitude of forecast errors. It does not consider direction, providing a linear score that treats all individual differences equally. It is computed as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|,$$

  where y_i represents the actual values, $\hat{y}_i$ represents the forecasted values, and n is the number of observations (Hyndman and Athanasopoulos, 2018).
- Root Mean Squared Error (RMSE): measures the average magnitude of error between the predicted and actual values. More simply, it can be described as the square root of the mean squared error (MSE), which measures the mean vertical squared distance of the actual values from the corresponding predicted values (Jierula et al., 2021). It is calculated as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

  Hyndman and Athanasopoulos (2018) describe MAE and RMSE as the two most commonly used measures which keeps the errors on the same scale of the data.
- Mean Absolute Percentage Error (MAPE): calculates the average of the percentage error. It's a measure of prediction accuracy, especially in trend estimation

(Jierula et al., 2021). It is computed as

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%.$$

As stated by Hyndman and Athanasopoulos (2018), metrics based on percentage errors have some disadvantages. They are scale-sensitive, so it will get extreme values if the actual value is small, and they will be infinite or undefined if $y_t = 0$ for any t in the period of interest. They also put a heavier penalty on negative errors than on positive errors.

- Mean Absolute Scaled Error (MASE): it's an alternative to percentage errors. The one used is the generalized formula for non-seasonal data:

$$MASE = \frac{\frac{1}{h}\sum_{t=n+1}^{n+h}|\hat{y}_t - y_t|}{\frac{1}{n-1}\sum_{t=2}^{n}|y_t - y_{t-1}|}$$

This metrics produce an error independent from the scale of the data. Some studies consider this to be the best metrics for evaluating forecasting models (e.g., in the one conducted by Franses (2016))

However, when forecasting variables such as stock prices, traditional error metrics alone may not fully capture a model's performance. The metrics listed above provide information on average accuracy, but they do not account for other important aspects such as the direction of the movement, the ability to follow the actual price path, or how well the model captures short-term fluctuations.

For this reason, visual inspection of forecasts plays a crucial role in the evaluation process. By plotting the predicted values against the actual ones, it becomes easier to assess whether the model captures the overall dynamics and volatility structure of the series, even in cases where numerical error metrics might suggest limited success.

Chapter 3

Sentiment Methods

3.1 Score Extraction

Although most of the data preparation was carried out during the retrieval process, one important task required further post-processing: sentiment generation. The underlying idea is that financial markets are influenced by factors beyond just price movements. Providing contextual information about external events can, therefore, enhance the quality of the analysis, and one way to incorporate such context is through financial news.

However, since regression models cannot directly interpret textual data, it was necessary to convert text into numerical representations. Therefore, sentiment analysis was selected as a method to translate news content into quantitative features. The choice of this method was due to the substantial body of scientific evidence that suggests that sentiment plays an important role in financial forecasting.

Several studies support the claim that sentiment can significantly affect financial predictions. For example, Mohan et al. (2019) found a clear correlation between financial news and the performance of an index tracking the 500 largest companies in the United States, regardless of their specific market sector. Similarly,Gupta and Chen (2020) demonstrated a strong connection between social media sentiments and the movements registered in the stock market. However, in recent years, access to comprehensive and updated social media data has become increasingly restricted, making its use both technically and financially challenging. As a result, incorporating such data would have conflicted with one of the goals of this research: to remain fully replicable and accessible.

The first step in working with sentiment data is therefore to generate the sentiment scores themselves. For this purpose, the Syuzhet R package was used (Jockers, 2023). This tool enables sentiment extraction using a robust and computationally intensive algorithm developed by the Natural Language Processing (NLP) group

at Stanford University (Manning et al., 2020). The core NLP process follows a structured pipeline, illustrated in Figure 10.

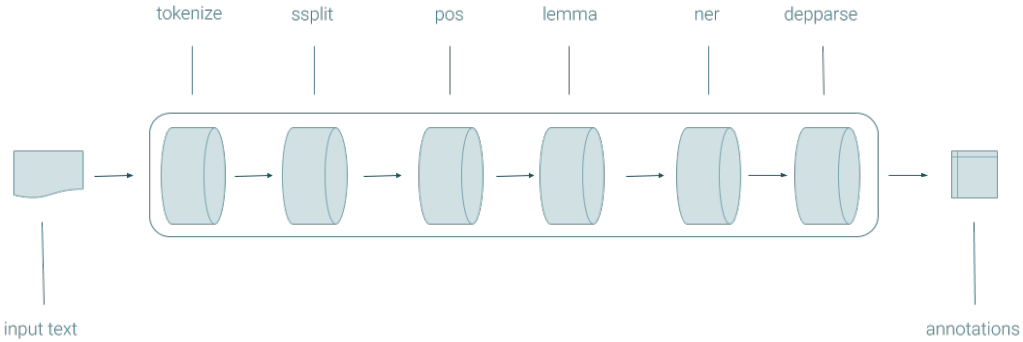

Figure 10: Core NLP pipeline
Source: `https://stanfordnlp.github.io/CoreNLP/`, Stanford NLP Group Manning et al. (2020)

One of the advantages of using Syuzhet is that it leverages the entire NLP pipeline to automatically handle essential preprocessing tasks. This includes tokenization, defined by Grefenstette (1999) as "dividing up the input text, which for a computer is just one long string of characters, into subunits, called tokens", and lemmatization, which Plisson et al. (2004) describes as the process of reducing a word to its normalized or dictionary form. These steps are critical to ensure consistent and meaningful sentiment extraction.

Two types of sentiment analysis were conducted. The first relies on the National Research Council (NRC) sentiment framework. According to Mohammad and Turney (2013), this method scores text based on the association of each word with eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiment polarities (positive and negative). The resulting distribution of sentiments can be observed in Figure 11, while the polarities can be seen in Figure 12.

The second approach is a more general sentiment scoring method. Among the available options, the Syuzhet method was selected for this research, as it is the only one that returns a preciser float value that represents the overall sentiment of the text. The sign of the value indicates whether the sentiment is positive or negative, providing a more meaningful single-score representation of each article.

The next step was to transform this sentiment data into a time series format suitable for training the model. A key challenge here was determining how to aggregate sentiment scores into a single representative value for each trading day. First, each news article was mapped to the appropriate trading day. For articles

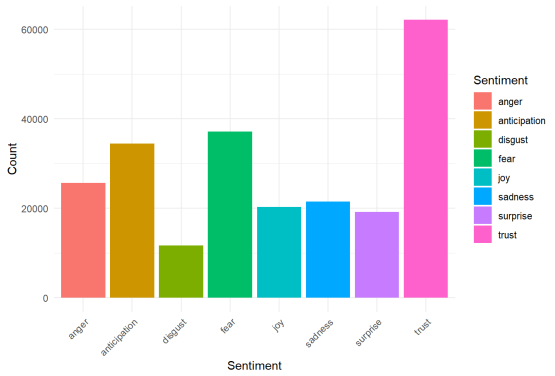

Figure 11: General Distribution of NRC Emotions

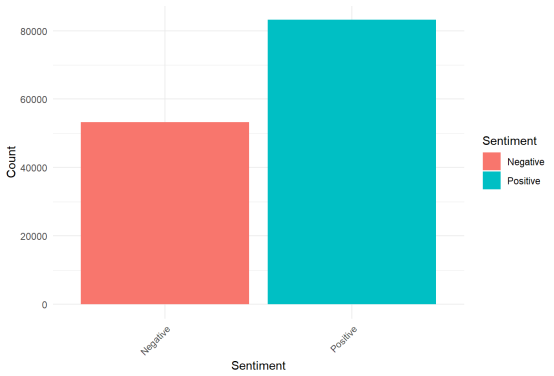

Figure 12: General NRC Polarities

published on non-trading days, the sentiment score was assigned to the first available trading day. For each trading day, the NRC emotion scores were summed and the general sentiment scores were averaged across all news articles. This was done separately for each category of news.

The resulting data structure is a time series table indexed by trading day. For each day it stores four records, one for each category ('Politics', 'Business', 'Technology', and 'Companies'). A visual representation of the resulting sentiment scores for the last 50 trading days can be observed in Figure 13.

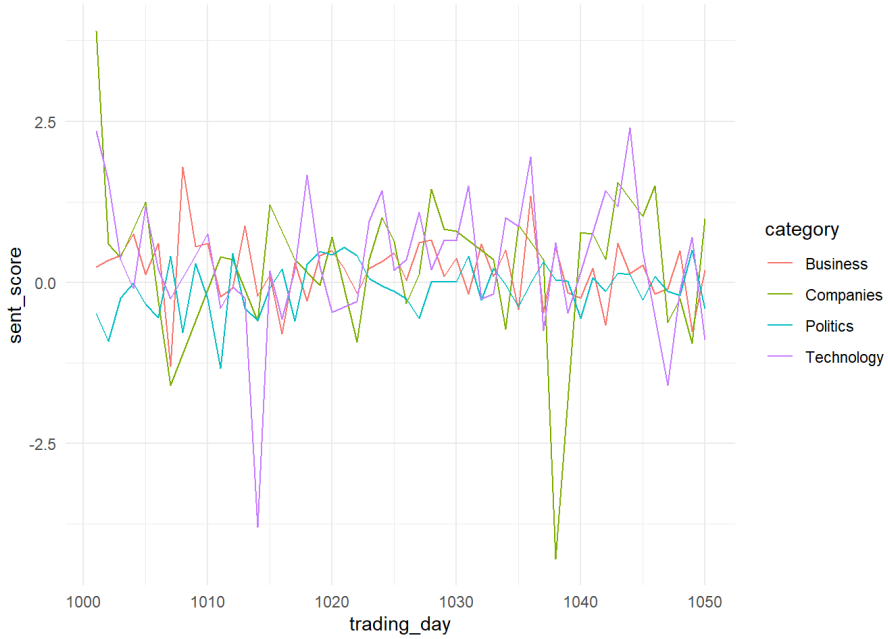

Figure 13: Visual Representation of Sentiment Scores

3.2 Model Definition

To add sentiments into the forecast process, it was necessary to choose the model to use to include these features. Models like ETS and Prophet can't handle multiple features at time, so they were not included in the choosing process. Other models were considered instead, such as Random Forest and LSTM. However, results on prices alone were not returning great results if compared with the others. So, the final choice was XGBoost. The objective was to take the best performing model on financial data and study the influences of news on the prediction outcome. And XGBoost was able to capture short-term fluctuations while being accurate in the long run, despite compound effect.

One of the challenges in this section of the analysis was deciding on how to implement future sentiments while avoiding look-ahead bias. The issue here is that when using a recursive multi-step ahead forecast, it is necessary to update all the features in each step. However, at time t, news referring to time $t + 1$ can't be known. So, the fist step was to find the best way to generate future data. Different methods were applied, using also in this step data up to the end of 2024 as training, and 2025 data only as test.

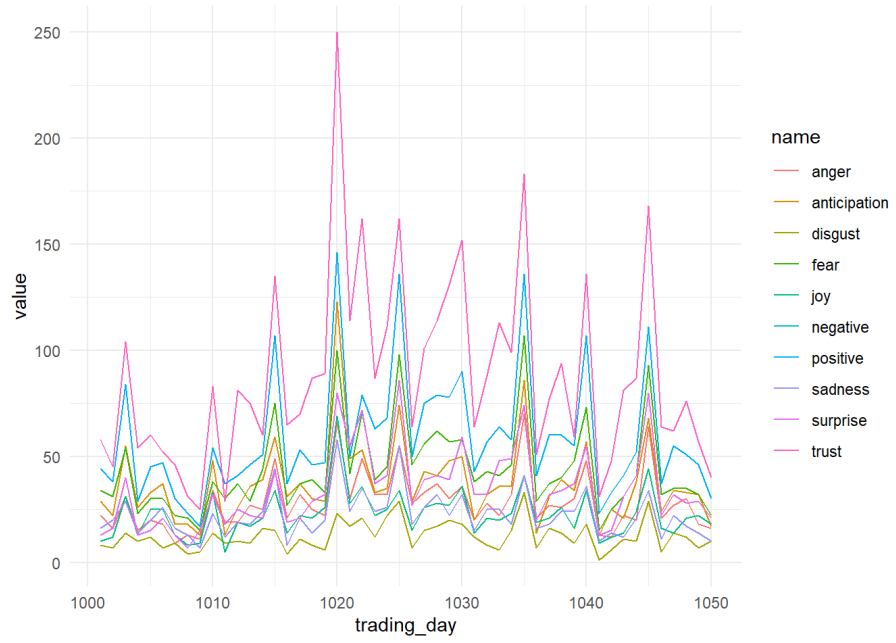

Figure 14: Time Distribution of NRC Sentiments

By looking at the time distribution of NRC sentiments in Figure 14, a time

dependant pattern can be observed. So ultimately, the models that were used to create future data where Prophet and XGBoost himself. A replica of the last trading days was also used, assuming similar features. However, for the sentiment score, this approach was not capturing the time independent spikes that can be observed in Figure 13. So for this particular feature, other than the three methods already listed, a Bootstrap was applied. As stated by Hyndman and Athanasopoulos (2018), Bootstrap only assumes that the forecast errors are uncorrelated. So when no general distributions or time dependence can be observed, this method is a good alternative.

The next step was including sentiments into the feature creation function used in the multi-step ahead forecast. The first approach was to include all the sentiments features of the day before, distinguishing by category. However, after some analysis and fine tuning, the final approach involves only the sentiment score for each category, as well as some feature interactions and some moving averages. Every model had some slightly different features depending on which was the most influent on Returns. For example, the one applied on the title tracking the general sector included:

- The same features used in the forecast without sentiments,
- Sentiment score from the day before for all the categories (Politics, Business, Technology and Companies),
- Some more lagged sentiment scores such as the lags 2, 4 and 20
- A 5 day rolling mean also for sentiment scores,
- Cross sentiments interactions such as the multiplication and the divergence between couples of sentiment scores appertaining to different categories.

The forecasts have then been compared with the models that were not using sentiment data based on the accuracy on the test set, simulating a real word scenario of future prediction.

3.3 Sentiment Dependence Study

In addition to evaluating base predictive accuracy, a whole study has been dedicated to understanding how different types of news can influence future asset prices. By manually injecting different scenarios into the data, a comparison between possible future outcomes was conducted. The scenarios studied included singular good/bad news and entire good and bad periods.

As a baseline, prophet-generated sentiment scores were used. Then, by manually tweaking specific days, different possible scenarios were created. The first two involved the publication of general news with extreme sentiment values (one really positive and really negative) across all four categories (Politics, Business, Technology and Companies). Next, a different scenario was created to isolate the impact of category-specific news. In this case, two extreme sentiment values were injected into one single category at a time, allowing analysis of the differential impact of each type of news.

Another scenario wanted to simulate prolonged sentiment trends by modifying sentiment scores over an extended period. Starting from a specific day, the sentiment scores where either increased or decreased by $0.5 * \sigma_n$ where σ_n is the standard deviation registered in the sentiment score for the specific category of news. Also in this case, this was done both across all categories simultaneously and for individual categories independently.

The results were then compared to the expected effects based on both the individual and mean contribution that each feature had on the outcome (Log_returns in this case). This was achieved using SHapley Additive exPlanations (SHAP) values. Lundberg and Lee (2017) describes SHAP values as an "unified measure of feature importance". Marcílio and Eler (2020) instead describes them as values that describes the "contribution for a models output, for each feature of each data point". The fundamental idea is to determine how much each feature contributes to a particular prediction by calculating its marginal contribution across all possible subsets of features. Analysing the sign and magnitude of the contribution, the general effect can be studied and compared with the observed outcomes.

Chapter 4

Results

4.1 Trend Analysis

Upon initial inspection, all financial time series exhibited a strong upward trend throughout the entire analysis period. As shown in Figure 2, which presents the STL decomposition for the IUIT.L ETF, this trend was consistently the dominant component, and all other decompositions look pretty similar. Most series revealed negligible seasonality, which could be safely ignored, and a remainder component that showed increasing variance towards the end of the period.

This observation suggested that models designed to capture trend components should perform relatively well in modelling the overall trajectory of the time series. Consequently, the first models applied were Exponential Smoothing (ETS) and ARIMA. The main issue with ETS models is that they are not able to capture short term fluctuations, but they still can be a good indicator of long term trends. Similarly, ARIMA models (particularly those without large MA components) also fail in capturing sudden changes or volatility in the data.

And after applying all the validation techniques, the results were not obvious at all. In Table 3 are shown all the Accuracy Metrics by Model and Method. For the blocked, rolling blocked and split cv, the results displayed are averages across all folds. As expected, the double ETS tends to be the more consistent between all the analysed models, being the best one for the blocked and rolling blocked, and ranked only second in the split cv. However, when looking to the traditional validation, results diverged significantly from these findings.

This discrepancy underscores one of the fundamental limitations of trend-based models: inability to detect turning points or shifts in volatility. The first two months of 2025 were marked by increased volatility and a turning point in the general trend. When comparing the two series in Figure 15 (showing test data and the traditional validation) and Figure 16 (showing the training set with the rolling blocked cv), the

turning point can be observed. Up to the end of 2025, all assets showed similar features to what was observed in the general infotech sector (shown in Figure 1). However, by simulating a real word scenario and supposing data past the end of 2024 were not observed, a forecast based on these observations would have been terrible.

This instability is further highlighted in Table 4, which shows the two best performing models per asset and validation method. There is no single model that consistently ranks among the top across all evaluations, indicating a lack of robustness. The reason is that even if custom Arimas and double Ets were showing the best performance for most folds, when they were failing, their errors were substantial, consistently inflating average error metrics. This is illustrated in Figure 17, where on the left panel there is a fold for AVGO in which the custom Arima model was getting really close to overlapping with the real data. On the other side there is a dramatic forecasting failure for NVDA, where all models erroneously projected a continued rise while the actual prices reversed sharply.

So, this first analysis revealed the two primary challenges that needed to be addressed: market volatility and trend reversals. However, before drastically changing the focus under analysis, another trend-seasonality model was employed: Prophet. The main intuition was that maybe there were some patterns that were not being captured by the Ets and Arima models. However, even Prophet, when applied to Close prices, did not produce satisfying results. As seen in Table 3, although it outperformed the other models in blocked CV on average, it performed poorly in the other validation strategies. Some examples can be seen in Figure 18. The problem here is that the model sometimes captures seasonalities that do not exist, introducing fluctuations that are not present in the real data and occasionally failing to follow the overall trend.

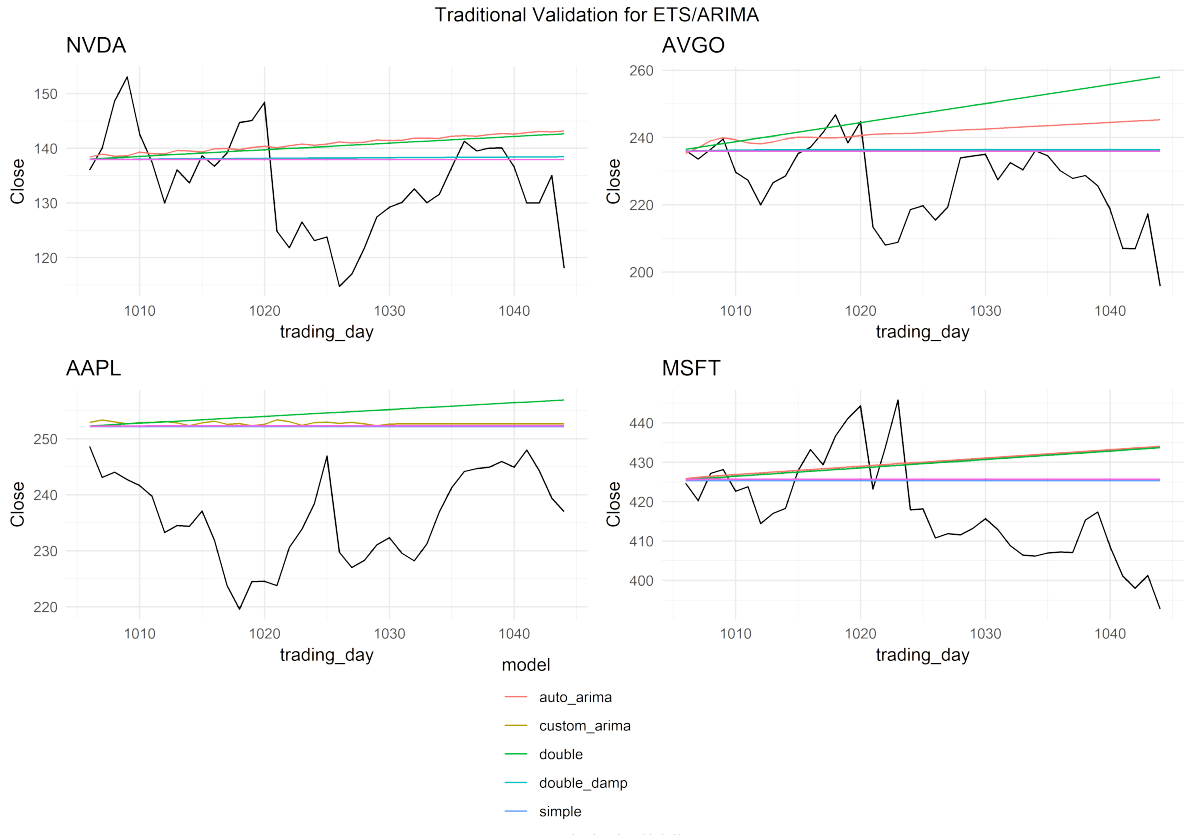

Figure 15: ETS/Arima Traditional Validation

Figure 16: ETS/Arima Rolling Blocked Validation

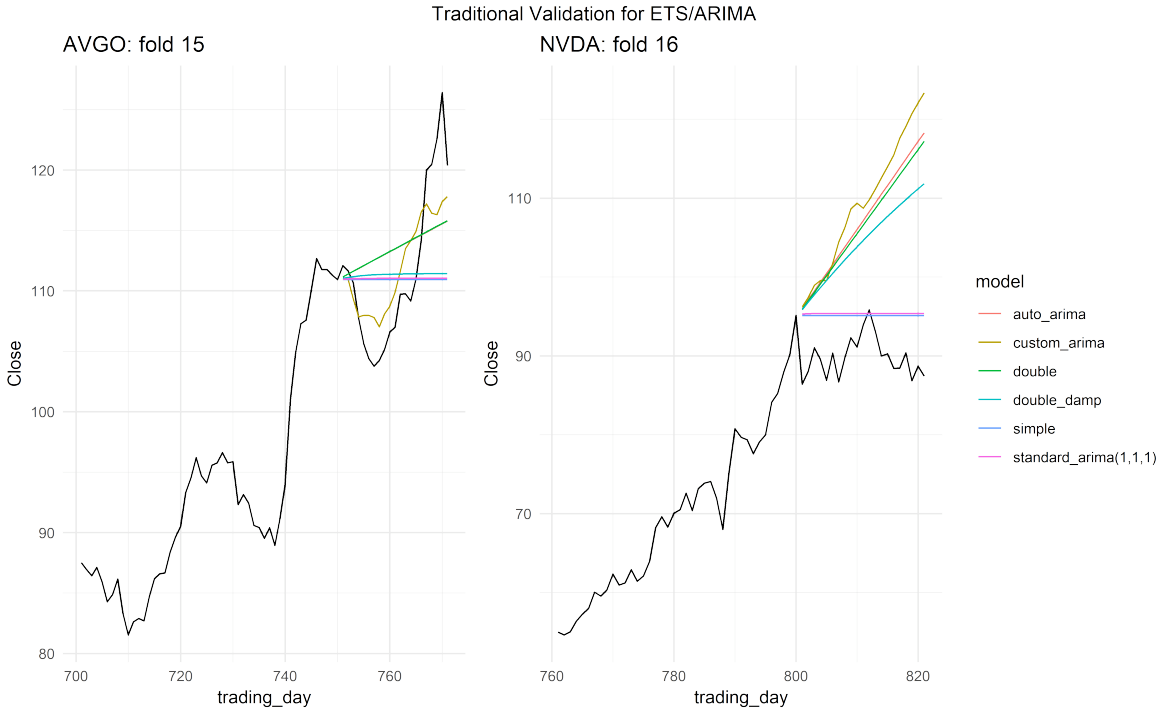

Figure 17: Good and Bad folds focus

Figure 18: Prophet Blocked Validation

4.2 Volatility Analysis

To address one of the main issues that were encountered in the first section, GARCH model was chosen. All assets showed volatility clustering throughout the whole period of analysis. ARCH effects instead were present in all assets, except for the NVDIA stock, which by testing on the full period resulted to not be able to reject the null hypothesis of No-ARCH-Effects. However, when testing only on the last years, the ARCH effects resulted to be present also in this asset.

In Table 5 are listed some results for each model by validation method. Also in this case strong inconsistency can be observed. There is no model that is staying at the top across every test, and the error metrics are quite high for all models.

This inconsistency is showed also by looking at the visual representation of results for the rolling blocked validation in Figure 19. Here can be seen that in some assets

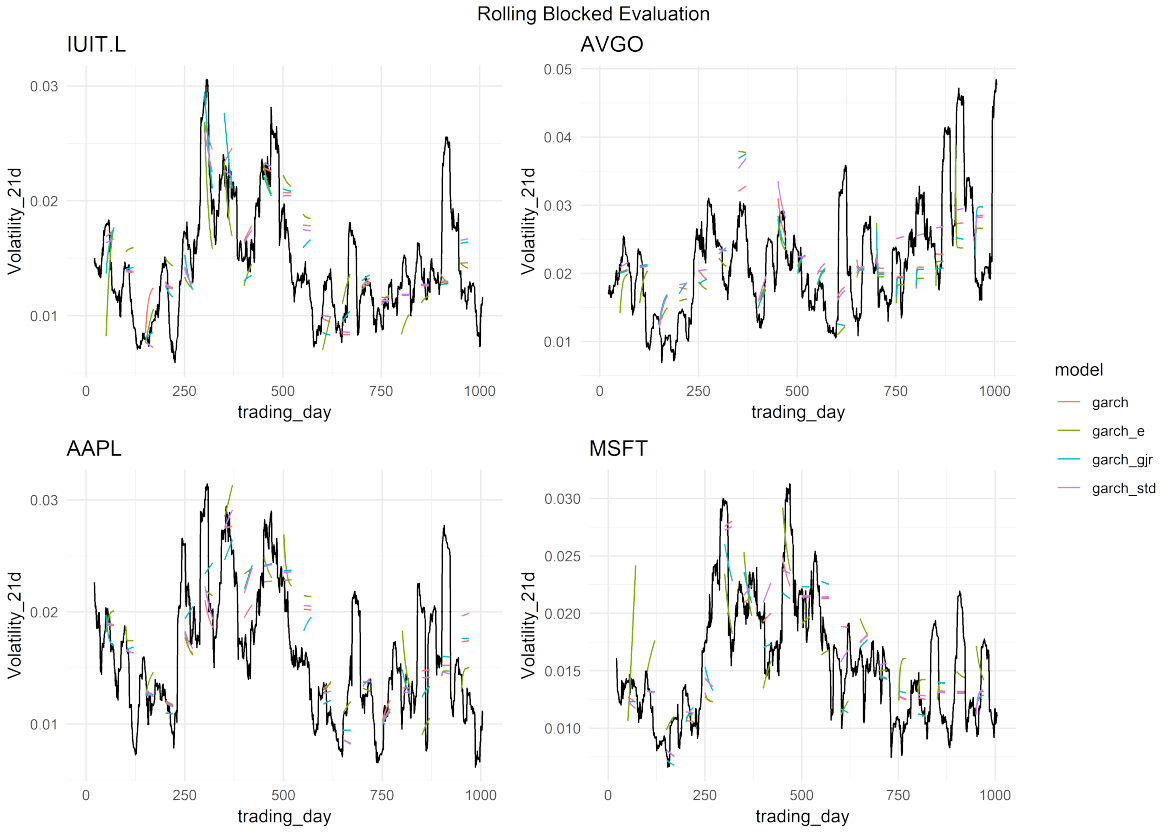

Figure 19: GARCH Rolling Blocked Validation

(e.g. in the IUIT.L ETF) the model was able to accurately predict most of the drops and the spikes in the volatility. In some others (e.g. in AAPL and MSFT stocks), predictions were all over the places, highlighting a lack of accuracy in different time

windows.

As a result, this approach was able to predict the increasing volatility for most stocks (as showed in Figure 20). However, the magnitude of the increase is not

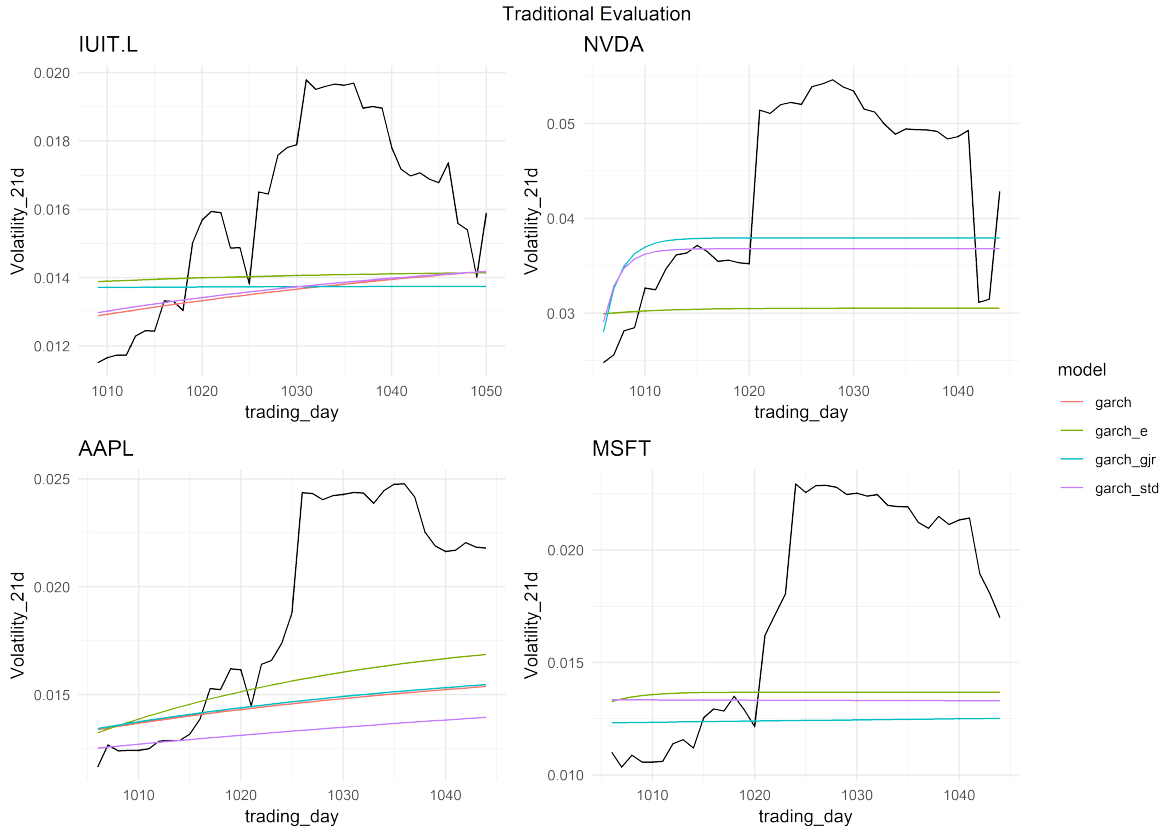

Figure 20: GARCH Traditional Validation

nearly close to the one that was observed in the reality. Moreover, even if forecasting volatility can help investors in managing risks, for this research this type of approach was not translating into good price forecasts. So from this section, volatility clustering and ARCH effects were highlighted, but without being able to model them.

4.3 Returns Analysis

To be able to account for fluctuations and turning points, a drastic change was needed. Since modelling prices was not yielding satisfactory results, and accounting only for volatility did not translate into good market forecasts, the analysis shifted onto Returns, specifically Logarithmic Returns.

By forecasting log(returns), models should be better equipped to capture fluctuations and turning points. However, a decision still had to be made regarding which model to use. Models like ETS are not suitable for this type of analysis, and applying ARIMA models produced results almost identical to those obtained from Close prices. In fact, since ARIMA models include differencing (via the integrated term), they inherently address what is achieved when working directly with log(returns).

Some additional models (such as Random Forest and LSTM) were also tested out of curiosity, but once again, they did not yield the desired results. Ultimately, the XGBoost model was selected. Each time series required a different set of parameters, so multiple fine tuning processes were performed. In the end, XGBoost was able to model every asset, with results that varied slightly depending on the characteristics of each stock.

As shown in Figure 21, forecasting this type of measure forces the models to focus on short-term movements. However, by also observing the mean returns, trends can be highlighted. Table 6 displays the related metrics. Here, the issue of scale-independent errors already mentioned arises: since the model works with values very close to zero, MAPE explodes while MASE is vanishing.

In Figure 22, the model's performance on the test periods is shown. Although some extreme drops are still missed, this is a substantial improvement over previous models. By reconstructing prices (starting from the last observed Close price and adding the forecasted returns), it's possible to get a general idea of the path prices might follow in the coming days. This is shown in Figure 23.

For AAPL and MSFT assets, the model managed to capture the general trend of the series, even if omitting some extreme spikes that were observed in the middle of the time window of reference. In NVDA, the model struggled at the start: it underestimated the first upward spikes and than failed to predict the subsequent

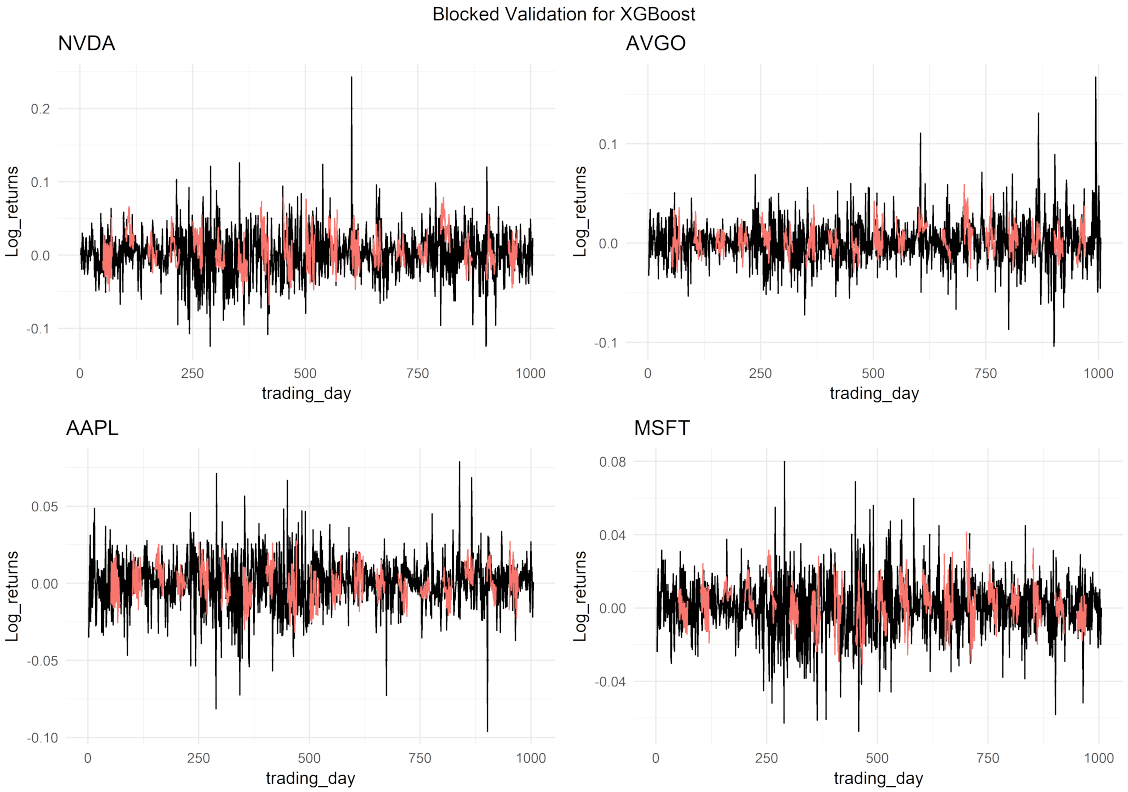

Figure 21: XGBoost Blocked Validation

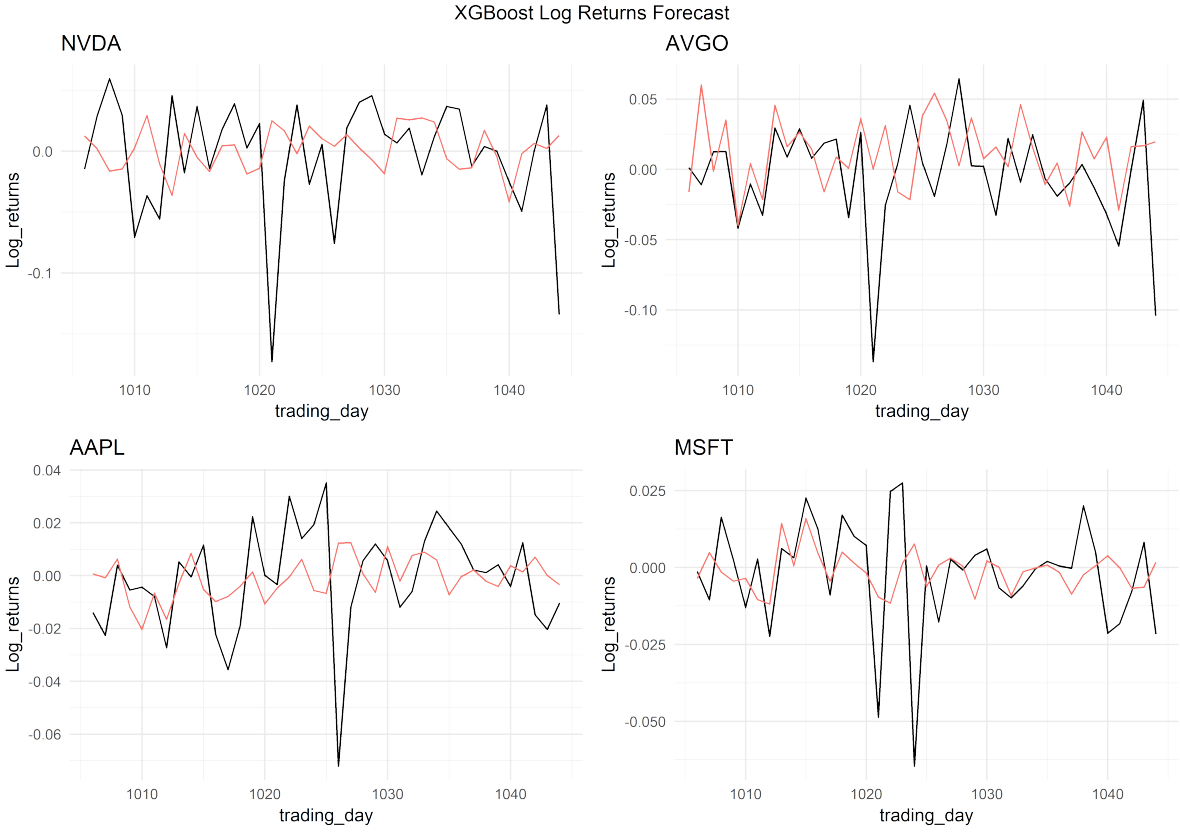

Figure 22: XGBoost 2025 Returns Forecast

Figure 23: XGBoost 2025 Close Prices Forecast

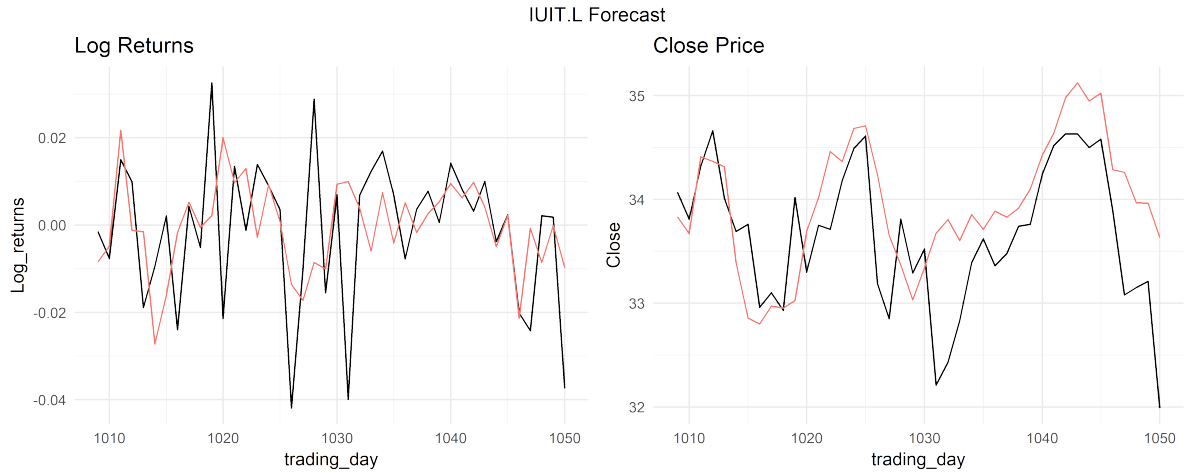

Figure 24: XGBoost 2025 Forecast for IUIT.L

drop. This may be due in part to compounding errors, as the final movements were quite close in direction, though slightly higher. The most problematic case was AVGO. As visible in earlier figures (e.g., in Figure 16), a sharp upward spike occurred just before the forecasting window. This was probably skewing the model's parameters, forcing it into unfamiliar territory. As a result, the forecast has some mean returns that overestimate the reality. However, as seen in Figure 22, the trend

in Log_Returns wasn't entirely off.

Even if individual stocks leave some open questions, the general market asset can serve as a strong reference point. The ETF tracking the infotech sector showed the best results. This is evident both in the validation metrics (Table 6) and in the plot shown in Figure 24, where the asset was almost perfectly tracked by the model, providing a highly accurate forecast of future market trends.

4.4 Sentiment Analysis

The first step in including sentiments into the forecast involved comparing the results of each model used to generate future sentiment data. In Figure 25, the outputs of different forecasting approaches are displayed. Prophet and XGBoost were able to capture the general trend of the sentiment trajectories. However, upon closer inspection of the y-axis, it becomes evident that the magnitude of movements is not perfectly aligned with reality. The replicated data follows similar general patterns to the real sentiments, but the actual series includes extreme spikes that are difficult to forecast.

Figure 25: Future Data Comparison: NRC Sentiments

Looking at the sentiment scores in figure 26, the situation differs slightly. These

series are not time dependent, unlike the NRC sentiment series, and these models struggled more with forecasting their dynamics. For this reason, multiple forecasts were generated using both predicted and bootstrapped sentiment scores.

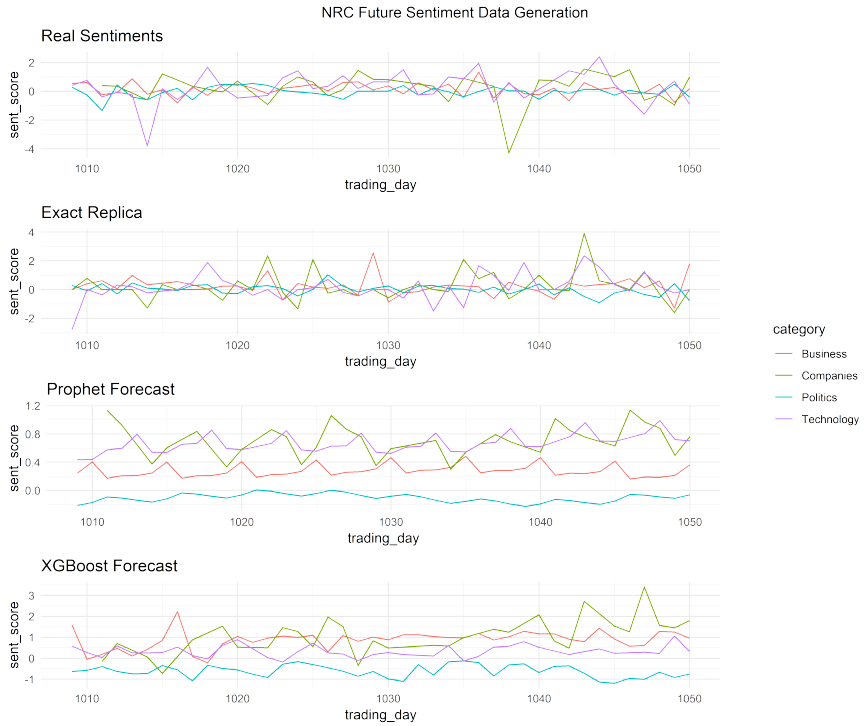

Figure 26: Future Data Comparison: Sentiment Scores

The results (displayed in Figure 27) show that Prophet seemed to be the more consistent one even without bootstrapping. Hence, the base Prophet generated sentiments were selected as the base for analysing the effect of sentiments on prices. Before testing different scenarios, mean SHAP values were computed to assess mean feature influences of each sentiment category. The results are presented in Table 1,

	Feature	Importance	Direction	Relative_Importance
1	sent_score_Politics	0.0013701	0.0000015	0.7757019
2	sent_score_Business	0.0011975	-0.0000245	0.6779727
3	sent_score_Companies	0.0011808	0.0000354	0.6684963
4	sent_score_Technology	0.0009615	-0.0000601	0.5443688

Table 1: Mean Effect of Sent_Score Features

where it is shown that sentiment categories related to Politics and Companies have a positive average effect on returns, while Business and Technology sentiments show a negative average influence. However, the magnitude of these effects is relatively small, making it difficult to anticipate how injecting sentiment signals would impact

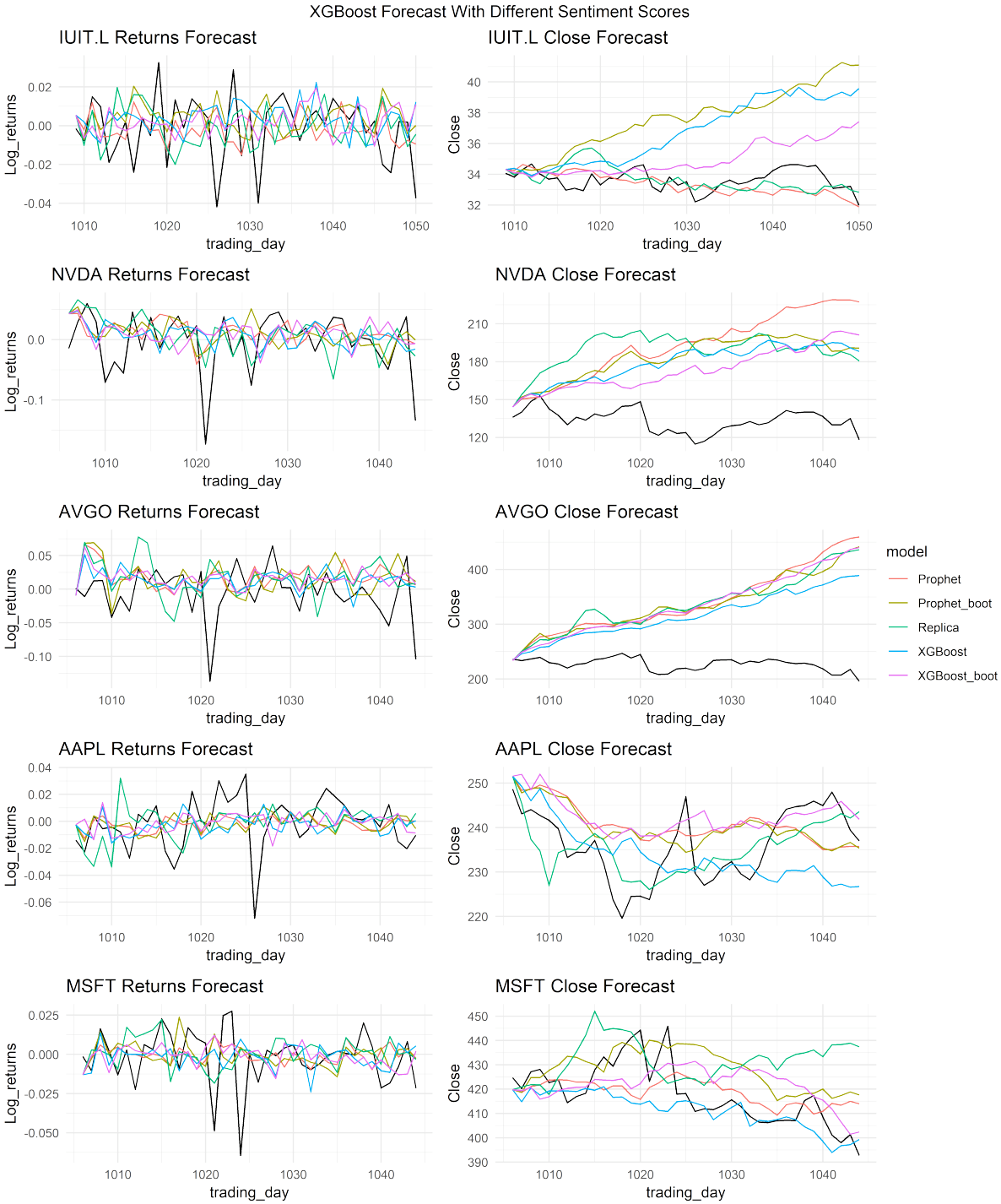

Figure 27: All XGBoost Forecast with different Future Data Generation Techniques

forecasts.

The first scenario tested involved injecting a single extreme sentiment value across all categories on day 1020. The divergences starting from that point are illustrated in Figure 28, where an unexpected result emerged: both the "Good" and "Bad" sentiment injections produced an upward price spike. This altered the

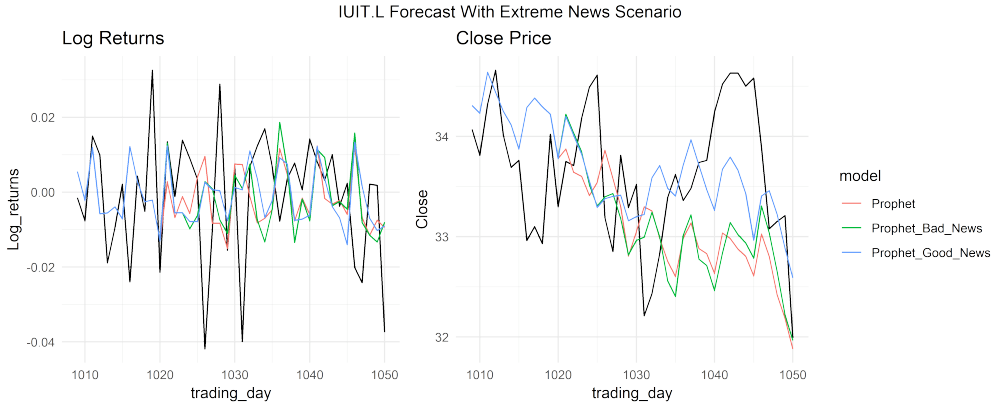

Figure 28: Extreme Sentiments Comparison

original forecast path: the good news series remained elevated, while the bad news series rejoined the baseline trajectory. Although IUIT.L (the ETF tracking the tech sector) is used as a representative example of all assets, almost equal results were observed across individual stocks. Initially, this was thought to be a consequence of the opposing SHAP effects of the singular categories: if the market is more sensible to upwards movements, injecting two extreme measures with opposite signs could have resulted in an upward movement in both cases. However, that intuition was quickly dismissed.

When focusing individual sentiment shocks, the results were also surprising. In Figure 29 can be seen what it's being referenced. Following the influence observed in Table 1, the series in the first row of plots were expected to increase in prices, while the bottom ones should have led to a drop. Instead, most scenarios showed downward shifts with respect to the base scenario, regardless of sentiment direction.

Before jumping to conclusions, a double check was performed. It was hypothesized that injecting extreme values might push the model into regions it had rarely encountered during training. To test this, new scenarios were created using more moderate shocks (0.5 * standard deviations) for extended periods. The results can be seen in Figure 30. Also in this case, the observed change in path did not match

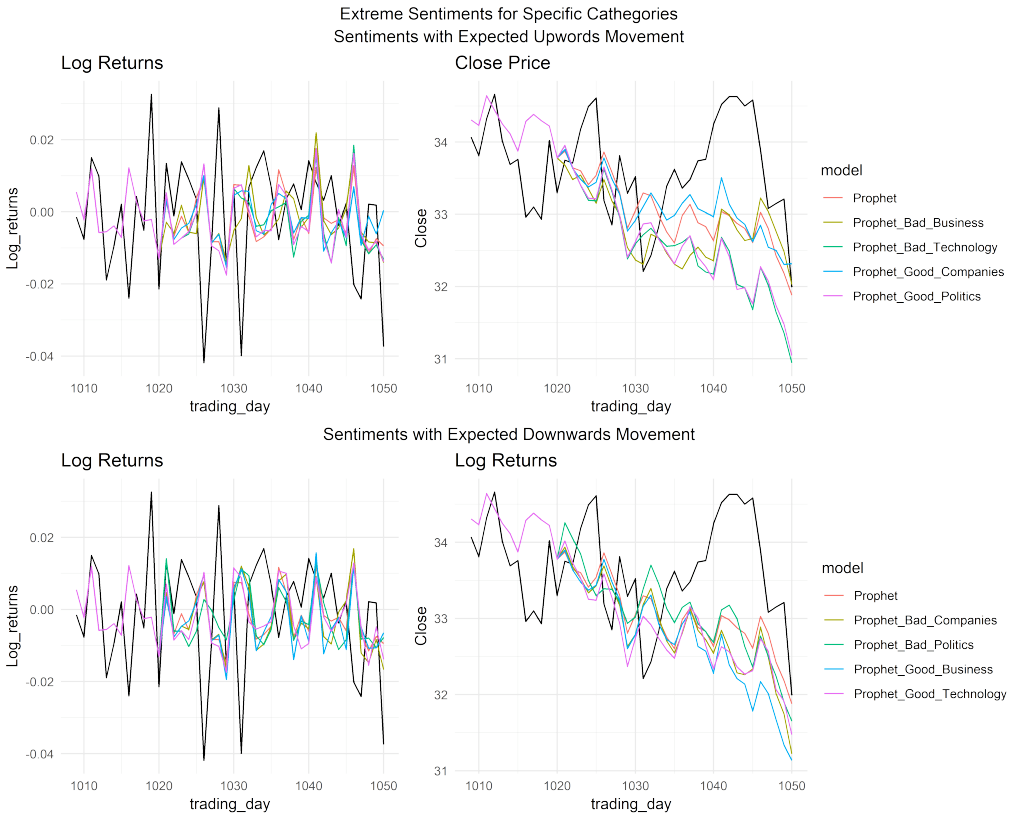

Figure 29: Extreme Sentiments Comparison between Specific Categories

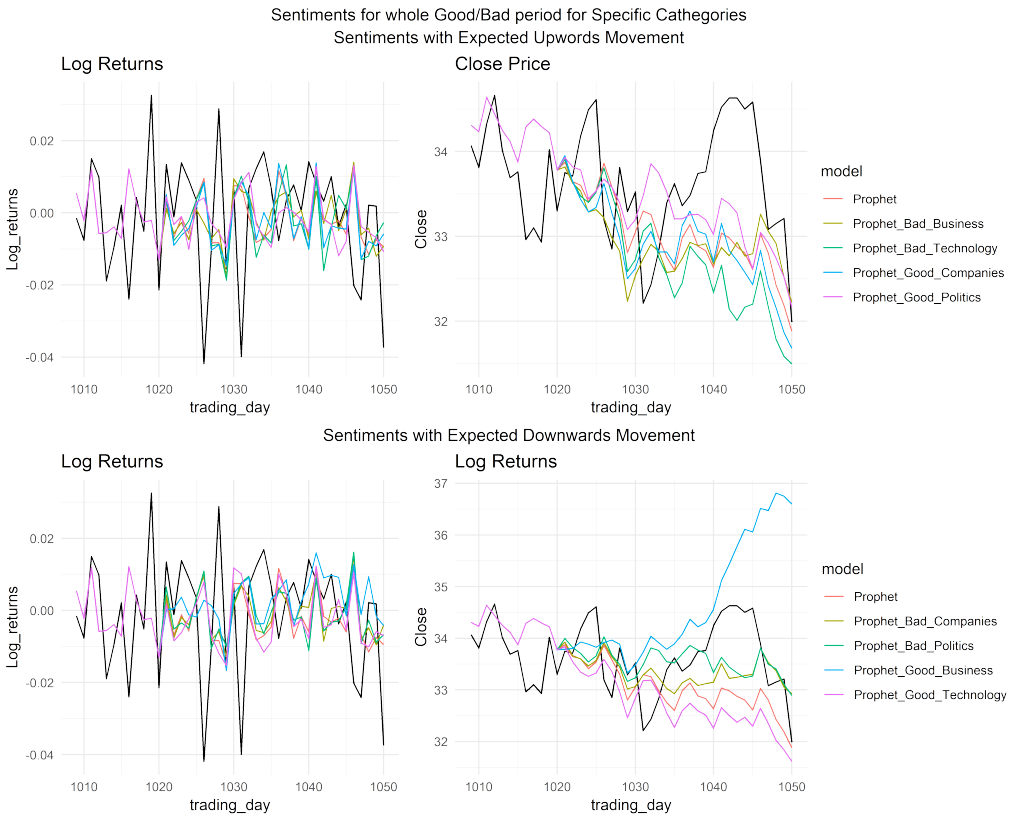

Figure 30: Sentiments Comparison between whole Good/Bad for Specific Categories

the SHAP-based expectations.

This discrepancy stems from a fundamental SHAP limitation: it reflects only the mean direction of each features impact, but it's only an average effect. The actual impact can vary significantly depending on current market context, interactions with other features, and non-linear relationships in the XGBoost model. Table 2 reports the ranges and the standard deviations of the effects. Even if the mean effects

	Attribute	Business	Companies	Politics	Technology
1	mean_shap	-0.0003137	0.0000531	0.0001022	-0.0001950
2	median_shap	-0.0005033	-0.0002412	0.0004052	-0.0000669
3	sd_shap	0.0015656	0.0008290	0.0015207	0.0013846
4	min_shap	-0.0041569	-0.0011323	-0.0044454	-0.0030075
5	max_shap	0.0047556	0.0028050	0.0029695	0.0051539
6	Positive count	16	19	30	24
7	Negative count	35	32	21	27
8	Positive %	31.37	37.25	58.82	47.06
9	range_shap	0.0089125	0.0039373	0.0074148	0.0081614

Table 2: XGBoost SHAP Focus for Sent_Score of each Category

are positive/negative, the ranges are quite great compared to the effects, and they space in both positive and negative directions. In other words, it's not possible to be confident that one news will influence the series in a predetermined way, but also in this case the result varies depending on the context.

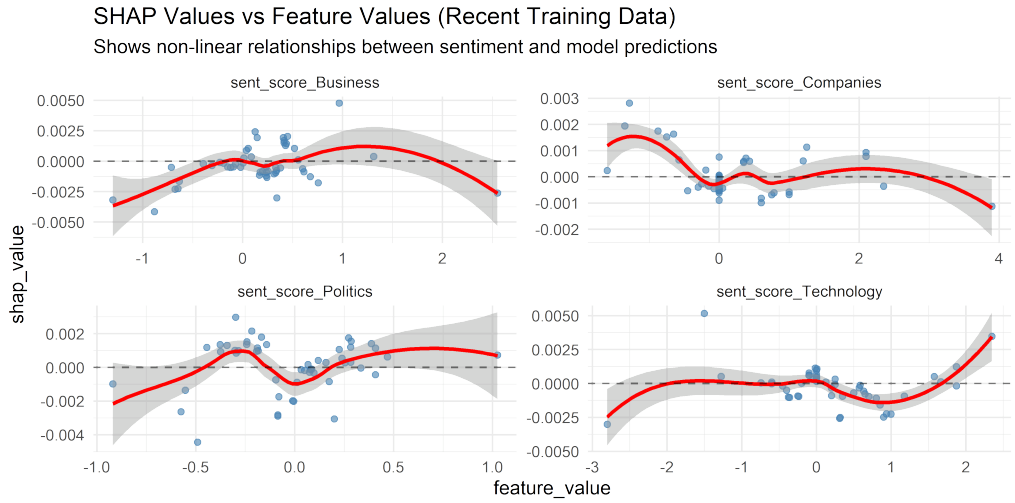

Figure 31: Non Linear Relationships Between Sentiments and Returns

A visual representation of the influence that each sentiment category have on the Returns can be seen in Figure 31. From the red curves, complex and non-monotonic

relationships emerge, suggesting that the same sentiment values can have opposite effects depending on other factors. There are no clearly defined sentiment thresholds where the effect consistently switches direction.

This last analysis highlights that changes in sentiment do not produce uniform outcomes across different contexts. However, this does not imply that sentiment is not a good predictor of returns. The problem is that, so far, forecasts that included sentiment data never outperformed those relying on returns alone. Therefore, before questioning the methods for future sentiment data generation, a further experiment was conducted under the best case scenario: perfect information. Assuming full knowledge of how news will be published in the future, the actual sentiment values from the test set were used to build the model and compare it to the one without sentiment features.

The first finding was that the NRC sentiments were only adding uncertainty to the model, without really giving any meaningful signals. They were therefore removed from the feature engineering function. Then, since the previous analysis highlighted that the influence of sentiments are context specific, new interaction terms, lags and derived features were introduced. The resulting model is the one described in Chapter 3.2.

Some visual examples of the results are shown in Figure 32. Compared to what was observed before (Figure 27) this represents a clear improvement. However, even under the perfect scenario (so using the actual sentiments from the test set), for these assets the model without the sentiment scores performed better. This suggests that unless additional features that are both good predictors of returns and correlated with sentiment scores are included, with this data it's better to rely solely on market-based informations.

Several explanations can be proposed for this outcome. First, the algorithm generating the sentiment scores may be introducing variability. The same words in different contexts can express very different meanings, but the algorithm might assign them similar scores. So, instead of providing context, sentiment scores might be injecting bias. Another possible reason is the source of the news: the New York Times, while reputable, is not a financial-focused publisher, and the resulting sentiment data may be diluted by unrelated articles or general news that is less

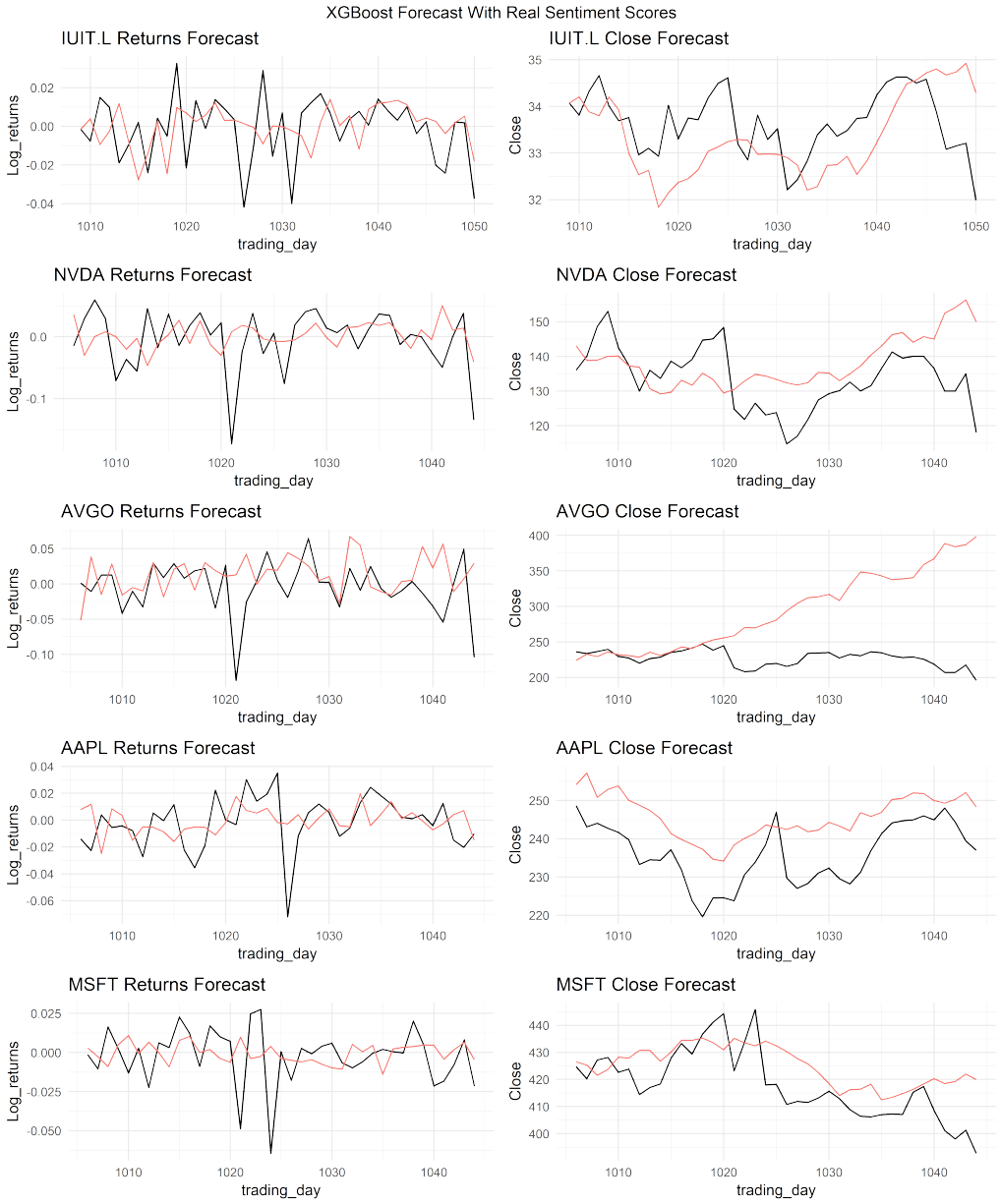

Figure 32: All XGBoost Forecast with Real Sentiment Scores

relevant to market behaviour and cannot be easily filtered out.

Chapter 5

Conclusions

This analysis compared different forecasting techniques that can be used by individuals interested in financial markets to obtain some meaningful insights on future trends. In particular, the infotech market was the sector under analysis. Through the implementation of different techniques, this research was able to address several important characteristics of the subject under analysis.

Using STL decomposition it was highlighted that this sector is distinguished by a strong trend that goes up for the most part, negligible seasonality, and a reminder component getting higher towards the end, signalling an increase in unpredictability that has been characterising the last period under analysis.

This increased unpredictability was cause of failure for most models analysed. Techniques like ETS and ARIMA, which model trend, were returning good results for most validation folds, but then failed in forecasting the test period. Also when implementing GARCH, meaningful informations were obtained (such as the presence of volatility clustering and ARCH effects), but the models failed to predict the test period. This highlighted the need of implementing a different approach that could have captured short-term movements and strong changes in trends.

After some exploratory analysis in which different models were tested, XGBoost was found to be more suitable for this objective. By changing the focus from Close prices to Logarithmic Returns and implementing this new model, results were improved and the general trend was better captured. There are still some assets in which this technique is struggling, mostly because of the observation of some extreme values right before the test period. However, the general cross-assets performance was already able to give good insights into the future movements of this sector.

With the next objective being trying to improve the accuracy of the last models, a sentiment analysis was performed on different categories of news. While doing this, a full study of the relationship between Returns and Sentiments was performed. Results highlighted a complex and non-monotonic relationships, with sentiment

values that can push Returns in opposite directions depending on the context. Even if the mean effects show a positive/negative sign, this is not enough to confidently determine how a news will influence the trends registered by each asset.

While analysing different scenarios and different methods of generating future data for the Sentiment Scores, no predictions was found to be better than the one relying only on financial data. Hence, some tests under the best case scenario were performed. After some fine tuning using sentiment scores appertaining to the test period, a big improvement was made. However, even under perfect information assumption, the results have not improved what was done with Returns alone.

Probably the algorithm used to generate the sentiment scores may have introduced variability into the model by giving the same score to different words that used in different context have different meanings. Another possible problem could have been the source of news data. The New York Times is not a finance-specific publisher, so sentiment data may be diluted by unrelated articles or general news that are less relevant to market behaviour and cannot be easily filtered out.

The intuition when deciding on the data source was that even big politics and technology news could have influenced the market. But among all the news appertaining to such categories, those are probably only a subset difficult to identify. In addition, the objective of the analysis was to keep it accessible, and the New York Times was a perfect fit.

However, considering the obtained results, a change in data source might be needed to improve the forecasting capabilities of the model using sentiment analysis. Probably using a financial specific publisher (whose data should have more correlation with the analysed series), results could diverge from what was observed during this study. The main issue is that such data is generally either paid or limited to recent publications.

In conclusion, this analysis managed to give good insights on the Infotech Sector, analysing general trends and highlighting what can be achieved with different forecasting techniques. Sentiment data coming from non financial-specific news was found to have a complex relation with asset's trends but was not found to improve the models' performance. A more specific data source is suggested to replicate this type of analysis.

Metrics Tables

	method	model	MAE	RMSE	MAPE	MASE
1	blocked cv	double	5.17	6.07	5.27	3.99
2	blocked cv	auto_arima	5.19	6.06	5.37	4.13
3	blocked cv	standard_arima(1,1,1)	5.28	6.19	5.24	4.13
4	blocked cv	simple	5.32	6.22	5.20	4.09
5	blocked cv	double_damp	5.44	6.35	5.41	4.21
6	blocked cv	custom_arima	5.69	6.62	5.47	4.12
7	blocked cv	prophet	7.71	8.66	6.87	5.45
8	rolling blocked	prophet	5.13	6.36	21.24	8.77
9	rolling blocked	double	5.40	6.39	5.41	4.31
10	rolling blocked	simple	5.48	6.54	5.27	4.27
11	rolling blocked	standard_arima(1,1,1)	5.51	6.58	5.31	4.30
12	rolling blocked	auto_arima	5.52	6.53	5.44	4.38
13	rolling blocked	double_damp	5.59	6.66	5.43	4.40
14	rolling blocked	custom_arima	5.94	6.92	5.92	4.65
15	split cv	custom_arima	10.95	12.81	10.98	9.51
16	split cv	double	11.20	13.09	10.77	9.74
17	split cv	double_damp	11.36	13.43	10.72	9.92
18	split cv	standard_arima(1,1,1)	11.43	13.53	10.74	10.02
19	split cv	simple	11.45	13.54	10.73	10.01
20	split cv	auto_arima	11.91	13.86	11.12	10.55
21	split cv	prophet	13.39	15.46	10.98	10.48
22	traditional	simple	9.51	11.53	4.59	5.60
23	traditional	standard_arima(1,1,1)	9.57	11.60	4.61	5.62
24	traditional	double_damp	9.59	11.64	4.64	5.66
25	traditional	custom_arima	10.65	12.70	5.19	6.28
26	traditional	auto_arima	11.47	13.78	5.59	6.89
27	traditional	double	12.89	15.32	6.16	7.69
28	traditional	prophet	14.76	16.95	7.60	9.55
29	traditional rec	standard_arima(1,1,1)	9.08	11.17	4.36	3.04
30	traditional rec	simple	9.51	11.54	4.59	3.30
31	traditional rec	double_damp	9.57	11.60	4.62	3.31
32	traditional rec	auto_arima	9.71	11.78	4.65	3.34
33	traditional rec	double	11.97	14.16	6.01	4.17
34	traditional rec	custom_arima	12.56	14.52	6.38	4.34
35	traditional rec	prophet	18.95	21.55	8.23	10.15

Table 3: ETS, ARIMA and Prophet Accuracy Metrics by Method and Model

	ticker	method	model	MAE	RMSE	MAPE	MASE
1	AAPL	blocked cv	double	5.22	6.19	3.11	2.56
2	AAPL	blocked cv	auto_arima	5.22	6.03	3.08	2.62
3	AAPL	rolling blocked	simple	6.12	7.30	3.68	3.17
4	AAPL	rolling blocked	double_damp	6.14	7.33	3.69	3.17
5	AAPL	split cv	custom_arima	12.90	15.40	7.58	6.80
6	AAPL	split cv	standard_arima(1,1,1)	13.80	16.43	8.16	7.25
7	AAPL	traditional	double_damp	16.12	17.91	6.95	7.86
8	AAPL	traditional	simple	16.12	17.91	6.95	7.86
9	AVGO	blocked cv	standard_arima(1,1,1)	5.26	6.06	5.52	4.74
10	AVGO	blocked cv	double_damp	5.40	6.14	5.46	4.66
11	AVGO	rolling blocked	simple	4.16	5.27	4.92	4.19
12	AVGO	rolling blocked	standard_arima(1,1,1)	4.20	5.33	5.01	4.26
13	AVGO	split cv	double	9.24	11.02	9.94	10.90
14	AVGO	split cv	custom_arima	10.02	11.72	10.46	11.65
15	AVGO	traditional	custom_arima	9.29	11.71	4.18	6.21
16	AVGO	traditional	simple	10.77	14.61	4.97	7.20
17	IUIT.L	blocked cv	double_damp	0.76	0.88	3.50	3.43
18	IUIT.L	blocked cv	double	0.76	0.91	3.50	3.31
19	IUIT.L	rolling blocked	double	0.78	0.94	3.72	3.63
20	IUIT.L	rolling blocked	auto_arima	0.79	0.94	3.71	3.68
21	IUIT.L	split cv	double	1.39	1.62	6.54	6.54
22	IUIT.L	split cv	custom_arima	1.43	1.67	6.74	6.75
23	IUIT.L	traditional	double_damp	0.66	0.82	2.00	2.76
24	IUIT.L	traditional	simple	0.67	0.82	2.00	2.76
25	MSFT	blocked cv	auto_arima	8.70	10.74	2.74	2.57
26	MSFT	blocked cv	double	8.76	10.72	2.74	2.54
27	MSFT	rolling blocked	double	10.44	12.20	3.42	3.23
28	MSFT	rolling blocked	auto_arima	10.89	12.81	3.58	3.39
29	MSFT	split cv	double	22.09	25.64	7.36	7.07
30	MSFT	split cv	custom_arima	22.10	25.68	7.33	7.02
31	MSFT	traditional	simple	12.19	14.42	2.95	3.27
32	MSFT	traditional	double_damp	12.19	14.42	2.95	3.27
33	NVDA	blocked cv	simple	4.24	4.81	10.42	6.26
34	NVDA	blocked cv	standard_arima(1,1,1)	4.33	4.90	10.61	6.36
35	NVDA	rolling blocked	simple	4.70	5.43	10.30	6.67
36	NVDA	rolling blocked	standard_arima(1,1,1)	4.72	5.45	10.33	6.69
37	NVDA	split cv	custom_arima	8.28	9.58	22.78	15.34
38	NVDA	split cv	double_damp	8.41	9.94	20.22	15.46
39	NVDA	traditional	standard_arima(1,1,1)	7.79	9.90	6.06	6.88
40	NVDA	traditional	simple	7.79	9.90	6.07	6.88

Table 4: Best two ETS/ARIMA models by Asset and Method

	method	model	MAE	RMSE	MAPE	MASE
1	blocked cv	garch	0.00586	0.00645	24.71243	8.07399
2	blocked cv	garch_e	0.00685	0.00757	28.97031	9.64138
3	blocked cv	garch_std	0.00704	0.00759	30.39200	9.55112
4	blocked cv	garch_gjr	0.00765	0.00825	33.08416	10.12423
5	rolling blocked	garch_std	0.00464	0.00514	22.65488	6.62676
6	rolling blocked	garch	0.00465	0.00512	22.85935	6.63979
7	rolling blocked	garch_e	0.00514	0.00567	24.77286	7.47397
8	rolling blocked	garch_gjr	0.00523	0.00574	24.62304	7.13386
9	split cv	garch_gjr	0.00594	0.00699	32.48227	9.19631
10	split cv	garch	0.00594	0.00699	32.98997	9.23700
11	split cv	garch_std	0.00613	0.00721	33.41920	9.38235
12	split cv	garch_e	0.00621	0.00726	34.18888	9.77791
13	traditional	garch_gjr	0.00669	0.00790	22.71884	9.23689
14	traditional	garch	0.00671	0.00793	22.59961	9.17286
15	traditional	garch_std	0.00685	0.00812	23.15161	9.40884
16	traditional	garch_e	0.00757	0.00887	24.11135	9.86441

Table 5: Garch Results for Method and Model

	method	ticker	MAE	RMSE	MAPE	MASE
1	blocked cv	IUIT.L	0.01334	0.01656	213.81831	0.82337
2	blocked cv	AAPL	0.01427	0.01818	629.87605	0.78686
3	blocked cv	MSFT	0.01530	0.01958	387.73333	0.85923
4	blocked cv	AVGO	0.02138	0.02746	468.48024	0.88817
5	blocked cv	NVDA	0.03786	0.04781	412.73614	1.05408
6	rolling blocked	IUIT.L	0.01350	0.01729	235.99255	0.83311
7	rolling blocked	AAPL	0.01473	0.01945	362.18864	0.81218
8	rolling blocked	MSFT	0.01590	0.01976	379.34454	0.89273
9	rolling blocked	AVGO	0.02165	0.02781	42990.50000	0.89933
10	rolling blocked	NVDA	0.03314	0.04239	357.52757	0.92275
11	split cv	IUIT.L	0.01376	0.01769	234.51508	0.84914
12	split cv	MSFT	0.01418	0.01819	473.58806	0.79613
13	split cv	AAPL	0.01444	0.01919	831.28055	0.79653
14	split cv	AVGO	0.01990	0.02578	11300.80613	0.82665
15	split cv	NVDA	0.03016	0.03865	67763.57523	0.83973
16	traditional	IUIT.L	0.01199	0.01685	165.35942	0.73987
17	traditional	MSFT	0.01247	0.01840	222.95399	0.70036
18	traditional	AAPL	0.01543	0.02110	314.92440	0.85071
19	traditional	AVGO	0.03221	0.04418	341.99484	1.33824
20	traditional	NVDA	0.04136	0.05612	501.23624	1.15167

Table 6: XGBoost Results for Method and Asset

Bibliography

Agrawal, S., Kumar, N., Rathee, G., Kerrache, C. A., Calafate, C. T., and Bilal, M. (2024). Improving stock market prediction accuracy using sentiment and technical analysis. *Electronic Commerce Research*, pages 1–24.

Chatfield, C. and Xing, H. (2019). *The analysis of time series: an introduction with R*. Chapman and hall/CRC.

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

Chessum, M. (2024). Deepseek disruption: The new wave of volatility in tech shares. *S&P Global Market Intelligence*.

Deepan, S., Reddy, P. S., Vyshnevy, M., Tejasri, K., Akshaya, B., and Nikhil, P. (2024). Prediction and analysis of stock markets using arima models. In *2024 3rd International Conference on Computational Modelling, Simulation and Optimization (ICCMSO)*, pages 268–271. IEEE.

FinanceCharts.com (2025). Biggest companies stock screener. Last updated: Jun 03, 2025.

Franses, P. H. (2016). A note on the mean absolute scaled error. *International Journal of Forecasting*, 32(1):20–22.

Fu, K. and Zhang, Y. (2024). Incorporating multi-source market sentiment and price data for stock price prediction. *Mathematics*, 12(10):1572.

Grefenstette, G. (1999). Tokenization. In *Syntactic wordclass tagging*, pages 117–133. Springer.

Gupta, R. and Chen, M. (2020). Sentiment analysis for stock price prediction. In *2020 IEEE conference on multimedia information processing and retrieval (MIPR)*, pages 213–218. IEEE.

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

Jierula, A., Wang, S., Oh, T.-M., and Wang, P. (2021). Study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data. *Applied Sciences*, 11(5):2314.

Jockers, M. (2023). *Syuzhet: Extracting Sentiment and Sentiment-Derived Plot Arcs from Text*. R package vignette.

Li, Y., Chen, L., Sun, C., Liu, G., Chen, C., and Zhang, Y. (2024). Accurate stock price forecasting based on deep learning and hierarchical frequency decomposition. *IEEE Access*.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2020). *Stanford CoreNLP: A Suite of Core NLP Tools*. Version 4.5.6.

Marclio, W. E. and Eler, D. M. (2020). From explanations to feature selection: assessing shap values as feature selection mechanism. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 340–347.

Marisetty, N. (2024). Prediction of popular global stock indexes volatility by using arch/garch models. *GARCH Models (July 24, 2024)*.

Mohammad, S. M. and Turney, P. D. (2013). Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Mohan, S., Mullapudi, S., Sammeta, S., Vijayvergia, P., and Anastasiu, D. C. (2019). Stock price prediction using news sentiment analysis. In *2019 IEEE fifth international conference on big data computing service and applications (BigDataService)*, pages 205–208. IEEE.

Plisson, J., Lavrac, N., Mladenic, D., et al. (2004). A rule based approach to word lemmatization. In *Proceedings of IS*, volume 3, pages 83–86. Citeseer.

Shunway, R. H. and Stoffer, D. S. (2019). *Time series: a data analysis approach using R.* Chapman and Hall/CRC.

Sunki, A., SatyaKumar, C., Narayana, G. S., Koppera, V., and Hakeem, M. (2024). Time series forecasting of stock market using arima, lstm and fb prophet. In *MATEC Web of Conferences*, volume 392, page 01163. EDP Sciences.

Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.

Verma, V. (2024). Exploring key xgboost hyperparameters: A study on optimal search spaces and practical recommendations for regression and classification. *International Journal of All Research Education and Scientific Methods (IJARESM), ISSN*, 12:2455–6211.

Woodward, W. A., Gray, H. L., and Elliott, A. C. (2017). *Applied time series analysis with R.* CRC press.

Yahoo Finance (2025). Yahoo finance.