HEC MONTRÉAL

affiliée à l'Université de Montréal

Forecasting the Intraday Trading Volume of Stocks

par

Alireza Fallahi

Vincent Grégoire

HEC Montréal

Directeur de recherche

Saad Ali Khan

HEC Montréal

Codirecteur de recherche

Sciences de la gestion

Spécialisation Applied Financial Economics

Mémoire présenté en vue de l'obtention

du grade de maîtrise ès sciences

(M. Sc.)

August 2023

Resume

Cette étude offre une analyse comparative exhaustive des différentes méthodologies de prévision, visant à prédire le volume de trading intrajournalier pour les actions dans le contexte de l'indice S&P 100. En exploitant le vaste ensemble de données Trade and Quote (TAQ), nous évaluons de manière critique le potentiel des modèles économétriques traditionnels et des techniques d'apprentissage automatique, découvrant leurs capacités et leurs limites.

En commençant par le modèle Autoregressive Moving Average (ARMA), nous établissons une référence pour la comparaison avec les méthodologies suivantes. Sur cette base, nous nous aventurons dans le domaine de l'apprentissage automatique avancé, examinant les réseaux neuronaux Long Short-Term Memory (LSTM) et les Temporal Convolutional Networks (TCN), connus pour leur compétence à gérer les dépendances à long terme dans les données séquentielles. Notre analyse est de plus enrichie par l'exploration des méthodes d'apprentissage en ensemble telles que le Gradient Boosting et la Random Forest.

Une évaluation approfondie révèle le potentiel remarquable des techniques d'apprentissage en ensemble, spécifiquement la Random Forest, qui affiche la plus faible Erreur Quadratique Moyenne (MSE) parmi les modèles étudiés. Les caractéristiques uniques du modèle, telles que sa nature d'apprentissage séquentiel, sa manipulation habile des types de données mixtes, et sa flexibilité dans l'optimisation d'une fonction de perte différentiable arbitraire, contribuent à ses performances supérieures.

Mots-clés: Volume de trading intrajournalier, Prévision, ARMA, LSTM, TCN, Gradient Boosting, Random Forest, Analyse des séries chronologiques, Ensemble de données TAQ, Erreur Quadratique Moyenne (MSE), Apprentissage en ensemble, Marchés financiers, Indice S&P 100.

Abstract

This study provides an exhaustive comparative analysis of various forecasting methodologies, aimed at predicting intraday trading volume for stocks within the context of the S&P 100 index. Leveraging the vast Trade and Quote (TAQ) dataset, we critically assess the potential of traditional econometric models and machine learning techniques, unearthing their capabilities and limitations.

Starting with the Autoregressive Moving Average (ARMA) model, we set a benchmark for comparison with subsequent methodologies. Building upon this, we venture into the realm of advanced machine learning, examining the Long Short-Term Memory (LSTM) neural networks and Temporal Convolutional Networks (TCN), known for their proficiency in handling long-term dependencies in sequential data. Our analysis is further enriched with the exploration of ensemble learning methods like Gradient Boosting and Random Forest.

A thorough evaluation reveals the remarkable potential of ensemble learning techniques, specifically Random Forest, which yields the lowest Mean Squared Error (MSE) among the models studied. The model's unique features, such as its sequential learning nature, adept handling of mixed data types, and flexibility in optimizing an arbitrary differentiable loss function, contribute to its superior performance.

Keywords: Intraday trading volume, Forecasting, ARMA, LSTM, TCN, Gradient Boosting, Random Forest, Time-series analysis, TAQ dataset, Mean Squared Error (MSE), Ensemble learning, Financial markets, S&P 100 index.

Acknowledgements

My deepest appreciation extends to my thesis supervisors, Prof. Vincent Grégoire and Prof. Saad Ali Khan. Their insightful guidance, unwavering support, and continuous encouragement have been instrumental in the fruition of this research and in the completion of this thesis. Their expertise and dedication have significantly enriched my academic journey, and for this, I remain deeply grateful. I feel fortunate to have been under their exceptional mentorship and I wholeheartedly thank them for their invaluable assistance and support.

I am also immensely grateful to Prof. Sihem Taboubi, our MSc Academic Director, whose guidance and wisdom have greatly influenced my academic journey.

I am deeply grateful to my family. To my parents, thank you for your belief, dedication to my education, and financial support. To my little sister, your encouragement have been invaluable. This achievement is a reflection of our collective effort and love. Thank you for being there for me.

Finally, I'd want to express my unending thanks to Yasmin, my wonderful, loving wife, who has never left my side no matter what has come our way and who has always pushed me towards greater things.

Contents

List of Figures

List of Tables

List of Acronyms

WRDS	Wharton Research Data Services
TAQ	Trade and Quote
CRSP	Center for Research in Security Prices
PERMNO	Permanent Stock Identifier
NASDAQ	National Association of Securities Dealers Automated Quotations
MDH	Mixture of Distribution Hypothesis
VWAP	Volume Weighted Average Price
DVWAP	Dynamic Volume Weighted Average Price
HVWAP	Historical Volume Weighted Average Price
SVM	Support Vector Machine
ACF	Autocorrelation Function
AR	Autoregressive
MA	Moving Average
ARMA	Autoregressive Moving Average
LSTM	Long Short-Term Memory
RNNs	Recurrent Neural Networks
ReLU	Rectified Linear Unit

TCN	Temporal Convolutional Networks
BIC	Bayesian Information Criterion
AIC	Akaike Information Criterion
HQIC	Hannan-Quinn Information Criterion
MSE	Mean Squared Error

1 Introduction

Financial markets are complex systems influenced by a variety of factors, including asset prices and trading volumes. Traders frequently use tools like the Volume Weighted Average Price (VWAP), and VWAP is one of the most widely used trading algorithms by institutional investors. Understanding VWAP could potentially lead us to identify trading by institutional investors. The significance of understanding daily trading volumes extends well beyond this application. Accurate predictions facilitate better trading strategies, benefiting not just traders but also a wider audience such as individual investors and fund managers. Moreover, improved forecasts can aid in risk management by providing a clearer picture of market liquidity and volatility, and can assist regulatory bodies in market surveillance to identify irregular trading patterns.

We start our research using the Autoregressive Moving Average (ARMA) model, a well-known method in the field. It's simple yet effective in understanding patterns in data over time. This gives us a good starting point to measure how other, more complex methods perform.

After setting our foundation with the ARMA, we move on to more advanced methods used in machine learning. These methods include things like Long Short-Term Memory (LSTM) neural networks and Temporal Convolutional Networks (TCN), which are known to be very good at recognizing patterns in series of data. We also looked at techniques like Gradient Boosting and Random Forest that blend multiple models for better predictions.

Our work was influenced by various studies that have explored different aspects of market behavior. Some studies, like Kercheval and Zhang [2015], helped us understand certain trading methods. We also used ideas like the "trade information matrix" from Dixon [2018] to check out how trades are executed. Other research, such as Holden and Jacobsen [2014] and Bogousslavsky and Muravyev [2022], gave us insights into trading volumes and the effects of passive investing. With these studies, we're trying to improve our predictions on trading volumes.

Using the Trade and Quotation (TAQ) data for our research, which has lots of trading data for the S&P100 symbols. We aim to predict daily trading volumes. The TAQ dataset offers detailed trading and pricing information, making it helpful for those who want to understand market trends better. This information is especially useful for our project, where we're trying to predict daily trading volumes.

In our study, we're comparing different methods to see which one works best for predicting daily trading volumes. We're also using ideas from Kercheval and Zhang [2015] and Dixon [2018] to improve our predictions. To check how good our predictions are, we use the Mean Squared Error (MSE) measure. So far, methods like Gradient Boosting and Random Forest appear to be the most accurate.

In the subsequent sections, the paper will explain how the models were set up and tested. Additionally, the study examined how order imbalances might influence daily trading volume predictions, drawing on ideas from Bogousslavsky and Collin-Dufresne [2022]. The primary objective is to enhance understanding of daily trading volumes and contribute to the improvement of trading strategies.

1.1 Literature Review

Predicting daily trading volumes for stocks and ETFs is a challenging job, and it's crucial for methods like the VWAP trading strategies. To get a handle on this challenge, it's important to understand the various theories that explain the workings of financial trading. One research that stands out is by Andersen [1996]. Andersen studied how trading volumes might be affected by things like the spread of new information and the immediate need to buy or sell assets. In his work, Andersen builds upon what's called the microstructure theory. This theory suggests that the amount of trading activity is heavily influenced by the gaps in information between traders and their immediate needs to buy or sell. This means is that when traders have different bits of news or information, and they need to act quickly, this can drive up the number of trades in a day. A central part of Andersen's work is his take on the "Mixture of Distribution Hypothesis" (MDH). While the usual idea of MDH links the amount of new information directly to changes in prices, Andersen offers an alternate explanation or modification to the MDH. He posits that the flow of this new information, which can vary in its intensity, is a major factor affecting how much trading takes place. This new angle, different from many traditional views, could offer a deeper insight into how markets move and can be very useful for predicting trading volumes. One of the key findings from Andersen's study is that his approach can better explain real-world trading behaviors than the original MDH approach. This suggests that his model has significant practical use, especially for those attempting to foresee trading volumes.

In their research, Gourieroux, Jasiak, and Le Fol [1999] examined trading patterns throughout a single trading day and introduce duration-based measurements for individual stocks and portfolios. Central to these measurements are "weighted durations," which refer to the time needed to sell (or purchase) a specified number of shares at a consistent price. These weighted durations can be seen as liquidity indicators, capturing the interactions among intra-trade durations, transaction volumes, and stock prices. By exploring how assets might substitute for one another and by observing the interconnectedness of trading activities within portfolio components, a multivariate approach becomes instrumental. This approach provides clarity on the liquidity of portfolio assets - how swiftly they can be traded. Moreover, Gourieroux et al. [1999] not only discussed liquidity costs and volatility but also ranked liquidity across different assets. They used multivariate metrics to examine relationships between assets, showing if assets can replace one another or if they complement each other. In addition to these insights, the study also presented an alternative to the conventional ARCH specification, for a more integrated modeling of duration, volume, and price processes.

Lobato and Velasco [2000] investigated the estimation of trading volume and volatility of stock market activities over long periods without removing trend patterns from the data. Instead of traditional methods, this study uses a frequency domain approach with tapering. This offers a fresh way to understand the patterns of stock market activity. This study introduces a formula for estimating the memory attributes of a non-stable vector process. This new method helps in analyzing trading volume and volatility, which are both seen as non-stable vector processes in this research. This method is useful because it gives a deeper understanding of financial market movements, especially when common time-focused analyses might fall short. Lobato and Velasco [2000] took a closer look at the patterns of trading volumes for stocks in the Dow Jones Industrial Average index. The findings show that stock market activities tend to repeat over time. This suggests that trading patterns are consistent, which is important for those involved in market predictions and risk evaluations. An interesting finding from this study is that while both volatility and trading volume have patterns that persist over time, these patterns aren't the same for both. This finding challenges the idea that these two key market factors are driven by the same things. Therefore, this study highlights the need to see volume and volatility as different when trying to predict or understand market movements.

Understanding the effectiveness of trading methods is essential for comprehending market dynamics. Madhavan [2002] discussed the application of volume-weighted average pricing (VWAP) as a metric for evaluating traders. Though VWAP is commonly used, Madhavan [2002] suggested that its performance can vary based on factors such as the chosen benchmark, the duration of trading, and the place of execution. In Madhavan [2002]'s findings, these variables can significantly impact aspects like transaction costs, associated risks, and potential returns. This study questions the validity of a uniform approach to assess trading strategies, highlighting the factors that might affect trading outcomes. While methods like VWAP can appear straightforward, their practical application can be challenging due to inherent market complexities. For instance, a VWAP strategy might reduce transaction costs but could introduce heightened market risks or lead to smaller returns. Madhavan [2002]'s research underscores the need for a thorough evaluation of trading techniques, emphasizing the importance of balancing theoretical understanding with real-world dynamics. It also highlights the need to tailor trading approaches based on individual objectives, market conditions, and the specific trading context.

Białkowski, Darolles, and Le Fol [2008] provided an approach to model intraday volume dynamics, significantly lowering execution risk in VWAP (volume weighted average price) orders. The models are based on separating the total volume of trades into two components: one that accounts for fluctuations in volume due to general market movements and another that describes the volume pattern for a certain stock. In this case, Białkowski et al. [2008] used a cross-historical average for the first component of volume and ARMA and SETAR models to illustrate the dynamics of the second component. These forecasts are then integrated into a pricing structure that tracks market benchmarks. Additionally, Białkowski et al. [2008] showed that simple time-series models can be useful for predicting volume. Their method can also be applied to VWAP strategies to reduce the difference from the actual VWAP. This can help in reducing the risk and cost related to using these orders. The decrease can be over 10%, and in some cases, even up to 50% for certain companies, based on how they handle the data.

Bouchaud, Farmer, and Lillo [2009] discussed the microstructure approach to how prices are set, adding to our understanding of market behavior. They introduced the concept of order fragmentation, where large buy and sell orders are divided into smaller parts because of limited market liquidity. This leads to a persistent pattern in order flow that influences market outcomes and price changes. Bouchaud et al. [2009] suggested that market participants often work with partial information, which can cause unintended price effects. Instead of focusing on external news, they emphasize the influence of supply and demand.

Manchaldore, Palit, and Soloviev [2010] applied wavelet decomposition to study past intraday volume. Their model has two main components: diffusion and jumps, using data processed with wavelets. They started with a simple constant volatility model and attribute volume signal changes to the absence of a regular daily pattern. The volume consists of regular background noise and occasional large jumps. The average noise over many days often forms a U-shaped pattern. Interestingly, there's no consistent pattern for large changes in volume. Their results show familiar patterns like the U-shape and frequent volume changes. Manchaldore et al. [2010] suggest their model might help improve various trading strategies. For instance, it can give brokers a sense of additional costs from unexpected volume changes, which can affect trading fees.

Dealing with noise during intraday VWAP trading can be a challenge often overlooked in algorithmic trading systems. However, Humphery-Jenner [2011] addressed this issue with a new approach. They introduced the Dynamic VWAP (DVWAP) to help traders adjust to unexpected news, especially given the rapid spread of information in volatile markets. Unlike the more traditional Historical VWAP (HVWAP), the DVWAP adjusts trading frequency based on market volume. It trades more when the volume is high and less when it's low. This approach makes DVWAP more adaptable with daily volume changes compared to HVWAP, aiming to minimize errors and enhance trading effectiveness.

Brownlees, Cipollini, and Gallo [2011] introduced a dynamic model designed to capture the behavior of trading volumes relative to outstanding shares. This model incorporates both daily and intraday perspectives. Using the Generalised Method of Moments, the model's parameters are effectively estimated. Upon testing with three primary ETFs, Brownlees et al. [2011] found that both consistent and adaptive VWAP strategies are more predictive than a basic rolling mean approach for intraday volume forecasts.

Taking into account daily and (periodic and nonperiodic) intra-daily time perspectives, Holden and Jacobsen [2014] provided a dynamic model with various components reflecting the behavior of traded volumes (relative to outstanding shares). The Generalized Method of Moments is used to estimate the parameters of this Component Multiplicative Error Model with a single calculation. In an out-of-sample forecasting exercise, the application to three major ETFs demonstrates that the static and dynamic VWAP replication strategies often outperform a regularly used naive method of rolling means for intra-daily volumes.

Kercheval and Zhang [2015] explored the dynamics of high-frequency limit order books in financial equity markets, aiming for real-time predictions of metrics like mid-price movement and price spread crossing. The study employed multi-class support vector machines to derive a model from limit order book data characterized by attributes such as price and volume. The application of multi-class Support Vector Machine (SVM) algorithms for short-term predictions represents a notable contribution to the field. By analyzing real data from the NASDAQ, Kercheval and Zhang [2015] found that their models demonstrate reliable predictive accuracy. Additionally, trading strategies based on these predictions have shown promising results in simulations.

Wang and Wang [2016] investigated volatility forecasting, given its relevance to derivative pricing and risk management. While prior research suggests that implied volatility from option pricing serves as a robust predictor of future volatility, Wang and Wang [2016] focused on intraday patterns of implied volatility informed by high-frequency data. The research raises the possibility that intraday implied volatility, rather than daily closing levels, may offer a more precise prediction of future volatility. This emphasis on the intraday dynamics of trade provides a nuanced perspective on volatility forecasting, challenging the reliance on end-of-day data.

Intraday stock price curves and intraday volatility curves are two examples of the types of curves often used to represent financial data, both of which may be examined sequentially through time. These curves may be interpreted as a sequence of functions over time, and their behavior can be studied using dense, regularly spaced grids. The nature of high-dimensional data presents challenges from a statistical viewpoint because of the so-called curse of dimensionality; however, it also provides opportunities to analyze a rich source of information, allowing for a better grasp of the rapid changes that occur over relatively brief periods. Shang [2017] took a time series forecasting approach, proposing many statistical techniques that may predict intraday market returns one day in advance. An analysis of intraday S&P 500 index returns, measured every 5 minutes, was conducted to validate the effectiveness of these forecasting methods.

After decomposing trade volume percentages into two primary components—namely, the regular intraday volume pattern and the residual term (which accounts for outlier fluctuations)—Liu and Lai [2017] proposed a dynamic model for intraday volume percentage forecasting. The study finds that the regular component can be anticipated using a moving average of the prior day's volume percentage. Meanwhile, for the residual component, a support vector machine (SVM) proves effective, especially when the input pattern is binary. This prediction incorporates both current and previous day volume percentages spanning identical time intervals. An out-of-sample test on gold and S&P 500 futures indicates that this dynamic methodology outperforms the benchmark approach, which relies on historical averages. This method demonstrates an accuracy improvement of up to 14%, underlining the validity of the volume decomposition and emphasizing the distinct nature of its components. Furthermore, employing a dynamic SVM for volume percentage predictions bolsters the efficacy of the VWAP strategy, leading to enhanced forecast precision.

In the context of tick-level predictive classifiers, Dixon [2018] introduced the "trade information matrix" as an innovative tool to assess potential gains or losses. This matrix considers execution constraints, including the likelihood of order fill and position-based trade rules, to attribute projected profit and loss values to both correct and incorrect forecasts. To construct this matrix, Dixon [2018] employed Level II E-mini S&P 500 futures performance data. This matrix serves as a data-driven evaluation framework for market-making strategies aimed at refining trading tactics. Moreover, the study introduces a trade execution model resembling exchange matching engines. This model estimates an order's position in the queue, accounting for incoming market orders and departing limit orders. The model's predictive efficacy is enhanced by embracing actual exchange dynamics, aligning it more closely with real-world trading scenarios.

In Bogousslavsky and Muravyev [2022], the final auction's importance is mentioned, focusing on the rise of passive investing and its potential effects. Passive investing might affect asset values in different ways, leading to changes in market behavior. A main point in this research is the link between the volume at the close and the closing prices.

The author found that ETF ownership and institutional rebalancing mainly drive the closing volume. Interestingly, the study reveals that the closing price doesn't give much information about the earlier midquote, but instead adds confusion. This suggests that we might need to reconsider the traditional importance of closing prices. Bogousslavsky and Muravyev [2022] highlighted concerns about relying too much on closing prices and points out issues with indexing. This adds to the ongoing discussion on passive investing, emphasizing the role of closing volumes and prices.

Using a large dataset from U.S. equities after decimalization, Bogousslavsky and Collin-Dufresne [2022] looked into liquidity patterns. The study uses a measure of order imbalance changes to get a sense of how liquidity providers might be at risk. This measure is based on a model that helps understand these risks more deeply. The findings highlight that quick changes in order imbalances are crucial for liquidity. The study also considers the effects of order imbalance changes on stock returns, helping us see their broader influence on the stock market.

The rest of the thesis is organized as follows: Section 2 describes the data collection adopted in the present study, section 3 describes the methodologies adopted in the present study, including Autoregressive Moving Average (ARMA), Long Short-Term Memory networks (LSTM), Temporal Convolutional Network (TCN), Gradient Boosting, and Random Forest, section 4 details the empirical results, and section 5 concludes.

2 Data Collection

In this section, we discuss the data collection and preprocessing procedures we conduct to pursue our research goal - forecasting the intraday trading volume for stocks. We collect the data for our analysis from two reputable databases: the Trade and Quote (TAQ) dataset and the Center for Research in Security Prices (CRSP).

2.1 Trade and Quote Database

The TAQ (Trade and Quote) database is a comprehensive repository that encapsulates historical intraday trade and quotation data, timestamped to the millisecond, for all securities listed and traded on various platforms. This includes data from the New York Stock Exchange, the Nasdaq Stock Market, and other stock markets in the United States that make up the National Market System. The TAQ database contains all trade and consolidated quotes from US public exchanges.

For our study, we utilize the TAQ Daily Product from the Trade and Quote (TAQ) database for the year 2020. This database includes flags that specify opening and closing auction trades. The data covers trading activities from 9:30 AM to 4:00 PM (Eastern Time). We source our data from the Wharton Research Data Services (WRDS) Cloud server. Our chosen dataset provides accurate timestamps, which is crucial for our interval-based analysis.

Specifically, for our analysis, we extracted trade data at 5-minute intervals. The relevant columns from the TAQ Daily Product's trade files are:

- "Date" - Date of the trade.
- "Time" - Exact time of the trade, often in milliseconds from the start of the day.
- "Size" - Number of shares traded.
- "*tr_seqnum*" - Unique identifier for the trade.

Table 1 showcases a segment of our extracted data for Amazon, summarized at 5-minute intervals:

Table 1: In this table, we present a snapshot of trading data specific to Amazon, collected from our dataset. This data is structured to represent trades at 5-minute intervals, detailing the timestamp, volume of trades, sequence numbers, and the associated stock symbol. The segment aims to provide a concise view into the trading dynamics of Amazon within a short duration.

Timestamp	**Size**	**tr_seqnum**	**Symbol**
2020-01-02 9:30	366,158	259,267,208	AMZN
2020-01-02 9:35	104,856	316,013,429	AMZN
2020-01-02 9:40	78,865	359,074,139	AMZN
2020-01-02 9:45	59,485	419,532,793	AMZN
2020-01-02 9:50	71,287	571,326,630	AMZN

2.2 Centre for Research in Security Prices Database

To complement our TAQ dataset, we utilize the CRSP (Centre for Research in Security Prices) database. CRSP offers extensive information on securities, but for the purpose of our research, we specifically tap into its data on the number of outstanding shares, known as "shrout". This allows us to normalize trading volume and calculate turnover. A key feature of CRSP is its unique identifiers for each security. We employ the CRSP Permanent Stock Identifier (PERMNO) to maintain consistency and accuracy when merging data, especially crucial when handling vast datasets over multiple time frames.

2.3 Merging Datasets

The TAQ and CRSP datasets were merged to create a uniform dataset with trading volume and shrout for each stock. This merge facilitated the computation of the mean shrout, derived solely from the test data.

2.4 Exploratory Data Analysis

The key facets of the exploratory process include the normalisation of the data, the segmentation of the data into progressive temporal intervals, and the autocorrelation function (ACF) analysis. These elements, described in the following sections, ensure our data is consistent across different stocks, organized in time order, and analyzed to highlight key time-related patterns.

2.4.1 Data Normalisation: Adjusting for Individual Stock Differences

For the normalization of trading volumes, Shares Outstanding (SHROUT) is involved in the process. Specifically, for each stock, we retrieve the average SHROUT value from the CRSP dataset. We then take the average trading volume for each time bin over our sample period and divide this by the average SHROUT value to obtain a normalized trading volume. This can be expressed as:

$$\text{Normalized Volume } = \frac{\text{Volume}_{\text{ time bin , specific day}}}{\text{TSA} \times \text{ Average SHROUT}}$$

Where TSA is the time-specific average calculated as:

$$\text{Time-specific Average (TSA) } = \frac{\sum_{i=1}^{n} \text{ Volume}_{\text{ time bin },i}}{n}$$

This method helps reduce differences caused by different trading volumes in stocks. It makes sure the data is consistent across stocks so our model isn't overly affected by stocks with large volumes and can recognize patterns in all trading volumes.

2.4.2 Data Segmentation Through Progressive Temporal Intervals

In the context of our study, we employ a unique technique to segment the data using a progressive temporal intervals approach. This method, while drawing from the conventional rolling window technique, distinguishes itself in how it is applied.

Let N be the total number of data points in the time series dataset. Our approach segments the data into successive temporal intervals, each spanning an equivalent length of a month. For a given time series dataset represented as $d_1, d_2, d_3, ..., d_N$, and a temporal interval size, T, similar to a month's duration, the segmentation process is outlined as follows:

For $i = 1$ to $N - 2T + 1$:

- 1. The training phase is performed on the temporal interval, $[d_i, d_{i+1}, ..., d_{i+T-1}]$.
- 2. The testing phase is conducted on the subsequent temporal interval, $[d_{i+T}, d_{i+T+1}, ..., d_{i+2T-1}]$.

Consequently, we generate training and testing intervals that mirror monthly spans:

- Interval 1: Training Interval $d_1, d_2, ..., d_T$, Testing Interval: $d_{T+1}, d_{T+2}, ..., d_{2T}$.
- Interval 2: Training Interval $d_{T+1}, d_{T+2}, ..., d_{2T}$, Testing Interval: $d_{2T+1}, d_{2T+2}, ..., d_{3T}$.
- ...
- Interval $N - 2T$: Training Interval $d_{N-2T+1}, d_{N-2T+2}, ..., d_{N-T}$, Testing Interval: $d_{N-T+1}, d_{N-T+2}, ..., d_N$.

We carry out this sequence of advancing temporal intervals throughout the whole dataset. This method maintains an environment where the model is trained and tested on separate, non-overlapping temporal subsets, thereby preserving the temporal sequence and homogeneity in the data.

This is crucial, as temporal continuity requires an orderly progression of data points. Any disruption may distort the intrinsic patterns that the model attempts to learn. Moreover, the division into non-overlapping intervals guarantees a degree of independence between the training and testing sets, mitigating the risk of overfitting and data leakage.

Therefore, through the implementation of our progressive temporal intervals method, we not only preserve the temporal structure of the dataset, but also enhance our model's robustness by minimizing the potential for overfitting and data leakage. This places our model in a strong position to accurately predict intraday trading volumes, underpinning the analytical depth and significance of our study.

2.4.3 Autocorrelation Function (ACF) Analysis

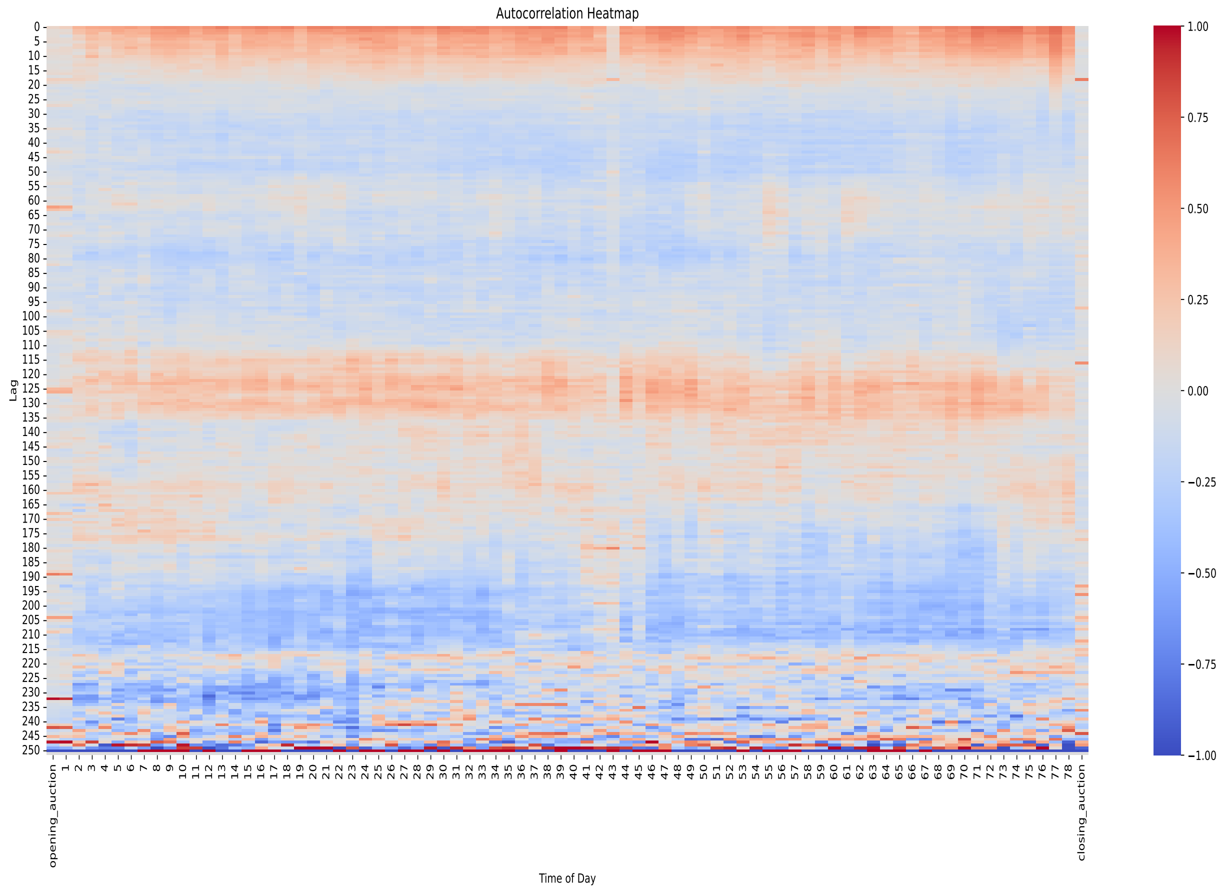
The Autocorrelation Function (ACF) is a tool that calculates the autocorrelation between values in a time series at various points in time. ACF is used in time series analysis to measure the linear predictability of the current value based on its past values. The ACF of a time series Y_t is defined as:

$$\rho_k = \frac{\operatorname{Cov}(Y_t, Y_{t-k})}{\sqrt{\operatorname{Var}(Y_t) \cdot \operatorname{Var}(Y_{t-k})}}$$

Where ρ_k is the autocorrelation function at lag k, $Cov(Y_t, Y_{t-k})$ is the covariance between the time series and its lag k, and $Var(Y_t)$ and $Var(Y_{t-k})$ are the variances of the time series and its lag k respectively.

Given the time-structured nature of our dataset, where observations from each symbol's intraday trading volume are collected at 5-minute intervals, an ACF analysis is particularly relevant.

Figure 1: The heatmap representation of the average trading volume at five-minute intervals (9:30 AM to 4:00 PM Eastern Time) over the trading year of 2020. The data covers 78 time buckets per trading day, with distinct columns for opening and closing auctions. The series facilitates an in-depth analysis of intra-day trading volume dynamics and their correlation with pivotal auction moments. A maximum lag of 252 trading days is considered for analysis.



Each data entry contains the average trading volume for all symbols on a specific time bucket, recorded at five-minute intervals from 9:30 AM to 4:00 PM Eastern Time (78 buckets), corresponding to the normal U.S. stock market trading days. This allows for granular analysis of the intra-day trading volume dynamics.

Additionally, the data set includes separate columns for the opening and closing auctions. Opening and closing auctions are key moments in the trading day when market participants can submit orders to be executed at the market opening or closing. By including these in the data set, it is possible to analyze how trading volumes at these pivotal moments correlate with past trading volumes.

The time series data spans a period of one year, giving a maximum lag of 252 days for analysis. The lag value of 252 days corresponds to the number of trading days in a year, assuming a five-day trading week and excluding public holidays.

3 Methods

The selection of methods in this study is motivated by the need for a comprehensive evaluation of predictive accuracy in intraday trading volumes. ARMA serves as the baseline model, widely recognized for its reliability in time-series forecasting. However, its limitations in capturing complex, non-linear relationships motivate the inclusion of neural networks, we also consider neural networks, specifically Long Short-Term, as cited by Cao, Li, and Li [2019]. Additionally, Temporal Convolutional Networks (TCN) are explored due to their capacity to handle long sequences, positioning them as an alternative to LSTMs, as described by Torres, Hadjout, Sebaa, Martínez-Álvarez, and Troncoso [2021]. To offer a comprehensive perspective, ensemble methods, namely Gradient Boosting as discussed by Papadopoulos and Karakatsanis [2015] and Random Forest as presented by Srinu Vasarao and Chakkaravarthy [2022], are also incorporated. These methods are renowned for their high accuracy and resistance to overfitting.

This section explains the strengths and weaknesses of our chosen methods as they deal with the financial market. We use the detailed data from S&P100 symbols to accurately predict intraday trading volume. We also evaluate how well each method works, looking at their overall accuracy and effectiveness.

We examine the intraday dynamics of trade volume using a four-pronged strategy to evaluate the accuracy of forecasts. The Autoregressive Moving Average (ARMA) is the cornerstone of our work, the standard against which more complex models are compared. We investigate neural networks using the memory-efficient Long Short-Term Memory (LSTM), and further, we investigate the potential of Temporal Convolutional Networks (TCN), which are well-known for their capacity to deal with long sequences. In order to round up our comparison, we add two additional methods from the realm of robust ensemble learning: Gradient Boosting and Random Forest.

Our study focuses on intraday trading volume, an important measure of market activity that guides investors and traders. Predicting these volumes accurately can help with making informed decisions and finding trading opportunities.

Guided by robust metrics like the Mean Squared Error (MSE), we map the landscape of reliability and accuracy of these forecasts, parsing through the data's underlying characteristics.

In the following sections, we outline how we implement and assess each method. We cover all steps, from data preprocessing and model training to hyperparameter tuning and evaluation. By applying these methods to intraday trading volume data, we aim to determine their accuracy in predictions, which is essential for making decisions in the financial markets.

3.1 Autoregressive Moving Average (ARMA) Model

This study uses the Autoregressive Moving Average (ARMA) model, a cornerstone of time series analysis in the field of econometrics, to analyse and forecast the financial series of interest; Box, Jenkins, Reinsel, and Ljung [2015]. The ARMA model combines the Autoregressive (AR) and Moving Average (MA) models, offering a powerful tool for modelling and forecasting realizations of the time series.

3.1.1 Autoregressive (AR) Component

The AR component of an ARMA model is a parametric representation, whereby a variable is regressed on its own lagged (i.e., previous) values. An AR(p) model, where p signifies the order of the autoregressive process, can be formally described as:

$$\mathbf{X}_t = \mathbf{c} + \sum_{i=1}^{p} \phi_i \mathbf{X}_{t-i} + \varepsilon_t, \tag{1}$$

- $\mathbf{X}_t$ is the variable of interest at time t,
- $\mathbf{c}$ is a constant,
- ϕ_i represents the coefficients of the model,
- $\mathbf{X}_{t-i}$ is the value of $\mathbf{X}$ at a prior time period.

3.1.2 Moving Average (MA) Component

The MA component of an ARMA model is simply a model where the variable of interest is regressed on the residual errors from prior predictions. Formally, an MA(q) model is expressed as follows:

$$\mathbf{X}_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}, \tag{2}$$

where:

- $\mathbf{X}_t$ is the variable of interest at time t,
- μ is the expectation of $\mathbf{X}_t$,
- ε_t is the error term at time t,
- θ_i represents the coefficients of the model,
- ε_{t-i} is the error term at a previous time period,
- q is the number of lagged errors included in the model.

3.1.3 ARMA Model

An ARMA(p, q) model is created by combining the AR and MA components. It may be formalised as follows:

$$\mathbf{X}_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i \mathbf{X}_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{3}$$

To identify the optimal order for each component of our ARMA model, we employ a systematic search over the model's hyperparameters, choosing the combination that minimizes the Bayesian Information Criterion (BIC). Formally, the BIC is calculated as:

$$BIC = \ln(n)k - 2\ln(L), \tag{4}$$

where:

- n is the number of observations,
- k is the number of parameters in the model,
- L is the model likelihood.

In addition to BIC, we also calculate the Akaike Information Criterion (AIC) and the Hannan-Quinn Information Criterion (HQIC) for comparative purposes:

$$AIC = 2k - 2\ln(L) \tag{5}$$

$$HQIC = -2\ln(L) + 2k\ln(\ln(n)) \tag{6}$$

The Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC) are used to estimate the lag order p, depending on which offers a more frugal model that nonetheless accurately captures the underlying structure of the data.

The MSE minimization implies the model has the highest prediction accuracy with the lowest degree of error variance, hence the best ARMA model will be the one that minimises the MSE on the out-of-sample data.

Finally, we utilize the Wald Test to check the statistical significance of our model's coefficients. The Wald Statistic is calculated as:

$$W = \left(\frac{\hat{\beta}}{se(\hat{\beta})}\right)^2, \tag{7}$$

where $\hat{\beta}$ is the estimated parameter and $se(\hat{\beta})$

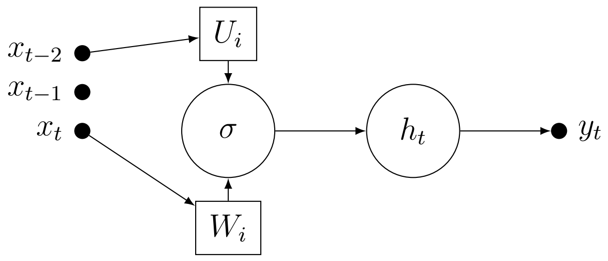
The Wald Statistic follows a chi-square distribution, allowing us to test the null hypothesis that a particular parameter is zero.

3.2 Long Short-Term Memory networks

After using the ARMA model as our initial benchmark for time-series forecasting, we sought to investigate more sophisticated and powerful machine-learning models that could outperform ARMA in predictive performance. This led us to explore Recurrent Neural Networks (RNNs), specifically, Long Short-Term Memory networks (LSTMs).

The LSTM model, which Hochreiter and Schmidhuber [1997] initially presented in 1997, significantly improves temporal sequences' modeling. LSTM networks' unique architecture was created to get over the long-term reliance issue that plagues traditional RNNs. As the sequence becomes longer, this issue, also known as vanishing gradients, often makes it challenging for RNNs to acquire and keep knowledge from previous time steps. LSTMs are particularly derived to recall information for long periods of time and can thus record patterns over a wide time range resulting from their distinctive structure, which consists of a memory cell, which is a complex arrangement of four linked layers.

Figure 2: Schematic depiction of an LSTM (Long Short-Term Memory) unit's single timestep. The diagram illustrates the path of input values x_{t-2}, x_{t-1}, and x_t, processed via weight matrices U_i and W_i, activated through a sigmoid function (σ), which subsequently generate the hidden state h_t and output y_t



The representation of an LSTM cell as shown above illustrates its operation and how it mitigates the vanishing gradient problem endemic to traditional RNNs. The cell is characterized by several components interacting together: x_t denotes the input at time t, and h_{t-1} signifies the prior hidden state. The weights W and U symbolize the input-to-state and state-to-state transition matrices, correspondingly, for each gate.

The cell employs an input gate i_t, a forget gate f_t, and an output gate o_t, each controlled by a sigmoid activation function (σ), to modulate information flow. The state of the cell is updated via a hyperbolic tangent function (tanh), denoted by C_t, that accounts for the influence of the input, forget, and output gates. The output at time t, denoted as h_t, is determined after the cell state has interacted with the output gate.

The recurrent nature of this architecture enables the LSTM to capture long-term dependencies and sustain temporal information across periods, making it particularly suitable for time-series forecasting.

3.2.1 Regularization Techniques

Regularization, which prevents overfitting by adding a penalty term to the loss function, is a crucial part of efficient model training. By discouraging the model from keeping unnecessary weights, the penalty term simplifies the model and improves its generalization abilities; Neumaier [1998].

- $L1$ Regularization: $L1$ Regularization, commonly known as Lasso regularisation, incorporates an absolute-value-proportional penalty term into the loss function. This can result in a sparse solution, which may be useful for feature selection because many weights will be absolutely zero.

  If L is the loss function, λ is the regularization parameter, and w represents the model weights, the $L1$ regularized loss function is:

  $$L_{L1} = L + \lambda \sum |w|$$

  In the above formula, λ is a hyperparameter that controls the strength of the regularization term. Larger values of λ lead to stronger regularization. The absolute value operation on weights leads to sparsity, effectively performing feature selection.

- $L2$ Regularization: Also known as Ridge regularization, $L2$ regularization adds a penalty term to the loss function that is proportional to the square of the weights. This tends to distribute weights more evenly and leads to a more balanced model.

  The $L2$ regularized loss function is:

  $$L_{L2} = L + \lambda \sum w^2$$

  In the above formula, λ is a hyperparameter that controls the strength of the regularization term. Larger values of λ lead to stronger regularization. The square operation on weights encourages smaller and more evenly distributed weights, helping the model to be more balanced and stable.

3.2.2 Early Stopping

Early stopping is a form of implicit regularization, introducing a complexity penalty to the loss function that we aim to minimize. It prevents overfitting by stopping the training process when the model starts to perform worse on the validation data, even though it might still be improving on the training data; Yao, Rosasco, and Caponnetto [2007].

In the context of optimization, the inclusion of early stopping can be conceptualized as adding a regularization term to the loss function:

$$L = L_D(\theta) + \lambda\Omega(\theta)$$

In the formula, L signifies the total loss, $L_D(\theta)$ represents the empirical loss on the training data, and $\Omega(\theta)$ is the regularization term. Here, θ symbolizes the parameters of the model, and λ serves as a hyperparameter that determines the trade-off between the empirical loss and the complexity penalty. The application of early stopping in training effectively controls the complexity of the model and thus the $\Omega(\theta)$ term, without the need for explicit regularization.

3.2.3 Adam Optimizer

We develope our computational model by fine-tuning the LSTM network parameters. For this, we used the Adam optimizer, an optimization method introduced by Kingma and Ba [2014]. This algorithm is popular in the deep learning community because it's efficient and doesn't require much manual adjustment of the learning rate.

At the core of Adam lies its capacity to calculate adaptive learning rates for individual parameters. This capability is realized through the estimation of the first and second moments of the gradients, which serve as statistical measures to enhance parameter updates. This technique combines features from two known extensions of stochastic gradient descent: AdaGrad and RMSProp. These contribute to Adam's efficacy, where AdaGrad is recognized for its performance with sparse gradients and RMSProp excels in online and non-stationary settings.

The Adam optimizer executes parameter updates as per the following procedure:

$$
\begin{aligned}
m_t &= \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \\
v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_{t+1} &= \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}
\end{aligned}
$$

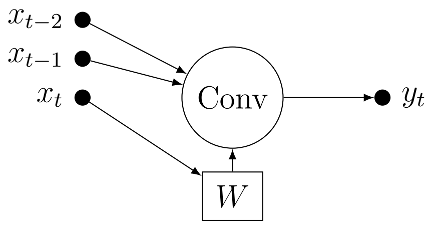
where:

θ_t denotes the parameters under optimization, g_t signifies the gradient at timestep t, m_t and v_t are statistical estimates of the first moment (mean) and the second moment (uncentered variance) of the gradients, respectively, $\hat{m}_t$ and $\hat{v}_t$ denote bias-corrected versions of m_t and v_t, α is the learning rate, β_1 and β_2 are exponential decay rates for the moment estimates, ϵ is a scalar of a minimal value to prevent any division by zero. Adam, an abbreviation of "adaptive moment estimation," is characterized by its straightforward implementation, computational efficiency, low memory requirement, and invariance to diagonal rescaling of gradients. It is optimally designed to handle large-scale problems with substantial data or parameters.

3.3 Temporal Convolutional Network (TCN)

Temporal Convolutional Networks (TCN), proposed by Bai et al. Bai, Kolter, and Koltun [2018], represent a class of deep learning models that capitalize on the 1-D convolution operation for sequence processing. Our model employs a TCN for predicting stock market behavior using historical data. The following elaborates the mathematical underpinnings behind the implemented TCN model.

Figure 3: Illustration of a Temporal Convolutional Network (TCN) block: A single-layer architecture with a kernel size of 2. The convolution operation (Conv) utilizes inputs from three time-steps (x_{t-2}, x_{t-1}, and x_t) to produce the output y_t.



The TCN model is defined in a sequential manner using Keras, with an architecture primarily composed of a TCN layer followed by a dense layer. TCN can be seen as a function $f(x;\theta)$, where x denotes the input sequence, and θ represents the set of all learnable parameters in the model.

The most vital component of the TCN model, the TCN layer, leverages a series of dilated convolutions that efficiently capture long-range dependencies. The dilated convolution is formulated as:

$$y(t) = \sum_{i=0}^{k} f\left(x(t - d \cdot i); \theta_i\right), \tag{8}$$

where $y(t)$ is the output at time t, $f()$ represents the activation function (ReLU in our case), $x(t)$ is the input at time t, d is the dilation factor, i is the index of the kernel, k is the kernel size, and θ_i denotes the trainable parameters of the kernel at index i.

The padding is set to "causal", meaning the network is causal and the output at time t is convolved only with elements from time t and earlier in the previous layer. Hence, the output sequence does not peek into the future and respects the temporal order of the data.

Following the TCN layer, we apply a dense layer, which performs the operation:

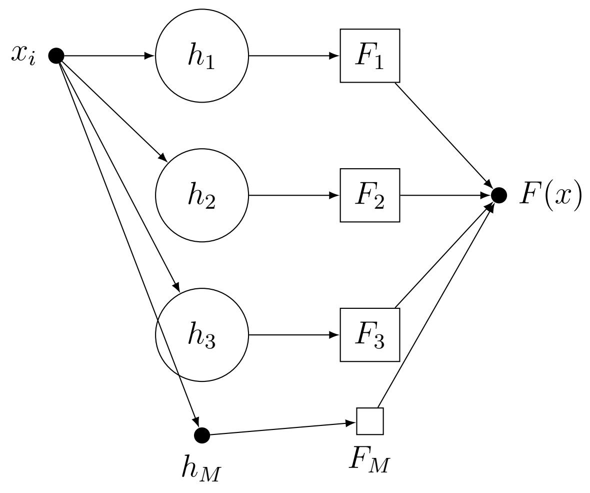
$$y = g(W \cdot x + b)$$

where W is the weight matrix, b is the bias vector, and $g()$ is the activation function (a linear function by default), all of which are learned during training.

3.4 Gradient Boosting

Gradient boosting, introduced by Friedman [2001], is a machine learning technique for regression and classification problems, which builds an ensemble model by optimizing a loss function in a stage-wise manner. It is recognized for its effectiveness in dealing with heterogeneous data and its inherent feature selection capability.

Figure 4: Representation of the gradient boosting algorithm. Each weak learner, denoted by h_m, receives input x_i and contributes to the cumulative strong learner, represented by F_m, at stage m. The final prediction is given by the aggregation of all F_m learners as $F(x)$.



Gradient boosting operates in the function space, treating function estimation as a numerical optimization problem. This contrasts with traditional machine learning approaches that optimize in the parameter space. The algorithm leverages the concept of steepest-descent minimization to construct an additive model, which consists of multiple weak learners that contribute towards the final model's decision-making process.

$$\hat{F} = \arg\min_F E_{y,\mathbf{x}} L(y, F(\mathbf{x})) = \arg\min_F E_{\mathbf{x}} \left[E_y(L(y, F(\mathbf{x})))|\mathbf{x} \right]. \tag{9}$$

Here, F is the final model composed of several weak learners, $\mathbf{x}_i$ denotes the feature vector for the i-th instance, y_i is the corresponding target value, and L is a differentiable loss function that measures the discrepancy between the target value y_i and the model's prediction $F(\mathbf{x}_i)$.

The weak learners are represented as $h_m(\mathbf{x}; \theta_m)$, where $\mathbf{x}$ denotes the input features and θ_m represents the parameters for the m-th weak learner. In the context of gradient boosting, these weak learners are generally decision trees, though they can be any model that provides a good fit to the residuals.

In each iteration of the boosting process, a weak learner is fitted to the negative gradient of the loss function with respect to the model's prediction. For the m-th iteration, this is expressed as:

$$r_{im} = -\left[\frac{\partial L\left(y_i, F\left(\mathbf{x}_i\right)\right)}{\partial F\left(\mathbf{x}_i\right)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \tag{10}$$

Here, y_i is the target value for the i-th instance, $F(\mathbf{x}_i)$ is the current prediction for the i-th instance, and L is a differentiable loss function that measures the discrepancy between the target and the prediction.

The parameters θ_m for the m-th weak learner are obtained by solving the following least-squares problem:

$$\theta_m = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(r_{im} - h_m\left(\mathbf{x}_i; \theta\right)\right)^2 \tag{11}$$

In essence, this equation is fitting the m-th weak learner to the residuals, which represent the direction of steepest descent in the function space, and finding the parameters θ that minimize the squared difference between the residuals and the weak learner's predictions.

After fitting the m-th weak learner, its contribution is added to the current prediction, weighted by a factor ρ_m:

$$\rho_m = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \rho h_m\left(\mathbf{x}_i; \theta_m\right)\right) \tag{12}$$

The weight ρ_m is computed via line search, a numerical optimization technique, by solving:

$$F(\mathbf{x}) = F_0(\mathbf{x}) + \sum_{m=1}^{M} \rho_m h_m\left(\mathbf{x}; \theta_m\right) \tag{13}$$

In this equation, $F_0(\mathbf{x})$ is the initial prediction, and the summation represents the cumulative contribution of all weak learners. Each weak learner is fitted to the residuals of the previous stage and contributes to the final prediction according to the optimized weight ρ_m.

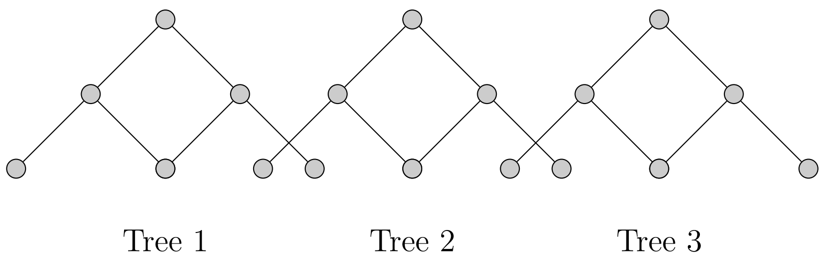
In this manner, the gradient boosting algorithm successively improves its prediction by adding new weak learners that focus on the most challenging instances. As such, it can construct a powerful model from an ensemble of relatively simple and weak models.

3.5 Random Forest

Random forests, initially introduced by Breiman [2001], are another popular ensemble learning method for both regression and classification tasks. They combine numerous decision trees to generate more robust and accurate predictions. Despite their simplicity, random forests have shown excellent performance on various machine learning tasks, often comparable to or even surpassing more complex models.

Random Forests make use of a collection of decision trees, where each tree is independently constructed via bootstrapping, i.e., resampling the data with replacement. Each decision tree is built using a randomly selected subset of features at each split. This results in an ensemble of trees that are uncorrelated with each other, making the model less prone to overfitting.

Figure 5: Illustration of the random forest algorithm with three depicted decision trees. Each tree in the diagram is an individual decision tree that contributes to the ensemble within the forest.



This figure demonstrates a random forest composed of three trees, each of which represents an individual decision tree within the forest. Every node in the tree signifies a decision based on feature values, and every edge represents the outcome of a decision at the parent node. Note that each tree is built independently, with different subsets of the data and features.

Each decision tree in the forest is a function $T_m(\mathbf{x}; \Theta_m)$ where $\mathbf{x}$ represents the feature vector and Θ_m is the parameters of the m-th tree.

The final prediction of the Random Forests model is given by:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} T_m\left(\mathbf{x}; \Theta_m\right) \tag{14}$$

where M is the number of trees in the forest. For a regression problem, $F(\mathbf{x})$ is the average of the output of all trees. For a classification problem, each tree votes for a class, and the class with the most votes is selected (i.e., mode of the outputs).

The randomness in the Random Forests algorithm comes from two sources:

- Bootstrap sampling: Each decision tree is built on a different sample of the data. These samples are created by drawing n instances at random with replacement from the original data, where n is the size of the original data. This process, known as bootstrap sampling, means that each sample will likely contain some instances multiple times and others not at all.

- Feature selection: At each split in each decision tree, a random subset of features is selected as candidates for the split. This number of features, often denoted as d, is typically much smaller than the total number of features. This process introduces further variability into the model and helps to decorrelate the trees, making the ensemble more robust.

When new data comes in, each tree in the forest generates a prediction independently. The final prediction is made by averaging the predictions from all the trees for regression, or by majority voting for classification.

This model is simple and interpretable, and yet it has been shown to perform very well on a wide range of machine learning tasks, often outperforming more complex models. Furthermore, it can handle large datasets with high dimensionality and missing values, and it provides a built-in measure of feature importance.

The strength of a Random Forest is a measure of the accuracy of its individual classifiers (the decision trees). It is defined as the expected value of the margin function, which measures the difference between the probability that a Random Forest classifier is correct and the maximum probability that it is incorrect:

$$s = E[mr(\mathbf{X}, Y)], \tag{15}$$

where:

$$mr(\mathbf{X}, Y) = P(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P(h(\mathbf{X}, \Theta) = j) \tag{16}$$

In this equation, $h(\mathbf{X}, \Theta)$ is the classifier (a decision tree in the Random Forest), $\mathbf{X}$ represents the input features, Y is the actual class, and Θ represents the parameters of the classifier. The probability is estimated over the random draws Θ.

3.6 Mean Squared Error (MSE)

The Mean Squared Error (MSE) will serve as the primary metric for evaluating the accuracy of our model's predictions during out-of-sample validation.

MSE quantifies the average squared differences between observed and predicted data, providing a clear indication of the model's performance. It is expressed as:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left(Y_i - \hat{\mathbf{Y}}_i \right)^2, \tag{17}$$

Where:

- Y_i represents the actual observation.
- $\hat{\mathbf{Y}}_i$ denotes the predicted observation.
- N signifies the total count of observations.

This formula ensures that larger errors have a disproportionately bigger impact on the overall error score, making it a robust measure for prediction accuracy.

4 Empirical Results

The results in this study are based on a detailed analysis of the S&P 100 symbols for 2020. We select the ARMA (AutoRegressive Moving Average) model as our benchmark due to its proven efficacy in capturing the autocorrelation structure inherent in financial time series data. The ARMA model's ability to integrate both autoregressive (past values) and moving average (lagged forecast errors) components makes it especially suited for modeling the temporal dependencies and inherent volatilities observed in financial markets.

The results are the average performance metrics across all studied symbols, combining data from various sources into one clear overview. By taking an average, we get a more stable perspective because it considers the variations and common features of different stock symbols.

It's important to note that when we average data, we might average out stock-level characteristics, potentially losing unique information that could enhance our forecast for individual stock symbols. To address this, our model is adjusted for each stock symbol using its unique ARMA parameters. This way, we can capture both the overall market trends and the unique patterns of individual stocks, allowing our model to effectively understand the diverse dynamics of the market.

Moreover, the computation of averaged performance metrics requires careful consideration of the potential heteroscedasticity across different symbols. The variability of a symbol's performance metrics can potentially depend on its trading volume, which further emphasizes the complexity of the S&P 100 financial ecosystem.

The parameters and performance metrics of the ARMA model, as shown in Table 2, reinforce the promising results obtained during our empirical study.

Table 2: The table summarizes the average parameters and performance metrics of the ARMA model used for forecasting the trading volume of individual stocks in the S&P 100 index, calculated across 100 stocks. The ARMA model, with an autoregressive (AR) component of 1.33 and a moving average (MA) component of 2.39, was selected based on the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and the Hannan-Quinn Information Criterion (HQIC). The model's accuracy was evaluated using the Mean Squared Error (MSE), while the Wald test was used to test the statistical significance of the coefficients.

Model	p **(AR)**	q **(MA)**	**AIC**	**BIC**	**HQIC**	**Avg MSE**	**Wald Test**
ARMA	1.33	2.39	-210556.80	-210516.50	-210543.00	1.87×10^{-10}	1.98×10^{26}

The Bayesian Information Criterion (BIC) favoured the ARMA model with the parameters autoregressive order (p) = 1.33 and moving average order (q) = 2.39. Important in defining the temporal relationships recorded by an ARMA model are the orders (p and q).

Within the ARMA model, p represents the order of the autoregressive component, and q signifies the order of the moving average component. They represent the average orders calculated across multiple models for the 100 stocks in the S&P 100 index.

Our empirical investigation follows a methodology that placed heavy focus on metrics for selecting models. To find the sweet spot between model fit and complexity, many information criteria are calculated, including the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and the Hannan-Quinn Information Criterion (HQIC).

These information criteria are designed to avoid overfitting by adding penalties for more complex models. Overfitting is a challenge in predictive modeling, where a model might perform well on the training data but not on new data. By using these criteria, we aim to strike a balance, ensuring our models are neither too simple nor overly complex.

In particular, the minimality of these measures is indicative of their optimality. An improved trade-off between model fit and complexity is shown by smaller values of AIC, BIC, and HQIC. Our ARMA model offers the optimal compromise between fit and complexity when seen through the BIC lens, as indicated by its low value (-210516.50) in the current analysis.

The BIC is a key component of the model, but it is not the only one. Consistently strong model fit is also shown by the AIC and HQIC, with values that are both very low. This consistency across different types of data strengthens the validity of our model selection.

It's important to mention that these criteria, while helpful, have their limitations. They may favour too simple models and are susceptible to assumptions regarding error distribution. Thus, model selection was informed by fundamental theoretical concerns coming from the financial time series literature, as well as the findings of the Mean Squared Error (MSE) and the Wald test. In the context of our analysis, the Wald test statistic, 1.98×10^{26}, plays a noteworthy role. This metric, with its large value, suggests a strong rejection of the null hypothesis that the coefficients of our ARMA model have no statistical significance.

A large Wald statistic typically implies a greater reduction in squared errors, thereby providing robust evidence against the null hypothesis of coefficient insignificance. Simply, the AR(p) and MA(q) parameters in our model significantly affect the prediction of S&P 100 trading volumes.

While the ARMA model results revealed meaningful insights into the S&P 100 index trading volume prediction, we acknowledged the necessity to leverage a more sophisticated tool that could model non-linear dependencies, which are common in financial time series. As such, we proceeded to employ a deep learning-based method, particularly the Long Short-Term Memory (LSTM) model.

The LSTM is a type of Recurrent Neural Network (RNN) specifically designed to tackle the vanishing gradient problem that plagues the conventional RNNs. It does this by maintaining a "cell state" across iterations, enabling it to learn and remember over long sequences, making it aptly suited for time series forecasting tasks.

For our LSTM model, we perform log transformations and scaling to normalise the data before feeding it into the model. The LSTM was structured with 4 units, which essentially indicates the dimensionality of the output space. Next, we utilize the Adam optimizer for its computational efficiency and relatively low memory requirements. Adam is an extension of stochastic gradient descent that updates network weights iteratively based on training data. It computes adaptive learning rates for different parameters. In addition to storing an exponentially decaying average of past squared gradients like RMSprop, Adam also keeps an exponentially decaying average of past gradients similar to momentum, making the updates more directed and hence speeding up the learning process.

We train the LSTM model for 50 epochs, with a batch size of 1. In machine learning parlance, an epoch is an entire pass through the entire training dataset, while a batch size is the number of samples processed before the model is updated. A smaller batch size, like 1, often provides a regularization effect, offering some level of resistance to overfitting.

The Average Mean Squared Error (MSE) of our model is 1.167×10^{-10}, a metric that quantifies the difference between the actual and predicted trading volumes, which is slightly smaller than that obtained from our ARMA model, indicating that the LSTM model might potentially outperform the ARMA model in predicting trading volumes; as shown in table 3.

Table 3: The table illustrates the average configuration and performance metrics of the LSTM model used for forecasting the trading volume of individual stocks in the S&P 100 index, calculated across 100 stocks. The model was implemented using the Keras library with 4 LSTM units and an Adam optimizer, trained for a total of 50 epochs with a batch size of 1. The model performance was evaluated using the Mean Squared Error (MSE), a common metric used for assessing the precision of forecasting models. A lower MSE denotes a more accurate model prediction.

Model	**LSTM Units**	**Optimizer**	**Number of Epochs**	**Batch Size**	**Avg MSE**
LSTM	4	Adam	50	1	1.167×10^{-10}

Next, we take a further step to incorporate $L1L2$ regularization into our LSTM model. Regularization techniques are commonly used in machine learning to prevent overfitting by penalising the model complexity, thus helping the model generalize better to unseen data. In this case, the $L1L2$ regularization is added to both the LSTM layer and the Dense layer with a regularization factor of 0.001.

Incorporating this regularization has a negligible effect on the Average MSE, with a slight increase to 1.170×10^{-10}, as shown in table 4.

Table 4: The table shows the average configuration and performance metrics of the LSTM model with $L1L2$ regularization, used for forecasting the trading volume of individual stocks in the S&P 100 index. These averages are calculated across 100 stocks. The model was implemented with the Keras library, using 4 LSTM units and an Adam optimizer, and trained 50 epochs with a batch size of 1. The regularization parameter was set to 0.001. The performance of the model was assessed with the Mean Squared Error (MSE), a standard metric for gauging the accuracy of forecasting models. A lower MSE indicates a more precise model prediction.

Model	**LSTM Units**	**Optimizer**	**Epochs**	**Batch Size**	**Reg.**	**Avg MSE**
LSTM with $L1L2$	4	Adam	50	1	0.001	1.170×10^{-10}

This implies that both the LSTM layer and the Dense layer were regularized. Despite this, the Average MSE was only marginally affected. This suggests that the model was already robust and not prone to overfitting. Consequently, the application of regularization did not markedly improve the model's generalization capabilities.

So, the use of an LSTM model for the prediction of the S&P 100 index trading volume, employing advanced techniques like Adam optimization and $L1L2$ regularization, lead to promising results, demonstrating that these complex, nuanced methods can be highly effective in the challenging realm of financial time series forecasting.

Building on our findings from the ARMA and LSTM models, we next explore the application of Temporal Convolutional Networks (TCN) to predict the S&P 100 stock trading volume. It possesses a unique advantage over other neural network architectures: it effectively captures long-term dependencies in sequence data, a critical attribute for accurate time-series forecasting.

Temporal Convolutional Networks (TCNs) are a unique class of neural networks designed to process sequential data. They offer two key advantages: causal convolutions and dilated convolutions, which make them exceptionally suited to time series forecasting.

Causal convolutions refer to the network's ability to maintain the temporal order of events. Unlike in standard convolutional layers, where future data points can influence the output at the current timestep, causal convolutions ensure that the model's output at each timestep is influenced only by data from that timestep or earlier. This not only helps to preserve the chronological integrity of the data but also eliminates any "look-ahead" bias, as future data is prevented from leaking into the past.

On the other hand, dilated convolutions allow TCNs to effectively aggregate and process information across increasingly larger temporal contexts without increasing the complexity of the model. By skipping input values with a certain step, these dilated convolutions expand the network's receptive field, allowing it to grasp longer-term dependencies in the data. This feature is particularly valuable for predicting trends that span large sequences, a common occurrence in many real-world time series data.

The implemented TCN model in this study employs 10 filters and a kernel size of 2. Filters in a convolutional network can be thought of as feature detectors. In this case, having 10 filters means the model can detect 10 different types of features at each layer. The kernel size is the size of the window that slides over the data to perform the convolution operation. Here, a kernel size of 2 means that the model is looking at two consecutive time steps at a time.

The use of a single stack in the TCN architecture implies that the model is relatively shallow, potentially leading to faster training times but at the risk of oversimplifying complex patterns. A deeper architecture with multiple stacks could potentially yield more accurate predictions, especially for more complex datasets, but it would also be more computationally expensive and could risk overfitting.

The Rectified Linear Unit (ReLU) activation function is utilized in the TCN. ReLU is a commonly used activation function in neural networks due to its efficiency and performance. It introduces non-linearity into the model, allowing the network to learn and predict complex patterns. The use of ReLU also helps mitigate the vanishing gradient problem common in deep learning models.

The model is trained over 50 epochs, which means the entire training dataset is passed forward and backward through the neural network 50 times. The batch size is set to 1, meaning that the model updates its parameters after each training sample. This approach can make the training process more computationally intensive but can sometimes result in a more refined model, as the network learns to adjust its parameters more frequently.

In our specific application, the TCN model demonstrates an impressive average MSE of 1.11×10^{-10}, outperforming both the ARMA and LSTM models, as shown in table 5.

Table 5: The table summarizes the average parameters and performance metrics of the Temporal Convolutional Network (TCN) model used for forecasting the trading volume of individual stocks in the S&P 100 index. These averages are calculated across 100 stocks. The TCN model parameters include 10 filters, a kernel size of 2, 1 stack, and uses the Rectified Linear Unit (ReLU) activation function. The model's accuracy was evaluated using the Mean Squared Error (MSE).

Model	**Filters**	**Kernel Size**	**Stacks**	**Number of Epochs**	**Activation**	**Avg MSE**
TCN	10	2	1	50	ReLU	1.11×10^{-10}

There are several potential reasons for this improved performance. One is that the TCN, with its extensive receptive field due to dilated convolutions, can effectively grasp longer-term dependencies in the data that may be missed by other models. Second, the TCN model's causal convolutions help maintain the integrity of the time series data, ensuring accurate predictions. Finally, unlike LSTMs, TCNs avoid issues with vanishing or exploding gradients, resulting in more stable training and potentially more accurate forecasts.

In pursuit of more accurate S&P 100 stock trading volume predictions, our research further explores the Gradient Boosting methodology. Gradient Boosting is an ensemble machine learning algorithm that relies on the idea of boosting, that is, converting a set of weak learners into a strong learner. The algorithm operates by fitting new models sequentially to the residuals of the previous ones to minimize the overall prediction error.

In the context of our model, the Gradient Boosting model serves as a regressor, predicting the trading volume as a continuous variable. This model's strengths lie in its flexibility and robustness, as it can effectively model complex non-linear relationships and is relatively immune to overfitting due to the boosting mechanism.

The model architecture involves three main hyperparameters: `n_estimators`, `max_depth`, and `min_samples_split`. `n_estimators` corresponds to the number of boosting stages or the number of sequential trees in the model. `max_depth` regulates the maximum depth of the individual trees, influencing the model's complexity. `min_samples_split` represents the minimum number of samples needed for a node to be split, which helps control the tree's growth and prevent overfitting.

Our Gradient Boosting model is constructed using a comprehensive grid search, which is an exhaustive searching method for selecting the optimal hyperparameters. This procedure iterates over a manually specified subset of the hyperparameter space of a learning algorithm. With a range from 1 to 10 for `n_estimators`, `max_depth`, and `min_samples_split`, we ensure an extensive exploration of the parameter space for the optimal configuration.

This enhanced performance can be attributed to the Gradient Boosting model's inherent attributes. The ability to capture complex non-linear relationships through a combination of weak learners, the flexibility offered by tuning hyperparameters, and the resistance to overfitting due to the boosting mechanism all contribute to this model's efficacy. Furthermore, Gradient Boosting can effectively capture various trends and patterns, making it particularly suites to stock trading volume prediction tasks that often encompass intricate and stochastic movements.

It's also important to acknowledge that Gradient Boosting, while computationally more expensive than some other models due to its iterative nature, can provide high-accuracy results. It combines the power of numerous decision trees and leverages boosting to minimize errors, making it an effective tool for time series forecasting tasks such as ours.

In our experiments, the Gradient Boosting model has an average MSE of 9.94×10^{-11}.

This performance is achieved using the optimal hyperparameters of `max_depth` = 7.70, `min_samples_split` = 3.68, and `n_estimators` = 10. The results are shown in Table 6.

Table 6: The table illustrates the average optimal parameters and performance metrics of the Gradient Boosting model, used for forecasting the trading volume of individual stocks in the S&P 100 index. These averages are calculated across 100 stocks. The model's optimal hyperparameters, attained via a comprehensive grid search, were a max depth of 7.70, a minimum samples split of 3.68, and 10 estimators. The Mean Squared Error (MSE), a prevalent metric for gauging the precision of a forecasting model, was used to assess the performance of this model. Lower MSE values denote more accurate model predictions.

Model	**Max Depth**	**Min Samples Split**	**Number of Estimators**	**Avg MSE**
Gradiant Boosting	7.70	3.68	10.0	9.94×10^{-11}

Next, we examine the potential reasons for the superior performance of the Gradient Boosting model. Primarily, it contrasts with the ARMA model, which assumes a linear relationship between observations. The Gradient Boosting model algorithm, however, can capture non-linear relationships present in the data. This attribute is extremely valuable when dealing with complex data sets such as stock market data, where relationships between variables are not always linear.

Moving forward, we then explore the application of the Random Forest model to predict the S&P 100 stock trading volume. The Random Forest model is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the mean prediction of the individual trees. It is particularly favored for its ability to reduce overfitting and improve generalization compared to a single decision tree.

Random Forests have three key hyperparameters: the maximum depth of the trees, the minimum number of samples required to split an internal node, and the number of trees in the forest (or estimators).

In our implementation, we use Grid Search to find the optimal hyperparameters for our Random Forest model. Grid Search is a hyperparameter tuning method where we train a model for every combination of hyperparameters and select the set that yields the best performance. The parameters we try for our model ranged from 1 to 10 for both maximum depth and minimum samples split, and from 1 to 10 for the number of estimators.

The best hyperparameters find from Grid Search for the Random Forest model are an average maximum depth of 6.81, an average minimum sample split of 5.95, and an average number of estimators of 2.90. These values suggest that the optimal Random Forest model has relatively shallow trees, requires a modest number of samples to make a split, and includes a small number of trees. This could indicate that the underlying data distribution can be captured by relatively simple tree structures, and a small number of trees are sufficient to reduce the variance of the model.

The Random Forest model demonstrates strong performance during testing, achieving an average Mean Squared Error (MSE) of 4.08×10^{-11}. This performance data is presented in Table 7 for further reference.

Table 7: The table outlines the average optimal parameters and performance metrics of the Random Forest model, employed for forecasting the trading volume of individual stocks in the S&P 100 index. These averages are calculated across 100 stocks. The model's optimal hyperparameters, achieved through extensive tuning, were a max depth of 6.81, a minimum samples split of 5.95, and 2.90 estimators. The Mean Squared Error (MSE), a common metric for evaluating the accuracy of a forecasting model, was used to assess the performance of this model. Lower MSE values denote more precise model predictions.

Model	**Max Depth**	**Min Samples Split**	**Number of Estimators**	**Avg MSE**
Random Forest	6.81	5.95	2.90	4.08×10^{-11}

Random Forest model is equipped to handle intricate interactions between various features, a capability not always directly addressed by the LSTM or TCN models. This benefit lets the model understand how various variables together affect trading volume, contributing to better prediction accuracy.

In terms of Random Forest performance, it's important to note several attributes that could explain its superior performance when compared to ARMA, LSTM, TCN, and even Gradient Boosting in the context of our forecasting task for S&P 100 stock trading volume.

- Ensemble Learning: Unlike Gradient Boosting where trees are grown sequentially to correct errors, Random Forest builds each decision tree independently, using a subset of the data. This characteristic allows the model to maintain robustness against overfitting and could potentially explain why it performs better than Gradient Boosting.

- Handling of Mixed Data Types: Stock market data often contains a mixture of numeric and categorical variables (for example, the impact of specific news events). Random Forest is adept at handling such mixed data types. While ARMA, LSTM, and TCN are better suited for sequential numeric data, their performance may falter when encountering mixed data types.

- Avoidance of Overfitting: Random Forest, by design, helps to prevent overfitting that can be problematic in Gradient Boosting. By using a subset of data and features for each tree, the model creates an ensemble that is, on average, less likely to overfit to the training data.

- Model Complexity and Flexibility: Random Forest combines simple base learners (decision trees) to build a complex model. This combination of simplicity and complexity makes it a very flexible tool, capable of modeling complex nonlinear relationships that could potentially be missed by ARMA, LSTM, TCN, and even Gradient Boosting.

- Non-linear and Non-parametric Nature: The Random Forest model is non-parametric and can capture complex non-linear relationships, which are common in financial time-series data such as S&P 100 stock trading volume. ARMA, although effective for certain stationary time-series data, can sometimes fall short in accurately modeling the non-linear dynamics of financial markets. Even though the TCN model, being a deep learning-based model, can handle non-linear relationships, it may not be as effective as Random Forest when the dependencies are particularly complex or when the available dataset is not sufficiently large to train such a complex model effectively.

- Robustness to Noise: Financial data can often be noisy, with sudden jumps and drops in trading volume. Random Forest, due to its inherent averaging process, tends to be robust to such noise. In contrast, both ARMA and TCN, and even Gradient Boosting, may be sensitive to these fluctuations, potentially leading to less accurate predictions.

In our research, we gain a deeper understanding of stock trading volumes for the S&P 100. A particular point of interest is examining the outliers: the highest 10% and the lowest 10% of trading volumes. By focusing on these extremes, we hope to identify patterns or unique characteristics that might not be evident when looking at average trading volumes.

To achieve our goals, we use all mentioned methods on these subsamples. From our analysis, the Random Forest model consistently delivers the best performance, based on the Mean Squared Error (MSE) measure, both for the general dataset and for the subsets representing the highest and lowest 10% of trading volumes.

Table 8 offers a comparative perspective for the 10% highest trading volumes. It is evident that the Random Forest model, with an average MSE of 2.74×10^{-12}, significantly outperforms its counterparts. For instance, even the Gradient Boost, another ensemble method, records an average MSE that is nearly an order of magnitude higher.

Table 8: This table presents a detailed comparison of various forecasting models, each evaluated based on their average Mean Squared Error (MSE), for predicting the top 10% of trading volumes in the S&P 100. The results indicate varied performances across models with the Random Forest showcasing the lowest MSE.

Model	**Average MSE**
ARMA	6.99×10^{-11}
LSTM	6.76×10^{-11}
TCN	7.13×10^{-11}
Gradient Boost	1.55×10^{-11}
Random Forest	$\mathbf{2.74 \times 10^{-12}}$

Turning to the data on the 10% lowest trading volumes, the findings align with our previous observations. As presented in table 9, the Random Forest model performs well with an MSE of 7.44×10^{-11}. Other models such as ARMA, LSTM, and TCN exhibit higher error rates.

Table 9: Performance Evaluation of Different Models for Forecasting S&P 100 Stock Trading Volume for the 10% Lowest Trading Volumes. The performance of each model was evaluated using the average Mean Squared Error (MSE). The Random Forest model again delivering the most favorable outcome.

Model	**Average MSE**
ARMA	1.08×10^{-9}
LSTM	1.57×10^{-9}
TCN	2.26×10^{-9}
Gradient Boost	2.13×10^{-10}
Random Forest	$\mathbf{7.44 \times 10^{-11}}$

5 Conclusion

In conclusion, our study embarked on a detailed exploration of multiple forecasting methodologies for intraday trading volume, a critical component for the implementation of Volume Weighted Average Price (VWAP) trading algorithms. Drawing from the intraday Trade and Quote (TAQ) dataset, our investigation traversed a broad spectrum of financial instruments, scrutinizing the nuances of intraday trade volume through an array of analytical lenses.

We began our study with the Autoregressive Moving Average (ARMA) model as a benchmark and then ventured into advanced machine learning techniques like Long Short-Term Memory (LSTM) neural networks, Temporal Convolutional Networks (TCN), Gradient Boosting, and Random Forest. Each model was thoroughly evaluated for its proficiency to encapsulate trends and unveil latent patterns in intraday trading volume data. While seasonality is acknowledged to be an important factor in financial markets, it was not the focus of this current study and remains an avenue for future research.

The insights gained from our study underscore the power and potential of the methodologies we explored, shedding light on their unique strengths and limitations in forecasting intraday trading volume. Among these, ensemble learning techniques, notably Random Forest, emerged as the most potent tools. Random Forest demonstrated exceptional performance, achieving the lowest Mean Squared Error (MSE) among all the studied models, a testament to its robust predictive capabilities.

Delving deeper into the factors contributing to its success, Random Forest stood out due to its inherent averaging process, making it robust to noise, and its ability to handle mixed data types, which accommodates the multifaceted nature of financial market data. These unique features make Random Forest an incredibly robust and flexible tool, well-suited for the complex task of forecasting in the unpredictable landscape of financial markets.

Furthermore, our research draws upon several pioneering studies in the field, effectively synthesizing diverse analytical perspectives to enhance the depth of our approach. The work of Kercheval and Zhang [2015] and Dixon [2018] provided valuable insights into the application of predictive classifiers and trade execution models, while Holden and Jacobsen [2014] and Bogousslavsky and Muravyev [2022] enriched our understanding of dynamic volume models and the impact of passive investing.

Our study offers a valuable contribution to the ongoing discourse on intraday trading volume forecasting, paving the way for further research in this area. While our investigation provides a comprehensive assessment of each model, future work can delve deeper into the hyperparameter optimization of these models, potentially further enhancing their performance. Moreover, the examination of other predictive metrics and variables could be integrated to construct even more robust and accurate forecasting models. In the complex, ever-evolving world of financial markets, the quest for the most accurate prediction models remains a vital and challenging pursuit.

References

Torben G Andersen. Return volatility and trading volume: An information flow interpretation of stochastic volatility. *The Journal of Finance*, 51(1):169–204, 1996.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Jedrzej Białkowski, Serge Darolles, and Gaëlle Le Fol. Improving vwap strategies: A dynamic volume approach. *Journal of Banking & Finance*, 32(9):1709–1722, 2008.

Vincent Bogousslavsky and Pierre Collin-Dufresne. Liquidity, volume, and order imbalance volatility. *Journal of Finance, Forthcoming*, 2022.

Vincent Bogousslavsky and Dmitriy Muravyev. Who trades at the close? implications for price discovery and liquidity. *Implications for Price Discovery and Liquidity (October 25, 2022)*, 2022.

Jean-Philippe Bouchaud, J Doyne Farmer, and Fabrizio Lillo. How markets slowly digest changes in supply and demand. In *Handbook of financial markets: dynamics and evolution*, pages 57–160. Elsevier, 2009.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Christian T Brownlees, Fabrizio Cipollini, and Giampiero M Gallo. Intra-daily volume modeling and prediction for algorithmic trading. *Journal of Financial Econometrics*, 9 (3):489–518, 2011.

Jian Cao, Zhi Li, and Jian Li. Financial time series forecasting model based on ceemdan and lstm. *Physica A: Statistical mechanics and its applications*, 519:127–139, 2019.

Matthew Dixon. A high-frequency trade execution model for supervised learning. *High Frequency*, 1(1):32–52, 2018.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Christian Gourieroux, Joanna Jasiak, and Gaelle Le Fol. Intra-day market activity. *Journal of Financial Markets*, 2(3):193–226, 1999.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Craig W Holden and Stacey Jacobsen. Liquidity measurement problems in fast, competitive markets: Expensive and cheap solutions. *The Journal of Finance*, 69(4):1747–1785, 2014.

Mark L Humphery-Jenner. Optimal vwap trading under noisy conditions. *Journal of Banking & Finance*, 35(9):2319–2329, 2011.

Alec N Kercheval and Yuan Zhang. Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15(8):1315–1329, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Xiaotao Liu and Kin Keung Lai. Intraday volume percentages forecasting using a dynamic svm-based approach. *Journal of Systems Science and Complexity*, 30(2):421–433, 2017.

Ignacio N Lobato and Carlos Velasco. Long memory in stock-market trading volume. *Journal of Business & Economic Statistics*, 18(4):410–427, 2000.

Ananth N Madhavan. Vwap strategies. *Trading*, 2002(1):32–39, 2002.

Jaisimha Manchaldore, Imon Palit, and Oleg Soloviev. Wavelet decomposition for intraday volume dynamics. *Quantitative Finance*, 10(8):917–930, 2010.

Arnold Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM review*, 40(3):636–666, 1998.

Sokratis Papadopoulos and Ioannis Karakatsanis. Short-term electricity load forecasting using time series and ensemble learning methods. In *2015 IEEE Power and Energy Conference at Illinois (PECI)*, pages 1–6. IEEE, 2015.

Han Lin Shang. Forecasting intraday s&p 500 index returns: A functional time series approach. *Journal of forecasting*, 36(7):741–755, 2017.

Parnandi Srinu Vasarao and Midhun Chakkaravarthy. Time series analysis using random forest for predicting stock variances efficiency. In *Intelligent Systems and Sustainable Computing: Proceedings of ICISSC 2021*, pages 59–67. Springer, 2022.

José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, 9(1):3–21, 2021.

Yaw-Huei Wang and Yun-Yi Wang. The information content of intraday implied volatility for volatility forecasting. *Journal of Forecasting*, 35(2):167–178, 2016.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26:289–315, 2007.