www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Faculteit Wetenschappen

School voor Informatietechnologie

master in de informatica

Masterthesis

Enhancing Trustworthiness in Algorithmic Stock Forecasting using Multi-Model Machine Learning and Historical Similarity

Xander Corvers
Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :
Prof. dr. Gustavo Alberto ROVELO RUIZ

BEGELEIDER :
De heer Gilles EERLINGS

2024
2025

Faculteit Wetenschappen
School voor Informatietechnologie
master in de informatica

Masterthesis

Enhancing Trustworthiness in Algorithmic Stock Forecasting using Multi-Model Machine Learning and Historical Similarity

Xander Corvers
Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :
Prof. dr. Gustavo Alberto ROVELO RUIZ

BEGELEIDER :
De heer Gilles EERLINGS

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to the individuals who have supported me throughout my master's thesis journey.

First and foremost, I extend my deepest appreciation to my promotor, Prof. Dr. Gustavo Rovelo Ruiz. I am incredibly grateful for the opportunity he provided me to propose and pursue my own research topic, allowing me to delve into areas that align closely with my interests. It has been a privilege to work under his guidance and expand my knowledge of machine learning. The initial brainstorm sessions we had were instrumental in laying a robust foundation for this research. His expert guidance on both the machine learning aspects and the principles of information visualization for the developed system was invaluable. I thank him for his supervision and for entrusting me with this engaging research.

I am also profoundly thankful to my supervisor, Gilles Eerlings. His day-to-day guidance, consistent practical support, insightful feedback, and ready availability were crucial to the progress of this work. I particularly appreciate the way he encouraged exploration of different options and research directions, and our collaborative sessions for thinking through challenges and generating new ideas. His practical advice, especially concerning multi-model machine learning and effective information representation, significantly shaped the outcome of this thesis.

My sincere thanks go to all the participants of the user study. Their willingness to dedicate their time and provide thoughtful feedback was essential for evaluating the system and gathering insights into user trust and perception. This research would not have been possible without their valuable contributions.

I would also like to acknowledge the Digital Future Lab research center at Hasselt University for providing essential resources, particularly the AI workstation, which was instrumental for the training and evaluation of the machine learning models.

Finally, I wish to thank my family and friends for their unwavering support, patience, and encouragement throughout this demanding but rewarding period. Their belief in me has been a constant source of motivation.

Samenvatting

Het voorspellen van de aandelenmarkt met geavanceerde computertechnieken, met name machine learning (ML), biedt aanzienlijk potentieel om complexe marktdynamiek te analyseren. Veel geavanceerde ML-modellen functioneren echter als black boxes, waarvan de interne besluitvormingsprocessen ondoorzichtig blijven voor gebruikers. Dit gebrek aan transparantie vormt een kritieke uitdaging, vooral in financiële toepassingen met hoge belangen, en leidt vaak tot een tekort aan gebruikersvertrouwen, wat de effectieve adoptie van deze krachtige tools belemmert. Deze thesis pakt dit vertrouwenstekort aan door een nieuwe benadering te onderzoeken die de betrouwbaarheid van ML-gedreven beursvoorspellingen verbetert, specifiek voor gebruikers zonder diepgaande technische of financiële expertise.

De kernbijdrage van dit onderzoek is het ontwerpen, implementeren en empirisch evalueren van een beslissingsondersteunend systeem. Dit systeem integreert voorspellingen van meerdere, diverse machine learning-modellen met inzichten uit historisch vergelijkbare marktpatronen. De centrale hypothese is dat een meer holistische en interpreteerbare kijk op potentiële marktbewegingen; door consensus of divergentie tussen modellen te tonen en voorspellingen te contextualiseren met vergelijkbare situaties uit het verleden, het gebruikersvertrouwen aanzienlijk kan verbeteren. Het ontwikkelde systeem, met name via zijn “Multiple Models View” (MMV), omvat zes verschillende ML-modellen voor het voorspellen van aandelenkoersen: Random Forest, eXtreme Gradient Boosting, Light Gradient Boosting Machine, Long Short-Term Memory, Neural Basis Expansion Analysis for Time Series Forecasting en Temporal Fusion Transformer. Een belangrijke innovatie is de zoekmachine voor historische overeenkomsten, die marktperioden van 20 dagen uit het verleden identificeert die qua kenmerken sterk overeenkomen met de huidige marktomstandigheden. Deze zoektocht werkt op basis van kenmerken, afgeleid via Principal Component Analysis (PCA) uit dezelfde set van acht technische indicatoren die de ML-modellen als directe input gebruiken. Het systeem laat zien hoe de geïntegreerde ML-modellen zouden hebben gepresteerd voor deze geïdentificeerde historische analogieën, door hun voorspellingen naast de daadwerkelijke marktresultaten en hun voorspellingsfouten te presenteren. Dit levert tastbaar, op voorbeelden gebaseerd bewijs van de modelbetrouwbaarheid onder omstandigheden analoog aan de huidige. De MMV presenteert vervolgens een algemene beleggingsaanbeveling, transparant gekoppeld aan de prestaties van de best presterende modellen op deze historisch vergelijkbare patronen.

De onderzoeksmethodologie omvatte verschillende kernfasen. Eerst zijn historische dagelijkse beursgegevens van IBM (1962-2024) verzameld en voorbewerkt, en is een set van acht technische indicatoren ontwikkeld als inputkenmerken voor de voorspellende modellen. De zes geselecteerde ML-modellen zijn geïmplementeerd, en hun voorspellende prestaties zijn geanalyseerd voor procentuele rendementsprognoses op een horizon van 1 en 10 dagen. Deze analyse bracht significante overfitting aan het licht voor de meeste modellen bij gebruik van standaard hyperparameters, met over het algemeen zwakke statistische voorspellende nauwkeurigheid op ongeziene testdata. Er werd echter een interessante divergentie waargenomen: sommige modellen, ondanks zwakke statistische meetwaarden, leverden positieve totale rendementen op in een geïdealiseerde handelssimulatie en presteerden beter dan een Buy & Hold-benchmark. Hierna is het beslissingsondersteunend systeem ontwikkeld, met twee hoofdinterfaces: de uitgebreide MMV en een eenvoudigere “Single Model View” (SMV). De SMV presenteerde enkel voorspellingen van het Random Forest-model, zonder de multimodale vergelijkingen of diepgaande historische similariteitsanalyse. Ten slotte is een empirisch gebruikersonderzoek uitgevoerd met 22 deelnemers die met zowel de SMV als de MMV werkten. De studie verzamelde kwantitatieve en kwalitatieve gegevens over gebruikersvertrouwen, vertrouwen in hypothetische beleggingsbeslissingen, waargenomen systeembruiksvriendelijkheid, begrip van aanbevelingen en risicobewustzijn.

De bevindingen van het gebruikersonderzoek ondersteunden krachtig de centrale hypothese van de thesis. De Multiple Models View (MMV) kreeg de overweldigende voorkeur van de deelnemers en werd als significant betrouwbaarder ervaren dan de Single Model View (SMV). Gebruikers rapporteerden significant meer vertrouwen in hun beslissingen bij gebruik van de MMV. Zij schreven de grotere betrouwbaarheid ervan toe aan de transparantie door het zien van meerdere modeloutputs, de contextualisering door historische patroonvergelijkingen, en de op bewijs gebaseerde aard van de aanbevelingen. De MMV bleek ook significant effectiever om gebruikers te helpen potentiële risico's en onzekerheden verbonden aan voorspellingen te begrijpen, grotendeels door modelverschillen en de variabiliteit van uitkomsten in vergelijkbare scenario's uit het verleden bloot te leggen. Hoewel de MMV als complexer werd ervaren dan de SMV, wees de System Usability Scale (SUS)-score nog steeds op goede tot uitstekende bruikbaarheid. Dit suggereert dat gebruikers bereid waren een hogere complexiteit te tolereren in ruil voor de substantiële winst in vertrouwen en contextueel begrip.

Deze thesis toont aan dat een aanpak die meerdere machine learning-modellen combineert met transparante, op bewijs gebaseerde historische similariteitsanalyse, de betrouwbaarheid voor de gebruiker in algoritmische beursvoorspellingssystemen effectief kan verhogen. Door verder te gaan dan enkelvoudige, ondoorzichtige voorspellingen en gebruikers rijkere context, vergelijkende inzichten en begrip van modelgedrag in analoge situaties uit het verleden te bieden, bevorderde het ontwikkelde systeem een beter gekalibreerd en geïnformeerd vertrouwen. Hoewel de uitdaging om consistent accurate beursvoorspellingen te realiseren groot blijft, biedt dit onderzoek waardevolle inzichten voor het ontwerpen van door kunstmatige intelligentie (KI) gedreven financiële tools. Met dergelijke tools kunnen gebruikers met meer vertrouwen en verantwoordelijkheid omgaan, wat de kritieke rol van transparantie en contextualisering bij het opbouwen van vertrouwen in complexe KI-systemen onderstreept.

Summary

Stock market forecasting using advanced computational techniques, particularly machine learning (ML), offers significant potential for analyzing complex market dynamics. However, many sophisticated ML models operate as black boxes, with their internal decision-making processes remaining opaque to users. This lack of transparency poses a critical challenge, especially in high-stakes financial applications, often leading to a deficit of user trust and hindering the effective adoption of these powerful tools. This thesis addresses this trust deficit by investigating a novel approach to enhance the trustworthiness of ML-driven stock market predictions, particularly for users who may not possess deep technical or financial expertise.

The core contribution of this research is the design, implementation, and empirical evaluation of a decision support system that integrates predictions from multiple diverse machine learning models with insights derived from historically similar market patterns. The central hypothesis is that presenting users with a more holistic and interpretable view of potential market movements; by showcasing consensus or divergence among models and contextualizing predictions with past analogous situations, can significantly improve user trust. The system developed, particularly through its "Multiple Models View" (MMV), incorporates six distinct ML models for stock price prediction: Random Forest, eXtreme Gradient Boosting, Light Gradient Boosting Machine, Long Short-Term Memory, Neural Basis Expansion Analysis for Time Series Forecasting, and Temporal Fusion Transformer. A key innovation is the historical similarity search engine, which identifies past 20-day market windows exhibiting strong feature similarity to the current market conditions. This similarity search operates on features derived through Principal Component Analysis (PCA) from the same set of eight technical indicators the ML models use as their direct input. The system demonstrates how the integrated ML models would have performed for these identified historical analogies, showing their predictions alongside the actual market outcomes and their prediction errors. This provides tangible, example-based evidence of model reliability under conditions analogous to the present. The MMV then presents an overall investment recommendation transparently linked to the performance of the best-performing models on these historically similar patterns.

The research methodology involved several key stages. First, historical daily stock data for IBM from 1962 to 2024 was acquired and preprocessed, and a set of eight technical indicators was engineered to serve as input features for the predictive models. The six selected ML models were implemented, and their predictive performance was analyzed for 1-day and 10-day horizon percentage return forecasts. This analysis revealed significant overfitting for most models using default hyperparameters, with generally weak statistical predictive accuracy on unseen test data. However, an interesting divergence was observed where some models, despite poor statistical metrics, yielded positive total returns in an idealized trading strategy simulation, outperforming a Buy & Hold benchmark. Following this, the decision support system was developed, featuring two main interfaces: the comprehensive MMV and a simpler "Single Model View" (SMV), which presented predictions from only the Random Forest model without the multi-model comparisons or deep historical similarity analysis. Finally, an empirical user study was conducted with 22 participants who interacted with both the SMV and MMV. The study collected quantitative and qualitative data on user trust, confidence in hypothetical investment decisions, perceived system usability, understanding of recommendations, and risk

awareness.

The findings from the user study strongly supported the thesis's central hypothesis. The Multiple Models View (MMV) was overwhelmingly preferred by participants and perceived as significantly more trustworthy than the Single Model View (SMV). Users reported significantly higher confidence in their decisions when using the MMV, attributed its greater trustworthiness to the transparency afforded by seeing multiple model outputs, the contextualization provided by historical pattern comparisons, and the evidence-based nature of its recommendations. The MMV was also found to be significantly more effective in helping users understand potential risks and uncertainties associated with predictions, largely by exposing model disagreements and the variability of outcomes in similar past scenarios. While the MMV was perceived as more complex than the SMV, its System Usability Scale (SUS) score still indicated good to excellent usability, suggesting that users were willing to tolerate increased complexity for the substantial gains in trust and contextual understanding.

This thesis demonstrates that an approach combining multiple machine learning models with transparent, evidence-based historical similarity analysis can effectively enhance user trustworthiness in algorithmic stock forecasting systems. By moving beyond single, opaque predictions and providing users with richer context, comparative insights, and an understanding of past model behavior in analogous situations, the developed system fostered a more calibrated and informed sense of trust. While the challenge of achieving consistently accurate stock market predictions remains profound, this research offers valuable insights into designing Artificial Intelligence (AI)-driven financial tools that users can engage with more confidently and responsibly, underscoring the critical role of transparency and contextualization in building trust in complex AI systems.

Contents

Chapter 1

Introduction

Stock markets have long intrigued investors and researchers, representing a domain where strategic decision-making can lead to significant financial outcomes [BK12]. Investors frequently rely on technical indicators like the Relative Strength Index (RSI) and Bollinger Bands to identify patterns signaling potential price reversals [Huy24]. For instance, RSI values below 30 may suggest an oversold asset, while Bollinger Bands provide insight into price volatility and deviations from the mean, potentially indicating an inevitable price move [Huy24]. However, despite strategies aligned with established technical principles, trades do not always yield the expected results, highlighting the apparent randomness of market movements. The advent of sophisticated computing techniques, particularly machine learning (ML), has opened new frontiers for analyzing these complex market dynamics and forecasting price movements [XL24]. Yet, despite their potential predictive power, many advanced ML models operate as black boxes, with their internal decision-making processes remaining opaque to users. This lack of transparency poses a critical challenge, especially in high-stakes financial applications, often leading to a deficit of user trust and hindering the effective adoption of these powerful tools [Bar+20; BB21; Li+24].

This thesis addresses this trust deficit by investigating a novel approach to enhance the trustworthiness of ML-driven stock market predictions, particularly for users who may not possess deep technical or financial expertise. We posit that presenting predictions from multiple diverse machine learning models, rather than relying on a single, potentially opaque forecast, can offer a more robust and nuanced perspective, especially given the market's inherent complexity [XL24]. By allowing users to observe areas of consensus or divergence among these models and by further contextualizing these outputs with analytically identified similar historical market patterns and the models' past performance in those analogous situations, we aim to bridge this trust gap. Therefore, the primary research question guiding this study is: *"How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?"*.

The pursuit of predictive advantages in financial markets is often framed by the Efficient Market Hypothesis (EMH), formalized by Eugene Fama, which, in its strong form, posits that all available information is already reflected in prices, rendering consistent outperformance through technical or fundamental analysis unachievable [Fam70]. Indeed, empirical evidence often supports this view, with 61% of actively managed funds failing to outperform market benchmarks [VV14]. Yet, the persistent success of quantitative firms like Renaissance Technologies, whose Medallion Fund has achieved extraordinary returns far exceeding market averages for decades, as illustrated in Figure 1.1, challenges the strictest interpretations of EMH [Zuc19; Mag23]. These successes, achieved by applying sophisticated probabilistic models and statistical techniques such as Markov processes to segment market states and exploit nonrandom anomalies amid market noise, suggest that market inefficiencies can, at least by some, be systematically

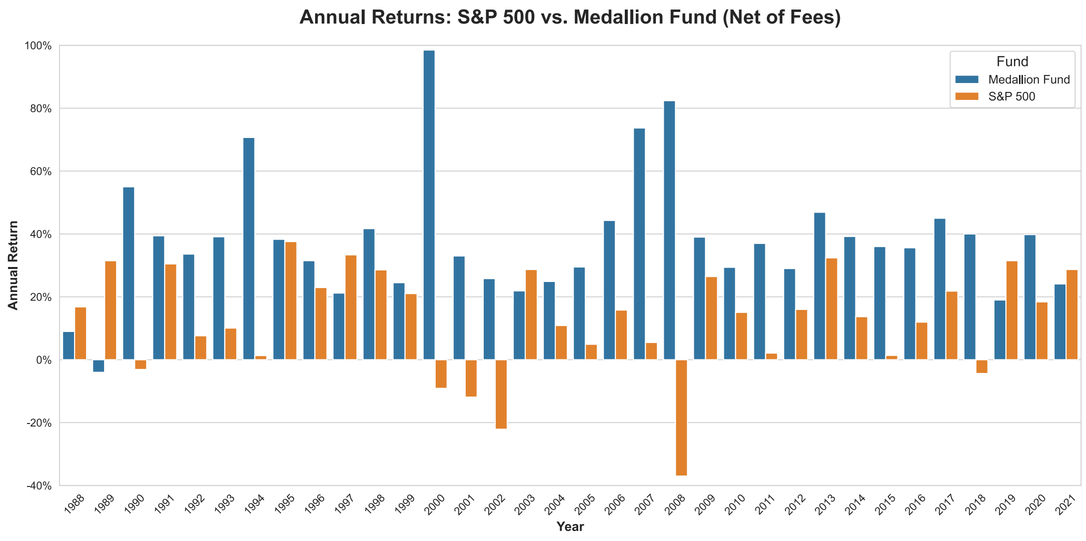


Figure 1.1: Annual returns of the S&P 500 Index compared to the Medallion Fund (net of fees) from 1988 to 2021. The orange bars represent the annual returns of the S&P 500, while the blue bars depict the returns of the Medallion Fund. The vertical axis measures the percentage annual return, while the horizontal axis represents the years within the given period. The figure illustrates the substantial outperformance of the Medallion Fund relative to the S&P 500, with consistently higher returns across nearly all years, even during market downturns. Figure inspired by [Mag23].

identified and leveraged [Zuc19]. As Jim Simons reportedly noted, "the assumption that prices are always correct is false - anomalies persist, and with the right quantitative techniques, they can be used for predictive insights" [Zuc19]. This ongoing debate underscores the potential value of advanced computational methods if their outputs can be made sufficiently reliable and trustworthy.

Effectively integrating ML models into financial decision-making requires addressing the aforementioned black-box problem. Explainable Artificial Intelligence (XAI) has emerged as a crucial field dedicated to developing methods that render ML model outputs and reasoning processes understandable to humans [BB21]. In finance, where forecasts carry significant economic implications, understanding why a model makes a particular recommendation is essential. A lack of transparency can lead to distrust, where effective models are ignored, or overtrust, where flawed recommendations are blindly followed, potentially resulting in financial losses or regulatory issues [Li+24; Bar+20]. XAI aims to empower users with the insights needed to assess algorithmic advice critically, fostering informed decision-making, better risk management, regulatory compliance, and broader financial inclusion [BB21; Bar+20]. Our approach, leveraging multiple models and historical similarity, aligns with XAI principles by seeking to provide intuitive, evidence-based explanations for predictions.

This thesis focuses on technical analysis for stock price prediction using machine learning. Technical analysis utilizes quantifiable, high-frequency historical market data, primarily price movements (Open, High, Low, Close) and trading volumes, to identify patterns and predict future trends [PJ16]. Technical analysis contrasts with fundamental analysis, which seeks to determine a stock's intrinsic value by examining company financials and broader economic conditions for long-term investment [PJ16]. Technical analysis is well-suited for ML models as it provides structured inputs and is believed to rapidly incorporate all available market information, bypassing potential delays and subjectivity associated with fundamental factors [BK23; PJ16]. However, the stock market is inherently multifaceted, nonlinear, and dynamic, influenced by macroeconomic shifts, corporate performance, geopolitical events, investor sentiment, and regulatory changes [XL24]. While technical analysis provides structured data, the investment environment remains fraught with risk arising from the uncertainty of future price movements. Furthermore, real-world decision-making is considerably more complex than pure rational analysis, often influenced by behavioral biases such as overconfidence, loss aversion, and herd behavior [Vir13]. These psychological factors can lead to deviations from rational choices, complicating the straightforward application of predictive models even when based on robust technical data.

To empirically investigate our research question, this thesis focuses on the stock of International Business Machines Corporation (IBM). IBM was chosen for its extensive and consistent daily trading history dating back to 1962, providing a rich and robust dataset crucial for training and validating diverse predictive models over various market regimes [Sta24]. We develop and evaluate a decision support system integrating predictions from six distinct ML models. These predictions are contextualized by a historical similarity search engine and presented to users via an interactive dashboard to make the outputs more interpretable. The impact of this system on user trust, confidence, and decision-making is then assessed through a dedicated user study involving participants without deep technical expertise.

The remainder of this thesis is structured as follows. Chapter 2: *Related Work* reviews existing literature relevant to this study. Key areas explored include techniques for data collection and processing within the financial sector, the application of machine learning models for stock market prediction, the principles and methods of Explainable AI (XAI) in finance, and theoretical frameworks for understanding user trust in AI-driven decision support systems, including the potential impact of multi-model approaches and similarity-based explanations.

Chapter 3: *Data Acquisition and Description* details the process of obtaining the IBM stock dataset, describes its characteristics, and discusses the rationale for its selection. Chapter 4: *Data Preprocessing and Feature Engineering* outlines the crucial steps to clean and prepare the raw data for modeling, including any feature engineering techniques applied to derive meaningful inputs for the predictive models.

Chapter 5: *Predictive Modeling and Performance Analysis* describes the architectures and implementation details of the six distinct machine learning models employed: Random Forest (RF), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), Recurrent Neural Network (RNN), Neural Basis Expansion Analysis for Time Series (N-BEATS), and Temporal Fusion Transformer (TFT). This chapter also presents a comprehensive comparative analysis of these models based on their predictive accuracy and computational efficiency.

Chapter 6: *Design of a Multi-Model Trust-Enhancing System* presents the architecture and functionality of the decision support tool developed in this research. This chapter explains how predictions from the multiple models are integrated and combined with a similarity search mechanism to identify analogous historical market periods. It also details the design of the user-facing dashboard, which incorporates views for both single-model and multi-model predictions augmented with these similarity insights.

Chapter 7: *User Study on System Trustworthiness* describes the methodology and presents the findings of a user study conducted with participants without deep technical expertise. This chapter outlines the experimental design, participant tasks involving interaction with the different dashboard views, and the data collection methods. It then analyzes the quantitative and qualitative data to evaluate the system's impact on user trust, confidence, and hypothetical investment decisions.

Finally, Chapter 8: *Conclusions* interprets the overall findings from the predictive modeling and user study, connecting them to the primary research question and existing literature. This chapter also considers the implications of the research, acknowledges its limitations, summarizes the main contributions, and suggests potential directions for future research in this area.

By structuring the thesis around these main components, this research aims to bridge the gap between the sophisticated capabilities of advanced predictive modeling and the critical need for user trust in these systems. It seeks to provide actionable insights and demonstrate practical pathways for enhancing this trust, particularly within the application of machine learning for stock market forecasting. The findings are intended to contribute empirical evidence to the ongoing debates on financial market efficiency, the role of algorithmic trading, and, more broadly, to inform the responsible integration of artificial intelligence into investment strategies. Ultimately, the goal is to foster more informed and confident decision-making by users better equipped to understand and leverage AI-driven financial tools.

Chapter 2

Related Work

This chapter reviews existing research relevant to this thesis, which centers on predicting stock market prices using machine learning and improving user trust in these predictive systems. This review examines current practices, challenges, and recent advancements across several key domains to contextualize this research. It begins by exploring financial data acquisition for market analysis (Section 2.1), covering common sources, types, and difficulties in obtaining necessary data. Subsequently, it delves into preprocessing and feature engineering for financial time series (Section 2.2), focusing on methods for cleaning, transforming, and creating useful features. The review then surveys machine learning models commonly employed for stock price prediction (Section 2.3), discussing their application, performance, and role in decision support. It further addresses the crucial topics of trust and explainability in AI-driven financial decision support (Section 2.4), including the challenges of black-box models and strategies for enhancing user confidence through transparency. Building upon this comprehensive review, the chapter concludes by presenting the specific approach proposed in this thesis to improve trust in AI predictions (Section 2.5).

2.1 Financial Data Acquisition for Market Analysis

The data collection process for forecasting stock markets has evolved, driven by the increasing availability of high-frequency financial data and alternative data sources. Many studies rely on established financial data providers and application programming interfaces (APIs) to obtain historical market data. For example, Khan et al. (2023) obtained Tesla Inc. stock data from the Alpaca broker from January 1, 2016, to December 31, 2021. They split the data into training and test sets to facilitate model evaluation [Kha+23]. Similarly, Weerapat and Kantavat (2023) and Mozaffari and Zhang (2024) used the Yahoo Finance platform, accessible via the Python yfinance library, to collect daily stock data over multiple years, capturing key characteristics such as open, high, low, close, volume and adjusted closing prices [BK23; MZ24]. In addition, Subasi et al. (2021) used Yahoo Finance to compile decades-long datasets of major stock indices such as NASDAQ, NYSE, Nikkei, and FTSE, providing a broad time perspective on market trends [Sub+21].

Several studies have expanded their data collection methods by integrating heterogeneous data types and using multiple sources. For example, Huyen Chau, Nguyen Thiand Doan, and Trung Phong (2024) collected stock price data of pharmaceutical, chemical, and fertilizer companies on the Hanoi Stock Exchange by combining web scraping with BeautifulSoup from cophieu68.com and API requests from vndirect.com.vn. This dual-source approach allowed the extraction and integration of attributes such as trading date, opening, closing, highest and lowest prices, along with trading volume [Huy24]. In another study, daily long-term OHLCV data for SPY were combined with a custom data table of over 83,000 securities collected from multiple exchanges using market data from Tiingo and fundamental data from Financial Modeling Prep to calculate variables such as the Piotroski score [Chi22]. Patil et al. (2020) further diversified their data collection by including historical stock prices at both daily and minute level intervals from sources such as Yahoo Finance and Tiingo, as well as a dataset of one million financial news articles, blending quantitative market data with qualitative sentiment indicators [Pat+20].

Other studies have integrated nontraditional data sources to capture market sentiment and real-time dynamics. For example, studies by Sirimevan et al. (2019) and Karami et al. (2022) that combined stock market, social media, and news data illustrate the use of alternative data, such as Twitter feeds, Reuters news headlines, and Google Trends data, in predictive frameworks [Sir+19; Kha+22]. These approaches use APIs and web crawling techniques to collect unstructured data that, combined with conventional market indicators, are used in forecasting models. Moreover, Narkar (2019) addressed potential human bias in data selection by randomly choosing stock symbols and using adjusted closing prices from Alphavantage to ensure that the data reflect market values after taking into account corporate actions [Nar19].

The literature overviews diverse data collection approaches for stock market prediction. Researchers have utilized traditional market data sources, such as Yahoo Finance, Alpaca, Tiingo, and Alphavantage, and supplementary data gathered through web scraping, social media, and news outlets.

2.2 Preprocessing and Feature Engineering in Financial Time Series

A fundamental step in preprocessing financial data is addressing missing values and outliers. Khan et al. (2023) highlight removing unwanted data, including trade counts, and handling missing stock prices, which can be achieved through interpolation or by using the mean of adjacent data points [Kha+23]. Mostafavi and Hooman (2025) specifically used linear interpolation to handle missing values, particularly for indicators like the Average True Range (ATR) and Relative Vigor Index (RVI). They employed Interquartile Range (IQR) based filtering to remove outliers, which can significantly distort model training [MH25]. Nguyen Thi Huyen Chau and Trung Phong Doan (2024) adopted specific rules for missing data, such as using the previous day's closing price for absent opening prices. They addressed logical inconsistencies, for instance, a closing price exceeding the day's high, by recalculating high and low values from available opening and closing prices [Huy24].

Data normalization or scaling is another crucial preprocessing step to ensure that features with larger magnitudes do not disproportionately influence model learning. Aldin et al. (2012) emphasize that normalization, for example, to a range of negative one-to-one, prevents the excessive effect of high-value data points and reduces prediction errors in neural networks [ADE12]. Khan et al. (2023) and Mostafavi and Hooman (2025) both utilized Min-Max normalization to rescale features, which aids in faster convergence of gradient descent algorithms [Kha+23; MH25]. Nguyen Thi Huyen Chau and Trung Phong Doan (2024) employed both Min-Max scaling for features within a zero to one hundred range and Z-Score standardization for other features to achieve uniform scales [Huy24].

Feature engineering in financial time series often involves deriving technical indicators from basic Open, High, Low, Close, and Volume (OHLCV) data. These indicators are designed to capture various aspects of market dynamics, such as trend, momentum, volatility, and volume patterns. Aldin et al. (2012) converted OHLC prices into ten technical indicators, including the Accumulation Distribution Oscillator, Moving Average Convergence Divergence (MACD), RSI, and Stochastic K percent [ADE12]. Similarly, Nguyen Thi Huyen Chau and Trung Phong Doan (2024) engineered 65 features, incorporating technical indicators like Momentum, Stochastic Oscillator, MACD, RSI, Bollinger Bands, and Money Flow Index (MFI), alongside features representing price change ratios and trading volumes over specified periods [Huy24]. Khan et al. (2023) selected nine input features for their models, including RSI, Simple Moving Average

(SMA), ADX, and Momentum, often specifying particular look-back periods, illustrated by a 14-day RSI and a 50-day SMA [Kha+23]. It was noted, however, that their construction of features such as “Previous (Open-Close)”, “Previous (Close-High)”, and “Previous (Close-Low)” introduced look-ahead bias. An observed one-row shift in the data for these features meant that they effectively used information from the period being predicted rather than strictly historical data. This form of data leakage can artificially inflate performance metrics and underscores the importance of meticulous data handling in time series forecasting [Kha+23]. Mostafavi and Hooman (2025) generated an extensive set of 88 technical indicators using the TA-Lib library, categorizing them into momentum, trend, volatility, and volume groups, underscoring the belief that technical indicators can reveal underlying market sentiment and patterns [MH25]. Htun et al. (2023) also note that technical indicators like RSI, stochastic oscillator, and MACD are commonly extracted from historical price series to analyze past patterns and predict future movements [HBP23].

Once a potentially large set of features is engineered, feature selection and dimensionality reduction become essential to mitigate the curse of dimensionality, reduce noise, prevent overfitting, and improve model interpretability and computational efficiency. Htun et al. (2023) provide a comprehensive survey of feature selection techniques, categorizing them into filter, wrapper, embedded, and information theory-based methods, and feature extraction techniques such as Principal Component Analysis (PCA) and Autoencoders (AE) [HBP23]. In practice, Mostafavi and Hooman (2025) applied PCA to their 88 technical indicators, retaining 95 percent of the variance and reducing the feature set to 35 principal components. They also performed rule-based feature selection by excluding low variance indicators, like the Chaikin Oscillator, and highly correlated features, specifically those with a Pearson’s correlation coefficient absolute value greater than 0.95, before applying PCA [MH25]. While Htun et al. (2023) highlight Random Forest (RF) and correlation criteria as widely used feature selection methods and PCA and AEs for extraction, the specific application of diverse filter, wrapper, or embedded selection techniques beyond PCA and rule-based exclusion was less detailed in the other primary research sources provided [HBP23].

2.3 Machine Learning Models for Stock Price Prediction

The prediction of stock market prices and trends has been a significant area of research, with machine learning (ML) techniques increasingly being applied to tackle this complex and volatile domain. The literature reveals various ML models, from traditional algorithms to advanced deep learning architectures. Consistent with the methodological approach of this thesis, this review will concentrate on studies where such models are primarily driven by technical indicators derived from historical price data, highlighting the machine learning models frequently adopted for this specific forecasting application.

2.3.1 Traditional Machine Learning Approaches

Simpler ML models have been foundational in stock prediction research, often serving as baselines or components in more complex systems.

Linear Regression, a basic statistical method, has been applied to predicting stock prices. Bansal et al. (2022) included Linear Regression in their comparative study of five algorithms for predicting stock prices of Indian companies, evaluating its performance alongside more complex models [BGC22]. The intrinsic non-linear nature of financial markets can limit its efficacy.

Support Vector Machines (SVM), particularly Support Vector Regression (SVR) for predicting continuous price values, are frequently employed. Henrique et al. (2018) conducted an extensive study on SVR to predict daily and up-to-the-minute stock prices using various technical indicators. They found that SVR with periodic retraining and linear kernels showed predictive power compared to a random walk model [HSK18]. Narkar (2019) also utilized SVM, among other classifiers, for predicting the direction of stock price changes based on numerous technical indicators extracted from past data [Nar19]. In their survey, Htun et al. (2023) identify SVM as a popularly deployed ML method for regression and classification tasks in finance, often using technical indicators and macroeconomic factors as inputs [HBP23].

Other traditional classifiers such as K-Nearest Neighbors (KNN), Naïve Bayes (NB), Logistic Regression (LR), and Decision Trees (DT) have also been explored, primarily for predicting the direction (up/down/neutral) of stock price movements. Narkar (2019) included KNN in his comparative study for directional prediction [Nar19]. Khan et al. (2023) evaluated nine ML models: SVM, DT, LR, NB, KNN, RF, Adaptive Boosting (AdaBoost), XGBoost, and Artificial Neural Network (ANN) for predicting stock market direction, reporting that Logistic Regression achieved the highest accuracy (85.51%) in their 1-day time frame strategy using traditional methodology [Kha+23]. Yiqiong and Xiaodong (2024) reviewed the use of Hidden Markov Models (HMM) and Bayesian Networks, noting HMM's interpretability but limitations in predicting discrete data and representing inter-variable relationships, which Bayesian Networks can address by modeling causal relationships between stock indicators [XL24].

2.3.2 Ensemble Methods

Ensemble learning techniques, which combine multiple models to improve predictive performance and robustness, are prominent in stock market forecasting.

Random Forest (RF) is a widely adopted ensemble method. Zheng et al. (2024) demonstrated that their RF model, using six technical indicators, achieved high accuracy (80-99%) in predicting long-term stock trends for Apple, Samsung, and GE, with accuracy stabilizing around 98% for forecast horizons greater than 60 days [Zhe+24]. Narkar (2019) also found RF to perform well in his study on directional prediction [Nar19]. Yang (2025) utilized RF as a benchmark model in the context of corporate financial forecasting [Yan25]. González et al. (2020) provide an extensive review and empirical comparison of various bagging (like RF) and boosting ensembles for classification tasks [Gon+20]. Htun et al. (2023) highlight RF as a successful ensemble method, often used with technical indicators for predicting stock direction and stock selection [HBP23]. Khan et al. (2023) found RF to be one of the top performers, achieving 91.27% accuracy with their proposed 15-min interval strategy [Kha+23].

Gradient Boosting Machines, such as XGBoost and LightGBM, are another powerful class of ensemble methods. Mostafavi and Hooman (2025) applied XGBoost (alongside RF, SVR, and LSTM) to predict S&P 500 index prices using 88 technical indicators, identifying it as a strong performer after PCA-based feature selection [MH25]. Khan et al. (2023) also included XGBoost and AdaBoost in their comparative study, where XGBoost showed performance comparable to RF [Kha+23]. Yang (2025) used XGBoost as a benchmark in their CNN-LSTM study [Yan25]. González et al. (2020) cover XGBoost, LightGBM, AdaBoost, and LogitBoost in their comprehensive analysis of ensemble techniques [Gon+20].

2.3.3 Neural Networks and Deep Learning

Neural networks, particularly deep learning models, have gained significant traction due to their ability to model complex non-linear relationships in financial time series.

Artificial Neural Networks (ANNs) represent an early application of neural networks in this domain. Aldin et al. (2012) focused on ANNs for predicting stock price index variations using technical indicators such as Moving Averages, RSI, CCI, and MACD [ADE12]. Htun et al. (2023) and Khan et al. (2023) also reference or apply ANNs in their respective studies [HBP23; Kha+23].

Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures are popular for their ability to capture temporal dependencies. Sirimevan et al. (2019) employed LSTM-RNN models, integrating them with sentiment data (Twitter, web news, search queries) for stock price prediction of Dow Jones Industrial

Average (DJIA) components [Sir+19]. Mozaffari and Zhang (2024) compared LSTM with Transformer and Prophet for stock index prediction, finding LSTM to be a strong contender, though ultimately outperformed by the Transformer model [MZ24]. Chang et al. (2024) conducted a comparative analysis of LSTM and GRU for predicting technology stock prices (Apple, Amazon, Google, Microsoft), concluding that GRU generally outperformed LSTM in terms of Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and training time [Cha+24]. Phuoc et al. (2024) applied LSTM with technical indicators (SMA, MACD, RSI) to predict stock trends in the Vietnamese market, achieving high accuracy (around 93%) [Phu+24]. Yang (2025) also used standalone LSTM as a benchmark against their hybrid CNN-LSTM model [Yan25]. Htun et al. (2023) list LSTM as a frequently used DL model in stock market prediction, often combined with feature selection [HBP23]. Patil et al. (2020) mentioned LSTM in the context of their graph-based deep learning models [Pat+20].

Convolutional Neural Networks (CNNs) have also been adapted for financial forecasting. Chandar (2022) proposed a TI-CNN model, which converts ten technical indicators into Gramian Angular Field images and uses a CNN for buy/sell/hold signal prediction, reporting high accuracy (mean 77.75%) on NASDAQ and NYSE data [Cha22]. Buachuen and Kantavat (2023) explored hybrid LSTM-CNN and CNN-LSTM architectures for an automated stock trading system, leveraging CNNs for spatial feature extraction from stock data and LSTMs for sequence modeling [BK23]. Yang (2025) implemented a hybrid CNN-LSTM model for corporate financial forecasting, finding it superior to standalone LSTM and other benchmarks [Yan25]. Patil et al. (2020) also utilized CNNs within their graph-based prediction framework, creating spatio-temporal convolution layers [Pat+20]. Htun et al. (2023) list CNNs as a DL method used for stock price and trend prediction [HBP23].

Transformer models, known for their attention mechanisms, are a recent advancement in time series forecasting. Mozaffari and Zhang (2024) provided a direct comparison, showing that their Transformer model outperformed both LSTM and Prophet for stock index prediction (AAL and AAME stocks) [MZ24]. A specific architecture, the Temporal Fusion Transformer (TFT), has also been explored for its ability to integrate static metadata with temporal data and its use of multi-horizon forecasting. Hu (2021) utilized TFT for stock price prediction, comparing its performance against SVR and LSTM and noting its effectiveness when sufficient training data is available [Hu21]. Buachuen and Kantavat (2023) incorporated Attention Layers, a core component of Transformers, into their deep learning models for an automated stock trading system [BK23]. Htun et al. (2023) and Liu and Wang (2024) acknowledge Transformers in their surveys of DL methods for time series forecasting, with Liu and Wang (2024) providing an extensive review of Transformer-based models like Informer and Autoformer [HBP23; LW24].

Other deep learning architectures such as Deep Belief Networks (DBNs) and Restricted Boltzmann Machines (RBMs) are mentioned by Htun et al. (2023) as having been applied in stock market prediction, often for feature extraction or as part of hybrid systems [HBP23]. Liu and Wang (2024) also survey Multilayer Perceptron (MLP)-based models like N-BEATS and DLinear for time series forecasting [LW24].

2.3.4 Reinforcement Learning

Deep Reinforcement Learning (DRL) is an emerging area in financial forecasting. Chiumera (2022) focused on Proximal Policy Optimization (PPO), a DRL algorithm, for time series forecasting in quantitative finance, using OHLCV data and technical indicators. The study compared PPO's performance against buy-and-hold strategies and explored optimal hyperparameter configurations [Chi22]. Patil et al. (2020) also explored DRL within their graph-based models [Pat+20], and Htun et al. (2023) included DRL in their survey of advanced techniques [HBP23]. Van et al. (2024) proposed hybridizing deep learning predictions with reinforcement learning; Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and Rainbow DQN to generate trading signals [TNP23].

2.3.5 Hybrid and Graph-Based Models

Several studies propose hybrid models that combine or integrate different ML techniques with other paradigms like graph theory. Patil et al. (2020) introduced a novel approach using graph theory to model spatiotemporal relationships between stocks, creating correlation-based and causation-based graphs from news co-mentions. They then applied deep learning with Graph Convolutional Networks (GCN) and traditional ML in the form of Linear Regression with community detection to this graph structure for prediction [Pat+20]. As mentioned, hybrid CNN-LSTM models are also common [Yan25; BK23].

2.3.6 Performance Metrics and Validation Strategies

The evaluation of machine learning models applied to stock prediction using technical indicators is typically characterized by consistent performance metrics and validation strategies within the reviewed literature. Key performance metrics frequently cited include accuracy, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), F1-score, and R-squared, alongside finance-specific measures such as the Sharpe ratio and maximum drawdown [Kha+23; Chi22; Cha+24; Zhe+24; HSK18]. Furthermore, these models are often benchmarked against traditional forecasting methods like ARIMA or baseline investment strategies such as buy-and-hold to gauge their relative effectiveness. Robust validation techniques, including k-fold cross-validation or rolling window approaches, are also emphasized to ensure the generalizability and reliability of the prediction results [HSK18; Huy24; Cha22]. The development of decision support systems frequently follows, incorporating these evaluated models to provide actionable insights for investors [Yan25].

2.4 Trust and Explainability in AI-Driven Financial Decision Support

Integrating Artificial Intelligence (AI) into financial decision-making has brought significant advancements in predictive accuracy and efficiency [Yan25; Cha+24; XL24]. However, the increasing complexity of these AI models, particularly black-box algorithms like deep neural networks, raises serious concerns regarding user trust and the need for model transparency [Bar+20; BB21]. This related work section reviews literature pertinent to trust and explainability in AI-driven financial decision-support systems, beginning with a brief overview of the sociological and psychological underpinnings of trust, then focusing on trust in AI decision-making, particularly within the high-stakes financial domain. It will also explore the challenges posed by opaque models and discuss strategies for building user confidence through transparency and clear explanations, including the potential of multi-model approaches and similarity-based explanations.

From a sociological and psychological perspective, trust is a fundamental component of human interaction and societal functioning. *"It involves a willingness to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party"* [Cur24; Afr+24]. This concept extends to interactions with technology, where users must trust that AI systems will operate reliably and in their best interest. In AI decision-making, particularly in finance, where outcomes can have significant monetary and personal consequences, establishing and maintaining user trust is paramount [SMM23; Yeo+25].

The financial sector, characterized by high risk and the need for accountability, presents unique challenges and opportunities for AI. While AI models can analyze vast amounts of data to identify profitable investment strategies or assess creditworthiness with high accuracy [Yan25; Cha+24; XL24], their black-box nature often hinders user acceptance and regulatory compliance [CK24; Yeo+25]. Investors and financial professionals may be hesitant to rely on predictions or recommendations if they cannot understand the underlying reasoning of the AI model [BB21].

This lack of transparency can lead to a deficit of trust, even if the model demonstrates superior performance [Afr+24].

Explainable AI (XAI) has emerged as a pivotal field in addressing these challenges. XAI aims to develop methods and techniques that make the decisions of AI systems understandable to humans [Bar+20]. In finance, XAI can justify algorithmic trading decisions, credit scoring outcomes, or fraud detection alerts, thereby fostering user confidence and enabling more informed decision-making [CK24; Yeo+25]. Several XAI techniques have been proposed, ranging from model-agnostic methods like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations), which can explain any black-box model, to model-specific approaches that leverage the internal structure of particular algorithms [Bar+20; Han+21; WL21]. These methods often focus on identifying important features, providing local explanations for individual predictions, or generating simplified surrogate models that approximate the behavior of the complex AI [WL21; Bar+20].

The concept of trustworthiness in AI is multifaceted, encompassing not only the accuracy and reliability of the model but also its fairness, robustness, and transparency [Afr+24]. Research by Schreibelmayr et al. (2023) investigated user perceptions of a financial AI assistant, finding significant differences in trust levels based on initial impressions and the perceived competence and understandability of the system [SMM23]. This highlights the importance of tailoring explanations to the target audience and their existing knowledge.

One promising avenue for enhancing trust and understanding in AI-driven financial predictions is using multi-model approaches. Instead of relying on a single, potentially opaque model, decision support systems can integrate predictions and explanations from several diverse models [Yeo+25; WL21]. This can provide a more robust and nuanced view of the prediction, highlighting areas of consensus or disagreement among models. Furthermore, comparing the reasoning of different models can offer deeper insights than a single explanation might provide.

Similarity-based explanations, which provide users with historical examples or analogous situations similar to the current prediction context, represent another valuable approach [Han+21]. In finance, this could involve showing past market scenarios or individual stock behaviors that resemble the current situation, thereby helping users to contextualize and understand the model's forecast [NSK22]. Hanawa et al. (2021) evaluated different relevance metrics for similarity-based explanations, emphasizing the importance of selecting metrics that align with human intuition [Han+21]. Naveed et al. (2022) further explored user-centric taxonomies of explanations in the financial domain, stressing the need for domain-specific information and shared understanding between the user and the AI system [NSK22].

Ultimately, building trust in AI-driven financial decision support requires a holistic approach that considers the technical aspects of model accuracy and explainability, the psychological factors influencing user perception and confidence, and the specific contextual demands of the financial domain [Afr+24; Yeo+25].

2.5 A Proposed Approach to Enhance Trust in AI Predictions

The preceding review of related work underscores both the advancements in applying machine learning to stock market prediction and the persistent challenges surrounding user trust, particularly due to the black-box nature of many sophisticated models. Drawing inspiration from these areas, this thesis proposes to investigate a novel approach to enhance trustworthiness in AI-driven stock market predictions, specifically for users who may not possess deep technical or financial expertise.

Our planned research will focus on a decision support system's design, implementation, and empirical evaluation. This system will be engineered to address the trust deficit by integrating insights from multiple machine-learning models with context derived from historically similar market patterns. The core components of our proposed approach are:

- **Leveraging Multiple Predictive Models:** Inspired by the common practice of using diverse ML models in financial forecasting (Section 2.3) and the XAI principle that multiple perspectives can offer richer insights, our system will incorporate several distinct machine learning algorithms [Yeo+25; WL21]. Instead of relying on a single prediction, users will be presented with forecasts from various models, allowing them to gauge consensus or divergence.
- **Incorporating Historical Similarity Analysis:** Building on the concepts of similarity-based explanations in XAI, the proposed system will feature a mechanism to identify and present past market periods that exhibit strong feature similarity to the current market conditions [Han+21; NSK22]. The key here is that the features used for similarity matching will be the same technical indicators and temporal window length used as input by the predictive models. This ensures that similar historical patterns are analogous regarding the information the ML models are processing.
- **Demonstrating Past Model Behavior in Similar Contexts:** A crucial aspect of our proposed approach, differentiating it from generic similarity displays, is that for each identified historically similar pattern, the system will demonstrate how the integrated machine learning models would have performed had they been presented with that specific past input sequence. This will involve showing the models' predictions for that past instance alongside the actual market outcome and the resulting prediction error. This aims to provide tangible, example-based evidence of model reliability or lack thereof under conditions analogous to the present.
- **Developing Transparent Recommendation Logic:** The system will aim to provide an overall investment recommendation (e.g., Buy, Sell, Hold). However, departing from opaque recommendations, the logic behind this suggestion will be transparently linked to the performance of the models on the identified historically similar patterns rather than being a simple aggregation of current predictions.

What sets this proposed research apart is not merely the use of multiple models or historical data, but the specific synergistic integration of these elements into a user-facing system designed explicitly to foster calibrated trust. While the literature discusses multi-model systems and similarity-based XAI, the intended contribution here is to:

1. **Implement a specific form of similarity-based explanation** where historical analogies are directly tied to the ML models' input space and are used to showcase concrete past performance of these very models. This moves beyond just showing similar past price charts to showing similar past model input and performance scenarios.
2. **Empirically evaluate the impact of this integrated system on user trust, confidence, and decision-making**, particularly for an audience without deep technical expertise. Many XAI techniques are proposed, but rigorous user studies evaluating their effectiveness in enhancing trust within the specific context of financial predictions derived from multiple, complex models augmented by such historical performance insights are less common. This evaluation will compare user responses to this multifaceted system against a more traditional, single-model prediction interface.

The ultimate goal is to investigate whether this approach; providing users with multiple model perspectives, relevant historical analogies, and direct evidence of past model behavior in similar situations, can make AI-driven stock market predictions more understandable, interpretable, and, consequently, more trustworthy. This research endeavors to provide actionable insights for designing financial AI tools that users can engage with confidently and responsibly.

Chapter 3

Data Acquisition and Description

The foundation of this research rests on reliable financial data. This chapter details the acquisition and description of the historical stock market information used throughout the study. It begins by identifying the data source and the tool used for retrieval (Section 3.1). Subsequently, the core components of the dataset, specifically Open-High-Low-Close (OHLC) prices and trading volume, are defined and explained (Section 3.2, Section 3.3). The chapter also clarifies the decision to use unadjusted price data (Section 3.4) and presents the rationale for selecting International Business Machines Corporation (IBM) stock and the daily data frequency for analysis (Section 3.5). This establishes the specific dataset that forms the basis for the preprocessing steps detailed in Chapter 4.

3.1 Data Source

The data used in this study is obtained from Yahoo Finance, a widely recognized platform for accessing comprehensive financial information, including stock data, historical prices, market summaries, and financial news. Yahoo Finance serves as an accessible and reliable data provider that is frequently used in research to retrieve historical and real-time market information as discussed in Section 2.1. The platform provides detailed data on various financial instruments, including stocks and exchange-traded funds (ETFs), giving researchers access at multiple timeframes, including intraday, daily, weekly, and monthly intervals [Sub+21].

To automate data retrieval, this research uses yfinance, a Python library developed to enable easy access to Yahoo Finance's financial data. The yfinance package provides a convenient API for downloading data directly from Yahoo Finance in a structured format suitable for analysis. With yfinance, users can access historical stock data, including open, high, low, close, and adjusted close prices, as well as volume information [MZ24]. This package is often used in quantitative finance and data science research because of its accessibility, easy integration with Python, and ability to provide reliable financial data, as discussed in Section 2.1.

3.2 OHLC Data

OHLC data is an abbreviation for open, high, low, and close prices, which are fundamental components in the technical analysis of financial data. These four data points are provided for a specific time interval, such as daily, weekly, or monthly. Analysts use them to evaluate price movements within that time period. The OHLC format is used for charting price action as it provides insight into the price behavior of a security during a specific time frame [BGC22].

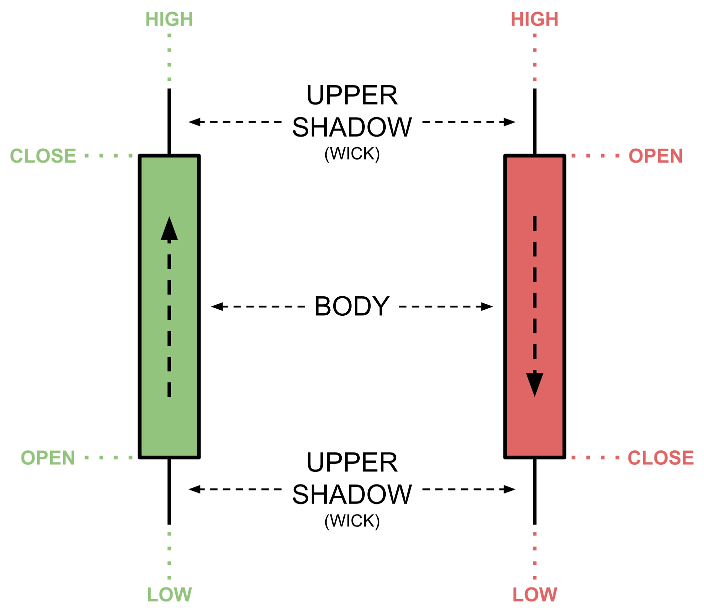


Figure 3.1: Illustrating the structure of candlestick charts used in financial markets. The left candlestick represents a bullish (green) candle, where the closing price is higher than the opening price. In contrast, the right candlestick represents a bearish (red) candle, where the closing price is lower than the opening price. Each candlestick consists of a body, which denotes the range between the opening and closing prices, and upper and lower shadows (wicks), which indicate the highest and lowest prices reached during the trading period. Figure inspired by [Nna24].

Open Price The open price refers to the price at which a financial instrument, such as a stock, begins trading at the beginning of the chosen time period. This price is essential because it is the starting point from which the price movement for the session is measured. A comparison between the open price and the closing price of the previous period can provide insight into investor sentiment and help identify gaps or sudden shifts in market dynamics [Jog24].

High Price The highest price represents the asset's highest traded price during the specified period. This value is significant because it indicates the peak of buying interest or bullish momentum within the time frame. High prices are especially relevant when detecting resistance levels, where the price may struggle to move past a certain point due to oversupply [Jog24].

Low Price The lowest price is the lowest point at which the asset trades during the selected period. This value indicates the maximum bearish momentum or lowest level of buying interest during that session. Low prices are crucial for identifying support levels, where downward pressure may ease due to an increase in demand [Jog24].

Close Price The close price is the final price at which the security trades during the period. It is often considered the most crucial of the OHLC components because it reflects the market's final consensus on the asset's value at the end of the period. The closing price is commonly used in various technical indicators. It is often compared to the open price to assess whether the market has gone up or down during the period [Jog24].

3.2.1 Interpreting OHLC Data from Candlestick Charts

Candlestick charts are a standard method of visualizing OHLC data. Each candlestick represents the open, high, low, and close prices for a given period [Mit24]. The candlestick's body is formed between the open and close prices, while the shadows, also called wicks, extend to the high and low prices, as shown in Figure 3.1.

With a green (bullish) candlestick, the open price is on the lower boundary of the body, and the close price is on the upper boundary. This configuration indicates that the price has risen during the period. In contrast, in a red (bearish) candlestick, the open price is at the body's upper limit, and the close price is at the lower limit, indicating that the price fell during the period [Mit24].

- The top shadow extends from the body to the highest price and represents the highest price level reached during the period [Jog24].
- The bottom shadow extends from the body to the lowest price and represents the lowest price level achieved during the period [Jog24].

Candlestick charts are handy for identifying patterns that signal potential price movements, such as bullish reversals, bearish continuations, or indecision in the market. By examining the relative lengths of the body and shadows, along with the color of the candlestick, traders can infer market sentiment and anticipate future price movements [Jog24].

3.3 Volume

Trading volume is an important measure in financial markets and refers to the total number of shares or contracts exchanged between buyers and sellers within a given time period [Nic23]. It is an important indicator of market activity and liquidity and helps measure the strength and importance of price movements. Higher trading volumes generally indicate a more liquid and active market where orders can be executed efficiently and with minimal price movements [Cha21]. Conversely, lower volumes indicate less activity and possibly higher slippage due to less liquidity.

In technical analysis, volume is particularly valuable because it provides context for price movements. A price movement associated with high volume is viewed as more robust and credible because it indicates broad participation by market participants. Conversely, low-volume price movements may be viewed with skepticism because they may result from limited market involvement and may not have sustainability [Nic23].

3.4 Unadjusted Data

The closing price of a stock represents the final price at the end of a market period and serves as an essential measure of the stock's actual market value [Jog24]. Although adjusted closing prices adjust this value to account for corporate actions such as stock splits, dividends, or rights issues, they do not reflect the actual transaction price of the stock at the time the market closes. Instead, these adjustments are intended to normalize the stock's historical performance for comparison purposes, removing fluctuations caused by non-market events that may affect nominal prices. However, this normalization may mask the true price dynamics within the market [Gan20].

For predictive modeling purposes, this study employs the unadjusted closing price since it directly reflects the stock's actual transaction history. Using this measure, the analysis avoids distortions from corporate actions, ensuring that the data truly represents market-driven values. Consequently, in this study, reference to the "close price" of a stock will refer to the unadjusted closing price.

3.5 Stock Selection

Negative transfer refers to the adverse impact on the performance of a target model when knowledge from a source domain is transferred, particularly when there is a significant divergence between the source and target data distributions [Wan+19]. This phenomenon can occur in stock prediction when data from multiple stocks with inherently different market dynamics, behavioral patterns, and corporate actions are used, as the resulting conflicting signals may hinder the model's ability to generalize effectively [Jog24]. Consequently, focusing the training on a single stock mitigates the risk of negative transfer, allowing the model to be more precisely tailored to that stock's unique characteristics and historical patterns, thus enhancing the precision and relevance of prediction [Wan+19].

The stock selected for this study is International Business Machines Corporation (IBM), a world leader in the technology sector. Known for its hardware, software, and cloud services innovations, IBM has an established reputation and a long-standing presence in the market [PM23]. The company went public on January 2, 1962, providing significant historical data for building and validating predictive models [Sta24]. IBM's prominent presence in the technology industry and consistent data availability over several decades make it a strong candidate for this study, providing a stable and reliable basis for analysis.

Daily data of IBM stock is used in this study as it captures meaningful price movements without losing the fundamental trends. While intraday data introduces noise and weekly data is too broad and slow to capture short-term trends, daily data strikes a balance by providing timely insights that preserve the essence of the stock's underlying story. Moreover, this frequency is the most commonly utilized in the discussed related work, Chapter 2, thereby enhancing the comparability and robustness of the findings. The dataset spans from January 2, 1962, to December 31, 2024.

Chapter 4

Data Preprocessing and Feature Engineering

Building upon the data acquisition detailed in Chapter 3, this chapter addresses the essential preparation of the historical IBM stock data for machine learning analysis. Since raw financial time series data requires careful cleaning and transformation for effective modeling, this chapter outlines the application of specific preprocessing and feature engineering methods.

The process begins with addressing missing values within the dataset to ensure completeness (Section 4.1). Following this, the target variable for the predictive models is clearly defined (Section 4.2). A core part of this chapter focuses on feature engineering, explaining how relevant technical indicators were derived from the open, high, low, close, and volume (OHLCV) data to serve as informative inputs for the models (Section 4.3). The chapter concludes by describing the procedures for chronologically splitting the data into training, validation, and test sets, applying feature scaling, and structuring the data into the sequential format required by the time series models used later in the study (Section 4.4). Together, these steps provide the properly formatted and structured data foundation necessary for the predictive modeling phase presented in Chapter 5.

4.1 Handling Missing Values

Financial datasets, despite careful collection, can contain missing entries. These gaps can arise from various factors, such as trading holidays where no data is recorded, data feed interruptions, errors in data recording, or periods of no trading activity for a specific stock. The presence of missing values can significantly impair the data quality, leading to biased analyses and unreliable performance of machine learning models, as these models often require complete datasets to function correctly or to learn patterns accurately [Kha+23]. As discussed in Section 2.2, addressing missing data is a fundamental preprocessing step.

Upon retrieving the IBM daily stock data spanning from January 2, 1962, to December 31, 2024, an initial analysis was conducted to identify missing values. Before this analysis, a precautionary step was taken to convert any potential placeholders for missing or faulty data, such as infinities, empty strings, or zero values, into Not a Number (NaN) values. The examination revealed that the "Open", "High", "Low", and "Close" price columns, as well as the "Date" column, were complete, containing no missing entries. However, the "Volume" column was found to have three missing values. While this represents a very small fraction (approximately 0.02%) of the 15,859 data points in the dataset, their presence necessitates a handling strategy to ensure data integrity for subsequent modeling.

Given the time-series nature of stock market data, where the sequence and continuity of observations are crucial, simply deleting rows with missing values is often undesirable, even for a few instances, as it can disrupt the temporal structure. Instead, imputation techniques, such as interpolating or using adjacent data points, are preferred to handle missing stock prices [Kha+23; MH25]. For this study, a sequential imputation strategy was employed. First, a forward fill (ffill()) method was applied. This method propagates the last known valid observation to fill the NaN values, a technique similar to using the previous day's value as adopted in other financial studies [Huy24]. This is based on the assumption that the most recent trading information is the best estimate for a missing data point until new information becomes available. It crucially avoids any look-ahead bias or data leakage as it exclusively uses past observations to fill the gaps. A backward fill (bfill()) method was applied following the forward fill. This fills any remaining NaN values (which could occur if missing values are present at the very beginning of the dataset, although not the case here) by propagating the following known valid observation backward. This combined approach ensures that all missing values in the "Volume" column were effectively imputed. After applying these imputation steps, a subsequent check confirmed that there were no remaining missing values in any column of the dataset, rendering it complete and ready for further preprocessing and feature engineering.

4.2 Target Variable Definition

In predictive modeling for stock markets, the target variable, often denoted as Y, is the specific outcome the model aims to forecast. This study aims to predict the daily percentage change in the stock's closing price. This metric is a common choice in financial forecasting. It represents the relative return over a single trading day and inherently normalizes price changes, making them comparable across different price levels and historical periods. This choice aligns with common practices in financial time series analysis where returns or price change ratios, rather than absolute price levels, are often modeled due to their more desirable statistical properties, such as a tendency towards stationarity, which can be beneficial for certain modeling techniques [ZE17; MH25; Pat+20].

The daily percentage change, Y_t, for a given day t is calculated using the closing price of that day, C_t, and the closing price of the immediately preceding trading day, C_{t-1}. The target Y_t thus represents the actual return realized on day t. It is crucial to note that Y_t represents the outcome for day t itself. The machine learning models will subsequently be trained to predict Y_t using input features derived from data available up to and including day $t-1$. Constructing input sequences via a sliding window ensures that only past information is used to predict Y_t, thereby automatically handling the temporal alignment without requiring an artificial backward shift of the target variable column in the dataset. The formula for the target variable is:

$$Y_t = \frac{C_t - C_{t-1}}{C_{t-1}} \tag{4.1}$$

As shown in Equation 4.1, this calculation yields a continuous variable representing the proportional increase or decrease in the stock's closing price from day $t-1$ to day t. After computing this target variable for the entire IBM dataset, its distribution was analyzed. The key descriptive statistics are summarized below:

- Mean: 0.000374 (or approximately 0.04%)
- Standard Deviation: 0.015778 (or approximately 1.58%)
- Minimum: -0.235185 (or approximately -23.52%)
- 25th Percentile (Q1): -0.007740 (or approximately -0.77%)
- Median (50th Percentile): 0.000000 (or 0.00%)
- 75th Percentile (Q3): 0.008140 (or approximately 0.81%)

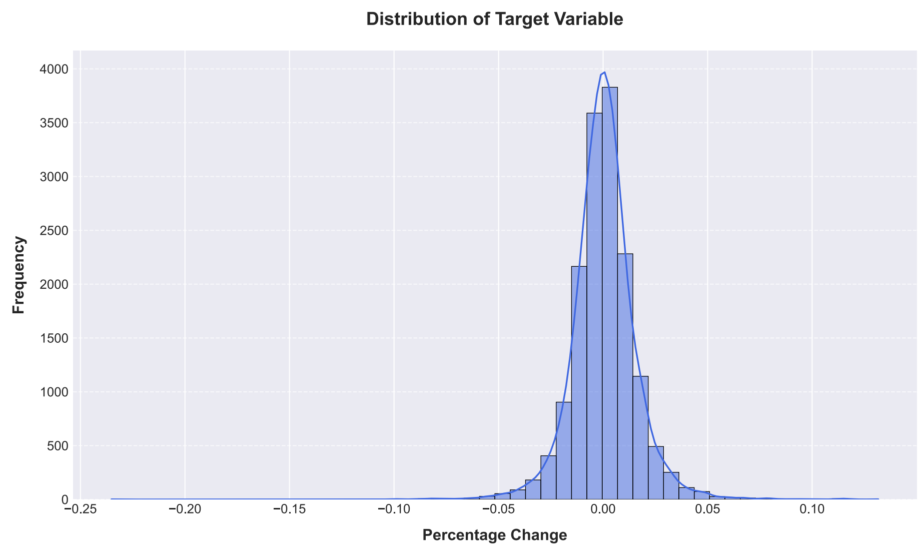


Figure 4.1: Distribution of the daily percentage change (target variable) for IBM stock from January 1962 to December 2024. The histogram displays the frequency of different percentage changes, with an overlaid kernel density estimate (KDE) curve smoothing the distribution. The x-axis represents the percentage change, and the y-axis represents the frequency.

- Maximum: 0.131636 (or approximately 13.16%)

The distribution of this target variable is visualized in Figure 4.1. The histogram illustrates that daily percentage changes are predominantly clustered near zero, indicating that most daily movements are small. The presence of tails in the distribution signifies that larger price swings, though less frequent, occur.

With the target variable defined as the one-day-ahead percentage return, the subsequent focus shifts to engineering a set of input features from the historical OHLCV data. These features, derived from information available up to and including day $t-1$, will be the basis for the machine learning models to predict Y_t.

4.3 Feature Engineering

Following the handling of missing values and the definition of the target variable (daily percentage return, Y_t), the next step is feature engineering. This process involves transforming the raw historical open, high, low, close, and volume (OHLCV) data into a set of informative input features that capture relevant market dynamics and can be used by machine learning models to predict the target variable. As discussed in Section 2.2, technical indicators derived from OHLCV data are commonly used for this purpose in financial time series forecasting [ADE12; Huy24; Kha+23; MH25].

The strategy adopted in this thesis is to carefully select a diverse set of technical indicators covering different aspects of market behavior. Following the categorization suggested by Mostafavi and Hooman (2025), the engineered features are grouped into trend, momentum, volatility, and volume indicators [MH25]. This diversification aims to give the models a comprehensive view of the market state preceding the prediction target day t. Furthermore, the selection process prioritized indicators known for their potential predictive value while aiming for relatively low inter-correlation and stationarity. As demonstrated later in this section, the chosen features exhibit these desirable properties, mitigating the need for extensive dimensionality reduction techniques like PCA or complex feature selection algorithms often required when starting with a very large, potentially redundant feature set [HBP23; MH25]. Ensuring feature stationarity is particularly important as it aligns with the assumptions of many time series models and can lead to more stable and reliable predictions [ZE17; Pat+20].

The feature set incorporates indicators inspired by Chapter 2: *Related Work*, Section 2.2, such as the Average Directional Movement Index (ADX), Exponential Moving Average (EMA) correlations, and RSI from Khan et al. (2023), and Bollinger Bands Width (BBW, referred to as `bbb` here) from Nguyen Thi Huyen Chau and Trung Phong Doan (2024) [Kha+23; Huy24]. Additional technical indicators were added to achieve a well-rounded set covering trend, momentum, volatility, and volume dynamics. All features are calculated using data available up to and including day $t-1$ to predict the target Y_t. Below are the descriptions and mathematical formulations of the eight engineered features:

4.3.1 Trend Indicators

Trend indicators aim to capture the direction and strength of the prevailing price movement [MH25].

- **Average Directional Movement Index (`adx`):** Measures the strength of a trend, regardless of its direction (up or down). A higher ADX value suggests a stronger trend, while a lower value indicates a weaker trend or a ranging market. It is calculated based on smoothed averages of directional movement over a specified period, typically 14 days [MH25]. Let $+DI_{14}$ and $-DI_{14}$ be the 14-period Positive and Negative Directional Indicators. The Directional Index (DX) is calculated as $DX = 100 \times \frac{|+DI_{14}-(-DI_{14})|}{|+DI_{14}+(-DI_{14})|}$. The ADX is typically a smoothed moving average (often an EMA or a Wilder's smoothing) of the DX values [MH25].

$$\texttt{adx}_t = \text{ADX}(\text{High}_{[t-13:t]}, \text{Low}_{[t-13:t]}, \text{Close}_{[t-13:t]}, \text{length} = 14) \tag{4.2}$$

  Where the function uses High, Low, and Close prices over the relevant look-back period ending at time t.

- **Short-Term Close-EMA Correlation (`short_close_ema_corr`):** Calculates the rolling correlation between the closing price C and its 10-period Exponential Moving Average $EMA(C, 10)$ over the preceding 10 periods. This feature aims to capture how closely the price is tracking its short-term trend line [Kha+23]. Let C_i be the closing price at time i.

$$\texttt{short\_close\_ema\_corr}_t = \text{Corr}(\{C_{t-9}, ..., C_t\}, \{EMA(C, 10)_{t-9}, ..., EMA(C, 10)_t\}) \tag{4.3}$$

  Where $\text{Corr}(X, Y)$ denotes the Pearson correlation coefficient between the sequence of values X and Y over the specified 10-period window ending at time t.

- **Long-Term Close-EMA Correlation (`long_close_ema_corr`):** Similar to the short-term version, but uses a 20-period EMA and a 20-period rolling window. This captures the relationship between the price and its longer-term trend [Kha+23].

$$\texttt{long\_close\_ema\_corr}_t = \text{Corr}(\{C_{t-19}, ..., C_t\}, \{EMA(C, 20)_{t-19}, ..., EMA(C, 20)_t\}) \tag{4.4}$$

  Where the correlation is calculated over the specified 20-period window ending at time t.

4.3.2 Momentum Indicators

Momentum indicators measure the rate of change in prices, gauging the speed and strength of price movements [MH25].

- **Relative Strength Index (`rsi`):** A popular momentum oscillator that measures the speed and change of price movements, oscillating between 0 and 100. *"It compares the magnitude of recent gains to recent losses over a specified time period"* (here, 14 days) to determine overbought or oversold conditions [Fas24; Huy24; TNP23]. Let $AvgGain_{14}$ and $AvgLoss_{14}$ be the average gains and losses over the past 14 periods ending at time t.

$$\texttt{rsi}_t = 100 - \frac{100}{1 + \frac{AvgGain_{14}}{AvgLoss_{14}}} \tag{4.5}$$

- **David Varadi Oscillator (`dv2`):** This custom indicator measures the deviation of the closing price from the midpoint of the daily high-low range, averages this deviation over the past 2 days and then calculates the percentile rank of this 2-day average deviation compared to its values over the last 126 days. It aims to capture relative price momentum or short-term overextension compared to a longer historical context, scaled between 0 and 100 [Qua24]. Let H_i, L_i, C_i be the high, low, and close prices on day i.

$$Mid_i = (H_i + L_i)/2 \tag{4.6}$$
$$D1_i = (C_i/Mid_i) - 1 \tag{4.7}$$
$$DV_i = \frac{D1_i + D1_{i-1}}{2} \tag{4.8}$$
$$\texttt{dv2}_t = \text{Rank}_{pct}(DV_t \text{ within } \{DV_{t-125}, ..., DV_t\}) \times 100 \tag{4.9}$$

Where Rank_{pct} denotes the percentile rank function over the 126-day window ending at time t.

4.3.3 Volatility Indicators

Volatility indicators measure the magnitude of price fluctuations, irrespective of direction [MH25].

- **Intraday Range (`intraday_range`):** Calculates the difference between the high (H_t) and low (L_t) price for the day t, normalized by the low price. This provides a measure of the day's price volatility relative to its price level.

$$\texttt{intraday\_range}_t = \frac{H_t - L_t}{L_t} \tag{4.10}$$

- **Bollinger Bands Width (`bbb`):** Measures the width between the upper and lower Bollinger Bands, normalized by the middle band (typically a Simple Moving Average). *"A wider band indicates higher volatility, while a narrower band suggests lower volatility"* [Gee24; Huy24]. Here, a 20-period SMA and two standard deviations are used for the bands. Let $SMA(C, 20)_t$ be the 20-period SMA of closing prices, and $StdDev(C, 20)_t$ be the 20-period standard deviation, calculated over the window ending at time t.

$$MiddleBand_t = SMA(C, 20)_t \tag{4.11}$$
$$UpperBand_t = MiddleBand_t + 2 \times StdDev(C, 20)_t \tag{4.12}$$
$$LowerBand_t = MiddleBand_t - 2 \times StdDev(C, 20)_t \tag{4.13}$$
$$\texttt{bbb}_t = \frac{UpperBand_t - LowerBand_t}{MiddleBand_t} \tag{4.14}$$

4.3.4 Volume Indicator

Volume indicators relate price movements to trading activity [MH25].

- **Volume Momentum (`volume_momentum`):** Compares the current day's trading volume (V_t) to its average volume over a recent period (here, 20 days). Values significantly above 1 suggest an unusually high trading activity, potentially confirming a price move, while values below 1 indicate lower-than-average activity [MH25]. Let $SMA(V, 20)_t$ be the 20-period SMA of volume over the window ending at time t.

$$\texttt{volume\_momentum}_t = \frac{V_t}{SMA(V, 20)_t} \tag{4.15}$$

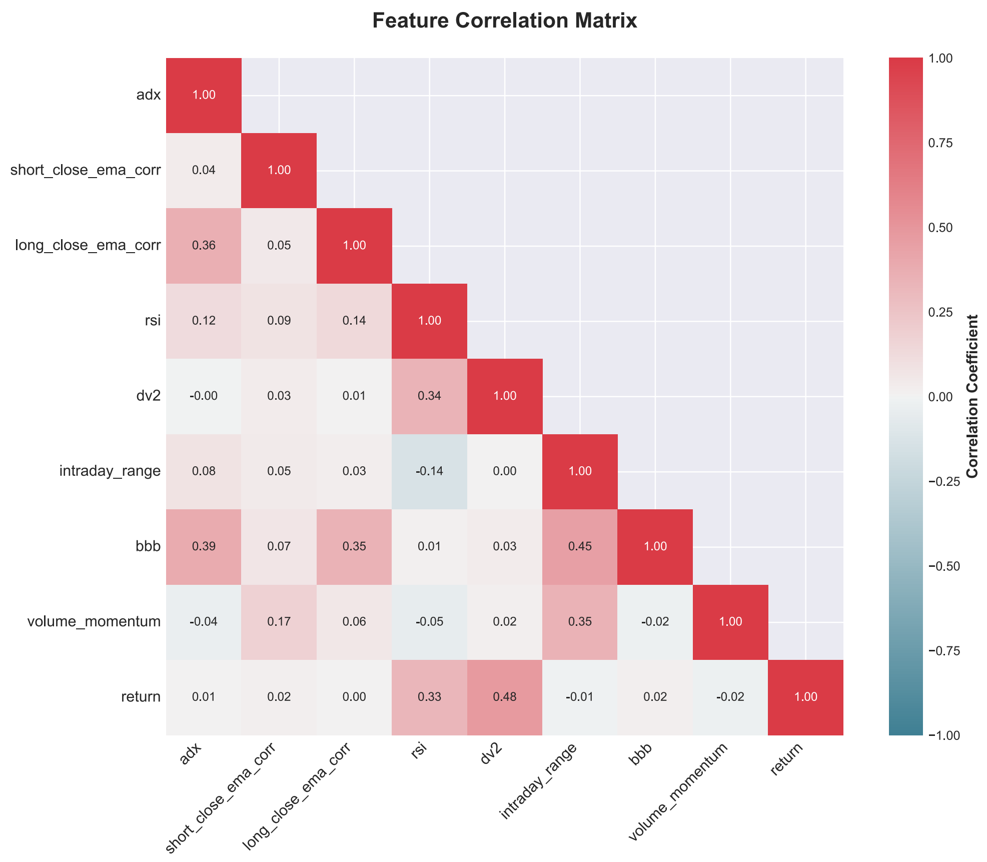


Figure 4.2: Feature Correlation Matrix. This heatmap displays the Pearson correlation coefficients between the engineered features and the target variable ('return'). Red indicates a positive correlation, blue indicates a negative correlation, and the color's intensity represents the correlation's strength. Values range from -1.00 to 1.00. The diagonal shows perfect correlation (1.00) of each feature with itself. Only the lower triangle is shown due to symmetry.

4.3.5 Feature Validation: Correlation and Stationarity

After engineering these eight features, two important checks were performed: assessing multicollinearity among features and testing for stationarity.

Correlation Analysis: A Pearson correlation matrix was computed for the engineered features and the concurrent daily percentage return (labeled "return" in the matrix, calculated as $(C_t - C_{t-1})/C_{t-1}$), visualized in Figure 4.2. The analysis revealed that the absolute correlation coefficients between all pairs of the eight *input* features remain relatively low. The highest absolute correlation observed between any two distinct input features was 0.45, occurring between `bbb` and `intraday_range`, indicating that the selected features capture different facets of market information without significant redundancy among themselves. The matrix also shows the concurrent correlation between each feature and the daily return realized on the same day. Notably, the feature `dv2` exhibits the highest absolute concurrent correlation with the daily return at 0.48. While this indicates a relatively strong relationship between the `dv2` value and the price movement on the same day, it does not directly measure the predictive power of `dv2` at time $t-1$ for the target return at time t. However, the primary goal of checking for multicollinearity among the input features is achieved, confirming they are sufficiently independent.

Stationarity Test: Many time series models perform better with stationary input data, where statistical properties like mean and variance do not change over time [ZE17; Pat+20]. The

Augmented Dickey-Fuller (ADF) test was applied to each of the eight engineered features to check for stationarity. *"The null hypothesis of the ADF test is that the time series has a unit root, meaning it is non-stationary"* [Ali23]. The test results provided strong evidence against this null hypothesis for all features. Each feature yielded a highly significant negative ADF statistic, and the corresponding p-value was effectively 0.0, which is definitively less than the conventional significance level of 0.05. Therefore, the null hypothesis was rejected for all features, concluding that all engineered input features are stationary.

4.4 Data Splitting, Scaling, and Sequence Creation

Following the definition of the target variable and the engineering and validation of features, the final data preparation phase involves partitioning the dataset, applying feature scaling, and structuring the data into sequences suitable for the time series forecasting models employed in this study.

4.4.1 Data Splitting

To evaluate the generalization capability of the predictive models and mitigate the risk of overfitting, the historical dataset is partitioned into distinct training, validation, and testing subsets. Due to the temporal dependency inherent in financial time series, this partitioning must strictly adhere to chronological order. Applying random shuffling, a technique suitable for cross-sectional data would introduce look-ahead bias and is therefore inappropriate for this context, particularly as temporal models are employed in this study [Nar19].

After an initial period required for the computation of features with the longest look-back, the IBM dataset comprises 15,732 usable daily observations spanning from July 2, 1962, to December 31, 2024. This dataset was chronologically divided according to a 70%-10%-20% allocation:

- **Training Set:** Constitutes the initial 70% of the data, encompassing 11,012 observations. This subset is utilized exclusively for training the parameters of the machine learning models. The number of predictable target samples Y_t derived from this set using a 20-day look-back is 10,992.
- **Validation Set:** Comprises the subsequent 10% of the data, consisting of 1,593 observations. This subset serves to fine-tune model hyperparameters. After applying the 20-day look-back, this split yields 1,573 predictable target samples Y_t.
- **Test Set:** Represents the final 20% of the data, containing 3,167 observations. This subset remains entirely unseen by the models during training and validation. It provides 3,147 predictable target samples Y_t for the final, unbiased assessment of the selected models' predictive performance.

4.4.2 Feature Scaling

Feature scaling is recognized as an important preprocessing step, particularly for algorithms whose performance can be influenced by the relative magnitudes of input variables, including neural networks and gradient-based optimization methods, as discussed in Section 2.2 [ADE12; Kha+23; MH25]. Appropriate scaling helps ensure that all features contribute effectively to the learning process [MH25].

Initially, an approach similar to that described by Nguyen Thi Huyen Chau and Trung Phong Doan (2024) was considered, which involved applying Min-Max scaling to features with naturally defined bounds and Z-score standardization to other features [Huy24]. Scaling the target variable was also part of this initial consideration. However, experimentation and preliminary investigations into model performance revealed that a more straightforward, uniform approach yielded greater baseline results. Consequently, the strategy of applying Z-score standardization to all eight engineered input features while leaving the target variable unscaled was adopted for this study.

Z-score standardization transforms each original feature value x to a standardized value z, such that the distribution of z has a mean of approximately zero and a standard deviation of approximately one [MZ24]. This transformation is achieved using the formula:

$$z = \frac{x - \mu}{\sigma} \tag{4.16}$$

In this equation, μ represents the mean of the feature, and σ represents its standard deviation, calculated exclusively from the values within the training dataset.

An important aspect of this scaling procedure is that the parameters μ and σ for each of the eight features were computed solely using the training data. These derived statistical parameters were then consistently applied to transform the corresponding features in the validation and test sets. This methodology is essential to prevent any information leakage from the validation or test data distributions into the model training and scaling process, thereby maintaining the integrity and unbiased nature of the following model evaluation.

The decision to leave the target variable unscaled stems from its intrinsic characteristics as a percentage change. This form already possesses desirable scale-invariant properties and often exhibits tendencies towards stationarity, making further transformation unnecessary and potentially complicating the interpretation of predictions. Maintaining the original scale of the target variable also simplifies the direct evaluation of model predictions and associated error metrics. The rationale for adopting this specific scaling approach, which involves standardizing input features and leaving the target variable unscaled, is further substantiated by the resultant model performance. This performance will be elaborated upon in detail in Chapter 5.

4.4.3 Sequence Creation for Model Input

Input sequences for the models were constructed using a look-back window of 20 trading days. This period, approximating one calendar month defines the historical feature context for predicting Y_t. A sliding window technique with a unit stride was implemented in the training, validation, and test datasets to generate input-target pairs. For every feasible prediction time point t within a given split, beginning on the 21st available day to accommodate the 20-day look-back:

- The input sequence, X_t, comprises a matrix of the eight scaled feature values observed from day $t - 20$ through $t - 1$. The dimensionality of each X_t is (20×8).
- The associated target value, Y_t, is the unscaled daily percentage return for day t, calculated according to Equation 4.1.

This methodology generates a series of overlapping input-target pairs, (X_t, Y_t), structuring the data to facilitate the models' learning of temporal dependencies.

Chapter 5

Predictive Modeling and Performance Analysis

Following the acquisition (Chapter 3) and preprocessing (Chapter 4) of the IBM stock data, this chapter focuses on the core predictive task. It begins by detailing six distinct machine learning models' selection, architecture, and implementation, as presented in Section 5.1. Subsequently, Section 5.2 presents a comprehensive comparative analysis of these models. This analysis evaluates their performance in predicting IBM's daily percentage returns across 1-day and 10-day horizons using default hyperparameters, assessing predictive accuracy (both regression and directional), computational efficiency, and the outcomes of a simple trading strategy. The results from this chapter establish a critical baseline understanding of each model's capabilities within this financial forecasting context.

5.1 Model Selection: Rationale, Architecture, and Implementation

This section details the six machine learning models selected for this study. The chosen models are Random Forest (RF), Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), a Recurrent Neural Network (RNN) implemented as Long Short-Term Memory (LSTM), Neural Basis Expansion Analysis for Time Series (N-BEATS), and Temporal Fusion Transformer (TFT). For each model, the following subsections will describe its underlying architecture, specific implementation details relevant to this research, and its key hyperparameter configurations, which were primarily kept to their default settings to establish a baseline for comparative performance analysis.

These particular models were selected by a strategy to encompass a diverse range of well-established and contemporary machine learning techniques common in financial forecasting, as discussed in Section 2.3. The models can be broadly categorized into ensemble methods and dedicated time-series deep learning architectures. The ensemble methods, Random Forest, XGBoost, and LightGBM, were chosen for their robust performance and widespread use in various predictive tasks, including financial market prediction [Gon+20; Zhe+24; MH25; Kha+23]. These models operate by aggregating the outputs of multiple decision trees, which often leads to improved predictive accuracy and better generalization compared to individual models [Gon+20]. RF is a foundational ensemble technique known for its simplicity and effectiveness [Bre01], while XGBoost and LightGBM represent advanced gradient boosting frameworks recognized for their efficiency and state-of-the-art results in many financial applications [CG16; Ke+17].

In contrast, the deep learning models, LSTM, N-BEATS, and TFT, were selected for their specialized capabilities in handling sequential data and capturing complex temporal dependencies, which are fundamental in financial time series [LW24]. LSTMs are a type of RNN specifically designed to learn long-range dependencies and are a common choice for stock prediction tasks [HS97; Sir+19; MZ24]. N-BEATS and TFT represent more recent advancements in deep learning for time series forecasting. N-BEATS offers a pure deep learning architecture based on residual blocks that have achieved strong benchmark results [Ore+20], while TFT is an attention-based model designed for multi-horizon forecasting and interpretability, capable of integrating various types of input features effectively [Lim+20; Hu21].

All six models are suitable for the regression task of predicting continuous daily percentage returns and can utilize the engineered technical indicators (past covariates) as input sequences, as reviewed in Section 2.3. Their inclusion allows for comparing different modeling philosophies, from tree-based ensembles to sophisticated neural networks. The subsequent subsections provide a detailed overview of each model's principles and specific setup used in this thesis.

5.1.1 Random Forest (RF)

This subsection details the Random Forest (RF) algorithm. *The core concepts and mathematical formulas discussed are taken from the original architecture introduced by Leo Breiman* [Bre01]. Random Forest is an ensemble learning method introduced by Leo Breiman (2001), widely utilized for both classification and regression tasks [Bre01; Zhe+24]. It operates by constructing a multitude of decision tree predictors during the training phase. *"For regression, the final prediction is the average of the predictions from all individual trees"* in the forest [Swa+25; Bre01].

The core concept of Random Forests is to build an ensemble of tree predictors $\{h(x, \Theta_k), k = 1, \ldots, K\}$, where each tree $h(x, \Theta_k)$ is grown based on a random vector Θ_k. These random vectors $\{\Theta_k\}$ are independent and identically distributed for all K trees in the forest [Bre01]. The introduction of randomness aims to decorrelate the individual trees, which, when combined, can lead to improved generalization performance and robustness compared to a single tree. According to Breiman (2001), this randomness is typically injected in two key ways:

1. **Bagging (Bootstrap Aggregating):** Each tree in the forest is trained on a different bootstrap sample of the original training data. A bootstrap sample is created by drawing N samples with replacements from the original training set of N examples [Bre01]. This means that, on average, each tree is trained on about two-thirds of the original data, with the remaining one-third (the out-of-bag samples) being left out. This process contributes to the diversity of the trees. The Scikit-learn implementation enables this by the `bootstrap=True` parameter.
2. **Feature Randomness:** When determining the best split at each decision tree node, instead of considering all available input features, Random Forests select a random subset of features. The split is optimized using only this subset [Bre01]. This further diversifies the trees and reduces the correlation between them. The number of features considered at each split is a critical parameter, often denoted as `max_features`. Breiman (2001) noted that *"using a random selection of features to split each node yields error rates that compare favorably to other methods"* [GMR04].

Individual trees within the forest are typically grown to their maximum possible depth without pruning, often using a CART (Classification and Regression Trees) like methodology [Bre01]. This corresponds to Scikit-learn parameters such as `max_depth=None`. The quality of a split for regression trees is commonly measured by criteria that aim to reduce variance, such as squared error.

As in this study, the Random Forest predictor aggregates the predictions of all K individual trees for regression tasks. If $h_k(x)$ is the prediction of the k-th tree for an input instance x, the final Random Forest prediction $\hat{y}_{RF}(x)$ is the average of these individual predictions

[Bre01]:

$$\hat{y}_{RF}(x) = \frac{1}{K} \sum_{k=1}^{K} h_k(x) \tag{5.1}$$

Breiman (2001) showed that the generalization error for Random Forests converges to a limit as the number of trees (K) becomes large, implying that Random Forests do not overfit with an increasing number of trees. The generalization error depends on the strength (accuracy) of the individual trees and the correlation between them: higher strength and lower correlation lead to better performance [Bre01]. The method is also noted for its robustness to outliers and noise [Bre01].

In this research, the Random Forest model is implemented using the `RandomForestRegressor` class from the `sklearn.ensemble` module in Scikit-learn, *with the specific parameters and their default behaviors taken from the library's official documentation* [Ped+11]. The model was configured using these default parameter settings, with the most pertinent parameters for its construction and behavior being:

- `n_estimators=100`: This sets the number of trees (K) in the forest to 100.
- `criterion='squared_error'`: The function used to measure the quality of a split. For regression, this criterion minimizes the L2 loss using the mean of each terminal node and is equivalent to variance reduction.
- `max_depth=None`: Trees are expanded until all leaves are pure or until all leaves contain fewer samples than `min_samples_split`. This aligns with Breiman's recommendation of growing trees fully without pruning.
- `min_samples_split=2`: The minimum number of samples required to split an internal node.
- `min_samples_leaf=1`: The minimum number of samples required to be at a leaf node.
- `max_features=1.0`: This parameter controls the size of the random subset of features to consider when looking for the best split. For `RandomForestRegressor`, a float value of `1.0` means that `int(1.0 * n_features)` features are considered at each split, which implies all features are considered. In this specific configuration, the diversity among trees primarily stems from the bootstrap sampling of data instances (bagging), as the feature selection randomness at each split, as described by Breiman (2001), is not active when all features are considered.
- `bootstrap=True`: Bootstrap samples are used to build each tree, implementing the bagging procedure.
- `oob_score=False`: Out-of-bag samples are not used to estimate the R^2 on unseen data.

5.1.2 Extreme Gradient Boosting (XGBoost)

This subsection details the Extreme Gradient Boosting (XGBoost) algorithm. *The core concepts and mathematical formulas discussed are taken from the original architecture introduced by Tianqi Chen and Carlos Guestrin* [CG16]. Extreme Gradient Boosting, developed by Chen and Guestrin (2016), is a highly efficient and widely adopted gradient boosting framework. Its state-of-the-art performance in many machine learning competitions and real-world applications has gained popularity due to its scalability, speed, and accuracy [CG16; MH25]. XGBoost is an ensemble learning method that builds trees sequentially, where each new tree attempts to correct the errors made by the previously trained ensemble of trees [CG16].

The core of XGBoost lies in its regularized learning objective and advanced tree construction algorithms. Given a dataset $D = \{(x_i, y_i)\}$ with n examples and m features, a tree ensemble model uses K additive functions (trees) to predict the output $\hat{y}_i$:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{5.2}$$

The space of regression trees (CARTs) is represented by the notation $\mathcal{F} = \{f(x) = w_{q(x)}\}$. Each f_k represents an independent tree structure q that maps an example to a leaf index, and w is a vector of leaf weights (scores) for the corresponding leaves. T is the number of leaves in a tree [CG16].

To learn the set of functions f_k, XGBoost minimizes a regularized objective function:

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{5.3}$$

"Where l is a differentiable convex loss function that measures the difference between the target y_i and the prediction $\hat{y}_i$" [CH15]. The term $\Omega(f_k)$ penalizes the complexity of the model's trees and is defined as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2 \tag{5.4}$$

Here, γ is the complexity cost of introducing an additional leaf (controlling pruning), and λ is the L2 regularization parameter on the leaf weights w. This regularization term helps to avoid overfitting by smoothing the final learned weights [CG16].

Because the model in Equation 5.2 uses functions as parameters, it is incompatible with conventional optimization methods that operate in Euclidean space. An additive training strategy is therefore adopted. For the t-th iteration, the prediction for a given instance i is denoted by $\hat{y}_i^{(t)}$. A new tree, f_t, is then introduced to the model with the express purpose of minimizing the following objective:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{5.5}$$

This objective can be approximated using a second-order Taylor expansion around $\hat{y}_i^{(t-1)}$:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{5.6}$$

The terms g_i and h_i denote the first and second-order gradient statistics for the loss function, defined as $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ [CG16]. After removing constant terms, the simplified objective at step t becomes:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{5.7}$$

Substituting $\Omega(f_t)$ and defining $I_j = \{i | q(x_i) = j\}$ as the set of instances in leaf j, the objective can be rewritten by summing over the leaves of the tree:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \tag{5.8}$$

For a fixed tree structure $q(x)$, the optimal weight w_j^* for leaf j is:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{5.9}$$

And the corresponding optimal objective value (structure score) is:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \tag{5.10}$$

This score is used to evaluate the quality of a tree structure q. A greedy algorithm is used to find the best tree structure by iteratively adding splits that maximize the loss reduction, given by Equation (7) [CG16]. XGBoost also incorporates novel algorithms for handling sparse data, approximate tree learning using weighted quantile sketch, and system optimizations like cache-aware access and out-of-core computation [CG16].

This study implements the XGBoost model using the `xgb.XGBRegressor` class from the `xgboost` Python package. The model was configured using its default parameter settings, *with the specific parameters and their default behaviors taken from the library's official documentation* [Dev22]. The most significant default parameters influencing tree construction and regularization for the `gbtree` booster are:

- `booster='gbtree'`: Uses tree-based models.
- `eta=0.3` (alias: `learning_rate`): Step size shrinkage to prevent overfitting.
- `gamma=0` (alias: `min_split_loss`): Minimum loss reduction required to make a further partition on a leaf node.
- `max_depth=6`: Maximum depth of a tree.
- `min_child_weight=1`: Minimum sum of instance weight (hessian) needed in a child.
- `subsample=1`: Subsample ratio of the training instances.
- `colsample_bytree=1`: Subsample ratio of columns when constructing each tree.
- `lambda=1` (alias: `reg_lambda`): L2 regularization term on weights.
- `alpha=0` (alias: `reg_alpha`): L1 regularization term on weights.
- `tree_method='auto'`: XGBoost automatically selects the best tree construction algorithm.
- `objective='reg:squarederror'`: Specifies regression with squared loss as the learning task.

5.1.3 Light Gradient Boosting Machine (LightGBM)

This subsection details the Light Gradient Boosting Machine (LightGBM) algorithm. *The core concepts and mathematical formulas discussed are taken from the original architecture introduced by Ke et al.* [Ke+17]. Light Gradient Boosting Machine, proposed by Ke et al. (2017), is another gradient boosting framework designed for high efficiency and scalability, particularly with large datasets and high feature dimensions [Ke+17; Gon+20]. While sharing the foundational principles of Gradient Boosting Decision Trees (GBDT) with algorithms like XGBoost, LightGBM introduces several novel techniques to accelerate the training process and reduce memory usage without significant loss in accuracy [Ke+17].

The primary challenge in conventional GBDT implementations, especially with large datasets, is the time-consuming process of finding the best split points. This often involves scanning all data instances for each feature to estimate the information gain of all possible splits. LightGBM addresses this bottleneck through two main architectural innovations [Ke+17]:

1. **Gradient-based One-Side Sampling (GOSS):** In GBDT, data instances with different gradients contribute differently to the computation of information gain. Instances with larger gradients (i.e., those currently under-trained or poorly predicted) play a more significant role. GOSS leverages this insight by keeping all instances with large gradients

and randomly sampling instances with small gradients. GOSS introduces a constant multiplier for the information gain calculation of the small-gradient instances to compensate for the change in data distribution due to sampling. This method allows for a more accurate estimation of information gain with a much smaller data size than uniform random sampling, especially when the information gain values have a large range [Ke+17].

2. **Exclusive Feature Bundling (EFB):** In many real-world applications, feature spaces are often very sparse, with many features being mutually exclusive (i.e., they rarely take non-zero values simultaneously, such as one-hot encoded features). EFB bundles such exclusive features into a single feature, effectively reducing the number of features to consider. The problem of finding the optimal bundling is NP-hard, but LightGBM employs a greedy algorithm that can achieve a good approximation. This bundling reduces the complexity of histogram building from $O(\text{data} \times \text{feature})$ to $O(\text{data} \times \text{bundle})$, where bundle is much smaller than feature. This significantly speeds up training without substantially hurting accuracy [Ke+17].

LightGBM primarily uses a histogram-based algorithm to find split points. Instead of sorting continuous feature values, it buckets them into discrete bins and constructs feature histograms. This approach is more efficient regarding memory consumption and training speed than pre-sorted algorithms, especially for large datasets [Ke+17]. The GOSS and EFB techniques are integrated with this histogram-based approach.

Another distinctive feature of LightGBM is its use of a leaf-wise tree growth strategy with depth constraints, as opposed to the level-wise growth strategy common in many other GBDT implementations (though XGBoost also supports leaf-wise). In leaf-wise growth, the tree grows by splitting the leaf, resulting in the largest loss reduction. This can lead to more complex trees and potentially overfitting if not regularized properly, but it often converges faster and can achieve lower loss on the training data. LightGBM controls complexity by limiting the maximum depth (`max_depth`) and the maximum number of leaves (`num_leaves`) [Ke+17].

The learning process and objective function in LightGBM are similar to those in other GBDT frameworks like XGBoost, involving the sequential addition of trees that fit the current ensemble's negative gradients (residual errors). Regularization techniques are also employed.

This study implements the LightGBM model using the `lightgbm.LGBMRegressor` class from the `lightgbm` Python package. The model was configured using its default parameter settings, *with the specific parameters and their default behaviors taken from the library's official documentation* [Dev25]. Key default parameters relevant to its architecture and performance include:

- `boosting_type='gbdt'`: Specifies the traditional Gradient Boosting Decision Tree.
- `num_leaves=31`: Maximum tree leaves for base learners. This is a primary parameter for controlling tree complexity in leaf-wise growth.
- `max_depth=-1`: Maximum tree depth for base learners, where -1 means no limit. However, complexity is often better controlled by `num_leaves`.
- `learning_rate=0.1`: Boosting learning rate.
- `n_estimators=100`: Number of boosted trees to fit.
- `subsample_for_bin=200000`: Number of samples used for constructing feature bins (histograms).
- `objective='regression'`: Specifies regression with L2 loss as the learning task.
- `min_split_gain=0.0`: Minimum loss reduction required to make a further partition.
- `min_child_weight=0.001`: Minimum sum of instance weight (hessian) needed in a child.
- `min_child_samples=20`: Minimum data instances needed in a child (leaf).
- `subsample=1.0`: Subsample ratio of the training instance.

- `colsample_bytree=1.0`: Subsample ratio of columns when constructing each tree.
- `reg_alpha=0.0`: L1 regularization term on weights.
- `reg_lambda=0.0`: L2 regularization term on weights.

5.1.4 Recurrent Neural Network: Long Short-Term Memory (LSTM)

This subsection details the Long Short-Term Memory (LSTM) algorithm. *The core concepts and mathematical formulas discussed are taken from the original architecture introduced by Sepp Hochreiter and Jurgen Schmidhuber* [HS97]. Long Short-Term Memory networks, introduced by Hochreiter and Schmidhuber (1997), are a special type of Recurrent Neural Network (RNN) specifically designed to address the vanishing and exploding gradient problems that can occur when training traditional RNNs on long sequences [HS97]. LSTMs are well-suited for learning from sequential data with long-range dependencies, making them a popular choice for time series forecasting [HS97; Sir+19].

The core architectural innovation of an LSTM unit is its memory cell (c_t) and a system of gates that regulate the flow of information into and out of this cell. These gates are multiplicative units that learn to open and close access to the cell's state, allowing the LSTM to store information over extended time intervals and selectively update or forget it [HS97]. An LSTM cell typically consists of three main gates:

1. **Input Gate (i_t):** Controls which new information from the current input (x_t) and the previous hidden state (h_{t-1}) should be stored in the cell state. It decides how much of the candidate cell state ($\tilde{c}_t$) is added to the current cell state.

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{5.11}$$

$$\tilde{c}_t = \tanh(W_{ic}x_t + b_{ic} + W_{hc}h_{t-1} + b_{hc}) \tag{5.12}$$

   Where σ is the sigmoid activation function, and W and b terms represent weights and biases for the respective connections.

2. **Forget Gate (f_t):** Determines what information should be discarded from the previous cell state (c_{t-1}). *"It looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each number in the cell state c_{t-1}. A 1 represents 'completely keep this', while a 0 represents 'completely get rid of this'"* [Cor21].

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{5.13}$$

3. **Output Gate (o_t):** Regulates which parts of the current cell state (c_t) should be outputted as the hidden state (h_t). *"The cell state is passed through a* tanh *function (to push values between -1 and 1) and then multiplied by the output of the sigmoid gate"* [KYA19].

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{5.14}$$

$$h_t = o_t \odot \tanh(c_t) \tag{5.15}$$

The cell state itself is updated by combining the influence of the forget gate on the previous cell state and the input gate on the candidate cell state:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5.16}$$

This gating mechanism allows LSTMs to maintain a constant error carousel within the memory cells, enabling constant error flow through self-connected units and thus mitigating the vanishing gradient problem over long time lags [HS97]. The ability to learn what to remember, what to forget, and what to output makes LSTMs powerful for modeling complex temporal dynamics. Multiple LSTM layers can be stacked to create deeper recurrent networks, potentially capturing more abstract temporal features.

This study implements the LSTM model using the `pytorch_forecasting.LSTM` class from the PyTorch Forecasting library, *with the specific parameters and their default behaviors taken from the library's official documentation* [Bei20]. The model was configured using these default parameter settings, with the primary architectural choices being:

- `cell_type='LSTM'`: Explicitly uses the LSTM cell architecture (as opposed to GRU, another gated RNN variant).
- `hidden_size=10`: The number of features in the hidden state h. This is a key hyperparameter that determines the capacity of the LSTM layer.
- `rnn_layers=2`: The number of recurrent (LSTM) layers stacked on top of each other.
- `dropout=0.1`: Dropout probability applied to the outputs of each LSTM layer except the last layer for regularization.
- `loss=NormalDistributionLoss()`: The default loss function assumes the target variable follows a normal distribution and predicts its mean and standard deviation.

5.1.5 Neural Basis Expansion Analysis for Time Series (N-BEATS)

This subsection details the Neural Basis Expansion Analysis for Time Series (N-BEATS) algorithm. *The core concepts and mathematical formulas discussed are taken from the original architecture introduced by Oreshkin et al.* [Ore+20]. Neural Basis Expansion Analysis for Time Series, proposed by Oreshkin et al. (2020), is a deep neural architecture designed for univariate time series point forecasting. It is notable for its pure deep learning approach, which achieved state-of-the-art results on several well-known forecasting benchmarks without relying on time-series-specific components or complex feature engineering [Ore+20; LW24].

The architecture of N-BEATS is based *"on a deep stack of fully connected layers with backward and forward residual links"* [Kar23]. The fundamental building unit is a **block**, which takes a segment of the time series (lookback window or backcast from a previous block) as input and produces two outputs: a forecast for a future segment and a backcast, which is an estimate of its input [Ore+20].

A block internally consists of a stack of fully connected (FC) layers with ReLU non-linearities. This part of the block processes the input x_l (where l is the block index) to predict expansion coefficients for a forward basis (θ_l^f) and a backward basis (θ_l^b). These coefficients are then projected onto their respective basis functions (g_l^f and g_l^b) to produce the partial forecast $\hat{y}_l$ and the backcast $\hat{x}_l$ for that block. The operation can be described as [Ore+20]:

$$h_{l,1} = \mathrm{FC}_{l,1}(x_l), \quad h_{l,2} = \mathrm{FC}_{l,2}(h_{l,1}), \ldots, h_{l,N_L} = \mathrm{FC}_{l,N_L}(h_{l,N_L-1}) \tag{5.17}$$

$$\theta_l^b = \mathrm{LINEAR}_l^b(h_{l,N_L}), \quad \theta_l^f = \mathrm{LINEAR}_l^f(h_{l,N_L}) \tag{5.18}$$

$$\hat{x}_l = g_l^b(\theta_l^b), \quad \hat{y}_l = g_l^f(\theta_l^f) \tag{5.19}$$

Where ReLU is included in FC layers, and LINEAR layers are simple linear projections. N_L is the number of layers within a block.

Blocks are organized into **stacks**. N-BEATS employs a doubly residual stacking principle. The input to the first block of a stack is the original lookback window (or the output of a previous stack). For subsequent blocks within the same stack, the input x_l is the residual from the previous block: $x_l = x_{l-1} - \hat{x}_{l-1}$ [Ore+20]. This allows downstream blocks to focus on signal components not well approximated by earlier blocks.

The overall forecast from a single stack is the sum of the partial forecasts from all blocks within that stack: $\hat{y}_{\text{stack}} = \sum_l \hat{y}_l$. The architecture can consist of multiple stacks, where the input to a subsequent stack is the backcast residual from the previous stack. The final model forecast is the sum of the forecasts from all stacks [Ore+20].

N-BEATS can be configured in two main ways [Ore+20]:

1. **Generic Architecture:** The basis layers g_l^b and g_l^f are learnable linear projections. This configuration does not rely on time-series-specific knowledge and aims to learn the basis functions from data.
2. **Interpretable Architecture:** The basis layers g_l^b and g_l^f are fixed to specific functional forms to reflect common time series decompositions, such as trend and seasonality. For the trend stack, g_l^f produces a polynomial trend based on θ_l^f. For the seasonality stack, g_l^f produces a Fourier series based on θ_l^f. This allows the model's output to be decomposed into interpretable components.

This study implements the N-BEATS model using the `pytorch_forecasting.NBeats` class from the PyTorch Forecasting library, *with the specific parameters and their default behaviors taken from the library's official documentation* [Bei20]. The model was configured using these default parameter settings for the generic version, as the primary goal is predictive performance rather than interpretability in this specific context. Key default parameters for the generic configuration include:

- `stack_types=['generic']`: Uses only generic stacks.
- `num_blocks=[1]`: The number of blocks within each stack. For a purely generic N-BEATS, the default often implies one block per stack, with the effective depth controlled by the number of stacks.
- `num_block_layers=[4]`: Number of fully connected layers with ReLU activation per block.
- `widths=[512]`: Widths of the fully connected layers in the blocks.
- `sharing=[False]`: Weights are not shared across blocks within a stack.
- `expansion_coefficient_lengths=[32]`: Length of the expansion coefficients θ^f and θ^b.
- `prediction_length`: Set based on the forecast horizon (1 and 10 days in this study).
- `context_length`: Length of the lookback window (20 days in this study).
- `loss=MASE()`: Default Mean Absolute Scaled Error.
- `backcast_loss_ratio=0.0`: No weight is given to the backcast loss during training by default.

5.1.6 Temporal Fusion Transformer (TFT)

This subsection details the Temporal Fusion Transformer (TFT) algorithm. *The core concepts and mathematical formulas discussed are taken from the original architecture introduced by Lim et al.* [Lim+20]. The Temporal Fusion Transformer, introduced by Lim et al. (2020), is a novel attention-based deep learning architecture designed explicitly for interpretable multi-horizon time series forecasting. It aims to combine high performance with the ability to provide insights into temporal dynamics by effectively handling a complex mix of inputs, including known future inputs, static covariates, and past-observed extrinsic time series [Lim+20].

TFT's architecture is built upon several specialized components designed to address the heterogeneity of forecasting inputs and learn temporal relationships at different scales:

1. **Gating Mechanisms (Gated Residual Networks - GRNs):** Throughout the network, Gated Linear Units (GLUs) and Gated Residual Networks (GRNs) are used. GRNs allow the network to adapt non-linear processing or skip over layers if simpler processing is sufficient for a given input or dataset. This provides flexibility and can improve learning efficiency by controlling the information flow and depth of processing [Lim+20]. A GRN

takes a primary input a and an optional context vector c, yielding:

$$\mathrm{GRN}_\omega(a, c) = \mathrm{LayerNorm}(a + \mathrm{GLU}_\omega(\eta_1)) \tag{5.20}$$

$$\eta_1 = W_{1,\omega}\eta_2 + b_{1,\omega} \tag{5.21}$$

$$\eta_2 = \mathrm{ELU}(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega}) \tag{5.22}$$

Where LayerNorm is layer normalization, ELU is the Exponential Linear Unit, and GLU is a Gated Linear Unit:

$$\mathrm{GLU}_\omega(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega}) \odot (W_{5,\omega}\gamma + b_{5,\omega}) \tag{5.23}$$

This gating allows TFT to control the contribution of non-linear transformations [Lim+20].

2. **Variable Selection Networks:** TFT employs variable selection networks to handle the potentially large number of input features and select the most relevant ones for a given time step and entity. These networks assign instance-wise importance weights to each input variable (static, past, and future) using GRNs and a Softmax layer. This allows the model to focus on salient features and ignore irrelevant or noisy ones, which also aids interpretability [Lim+20]. For a flattened vector of transformed past inputs Ξ_t and a static context c_s, the selection weights v_{χ_t} are:

$$v_{\chi_t} = \mathrm{Softmax}(\mathrm{GRN}_{v_\chi}(\Xi_t, c_s)) \tag{5.24}$$

Each selected variable $\xi_t^{(j)}$ is then further processed by its own GRN before being combined: $\tilde{\xi}_t = \sum_j v_{\chi_t}^{(j)} \mathrm{GRN}_{\tilde{\xi}(j)}(\xi_t^{(j)})$ [Lim+20].

3. **Static Covariate Encoders:** Static metadata (e.g., store location, item ID) are encoded using separate GRN encoders into context vectors (c_s, c_c, c_h, c_e). These context vectors are then injected into various parts of the temporal processing pipeline to condition the learning of temporal dynamics on static attributes. This allows the model to learn entity-specific patterns [Lim+20].

4. **Temporal Processing:** TFT employs a sequence-to-sequence layer (typically LSTM-based) for local processing of known and observed time-varying inputs. This captures short-term temporal patterns. TFT utilizes an interpretable multi-head self-attention mechanism to learn long-term dependencies across different time steps. This attention is modified from standard transformer attention to share values across heads and use additive aggregation, which enhances interpretability by allowing direct assessment of attention weights for feature importance over time [Lim+20]. The interpretable multi-head attention output $\tilde{H}$ is:

$$\mathrm{InterpretableMultiHead}(Q, K, V) = \tilde{H}\tilde{W}_H, \quad \text{where } \tilde{H} = \left(\frac{1}{N_H}\sum_{h=1}^{N_H} A(QW_Q^{(h)}, KW_K^{(h)})\right) VW_V \tag{5.25}$$

Where N_H is the number of heads, and $A(\cdot)$ is the scaled dot-product attention.

5. **Prediction Intervals:** TFT generates quantile forecasts to provide prediction intervals, simultaneously predicting multiple percentiles (e.g., 10th, 50th, 90th) for each future time step using linear transformations of the final decoder output [Lim+20].

The architecture involves processing static inputs and then concurrently processing past and future time-varying inputs through their respective variable selection networks and LSTM encoders/decoders. The outputs are then fed into a temporal self-attention layer, followed by GRNs and a final output layer for quantile predictions [Lim+20].

This study implements the TFT using the `pytorch_forecasting.TFT` class from the PyTorch Forecasting library, *with the specific parameters and their default behaviors taken from the library's official documentation* [Bei20]. The model was configured using these default parameters. Key defaults include:

- `hidden_size=16`: Main hidden size of the network.
- `lstm_layers=1`: Number of LSTM layers for local processing.
- `dropout=0.1`: Dropout rate.
- `output_size=7`: Number of quantiles to predict.
- `loss=QuantileLoss()`: Default loss function.
- `attention_head_size=4`: Number of attention heads.
- `hidden_continuous_size=8`: Hidden size for processing continuous variables.
- `learning_rate=0.001`.

5.2 Performance Analysis and Discussion

Building upon the data preprocessing and feature engineering detailed in Chapter 4 and the model architectures described in Section 5.1, this section presents a comparative analysis of the six selected machine learning models. The evaluation focuses on their predictive accuracy for IBM stock's daily percentage returns, their computational efficiency during training and inference, and the performance of a simple trading strategy based on their predictions. This analysis is conducted for two distinct prediction horizons: 1 day ahead and 10 days ahead, using default hyperparameter settings for each model to establish a baseline comparison. The intrinsic difficulty in stock market prediction due to its volatile and non-linear nature is a well-acknowledged challenge, as discussed in Chapter 1 [Pat+20; MZ24].

5.2.1 Evaluation Metrics

To provide a holistic view of model performance, a diverse set of metrics is employed, consistent with practices in financial forecasting literature as outlined in Section 2.3 [HBP23; Cha+24; HSK18].

Computational Efficiency Metrics

These metrics quantify the resource utilization and speed of the models:

- **CPU Usage (%):** Average percentage of CPU utilized during training.
- **Memory Usage (MB):** Peak RAM utilized during training.
- **GPU Usage (%):** Average percentage of GPU utilized during training.
- **Training Time (seconds):** Total time taken to train the model.
- **Single Instance Prediction Time (seconds):** Time taken to predict a single input sequence.

Regression Performance Metrics

These metrics evaluate how well the models predict the continuous value of the daily percentage return.

- **Mean Absolute Error (MAE):** The mean absolute deviation between the predicted outcomes $\hat{y}_i$ and the true values y_i [LW24]. It measures the average magnitude of errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{5.26}$$

- **Root Mean Squared Error (RMSE):** The root of the mean of the squared deviations between the predicted and observed values [LW24]. It penalizes larger errors more heavily.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5.27}$$

- **R-squared (R^2) Score:** The extent to which variation in the dependent variable can be explained by the predictor variables [LW24]. An R^2 of 1 indicates perfect prediction, zero indicates the model performs no better than predicting the mean $\bar{y}$, and negative values indicate worse performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5.28}$$

Directional (Classification) Performance Metrics

These metrics assess the models' ability to correctly predict the direction of the price movement (up or down), treating the regression output as a binary classification problem (positive return = Up, negative/zero return = Down).

- **Average Precision Score:** Summarizes the precision-recall curve, representing the weighted mean of precisions achieved at each threshold. It is useful for evaluating performance on potentially imbalanced class distributions of up/down movements.
- **Directional Prediction Accuracy (%):** The percentage of times the model correctly predicts whether the stock return will be positive (Up) or non-positive (Down).

Strategy Performance Metrics

These metrics evaluate the profitability of a hypothetical trading strategy based on the models' predictions. The strategy is to go long (buy) if the predicted return is positive and short (sell) if the expected return is negative or zero. Positions are held for the prediction horizon (1 day or 10 days). No transaction costs or slippage are considered, which is a common simplification in initial strategy backtests [BK23]. The Buy & Hold return serves as a benchmark.

- **Total Return (%):** The cumulative percentage return generated by the strategy over the test period. If P_0 is the initial portfolio value and P_f is the final portfolio value:

$$\text{Total Return} = \frac{P_f - P_0}{P_0} \times 100\% \tag{5.29}$$

- **Annualized Return (%):** The geometric average amount of money earned by an investment each year over a given time period N_y (in years).

$$\text{Annualized Return} = \left(\left(1 + \frac{\text{Total Return}}{100}\right)^{\frac{1}{N_y}} - 1\right) \times 100\% \tag{5.30}$$

- **Sharpe Ratio:** This value evaluates the return generated above the risk-free rate (R_f) for each unit of total risk, with risk being quantified by the standard deviation of returns (σ_P). A higher Sharpe Ratio indicates better performance for the level of risk taken [BK23]. The general formula is:

$$\text{Sharpe Ratio} = \frac{R_P - R_f}{\sigma_P} \tag{5.31}$$

This study calculates the Sharpe Ratio using daily returns and then annualizes them. Assuming a risk-free rate R_f of 0, the annualized Sharpe Ratio is computed as:

$$\text{Annualized Sharpe Ratio} = \sqrt{N_{trading\_days}} \times \frac{\text{Mean(Daily Returns)}}{\text{StdDev(Daily Returns)}} \tag{5.32}$$

Where $N_{trading\_days}$ is the number of trading days in a year, taken as 252 in this research. This formula corresponds to the implementation used to report Sharpe Ratios in Table 5.6.

- **Maximum Drawdown (%):** The largest peak-to-trough decline during a specific period of an investment. It is an indicator of downside risk.

$$\text{Maximum Drawdown} = \frac{\text{Trough Value} - \text{Peak Value}}{\text{Peak Value}} \times 100\% \tag{5.33}$$

5.2.2 Computational Efficiency Analysis

The computational resources and time required for training and prediction are practical considerations, especially when dealing with large financial datasets and the need for timely forecasts [Ke+17]. All model training, evaluation, and computational efficiency metrics reported in this section were performed on a high-end workstation equipped with an Intel Core i9-14900KS Central Processing Unit (CPU), 64GB of Random Access Memory (RAM), and an NVIDIA GeForce RTX 4090 Graphics Processing Unit (GPU). Table 5.1 summarizes these metrics for all models across both 1-day and 10-day prediction horizons.

Table 5.1: Computational Efficiency Metrics for 1-Day and 10-Day Prediction Horizons

Model	CPU (%)	Mem (MB)	GPU (%)	Train (s)	Pred (s)
1-Day Prediction Horizon					
Random Forest	6	715	0	402	0.00800
XGBoost	12	950	45	0.6	0.01900
LightGBM	27	635	35	0.5	0.00445
RNN LSTM	7	940	100	145	0.02130
N-BEATS	7.5	1020	100	1201	0.14921
TFT	8	1080	100	796	0.05129
10-Day Prediction Horizon					
Random Forest	6	830	0	992	0.00400
XGBoost	14	950	100	2.5	0.05425
LightGBM	96	760	85	5.5	0.00950
RNN LSTM	7	940	100	143	0.02824
N-BEATS	7.5	1020	100	1203	0.09185
TFT	8	1080	100	826	0.06145

Key observations from Table 5.1:

- **Training Time:** Gradient boosting models (XGBoost, LightGBM) are exceptionally fast to train, taking only seconds. LightGBM is the fastest. Random Forest training time is considerably longer. Among the deep learning models, LSTM is the quickest to train, while N-BEATS and TFT require significantly more time, with N-BEATS being the slowest. Training times generally increase for the 10-day horizon, especially for Random Forest and the boosting models.
- **Prediction Time:** LightGBM offers the fastest single instance prediction time, followed by Random Forest. The deep learning models, particularly N-BEATS, have higher prediction latencies.
- **Resource Usage:** Random Forest is purely CPU-based. XGBoost and LightGBM can leverage GPU, significantly speeding up their training. The deep learning models (LSTM,

N-BEATS, TFT) fully utilize the GPU when available. Memory usage is relatively consistent, with TFT and N-BEATS using slightly more. LightGBM has notably high CPU usage when training for the 10-day horizon, possibly due to more complex tree structures or optimization routines for the multi-output regression.

Overall, LightGBM and XGBoost demonstrate superior computational efficiency in training speed, while LightGBM also excels in prediction speed. Deep learning models, especially N-BEATS and TFT, are more computationally intensive.

5.2.3 Predictive Accuracy Analysis

Regression Performance Analysis

The regression performance assesses how accurately models predict the magnitude of daily percentage returns. Results for the training and test sets are presented in Table 5.2 and Table 5.3, respectively.

Table 5.2: Regression Metrics on the Training Set for 1-Day and 10-Day Prediction Horizons

Model	MAE	RMSE	R^2
1-Day Prediction Horizon			
Random Forest	0.0043	0.0061	0.8572
XGBoost	0.0044	0.0058	0.8713
LightGBM	0.0084	0.0113	0.5180
RNN LSTM	0.0101	0.0135	0.3053
N-BEATS	0.0031	0.0045	0.9230
TFT	0.0151	0.0203	-0.5707
10-Day Prediction Horizon			
Random Forest	0.0136	0.0183	0.8658
XGBoost	0.0136	0.0177	0.8744
LightGBM	0.0270	0.0350	0.5094
RNN LSTM	0.0330	0.0428	0.2678
N-BEATS	0.0112	0.0139	0.9229
TFT	0.0440	0.0584	-0.3574

Table 5.3: Regression Metrics on the Test Set for 1-Day and 10-Day Prediction Horizons

Model	MAE	RMSE	R^2
1-Day Prediction Horizon			
Random Forest	0.0098	0.0145	-0.0152
XGBoost	0.0107	0.0156	-0.1792
LightGBM	0.0099	0.0146	-0.0283
RNN LSTM	0.0106	0.0158	-0.2066
N-BEATS	0.0113	0.0159	-0.2246
TFT	0.0141	0.0195	-0.8230
10-Day Prediction Horizon			
Random Forest	0.0332	0.0446	0.0036
XGBoost	0.0370	0.0489	-0.1949
LightGBM	0.0335	0.0452	-0.0220
RNN LSTM	0.0367	0.0499	-0.2461
N-BEATS	0.0374	0.0498	-0.2435
TFT	0.0494	0.0644	-1.0661

Training Set Performance: On the training data (Table 5.2), N-BEATS achieves the best fit for both 1-day and 10-day horizons, with R^2 scores around 0.923, indicating it explains over 92% of the variance in training returns. This is visually confirmed in Figure 5.1, where N-BEATS predictions closely follow the actual target train values. XGBoost and Random Forest also show strong performance on the training set with R^2 scores around 0.86-0.87. LightGBM and LSTM exhibit moderate fits, while TFT performs poorly on the training data, with negative R^2 scores suggesting it fails to capture even basic patterns in the training data with default settings. This poor performance of TFT on training data with default settings for this specific univariate task using only past covariates might indicate that its architecture, designed for complex multi-horizon forecasting with various input types, may require more specific tuning or richer input features to be effective here [Lim+20; Hu21].

Test Set Performance: The performance on the test set (Table 5.3) reveals a stark contrast. For the 1-day horizon, all models yield negative R^2 scores, signifying that their predictions are worse than simply predicting the average return of the test set. MAE and RMSE values are also considerably higher than on the training set. This indicates significant overfitting for models like N-BEATS, Random Forest, and XGBoost, which performed very well on training data. Figure 5.2 illustrates this for N-BEATS, where test set predictions deviate substantially from actuals. The difficulty of stock prediction is well-documented, with many studies showing

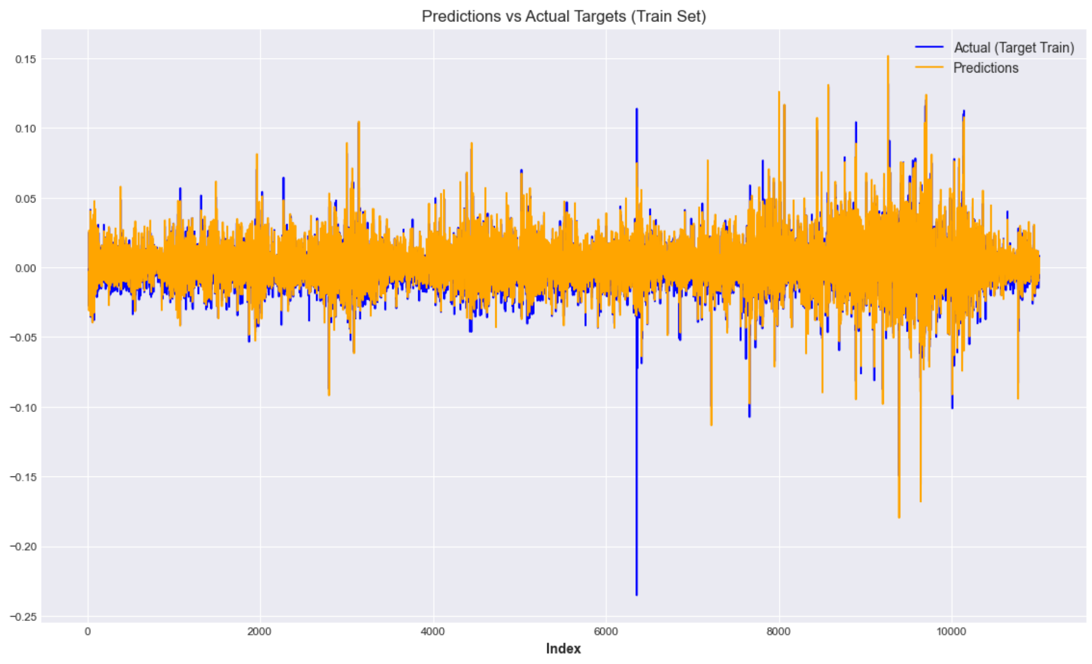


Figure 5.1: Predictions vs Actual Targets (Train Set) for N-BEATS (1-Day Horizon). The blue line represents the actual daily percentage returns (target train values), and the orange line represents the model's predicted returns on the training data. The close tracking indicates a good fit on training data.

limited predictive power, especially with purely technical data [HBP23; Cha+24]. For the 10-day horizon, the Random Forest model shows a slightly positive R^2 (0.0036), suggesting a marginal predictive capability beyond the mean, but still very weak. All other models continue to have negative R^2 scores on the test set for the 10-day horizon, with TFT performing the worst.

The dramatic drop in performance from training to test sets across most models highlights the difficulty of generalizing learned patterns from historical IBM stock data to future, unseen data, especially with default hyperparameters. Overfitting appears to be a major challenge, a common issue in financial time series forecasting [MH25; HSK18].

Directional Performance Analysis

Directional performance metrics evaluate the models' ability to predict whether the stock price will increase or decrease. Results are in Table 5.4 and Table 5.5.

Training Set Performance: Similar to regression metrics, directional accuracy and average precision are very high on the training set (Table 5.4) for Random Forest, XGBoost, and N-BEATS (accuracies often >89%, average precision >0.96). This indicates these models learn to classify the direction of movement in the training data very well, as exemplified by the N-BEATS 1-day training confusion matrix (a) in Figure 5.3. LSTM, LightGBM, and especially TFT show weaker directional performance on the training set.

Test Set Performance: On the test set (Table 5.5), directional accuracy for all models drops to around 50–52% for both 1-day and 10-day horizons. This performance is close to random guessing for a binary outcome, consistent with the Efficient Market Hypothesis, which suggests prices follow a random walk [HSK18; Fam70]. Average precision scores also hover around 0.51-0.56. For instance, the N-BEATS 1-day test confusion matrix (b) in Figure 5.3 shows that for 1514 actual down movements, only 548 were correctly predicted. This is close to the chance level for a dataset with roughly balanced up/down movements. This poor directional performance

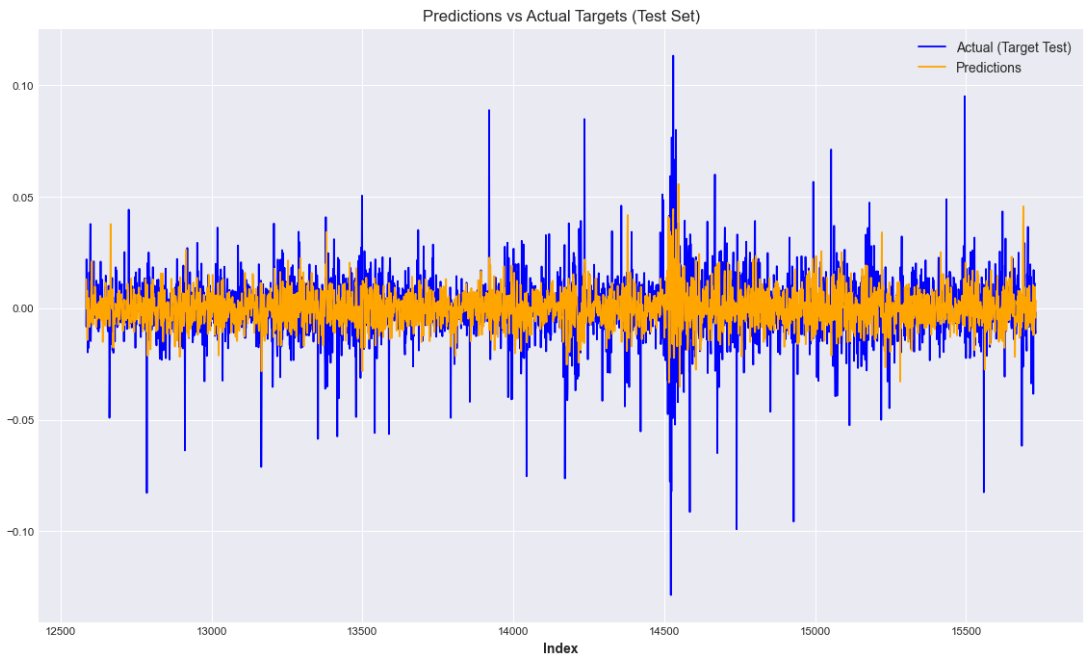


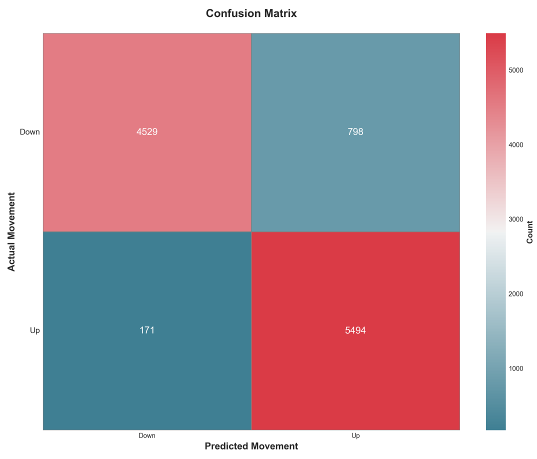
Figure 5.2: Predictions vs Actual Targets (Test Set) for N-BEATS (1-Day Horizon). The blue line represents the actual daily percentage returns (target test values), and the orange line represents the model's predicted returns on unseen test data. The divergence highlights overfitting and poor generalization.

Table 5.4: Directional Classification Metrics on the Training Set for 1-Day and 10-Day Prediction Horizons

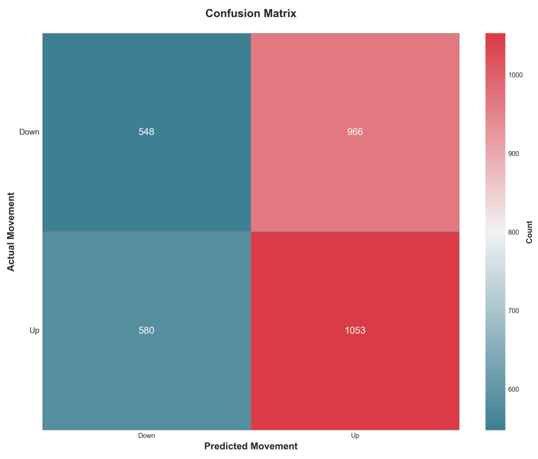
Model	**Precision**	**Dir. Acc (%)**
1-Day Prediction Horizon		
Random Forest	0.9923	95.54
XGBoost	0.9671	89.20
LightGBM	0.8911	78.96
RNN LSTM	0.6647	59.25
N-BEATS	0.9835	91.18
TFT	0.5735	54.43
10-Day Prediction Horizon		
Random Forest	0.9943	95.53
XGBoost	0.9707	89.46
LightGBM	0.8924	78.33
RNN LSTM	0.7081	61.20
N-BEATS	0.9861	89.60
TFT	0.6883	62.65

Table 5.5: Directional Classification Metrics on the Test Set for 1-Day and 10-Day Prediction Horizons

Model	**Precision**	**Dir. Acc (%)**
1-Day Prediction Horizon		
Random Forest	0.5299	51.22
XGBoost	0.5128	49.86
LightGBM	0.5101	48.52
RNN LSTM	0.5288	50.21
N-BEATS	0.5269	50.87
TFT	0.5358	50.53
10-Day Prediction Horizon		
Random Forest	0.5596	52.20
XGBoost	0.5317	48.79
LightGBM	0.5600	51.31
RNN LSTM	0.5419	51.37
N-BEATS	0.5484	51.40
TFT	0.5149	50.36



(a) Training Set



(b) Test Set

Figure 5.3: Confusion matrices for N-BEATS (1-Day Horizon). Rows represent actual movement (down/up), and columns represent predicted movement. **(a)** Training set: For down movements (5327 actual), 4529 were correctly predicted as down and 798 were incorrectly predicted as up. For up movements (5665 actual), 171 were incorrectly predicted as down, and 5494 were correctly predicted as up. **(b)** Test set: For down movements (1514 actual), 548 were correctly predicted as down' and 966 were incorrectly predicted as up. For up movements (1633 actual), 580 were incorrectly predicted as down and 1053 were correctly predicted as up.

on unseen data further underscores the overfitting issue and difficulty predicting stock market movements.

5.2.4 Strategy Performance Analysis

The performance of a simple trading strategy (long on predicted positive return, short on predicted negative return) is shown in Table 5.6. The total Buy & Hold return for the test period (July 2012 to December 2024) was 19.14% for the 1-day horizon and 63.40% for the 10-day horizon.

1-Day Horizon Strategy: Surprisingly, despite poor regression and directional metrics on the test set, N-BEATS, TFT, and Random Forest generated positive total returns (121.28%, 115.79%, and 104.14% respectively) for the 1-day strategy, all outperforming the Buy & Hold strategy. The performance of the Random Forest 1-day strategy is visualized in Figure 5.4. Their Sharpe Ratios (0.40, 0.38, 0.37) are modest but positive. XGBoost and LightGBM resulted in significant losses. This suggests that even if the overall R^2 is negative and directional accuracy is near chance, these models might occasionally capture large correct moves or have a slight, albeit statistically weak, edge that accumulates over many trades in this idealized backtest. However, the high maximum drawdowns (e.g., -48.34% for N-BEATS) indicate considerable risk.

10-Day Horizon Strategy: For the 10-day horizon, Random Forest leads with a total return of 105.72%, a notably improved Sharpe Ratio of 0.86, along with a lower maximum drawdown of -26.29%. This is visualized in Figure 5.5, where the strategy (blue line) significantly outperforms the Buy & Hold approach (grey line) over the test period. N-BEATS also shows a positive return (48.46%) with a Sharpe Ratio of 0.49, beating Buy & Hold. RNN LSTM and LightGBM are roughly breakeven or slightly positive in total return, while XGBoost and TFT incur losses. The improved Sharpe ratio for Random Forest on the 10-day horizon suggests it might be better at capturing slightly longer-term signals or that the 10-day aggregation smooths out some of the daily noise.

Table 5.6: Strategy Performance Metrics on the Test Set for 1-Day and 10-Day Prediction Horizons. (Buy & Hold Total Return for 1-Day Horizon: 19.14%; for 10-Day Horizon: 63.40%)

Model	**Total Ret (%)**	**Ann. Ret (%)**	**Sharpe Ratio**	**Max DD (%)**
1-Day Prediction Horizon				
Random Forest	104.14	5.88	0.37	-38.31
XGBoost	-74.60	-10.39	-0.36	-85.58
LightGBM	-79.74	-12.00	-0.44	-83.89
RNN LSTM	-19.10	-1.68	0.04	-60.59
N-BEATS	121.28	6.57	0.40	-48.34
TFT	115.79	6.44	0.38	-47.55
10-Day Prediction Horizon				
Random Forest	105.72	5.96	0.86	-26.29
XGBoost	-26.71	-2.46	-0.32	-45.60
LightGBM	-0.54	-0.04	0.04	-45.13
RNN LSTM	2.64	0.21	0.06	-30.77
N-BEATS	48.46	3.22	0.49	-20.48
TFT	-3.07	-0.27	0.00	-38.81

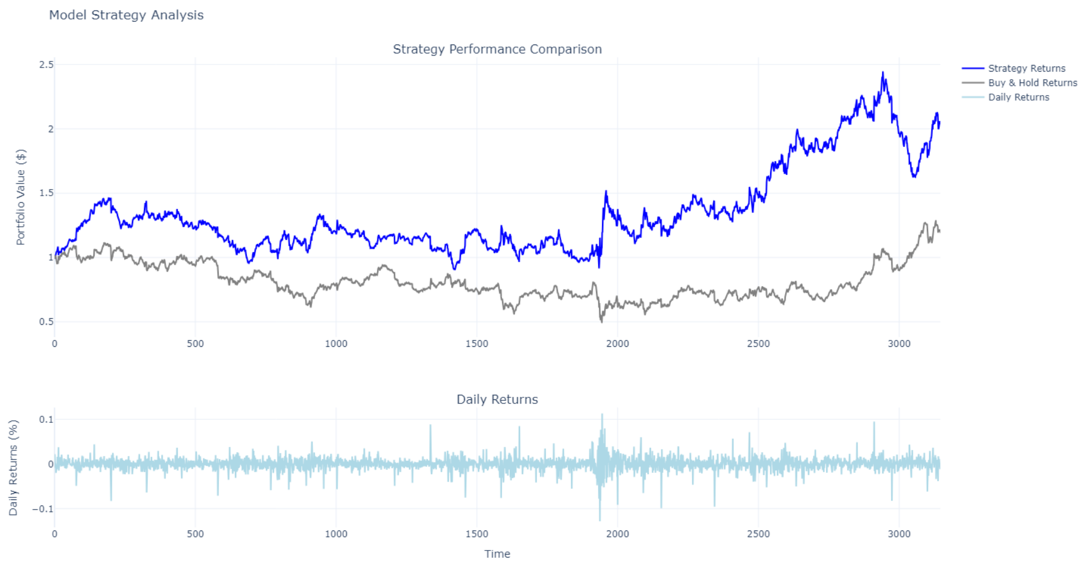


Figure 5.4: Strategy Performance Comparison for Random Forest (1-Day Horizon) on the Test Set. The blue line shows the cumulative portfolio value of the trading strategy based on Random Forest predictions. The grey line represents the cumulative portfolio value of a Buy & Hold strategy. The light blue area at the bottom shows the daily returns of the 1-day Buy & Hold strategy. The strategy significantly outperforms Buy & Hold despite weak statistical metrics.

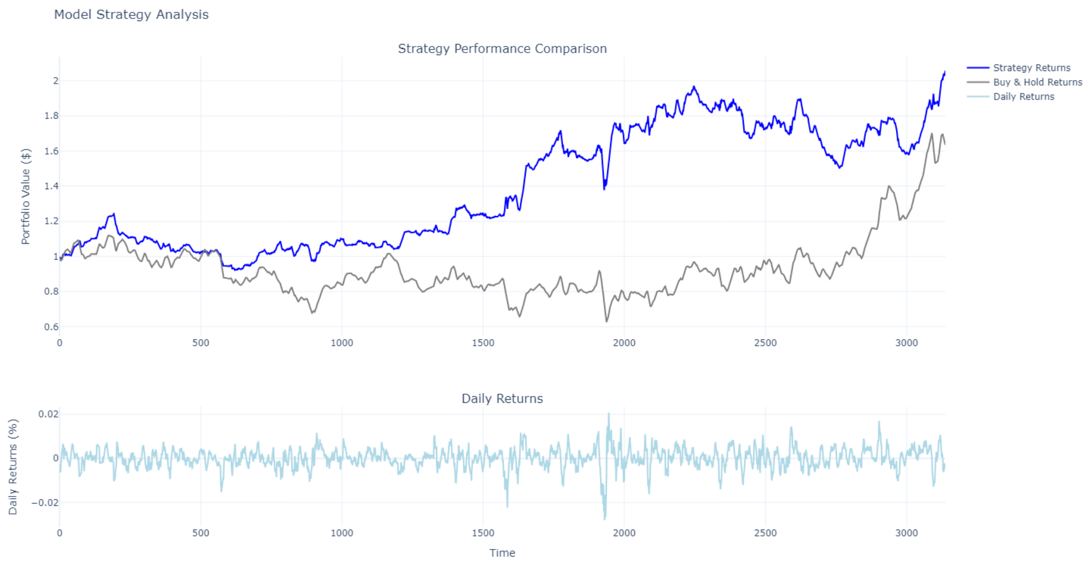


Figure 5.5: Strategy Performance Comparison for Random Forest (10-Day Horizon) on the Test Set. The blue line shows the cumulative portfolio value of the trading strategy based on Random Forest predictions. The grey line represents the cumulative portfolio value of a Buy & Hold strategy. The light blue area at the bottom shows the daily returns of the 10-day Buy & Hold strategy. The strategy significantly outperforms the Buy & Hold approach over the test period.

The strategy results, particularly for Random Forest, N-BEATS and TFT (1-day) and Random Forest (10-day), are somewhat counterintuitive given their poor statistical predictive accuracy on the test set. This discrepancy might be due to the non-linear nature of returns, where a few correctly predicted large moves can offset many small errors.

5.2.5 Overall Performance Summary and Discussion

The evaluation of the six machine learning models with primarily default hyperparameters reveals several key insights regarding their application to IBM stock price prediction:

1. **Overfitting is a Dominant Challenge:** Most models, particularly N-BEATS, Random Forest, and XGBoost, demonstrated strong performance (high R^2, high directional accuracy) on the training data but failed to generalize to the unseen test data. Test set R^2 scores were predominantly negative, and directional accuracies hovered around 50–52%. This highlights the significant difficulty in capturing truly predictive, generalizable patterns in stock market data with out-of-the-box models, a common finding in financial machine learning [HSK18; BGC22]. The models largely learned noise or specific patterns in the training data that did not persist.

2. **Computational Efficiency Varies Widely:** Gradient boosting models (LightGBM, XGBoost) are extremely fast to train, often completing in seconds, making them suitable for rapid experimentation. Deep learning models, particularly N-BEATS and TFT, are comparatively more computationally intensive. While they require substantial training time as shown in Table 5.1, these durations are generally well within practical limits for daily model retraining, aligning with the daily frequency of the financial data used, described in Chapter 3.

3. **Regression vs. Directional Accuracy on Test Data:** Poor regression performance, exemplified by metrics such as a negative R^2 on the test set, generally correlated with

weak directional accuracy, which typically ranged between 50–52%. For context, a naive benchmark strategy of consistently predicting positive returns within this test set would have achieved an accuracy of 52.26%. This outcome is anticipated, as accurately predicting both the magnitude and sign of returns is intrinsically more complex than forecasting the sign alone.

4. **Divergence Between Statistical Metrics and Strategy Performance:** A key observation was the divergence between conventional predictive accuracy metrics and the performance in an idealized trading strategy. Despite generally low R^2 scores and near-random directional accuracy on the test set, certain models, notably Random Forest (1-day and 10-day), N-BEATS (1-day and 10-day), and TFT (1-day), generated positive total returns that surpassed a Buy & Hold benchmark. This suggests that the standard metrics, while important, might not fully capture the nuances that contribute to profitability in this simplified trading context. However, these strategy outcomes must be viewed cautiously, given the idealized backtest and the lack of robust statistical underpinning on the test data. Achieving consistent alpha remains a significant challenge [Fam70; VV14].

5. **Impact of Prediction Horizon:**

   - For regression and directional accuracy on the test set, performance remained poor for both 1-day and 10-day horizons, with a marginal improvement in R^2 for Random Forest at 10 days.
   - For strategy performance, the 10-day horizon seemed to benefit Random Forest significantly (improved Sharpe ratio, lower drawdown) and N-BEATS to some extent. This might suggest that predicting slightly longer-term aggregated movements could be less noisy or that these models are better suited for such horizons, potentially capturing weaker, longer-term signals.

6. **Model Families:**

   - **Ensemble Tree-Based Models (RF, XGBoost, LightGBM):** Showed strong fitting capabilities on training data (especially RF and XGBoost). RF showed some promise in the 10-day strategy. However, all suffered from severe overfitting regarding statistical predictive accuracy on the test set.
   - **Deep Learning Models (LSTM, N-BEATS, TFT):** N-BEATS was the best at fitting training data. LSTM performance was modest. TFT struggled significantly even on training data with default settings for this specific task. N-BEATS and TFT showed noteworthy strategy performance on the 1-day horizon (N-BEATS also on 10-day). These models are generally more complex and might require more careful tuning, different feature engineering, or architectural adjustments to generalize well on this univariate forecasting task [LW24; Hu21].

In summary, the models struggled to achieve robust statistical predictive power on unseen IBM stock data using default hyperparameters, primarily due to overfitting. Test set regression and directional metrics were generally weak. However, an interesting divergence emerged where some models, despite these statistical shortcomings, yielded positive total returns in an idealized trading simulation, outperforming a Buy & Hold strategy. Specifically, the Random Forest and N-BEATS models demonstrated this noteworthy strategy performance. Even a marginal, consistent predictive edge in financial markets can translate to significant alpha or excess returns [Kha+23; HSK18]. While these preliminary strategy results are encouraging, they must be interpreted cautiously due to the idealized backtesting conditions (e.g., no transaction costs) and the underlying weak statistical validation. Nevertheless, the performance of Random Forest and N-BEATS, in particular, suggests they are promising candidates for further, more rigorous investigation. Future work should prioritize addressing overfitting through robust regularization, comprehensive hyperparameter optimization, potentially richer feature sets, and more realistic backtesting protocols to determine if a truly reliable predictive edge can be consistently extracted [Pat+20; Sir+19].

Chapter 6

Design of a Multi-Model Trust-Enhancing System

Building on the predictive modeling performance analysis in Chapter 5, this chapter details the architecture, components, and design rationale of a decision support system developed to address the challenges of user trust in stock market predictions. Many machine learning models' inherent complexity and black-box nature, as discussed in Section 2.4, can hinder user acceptance, especially in high-stakes financial decision-making. This system aims to mitigate these concerns by integrating predictions from multiple machine learning models with insights derived from historically similar market patterns, thereby fostering greater transparency and contextual understanding.

6.1 Introduction and Design Philosophy

The primary research question guiding this thesis is: *"How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?"*. The system described in this chapter directly attempts to answer this question by providing users with a more holistic and interpretable view of potential market movements.

The design philosophy is rooted in several key principles aimed at enhancing user trust:

- **Transparency through Comparison:** Instead of presenting a single, opaque prediction, the system showcases outputs from the six distinct machine learning models detailed in Section 5.1. This allows users to observe areas of consensus or divergence among the models.
- **Contextualization through Historical Analogy:** The system identifies and presents past market periods that exhibit strong similarity to current conditions, aligning with similarity-based explanation concepts discussed in Section 2.4 [Han+21; NSK22]. The window length for defining these patterns (20 days) deliberately matches the look-back period used by the machine learning models, ensuring that the historical analogies are based on the same temporal input scale as the models' predictions.
- **Evidence-based Insight into Past Model Behavior:** The system demonstrates how the integrated machine learning models would have performed when presented with those past input sequences for each similar historical period. This provides tangible evidence of their past accuracy under analogous feature conditions.
- **User Empowerment and Facilitated Interpretation:** The system provides information and tools for exploration. Visualizations like "Model Performance on Historical

Patterns" and a statistically derived "Recommendation" are included to aid non-expert users.

These elements are integrated into an interactive dashboard designed to allow users to explore predictions, examine historical similarities, and understand the basis for system recommendations.

6.2 System Architecture

The system is architected as a three-tier application, illustrated in Figure 6.1. The components are:

Frontend Web Application (React): The user-facing layer for data visualization, user interaction, and API communication.

Backend API Server (FastAPI): The core logic engine developed in Python, exposing API endpoints. It handles data preprocessing, calculation of the eight technical indicators defined in Section 4.3, scaling, PCA transformation, pattern matching using Dynamic Time Warping (DTW), and model predictions.

Data Storage & Machine Learning Models: Consists of the historical IBM stock data and the six pre-trained ML models for 1-day and 10-day horizons from Section 5.1.

User interactions on the React frontend trigger HTTP API requests to the FastAPI backend, which processes them and returns JSON data for display.

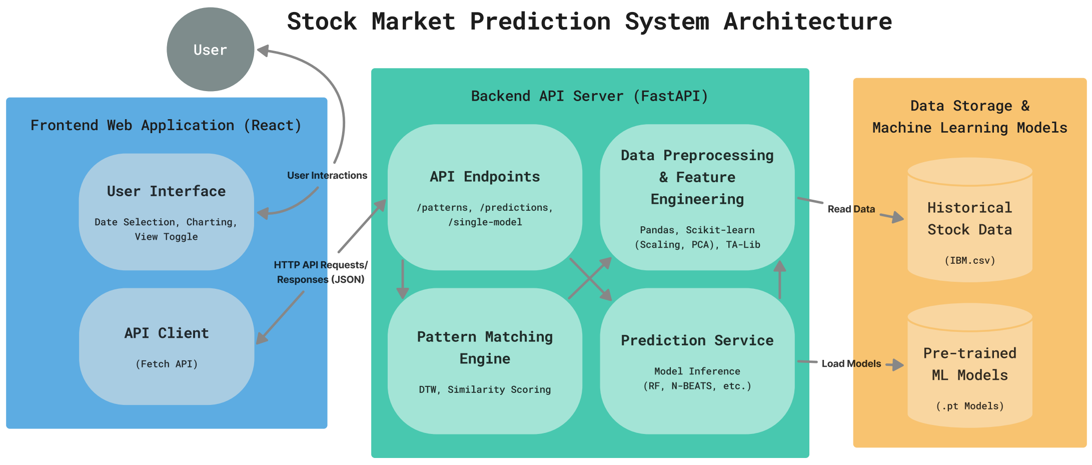

Stock Market Prediction System Architecture
User
Frontend Web Application (React)
User Interface
Date Selection, Charting, View Toggle
API Client
(Fetch API)
User Interactions
HTTP API Requests/ Responses (JSON)
Backend API Server (FastAPI)
API Endpoints
/patterns, /predictions, /single-model
Data Preprocessing & Feature Engineering
Pandas, Scikit-learn (Scaling, PCA), TA-Lib
Pattern Matching Engine
DTW, Similarity Scoring
Prediction Service
Model Inference (RF, N-BEATS, etc.)
Read Data
Load Models
Data Storage & Machine Learning Models
Historical Stock Data
(IBM.csv)
Pre-trained ML Models
(.pt Models)

Figure 6.1: High-level architecture of the Stock Market Prediction System. This diagram illustrates the three main tiers: the Frontend Web Application (React) handling user interactions and API calls; the Backend API Server (FastAPI) containing the core logic for data processing, pattern matching, and model predictions; and the Data Storage & Machine Learning Models layer providing historical stock data and pre-trained models. Arrows depict the flow of data and user interactions.

6.3 Core System Components – Design and Implementation Details

6.3.1 Multi-Model Prediction Integration

The system integrates all six machine learning models analyzed in Chapter 5: Random Forest, XGBoost, LightGBM, LSTM, N-BEATS, and Temporal Fusion Transformer. As detailed in

Section 4.4.3, these models input a 20-day sequence of the eight engineered technical features. For any given 20-day window, current or historical, each model generates a 1-day and 10-day percentage return forecast. Presenting these diverse outputs simultaneously, as discussed in Section 2.4, allows users to assess model agreement based on identical input feature sequences [Yeo+25; WL21].

6.3.2 Historical Similarity Search Engine

A core component for contextualization is the historical similarity search engine, inspired by Velasquez (2023) [Vel23]. It identifies past 20-day market windows most similar to a selected current window. This 20-day window length is essential, as it matches the look-back period used to train and feed input to the machine learning models, ensuring that the historical analogies are based on patterns of the same temporal scale and feature set that the models themselves process.

Defining Similarity: Features and Metric

Similarity is defined based on comparing 20-day windows of two types of data:

1. **Daily Percentage Returns:** The sequence of raw daily returns.
2. **PCA-Reduced Technical Indicators:** The eight engineered technical features (ADX, short/long close-EMA correlation, RSI, DV2, intraday range, Bollinger Bands Width, and volume momentum, as detailed in Section 4.3) serve as the primary basis for similarity. These are the same features used as input to the machine learning models. To make the similarity search more robust and less prone to noise from eight individual indicator series, they are first standardized, and then Principal Component Analysis (PCA) is applied to reduce them to three principal components. This PCA model is fit on the training portion of the data as per Section 4.4.1 and then used to transform the features for any given 20-day window. These three PCA components, which capture 58.5% of the variance of the original eight indicators, form the technical indicator feature set for similarity matching.

The distance D between the current window and each historical window is calculated using a weighted Dynamic Time Warping (DTW) distance. DTW is chosen for its ability to find similarities between time series that might be warped or shifted in time [Vel23]. The composite distance is:

$$D = \alpha \cdot D_{\text{returns}} + (1 - \alpha) \cdot D_{\text{indicators}} \tag{6.1}$$

Where D_{returns} is the DTW distance between the normalized daily percentage returns of the two windows, and $D_{\text{indicators}}$ is the DTW distance (using Euclidean distance for point-wise comparison within DTW) between the 3-dimensional PCA-reduced technical indicator series of the two windows. The weight α is set to 0.2. This assigns 80% importance to the similarity of the PCA-reduced technical indicator patterns and 20% to the similarity of the raw return patterns. This specific weighting acknowledges that while price movement patterns are important, the richer, multifaceted information captured by combining technical indicators should be more heavily emphasized when defining similarity in market states. Since four of the eight original technical indicators are mathematical transformations of the close price and thus directly related to returns calculated over the 20-day window, a 20% weight is still assigned to the direct returns component. The primary reliance on the PCA-reduced indicators for similarity ensures that the identified past patterns are analogous in terms of the features the ML models consider, enhancing the relevance of these historical comparisons for understanding potential model behavior. The resulting DTW distance is then converted to a normalized similarity score, where higher values indicate greater similarity.

Search Process and Presentation

The backend identifies the top three most similar historical 20-day patterns, ensuring a minimum temporal separation of 5 days between them to prevent overlap. For each of these patterns, the system retrieves:

- The 20-day sequence of their historical returns and PCA-reduced features.
- The actual subsequent 10-day returns that followed that pattern.
- The 1-day and 10-day predictions each of the six ML models would have generated if provided with that specific historical 20-day window of features as input.
- The absolute error of each model's prediction against the actual outcome for that past instance.

The "IBM Stock Historical Pattern Similarity" chart (Figure 6.2) visualizes the current 20-day window (based on relative price movement), the three most similar past patterns (also shown as relative price movement, including their actual continuations over the next 10 days), and a median pattern projection. This median projection, derived from the outcomes of these historically similar input sequences, offers a form of transparent, example-based forecast.

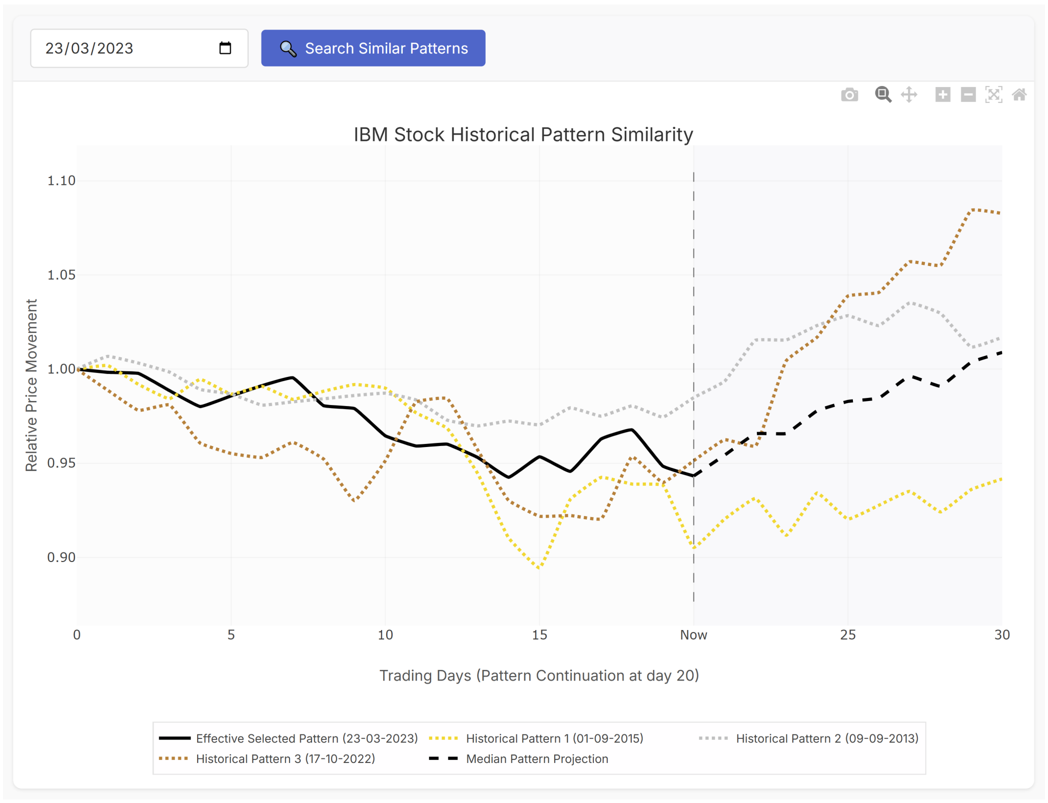


Figure 6.2: The Historical Pattern Similarity chart presented in the Multiple Models View of the user interface. The solid black line ("Effective Selected Pattern") represents the relative price movement of IBM stock over the current 20-day window selected by the user (ending 23 March 2023 in this example). Three dotted lines (gold, silver, bronze) depict the three most historically similar 20-day patterns, showing their actual relative price movements and continuations over the subsequent 10 trading days. The dashed black line ("Median Pattern Projection") illustrates the median price trajectory derived from these three historical continuations. The vertical dashed line labeled "Now" marks the end of the 20-day pattern window and the beginning of the 10-day projection/actual outcome period. The x-axis represents trading days, and the y-axis shows relative price movement normalized to the start of each pattern.

6.3.3 Aggregation and Analysis for User Interpretation

To simplify interpretation for non-expert users, the system provides aggregated insights:

- **Model Performance on Historical Patterns:** For each of the six models, a weighted average absolute error for both 1-day and 10-day predictions is calculated across the three most similar historical patterns. The weights are the normalized similarity scores of these patterns, emphasizing performance in more analogous past situations. The formula is:

$$\text{Avg Error} = \sum_{i=1}^{3} (w_i \times \text{error}_i) \tag{6.2}$$

  Where w_i is the normalized weight of the i-th pattern and error_i is the model's absolute prediction error on that pattern. This is visualized in Figure 6.3, highlighting the best model (lowest error) in green for each forecast horizon.

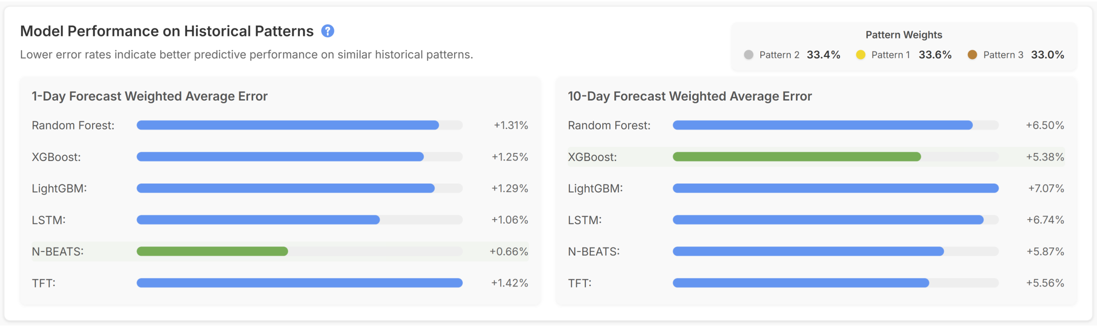
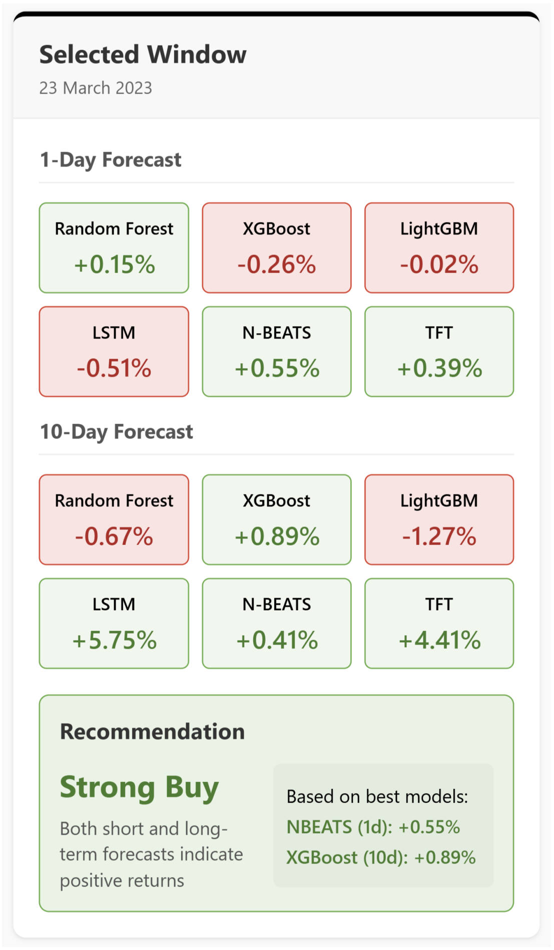


Figure 6.3: The "Model Performance on Historical Patterns" summary component from the Multiple Models View. This visualization presents the weighted average absolute error for each of the six machine learning models, calculated based on their predictive performance across the three most similar historical patterns. Separate bar charts display these errors for 1-Day and 10-Day Forecasts. The length of each blue bar is proportional to the error, with lower error values indicating better historical performance. The model with the lowest error in each category is highlighted with a green bar. The "Pattern Weights" used to calculate these weighted average errors are also displayed, reflecting the normalized similarity scores of the historical patterns.

- **Statistical Recommendation:** The best performing models, those with the lowest weighted average error on historical patterns, for the 1-day and 10-day horizons are used to generate a qualitative recommendation for the current window: "Strong Buy", "Weak Buy", "Hold", or "Sell". The logic is:
  - **Strong Buy:** The Best 1-day model predicts positive, and the best 10-day model predicts positive.
  - **Weak Buy:** The Best 1-day model predicts positive, but the best 10-day model predicts non-positive.
  - **Hold:** The Best 1-day model predicts non-positive, but the best 10-day model predicts positive.
  - **Sell:** The Best 1-day model predicts non-positive, and the best 10-day model predicts non-positive.

  This approach of using buy/hold/sell signals is common in technical analysis, as noted in Section 2.3, and extending to four states, offers an actionable suggestion grounded in the historical reliability of the models in similar contexts.

The "Selected Window" card (Figure 6.4) presents the current predictions from all six models alongside this statistical recommendation. The "Historical Similar Patterns" cards (Figure 6.5) provide a compelling drill-down, showing each model's specific prediction and error for each past similar instance, facilitating inter-model comparisons and offering transparency into their past behaviors.



Selected Window
23 March 2023

1-Day Forecast

Random Forest	XGBoost	LightGBM
+0.15%	-0.26%	-0.02%
LSTM	**N-BEATS**	**TFT**
-0.51%	+0.55%	+0.39%

10-Day Forecast

Random Forest	XGBoost	LightGBM
-0.67%	+0.89%	-1.27%
LSTM	**N-BEATS**	**TFT**
+5.75%	+0.41%	+4.41%

Recommendation

Strong Buy

Both short and long-term forecasts indicate positive returns

Based on best models:
NBEATS (1d): +0.55%
XGBoost (10d): +0.89%

Figure 6.4: The "Selected Window" predictions card from the Multiple Models View, corresponding to the date 23 March 2023, representing today. It displays the 1-day and 10-day percentage return forecasts from each of the six integrated machine learning models. At the bottom, the card presents an overall "Recommendation", which is explicitly stated as being "Based on best models" historically, along with the specific forecasts from those best-performing models that led to the recommendation.

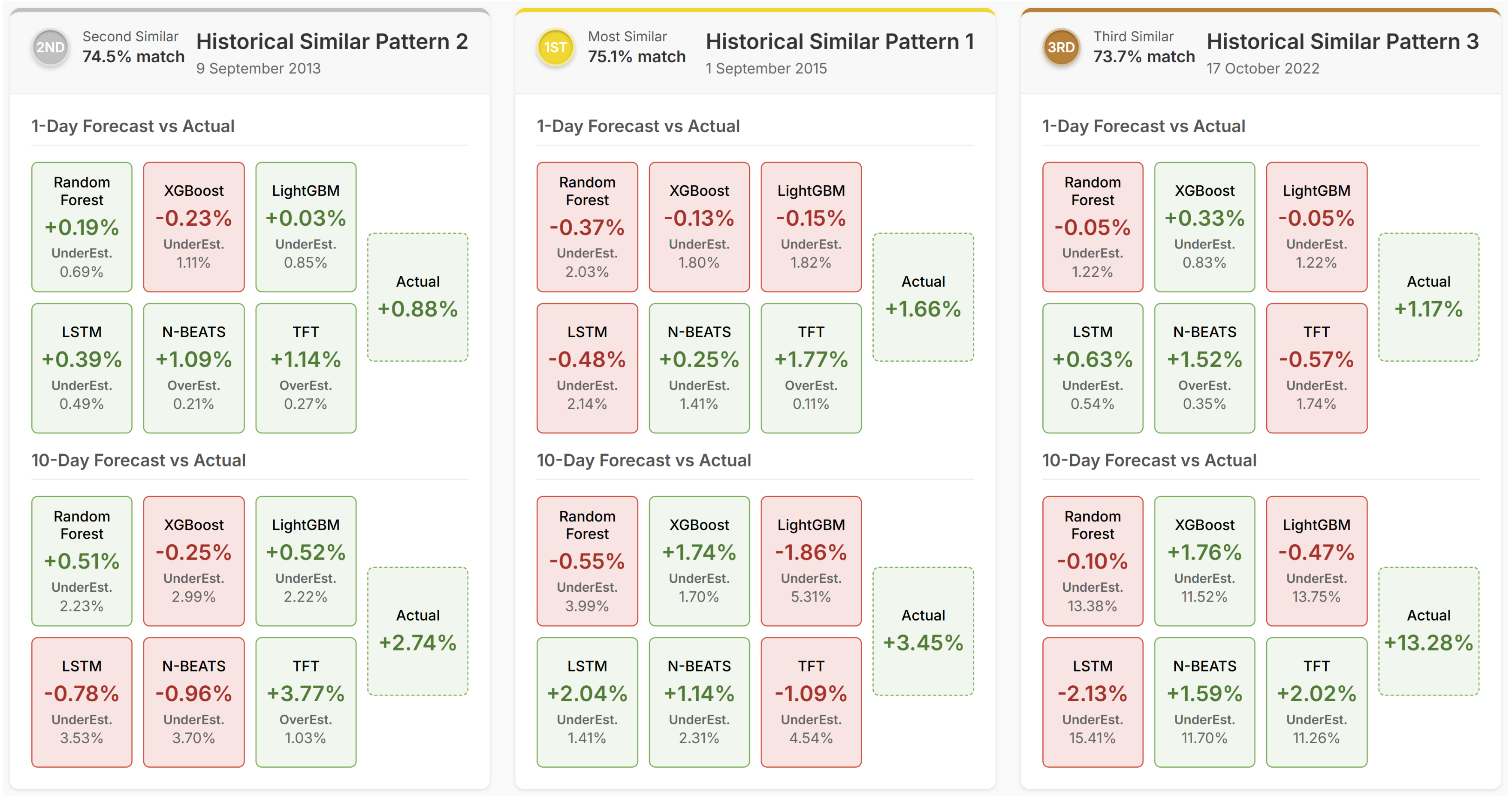
6.4 User Interface (UI) and Dashboard Design

The React frontend offers a "Multiple Models View" and a "Single Model View".

6.4.1 Multiple Models View

This core view integrates components designed to enhance trust:

- **IBM Stock Historical Pattern Similarity Chart (Figure 6.2):** Displays the current 20-day window, the three most similar historical patterns with their actual subsequent 10-day price movements, and a median 10-day projection derived from these patterns.
- **Selected Window Predictions Card (Figure 6.4):** Presents 1-day and 10-day forecasts from all six models for the current date, plus the system's overall recommendation.



2ND Second Similar 74.5% match

Historical Similar Pattern 2

9 September 2013

1-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM
+0.19% UnderEst. 0.69%	-0.23% UnderEst. 1.11%	+0.03% UnderEst. 0.85%

LSTM	N-BEATS	TFT
+0.39% UnderEst. 0.49%	+1.09% OverEst. 0.21%	+1.14% OverEst. 0.27%

Actual +0.88%

10-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM
+0.51% UnderEst. 2.23%	-0.25% UnderEst. 2.99%	+0.52% UnderEst. 2.22%

LSTM	N-BEATS	TFT
-0.78% UnderEst. 3.53%	-0.96% UnderEst. 3.70%	+3.77% OverEst. 1.03%

Actual +2.74%

1ST Most Similar 75.1% match

Historical Similar Pattern 1

1 September 2015

1-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM
-0.37% UnderEst. 2.03%	-0.13% UnderEst. 1.80%	-0.15% UnderEst. 1.82%

LSTM	N-BEATS	TFT
-0.48% UnderEst. 2.14%	+0.25% UnderEst. 1.41%	+1.77% OverEst. 0.11%

Actual +1.66%

10-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM
-0.55% UnderEst. 3.99%	+1.74% UnderEst. 1.70%	-1.86% UnderEst. 5.31%

LSTM	N-BEATS	TFT
+2.04% UnderEst. 1.41%	+1.14% UnderEst. 2.31%	-1.09% UnderEst. 4.54%

Actual +3.45%

3RD Third Similar 73.7% match

Historical Similar Pattern 3

17 October 2022

1-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM
-0.05% UnderEst. 1.22%	+0.33% UnderEst. 0.83%	-0.05% UnderEst. 1.22%

LSTM	N-BEATS	TFT
+0.63% UnderEst. 0.54%	+1.52% OverEst. 0.35%	-0.57% UnderEst. 1.74%

Actual +1.17%

10-Day Forecast vs Actual

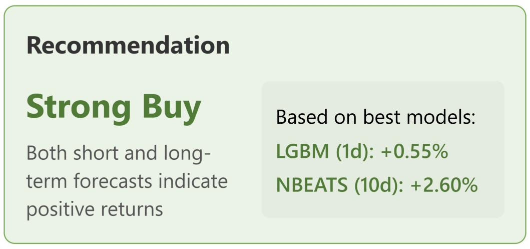
Random Forest	XGBoost	LightGBM
-0.10% UnderEst. 13.38%	+1.76% UnderEst. 11.52%	-0.47% UnderEst. 13.75%

LSTM	N-BEATS	TFT
-2.13% UnderEst. 15.41%	+1.59% UnderEst. 11.70%	+2.02% UnderEst. 11.26%

Actual +13.28%

Figure 6.5: Detailed analysis cards for the three "Historical Similar Patterns" identified by the system, presented in a tri-level podium arrangement (gold for 1st/Most Similar, silver for 2nd, bronze for 3rd) in the Multiple Models View. Each card corresponds to one of the patterns also visualized in Figure 6.2. The card displays its similarity match percentage and end date for each historical pattern. It then shows the 1-day and 10-day percentage return predictions that each of the six machine learning models would have generated for that past period, alongside the actual percentage return that occurred. The absolute prediction error for each model is also provided, along with an indication of whether the model underestimated or overestimated the actual movement, facilitating direct comparison of model performance in analogous past scenarios.

Examples of the four recommendation states are shown in Figure 6.6.



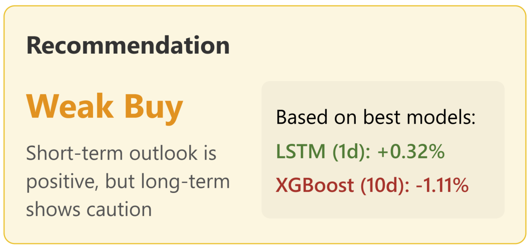
Recommendation

Strong Buy

Both short and long-term forecasts indicate positive returns

Based on best models:
LGBM (1d): +0.55%
NBEATS (10d): +2.60%

(a) Strong Buy



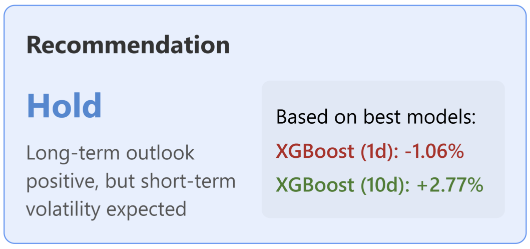
Recommendation

Weak Buy

Short-term outlook is positive, but long-term shows caution

Based on best models:
LSTM (1d): +0.32%
XGBoost (10d): -1.11%

(b) Weak Buy



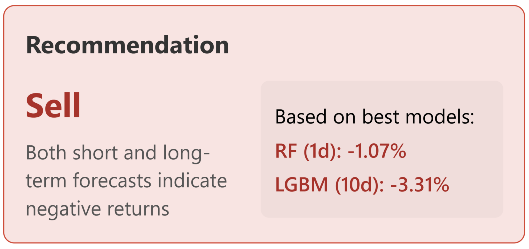
Recommendation

Hold

Long-term outlook positive, but short-term volatility expected

Based on best models:
XGBoost (1d): -1.06%
XGBoost (10d): +2.77%

(c) Hold



Recommendation

Sell

Both short and long-term forecasts indicate negative returns

Based on best models:
RF (1d): -1.07%
LGBM (10d): -3.31%

(d) Sell

Figure 6.6: Examples of the four distinct qualitative recommendation states provided by the system in both Views. These states are: (a) "Strong Buy", indicating positive forecasts from the best-performing historical models for both 1-day and 10-day horizons; (b) "Weak Buy", where the best 1-day model is positive but the best 10-day model is non-positive; (c) "Hold", where the best 1-day model is non-positive but the best 10-day model is positive; and (d) "Sell", indicating non-positive forecasts from the best historical models for both horizons. Each recommendation card explicitly shows the forecasts from the historically best-performing 1-day and 10-day models that determined the specific recommendation.

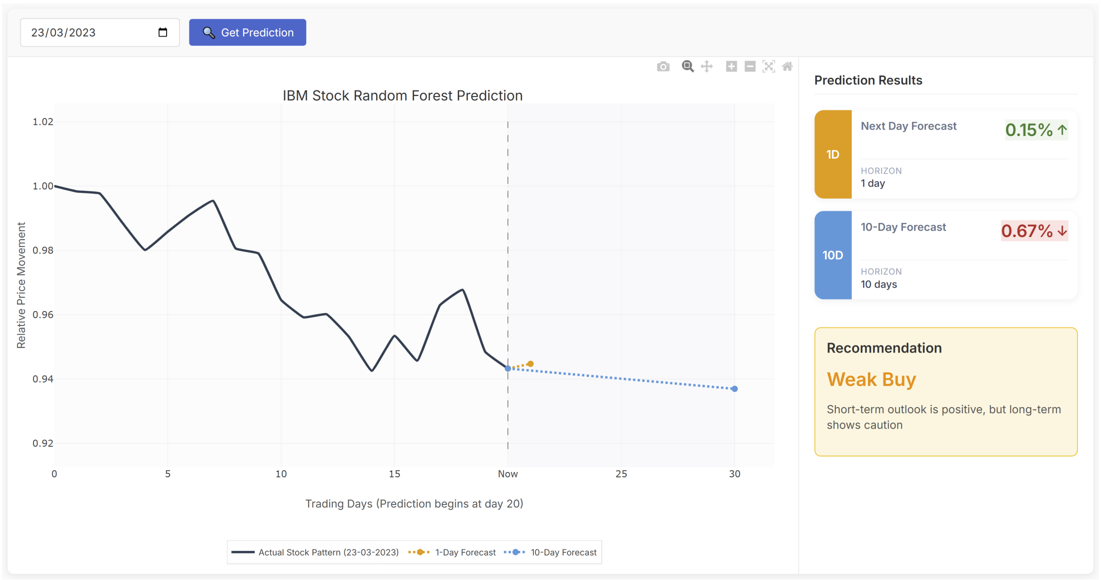
- **Historical Similar Patterns Analysis Cards (Figure 6.5):** For each of the three similar historical patterns, a card details its similarity score, date, and essentially, the 1-day/10-day predictions from all six models versus the actual outcome, alongside their prediction errors (over/underestimation and magnitude). This offers direct evidence of past model reliability in comparable situations.

- **Model Performance on Historical Patterns Summary (Figure 6.3):** Visualizes the weighted average absolute error for each model (1-day and 10-day) based on performance in the three similar historical patterns. It highlights the best models (lowest error) and displays the pattern weights used in the calculation, providing a quantitative summary of historical model reliability.

Interactive features such as date selection and hover effects linking cards to the chart facilitate exploration. Info icons offer on-demand explanations for calculations.

6.4.2 Single Model View

For the user study (Chapter 7), a "Single Model View" (Figure 6.7) is provided. It uses only the Random Forest model for predictions and recommendations, omitting multi-model comparisons and historical similarity analysis, to serve as a baseline for evaluating the trust impact of the Multiple Models View. The Random Forest model was selected as the best model from the analysis in Section 5.2.



Prediction Results

1D — Next Day Forecast 0.15% ↑ — HORIZON 1 day

10D — 10-Day Forecast 0.67% ↓ — HORIZON 10 days

Recommendation

Weak Buy

Short-term outlook is positive, but long-term shows caution

Figure 6.7: Overview of the Single Model View interface. This view presents a simplified prediction dashboard based solely on the Random Forest model. The main chart displays the "Actual Stock Pattern" for the selected 20-day window (ending 23 March 2023 in this example), with the 1-day and 10-day Random Forest forecasts shown as dotted lines. To the right, "Prediction Results" cards show the numerical percentage change for the "Next Day Forecast" and the "10-Day Forecast". Below these, a "Recommendation" is provided, based exclusively on the Random Forest model's current predictions for the 1-day and 10-day horizons.

6.5 User Workflow and Transition to Evaluation

The system is designed with two primary interfaces to facilitate user interaction and comparative analysis: the Multiple Models View and the Single Model View.

In the **Multiple Models View**, users can select a date, observe the current market pattern against historically similar ones and their outcomes (Figure 6.2), review current predictions from all six models and the system's evidence-based recommendation (Figure 6.4), delve into how each model performed in those specific past similar instances (Figure 6.5), and understand the aggregated historical reliability that underpins the recommendation (Figure 6.3). This multifaceted approach aims to allow users to build a more informed sense of trust by providing transparent insights into model behavior and historical performance within analogous contexts, leveraging a similarity search.

The **Single Model View** (Figure 6.7) intentionally presents a streamlined workflow, offering predictions and recommendations based solely on the Random Forest model's output, without the multi-model comparisons or historical similarity analysis.

The distinct design of these two views, the comprehensive and evidence-rich Multiple Models View versus the simpler Single Model View, is central to the empirical investigation of this thesis. The following chapter, Chapter 7, will detail the methodology and findings of a user study to evaluate the impact of these different system presentations on user trust, confidence, and hypothetical investment decisions. This evaluation will directly assess whether integrating multiple models and similarity-based historical insights, as designed in the Multiple Models View, effectively enhances trustworthiness in stock market predictions compared to a more conventional single-model approach.

Chapter 7

User Study on Evaluating System Trustworthiness

This chapter details a user study conducted to empirically investigate this thesis's primary research question: *"How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?"*. The study aimed to assess the impact of a multi-model decision support system, augmented with historical similarity insights, on user trust, confidence, and perceived system utility compared to a more conventional single-model system. By systematically evaluating user responses to two distinct interface designs; the Single Model View (SMV) and the Multiple Models View (MMV) as described in Chapter 6, this research seeks to provide evidence-based insights into designing more trustworthy AI-driven financial tools.

7.1 Methodology

A within-subjects experimental design was employed, where participants interacted with the SMV and MMV interfaces. The order of presentation was counterbalanced to mitigate potential learning or fatigue effects.

7.1.1 Participants

A total of 22 participants completed the user study. The demographic breakdown was as follows:

- **Age Range:** The majority were 18-24 years old (13 participants, 59.1%), followed by 25-34 (7 participants, 31.8%). Smaller representations were from the age groups of 35-44 (1 participant, 4.5%) and 55-64 (1 participant, 4.5%).
- **Gender:** The participant pool was predominantly male (20 participants, 90.9%), with 2 female participants (9.1%).
- **Stock Market Investing Experience:** Half of the participants (11, 50.0%) identified as beginners with basic knowledge or have made a few investments. A significant portion (9 participants, 40.9%) had no prior investing experience. One participant (4.5%) had advanced expertise, and one (4.5%) had intermediate experience.
- **Data Visualization Experience:** Participants generally had some experience with data visualization tools. Nine (40.9%) reported basic experience (viewed occasionally), another nine (40.9%) reported intermediate experience (use regularly), one (4.5%) had advanced expertise, and one (4.5%) had no experience.

- **Machine Learning (ML) Familiarity:** A substantial number of participants had exposure to ML concepts, with 10 (45.5%) reporting advanced familiarity (studied/worked with ML) and 9 (40.9%) having some practical exposure. Two participants (9.1%) had no understanding of ML, and one (4.5%) had a basic understanding.

This diverse group, particularly in terms of stock market and ML experience, provided a range of perspectives on the system interfaces.

7.1.2 System Interfaces

Participants interacted with two distinct web-based dashboard views designed for this study, as detailed in Chapter 6:

- **Single Model View (SMV):** This interface presented stock predictions (1-day and 10-day IBM stock returns) based solely on the Random Forest model. It displayed the current 20-day stock pattern, the model's forecasts, and a system-generated investment recommendation (Strong Buy, Weak Buy, Hold, Sell) derived from these forecasts. This view is depicted in Figure 7.1.
- **Multiple Models View (MMV):** This interface provided predictions from all six machine learning models (Random Forest, XGBoost, LightGBM, LSTM, N-BEATS, TFT). Crucially, it integrated a historical similarity search engine, displaying the current 20-day pattern alongside the three most similar historical patterns and their actual subsequent outcomes. Each historical pattern showed how all six models would have performed. The MMV also provided a system-generated investment recommendation based on the performance of the historically best-performing models in those similar past scenarios. This view is depicted in Figure 7.2.

7.1.3 Experimental Design

A within-subjects experimental design was employed, meaning each participant experienced and evaluated both the Single Model View (SMV) and the Multiple Models View (MMV). This approach directly compared the two interface views (effectively, View A: SMV versus View B: MMV) based on each participant's responses to both. To control for potential order effects, where experiencing one view first might influence perceptions or performance on the second, the presentation sequence of the SMV and MMV was counterbalanced across participants:

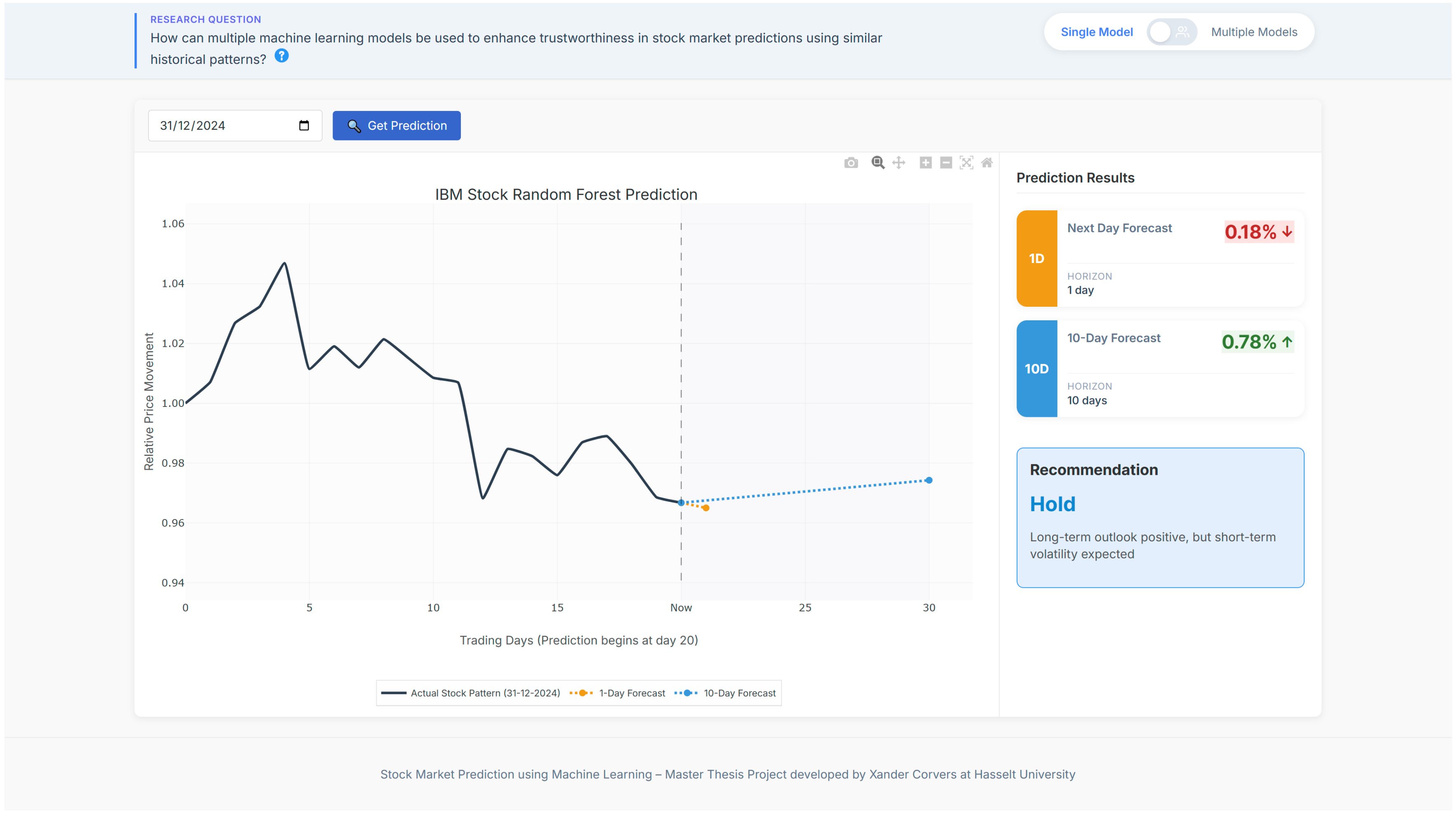
- Group 1 (11 participants) first interacted with the MMV (View B), followed by the SMV (View A).
- Group 2 (11 participants) first interacted with the SMV (View A), followed by the MMV (View B).

This counterbalancing strategy within the within-subjects design helps ensure that any observed differences in the comparison between SMV and MMV are more likely attributable to the interfaces' characteristics rather than the specific order in which the participants encountered them.

7.1.4 Procedure

The study was conducted in a controlled on-location setting, with the researcher present as an observer and facilitator. Participants completed the study using a structured online form on a dedicated computer. The procedure for each participant was as follows:

1. **Introduction and Consent:** Participants first received a verbal briefing from the facilitator detailing the study's purpose and procedure. They were then given a physical informed consent form (detailing data processing, rights, and confidentiality), which they read and signed. Following this, they accessed the online platform, which began with a digital reiteration of the study's introduction.



RESEARCH QUESTION

How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?

Single Model | Multiple Models

31/12/2024

Get Prediction

IBM Stock Random Forest Prediction
Relative Price Movement
1.06
1.04
1.02
1.00
0.98
0.96
0.94
0
5
10
15
Now
25
30
Trading Days (Prediction begins at day 20)
Actual Stock Pattern (31-12-2024)
1-Day Forecast
10-Day Forecast

Prediction Results

1D — Next Day Forecast — 0.18% ↓
HORIZON
1 day

10D — 10-Day Forecast — 0.78% ↑
HORIZON
10 days

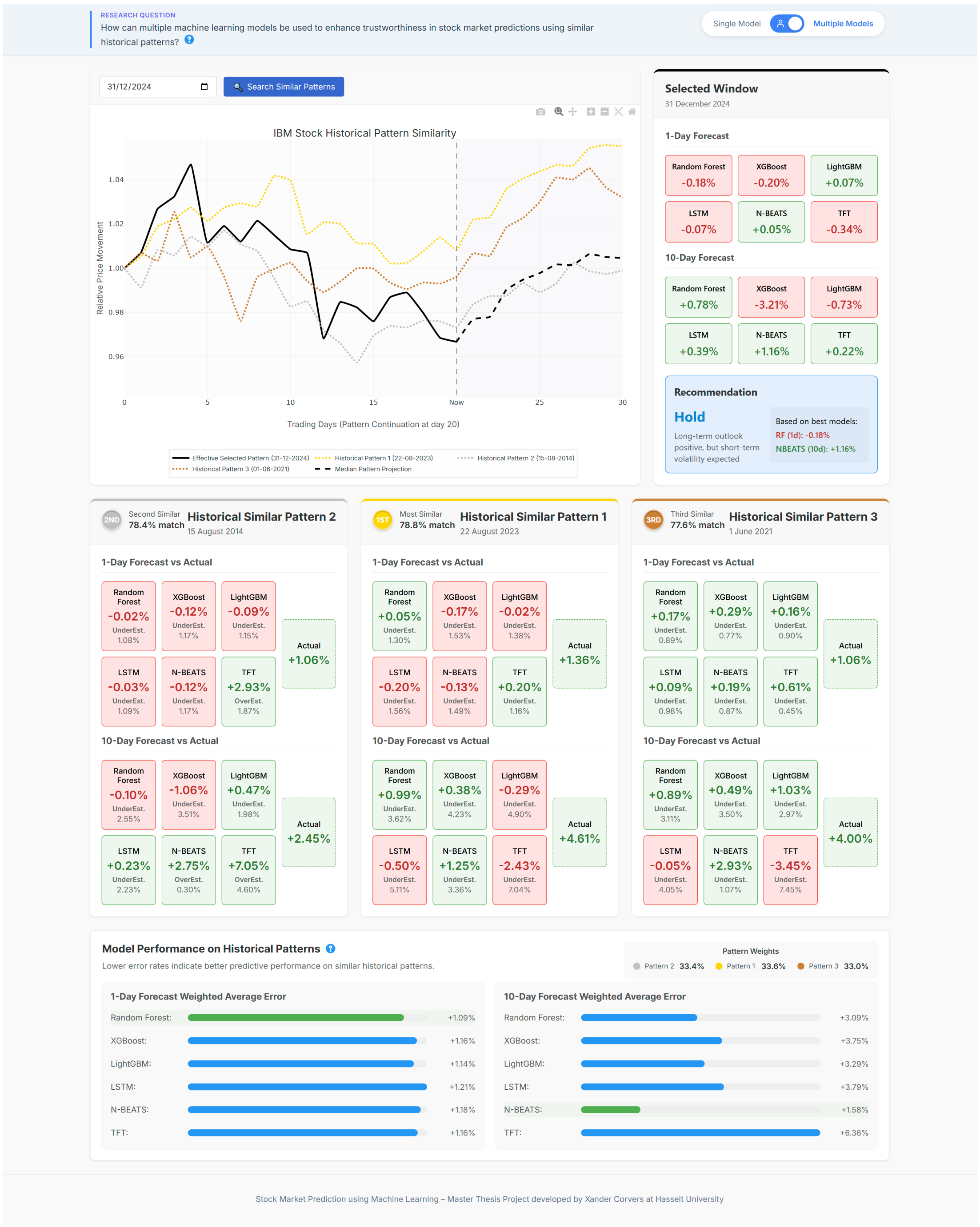
Recommendation

Hold

Long-term outlook positive, but short-term volatility expected

Stock Market Prediction using Machine Learning – Master Thesis Project developed by Xander Corvers at Hasselt University

Figure 7.1: The Single Model View (SMV) dashboard, illustrated for IBM stock prediction as of 31/12/2024. This interface presents forecasts derived solely from the Random Forest model. The main chart displays the recent 20-day stock price pattern alongside the model's 1-day and 10-day projections. The right panel provides numerical forecast values and a system-generated investment recommendation based exclusively on this single model's output, serving as a baseline system for comparison.


Single Model
Multiple Models
31/12/2024
Search Similar Patterns
IBM Stock Historical Pattern Similarity
Relative Price Movement
1.04
1.02
1.00
0.98
0.96
0
5
10
15
Now
25
30
Trading Days (Pattern Continuation at day 20)
Effective Selected Pattern (31-12-2024)
Historical Pattern 1 (22-08-2023)
Historical Pattern 2 (15-08-2014)
Historical Pattern 3 (01-06-2021)
Median Pattern Projection

RESEARCH QUESTION

How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?

Selected Window
31 December 2024

1-Day Forecast

Random Forest	XGBoost	LightGBM
-0.18%	-0.20%	+0.07%
LSTM	**N-BEATS**	**TFT**
-0.07%	+0.05%	-0.34%

10-Day Forecast

Random Forest	XGBoost	LightGBM
+0.78%	-3.21%	-0.73%
LSTM	**N-BEATS**	**TFT**
+0.39%	+1.16%	+0.22%

Recommendation

Hold

Long-term outlook positive, but short-term volatility expected

Based on best models:
RF (1d): -0.18%
NBEATS (10d): +1.16%

2ND Second Similar 78.4% match — **Historical Similar Pattern 2** 15 August 2014

1-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM	
-0.02% UnderEst. 1.08%	-0.12% UnderEst. 1.17%	-0.09% UnderEst. 1.15%	Actual +1.06%
LSTM	**N-BEATS**	**TFT**	
-0.03% UnderEst. 1.09%	-0.12% UnderEst. 1.17%	+2.93% OverEst. 1.87%	

10-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM	
-0.10% UnderEst. 2.55%	-1.06% UnderEst. 3.51%	+0.47% UnderEst. 1.98%	Actual +2.45%
LSTM	**N-BEATS**	**TFT**	
+0.23% UnderEst. 2.23%	+2.75% OverEst. 0.30%	+7.05% OverEst. 4.60%	

1ST Most Similar 78.8% match — **Historical Similar Pattern 1** 22 August 2023

1-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM	
+0.05% UnderEst. 1.30%	-0.17% UnderEst. 1.53%	-0.02% UnderEst. 1.38%	Actual +1.36%
LSTM	**N-BEATS**	**TFT**	
-0.20% UnderEst. 1.56%	-0.13% UnderEst. 1.49%	+0.20% UnderEst. 1.16%	

10-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM	
+0.99% UnderEst. 3.62%	+0.38% UnderEst. 4.23%	-0.29% UnderEst. 4.90%	Actual +4.61%
LSTM	**N-BEATS**	**TFT**	
-0.50% UnderEst. 5.11%	+1.25% UnderEst. 3.36%	-2.43% UnderEst. 7.04%	

3RD Third Similar 77.6% match — **Historical Similar Pattern 3** 1 June 2021

1-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM	
+0.17% UnderEst. 0.89%	+0.29% UnderEst. 0.77%	+0.16% UnderEst. 0.90%	Actual +1.06%
LSTM	**N-BEATS**	**TFT**	
+0.09% UnderEst. 0.98%	+0.19% UnderEst. 0.87%	+0.61% UnderEst. 0.45%	

10-Day Forecast vs Actual

Random Forest	XGBoost	LightGBM	
+0.89% UnderEst. 3.11%	+0.49% UnderEst. 3.50%	+1.03% UnderEst. 2.97%	Actual +4.00%
LSTM	**N-BEATS**	**TFT**	
-0.05% UnderEst. 4.05%	+2.93% UnderEst. 1.07%	-3.45% UnderEst. 7.45%	

Model Performance on Historical Patterns

Lower error rates indicate better predictive performance on similar historical patterns.

Pattern Weights
● Pattern 2 33.4% ● Pattern 1 33.6% ● Pattern 3 33.0%

1-Day Forecast Weighted Average Error		10-Day Forecast Weighted Average Error	
Random Forest:	+1.09%	Random Forest:	+3.09%
XGBoost:	+1.16%	XGBoost:	+3.75%
LightGBM:	+1.14%	LightGBM:	+3.29%
LSTM:	+1.21%	LSTM:	+3.79%
N-BEATS:	+1.18%	N-BEATS:	+1.58%
TFT:	+1.16%	TFT:	+6.36%

Stock Market Prediction using Machine Learning – Master Thesis Project developed by Xander Corvers at Hasselt University

Figure 7.2: The Multiple Models View (MMV) dashboard for IBM stock prediction as of 31/12/2024, designed to enhance user trust through multi-faceted analysis and historical context. Key components include: (i) the "IBM Stock Historical Pattern Similarity" chart, comparing the current 20-day pattern to the three most similar historical analogs and their actual subsequent outcomes; (ii) current 1-day and 10-day forecasts from six distinct machine learning models, alongside a system recommendation derived from historically best-performing models; (iii) detailed performance cards for each model on each identified historical pattern, showing predictions versus actuals and errors; and (iv) a "Model Performance on Historical Patterns" summary visualizing weighted average errors. This view offers a rich, evidence-based context for interpreting stock predictions.

2. **Demographic Questionnaire:** Participants answered questions on the online form about their age, gender, and experience with stock investing, data visualization, and machine learning.

3. **First Interface Evaluation (SMV or MMV):**

   - Participants were guided to open the computer's assigned interface (SMV or MMV). The facilitator encouraged them to spend a few minutes familiarizing themselves with its components and asked if they had any initial questions about navigation.

   - For five pre-defined historical dates (scenarios), participants selected the date in the dashboard, analyzed the presented information, made a hypothetical 10-day investment decision (Strong Buy, Weak Buy, Hold, Sell) via the online form, and rated their confidence in that decision on a 7-point Likert scale (1 = Not at all confident, 7 = Extremely confident) within the form. The scenarios were chosen to represent varied market conditions and system recommendations. The facilitator was available to clarify task instructions if needed but did not influence decision-making.

   - After completing the five scenarios for the first interface, participants answered questionnaires on the online form regarding their experience with that specific view: the Trust in Automated Systems Test (TOAST), the System Usability Scale (SUS), and a set of additional feedback questions.

4. **Second Interface Evaluation (MMV or SMV):** Participants repeated step 3 for the other interface, using a different set of five historical scenarios presented through the online form.

5. **Overall Comparison and Qualitative Feedback:** After evaluating both views, participants answered questions on the online form comparing the two interfaces regarding overall preference, trustworthiness, impact on confidence, information complexity, most helpful features, suggestions for improvement, and risk awareness. They also rated their willingness to invest a hypothetical €1000 based purely on the MMV's recommendations via the form.

6. **Debrief and Submission:** The facilitator briefly discussed the experience with the participant, answered any final questions, and the participant then submitted their completed online form.

The study was designed to take approximately 40-50 minutes per participant. The controlled environment and presence of a facilitator ensured consistency in procedure and allowed for immediate clarification of non-leading questions.

7.1.5 Data Collection and Measures

A combination of quantitative and qualitative data was collected:

Scenario-Based Decisions and Confidence

For each of the 10 scenarios (5 per view), participants' hypothetical investment actions and confidence scores (1-7 Likert scale) were recorded. An average confidence score per view per participant was calculated.

User Action and System Recommendation Correctness

The correctness of participants' hypothetical investment decisions and the system's own recommendations was evaluated against the actual IBM stock performance over the subsequent 10 trading days for each scenario date. For this evaluation, both user choices (Strong Buy, Weak Buy, Hold, Sell) and system recommendations were first mapped to one of three definitive market actions: "Buy", "Hold", or "Sell". Specifically, "Strong Buy" and "Weak Buy" were mapped to "Buy". The actual market outcome was also categorized into one of these three actions. This categorization utilized a 1% threshold, a level considered significant for defining price movements and targeted in predictive financial studies [SK21; DA12]. Specifically, a percentage change in stock price over the 10-day period greater than +1% was considered a "Buy" situation, a change less than -1% was a "Sell" situation, and changes between -1% and +1% (inclusive) were considered "Hold" situations. A decision or recommendation was deemed correct if its mapped action matched the categorized actual market outcome.

Trust in Automated Systems Test (TOAST)

TOAST is a 9-item questionnaire designed to measure user trust in automated systems, including aspects of understanding and performance [Woj+20]. Participants rated their agreement with statements (e.g., "I understand what the system should do" and "The system helps me achieve my goals") on a 7-point Likert scale (1 = Strongly Disagree, 7 = Strongly Agree). Scores were calculated for:

- **TOAST Overall:** Average of all 9 items.
- **TOAST Understanding:** Average of items 1, 3, 4, 8 (e.g., "I understand the limitations of the system" and "I understand how the system executes tasks").
- **TOAST Performance:** Average of items 2, 5, 6, 7, 9 (e.g., "The system performs consistently" and "I feel comfortable relying on the information provided by the system").

Higher scores indicate greater trust, understanding, or perceived performance [Woj+20].

System Usability Scale (SUS)

SUS is a 10-item questionnaire providing a global measure of perceived usability [Sau11]. Participants rated their agreement with statements (e.g., "I thought the system was easy to use" and "I found the system unnecessarily complex") on a 5-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree). SUS scores are calculated by summing item contributions (odd items: score - 1; even items: 5 - score) and multiplying by 2.5, resulting in a score from 0 to 100 [Sau11]. A score of 68 is considered average [Sau11].

Additional Feedback Metrics

Participants rated their agreement (1-7 Likert scale) with statements specifically about each view:

- "I feel the information provided by this view is trustworthy".
- "I understand how this view arrives at its final recommendation".
- "I would feel comfortable using this view to inform real financial decisions".
- "This view helps me understand the potential risks or uncertainty associated with the prediction".

They also rated the "appropriateness of the amount of information presented" (1 = Far too little, 4 = Just right, 7 = Far too much).

Qualitative Feedback

Open-ended questions captured participants' reasoning for their preferences, perceived trustworthiness, helpful features, suggestions for improvement, and risk awareness.

Observational Data

The facilitator observed participants' interactions with the system interfaces during the study sessions. Specific interactions with key features (e.g., analyzing historical pattern charts and examining detailed prediction cards) were noted for each participant to understand engagement patterns with different components of the SMV and MMV.

7.2 Results

This section presents the analysis of the data collected from the 22 participants. Statistical significance was assessed at the $p < 0.05$ level using paired t-tests for within-subjects comparisons and independent samples t-tests for order effect analysis.

7.2.1 Analysis of Presentation Order Effects

An essential step in validating the within-subjects design was to assess whether the order in which participants experienced the Single Model View (SMV) and the Multiple Models View (MMV) influenced their evaluations. To this end, independent samples t-tests were conducted. These tests compared the mean scores on key outcome measures: the TOAST overall score, the System Usability Scale (SUS) score, and the average scenario confidence, between the participant group that interacted with the SMV first and the group that encountered the MMV first.

These t-tests consistently indicated no statistically significant differences (at the $p < 0.05$ threshold) for any primary metrics when evaluating the SMV or the MMV. For example, the p-value for the difference in TOAST overall scores for the SMV between the two order groups was $p = 0.070$, which, while showing a slight numerical divergence, did not meet the conventional criterion for statistical significance. All other comparisons for TOAST scores on the MMV, SUS scores for both views and average scenario confidence scores for both views yielded p-values comfortably above this threshold, typically ranging from $p = 0.268$ to $p = 0.840$.

The consistent absence of statistically significant order effects across these key measures suggests that the counterbalancing of the interface presentation effectively mitigated biases arising from the sequence of exposure. This finding is important as it lends greater confidence to interpreting subsequent, direct comparisons between the SMV and MMV, allowing observed differences to be more robustly attributed to the fundamental characteristics of the interfaces themselves rather than an artifact of the presentation order.

7.2.2 Quantitative Comparison of Interface Views

The quantitative data gathered from the 22 participants thoroughly compared the Single Model View (SMV) and the Multiple Models View (MMV) across several dimensions. Analysis using paired t-tests, with statistical significance set at $p < 0.05$, revealed notable differences in how users perceived and interacted with the two interfaces.

Scenario-Based Confidence

A primary indicator of user experience, **scenario-based confidence**, was significantly higher when participants used the MMV. On a 7-point scale, average confidence with the MMV reached a Mean of 4.96 (Standard Deviation = 0.91), compared to a Mean of 3.89 (Standard Deviation = 1.18) for the SMV ($t(21) = -3.94, p = 0.0007$). This suggests that the richer informational context of the MMV, visualized in Figure 7.3, instilled greater assurance in users' hypothetical investment decisions.

User Action and System Recommendation Correctness

Examining the **correctness of these decisions**, as shown in Figure 7.4, users tended to perform better with the MMV. Across 110 decisions per view, the user action correctness rate was 44.55% for the MMV, compared to 31.82% for the SMV. A z-test for two proportions indicated this difference approached statistical significance ($z = -1.94, p = 0.0520$). While the

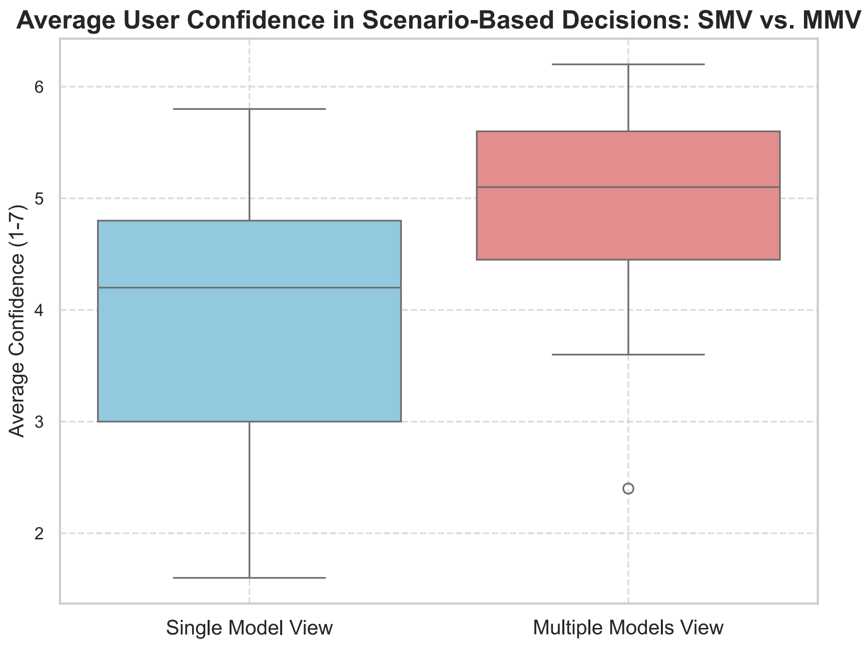


Figure 7.3: Average User Confidence in Scenario-Based Decisions: SMV vs. MMV. Boxplots comparing average confidence scores (N=22, 1-7 scale) for decisions made with the Single Model View (SMV, light blue) and Multiple Models View (MMV, light red). The MMV elicited notably higher median confidence (MMV Median=5.10) compared to the SMV (SMV Median=4.20), a statistically significant difference ($p < 0.001$). Each boxplot visualizes the median, interquartile range, and data spread.

MMV's system recommendations were numerically more accurate (60.00% vs. 40.00% for SMV over 10 scenarios), this difference was not statistically significant ($z = -0.89, p = 0.3711$), likely due to the limited number of scenarios for this specific evaluation.

Adherence to System Recommendations

The study also examined how frequently participants' decisions aligned with the system's recommendations for each view, referred to as **adherence**. For the Single Model View (SMV), participants followed the system's recommendations 63.64% of the time across 110 decisions. For the Multiple Models View (MMV), this rate was very similar, at 62.73% over 110 decisions. A z-test for two proportions confirmed no statistically significant difference in adherence rates between the two views ($z = 0.14, p = 0.8888$). This suggests that while the MMV was perceived as more trustworthy and led to higher confidence, it did not necessarily lead to greater direct adherence to its specific final recommendations compared to the simpler SMV.

Trust (TOAST Scores)

The core construct of **trust**, measured by the Trust in Automated Systems Test (TOAST), also favored the MMV, see Figure 7.5. The MMV achieved a significantly higher TOAST overall score (Mean = 5.54, SD = 0.71) than the SMV (Mean = 5.02, SD = 0.87; $t(21) = -3.41, p = 0.0026$). This superior trust perception appears largely driven by the MMV's perceived performance, as its TOAST performance subscale score (Mean = 5.56, SD = 0.66) was significantly higher than the SMV's (Mean = 4.68, SD = 0.89; $t(21) = -5.03, p < 0.0001$). Interestingly, both views were perceived similarly regarding system understanding, with no significant difference in their TOAST understanding subscale scores (MMV Mean = 5.50, SD = 1.00 vs. SMV Mean = 5.44, SD = 1.12; $p = 0.7699$).

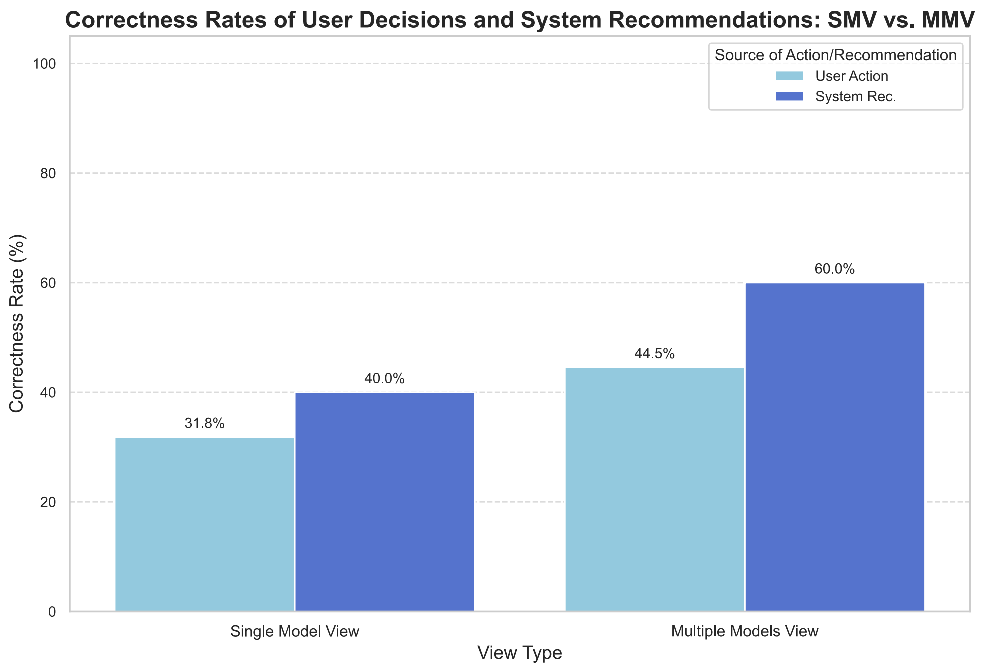


Figure 7.4: Correctness Rates of User Decisions and System Recommendations: SMV vs. MMV. Bar chart illustrating the percentage of correct hypothetical investment decisions by users and correct system recommendations for the Single Model View (SMV) and Multiple Models View (MMV). User actions with MMV (44.5%) tended to be more correct than with SMV (31.8%). MMV system recommendations (60.0%) were also numerically more accurate than SMV's (40.0%).

Usability (SUS Scores)

In contrast to trust, the simpler **Single Model View was perceived as significantly more usable**. The SMV obtained an average System Usability Scale (SUS) score of 83.86 (SD = 9.87), indicating excellent usability. The MMV, while still achieving a good to excellent SUS score of 76.82 (SD = 10.18), was rated lower ($t(21) = 2.99, p = 0.0069$). Both scores, detailed in Figure 7.6, comfortably exceeded the average SUS benchmark of 68.

Additional Feedback Metrics

Further direct **feedback metrics**, visualized in Figure 7.7, consistently underscored the MMV's advantages in fostering a positive user perception beyond basic usability. Participants found the information from the MMV significantly more trustworthy (MMV Mean = 5.18 vs. SMV Mean = 3.23; $p < 0.0001$) and reported a better understanding of its recommendation logic (MMV Mean = 5.68 vs. SMV Mean = 4.18; $p = 0.0046$). This translated into significantly greater comfort using the MMV to inform hypothetical real financial decisions (MMV Mean = 4.91 vs. SMV Mean = 2.86; $p < 0.0001$). Moreover, the MMV was rated significantly more effective in helping users understand potential risks and uncertainties (MMV Mean = 5.18 vs. SMV Mean = 3.55; $p < 0.0001$). Regarding the appropriateness of information amount (1=Far too little, 4=Just right, 7=Far too much), the SMV was seen as offering slightly too little information (Mean = 2.91). At the same time, the MMV was rated closer to optimal, though tending towards slightly more information (Mean = 4.50), a significant difference ($p < 0.0001$).

Perceived Impact of MMV Features and Willingness to Invest

Finally, the study probed the **perceived impact of MMV's unique features** and users' **willingness to invest** based purely on its recommendations. The MMV's features (multiple

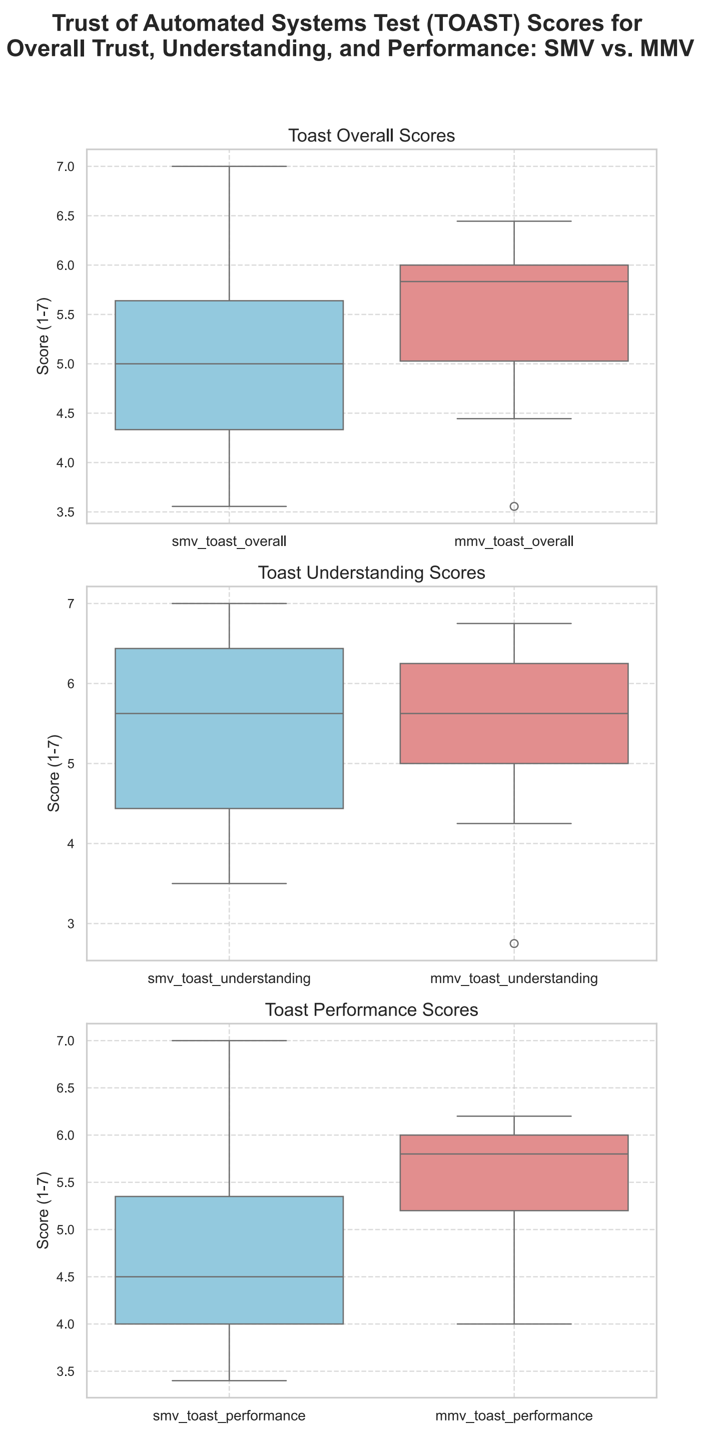


Figure 7.5: Trust of Automated Systems Test (TOAST) scores for overall trust, understanding, and performance: SMV vs. MMV. Comparative boxplots of TOAST subscale scores (N=22, 1-7 scale) for the Single Model View (SMV, light blue) and Multiple Models View (MMV, light red). The MMV showed significantly higher median scores for overall trust (MMV Median=5.83 vs. SMV Median=5.00; $p < 0.01$) and performance (MMV Median=5.80 vs. SMV Median=4.50; $p < 0.001$). Median scores for understanding were similar for both views. Each boxplot visualizes the median, interquartile range, and data spread.

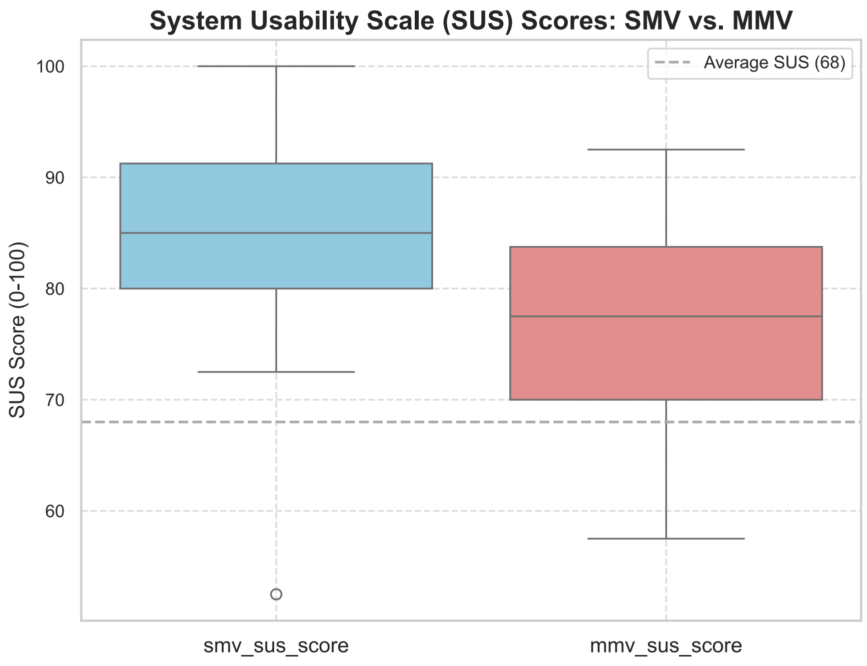


Figure 7.6: System Usability Scale (SUS) Scores: SMV vs. MMV. Boxplots comparing SUS scores (N=22, 0-100 scale) for the Single Model View (SMV, light blue) and Multiple Models View (MMV, light red). While both views scored above the average SUS benchmark of 68 (dashed line), the SMV (Median=85.00) was rated as significantly more usable than the MMV (Median=77.50; $p < 0.01$). Each boxplot displays the median, interquartile range, and data spread.

models, historical comparisons) were reported to significantly increase decision confidence relative to a single forecast, achieving a mean impact score of 5.77 (SD = 0.92) on a 7-point scale where 7 represented a significant increase (Figure 7.8). When asked about their willingness to invest a hypothetical €1000 based purely on MMV's recommendations, participants expressed a moderate inclination to trust the MMV for such a decision, with a mean score of 4.27 (SD = 1.42) on a 7-point scale (Figure 7.9).

7.2.3 Qualitative Feedback Analysis

Qualitative data from open-ended questions provided more profound insights into user perceptions.

Overall Interface Preference and Trustworthiness

- **Overall Interface Preference for Understanding Predictions:** 21 out of 22 participants (95.5%) preferred the MMV.
- **Overall Trustworthiness Preference:** All 22 participants (100.0%) found the MMV more trustworthy.

Reasons for Perceived Trustworthiness of MMV

Participants consistently cited the MMV's enhanced information and context as primary reasons for its higher trustworthiness. Key themes included:

- **More Information and Context:** The ability to compare with historical data and see multiple model outputs was highly valued.

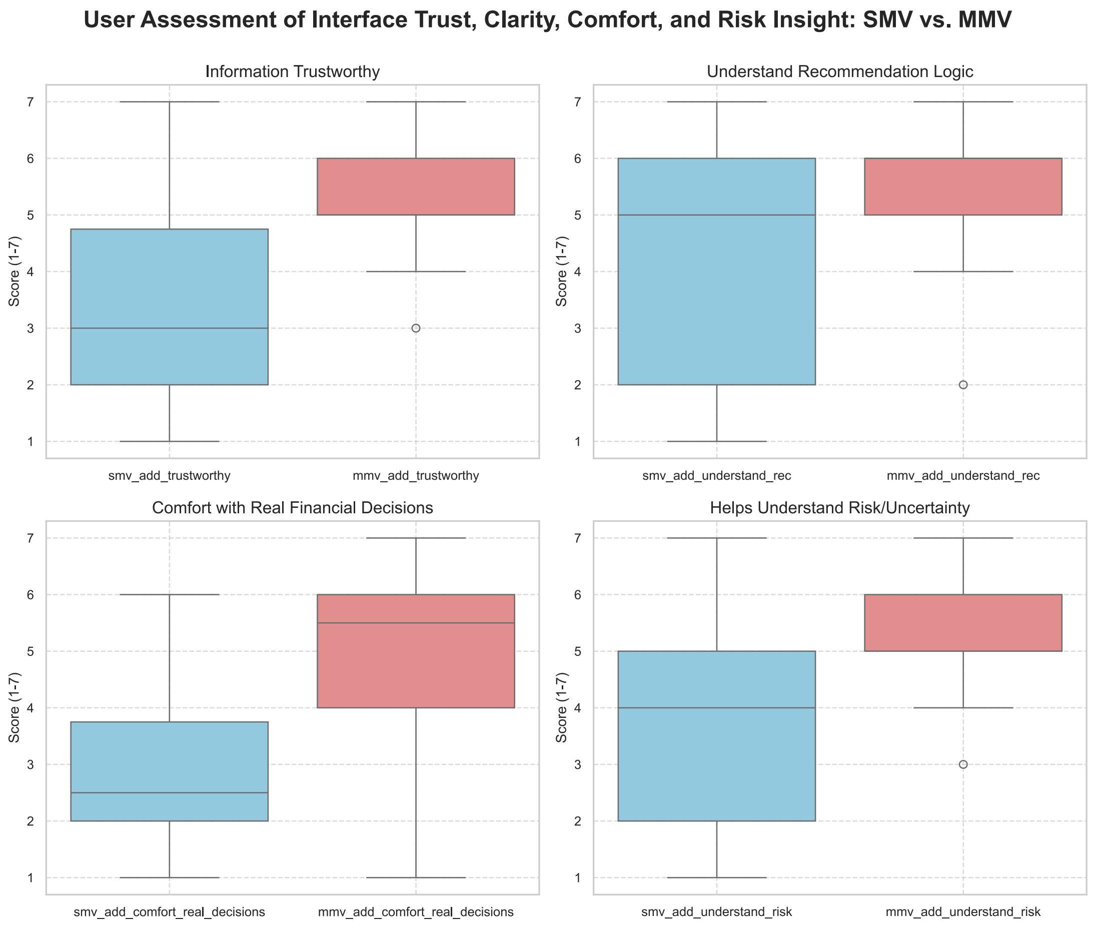


Figure 7.7: User Assessment of Interface Trust, Clarity, Comfort, and Risk Insight: SMV vs. MMV. Comparative boxplots of participant ratings (N=22, 1-7 scale) for the Single Model View (SMV, light blue) and Multiple Models View (MMV, light red) across four dimensions. The MMV consistently received higher median ratings for information trustworthy (MMV Median=5.0 vs. SMV Median=3.0), understanding recommendation Logic (MMV Median=6.0 vs. SMV Median=5.0), comfort with real financial decisions (MMV Median=5.5 vs. SMV Median=2.5), and helping understand risk/uncertainty (MMV Median=5.0 vs. SMV Median=4.0). These differences were statistically significant ($p < 0.01$ for all comparisons). Each boxplot shows the median, interquartile range, and whiskers.

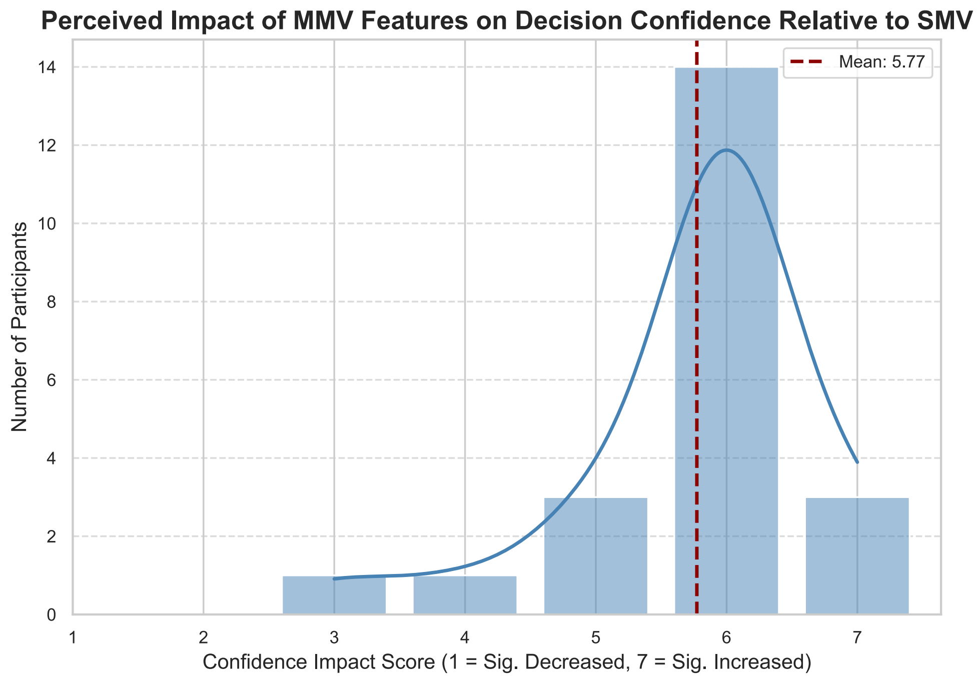


Figure 7.8: Perceived Impact of MMV Features on Decision Confidence Relative to SMV. Histogram and overlaid kernel density estimate (KDE) showing the distribution of participant ratings (N=22, 1-7 scale) on how MMV's features influenced their confidence. The dashed red line indicates the mean impact score of 5.77, suggesting a strong positive influence of MMV features on user confidence.

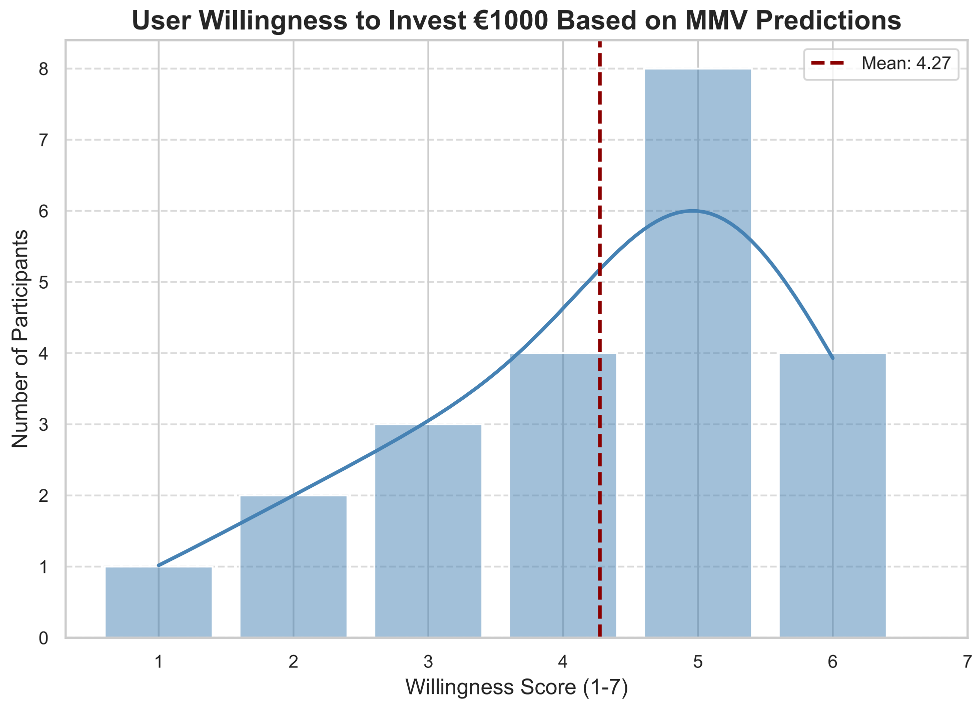


Figure 7.9: User Willingness to Invest €1000 Based on MMV Predictions. Histogram and overlaid kernel density estimate (KDE) illustrating participant willingness (N=22, 1-7 scale) to invest based on the Multiple Models View. The dashed red line represents the mean willingness score of 4.27, indicating a moderate investment inclination.

- **Multiple Perspectives:** Users found reassurance in seeing agreement (or disagreement) among multiple models, believing this offered a more credible basis than a single model's output.
- **Transparency:** Exposing details behind recommendations, such as model performance on historical patterns, made the system feel less like a black-box.

Perceived Impact of MMV Features on Confidence

The qualitative feedback aligned with the quantitative finding that MMV features generally increased confidence (Mean quantitative impact: 5.77/7). The reasons provided included:

- **Historical Validation:** Seeing how models performed on similar past patterns provided tangible evidence of their potential reliability.
- **Consensus Building:** Agreement among multiple models or consistency with historical trends bolstered confidence.
- **Realistic Assessment:** Some noted that while seeing model disagreements could lower immediate confidence in a specific prediction, it fostered a more realistic (and thus trustworthy) understanding of market uncertainty.

Information Amount and Complexity

When comparing the two views:

- The vast majority (72.7%) felt "The Multiple Models View provided valuable context despite its complexity".
- Smaller, equal proportions (9.1% each) felt "The Single Model View lacked sufficient detail or context", "Both views had strengths and weaknesses regarding information presentation", or "The Single Model View was refreshingly simple and clear".

This suggests that while MMV was more complex, its informational benefits often outweighed this complexity for most users.

Most Helpful and Trust-Enhancing Features

Across both views, but particularly for MMV, participants highlighted:

- **Historical Comparison/Pattern Similarity Chart:** This was very frequently mentioned as the most helpful feature for contextualizing current predictions.
- **Multiple Model Predictions:** The ability to see outputs from different models and their past performance (error metrics) was key to building trust.
- **Clear Recommendation:** The system-generated recommendation was appreciated, especially when its basis (e.g., "based on best models historically in MMV") was clear.

Suggestions for Improvement

Participants offered several constructive suggestions:

- **Enhanced Explainability (XAI):** Desire for more insight into why models make certain predictions (e.g., feature importance).
- **More Contextual Data:** Suggestions included adding fundamental data (company news, reports), sector comparisons, or broader market indicators.
- **Customization and Control:** Ideas like user-defined thresholds for recommendations or longer lookback periods for historical analysis.

- **UI/UX Refinements:** Better guidance on information flow, particularly in MMV, and clearer explanations for some metrics. Some suggested incorporating historical similarity into SMV for a fairer comparison.

Awareness of Risk and Uncertainty

A central aspect of trustworthy AI is its ability to convey its predictions' inherent limitations and uncertainties. The study found a strong consensus among participants that the Multiple Models View (MMV) was noticeably superior to the Single Model View (SMV) in this regard. When asked which view made them more aware of potential risks or uncertainty, 20 out of 22 participants (90.9%) explicitly stated the MMV. Conversely, no participant indicated that the SMV was more effective for risk awareness. Two participants (9.1%) found neither view particularly helpful or both views equally (un)helpful in conveying risk.

The primary reasons cited by participants for the MMV's effectiveness in highlighting risk and uncertainty included:

- **Visible Model Disagreements:** Seeing different models predict different outcomes highlighted market volatility and prediction uncertainty.
- **Historical Outcome Variability:** Observing that similar past patterns could lead to vastly different actual outcomes was a key factor.
- **Exposure of Model Errors:** The MMV's display of past prediction errors made the fallibility of models apparent.

These elements collectively contributed to a more nuanced and cautious interpretation of the predictions offered by the MMV, aligning with the goal of fostering calibrated trust.

7.2.4 Observational Analysis of Feature Interaction

During the study sessions, the facilitator observed and logged participants' interactions with key features of both the Single Model View (SMV) and the Multiple Models View (MMV). This observational data provided insights into user engagement patterns with different interface components.

For the **Multiple Models View (MMV)**, core features demonstrated high levels of interaction. All participants (100%) actively engaged with the "IBM Stock Historical Pattern Similarity" chart, utilizing the "Hover over prediction cards" functionality (for both the selected window and historical patterns), and analyzed the "Selected window" predictions card. The detailed "Historical Similar Patterns" analysis cards, which allow for in-depth comparison, also saw high engagement, with 17 out of 22 participants (77.3%) interacting with them. Interestingly, the "Model Performance on Historical Patterns" summary, which provides an aggregated view of model reliability in similar past scenarios, was engaged with by 11 out of 22 participants (50.0%). This moderate engagement might be attributed to participants understanding that the system's final recommendation (presented in the "Selected window" card) was already derived from these underlying analyses. Consequently, some users may have found it less necessary to perform their own detailed examination of this specific component, trusting the system to have factored its insights into the final output. Nevertheless, qualitative feedback often cited this feature as being trust-enhancing for those who engaged with it.

The interaction was similarly focused on its primary components in the **Single Model View (SMV)**. All participants (100%) engaged with the "IBM Stock Random Forest Prediction" chart and the "Prediction results" cards. The "Hover over prediction cards" feature in the SMV was used by 13 out of 22 participants (59.1%).

Regarding features common to both views, 10 out of 22 participants (45.5%) were observed clicking on info icons (designed to provide more information about specific features or formulas), and 12 out of 22 participants (54.5%) utilized advanced chart functionalities offered by the Plotly library, such as zooming or panning.

The high observed interaction rates with the MMV's historical comparison and multi-model prediction features strongly corroborate the qualitative feedback where users cited these elements as particularly useful and trust-enhancing. Both views' universal engagement with core prediction elements confirms their fundamental importance. The varied interaction levels with more detailed summaries (like the MMV's model performance summary) or auxiliary features (like info icons and advanced chart tools) suggest differing user needs for depth of information versus reliance on readily available, aggregated system outputs or varying levels of technical comfort and exploratory behavior.

7.3 Discussion

The user study yielded significant insights into how different presentations of machine learning-based stock predictions affect user trust, confidence, and perceived utility.

7.3.1 Key Findings and Interpretation

The primary finding is that the Multiple Models View (MMV), which integrated predictions from six models with historical similarity analysis, was significantly preferred and perceived as more trustworthy than the Single Model View (SMV). This was evident in higher TOAST overall and performance scores, direct ratings of trustworthiness, comfort with real financial decisions, and understanding of recommendation logic for the MMV. Specifically, when asked directly, all 22 participants (100.0%) stated they found the MMV more trustworthy overall, and 21 out of 22 participants (95.5%) preferred the MMV for understanding the stock predictions.

This enhanced trust appears to be driven by the MMV's ability to provide richer context and transparency. The historical pattern comparisons, coupled with the performance of multiple models on these past analogous situations, offered users an evidence-based framework for evaluating current predictions. This aligns with XAI principles advocating for explanations that help users understand an AI's reasoning and reliability [Bar+20]. The features of the MMV were reported to increase decision confidence significantly relative to a single forecast.

Interestingly, while the Multiple Models View (MMV) was perceived as more complex, reflected in its System Usability Scale (SUS) score of 76.82 being significantly lower than the SMV's score of 83.86, this complexity was often deemed acceptable due to the valuable context it provided. It is important to note that the MMV's SUS score still indicates good to excellent usability, comfortably exceeding the industry benchmark of 68. This suggests a trade-off: users may tolerate a degree of higher complexity if it leads to a more transparent, justifiable, and ultimately more trustworthy system, especially when the perceived usability remains at a good level.

The MMV also excelled in making users more aware of the inherent risks and uncertainties in stock prediction. Showcasing model disagreements and variable historical outcomes fostered a more calibrated and realistic sense of trust rather than blind faith or undue skepticism. This ability to accurately convey uncertainty is a hallmark of a truly trustworthy system [Afr+24].

While user decision correctness showed a positive trend for MMV (44.55% vs. 31.82% for SMV, $p = 0.052$), it remained below 50%, underscoring the difficulty of market prediction even with advanced tools. The system's recommendation correctness was numerically higher for MMV (60% vs. 40% for SMV), though this difference was not statistically significant with the limited number of scenarios. Adherence to system recommendations was similar for both views (SMV: 63.64%, MMV: 62.73%, $p = 0.8888$), indicating that the increased trust in MMV did not translate into a significantly higher rate of following its explicit advice over SMV, perhaps because the MMV also better-equipped users to make their own informed deviations.

7.3.2 Answering the Research Question

The research question was: *"How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?"*. The study findings strongly suggest that such an approach, as embodied by the MMV, can indeed enhance user trustworthiness. The mechanisms for this enhancement include:

1. **Increased Transparency and Reduced Opacity:** Presenting multiple model outputs and their past performance on similar patterns clarifies the prediction process compared to a single black-box output. Users could see areas of model consensus and divergence.
2. **Contextualization through Historical Analogy:** Grounding abstract ML predictions in concrete, similar past market scenarios (sharing the same 20-day feature input structure as the models) made them more relatable and interpretable, a key aspect of similarity-based XAI [Han+21; NSK22]. High user engagement with these features confirmed their value.
3. **Evidence-Based Reasoning for Recommendations:** The MMV's recommendation, derived from the historical performance of models in analogous situations, offered a more justifiable basis for action than the SMV's direct model output.
4. **Calibrated Understanding of Uncertainty and Risk:** The MMV's design explicitly exposed model fallibility and market unpredictability, leading to a more informed and realistic level of trust.

Despite its higher complexity, the combination of these factors contributed to the MMV being perceived as a more reliable and confidence-inspiring tool.

7.3.3 The Interplay of Information, Trust, and Usability: A Design Trade-off

The study highlights a nuanced interplay between the amount of information presented, perceived system trust, and usability. While the richer informational context of the Multiple Models View (MMV) significantly boosted user trust and confidence, it also led to a perception of higher complexity, as evidenced by its lower System Usability Scale (SUS) score compared to the more streamlined Single Model View (SMV). However, with the MMV still achieving a SUS score indicative of good to excellent usability, this finding suggests that users are often willing to navigate a more complex interface if the additional information demonstrably enhances transparency and provides a more justifiable basis for decision-making, particularly in domains characterized by high stakes and inherent uncertainty like financial forecasting. This underscores a critical design challenge: optimizing the presentation of complex, trust-enhancing information in a manner that minimizes cognitive load and maintains high usability.

7.3.4 Insights for Future System Design

The user feedback provided valuable directions for future development:

- **Deeper Explainability:** Users expressed a desire for more granular explanations of model behavior, such as identifying which input features most influenced a prediction. Incorporating techniques like SHAP or LIME could address this.
- **Richer Contextual Information:** Integrating external data sources (e.g., news sentiment, sector trends, fundamental company data) could further enhance the system's analytical power and user understanding. One participant suggested comparing IBM's performance to a tech sector moving average, which aligns with concepts like statistical arbitrage and could be a valuable feature [JCC23].
- **Adaptive Presentation of Information:** Future systems could offer different levels of detail or explanation based on user expertise or preference, potentially mitigating the

complexity issue of the MMV while retaining its trust-enhancing benefits.

- **Refined Recommendation Logic:** The method of deriving the final recommendation could be further sophisticated. For instance, using the results of "model performance on historical patterns" as weights to create a weighted average prediction from multiple models, rather than relying on the best model, could be explored. Another insight was applying the average historical residual; the deviation of past predictions from actuals, on similar patterns to current predictions to assess their significance.
- **Fairer Benchmarking in Studies:** For future comparative studies, including historical pattern similarity features in both single-model and multi-model views could provide a more controlled comparison of the multi-model aspect itself.

7.3.5 Limitations of the Study

This study has several limitations:

- **Sample Characteristics:** The participant pool, while diverse in ML/visualization experience, was somewhat skewed towards younger males with limited direct stock investing experience. Results might differ with a more experienced investor population.
- **Hypothetical Decisions:** Investment decisions were hypothetical, which may not fully reflect behavior under real financial risk.
- **Limited Scenarios for System Correctness:** The assessment of system recommendation correctness was based on only 5 scenarios per view, limiting the statistical power of this particular comparison.
- **Fixed Stock and Features:** The study focused on a single stock (IBM) and a specific set of technical indicators. Generalizability to other assets or feature sets requires further investigation.
- **Default Model Hyperparameters:** The underlying ML models used default hyperparameters. Performance and user trust could potentially be further improved with optimized models.

7.4 Conclusion

The user study provides compelling evidence that integrating multiple machine learning models with explanations grounded in similar historical patterns can significantly enhance user trustworthiness in stock market prediction systems. The Multiple Models View, despite its increased complexity, was overwhelmingly preferred and trusted over the Single Model View. This was attributed to its greater transparency, contextualization through historical analogies, evidence-based recommendations, and a more realistic portrayal of prediction uncertainty. These findings underscore the importance of designing AI-driven financial tools that not only aim for predictive accuracy but also prioritize interpretability and provide users with the necessary context to build informed confidence in algorithmic outputs. While challenges in managing information complexity remain, the benefits of a multi-faceted, evidence-rich approach appear decisive for fostering user adoption and responsible use of AI in financial decision-making.

Chapter 8

Conclusions

The endeavor to predict stock market movements is a formidable challenge, one that has captivated researchers and practitioners for decades. This master's thesis embarked on this complex domain not merely to chase predictive accuracy, but to confront an equally, if not more, critical hurdle: the inherent opacity of many algorithmic forecasting methods and the resultant deficit in user trust. The central question guiding this research, *"How can multiple machine learning models be used to enhance trustworthiness in stock market predictions using similar historical patterns?"*, sought to explore a pathway towards more transparent and confidence-inspiring AI-driven financial tools. This concluding chapter moves beyond a simple recapitulation of findings; it offers a personal and critical reflection on the journey undertaken, the insights gleaned, the lessons learned, and the profound complexities encountered at the intersection of machine learning, financial markets, and human-computer interaction.

Reflecting on the core research question, the development and empirical evaluation of the Multi-Model View (MMV) system provided compelling, albeit nuanced, answers. The user study detailed in Chapter 7 unequivocally demonstrated that an approach integrating predictions from multiple diverse models, contextualized by historically similar market patterns and transparently showcasing past model performance in those analogous situations, can significantly enhance user trustworthiness. Participants overwhelmingly preferred the MMV, citing its richer informational context, the ability to see model consensus or divergence, and the evidence-based grounding of its recommendations as key factors. This resonates deeply with the principles of Explainable AI (XAI), suggesting that even without delving into the intricate mathematical innards of each model, providing users with comparative, historical, and performance-based evidence can substantially demystify the black-box. The enhanced sense of understanding the recommendation logic, comfort with making hypothetical financial decisions, and a more acute awareness of potential risks and uncertainties were all testament to the MMV's success in fostering a more calibrated and informed trust.

However, the journey also illuminated the intricate nature of trust itself. While the MMV fostered significantly higher confidence and perceived trustworthiness, this did not directly translate into a statistically significant increase in users' adherence to its specific recommendations compared to the simpler Single Model View (SMV). This suggests that the richer information, while building trust in the system's process and integrity, also empowered users to make more independent, possibly divergent, judgments. Perhaps the MMV, by making model fallibility and market uncertainty more apparent, encouraged a healthy skepticism alongside trust, which is arguably a more desirable outcome than blind adherence. Furthermore, the trade-off between informational richness and perceived usability was evident. The MMV, while still rated as highly usable, was inevitably more complex than the SMV. The willingness of most users to navigate this increased complexity for the sake of enhanced understanding and trust underscores a critical design insight: in high-stakes domains like finance, users may prioritize transparency and justifiable reasoning over mere simplicity, provided the interface remains manageable.

The predictive modeling phase (Chapter 5) was, in itself, a sobering lesson in the inherent difficulties of stock market forecasting. The pervasive challenge of overfitting was starkly evident; models that exhibited impressive performance on training data often failed to generalize to unseen test data, with most yielding negative R-squared scores. This reality reinforced the already deeply held understanding of the immense challenge of consistently finding true market alpha. Yet, an intriguing and counterintuitive finding was the divergence between these weak statistical predictive metrics and the performance of some models, notably Random Forest and N-BEATS, in an idealized trading strategy simulation. Despite their poor statistical fit on test data, these models generated positive total returns that outperformed a buy-and-hold strategy. This discrepancy provokes critical questions about the suitability of standard regression metrics alone for evaluating financial AI and hints that even a marginal, statistically weak predictive edge, if consistently applied, might hold potential in financial markets' non-linear and often event-driven dynamics. It highlights the immense difficulty in capturing truly generalizable patterns and underscores the need for robust regularization, meticulous hyperparameter optimization, and perhaps more sophisticated feature engineering than was employed with default settings in this initial exploration.

The development process itself was a significant learning curve. Integrating six distinct machine learning models, each with its own input requirements and characteristics, into a cohesive system was a substantial software engineering task. Designing the historical similarity engine, particularly the weighting of raw returns versus PCA-reduced technical indicators using Dynamic Time Warping, required careful consideration to ensure that similarity was meaningful in the context of the models' input space. The subsequent design of the user interface, striving to present a wealth of information without overwhelming the user, was an iterative process. The realization of how profoundly the presentation of information impacted user perception was one of the most significant "aha!" moments. It became clear that trustworthiness is not solely a function of a model's latent predictive capability but is heavily mediated by how its outputs, uncertainties, and rationale are communicated to the user.

Personally, this master's thesis has been a journey of immense growth. It has solidified my fascination with the intersection of artificial intelligence and quantitative finance, revealing not just the potential of algorithms but also the paramount importance of human-centric design. The challenge of engineering meaningful input features, debugging intricate code, interpreting often ambiguous results, and structuring these findings into coherent narratives has honed my analytical and problem-solving skills. The process of designing and conducting the user study, in particular, provided invaluable experience in empirical research methods and the subtleties of human-computer interaction. While the inherent difficulty of finding alpha was clear from the outset, actively confronting the stubborn unpredictability of financial data and the limitations of the models necessitated strategic decision-making at critical junctures. Yet, overcoming these hurdles brought a profound sense of accomplishment and a deeper appreciation for the rigor and perseverance required in research. This project has certainly fueled my desire to delve further into areas like advanced time-series analysis, XAI techniques specifically tailored for both financial contexts and high-stakes environments, and the cognitive aspects of decision-making with AI support.

Acknowledging the limitations of this study, as detailed in Chapter 7, is crucial. The relatively small and somewhat homogenous participant sample, the hypothetical nature of investment decisions, the focus on a single stock, and the use of default model hyperparameters all circumscribe the generalizability of the findings. These limitations, however, also illuminate clear pathways for future research. Exploring more sophisticated XAI techniques to provide feature-level explanations, integrating richer contextual data, such as news sentiment or macroeconomic indicators, developing adaptive interfaces that cater to varying user expertise, and rigorously backtesting with transaction costs and slippage are all promising avenues. Furthermore, a more refined recommendation logic, incorporating a weighted ensemble based on historical performance in similar situations, could offer further improvements.

In conclusion, this thesis ventured into the challenging terrain of enhancing trustworthiness in AI-driven stock forecasting. The findings strongly suggest that a multi-faceted approach, combining diverse model perspectives with transparent historical evidence, can indeed foster a more robust and calibrated sense of user trust. While the quest for perfectly accurate stock prediction remains elusive, the journey towards creating more understandable, interpretable, and ultimately more trustworthy AI tools is a critical one. The insights gained from this research, I believe, contribute a small but meaningful step in that direction, emphasizing that in the complex dance between algorithms and human decision-makers, especially in high-stakes financial domains, transparency and context are not just desirable features, but essential foundations for responsible innovation. The path forward requires a continued commitment to designing systems that empower users, not by promising infallible predictions, but by providing the tools and insights needed to navigate uncertainty with informed confidence.

Bibliography

[ADE12] Mahmood Moein Aldin, Hasan Dehghan Dehnavi, and Somayye Entezari. "Evaluating the Employment of Technical Indicators in Predicting Stock Price Index Variations Using Artificial Neural Networks (Case Study: Tehran Stock Exchange)". In: *International Journal of Business and Management* 7.15 (2012), p. 25. DOI: 10.5539/ijbm.v7n15p25. URL: https://doi.org/10.5539/ijbm.v7n15p25.

[Afr+24] Saleh Afroogh et al. "Trust in AI: progress, challenges, and future directions". In: *Humanities and Social Sciences Communications* (2024). DOI: 10.1057/s41599-024-04044-8. URL: https://doi.org/10.1057/s41599-024-04044-8.

[Ali23] Ali. *Unlocking the Secrets of Financial Time Series: A Deep Dive into Autoregressive Models (AR).* 2023. URL: https://medium.com/@Alidotab/unlocking-the-secrets-of-financial-time-series-a-deep-dive-into-autoregressive-models-ar-b2e58428928e.

[Bar+20] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI". In: *Information Fusion* (2020). URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[BB21] Milo Bianchi and Marie Briere. "Robo-Advising: Less AI and More XAI?" In: *SSRN Electronic Journal* (2021). URL: https://ssrn.com/abstract=3825110.

[Bei20] Jan Beitner. *PyTorch Forecasting Documentation.* Read the Docs. 2020. URL: https://pytorch-forecasting.readthedocs.io/en/v1.2.0/index.html.

[BGC22] Malti Bansal, Apoorva Goyal, and Apoorva Choudhary. "Stock Market Prediction with High Accuracy using Machine Learning Techniques". In: *Procedia Computer Science.* Elsevier, 2022. URL: https://www.sciencedirect.com/science/article/pii/S1877050922020993.

[BK12] Marshall E. Blume and Donald B. Keim. "Institutional Investors and Stock Market Liquidity: Trends and Relationships". In: *Jacobs Levy Equity Management Center for Quantitative Financial Research Paper* (2012). URL: https://ssrn.com/abstract=2147757.

[BK23] Weerapat Buachuen and Pittipol Kantavat. "Automated Stock Trading System using Technical Analysis and Deep Learning Models". In: (2023). URL: https://doi.org/10.1145/3628454.3631670.

[Bre01] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001). DOI: 10.1023/A:1010933404324. URL: https://link.springer.com/article/10.1023/A:1010933404324.

[CG16] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Association for Computing Machinery, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[CH15] Tianqi Chen and Tong He. "Higgs Boson Discovery with Boosted Trees". In: *Proceedings of the HEPML 2014 Workshop.* Vol. 42. Journal of Machine Learning Research, 2015, pp. 69–80. URL: http://proceedings.mlr.press/v42/chen14.html.

[Cha+24] Victor Chang et al. “Predicting Economic Trends and Stock Market Prices with Deep Learning and Advanced Machine Learning Techniques”. In: *Electronics* 13.17 (2024). DOI: `10.3390/electronics13173396`. URL: `https://doi.org/10.3390/electronics13173396`.

[Cha21] Inc. Charles Schwab & Co. “Trading Volume as a Market Indicator”. In: *Charles Schwab* (2021). URL: `https://www.schwab.com/learn/story/trading-volume-as-market-indicator`.

[Cha22] S. Kumar Chandar. “Convolutional neural network for stock trading using technical indicators”. In: *Automated Software Engineering* 29 (2022). DOI: `10.1007/s10515-021-00303-z`. URL: `https://doi.org/10.1007/s10515-021-00303-z`.

[Chi22] David Joseph Chiumera. “Deep Reinforcement Learning for Quantitative Finance: Time Series Forecasting using Proximal Policy Optimization”. Master of Information Technology (M.I.T.) Carleton University, 2022. DOI: `10.22215/etd/2022-15153`. URL: `https://hdl.handle.net/20.500.14718/42718`.

[CK24] Jurgita Cerneviciene and Audrius Kabasinskas. “Explainable artificial intelligence (XAI) in finance: a systematic literature review”. In: *Artificial Intelligence Review* 57 (2024), p. 216. DOI: `10.1007/s10462-024-10854-8`. URL: `https://doi.org/10.1007/s10462-024-10854-8`.

[Cor21] Edoardo Corallo. *Analysis of the log files of the StoRM storage system used by the ATLAS experiment, performed with Anomaly Detection through Deep Learning*. Tech. rep. Alma Mater Studiorum – Università di Bologna, Scuola di Scienze, Dipartimento di Fisica e Astronomia, 2021. URL: `https://123dok.org/document/dy4e3j9q-analysis-storm-storage-experiment-performed-anomaly-detection-learning.html`.

[Cur24] Timothy Currie. “Trust in the 100% Remote Workplace in High Growth Technology Consulting Firms”. PhD thesis. University of Southern California, Rossier School of Education, 2024. URL: `https://doi.org/10.25549/usctheses-oUC113996WQ2`.

[DA12] Brett Drury and José Almeida. “Predicting Market Direction from Direct Speech by Business Leaders”. In: 2012. DOI: `10.4230/OASIcs.SLATE.2012.163`. URL: `https://www.researchgate.net/publication/267264720_Predicting_Market_Direction_from_Direct_Speech_by_Business_Leaders`.

[Dev22] XGBoost Developers. *XGBoost Parameters*. XGBoost Documentation. 2022. URL: `https://xgboost.readthedocs.io/en/stable/parameter.html`.

[Dev25] LightGBM Developers. *lightgbm.LGBMRegressor Class Reference*. Microsoft Corporation. 2025. URL: `https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html`.

[Fam70] Eugene F Fama. “Efficient Capital Markets A Review of Theory and Empirical Work”. In: *Journal of finance* (1970). URL: `https://doi.org/10.2307/2325486`.

[Fas24] FasterCapital. *30 70 Levels - Measuring Momentum in Stock Trends and The Ultimate Oscillator*. 2024. URL: `https://fastercapital.com/keyword/30-70-levels.html`.

[Gan20] Akhilesh Ganti. “Adjusted Closing Price: How It Works, Types, Pros & Cons”. In: *Investopedia* (2020). URL: `https://www.investopedia.com/terms/a/adjusted_closing_price.asp`.

[Gee24] The Forex Geek. *Moving Average Alternatives*. 2024. URL: `https://theforexgeek.com/moving-average-alternatives`.

[GMR04] Joao Gama, Pedro Medas, and Ricardo Rocha. “Forest trees for on-line data”. In: *Proceedings of the 2004 ACM Symposium on Applied Computing*. Association for Computing Machinery, 2004, pp. 632–636. DOI: `10.1145/967900.968033`. URL: `https://doi.org/10.1145/967900.968033`.

[Gon+20] Sergio González et al. “A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities”. In: *Information Fusion* (2020). URL: `https://doi.org/10.1016/j.inffus.2020.07.007`.

[Han+21] Kazuaki Hanawa et al. *Evaluation of Similarity-based Explanations.* 2021. URL: `https://arxiv.org/abs/2006.04528`.

[HBP23] Htet Htet Htun, Michael Biehl, and Nicolai Petkov. "Survey of feature selection and extraction techniques for stock market prediction". In: *Financial Innovation* 9 (2023). DOI: `10.1186/s40854-022-00441-7`. URL: `https://doi.org/10.1186/s40854-022-00441-7`.

[HS97] Sepp Hochreiter and Jurgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[HSK18] Bruno Miranda Henrique, Vinicius Amorim Sobreiro, and Herbert Kimura. "Stock price prediction using support vector regression on daily and up to the minute prices". In: *The Journal of Finance and Data Science* 4.3 (2018). DOI: `https://doi.org/10.1016/j.jfds.2018.04.003`. URL: `https://www.sciencedirect.com/science/article/pii/S2405918818300060`.

[Hu21] Xiaokang Hu. "Stock Price Prediction Based on Temporal Fusion Transformer". In: *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI).* 2021. DOI: `10.1109/MLBDBI54094.2021.00019`. URL: `https://ieeexplore.ieee.org/abstract/document/9731073`.

[Huy24] Trung Phong Huyen Chau Nguyen Thiand Doan. "Data Processing and Feature Engineering for Stock Price Trend Prediction". In: *Machine Learning and Other Soft Computing Techniques: Biomedical and Related Applications.* Springer Nature Switzerland, 2024. URL: `https://doi.org/10.1007/978-3-031-63929-6_17`.

[JCC23] Qi Jin, Mihai Cucuringu, and Alvaro Cartea. "Correlation Matrix Clustering for Statistical Arbitrage Portfolios". In: *Proceedings of the Fourth ACM International Conference on AI in Finance.* Association for Computing Machinery, 2023, pp. 557–564. DOI: `10.1145/3604237.3626894`. URL: `https://doi.org/10.1145/3604237.3626894`.

[Jog24] Aaryan Jogani. "The Basics of Technical Analysis". In: *SSRN Electronic Journal* (2024). URL: `https://ssrn.com/abstract=4870943`.

[Kar23] Evangelia Karageorgou. "Forecasting Spain's Electricity Load: A Comparative Analysis of Classical Time Series, Neural Networks, and Deep Learning Models". MA thesis. University of Macedonia, 2023. URL: `https://dspace.lib.uom.gr/bitstream/2159/30101/4/KarageorgouEvangeliaMsc2023.pdf`.

[Ke+17] Guolin Ke et al. "LightGBM: a highly efficient gradient boosting decision tree". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* Curran Associates Inc., 2017, pp. 3149–3157. URL: `https://dl.acm.org/doi/10.5555/3294996.3295074`.

[Kha+22] Wasiat Khan et al. "Stock market prediction using machine learning classifiers and social media, news". In: *Journal of Ambient Intelligence and Humanized Computing* (2022). URL: `https://www.researchgate.net/publication/339938321_Stock_market_prediction_using_machine_learning_classifiers_and_social_media_news`.

[Kha+23] Azaz Khan et al. "A performance comparison of machine learning models for stock market prediction with novel investment strategy". In: *PLOS ONE* (2023). URL: `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0286362`.

[KYA19] Eiman Kanjo, Eman M.G. Younis, and Chee Siang Ang. "Deep Learning Analysis of Mobile Physiological, Environmental and Location Sensor Data for Emotion Detection". In: *Information Fusion* 49 (2019), pp. 46–56. ISSN: 1566-2535. DOI: `10.1016/j.inffus.2018.09.001`. URL: `https://kar.kent.ac.uk/68930/`.

[Li+24] Jingshu Li et al. *Overconfident and Unconfident AI Hinder Human-AI Collaboration.* 2024. URL: `https://arxiv.org/abs/2402.07632`.

[Lim+20] Bryan Lim et al. *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting.* 2020. URL: `https://arxiv.org/abs/1912.09363`.

[LW24] Xinhe Liu and Wenmin Wang. “Deep Time Series Forecasting Models: A Comprehensive Survey”. In: *Mathematics* (2024). URL: https://www.mdpi.com/2227-7390/12/10/1504.

[Mag23] Nick Maggiulli. *Why the Medallion Fund is the Greatest Money-Making Machine of All Time*. 2023. URL: https://ofdollarsanddata.com/medallion-fund/.

[MH25] Seyed Mostafa Mostafavi and Ali Reza Hooman. “Key technical indicators for stock market prediction”. In: *Machine Learning with Applications* 20 (2025), p. 100631. DOI: https://doi.org/10.1016/j.mlwa.2025.100631. URL: https://www.sciencedirect.com/science/article/pii/S2666827025000143.

[Mit24] Cory Mitchell. “Understanding an OHLC Chart and How to Interpret It”. In: *Investopedia* (2024). URL: https://www.investopedia.com/terms/o/ohlcchart.asp.

[MZ24] Leila Mozaffari and Jianhua Zhang. “Predictive Modeling of Stock Prices Using Transformer Model”. In: *Proceedings of the 2024 9th International Conference on Machine Learning Technologies*. Association for Computing Machinery, 2024. URL: https://doi.org/10.1145/3674029.3674037.

[Nar19] Aditya Vijay Narkar. “Stock price prediction using feature engineering and machine learning techniques”. Master of Science Thesis. Kansas State University, Dec. 2019. URL: http://hdl.handle.net/2097/40219.

[Nic23] Steven Nickolas. “Trading Volume: Analysis and Interpretation”. In: *Investopedia* (2023). URL: https://www.investopedia.com/ask/answers/041015/why-trading-volume-important-investors.asp.

[Nna24] Eugene Nnamdi. “Top Sources for OHLC Candlestick Data in the Crypto Market”. In: *Coinmonks* (2024). URL: https://medium.com/coinmonks/top-sources-for-ohlc-candlestick-data-in-the-crypto-market-bd5111cb6340.

[NSK22] Sidra Naveed, Gunnar Stevens, and Dean-Robin Kern. “Explainable Robo-Advisors: Empirical Investigations to Specify and Evaluate a User-Centric Taxonomy of Explanations in the Financial Domain”. In: *IntRS@RecSys*. 2022. URL: https://api.semanticscholar.org/CorpusID:252782003.

[Ore+20] Boris N. Oreshkin et al. *N-BEATS: Neural basis expansion analysis for interpretable time series forecasting*. 2020. URL: https://arxiv.org/abs/1905.10437.

[Pat+20] Pratik Patil et al. “Stock Market Prediction Using Ensemble of Graph Theory, Machine Learning and Deep Learning Models”. In: Association for Computing Machinery, 2020. URL: https://doi.org/10.1145/3378936.3378972.

[Ped+11] Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2011. URL: https://scikit-learn.org/stable/index.html.

[Phu+24] Tran Phuoc et al. “Applying machine learning algorithms to predict the stock price trend in the stock market – The case of Vietnam”. In: *Humanities and Social Sciences Communications* 11.1 (2024). DOI: 10.1057/s41599-024-02807-x. URL: https://doi.org/10.1057/s41599-024-02807-x.

[PJ16] Nada Petrusheva and Igor Jordanoski. “Comparative Analysis Between the Fundamental and Technical Analysis of Stocks”. In: *Journal of Process Management and New Technologies* (2016). URL: https://doi.org/10.5937/JPMNT1602026P.

[PM23] Andy Patrizio and John Moore. “IBM (International Business Machines Corporation)”. In: *TechTarget* (2023). URL: https://www.techtarget.com/searchitchannel/definition/IBM-International-Business-Machines.

[Qua24] Quantitativo. *A different indicator: Using an unusual indicator to inform a strategy that delivers 21% annual returns since 2004*. 2024. URL: https://www.quantitativo.com/p/a-different-indicator.

[Sau11] Jeff Sauro. *Measuring Usability with the System Usability Scale (SUS)*. 2011. URL: https://measuringu.com/sus.

[Sir+19] Naadun Sirimevan et al. “Stock Market Prediction Using Machine Learning Techniques”. In: *2019 International Conference on Advancements in Computing (ICAC)*. 2019. URL: https://ieeexplore.ieee.org/document/9103381.

[SK21] Jaideep Singh and Matloob Khushi. “Feature Learning for Stock Price Prediction Shows a Significant Role of Analyst Rating”. In: *Applied System Innovation* 4.1 (2021). DOI: 10.3390/asi4010017. URL: https://www.mdpi.com/2571-5577/4/1/17.

[SMM23] Simon Schreibelmayr, Laura Moradbakhti, and Martina Mara. “First impressions of a financial AI assistant: differences between high trust and low trust users”. In: *Frontiers in Artificial Intelligence* Volume 6 - 2023 (2023). DOI: 10.3389/frai.2023.1241290. URL: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2023.1241290.

[Sta24] StatMuse. *When did International Business Machines go public?* 2024. URL: https://www.statmuse.com/money/ask/when-did-international-business-machines-go-public.

[Sub+21] Abdulhamit Subasi et al. “Stock Market Prediction Using Machine Learning”. In: *Procedia Computer Science* (2021). URL: https://www.sciencedirect.com/science/article/pii/S1877050921021128.

[Swa+25] Ashok Kumar Swami et al. “Artificial intelligence technology in materials selection, device engineering and parameter optimisation for triboelectric nanogenerator”. In: *Materials Today Communications* 46 (2025), p. 112553. DOI: https://doi.org/10.1016/j.mtcomm.2025.112553. URL: https://www.sciencedirect.com/science/article/pii/S2352492825010657.

[TNP23] Quy Tran Van, Tram Nguyen Bao, and Tu Pham Minh. “Integrated Hybrid Approaches for Stock Market Prediction with Deep Learning, Technical Analysis, and Reinforcement Learning”. In: Association for Computing Machinery, 2023, pp. 213–220. DOI: 10.1145/3628797.3629018. URL: https://doi.org/10.1145/3628797.3629018.

[Vel23] Cris Velasquez. “Mining Patterns in Stocks with PCA and DTW”. In: *Medium* (2023). URL: https://medium.com/@crisvelasquez/mining-patterns-in-stocks-with-pca-and-dtw-e98651657f37.

[Vir13] Agnes Virlics. “Investment Decision Making and Risk”. In: *Procedia Economics and Finance* (2013). URL: https://www.sciencedirect.com/science/article/pii/S2212567113001299.

[VV14] Javier Vidal-García and Marta Vidal. *Is Your Fund Watching Out for You?* 2014. URL: https://ssrn.com/abstract=2230247.

[Wan+19] Zirui Wang et al. “Characterizing and Avoiding Negative Transfer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2019. URL: https://openaccess.thecvf.com/content_CVPR_2019/html/Wang_Characterizing_and_Avoiding_Negative_Transfer_CVPR_2019_paper.html.

[WL21] Tong Wang and Qihang Lin. “Hybrid Predictive Models: When an Interpretable Model Collaborates with a Black-box Model”. In: *Journal of Machine Learning Research* 22.137 (2021), pp. 1–38. URL: http://jmlr.org/papers/v22/19-325.html.

[Woj+20] H. M. Wojton et al. “Initial validation of the trust of automated systems test (TOAST)”. In: *Journal of Social Psychology* 160.6 (2020), pp. 735–750. URL: https://testscience.org/measuring-user-trust-in-systems.

[XL24] Yiqiong Xue and Xiaodong Liu. “An Approach for Stock Modeling and Prediction Using Latency-based HMM and Bayesian Network”. In: (2024). URL: https://doi.org/10.1145/3695719.3695725.

[Yan25] Aixiang Yang. “Big data-driven corporate financial forecasting and decision support: a study of CNN-LSTM machine learning models”. In: *Frontiers in Applied Mathematics and Statistics* 11 (2025). DOI: 10.3389/fams.2025.1566078. URL: https://www.frontiersin.org/articles/10.3389/fams.2025.1566078.

[Yeo+25] Wei Jie Yeo et al. “A comprehensive review on financial explainable AI”. In: *Artificial Intelligence Review* 58 (2025), p. 189. DOI: 10.1007/s10462-024-11077-7. URL: https://doi.org/10.1007/s10462-024-11077-7.

[ZE17] Xiao Zhong and David Enke. “Forecasting daily stock market return using dimensionality reduction”. In: *Expert Systems with Applications* 67 (2017), pp. 126–139. DOI: https://doi.org/10.1016/j.eswa.2016.09.027. URL: https://www.sciencedirect.com/science/article/pii/S0957417416305115.

[Zhe+24] Jiajian Zheng et al. “The Random Forest Model for Analyzing and Forecasting the US Stock Market in the Context of Smart Finance”. In: *arXiv preprint arXiv:2402.17194* (2024). URL: https://arxiv.org/abs/2402.17194.

[Zuc19] Gregory Zuckerman. *The Man Who Solved the Market: How Jim Simons Launched the Quant Revolution.* Penguin Press, 2019.