
UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# Investigation of the range of variation in the future price of securities traded on the stock exchange, using stock market indicators and artificial intelligence tools.

Diploma Thesis

**Konstantinou Efstathios**

**Supervisor:** Thanos Georgios

June 2025


UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

# Investigation of the range of variation in the future price of securities traded on the stock exchange, using stock market indicators and artificial intelligence tools.

Diploma Thesis

**Konstantinou Efstathios**

**Supervisor:** Thanos Georgios

June 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

# Διερεύνηση του εύρους μεταβολής της μελλοντικής τιμής αξιογράφων που διαπραγματεύονται στο χρηματηστήριο αξιών, με τη βοήθεια χρηματηστηριακών δεικτών και εργαλείων τεχνητής νοημοσύνης.

Διπλωματική Εργασία

**Κωνσταντίνου Ευστάθιος**

**Επιβλέπων/πουσα:** Θάνος Γεώργιος

Ιούνιος 2025

Approved by the Examination Committee:

Supervisor **Thanos Georgios**
Laboratory Teaching Staff member, Department of Electrical and Computer Engineering, University of Thessaly

Member **Tousidou Eleni**
Laboratory Teaching Staff member, Department of Electrical and Computer Engineering, University of Thessaly

Member **Fevgas Athanasios**
Laboratory Teaching Staff member, Department of Electrical and Computer Engineering, University of Thessaly

# Acknowledgements

I would like to express my deepest gratitude to my family for their unwavering support, encouragement, and understanding throughout the course of my studies. Their belief in me has been a constant source of motivation.

I am also sincerely thankful to my friends for offering their encouragement, insight, and companionship during the challenges of this journey.

Finally, I would like to thank my supervisor, Professor Georgios Thanos, for his invaluable guidance, constructive feedback, and continuous support. His expertise and mentorship have been instrumental in shaping the direction and quality of this thesis.

# DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Konstantinou Efstathios

Diploma Thesis

**Investigation of the range of variation in the future price of securities traded on the stock exchange, using stock market indicators and artificial intelligence tools.**

**Konstantinou Efstathios**

# Abstract

This thesis presents a comparative analysis of models for next-day stock price direction, reflecting the complexity of financial markets. It evaluates statistical approaches (ARIMA with GARCH), classical machine learning (SVM), and deep learning architectures (LSTM, Transformer, and Kolmogorov-Arnold Networks). Using U.S. equity and SP 500 data, it examines the impact of feature selection (extensive vs. curated indicators) and architectural design.

Results show that directional prediction remains highly challenging, with most models offering limited improvements over chance. Feature selection emerged as a key factor—smaller, curated sets often matched or outperformed larger ones while improving efficiency. ARIMA models benefited from rolling forecasts and volatility modeling. SVMs performed near random. LSTMs and Transformers showed convergence and modest gains, with a tendency toward upward prediction. KANs, especially wider variants with selected features, showed potential but limited discrimination.

The study offers a benchmark for forecasting methods and emphasizes the critical role of feature selection in financial prediction tasks.

## Keywords:

Stock Market Prediction, Financial Forecasting, Machine Learning, Deep Learning, Time Series Analysis, ARIMA, SVM, LSTM, Transformer, Kolmogorov-Arnold Networks (KAN), Feature Engineering, Technical Indicators, Algorithmic Trading.

Διπλωματική Εργασία

**Διερεύνηση του εύρους μεταβολής της μελλοντικής τιμής αξιογράφων που διαπραγματεύονται στο χρηματηστήριο αξιών, με τη βοήθεια χρηματηστηριακών δεικτών και εργαλείων τεχνητής νοημοσύνης.**

**Κωνσταντίνου Ευστάθιος**

# Περίληψη

Η παρούσα διπλωματική εξετάζει συγκριτικά μοντέλα πρόβλεψης της κατεύθυνσης της τιμής μετοχών για την επόμενη ημέρα, εστιάζοντας στην πολυπλοκότητα των χρηματοοικονομικών αγορών. Αναλύονται στατιστικές μέθοδοι (ARIMA με GARCH), αλγόριθμοι μηχανικής μάθησης (SVM) και σύγχρονα βαθιά νευρωνικά δίκτυα (LSTM, Transformer και Kolmogorov-Arnold Networks - KANs).

Με βάση δεδομένα από μετοχές των ΗΠΑ και τον δείκτη S&P 500, αξιολογείται η ακρίβεια πρόβλεψης, ο ρόλος της επιλογής χαρακτηριστικών (εκτενής vs. επιλεγμένοι τεχνικοί δείκτες) και η επίδραση της αρχιτεκτονικής των μοντέλων. Τα αποτελέσματα δείχνουν ότι η πρόβλεψη κατεύθυνσης είναι ιδιαίτερα απαιτητική, με τις περισσότερες μεθόδους να ξεπερνούν οριακά την τυχαιότητα. Η επιλογή λίγων, στοχευμένων χαρακτηριστικών βελτιώνει συχνά την απόδοση και μειώνει τον υπολογιστικό φόρτο.

Η μελέτη προσφέρει ένα χρήσιμο σημείο αναφοράς για μεθόδους πρόβλεψης και υπογραμμίζει τη σημασία της σωστής επιλογής χαρακτηριστικών στις χρηματοοικονομικές εφαρμογές.

**Λέξεις-κλειδιά:**

Πρόβλεψη Χρηματιστηριακής Αγοράς, Χρηματοοικονομική Πρόβλεψη, Μηχανική Μάθηση, Βαθιά Μάθηση, Ανάλυση Χρονοσειρών, ARIMA, SVM, LSTM, Transformer, Δίκτυα Kolmogorov-Arnold (KAN), Τεχνικοί Δείκτες, Αλγοριθμικές Συναλλαγές.

# Table of contents

# List of figures

# List of tables

# Abbreviations

| | |
|---|---|
| ACF | Autocorrelation Function |
| AIC | Akaike Information Criterion |
| AR | Autoregressive |
| ARFIMA | Autoregressive Fractionally Integrated Moving Average |
| ARIMA | Autoregressive Integrated Moving Average |
| ARIMAX | Autoregressive Integrated Moving Average with exogenous variables |
| BIC | Bayesian Information Criterion |
| BiLSTM | Bidirectional Long Short-Term Memory |
| DCT | Deep Convolutional Transformer |
| DTNN | Differential Transformer Neural Network |
| EMH | Efficient Market Hypothesis |
| FATE | Feature-Axis Transformer |
| FastKAN | Fast Kolmogorov-Arnold Network |
| GANN | Genetic Algorithm Neural Network |
| GARCH | Generalized Autoregressive Conditional Heteroskedasticity |
| GDP | Gross Domestic Product |
| I | Integrated |
| KAF | Kolmogorov-Arnold Fourier Networks |
| KAN-ODEs | Kolmogorov-Arnold Network Ordinary Differential Equations |
| KANOP | KAN-based Option Pricing |
| KANs | Kolmogorov-Arnold Networks |
| KKANs | Kurkova-Kolmogorov-Arnold Networks |
| LOBs | Limit Order Books |
| LSTM | Long Short-Term Memory |
| MA | Moving Average |

| | |
|---|---|
| MACD | Moving Average Convergence Divergence |
| MAPE | Mean Average Percentage Error |
| MLPs | Multi-Layer Perceptrons |
| OHLC | Open, High, Low, and Close |
| OHLCA | Open, High, Low, Close, and Adjusted Close |
| PACF | Partial Autocorrelation Function |
| RBF | Radial Basis Function |
| RKANs | Recurring Kolmogorov-Arnold Networks |
| RMSE | Root Mean Square Error |
| RSI | Relative Strength Index |
| SARIMA | Seasonal Autoregressive Integrated Moving Average |
| SARIMAX | Seasonal Autoregressive Integrated Moving Average with exogenous variables |
| SVM | Support Vector Machine |
| TCN | Temporal Convolutional Networks |
| TKANs | Temporal Kolmogorov-Arnold Networks |
| TLOB | Transformer Limit Order Book |
| VIX | CBOE Volatility Index |
| Wav-KAN | Wavelet Kolmogorov-Arnold Networks |

# Chapter 1

# Introduction

The guessing of moves in the stock market has long been a fascinating and expert-driven preoccupation for researchers, investors, and financial analysts alike. The practical appeal of being correct in such matters lies in the potential gains that can be used to hedge against the risk of loss—gains that correlate directly with one's learning curve on the subject. Financial markets, particularly equity markets, exhibit the characteristics of complex and chaotic systems driven by various forces such as macroeconomic conditions, corporate earnings, political developments, the overall emotional state of the market, and the human brain's automatic processing of information. As a result, stock price time series display high volatility, non-stationarity, noise, and nonlinear complexity, making them especially challenging for empirical predictive modeling.

The science and practice of achieving reliable stock price predictions have led to the implementation of a wide array of methods. In the early stages, statistical time series models dominated, focusing on uncovering linear correlations and trends. However, the limitations of these models in addressing the nonlinear and chaotic nature of financial data prompted the adoption of machine learning techniques. These methods range from basic algorithms to highly advanced deep learning models designed to learn complex patterns directly from historical data. Despite advances in computing power and algorithmic sophistication over the past few decades, no single model has emerged with statistically significant evidence proving its ability to consistently predict stock price movements. This is partly due to market efficiency and the constantly evolving dynamics of finance, which continue to impede the effectiveness of predictive models. This work seeks to follow the path of inference that has led to these applications, introducing a novel approach based on a set of contemporary predictive

models.

## 1.1 Object of the Thesis

This thesis adopts a multifaceted approach to examining various statistical and modern machine learning models used to predict next-day stock price directions. The primary aim is to conduct a comparative analysis of a wide range of statistical, machine learning, and deep learning models, spanning from traditional time series approaches to advanced artificial intelligence techniques.

Specifically, this research seeks to answer the following core questions:

1. **Comparative Model Performance:** Systematically evaluate and compare the predictive performance of a broad array of models, including traditional time series methods (ARIMA with GARCH extensions), classical machine learning algorithms (Support Vector Machines), and several state-of-the-art deep learning architectures (Long Short-Term Memory networks, Transformer models, and the novel Kolmogorov-Arnold Networks).

2. **Impact of Feature Engineering:** Investigate the critical role of feature engineering by evaluating how different feature selection strategies—specifically, the use of a comprehensive set of engineered technical indicators versus a curated subset of high-importance features—affect the performance and robustness of the predictive models.

3. **Influence of Architectural Choices and Optimization:** Analyze how architectural configurations (e.g., network depth, attention mechanisms, kernel types) and hyperparameter optimization techniques influence the predictive accuracy and generalization capabilities of each model family.

4. **Stock-Specific Predictability:** Examine whether the accuracy and efficiency of stock price movement predictions vary across different financial instruments, sectors, and market categories.

To achieve these objectives, a robust methodology is employed. A comprehensive dataset encompassing selected U.S. equities—depicting daily price and volume movements—as well as the S&P 500 index is used. Each model family undergoes a series of carefully designed

experiments, analyzing various configurations and feature sets, as outlined in Chapter 3. Following this experimentation phase, the models are evaluated using a comprehensive set of classification metrics including accuracy, F1-score, precision, recall, and AUC-ROC, as detailed in Chapter 4. The goal is to provide insight into how different models are applied in real-world financial contexts, to assess their respective strengths and weaknesses in this complex domain.

### 1.1.1 Contribution

This thesis makes the following contributions:

1. A thorough literature review of financial time series analysis and an overview of stock prediction models—ranging from traditional statistical methods to modern deep learning architectures—was conducted (Chapter 2).

2. A clear and replicable methodology for data collection and preprocessing was developed, along with a rigorous feature engineering process tailored for stock market prediction tasks (Chapter 3).

3. A series of systematic experiments were carried out using five types of predictive models: Autoregressive Integrated Moving Average (ARIMA) models (including GARCH extensions), Support Vector Machines (SVM), Long Short-Term Memory (LSTM) networks, Transformer models, and Kolmogorov-Arnold Networks (KAN). These experiments included various architectural settings, feature sets (all engineered features vs. a selected subset of 25 important features), and hyperparameter optimization strategies (Chapter 3).

4. A transparent and rigorous cross-comparative study was conducted, evaluating the deployed models using several widely accepted performance metrics (accuracy, F1-score, precision, recall, AUC-ROC). These evaluations assessed their effectiveness in predicting next-day price directions for 11 financial instruments (Chapter 4).

5. The effects of specific feature selection strategies and hyperparameter optimization techniques on model performance and stability were thoroughly examined and quantified (Chapters 3 and 4).

6. Significant findings were achieved concerning the relative strengths and weaknesses of each model type, the differing predictability of certain stocks, and the practical implications of these insights for financial forecasting (Chapter 4)

## 1.2 Organization of the Volume

The remainder of this thesis is organized as follows.

- **Chapter 2** presents a review of previous research relevant to the subject of this thesis, covering foundational concepts in financial time series analysis and an overview of various modeling techniques employed for stock market prediction.
- **Chapter 3** details the methodology employed in this study. This includes the formulation of the prediction problem, the processes for data collection and preprocessing, the techniques used for feature engineering and selection, and the specific implementation details and experimental design for each of the predictive models evaluated.
- **Chapter 4** presents and analyzes the experimental results obtained from the application of the models described in Chapter 3. This chapter discusses the performance of each model family, compares different architectural and feature set configurations, and highlights key findings and stock-specific patterns.
- **Chapter 5** summarizes the main contributions and findings of the thesis, discusses the limitations encountered during the research, and proposes potential directions for future work in this challenging and dynamic field.

# Chapter 2

# Previous research overview

## 2.1 Introduction

In this chapter we will perform a literature review at the history of stock prediction models ranging from traditional statistical methods to, more recently, deep learning architecture. It has been found in the studies that the level of complexity and the possibilities of the models developed were continuously higher, and every new generation was able to correct the shortcomings of former models and at the same time, bring in new challenges. Traditional time-series models like ARIMA give a clear report but are not viable for complex, nonlinear market dynamics. Solutions using machine learning such as SVM and Random Forest help to manage nonlinearity better but sometimes they are really need good feature extractors.

Deep Neural Networks architectures specifically LSTMs have proved a much better performance in the time-series data of stock than their predecessors, even the new transformer-based models have shown the power of attention mechanisms. The still-evolving Kolmogorov-Arnold Networks break through the potential of Logic by using their learnable activation functions. Despite all of these developments, the field is volunteering to offer a set of multidimensional models for financial time series prediction, and thus proving the complexity of the financial market and the persistent research gap that continues to be a challenge for disruptive researchers.

## 2.2 Foundational Concepts

This part contains necessary background that helps to understand the following topics on various stock price prediction models. In this section, we will discuss financial time series data, common data types, data challenges, as well as machine learning principles relevant to applications in this area.

### 2.2.1 The Nature of Financial Time Series Data

A time series in finance is a series of data points indexed along the time line. The data points can represent elements such as daily stock closing prices, hourly trading volumes, or minute-by-minute market indices. Financial time series have some prominent features that complicate their modeling, including:

- **Non-Stationarity:** A large share of financial time series are non-stationary, which implies that their statistical properties, like the mean and variance, vary over time. They are often characterized by trends (e.g., upward, downward, or sideways movements) and seasonality (cyclical patterns occurring at fixed intervals). Stationarity is often a preferred property for many statistical models, and differencing (calculating the difference between consecutive observations) is one of the techniques employed to transform a non-stationary series into a stationary one.

- **Volatility Clustering:** This refers to the tendency for periods of high price fluctuation (high volatility) to be followed by more periods of high volatility, and similarly, periods of low price fluctuation (low volatility) to be followed by more periods of low volatility

- **Noise:** Financial markets are inherently noisy. Therefore, underlying patterns can be obscured by significant random, unpredictable movements. It is often the case that the signal-to-noise ratio is considerably low in financial data, making it a challenge to separate real trends from random fluctuations

- **Long-Range Dependencies:** Sometimes shocks or events that took place in the distant past can have a residual impact on current and future price dynamics.

- **Fat Tails (Leptokurtosis):** The distribution of financial returns frequently exhibits ”fat tails,” which means that extreme price variations (both large profits and large losses) occur more frequently than would be expected in a normal (Gaussian) distribution .

- **Autocorrelation:** This is the correlation of the time series with lagged versions of itself. Significant autocorrelation at different lags may suggest the presence of predictable patterns.

### 2.2.2 The Objective: Predicting Stock Market Dynamics

The main goal in stock market prediction can vary, but most commonly it involves predicting future market behavior. Common prediction tasks include:

- **Directional Prediction:** Forecasting the direction in which the price will move (e.g., up, down, or sideways/neutral). This is often framed as a classification task.
- **Price Level/Range Prediction:** Forecasting the actual future price level of a security or a probable range within which the price will fall. This is typically a regression task.
- **Volatility Prediction:** Forecasting the magnitude of future price changes, which is essential for risk management and option pricing.

The task of stock market prediction is known to be very challenging, partly because of the concepts encapsulated in the Efficient Market Hypothesis (EMH). The EMH, in its different versions, asserts that asset prices fully reflect all available information, which implies that it is not feasible to consistently achieve risk-adjusted excess returns (”beat the market”) on the basis of historical data alone. While the strict form of EMH is debated, it nevertheless stresses the intrinsic difficulty of financial prediction .

### 2.2.3 Evaluation of the Stock Prediction Model

Evaluating a stock-prediction model—i.e., measuring its performance—depends on whether the task is regression or classification.

**Regression Tasks**

- **Mean Squared Error (MSE)**: The average of the squared differences between predicted values and actual values.
- **Root Mean Squared Error (RMSE)**: The square root of MSE, expressed in the same units as the target variable.

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

- **R-squared ($R^2$)**: The proportion of variance in the dependent variable that is explained by the independent variables in the model.

**Classification Tasks**

- **Accuracy**: The ratio of correctly predicted instances to the total number of instances.

- **Precision**: The ratio of true positive predictions to all positive predictions made by the model.

- **Recall (Sensitivity)**: The ratio of true positive predictions to all actual positive instances (i.e., the model's ability to find every positive case).

- **F1-Score**: The harmonic mean of precision and recall, balancing both—especially useful for imbalanced datasets.

- **Confusion Matrix**: A table showing counts of true positives, true negatives, false positives, and false negatives.

- **AUC-ROC**: Area under the receiver operating characteristic curve, measuring the classifier's ability to distinguish between classes across all thresholds.

**Financial and Robustness Considerations**

Beyond these standard metrics, financial applications often assess profitability directly—e.g., by simulating trading based on model predictions—and use risk-adjusted return measures such as the *Sharpe ratio* or *maximum drawdown* (the largest peak-to-trough decline over a given period).

Robust out-of-sample evaluation is essential: performance should be tested on data not used during either training or validation to gauge the model's true predictive power.

The work of choosing, transforming, and creating input variables from original data is called **feature engineering**, which is an important step in developing effective prediction models.

### 2.2.4 General Modeling Paradigms

Two main frameworks are often used to predict the stock market:

- **Statistical Models:** These models are usually built on established statistical theories and make assumptions about the data-generating process, such as linearity, stationarity, or specific distributions. The Autoregressive Integrated Moving Average (ARIMA) model, which will be discussed in Section 2.3 , is one of the best examples.

- **Machine Learning (ML) Models:** These models learn patterns directly from data by identifying signals, rather than relying on explicit programming of human-derived rules. They are generally more effective at describing complex, non-linear dynamics.

  - **Supervised Learning:** This is the most commonly used method in stock prediction. The model learns from labeled data, in which every input (features) is tied to a known output (e.g., future price direction or level). This category includes models such as Support Vector Machines (Section 2.4), LSTMs (Section 2.5), Transformers (Section 2.6), and KANs (Section 2.7).

  - **Unsupervised Learning:** This is used for tasks like clustering similar stocks or anomaly detection, without pre-defined labels.

  - **Reinforcement Learning:** This involves an agent learning to make optimal trading decisions by interacting with the market environment and receiving rewards or penalties based on its actions.

### 2.2.5 Fundamental Concepts for Machine Learning on Time Series Data

A number of basic concepts from machine learning are paramount when applying ML models to time series data (e.g., stock prices):

#### Training, Validation, and Test Sets

Data used for constructing and evaluating ML models are split into three sets:

- **Training Set:** Used to fit the model to the observations, enabling it to learn underlying patterns.

- **Validation Set:** Used to tune hyperparameters (which are set prior to training) and to perform intermediate evaluations to prevent overfitting.
- **Test Set:** Used only once, after training and validation, to assess generalization performance on data never seen by the model.

For time series data, these splits must be made chronologically (e.g., earliest data for training, more recent data for validation, latest data for testing) to avoid data leakage. Walk-forward validation or rolling-window approaches are preferred for more robust evaluation.

#### Model Parameters vs. Hyperparameters

- **Parameters:** Values that the model learns automatically during training (e.g., weights in a neural network).
- **Hyperparameters:** Settings configured *before* training that control the learning process itself (e.g., learning rate, number of layers, SVM kernel type).

#### Overfitting and Underfitting

- **Overfitting:** Occurs when a model learns noise and idiosyncrasies of the training data, resulting in poor generalization to new data.
- **Underfitting:** Happens when a model is too simple to capture essential patterns, yielding poor performance on both training and test data.

#### Regularization Techniques

Techniques such as L1 or L2 regularization and dropout are commonly used to reduce overfitting by penalizing complexity or randomly omitting units during training.

## 2.3 ARIMA model

The Autoregressive Integrated Moving Average (ARIMA) model is one of the most traditional and widely used time series frameworks. It is a very useful statistical model for understanding, modeling, and predicting time-dependent data, including financial time series. After the appearance of more complex machine learning techniques, ARIMA models are still

valuable because of their clarity, theoretical basis, and capacity to describe linear temporal dependencies.

### 2.3.1 Theoretical Foundation of ARIMA Models

#### Core Components of ARIMA

The ARIMA model, expressed as ARIMA(p,d,q), integrates three central components that collaboratively build the time series data model [1]. These components address different temporal patterns:

1. **Autoregressive (AR) Component**: The AR(p) terms determine the relation between the current observation and a specified number (p) of lagged observations. This component displays the influence of the past values on the present ones, which in fact means regressing on these values and modeling the ”memory” of the process [2]. The formula that gives the joint AR(p) process is:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t$$

   Where $Y_t$ is the current value, $Y_{t-i}$ are lagged values, $\phi_i$ are the autoregressive parameters, and $\varepsilon_t$ is white noise.

2. **Integrated (I) Component**: The I(d) term marks the order of differencing required to convert a non-stationary series to a stationary one [3]. This process of differencing seeks to eliminate trends and stabilize the mean by calculating the difference between consecutive observations. Thus, the parameter d gives the number of times the differencing has to be applied.

3. **Moving Average (MA) Component**: The MA(q) term determines the current observation in terms of previous error terms [2]. As this term models the noise of the random shocks in the series, the general model of the MA(q) process is:

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

   Where $\mu$ is the mean of the series, $\varepsilon_t$ is the current error term, $\varepsilon_{t-i}$ are previous error terms, and $\theta_i$ are the moving average parameters.

### 2.3.2 Stationarity and Model Identification

The stationarity concept is the cornerstone of time series analysis, as something statistical can only be stationary if its statistical properties like mean, variance, and autocorrelation structure do not change over time [3]. Financial time series are usually non-stationary, meaning they exhibit trends or have changing variance, which should be handled before one can apply ARIMA.

The Box-Jenkins method is a systematic approach to model identification by ARIMA with four major steps [3]:

1. **Identification**: Evaluating the autocorrelation function (ACF) and partial autocorrelation function (PACF) to get the values for p, d, and q [4]. The ACF is the measure of the correlation at different time lags while the PACF is the correlation measure between the observations after the effects of intermediate observations have been excluded.

2. **Estimation**: This is the process by which the optimal parameters of the model structure that has been identified are determined.

3. **Diagnostic checking**: This involves validating the model through residual analysis, which ensures that the residuals resemble white noise [4].

4. **Forecasting**: This is where the model that has been fitted is used for making predictions.

Model selection is mostly based on information criteria like Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) that compare model fit with the number of parameters [5, 6].

### 2.3.3 Key Assumptions of ARIMA Models

There are several important assumptions for ARIMA models:

1. The time series must be (or be transformed to be) stationary after appropriate differencing.

2. The model's residuals must be independently and identically distributed (white noise).

3. It is assumed that the underlying process is linear, which means future values are a linear function of past values and random errors.

4. The model's parameters are supposed to stay the same over time.

These assumptions point out both the pros and cons of ARIMA models in complex economic scenarios.

### 2.3.4 Major ARIMA Variants in Financial Time Series

**Seasonal ARIMA (SARIMA)**

Regular fluctuations that happen at the same intervals are common features of many financial time series. The Seasonal ARIMA (SARIMA) model is the first of its kind to augment the pure ARIMA structure so as to cover these cyclical patterns [7, 6]. A typical format for a SARIMA model is SARIMA(p,d,q)(P,D,Q)s, where:

- (p,d,q) are the non-seasonal parameters
- (P,D,Q) are the seasonal parameters
- s is the length of the seasonal cycle

SARIMA has shown effectiveness in capturing periodic market behaviors as demonstrated in its applications to the Nifty 50 stock market index [6]. By explicit account of seasonality, the SARIMA model should be more accurate than the ARIMA model in making predictions of the future market. The literature indicates it can reduce Mean Average Percentage Error MAPE by more than 4% in some cases when seasonality is properly modeled [8].

### 2.3.5 ARIMAX and SARIMAX: Incorporating Exogenous Variables

Financial markets are affected not only by their past prices but also by many other external factors. Thus, beyond the standard models, (Seasonal) Autoregressive Integrated Moving Average with exogenous variables is introduced herein: ARIMAX (exogenous) and SARIMAX (Seasonal +exogenous) [8].

The SARIMAX model is upgraded to one that reflects the incorporation of exogenous factors to improve the accuracy of the target variable [8]. For instance, in the case of the stock market, these factors can be:

- Macroeconomic variables (GDP, interest rates, inflation)

- Market sentiment indicators
- Technical indicators
- Related market indices
- News sentiment scores

By incorporating external variables, SARIMAX is able to discover relationships that the traditional ARIMA model would miss as it focuses exclusively on the time series of interest.

### 2.3.6 ARFIMA: Addressing Long Memory Processes

Time series statistics commonly target long memory phenomena through the relationships between the variables where the autocorrelation decays slowly over time and so, this is not captured properly by the ARIMA model. For instance, Autoregressive Fractionally Integrated Moving Average (ARFIMA) model is a solution to this. This is achieved through fractional differencing [1].

The ARFIMA model is quite adequate for the production of ”red noise” and for long-term dependencies in financial data, where the autocorrelation function follows a much slower decay than the exponential decay usually found in standard ARIMA models [1]. Therefore, ARFIMA may be more suitable to explore these persistent market behaviors which the standard ARIMA may overlook.

### 2.3.7 Application of ARIMA Models in Stock Market Prediction

#### Empirical Studies and Performance

The stock market is the area where ARIMA models get the widest application. Research outcomes point to the use of ARIMA modeling for the prediction of the Bucharest Stock Exchange financial companies’ closing prices, as well as for the Nifty 50 Indian index’s trajectories [5, 6].

In the existing literature, the performance of these models has varied. When they were properly specified, ARIMA did manage to capture some key patterns in stocks and, therefore, provided quite accurate price predictions in the short term [5]. However, they are less effective in the long run due to the built-up errors and uncertain changes in the market.

The ARIMA performance in stock market forecasting depends mainly on the specific market characteristics and the time range studied. It has been observed that ARIMA models are better suited for markets with:

1. Lower volatility and fewer structural breaks
2. Strong seasonal or cyclical fluctuations
3. Inefficient market information

### 2.3.8 Strengths and Limitations in Market Contexts

The ARIMA model has several strengths in the field of stock market forecasting:

1. **Interpretability**: The coefficients have definite statistical meaning, making models transparent and easily understood.
2. **Statistical foundation**: They conform to reliable statistical principles and furnish standard methodologies for model establishment and testing.
3. **Parsimony**: In general, they can describe sophisticated structures with fewer parameters [9].
4. **Effectiveness for short-term forecasting**: They are usually good in the short term where factors remain quite stable.

Nevertheless, they possess certain shortcomings in financial market contexts:

1. **Linearity presumption**: However, the majority of financial markets are often driven by non-linear dynamics that cannot be directly captured by the ARIMA models [10].
2. **Poor handling of volatility clustering**: The unmodified ARIMA model does not factor in that financial data shows conditional heteroskedasticity.
3. **High sensitivity to structural breaks**: A regime shift in the market results in model performance being affected.
4. The ARIMA framework does not properly handle market sentiment and external shocks since it is based solely on the time series of interest.

5. The assumption of constant parameters: The model is empirical, and therefore, the relationships are expected to remain unchanged, while in a dynamic setting, this assumption may or may not hold.

### 2.3.9 Comparison with Other Forecasting Methods

The comparison of ARIMA with other forecasting methods brings out certain patterns:

1. **Against GARCH family models**: By contrast to ARIMA models, which target mean returns, GARCH models are mostly used to quantify volatility. The study suggests that the combination of ARIMA and GARCH (as in SARIMA-GARCH) is the best way to address both these aspects in financial time series [10].

2. **Versus Machine Learning Models**: Artificial networks and other machine learning models typically outperform ARIMA for complex and non-linear financial time series, particularly during periods of high volatility. However, during shorter forecasting horizons and in more stable market conditions, ARIMA models remain competitive.

3. **Versus Ensemble Methods**: ARIMA combined with other models often yields better predictions, as demonstrated by research that supports the application of mix-and-match and hybrid approaches [2, 9].

### 2.3.10 Practical Considerations in Implementation

#### Model Selection Challenges

When implementing ARIMA models for stock market prediction, a variety of practical problems can arise:

1. **Order selection**: Achieving correct values for p, d, and q remains partly subjective even though formal criteria can guide the selection. In practice, researchers prefer analyzing various models using criteria like AIC and selecting the most suitable one [5, 6].

2. **Complexity and parsimony**: The more complex the model is, the better it will fit the historically observed data. However, in this case, the model might perform worse in out-of-sample forecasting as a result of overfitting.

3. **Stationarity testing**: The finding of the appropriate order of differencing is made more difficult by potentially inconsistent results from various tests thus leading to further discussion of the tests [3].

4. Correctly identifying seasonal patterns in trading data is also difficult due to the presence of overlapping seasons of different lengths.

### 2.3.11 Data Preprocessing Requirements

The successful operation of ARIMA algorithms relies on prior meticulous handling of data:

1. Handling missing values and outliers: Financial time series typically contain gaps and extreme values which need proper treatment.

2. Transformations for variance stabilization: To deal with changing variance, log or Box-Cox transformations may need to be applied.

3. Differencing issue: It has to be determined whether to apply common differencing, seasonal differencing, or both, and how many times to apply each one.

4. Data blending: Identifying the right amount of data needed to train the model and the appropriate volume of out-of-sample data for tests, which should reflect different market conditions is critical.

### 2.3.12 Hybrid Models

The literature increasingly points toward hybrid approaches that combine ARIMA's strengths with complementary models:

1. ARIMA-GARCH combinations: These address both the mean and variance components of financial time series [10].

2. ARIMA with neural networks: Research suggests combining SARIMA with neural networks optimized through genetic algorithms (GANN) to handle both linear and nonlinear components of financial time series [7].

3. Combining forecasts: Rather than selecting a single ”best” model, averaging forecasts from multiple models often yields superior results a finding supported by empirical research in financial time series [9].

### 2.3.13 Methodological Innovations

Recent innovations are extending ARIMA’s capabilities for financial applications:

1. Improved interpolation techniques: Integrating methods like cubic spline interpolation with ARIMA can enhance short-term forecasting performance in stock markets [11].

2. Advanced parameter estimation: Bayesian methods and maximum likelihood variants provide more robust parameter estimates, especially for complex models.

3. Automated model selection: Machine learning approaches to automate the identification of appropriate ARIMA specifications can improve efficiency and reduce subjective judgment in model selection.

## 2.4 SVM

### 2.4.1 What is an SVM?

Support Vector Machine (SVM) is an extremely strong algorithm for machine learning that is usually used for supervised tasks related to finding classification boundaries or separating data. It is also possible to use it for regression tasks. The main task of SVM is to find a hyperplane that is optimally separating among the data points of different classes with the best margin. It is a concept that was introduced along with the Support Vector Machine algorithm that was built to find the best hyperplane that separates the data points of different classes while also maximizing the distance or margin between these classes [12] [13]. It is a supervised machine learning algorithm that can also be used for regression tasks.

SVM has its roots in the realization that a classification problem can be turned into an optimization problem. The practical implementation comes into play because SVM, given a set of labeled training data, generates hyperplanes in a possibly higher-dimensional space that best classify the data points [14]. This refers to a solution in which the hyperplane is located well away from the class closest to it, thus meaning better predictive power on samples from the distribution that were not presented during the learning stage [12].

### 2.4.2 The Maximal Margin Classifier

The margin in SVM terminology refers to the distance between the hyperplane and the closest data points from each class [12]. These closest points are known as support vectors, and they play a critical role in defining the position and orientation of the separating hyperplane [14].

The principle of maximal margin is central to SVM's effectiveness. By maximizing this margin, SVMs create a decision boundary that not only separates the training data accurately but also provides the greatest possible distance between classes [15]. This maximization is achieved by solving a quadratic optimization problem with constraints derived from the training data [12].

The importance of margin maximization goes beyond ensuring the correct classification of the training data alone. In general, a larger margin will lead to better generalization on unseen data because it forms a more robust separator whose parameters are less affected by noise and outliers in the training dataset [12]. This property provides added value in the

analysis of complicated, real-life datasets such as financial time series data.

### 2.4.3 Support Vectors

Support vectors are the main pieces in the design of an SVM, as they are the data points that are nearest to the decision boundary. They ”support” the optimal hyperplane, which is the reason for the term ”Support Vector” Machine [12].

The major concept behind the efficiency of SVMs is that only the support vectors are used in the final model, and not the entire dataset [14]. This property results in a computationally efficient model, especially for large datasets, as the complexity of the model depends on the number of support vectors rather than the dimensionality of the data [12].

Support vectors not only determine the distance at which the separating hyperplane is placed but also influence its direction. Any changes to support vectors will affect the decision boundary, while modifications to other data points generally will not impact the model as long as they remain outside the margin [16]. This capacity makes SVMs popular and practical for applications where outliers or points somewhat different from the rest exist.

### 2.4.4 How SVMs Work: Deeper Mechanics

#### Linearly Separable Data

In the case of a linearly separable dataset, the algorithm attempts to locate the hyperplane where the data points of different classes are perfectly separated while maintaining the maximum margin between them [12]. This hyperplane can be expressed mathematically as:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Where **w** is the normal vector to the hyperplane, **x** is an input vector, and $b$ is a bias term [12]. The optimization formulation for finding the parameters **w** and $b$ can be expressed as

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

with the following constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for all } i$$

Where $y_i \in \{-1, 1\}$ indicates the classes, while $(\mathbf{x}_i, y_i)$ are the training samples [14]. This optimization problem formulation aims to ensure that all the data points are assigned to the right side of the margin.

The procedure to optimize this formulation is to derive its dual form, through which the Lagrangian multipliers are calculated, which, for high-dimensional settings, leads to efficient computation [12].

### 2.4.5 Handling Non-Linearly Separable Data

#### Soft Margin Classification

In practice, data sets in finance and other fields are rarely linearly separable. In this case, SVMs are equipped with the soft margin classification concept, which allows a certain number of misclassifications to be made in order to obtain a better global fit [17].

The soft margin concept is achieved by slack variables $\xi_i \geq 0$ which quantify the error in the classification for each point. The optimization formula is hence as follows:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i$$

Subject to the following:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \text{ for all } i$$

$$\xi_i \geq 0 \text{ for all } i$$

The parameter $C$ regulates the balance between the maximization of the margin and the minimization of the classification error [17]. A larger $C$ favors correct classification of training examples, even if this results in a smaller margin. A smaller $C$ value allows more misclassifications, leading to a wider margin [12].

#### The Kernel Trick

The kernel trick is a concept that allows SVMs to effectively deal with non-linearly separable datasets through non-trivial transformations [17]. The kernel trick's essence is the overall mapping of the input data onto a higher-dimensional space, implicitly, into a space where the linear separation problem is manageable, without actually computing the coordinates in that space [18].

The kernel trick achieves this kind of transformation without the need for directly computing the transformation function $\phi$ [18]. In fact, the dot product $\mathbf{x}_i \cdot \mathbf{x}_j$ in the primal form of the SVM optimization problem is replaced with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ [12].

The kernel trick offers computational advantages. Instead of calculating coordinates in high-dimensional spaces, which may be excessively costly, the kernel function provides a straightforward way of calculating the inner product without explicitly computing the coordinates [12].

### 2.4.6 Common Kernel Functions

A number of kernel functions have been proposed for SVMs, each intended for specific problem areas and data types [17]:

**Linear Kernel**: The linear kernel is the most straightforward kernel function, given by: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$. This implies no transformation is carried out, making it suitable for data that can be separated linearly [12]. This function is computationally effective and performs well in cases where the number of features is large compared to the number of training instances.

**Polynomial Kernel**: The polynomial kernel, defined as $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)^d$, with the parameter $d$ being the polynomial degree, $\gamma$ the scaling factor and $r$ the constant term [17]. This kernel is particularly useful for capturing polynomial relationships in the data and is especially effective in situations where all training data is normalized.

**Radial Basis Function (RBF) Kernel**: The Radial Basis Function (RBF) kernel is another term for the Gaussian kernel, expressed as $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$ [18]. Models that use RBF as a kernel function are popular owing to their ability to address highly complex non-linear relationships. The parameter $\gamma$ determines the influence of individual training instances; larger values of $\gamma$ lead to a more complex decision boundary.

**Sigmoid Kernel**: This kernel is expressed as $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i \cdot \mathbf{x}_j + r)$ and is linked to neural networks while also featuring occasionally in multi-class classification scenarios [17]. However, this kernel is seldom used despite its connection to neural networks.

Choosing the right kernel function along with its hyperparameters is paramount to SVM performance and typically necessitates experimentation and validation on the specific dataset [18]. This selection has a direct impact on the model's ability to recognize patterns (whether linear or non-linear), particularly in complex tasks such as financial market analysis.

### 2.4.7 Positive and Negative Sides of SVMs

**Positive sides**

**Efficiency in High-Dimensional Spaces**: SVMs are efficient in high-dimensional spaces as their computational complexity depends primarily on the number of support vectors rather than the dimensionality of the space [19]. That is why they are perfect for financial applications with potential integration of numerous indicators and features.

**Memory Efficiency**: Deployment of SVMs is much more memory efficient as the final model relies only on a subset of the training data (the support vectors) [19]. This feature is particularly useful in resource-constrained environments or when handling large financial datasets.

**Versatility with Different Kernel Functions**: The availability of different kernel functions makes SVMs versatile, enabling them to adapt to different distributions and types of relationships in the data [18]. This helps financial practitioners who do not need to change the main algorithm but can tailor the model to the specificities of financial time series.

**Resistance to Overfitting**: The margin maximization principle intrinsic to SVMs serves as a regularization mechanism, thus helping to avoid overfitting, especially in high-dimensional spaces [18]. SVMs' stability is important when financial data is noisy, making it difficult to distinguish signal from noise.

**Statistical Underpinnings**: SVMs are based on statistical learning theory that gives them certain theoretical guarantees related to the generalization performance [12]. The strong mathematical support provided by it raises confidence in the predictive capabilities and the stability of the model.

**Negative sides**

**Computational Expense**: Training SVMs on large datasets can be very computationally expensive as the algorithm usually exhibits a time complexity ranging between $O(n^2)$ and $O(n^3)$, where n is the number of training samples [19]. This could pose an obstacle when dealing with vast historical financial data.

**Parameter Selection Sensitivity**: SVM performance is sensitive to kernel and hyperparameters such as the regularization parameter C and kernel-specific parameters like gamma in RBF kernels [18]. Finding appropriate parameters may involve time-consuming grid search

or cross-validation, thus contributing to computational costs.

**Reduced Interpretability**: Compared to decision trees or linear models, SVMs, especially those with non-linear kernels, are less interpretable [19]. This can be a challenge in finance to provide easy visualizations or explanations of the decision function in the transformed feature space.

**Performance Issues Due to Class Overlap**: When there is significant overlap between classes in the feature space, SVMs may struggle to find proper separating hyperplanes even with a soft margin [19]. Financial data often exhibits such characteristics due to the noise and complexity to which it is subject.

**Scaling Necessities**: SVMs are not scale-invariant, meaning that features should be normalized before the training process in order to prevent features with larger numeric ranges from dominating the outcome [19]. This necessitates careful preprocessing of financial indicators and measures.

### 2.4.8 SVMs in Financial Forecasting

The use of SVMs in financial time series forecasting is driven by multiple factors aligning with the peculiarities inherent to such data [20]. The high levels of noise and the presence of non-linear and non-stationary data make these series very difficult to model using traditional statistical methods [21].

SVMs offer a robust solution to the problems mentioned above due to their ability to capture the interdependencies among variables under these conditions [20]. Moreover, the margin maximization principle also enhances robustness against the noise inherent in financial data, while the use of the kernel trick enables the modeling of very complex patterns in stock price movements [22].

The high efficacy of SVM in high-dimensional spaces allows the adoption of a broad range of features (e.g., technical indicators) without excessive risk of overfitting the data [21]. This is particularly relevant in stock market forecasting where the number of potential price-affecting factors is large.

The use of the structural risk minimization principle by SVMs also aids in tackling the problem of generalization, which is important in financial forecasting where the goal is not merely to fit historical data but to make accurate predictions about future price directions [22].

### 2.4.9 Review of Key Studies and Methodologies

#### Input Features

The literature shows a diverse range of input features applied in stock market prediction models based on SVM [20] [21]. These include:

**Price history data**: Open, high, low, and close (OHLC) prices are primary inputs and often converted to returns or other normalized values to achieve stationarity [20].

**Technical indicators**: Common indicators such as Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), various moving averages, Bollinger Bands, and momentum indicators [21]. They are specifically designed to capture different trading behaviors such as trends, volatility, and momentum.

**Trading Volume**: Volume information often provides crucial context for changes in price and is frequently used as an input feature [20].

**Text-based features**: Recent studies have started including sentiment analysis from financial news, social media, and other textual sources [21]. One of the examples is the work by Kong et al. (2019), where the authors gathered 2.3 million news items spanning 2008 to 2015 to derive sentiment features for their SVM model that was predicting Chinese stock markets [21].

**Macroeconomic indicators**: Some models also include macroeconomic variables, such as interest rates, inflation rates, and economic growth rates, especially for longer-term forecasting horizons [22].

#### SVM Configurations

The literature shows several different SVM configurations for stock market prediction:

**Kernel selection**: The RBF kernel is very common in financial applications because of its ability to capture complex non-linear relationships [20] [21]. Some studies also use polynomial or linear kernels which have proven effective for specific markets and feature sets [22].

**Parameter optimization**: Grid search with cross-validation is the most common method used for tuning SVM hyperparameters (C and kernel-specific parameters) [22]. Some studies also used genetic algorithms for parameter optimization [22].

**Ensemble and Hybrid Approaches**: Other studies have explored ensemble methods that combine multiple SVMs or hybrid models involving SVMs [20]. By way of example, Nayak

et al. (2021) conducted comparative studies of several heterogeneous ensembles for financial forecasting that included combinations of SVM with LSTM and Linear Regression models [20].

### Performance Metrics

Research on SVM performance in stock market prediction usually reports:

**Classification accuracy**: The correct percentage of up/down movement predictions [20] [21].

**Precision, Recall and F1-Score**: Metrics that address the disparity in class distribution and different types of errors [20].

**Financial Performance Metrics**: Some studies convert predictive performance into financial metrics, such as annualized returns, Sharpe ratio or maximum drawdown to evaluate their practical applicability [22].

### Key Findings and Comparative Performance

The literature reports several significant results regarding SVM performance in stock market forecasting:

**Predictive Precision**: Reports indicate that SVM models generally achieve directional accuracy (up/down movements) of around 60%-75%, yet rates vary greatly depending on the market, time span, and set of features [20] [21].

**Comparison with Other Techniques**: In terms of performance with other machine learning algorithms, SVMs have often shown competitive performance. For instance, in one study by Nayak et al. (2021), it was observed that even though SVM scored lower than LSTM and Linear Regression in certain ensemble configurations, an ensemble of these models showed better results [20].

**Importance of Features**: Technical indicators and volume data have been frequently singled out as significant features across various studies [21]. Also, the incorporation of sentiment analysis in financial news has been reported to enhance prediction accuracy [21].

**Market Specifics**: The performance of different financial markets and indices indicates that SVMs often need to be tailored to the market's unique characteristics [21].

### 2.4.10 SVMs in Stock Market Prediction: Challenges and Limitations

Despite their positive performance, applying SVMs to stock market forecasting has some challenges and limitations:

**Concept drift**: Financial markets are dynamic, and the relationships between features and outcomes can change over time. This non-stationarity poses a major challenge to static SVM models [20].

**Feature Selection**: The application of the SVM model to the stock market is directly dependent on the quality and relevance of input features. Finding the most predictive features is still challenging due to the complexity of market dynamics and their evolving nature [21].

**Parameter Sensitivity**: The performance of SVM models depends heavily on parameter settings. Careful selection of parameters, which can be cumbersome due to the many available choices, is crucial [20].

**Data imbalance**: Imbalanced price movements (e.g., more up days than down days, or vice versa) can occur under certain market conditions, potentially biasing SVM predictions [20].

**Historical Pattern Overfitting**: There is a chance that SVMs will learn patterns that correlate with the historical data but do not persist in future market conditions [21].

## 2.5 LSTM

The use of LSTM networks has considerably increased in the prediction of stock market price movements. The methods of deep learning are widely employed and the Long Short-Term Memory (LSTM) networks have shown to be very effective among them. Besides, the review dives into the basic concepts of LSTM networks and their specific use for predicting the stock market directional movements. The literature illustrates that LSTM models are more efficient than conventional forecasting models as they can grasp intricate temporal patterns present in financial time series data. Meanwhile, the deficits remain in model interpretability, hyperparameter adjustment, and the gap between theoretical models in academics and actual trading practices. Through offering current insights related to the design, implementation, and performance metrics of LSTM, this review presents the advantages and drawbacks of these models in financial forecasting situations.

### 2.5.1 Introduction of the LSTM in the Recurrent Neural Networks (RNN) context.

The Recurrent Neural Networks (RNNs) are a kind of artificial neural networks that were designed very specifically to discern order within sequential data (e.g. time series, text, speech). In spite of being a type of feedforward neural network, RNNs are different because of their feedback connections. The resulting connection creates an internal state or memory allowing the RNNs to process the sequence of the rest of the input [23]. Just because of this architecture, RNNs are particularly useful for financial time series data, such as stock prices, where you definitely need temporal dependencies and patterns for predictions.

The original architecture of RNNs is theoretically capable of learning from sequential data; however, they face the problem of classical RNNs significantly with long-term dependencies. The major reason is the vanishing gradient problem, which means that gradients either get too small (vanish) or get too big (explode) when they are backpropagated through time [24]. This issue basically solidifies that the network is not able to learn about the connections that occur between long distances of time, which is especially hard for stock market predictions when the short term price movements are often influenced by the long-term trends.

The Long Short-Term Memory (LSTM) networks were developed as a particularly state-

of-the-art solution to tackle these restrictions. First created by Hochreiter and Schmidhuber in 1997, LSTMs are explicitly devised to solve the vanishing gradient problem by the implementation of special memory cells, which can store information for a long time and also forget it when necessary [25] [23]. As a result of this architecture at the LSTM level and the ability to keep long-range dependencies in the LSTM's vocabulary, they can be better than the any other machine learning models used in time series forecast like stock market [26].

**The Mechanism and Architecture of LSTMs**

The key part of the LSTM network is an especially structured cell because it is designed to hold excess information and at the same time, to dispose of it. Generally, an LSTM cell contains a cell state which passes through all the chain, with a few linear interactions, which means that the information is flowing with very few changes [23] [26]. This acts like a highway for information transmission across numerous time steps and is the major reason for overcoming the vanishing gradient problem.

The LSTM architecture consists of four core components that regulate the information flow.

**Forget Gate** The forget gate is responsible for deciding what information is to be removed from the cell state. The cell state is the previous hidden state and the current input passed through a sigmoid activation function, and the outputs that the gate returns are the values between 0 and 1 for each number in the cell state [23] [26]. A value of 1 means ”completely keep this information,” while 0 means ”completely discard this.” This mechanism permits the network to forget irrelevant information, such as the past trends of the market that do not affect current prices anymore.

**Input Gate** The input gate determines what new data will be inserted into the cell state. It includes two operations:

- A sigmoid layer responsible for the selection of the values that will be updated;
- A tanh layer that makes candidate values which can be added to the state

The addition of these two layers allows the network to completely update with new information selectively [26]. In stock prediction, this would typically involve updating the sys-

tem with new market data while keeping the information about relevant historic patterns unchanged.

**Cell State** The cell state travels through the whole chain and gets altered by the forget and input gates. It is a memory for the network, permitting the information that flows through with different littles difference unless the gateways explicitly modify them [23]. This advantage is critical in the stock market forecasting process since both the recent occurrences and the long-term trends have an effect on the currency fluctuations.

**Output Gate** The output gate regulates which portion's of the cell state will be outputted as the hidden state for the ongoing time step. It employs a sigmoid function to designate which components characteristics of the cell state to separate, and then apply, tanh to the cell state for the guarantee that the values range from -1 to 1 [26]. By this means, the network can focus just on the significant patterns of memory, while forecasting.

The networking of these components offers a mechanism for the transportation of regulated information throughout the nodal points. When it comes to stock exchange data that are fed to the machine, the LSTM is enabled to learn the forget process that includes the irrelevant market up and downs, remember useful patterns and cases, update its memory only when such data are available and present predictions based on a limited version of its knowledge base.

LSTMs, being the leading configuration for tapping long-term dependencies, the Gated Recurrent Units (GRUs) are a type that was also developed [23]. The GRUs shorten the Clojure code while keeping the LSTM paradigm by blending both the forget and input gates into an update gate and, in addition, merging the cell state and the hidden state. As a result, they have a smaller number of parameters and can be trained faster.

### 2.5.2 The Use of LSTMs in Monitoring the Stock Market.

The LSTMs have been very popular in financial markets for their capability of modeling complex sequences in stock price dynamics. The integration of LSTMs with up or down direction as a binary classification, along with the finding of the basic connections between the features and output, is thus a great and fruitful research and is evidenced by so many studies which have been conducted so far [27] [28] [29].

**The Input Feature**

The performance of LSTM models primarily depends on the choice of input features. The categories of features that are most prevalent:

**Historical Price Data** The main piece of price information which includes Open, High, Low, Close, and Volume (OHLCV) is the building block for basic LSTM stock prediction models [30] [27] [31]. The features are used to learn to predict the price of stock and the trend that will drive the price.

**Technical Indicators** In several studies to be said more positively, bidders use technical indicators that are mathematical tools based on the historical price and volume data to enhance the basic price data. Among the most frequently incorporated technical indicators are:

- Moving Average Convergence Divergence (MACD)
- Exponential Moving Averages (EMA) of a related time period (e.g., 26-day, 100-day, 200-day)
- Relative Strength Index (RSI)
- Bollinger Bands [32] [31]

The interconnection of these technical indicators with LSTM has been found to lead to much higher precision. So was the case for the research that one took with both MACD and different EMA variants used and get higher results than only OHLC features [32].

**Volatility Measures** Some attempts have been made to include the volatility as a input feature - in fact by the computation of the returns variance on a specific time basis [30]. It perceives uncertainty in market conditions and gives a sense of the risk which could help if the company wants to predict the direction.

**Sentiment Analysis** Advanced models sometimes attach the news sentiment derived from market agencies, social media, or financial magazines to price data as other input features [33]. The introduction of sentiment data to the technical ones might help to depict market dynamics more accurately and in the end, to predict changes in the future more precisely.

### Model Architectures and Configurations

The occasional stock market prediction literature tends to be RNN based. They highlight various stock market prediction LSTM architectures that they approached and which they would best recommend for stock market prediction.

**Standard LSTM** Standard LSTM networks are widely used in this field, and therefore, most of the studies follow this setup: one or more LSTM layers followed by one or more dense layers for final prediction [27] [34]. In general, these models will get the benefit of being fed by permanent sequence data from different streams of the time series, selecting lengths between 7 and 60 days.

**Hybrid Approaches** A few researchers have hybridized LSTM with other techniques so as to ameliorate performance:

- LSTM with Sequential Self-Attention Mechanism (LSTM-SSAM) which has a better effect on the model concerning the parts that should be focused on [35].
- Lower complexity on LSTM with technical analysis which helps featurization [31].
- LSTM-RF instead of LSTM-SVM algorithm for dual analysis and recommendation [36]

**Ensemble Methods** Additionally, the use of ensemble methods like mlp-lstm+svm and random forest which uses multiple classifiers to get the best result from all systems working together has been investigated. These types of ensemble methods usually yield better performance as compared to a single algorithm approach due to the effect of diversity on the ensemble [29] [33].

### Performance and Comparative Analysis

According to most of the literature, LSTM models outperform the standard time series instances as well as outdo most of the other machine learning algorithms when making stock predictions [28] [29] [37]. In one of the studies on LSTM, RNN, CNN, and MLP, it was discovered that LSTM was at the top for stock price prediction, then came CNN, RNN, and then MLP [29]. This advantage is due to the LSTM's capability of getting fluctuations about time in both short intervals and long-term dependencies at the same time.

The commonly used metrics for LSTM stock prediction models include:

- Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) for regression tasks
- Accuracy, precision, recall, and F1-score for classification tasks (predicting market direction)
- $R^2$ score to measure the proportion of variance explained by the model [32] [31] [36]

The literature has reported discrepancies in prediction accuracy levels owing to the various stock selections, periods of the study, and the geographic features, although many implementations have reached a reference performance of over 90% in the short-term directional forecasts [36]. Nevertheless, it is notable that the effectiveness of academic performance may not be the same in the stock market which difficults both the trading and the model implementation.

### 2.5.3 Reasons Why LSTMs Are Ideal for Stock Market Forecasting

LSTMs demonstrate various remarkable qualities, which make them especially appropriate for the prediction of stock market direction:

#### Capturing Long-Term Dependencies

What makes LSTMs unique is that they can store information over very long periods of time, this is actually their best feature. This way, they can detect and take advantage of macroeconomic cycles and long-term trends in the stock market that may affect today's prices [23] [26]. As this feature is the most prominent, it is particularly advantageous in financial markets, where the impact of macroeconomic cycles and long-term trends on short-term price movement is very common.

#### Non-Linear Relationships Handling

The financial market often contains complicated, non-linear relationships between variables. LSTMs can efficiently represent these non-linear forms without needing to state the underpinning relationships clearly [28] [37]. Thus, in this way, they are able to find more profound relations which could not have been found through other statistical methods.

**Noise Immunity**

Stock market data is inherently noisy due to random fluctuations and market inefficiencies. LSTM's matrices that gate filters may learn to remove irrelevant data so that it is easier to identify the meaningful patterns and noise can be disregarded [26]. The feature of filtering out excess noise contributes to the enhancement of prediction accuracy in the pullback markets.

**Feature Integration Makes If Flexible**

LSTMs can successfully deal with the integration of various types of features such as price data, technical indicators, volume information, and even sentiment data [32] [33]. This versatility allows for comprehensive market analysis incorporating various aspects of market behavior.

**Changing Market Conditions Are Not A Problem**

With the right training techniques, LSTMs are highly adaptive to the new market paradigms and learn new patterns through time [37]. However, as financial markets are constantly changing and developing, this adaptability is critical.

Putting all these capabilities together, LSTMs are not only highly suitable, but they also have the potential to be very effective in the difficult task of stock market direction prediction, where it is essential to balance the influence of both the past and the new data appropriately.

### 2.5.4 Limitations, Challenges, and Criticisms of LSTMs in this Context

Although they are strong, LSTMs are faced with some disadvantages, challenges, and problems in terms of stock market prediction:

**Hyperparameter Sensitivity**

Fine-tuning LSTM models usually requires adjustment of many hyperparameters including the number of layers, units per layer, sequence length, learning rate, and dropout rates [28] [37]. The fact that such hyperparameters are often sensitive can render this process very time-consuming as well as inconsistent, especially during different market phases.

**Price of Complexity**

When compared to other networks, LSTM networks are quite heavy while training and deploying them especially if you go for a multi-layer approach with plenty of units [26]. This computational complexity may make them impractical in situations like real-time high-frequency trading or for investors without powerful computational means.

**The Data Needs**

For LSTMs, the amount of historic data that is usually required to learn the associated patterns is also substantial [28]. This issue could raise some challenges in dealing with newly listed stocks or analyzing those stock markets that have limited data available.

**The Non-stationary Nature of Financial Markets**

The statistical properties of financial markets often change with time making them non-stationary [28] [37]. While LSTMs can handle some of these changes, the effects of market regime shifts, economic downturns, or structural changes are huge enough to counteract them.

**Overfitting Tendency**

Due to the high complexity of LSTM models, they tend to overfit, which is to say, they memorize data noise during the training instead of learning general patterns [29]. This risk is particularly acute in financial markets, where the signal-to-noise ratio is often low.

**Lacking Transparency and Interpretability of Black Box Nature**

Similar to other deep learning models, LSTMs also present interpretability issues since they are viewed as black box [28] [38]. The intricate relationships within the network system inhibit the explanation of which specific predictions are made and what is the basis for these predictions, thus financial professionals, who necessarily require a transparent decision-making approach, reduce trust and adoption among them.

**Market Efficiency Problems**

According to the Efficient Market Hypothesis, all the investors rapidly reflect the information available in the markets, thus it is tough to consistently predict price movements [37].

Opposing the view that any repeatable patterns would instantly be the target of market agents who would benefit until they no longer existed.

**The Implementation Gap**

There always seems to be a wide divergence between the academic performance that functions are claiming in real markets and what turns out in actual trading [28] [37]. Issues such as transaction costs, slippage, market impact, and an inability to short certain stocks can all lead to the erosion of the theoretical profits suggested by backtesting results.

These shortcomings bring to the fore the need to embrace realistic expectations on the LSTM-based stock market prediction project and to understand the challenges well. Even as LSTMs are promising agents for recognizing patterns in financial time series, they should be thought of as being just one instrument in a larger analytical machinery rather than being the only tool for market prediction.

# 2.6 Transformer Models

## 2.6.1 Transformer Architecture for Time Series

### Core Concepts and Self-Attention Mechanism

The Transformer architecture, initially formulated for natural language processing tasks, hinges upon the self-attention mechanism. This mechanism, in turn, allows the algorithm to quantify the different elements in the sequence when it tries to predict the next one. In contrast to recurrent models that process data one after the other, Transformers can evaluate all time points all together. This leads to lower training time and captures intricate relationships in financial time series data. The parallel processing of this approach is one of the main advantages when modeling stock market data, where short-term fluctuations and long-term trends both construct the price movements [39].

### Adaptation for Time Series Forecasting

While Transformers have strongly stood out in tasks related to language, their direct modeling of time series forecasting comes with several challenges. Classical Transformer architectures face issues with quadratic time complexity and memory-hogging when they process long sequences, a common occurrence in financial time series analysis. To counter these problems, researchers have put forward some customized architectures specifically tailored for time series data. The Informer model, for instance, is built on a ProbSparse self-attention mechanism that reaches O(L log L) time complexity and memory overhead, thus becoming the best option for long sequence forecasting tasks [40]. Likewise, the LogSparse Transformer takes advantage of locality-aware mechanisms at the cost of $O(L(\log L)^2)$ memory; thus, it is able to process fine-grained time series with strong long-term dependencies better [39].

### Architectural Modifications for Financial Data

Financial time series data is characterized by distinct properties that require certain architectural changes. Time series representing financial trading often consist of complex nonlinear patterns, high volatility, and noise no matter what measure is used, making it difficult for traditional Transformers to handle them efficiently. Current studies have been focusing on providing Transformer models with necessary modifications depending on the target of

stock market prediction. For example, the Fredformer is a model that addresses frequency biases in financial data, which is of significant relevance for capturing cyclical patterns in stock markets [41].

### 2.6.2 Application of Transformers in Stock Price Prediction

#### Key Research and Methodologies

The last few years have observed the groundbreaking elaboration of Transformer architectures that were initially developed for usage in various applications, now specifically aiming at stock price movement prediction. StockNet is a hybrid architecture which combines Long Short-Term Memory (LSTM) neural networks and Transformer networks to augment the utilization of contextual information and the correlation between truth and forecasted values. This combined architecture has shown stunningly good prediction results in tests on Chinese stock datasets, particularly 96.47% and 95.58% accuracies, and has demonstrated the cutting-edge technology of Transformer elements seamlessly integrated into other types of neural networks [42].

The Feature-Axis Transformer (FATE) represents another approach, which is based on the Efficient Market Hypothesis (EMH) model. FATE segregates the individual price components, unlike traditional models that process Open, High, Low, Close, and Adjusted Close (OHLCA) data as a unified input. It has three main sections: one that captures the temporal interaction between different stocks concerning each feature, another that generates data on the global market context, and the last one that performs contextual vector construction by correlating different prices with the closing price. Testing on six real datasets has shown that FATE clearly outperforms existing models and, besides that, it generates profitable portfolio trading signals [43].

#### Advanced Input Feature Processing

Various types of Transformer models utilize different techniques for processing input features used in stock price prediction. The Differential Transformer Neural Network (DTNN) attempts to address high-frequency trading explicitly through the extraction of information from Limit Order Books (LOBs). First, it uses a temporal attention-augmented bilinear layer with a temporal convolutional network to denoise the data. The differential layer then detects

fine distinctions between adjacent slices in the time series. Thereafter, the model has yielded the best results so far with the FI-2010 LOB benchmark and also shows strong potential for forecasting real stock data [44].

In the same way, the TLOB model based on the Transformer operates a dual attention mechanism to capture both spatial and temporal dependencies in LOB data. The flexibility in focusing the model adaptively on market microstructure activity was the reason for enhanced performance compared to alternative approaches while keeping a relatively simple architecture [45].

#### Hybrid Models and Architectural Innovations

The focus of researchers has been majorly on multi-modal models that capitalize on the unique strengths of each architecture, such as the Transformer and other neural network models, respectively. The BiLSTM-MTRAN-TCN model is a case in point; it includes bidirectional LSTM and a Transformer model improved with temporal convolutional networks. This hybrid method combines the respective advantages of each element, where: the BiLSTM extracts overall sequence knowledge, the Transformer provides full array distance knowledge, and the TCN improves sequence dependencies and generalization ability. Experiments on different stock datasets demonstrate that this method exceeded others by having $R^2$ improvements of 0.3% to 15.6% and RMSE reductions of 24.3% to 93.5% [46].

In the same spirit as the BiLSTM-MTRAN-TCN is the Deep Convolutional Transformer (DCT), which includes convolutional neural networks, Transformers, and multi-head attention mechanisms. It comes with an unprecedented inception convolutional token embedding architecture composed of separable fully connected layers. The DCT was first tested on the NASDAQ, Hang Seng Index, and Shanghai Stock Exchange Composite datasets with an average accuracy of 58.85% on the NASDAQ data over 30-day sliding windows as well as the lowest average prediction errors across multiple metrics [47].

### 2.6.3 Advantages of Using Transformers for Stock Prediction

#### Superior Handling of Long-Range Dependencies

The inherent ability of Transformer-based models to cover long-range dependencies is one of the most prominent perks when stock price prediction is concerned. Static forecasting

models such as ARIMA rely on the pre-setting of fixed lags, and recurrent neural networks usually run into problems with recognizing dependencies longer than certain time spans. The self-attention mechanism incorporated in Transformers, in theory, allows relationships to be formed regardless of the distance [40] [39]. This very quality is of immense advantage in the field of investment management, where the emotional response to large losses may lead to an investor abandoning a strategy that could ultimately be successful if implemented correctly.

**Parallel Processing and Computational Efficiency**

The parallel processing methods that Transformers include account for a lot of the computational advantages they gain. Instead of processing the data step-by-step as recurrent models do, these algorithms can study the whole set of time series simultaneously. This parallel computation not only leads to fast training but also to more efficient handling of complexity and forecasting errors in financial datasets. The upgraded variant, i.e., Informer, takes this efficiency further by reducing the time complexity from $O(L^2)$ to $O(L \log L)$, making it a better choice for real-life stock prediction applications of long historical sequences [40].

**Enhanced Pattern Recognition in Complex Financial Data**

Financial markets commonly show complicated, non-linear patterns where even the simplest models usually fail to replicate. Models based on the Transformer architecture have been revealed to have a much better capability in this field than any others, enabling them to establish intricate relationships even within a single market dataset. The Galformer model, for example, utilizes generative decoding integrated with a hybrid loss function that takes both quantitative error and a trend forecast into account, which in turn brings improved accuracy to this model on various stock market indices including CSI 300, S&P 500, Dow Jones Industrial Average, and Nasdaq Composite [48]. This greater recognition of patterns makes it reasonable for Transformers to become the tool of choice compared to traditional forecasting techniques which would have failed to discover such signals.

**Effective Handling of Multivariate Inputs**

Stock price movements are driven not only by historical prices but also by other factors such as technical indicators, fundamental data, and overall market conditions. Transformers are adept at handling multivariate inputs and discovering relationships both within variables

and across variables. The model named improved stock movement prediction Transformer displays this property effectively by assimilating the beneficial contribution of basic stock price indicators and technical indicators at the same time as inputs. The latter idea has demonstrated superior performance over time with various additional factors involved [49]. That very ability to process multiple inputs makes it possible to conduct a more thorough analysis of the market compared to univariate analysis.

### 2.6.4 Challenges and Limitations

#### Data Requirements and Overfitting Concerns

The outstanding performance of Transformer models generally implies their data appetite, due to which they achieve optimal results only under the condition of ample training data. This algorithm imposes some difficulties in financial applications where historical data may not be of high quality or may over time have a different regime. The architecture of Transformers, which is composite and made up of millions of parameters, leads to additional overfitting problems, especially when fed with noisy data. Many researchers who endeavor to lessen this problem resort to architectural alterations and regularization, but difficulties exist, particularly when predicting rare market events or when datasets are small and niche rather than typical financial datasets [47] [48].

#### Computational Expense and Implementation Challenges

Standard Transformer architectures are quadratic in time and space with respect to sequence length, making them infeasible for long financial time series due to high computational expense. However, the LogSparse Transformer, which is a variant that addresses this complexity, has an $O(L(\log L)^2)$ complexity, but on the other hand, during the training phase, it still requires sizable computational resources [39]. The heavy burden of computation can turn this into a practically infeasible task in high-frequency trading situations, where swift model updates are required, or in resource-constrained environments, where simple models might be preferable.

**Interpretability Concerns**

A big issue that is plaguing Transformer models in the financial sector is their low interpretability. They are capable of laying down impressive predictions, but discovering what makes the predictions is a tough challenge. This aspect, i.e., being a ”black box”, is very risky in making financial decisions due to the self-interpretability vision that regulatory requirements and risk management often prefer. Although the enhancement of interpretability through attention visualization or other techniques has been the target of some research, generally Transformers are still less interpretable than even simpler neural networks or traditional statistical models [43] [48].

**Sensitivity to Market Noise and Anomalies**

Financial markets move and are normally filled with a lot of noise and some anomalies that can deter models from predicting the right outcome. The regular Transformer architecture is among the most affected by such outliers because it is not localized per se. The issues that stem from these high transient rates have been approached through architectural alterations. For instance, the LogSparse Transformer proposes a convolutional self-attention mechanism to which local context is incorporated. By doing this, it is expected to become more robust against anomalies [39]. However, the uncertainty in effectively distinguishing between real market signals and noise still lingers as a significant challenge.

### 2.6.5 Comparative Analysis with Traditional Models

**Performance Comparison with ARIMA**

For years, ARIMA has been the gold standard among traditional time series models for making financial predictions, but it comes with the burden of making questionable assumptions about both data stationarity and linear relationships. In contrast, Transformer-based models, which are a class of neural networks, do not make such assumptions at all; hence, they can capture hidden non-linear relations that the ARIMA model is not capable of picking up. Statistical outcomes from several trials are the basis for saying that using Transformer models is better than using ARIMA models for stock prediction tasks. For example, the Galformer model has proven its effectiveness by leading traditional methods by a considerable margin on various stock indices, particularly in the case of multi-stepped predicted stock

price movements that require dealing with temporally complex dependencies [48].

**Advantages over SVM Models**

The application of Support Vector Machines (SVMs) has become commonplace in stock movement classification tasks recently due to their low training data requirement and excellent generalization performance. Despite this, SVM models often need painstaking feature engineering and are not good at handling large-scale data. However, Transformer models have removed these issues by allowing automatic feature extraction processes and scaling much better. The stock movement predictor improvement with the Transformer model is an example of this advancement, where it worked with both raw price indicators as well as technical indicators on its own, all the while achieving a better prediction rate without selecting features manually [49].

**Comparison with LSTM Networks**

Revolutionary LSTM networks led sequential data modeling as they addressed the vanishing gradient issue that had affected earlier recurrent networks. Although their sequential processing nature is the cause of computational bottlenecks and limits their long-range dependency capturing capabilities. Transformers are the champions who reverse these restrictions by means of parallel processing and unlimited attention spans. The hybrid model StockNet is an example of such combinations that exploit the advantages of both methods, employing LSTM for inner pattern recognition and Transformer capabilities for outer contextual information. Hybrid models have been able to churn out prediction accuracy rates greater than 95% on some stock datasets and thus achieved significantly better results than pure LSTMs [42] [46].

**Trade-offs in Model Selection**

The selection of the model between the two becomes a tacit agreement, in which case, although Transformers come forth as the more attractive option, the final decision must reflect the specific prediction demands and constraints after all. Generally, Transformers excel when the data is complex, multivariate, long-term dependencies are present, and in situations where sufficient training data and computational resources are available. Interpretability might be the deciding factor in choosing simpler models such as ARIMA or SVMs, along with sce-

narios when the available data is lacking or computational efficiency is vital. The BiLSTM-MTRAN-TCN hybrid model is a good example of this nuanced approach by bringing together different architectural elements to use complementary strengths while counteracting individual weaknesses [46].

## 2.7 KANs

In this segment, Kolmogorov-Arnold Networks (also known as KANs) are reviewed, which represents the latest in neural net technology and its application in stock price forecasting. KANs are fundamentally different from previous networks and inform us about new methods they might be using to perform better than the ones we were discussing, e.g., ARIMA, SVM, LSTM, and Transformer architectures.

### 2.7.1 Basics of Kolmogorov-Arnold Networks

The Kolmogorov-Arnold Networks owe their name to the Kolmogorov-Arnold representation theorem, which is their basis. This theorem is regarded as one of the main cornerstones of mathematical approximation theory, which speaks about how numbers can be represented in terms of simpler or more basic numerals. KANs operationalize the theorem by employing a new neural architecture that is completely different from traditional Multi-Layer Perceptrons (MLPs) [50]. The Kolmogorov-Arnold representation theorem of any continuous multivariate function via a finite composition of continuous functions of a single variable and addition operations serves as a solid theoretical base for the function approximation capabilities of KANs [50].

In contrast to typical neural networks where the activation functions work on nodes (also known as ”neurons”), KANs are designed in such a way that the activation functions can be learned and set on the edges (also known as ”weights”) [50]. This architectural innovation, instead of traditional linear weights, which are replaced with univariate functions parametrized as splines, allows KANs to adaptively learn activation functions specific to the data distribution they are modeling [50]. This difference in principle leads to KANs being able to work with fewer parameters than traditional MLPs, [50] which is quite encouraging, especially in areas that require efficient but powerful function approximation capabilities.

### 2.7.2 Structure and Function of KANs

The feature KANs boast of is replacing static weights with learnable univariate functions parametrized as splines [50]. In standard neural networks, the process of propagating information frequently requires a linear transformation followed by a set nonlinear activation. In contrast, KANs, which ascribe activation functions to network edges, process information in

a way that enables them to learn both the topology and the required characteristics for a given task [50][51].

Cubic B-splines are used for the edge functions in the prototypical implementation of KANs, but there is the option of using other forms. FastKAN, as a case in point, uses Gaussian radial basis functions to approximate these B-splines to improve computational efficiency [51]. Other versions include Chebyshev KANs, which apply Chebyshev polynomials for more efficient nonlinear function approximation [52], and Kolmogorov-Arnold Fourier Networks (KAF) that include Random Fourier Features that are trainable and a hybrid GELU-Fourier activation mechanism, thus allowing for a good balance between parameter efficiency and spectral representation capabilities [53].

The training of KANs involves the optimization of the spline functions' parameters, thereby enabling the network for the first time to learn complex patterns. The approach of using B-splines enables KANs to find irregular and complicated relationships in the data with minimal human intervention; thus, there is no need for manually defined features or predetermined activation functions [54].

### 2.7.3 KANs for Time Series and Financial Applications

Even if KANs are still on the threshold of an incline in the machine learning field, some scholars have started adding them to time series analysis and financial applications, which shows that these may be used for stock price prediction. Several specialized KAN variants have been identified both in the literature and empirical studies that are specifically tailored towards temporal data.

The Temporal Kolmogorov-Arnold Networks (TKANs) take the flexibility of KANs to another level by coupling them with Long Short-Term Memory (LSTM) mechanisms to perform multi-step time series forecasting with enhanced accuracy and efficiency [55]. This architecture introduces memory management features into Recurring Kolmogorov-Arnold Networks (RKANs) layers, therefore dealing with the traditional models' problem of how to handle complex sequential patterns [55]. Such capacities can be particularly relevant in stock price predicting, where both long-term trends and short-term fluctuations must be captured accurately at the same time.

The T-KAN and MT-KAN also represent an alternative method to KAN adjustments for time series analysis [56]. T-KAN is designed to identify concept drift within time series data

and explain nonlinear associations between predictions and prior time points, conversely to symbolic regression [56]. This explanatory aspect, which is crucial at times when the environment is dynamically changing, is even more so for the financial market, which often goes through regime shifts and changing correlations between variables [56].

On the other hand, for detailed financial applications, KANs have been used in forecasting the CBOE Volatility Index (VIX), thus demonstrating their power in analyzing financial markets [57]. Unlike MLP neural networks that are built on the black box premise frequently, KAN provides a more understandable path toward learning through spline-based functions that are learnable, thus potentially enhancing the level of trust and knowledge in financial forecasting models [57].

Furthering the scope of KAN's application to financial instruments, the KAN-based Option Pricing (KANOP) model has been introduced to value American-style options based on the Least Squares Monte Carlo algorithm [58]. This particular application shows KANs' capability to adapt to complex financial modeling tasks that involve using high-level function approximation [58].

### 2.7.4 Potential Advantages for Stock Price Prediction

KANs offer a range of benefits which may facilitate their application more effectively in stock price prediction compared to traditional approaches:

**Enhanced accuracy alongside parameter efficiency**

KANs have shown greater performance as compared to MLPs on a variety of tasks while still being economical in the use of parameters [50]. In the case of time series forecasting, KANs have outperformed traditional MLPs which had considerably more learnable parameters, demonstrating that they can catch the actual underlying patterns in sequential data more effectively [59]. The main implication of this parameter efficiency for stock prices could be a model that is both more accurate in its predictions and less likely to overfit, particularly when the training data is limited or noisy [50][59].

**Interpretability and Transparency**

One of the main advantages of KANs is their more direct interpretability than many other deep learning models [50]. The spline-based functions on network edges may be visualized

and analyzed at a glance, thus giving a more lucid route to the variables and their interactions. This transparency can be of far-reaching magnitude in the area of finance where model decisions should be clear and unambiguous, just as predictions often are [57]. By making the thought process behind predictions more accessible, KAN-based stock prediction models can promote a higher level of trust among financial analysts and decision-makers [50][57].

**Adaptive Learning of Complex Patterns**

Financial markets are characterized by their high complexity and dynamic nonlinearity where the relationships between variables evolve through time. The aptitude of KANs in dynamically learning activation patterns is just what is needed to capture the tangled relationships between variables [56]. T-KAN's ability to track concept drift and to explain nonlinear relationships could be wonderful tools for adjusting to regime changes in financial markets where the relationships between economic indicators and stock prices could be volatile in different conditions [56].

**Multi-Scale Analysis Capabilities**

Certain KAN versions like Wav-KAN (Wavelet Kolmogorov-Arnold Networks) use wavelet functions for efficiently capturing both high and low frequency components of the input data [60]. This ability of multi-resolution analysis could prove to be particularly advantageous for stock price prediction where there are different patterns such as intraday and long-term changes [60]. The Wav-KAN realization of the line between accurately captured underlying data and noise-induced overfitting could help in separating genuine market signals from random fluctuations [60].

### 2.7.5 Comparison with Traditional and Deep Learning Models

In introducing KANs vis-a-vis previously discussed models for stock price prediction, several comparative advantages and synergies emerge:

**KANs versus ARIMA Models**

Unlike ARIMA models, which are built on the assumption of linear relationships between lagged values and are thus limited in their ability to capture complex nonlinearities, KANs

have the ability to learn nonlinear functions adaptively from data without assuming a specific functional form [50]. While ARIMA models are best at capturing linear autocorrelations and seasonal patterns, KANs provide more flexibility for modeling complex stock price dynamics, potentially adding nonlinearity to them [50][59]. However, ARIMA models primarily depend on well-known statistical inference techniques and they are also simpler to apply.

**KANs versus SVMs**

Both SVMs and KANs focus on finding the best function approximation, although they are fundamentally different. SVMs use kernel functions which map data onto implicitly created higher-dimensional spaces where linear separation is then made possible; instead, KANs use explicit sequential learning of univariate function compositions for complex relational expressions [50]. The specific characteristics and KAN edge functions that are learnable allow for the adaptability of financial time series models, which in turn can result in better stock price predictions [50][59].

**KANs versus LSTMs**

On the one hand, the mainstream Long Short-Term Memory (LSTM) and KAN variants such as TKAN aim to capture temporal dependencies in sequential data [55]. The second way consists in their manners of memory and state management. LSTMs integrate an explicit gating mechanism to control the flow of information, whereas TKANs combine KAN architecture with memory management features inspired by LSTMs [55]. When the requirements call for programming that has both long-term memory and adaptive function approximation, the combination of these two in TKANs may be potentially better than standard LSTMs in some cases [55].

**KANs versus Transformers**

The attention mechanism of Transformer models, which is a weapon in a long-range strategy, is however vulnerable, being dependent on cabling and computational resources. Yet, KANs with their parameter efficiency and adaptive function learning gain a parity of performance with lower computational demands for certain tasks [50][59]. Nonetheless, the capacity of Transformers to process data streams in parallel connections and capture long-distance

time point interactions makes them advantageous over most implementations of KANs. A potential avenue for future R&D would be mixed technologies linking these structures.

### 2.7.6 Challenges and Limitations of Stock Markets Applications

Irrespective of their promising profile, KANs confront a number of challenges when applied to stock price prediction:

#### Training Dynamics and Optimization

For KANs, initialization schemes, optimizers, and learning rates need careful tuning procedures to get the best performance [54]. Research on different KAN architectures' training dynamics shows that locating the most appropriate training configurations is not easy; in fact, extensive trial and error is needed [54]. This issue may be especially problematic when trying to apply KANs to highly volatile, non-stationary stock market data.

#### Parameter Scaling for High-Dimensional Inputs

KANs are parameter-efficient in many cases; however, they might experience parameter explosion issues in high-dimensional settings [53]. The parameter p raises the issue of why KANs could possibly explode, especially in the high-dimensional case.

# Chapter 3

# Methodology and Model Implementation

This chapter details the methodology employed to predict next-day stock price direction. It begins by formally defining the prediction problem, followed by descriptions of data collection, preprocessing, and feature engineering. Subsequently, the configurations of various forecasting models are presented, including ARIMA, Support Vector Machines, LSTMs, Transformers, and Kolmogorov-Arnold Networks.

## 3.1 Problem Formulation

The primary objective of this research is to predict the direction of daily stock price movements. Let $S(t)$ denote the closing price of a given stock or index on day $t$. We aim to predict the sign of the change in closing price from day $t$ to day $t+1$, i.e., the sign of $S(t+1) - S(t)$.

This task is formulated as a binary supervised learning problem. For each day $t$, a set of input features is constructed using historical data available up to and including day $t$. The corresponding output label, $Y(t+1)$, is defined as:

- $Y(t+1) = 1$ (representing "price up") if $S(t+1) > S(t)$
- $Y(t+1) = 0$ (representing "price down") if $S(t+1) \leq S(t)$

A key assumption in this formulation is that days with no change in price (i.e., $S(t+1) = S(t)$) are treated as "price down". This binary classification framework provides a consistent target for all predictive models. The `Target_Direction` variable, generated in scripts for KAN, LSTM, and Transformer models, and inferred for ARIMA and SVM, corresponds to this $Y(t+1)$ label.

## 3.2 Data Collection and Preprocessing

The empirical basis of this research is historical daily stock market data (Open, High, Low, Close prices, and Volume - OHLCV) for ten selected U.S. equities (`AAPL`, `AMZN`, `GOOGL`, `JNJ`, `JPM`, `MSFT`, `PG`, `V`, `WMT`, `XOM`) and the S&P 500 Index (`GSPC`). Data was sourced from publicly available financial data providers (AlphaVantage). This pipeline follows a tiered structure:

1. **Raw Data**: This directory houses the initial, unaltered OHLCV data for each stock and the `GSPC` index, stored in individual CSV files.

2. **Processed Data**: Raw data undergoes initial cleaning, validation (e.g., ensuring correct data types, addressing missing values), and basic transformations, managed by. The resulting cleaned OHLCV data is stored in corresponding files), serving as the foundation for subsequent feature engineering.

3. **Features Data**: A comprehensive suite of technical indicators is generated (detailed in Section 3.3 which follows) from the processed data. These engineered features are combined with the processed OHLCV. These datasets are the primary inputs for the machine learning models.

Key preprocessing steps, orchestrated by the scripts and utility functions within model-specific scripts, include:

- **Target Variable Generation**: The binary `Target_Direction` (predicting next-day price movement, as defined in Section 3.1) is derived from the `'Close'` prices .

- **Data Splitting**: To prevent lookahead bias, datasets are chronologically partitioned. Splits include 80% training and 20% testing.

- **Data Scaling**: Feature scaling (e.g., standardization or normalization) is applied by fitting a scaler to the training data and then transforming all data splits (training, validation, and testing) to prevent data leakage.

This structured approach ensures data consistency, appropriate partitioning, and robust preparation for the diverse modeling techniques employed.

## 3.3 Feature Engineering

A major component of machine learning model design is the shift from purely raw financial time series data to the structured, mathematical explanatory variables. The noise removal in the data, through feature engineering, is what the ultimate goal is and thus the model will perform better and be more interpretable. The synergy of the feature engineering and the application of machine learning methods in financial predictions has been validated also in other domains like the energy market price forecasting [61]. In these studies, feature engineering refers to the generation of a whole new technical indicator, checking out its predictive power, and selection of various strategies to configure inputs for different modeling architectures. The first end of this step is the creation of the datasets for each stock and the GSPC index that have the new features.

The datasets that were created as a result of performance enhancement and testing of technical indicators are called engineered features datasets for the respective stock and the index . The original Open, High, Low, Close, and Volume (OHLCV) datasets plus numerous additional calculated indicators, which are presumed to be added by a data preparation script, are the new features in these datasets. The selection and the design of these indicators are based on financial literature and applications, with resources describing technical indicators and feature engineering research that have been the ground knowledge base.

### 3.3.1 Feature Importance Assessment using Random Forest

To quantitatively assess the relevance of the engineered features, Random Forest (RF) based feature importance was calculated. RF is an embedded feature selection method that inherently evaluates feature contributions during its training process, typically using metrics like Mean Decrease Impurity (MDI) or Mean Decrease Accuracy (MDA)). Its efficacy and widespread use make it a suitable choice for this task. The importance of feature selection and dimensionality reduction, for which RF is a suitable tool, is crucial for preparing raw financial data for machine learning algorithms [62].

The process involves two main stages:

1. **Individual Symbol Importance Calculation:** For each dataset containing engineered features corresponding to an individual stock or index, a dedicated process performs the following:

- Loads the feature-augmented dataset.
- Separates the features (X) from the target variable (`Target_Direction`).
- Trains a `sklearn.ensemble.RandomForestClassifier` (configured with `n_estimators=1000`, `random_state=42`, and `class_weight='balanced'` to handle potential class imbalances) on the complete set of features for that specific symbol.
- Extracts the feature importance scores using the `feature_importances_` attribute of the trained model.
- Saves these importance scores for each symbol.

2. **Consolidated Feature Importance:** This stage aggregates the individual importance scores to provide a broader perspective on feature relevance across all symbols:
   - It reads all previously saved individual importance scores.
   - It calculates the mean importance score for each technical indicator across all symbols.
   - These average importance scores are then saved into a consolidated report. This consolidated data serves as a key input for one of the feature selection strategies detailed below.

   Additionally, visualization tools are used to generate bar plots of the top N features, both from individual symbol importance calculations and from the consolidated average importances, aiding in the qualitative assessment of feature relevance.

### 3.3.2 Feature Selection Strategies

Given the large number of initially engineered features, and the inherent challenge of selecting relevant indicators in financial forecasting [63], we experimented with two main datasets to see if more features lead to better accuracy. One dataset was with all the engineered features and one with the first 25 features that were important according to the RF method above.

1. **"All Features":** This baseline approach utilizes all engineered technical indicators available in the feature-augmented datasets (after excluding raw price/target columns

like `'Close'`, `'Target_Direction'`, and the date/index).These engineered features are:

- **Simple Moving Average (SMA)** - with periods 5, 10, 20, 50 (indicated by `SMA_5`, `SMA_10`, `SMA_20`, `SMA_50`)
- **Historical Volatility** - with periods 10, 20 (indicated by `Volatility_10`, `Volatility_20`)
- **Relative Strength Index (RSI)** - `RSI`
- **Moving Average Convergence Divergence (MACD)** - `MACD`, with its components:
  - MACD Signal Line (`MACD_Signal`)
  - MACD Histogram (`MACD_Hist`)
- **Stochastic Oscillator** - with:
  - %K (`Stoch_K`)
  - %D (`Stoch_D`)
- **Williams %R** - `WillR`
- **Rate of Change (ROC)** - `ROC`
- **Know Sure Thing (KST)** - `KST`, with its signal line (`KST_Signal`)
- **Ultimate Oscillator (UltOsc)** - `UltOsc`
- **Triple Exponential Average (TRIX)** - `TRIX`
- **Average Directional Index (ADX)** - `ADX`, with its components:
  - Positive Directional Indicator (`ADX_Pos` or +DI)
  - Negative Directional Indicator (`ADX_Neg` or -DI)
- **Aroon Indicator** - with:
  - Aroon Up (`Aroon_Up`)
  - Aroon Down (`Aroon_Down`)
- **Aroon Oscillator** - `Aroon_Osc`
- **Parabolic SAR (PSAR)** - `PSAR`
- **Ichimoku Cloud** - components include:

  - Senkou Span A (`Ichimoku_A`)
  - Senkou Span B (`Ichimoku_B`)
  - Kijun-sen (Base Line) (`Ichimoku_Base`)
  - Tenkan-sen (Conversion Line) (`Ichimoku_Conv`)
- **On-Balance Volume (OBV)** - `OBV`
- **Chaikin Money Flow (CMF)** - `CMF`
- **Money Flow Index (MFI)** - `MFI`
- **Accumulation/Distribution Index (ADI)** - `ADI`
- **Ease of Movement (EMV)** - `EMV`, with its signal line (`EMV_Signal`)
- **Force Index** - `ForceIndex`
- **Average True Range (ATR)** - `ATR`
- **Bollinger Bands** - including:
  - Upper Band (`BB_High`)
  - Lower Band (`BB_Low`)
  - Middle Band (`BB_Mid`)
- **Bollinger Bands Width** - `BB_Width`
- **Bollinger Bands %B** - `BB_PctB`
- **Keltner Channels** - including:
  - Upper Channel (`KC_High`)
  - Lower Channel (`KC_Low`)
  - Middle Channel (`KC_Mid`)
- **Keltner Channels Width** - `KC_Width`
- **Keltner Channels %B** - `KC_PctB`
- **Detrended Price Oscillator (DPO)** - `DPO`

This strategy provides a comprehensive input set but may include redundant or noisy features.

2. **”Selected Features” (Predefined List):** For certain models, particularly SVM,KAN, LSTM and Transformer architectures, a predefined list of 25 specific technical indicators was used (according to RF as mostly significant). These features are:

- **Detrended Price Oscillator (DPO)** - `DPO`
- **Chaikin Money Flow (CMF)** - `CMF`
- **Ease of Movement (EMV)** - `EMV`
- **Ultimate Oscillator (UltOsc)** - `UltOsc`
- **Money Flow Index (MFI)** - `MFI`
- **Historical Volatility** - with period 10 (indicated by `Volatility_10`)
- **Average Directional Index (ADX)** - `ADX`, with its components:
  - Positive Directional Indicator (`ADX_Pos` or +DI)
  - Negative Directional Indicator (`ADX_Neg` or -DI)
- **Bollinger Bands Width** - `BB_Width`
- **Stochastic Oscillator %D** - `Stoch_D` (often used with %K)
- **Rate of Change (ROC)** - `ROC`
- **Keltner Channels Width** - `KC_Width`
- **Williams %R** - `WillR`
- **Historical Volatility** - with period 20 (indicated by `Volatility_20`)
- **Stochastic Oscillator %K** - `Stoch_K`
- **Bollinger Bands %B** - `BB_PctB`
- **Force Index** - `ForceIndex`

The above features where used in addition to the ”All features” to see if feature engineering matters and how the results where formed.

### 3.3.3 Feature Scaling

Feature scaling is a vital preprocessing step for many machine learning algorithms, especially those sensitive to feature magnitudes—such as distance-based methods (e.g., SVM) or neural networks. It ensures that all features contribute equally during training and helps stabilize the learning process. The effectiveness of feature selection and engineering, including scaling, in boosting the predictive power of ML models for financial forecasting is well documented.

1. **StandardScaler:** This scaler removes the mean and scales features to unit variance. It is used in experiments with KAN, SVM, and Transformer models. To prevent data leakage, the StandardScaler is fitted only on the training set and then applied to the training, validation, and test partitions.

2. **MinMaxScaler:** This scaler rescales features to a fixed range, typically $[0, 1]$. It is used in the LSTM field experiment. As with the StandardScaler, the MinMaxScaler is fitted exclusively on the training data and subsequently applied to the validation and test sets.

## 3.4 Analytical Examination of ARIMA Modeling Experiments

The utilization of Autoregressive Integrated Moving Average (ARIMA) models to forecast the subsequent day's stock price direction was methodically examined through a sequence of distinct, increasingly sophisticated experiments. Each experiment was crafted to examine particular elements of model formulation, estimation methodology, and diagnostic validation, with the objective of clarifying the circumstances under which ARIMA models may provide predictive utility for this complex binary classification challenge.

**Creation of a Baseline using Automated Order Selection (Static Model) (V1)**

**Background** This preliminary experiment established a baseline performance standard. An ARIMA(p,d,q) model was configured, with 'p' denoting autoregressive terms (past values affecting the present), 'd' indicating the degree of differencing for achieving stationarity, and 'q' representing moving average terms (previous errors impacting the current value). The precise (p,d,q) order was automatically established for each stock's training data, generally by minimizing an information criterion (e.g., AIC) that reconciles model fit with complexity. The model parameters, after being estimated on the training set, remained constant ("static") for producing forecasts during subsequent validation and test periods.

**Research Question** What degree of directional predictive accuracy may be attained using a standard ARIMA model, with its order (p,d,q) established via an automated information criterion-based search on training data, without further model adjustment?

**Rationale and Methodology** This established a reference point for a non-adaptive ARIMA. Price forecasts were transformed into directional movements (up/down relative to the previous day's close) to assess performance as a binary classification task.

**Evaluation of Adaptive Forecasting using Rolling Window Methodology with Automated Order Selection (V2)**

**Background** Financial markets demonstrate evolving dynamics. A "rolling forecast" methodology, also known as an expanding window approach, addresses this issue by re-estimating the model as new data is acquired. In this experiment, the ARIMA(p,d,q) order, initially determined automatically, was maintained constant. However, the model's parameters (coefficients for the AR and MA components) were revised at each step of the forecasting horizon by re-fitting the model using all historical data available up to that moment.

**Research Question** Can the predictive performance of an ARIMA model be enhanced by continuously re-estimating its parameters through an expanding or sliding data window, while maintaining an initially auto-selected model order?

**Rationale & Methodology** This examined whether adjusting to recent market dynamics by parameter re-estimation, despite a constant model structure, enhanced directional accuracy. Diagnostic plots from the initial model fit were also assessed for preliminary model adequacy.

**Investigation of Programmatically-Generated Orders within a Rolling Forecast Framework (V3)**

**Background** This experiment implemented a more systematic, data-oriented methodology for determining the (p,d,q) order for each stock, rather than depending exclusively on complete automation. The essential 'd' parameter (differencing for stationarity) was ascertained programmatically using statistical stationarity tests (e.g., Augmented Dickey-Fuller). Subsequent to determining 'd', suitable values for 'p' (autoregressive order) and 'q' (moving average order) were selected, frequently informed by information criteria for the given 'd'. This stock-specific (p,d,q) order was subsequently utilized within the rolling forecast mechanism.

**Research Question** Does a more systematic, theoretically-informed (though programmatically executed) method for selecting the ARIMA order (p,d,q) for each individual stock, in conjunction with a rolling forecast mechanism, enhance directional forecasting accuracy?

**Rationale and Methodology** This refined the selection of orders before implementing the adaptive rolling forecast.

**Systematic ARIMA Model Identification and Diagnostic Validation (Analytical Phase) (V4)**

**Background** This experiment explored the traditional Box-Jenkins approach for constructing ARIMA models. It is an iterative process encompassing:

- *Identification:* Examining Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots of the (differenced) time series to deduce potential (p,d,q) orders. ACF quantifies correlation with preceding values, while PACF assesses this association after adjusting for intermediate lags.
- *Estimation:* Fitting the model according to the determined order.
- *Diagnostic Checking:* Assessing whether the model's residuals (prediction errors) exhibit characteristics of random white noise, signifying that the model has effectively captured systematic patterns.

**Research Question** What is the methodology, and what are the resultant characteristics, of a comprehensive, sequential identification of ARIMA model orders (p,d,q) grounded in recognized time series analysis principles, including meticulous residual diagnostics?

**Rationale and Methodology** This emphasized the rigor of model specification on training data, prioritizing statistical validity above immediate out-of-sample forecasting.

**Examination of Conditional Heteroskedasticity in ARIMA Residuals (ARCH/GARCH Analysis)**

**Background** Standard ARIMA models presuppose homoscedasticity (constant error variance). Financial time series, however, usually demonstrate ”conditional heteroskedasticity,”

characterized by time-varying error variance, often manifesting as ”volatility clustering”—where periods of elevated volatility are typically succeeded by further high volatility, and conversely for low volatility. Autoregressive Conditional Heteroskedasticity (ARCH) and Generalized ARCH (GARCH) models are formulated to encapsulate such time-varying volatility. This experiment analyzed the residuals from the primary ARIMA model (which models the mean or price level) for these volatility patterns.

**Research Question** Do the residuals from ARIMA models applied to stock price series display time-varying volatility (ARCH/GARCH effects), and if so, can these effects be effectively modeled?

**Rationale & Methodology** Following the fitting of an ARIMA model, its residuals were examined for ARCH effects (e.g., utilizing the Lagrange Multiplier test). If significant ARCH effects were detected, a GARCH model was subsequently fitted to the ARIMA residuals to model their conditional variance.

## 3.5 Experimental Design for Support Vector Machine Modeling

This section outlines the comprehensive experimental methodology established to evaluate the effectiveness of Support Vector Machine (SVM) models in predicting the direction of next-day stock prices. The inquiry encompasses eleven unique financial products, facilitating an assessment of model robustness across diverse market conditions. This research primarily focuses on the thorough comparison of SVMs trained using two different feature paradigms: one utilizing the entire array of designed predictive variables and the other utilizing a carefully chosen subset of significant features 3.3.2. This comparison design is essential for comprehending the model’s sensitivity to feature dimensionality, the likelihood of information saturation or noise interference in high-dimensional financial data, and the practical advantages of informed feature selection.

### 3.5.1 Analytical Evaluation of Support Vector Machines Utilizing Comprehensive Feature Sets: Theoretical Rationale and Methodological Implementation

This experimental sequence exposes the SVM to the complete intricacy of the accessible feature space. The primary theoretical objective is to comprehend the intrinsic capabilities and limitations of SVM while learning from high-dimensional, potentially noisy financial data without prior feature reduction.

**Experiment Version 1 (Baseline Performance Evaluation – Measuring Untailored Effectiveness) (referred to as ”Baseline” in Chapter 4 results)**

- **Theoretical Justification:** This experiment serves as a crucial initial step to quantify the SVM's inherent predictive capability. It theoretically assesses whether the fundamental geometric principles of SVMs—specifically, the construction of an optimal separating hyperplane—can discern a meaningful predictive signal from the comprehensive suite of financial indicators for directional forecasting, even without hyperparameter tuning. This establishes a foundational performance baseline against which subsequent, more sophisticated experimental variations are benchmarked.

- **Methodological Implementation:** The SVM's baseline performance is isolated by employing a chronological `train_test_split` (maintaining temporal order via `shuffle=False`) and a standard `SVC` classifier with its default parameters (e.g., RBF kernel, C=1.0). To address potential class imbalance, `class_weight='balanced'` is strategically implemented, preventing the baseline from being skewed by trivial majority-class predictions. Feature scaling is performed using `StandardScaler`, fitted exclusively on the training data and then applied to the test data, a critical step to prevent data leakage and ensure the analytical validity of the baseline. The deliberate omission of hyperparameter optimization ensures that any observed predictive performance (or lack thereof) is attributable to the intrinsic SVM algorithm and the raw feature interactions, rather than to model tuning.

**Experiment Version 2 (Hyperparameter Optimization by Randomized Search – Investigating Efficient Hyperparameter Tuning) (referred to as "Random Search" in Chapter 4 results)**

- **Theoretical Justification:** The efficacy of SVMs is markedly dependent on their hyperparameter settings, which govern the trade-off between model complexity (fitting the training data accurately) and generalization ability (performing well on unseen data). This experiment posits that a systematic, albeit stochastic, exploration of the hyperparameter space can identify configurations that align the SVM's learning capacity with the underlying patterns in financial data, thereby yielding superior predictive accuracy over the baseline.

- **Methodological Implementation:** `RandomizedSearchCV` is employed, with hyperparameters such as C and gamma sampled from `loguniform` and `uniform` distributions, respectively. This approach facilitates an efficient exploration of a broad and potentially complex hyperparameter landscape; `loguniform` is particularly suited for parameters like C and gamma, which often operate on a multiplicative scale. The number of iterations (`n_iter`) quantitatively controls the balance between computational cost and the thoroughness of this search. The analytical goal is to ascertain if a comprehensive yet computationally feasible search can uncover configurations outperforming the baseline. Crucially, `TimeSeriesSplit` is used for cross-validation, ensuring that each fold preserves temporal order. This respects the non-i.i.d. nature of financial data, theoretically leading to more robust hyperparameter evaluation and improved generalization estimates. The final model, incorporating the optimal hyperparameters, is trained on the combined and re-scaled training and validation data to maximize data utilization.

**Experiment Version 3 (Hyperparameter Optimization using Grid Search - Assessing Comprehensive Refinement of Optimal Regions) (referred to as "Grid Search" in Chapter 4 results)**

- **Theoretical Justification:** While randomized search offers efficiency, grid search provides exhaustive evaluation within a defined subspace. This experiment theoretically investigates whether a meticulous, point-by-point examination of pre-specified hyper-

parameter values—potentially informed by prior experiments or domain knowledge—can further refine SVM performance. It explores the premise that local optima identified by randomized search might be improved through more granular exploration.

- **Methodological Implementation:** `GridSearchCV` is implemented with a predefined `param_grid` specifying discrete values for C, gamma, and kernel types (e.g., linear, rbf). This signifies a shift towards an exhaustive local search. The analytical question is whether a systematic evaluation of all combinations within this grid yields a more consistently optimal or marginally superior model compared to stochastic sampling. Although computationally more intensive, this method guarantees finding the best combination within the specified grid. The continued use of `TimeSeriesSplit` maintains analytical rigor in hyperparameter validation. Comparing results from `RandomizedSearchCV` and `GridSearchCV` allows for an analysis of the efficiency-effectiveness trade-off in hyperparameter optimization for this domain.

**Experiment Version 4 (Rolling Forecast Evaluation - Evaluating Adaptive Learning and Performance Consistency in Dynamic Environments) (referred to as ”Rolling Forecast” in Chapter 4 results)**

- **Theoretical Justification:** Financial markets exhibit non-stationarity; their statistical properties and underlying predictive relationships evolve. A model calibrated at one point in time may lose optimality. This experiment evaluates SVM performance in a more realistic, adaptive setting, testing the hypothesis that periodic retraining with new data, using previously optimized hyperparameters, can maintain predictive efficacy in the face of market drift.

- **Methodological Implementation:** `TimeSeriesSplit` is utilized to create expanding training windows and fixed (or rolling) test windows across the dataset, analytically simulating a real-world scenario of periodic model retraining. The analysis focuses on performance stability and adaptability: does the model, using pre-determined optimal parameters (from Version 3), maintain its effectiveness across different market phases? A key implementation detail is the re-fitting of the `StandardScaler` on each new training split, ensuring adaptive scaling. Aggregating predictions across all test windows provides a comprehensive performance metric reflecting average behavior over

time, theoretically offering a more robust assessment of long-term viability than a single train-test split. This approach provides insight into the temporal consistency of SVM performance and its robustness to market regime changes.

**Experiment Version 5 (Investigation of Polynomial Kernel Abilities – Analyzing Distinct Non-Linear Feature Interactions) (referred to as "Polynomial Kernel" in Chapter 4 results)**

- **Theoretical Justification:** Different SVM kernels implicitly map data into different feature spaces. Polynomial kernels construct new features from combinations and powers of the original features. This experiment theoretically explores whether explicit polynomial feature interactions (e.g., product of two indicators, square of an indicator) are beneficial for capturing non-linear dynamics influencing stock price direction, which might be missed by other kernel types.

- **Methodological Implementation:** The `param_grid` for `GridSearchCV` explicitly includes `{'kernel': ['poly'], 'degree': [2, 3], ...}`. This allows for a direct analytical assessment of whether polynomial transformations of features (up to a specified degree) can better model the decision boundary. The exploration of the `degree` parameter is crucial; testing different degrees facilitates an analysis of the optimal complexity for polynomial interactions, balancing model fit against the risk of overfitting. The polynomial kernel's ability to capture interaction effects between features may be particularly relevant for financial data where feature combinations often drive market behavior.

**Experiment Version 6 (Feature Selection Impact Assessment – Evaluating Recursive Feature Elimination) (referred to as "Feature Selection" in Chapter 4 results)**

- **Theoretical Justification:** This experiment investigates whether systematic feature selection can improve SVM performance by removing irrelevant or redundant features. The hypothesis is that a more focused feature set may reduce overfitting and improve generalization by eliminating noise while retaining predictive signal. In high-dimensional financial data, many features may be correlated or provide redundant information, making feature selection a critical step for model optimization.

- **Methodological Implementation:** Recursive Feature Elimination (RFE) or similar feature selection techniques are applied to identify the most important features for SVM classification. The selection process involves iteratively training the SVM and removing the least important features based on feature importance scores or coefficients. The selected features are then used to train the SVM with optimized hyperparameters determined through cross-validation. This approach tests whether algorithmic feature selection can achieve better results than using all available features, potentially improving both computational efficiency and predictive performance.

**Experiment Version 7 (Enhanced Data Preprocessing – Evaluating Improved Scaling and Data Handling) (referred to as "Standard Scaling" in Chapter 4 results)**

- **Theoretical Justification:** This experiment evaluates whether improved data preprocessing, particularly enhanced scaling techniques and data cleaning procedures, can boost SVM performance. The hypothesis is that more sophisticated preprocessing can better normalize the feature space and remove data quality issues that may hinder learning. Proper preprocessing is fundamental to SVM performance, as the algorithm is sensitive to feature scales and data quality.

- **Methodological Implementation:** Enhanced `StandardScaler` implementation with improved data splitting and preprocessing pipeline. This version incorporates lessons learned from previous experiments to optimize the data preparation phase, ensuring consistent scaling across training and test sets while maintaining temporal integrity. Key improvements include more robust handling of missing values, enhanced feature scaling methodology, and refined data partitioning strategies. The implementation features fixed `TRAIN_RATIO` and `VALIDATION_RATIO` for data partitioning, and definitive data cleaning steps (e.g., `dropna()`) to address data quality issues systematically.

### 3.5.2 Analytical Examination of SVMs Utilizing Selected Features: Theoretical Rationale and Methodological Implementation

This parallel series of studies employs identical theoretical inquiries and analytical methodologies as outlined in Subsection 3.5, but utilizes the SVM on a diminished feature set. The

fundamental theoretical premise posits that a more concise, signal-dense feature collection may result in SVMs that exhibit reduced susceptibility to overfitting, enhanced computing efficiency, and potentially greater interpretability, without a substantial decline—or possibly an improvement—in predicting efficacy.

**Experiment Versions 1-AVG to 7-AVG (Comparative Analysis of Feature Reduction Effects)**

- **Theoretical Justification (General):** For each ”AVG” version, the theoretical inquiry is: how does limiting the SVM to a predetermined subset of high-importance features influence its learning behavior, optimal configuration, and overall prediction performance in comparison to its access to the complete feature set? This examines the effectiveness of the feature selection technique and investigates whether the curse of dimensionality affects SVM performance in this financial prediction context.

- **Methodological Implementation (General):** The principal implementation distinction involves an initial data preparation step where a predefined list of high-importance features (Top 25 Features) is loaded, and the input data (X) is subsetted to include only these selected features. This constitutes the analytically distinct variable between the ”all features” and ”AVG features” experiments. The subsequent SVM instantiation, hyperparameter search methodologies (using consistent parameter grids for tuning), cross-validation strategies (employing `TimeSeriesSplit`), and scaling logic are implemented identically to their ”all features” counterparts. This consistent approach ensures that variations in outcomes (e.g., selected hyperparameters, performance metrics) can be more reliably attributed to the change in the feature set. For the rolling forecast experiment (Version 4-AVG), an essential detail is the utilization of hyperparameters previously optimized specifically for the diminished feature set (i.e., from Version 3-AVG results), thereby maintaining analytical consistency for that comparison. By systematically comparing metrics such as the optimal C value identified in Version 3 (all features) with that in Version 3-AVG (selected features), one can analytically infer whether the reduced feature set necessitates different regularization strengths. If, for instance, Version 1-AVG demonstrates markedly superior performance compared to Version 1, it would suggest that the feature selection process successfully mitigated noise that impeded the baseline SVM. The complete suite of ”AVG” experiments pro-

vides a thorough analytical perspective on the significance and implications of feature selection for SVM-based financial forecasting.

**Methodological Consistency and Comparative Analysis Framework**

The experimental design ensures that each version (1-7) and its corresponding AVG counterpart (1-AVG to 7-AVG) differ only in the feature set utilized, enabling direct comparison of the impact of feature dimensionality on SVM performance. This systematic approach allows for:

- **Performance Comparison:** Direct assessment of whether feature reduction improves, maintains, or degrades predictive accuracy across different optimization strategies.
- **Computational Efficiency Analysis:** Evaluation of training time and resource requirements between full and reduced feature sets.
- **Hyperparameter Sensitivity:** Investigation of whether optimal hyperparameters change significantly when feature dimensionality is reduced.
- **Generalization Assessment:** Analysis of whether reduced feature sets lead to better generalization across different stocks and market conditions.

This comprehensive experimental framework provides a robust foundation for understanding the interplay between feature selection, SVM configuration, and predictive performance in financial time series forecasting.

## 3.6 Experimental Design for LSTM Modeling

This section details the comprehensive experimental methodology designed to evaluate the efficacy of Long Short-Term Memory (LSTM) neural networks in predicting the next-day direction of stock prices. The investigation spans the same eleven unique financial products as previous analyses, allowing for an assessment of model generalization across varied market dynamics. A central theme of this research is the rigorous comparison of LSTMs based on architectural variations (baseline, stacked, bidirectional, CNN-hybrid, attention-augmented) and hyperparameter optimization strategies (fixed vs. Optuna-tuned). Furthermore, similar to the Support Vector Machine (SVM) experiments, each LSTM configuration is evaluated

under two distinct feature paradigms: one employing the complete set of engineered predictive variables and another utilizing a pre-selected subset of high-importance features. This dual-feature approach is crucial for understanding the LSTM's sensitivity to feature dimensionality, its capacity to discern signal from noise in high-dimensional financial data, and the practical implications of feature selection in the context of deep learning models.

### 3.6.1 LSTM Model Fundamentals and Experimental Rationale

LSTMs, a type of Recurrent Neural Network (RNN), are inherently designed to process sequential data, making them theoretically well-suited for time-series forecasting tasks like stock price direction prediction. Their gating mechanisms (input, forget, output gates) allow them to learn long-range dependencies and mitigate the vanishing gradient problem common in traditional RNNs. The experiments herein are designed to explore:

1. **Architectural Impact:** How different LSTM-based architectures influence predictive performance.

2. **Hyperparameter Sensitivity:** The extent to which performance relies on careful tuning of hyperparameters.

3. **Feature Set Influence:** The effect of using all available features versus a curated subset on model training, complexity, and generalization.

### 3.6.2 Analytical Evaluation of LSTMs Employing Comprehensive Feature Sets: Theoretical Rationale and Methodological Implementation

This series of experiments exposes diverse LSTM architectures to the complete intricacy of the accessible feature space without prior feature reduction. The main theoretical aim is to comprehend the inherent learning capacities and possible constraints of various LSTM configurations when faced with high-dimensional, potentially noisy financial data.

**Experiment Version 1 (Baseline 2-Layer LSTM with Fixed Hyperparameters — Establishing a Foundational Benchmark)**

- **Theoretical Justification:** This initial experiment aims to establish a performance benchmark for a conventional 2-layer LSTM architecture utilizing a standardized set of fixed hyperparameters. It evaluates the intrinsic capacity of a moderately deep LSTM to discern temporal patterns from the entire feature set without extensive tuning, serving as a benchmark for more intricate architectures and hyperparameter optimization endeavors.

- **Methodological Implementation:** A standard two-layer LSTM model is constructed. Each LSTM layer is configured with 50 units, employing `tanh` activation functions and `sigmoid` recurrent activations, with dropout regularization (rate of 0.2) applied after each LSTM layer to mitigate overfitting. The LSTM outputs are processed by a dense hidden layer of 25 units with `relu` activation, followed by a final sigmoid output layer for binary classification. The model is compiled using the Adam optimizer and binary cross-entropy loss. Training is conducted for a maximum of 50 epochs with a batch size of 32, using a sequence length of 30 time steps. The dataset is partitioned chronologically into an 80% training set and a 20% test set. Feature scaling is performed using Min-Max normalization, where the scaler is fitted solely on the training data and then applied to both training and test sets to prevent data leakage. Early stopping (monitoring validation loss with a patience of 10 epochs) and learning rate reduction on plateau (patience of 5 epochs) are employed during training to enhance generalization and training stability. The primary focus is to assess the inherent performance of this foundational LSTM setup with common, reasonable hyperparameter settings.

**Experiment Version 1-Optuna (Baseline 2-Layer LSTM with Optuna Hyperparameter Optimization — Augmenting Fundamental Performance)**

- **Theoretical Justification:** Building on the established baseline, this experiment examines whether systematic hyperparameter optimization via Optuna can substantially enhance the predictive performance of the 2-layer LSTM architecture. The hypothesis posits that default or arbitrarily selected hyperparameters are unlikely to be optimal for

particular financial datasets, and a directed search may reveal more efficacious configurations.

- **Methodological Implementation:** The core two-layer LSTM architecture is maintained, but its key hyperparameters are now subject to automated tuning. The initial dataset is divided into a main training/validation pool (first 80%) and a final hold-out test set (last 20%). This main pool is further chronologically subdivided, allocating 20% of its data for Optuna's validation phase, with the remainder used for training during the optimization trials. Min-Max feature scaling is carefully applied: the scaler is fitted *only* on the Optuna training partition and then used to transform the Optuna validation data, the final hold-out test set, and the full main training/validation pool (for final model training). Optuna, employing a Median Pruner to discard unpromising trials early, systematically searches for optimal values for the number of units in each LSTM layer (between 32 and 128), dropout rate (0.2 to 0.5), learning rate (logarithmically between 1e-4 and 1e-2), the number of epochs per trial (15 to 35), and batch size (testing 16, 32, and 64). Each batch size configuration undergoes 30 optimization trials. The optimization process aims to maximize the ROC AUC score on the Optuna validation set. Once the overall best hyperparameter combination is identified, a final model is trained using these optimal settings on the entire main training/validation pool (re-scaled appropriately) and its performance is evaluated on the final hold-out test set. This procedure represents a rigorous search for an optimized configuration of the baseline two-layer LSTM.

**Experiment Version 2 (Stacked 3-Layer LSTM with Fixed Hyperparameters — Investigating Deeper Architectures)**

- **Theoretical Justification:** This experiment investigates whether augmenting the depth of the LSTM network by incorporating a third LSTM layer enables the model to acquire more intricate hierarchical representations of temporal features from the entire dataset. The theory posits that deeper networks can encapsulate more abstract patterns, potentially enhancing predictive efficacy, even with static hyperparameters.

- **Methodological Implementation:** The model architecture is extended to a three-layer stacked LSTM. Each LSTM layer is configured with 50 units, `tanh` activation, and

`sigmoid` recurrent activation. The first two LSTM layers return full sequences to feed into the subsequent layer, while the final LSTM layer returns only the last output. Dropout regularization (rate of 0.2) is applied after each LSTM layer. The subsequent dense layers, compilation settings (Adam optimizer, binary cross-entropy loss), data partitioning (80/20 train/test split), feature scaling (Min-Max), sequence length (30), batch size (32), maximum epochs (50), and training callbacks (early stopping, reduce learning rate on plateau) are kept consistent with the baseline V1 experiment to facilitate direct comparison of the impact of increased architectural depth.

**Experiment Version 2-Optuna (Stacked 3-Layer LSTM with Optuna Hyperparameter Optimization — Enhancing Deeper Architectures)**

- **Theoretical Justification:** Akin to V1-Optuna, this experiment seeks to ascertain if hyperparameter optimization by Optuna can fully realize the capabilities of the deeper 3-layer stacked LSTM design. A more sophisticated model may possess a more elaborate hyperparameter landscape, rendering systematic adjustment increasingly essential.
- **Methodological Implementation:** The Optuna-based hyperparameter optimization process is applied to the three-layer stacked LSTM architecture. The number of units for each of the three LSTM layers is tuned (between 32 and 128 per layer). Other tuned hyperparameters include dropout rate (0.2 to 0.5), learning rate (logarithmically between 1e-4 and 1e-2), epochs per trial (15 to 35), and batch size (16, 32, or 64). The data splitting strategy for Optuna (initial 80/20 split, then a further 80/20 split of the training portion for Optuna's train/validation), feature scaling methodology, optimization objective (maximizing ROC AUC on Optuna validation), and the number of trials per batch size (30) are consistent with the V1-Optuna experiment. The best found hyperparameters are then used to train a final model on the combined Optuna training and validation data, evaluated on the hold-out test set.

**Experiment Version 3 (Bidirectional 2-Layer LSTM with Fixed Hyperparameters – Incorporating Context from Both Sequence Directions)**

- **Theoretical Justification:** Bidirectional LSTMs (BiLSTMs) process input sequences in both forward and backward directions, allowing each LSTM layer to leverage context from past and future (relative to the current time step within the sequence) infor-

mation. This experiment examines if such dual-context processing, using fixed hyperparameters, improves the model's capacity to comprehend temporal patterns pertinent to next-day prediction compared to a unidirectional approach.

- **Methodological Implementation:** The model is constructed with two Bidirectional LSTM layers. Each BiLSTM layer wraps an LSTM cell configured with 50 units, `tanh` activation, and `sigmoid` recurrent activation. The first BiLSTM layer returns sequences, while the second returns only the final output. Dropout (rate of 0.2) is applied after each BiLSTM layer. The subsequent dense layers, optimizer (Adam), loss function (binary cross-entropy), and other training parameters (sequence length, batch size, epochs, callbacks, data split, and scaling) are maintained consistently with the V1 baseline experiment to isolate the effect of bidirectionality.

**Experiment Version 3-Optuna (Bidirectional 2-Layer LSTM with Optuna Hyperparameter Optimization — Enhancing Dual-Context Models)**

- **Theoretical Justification:** This experiment employs Optuna for hyperparameter optimization of the 2-layer Bidirectional LSTM architecture. The objective is to determine the ideal configuration for a model that utilizes both historical and prospective sequence information, potentially leading to enhanced performance relative to the fixed-parameter BiLSTM and optimized unidirectional LSTMs.

- **Methodological Implementation:** The Optuna framework is used to tune the hyperparameters of the two-layer Bidirectional LSTM model. Tunable parameters include the number of units for each of the two BiLSTM layers (32 to 128), dropout rate (0.2 to 0.5), learning rate (logarithmically between 1e-4 and 1e-2), epochs per trial (15 to 35), and batch size (16, 32, or 64). The data partitioning strategy for Optuna, feature scaling, optimization objective (maximizing ROC AUC on Optuna validation), and trial management remain consistent with previous Optuna-based experiments, allowing for robust comparison of optimized bidirectional versus unidirectional models.

**Experiment Version 4 (CNN-LSTM Hybrid with Fixed Hyperparameters — Combining Spatial and Temporal Feature Extraction)**

- **Theoretical Justification:** This experiment presents a hybrid design that integrates 1D Convolutional Neural Networks (CNNs) with LSTMs. The CNN layers are intended to act as local feature extractors from the input sequences, identifying salient patterns across features or time steps. The LSTMs then model the temporal dependencies among these extracted convolutional features. This setup, with fixed hyperparameters, evaluates if this integrated methodology provides benefits.

- **Methodological Implementation:** The model architecture begins with a 1D Convolutional layer (64 filters, kernel size of 3, `relu` activation, 'same' padding to maintain sequence length), which processes the input sequences. The output of this CNN layer is then fed into a two-layer LSTM stack (each LSTM with 50 units, `tanh` activation, `sigmoid` recurrent activation, and 0.2 dropout), consistent with the V1 baseline's LSTM structure. Subsequent dense layers, compilation settings, and all other training parameters and data handling procedures are maintained as in the V1 experiment to assess the specific contribution of the preceding CNN layer.

**Experiment Version 4-Optuna (CNN-LSTM Hybrid with Optuna Hyperparameter Optimization – Optimizing Hybrid Architectures)**

- **Theoretical Justification:** This experiment seeks to identify the optimal hyperparameter configuration for the CNN-LSTM hybrid model. Optimizing parameters for both the CNN component (number of filters, kernel size) and the LSTM components (units, dropout), in addition to learning rate and batch size, is essential for achieving a balance between feature extraction effectiveness and temporal modeling.

- **Methodological Implementation:** Optuna's optimization procedure is applied to the CNN-LSTM hybrid architecture. Tunable parameters now include CNN-specific settings: number of filters (32, 64, or 128) and kernel size (2, 3, or 5). LSTM parameters (units for two layers, dropout rate), learning rate, epochs per trial, and batch size are tuned as in previous Optuna experiments. The data handling, scaling, and evaluation protocols remain consistent, allowing for a detailed analysis of the optimal configuration for this hybrid approach.

**Experiment Version 5 (Attention-Augmented LSTM with Fixed Hyperparameters – Emphasizing Salient Temporal Features)**

- **Theoretical Justification:** This experiment integrates an attention mechanism into a two-layer LSTM design. The attention layer enables the model to dynamically weigh the importance of different time steps in the input sequence when forming its representation, rather than relying solely on the final hidden state of the LSTM. This experiment, utilizing fixed hyperparameters, evaluates if this focusing capability enhances performance.

- **Methodological Implementation:** The model is constructed using the Keras Functional API to accommodate the attention mechanism. An initial LSTM layer (50 units, returning sequences, `tanh` activation, `sigmoid` recurrent activation) processes the input. Its output, after dropout (0.2), is fed as both query and value to a Keras `Attention` layer, performing self-attention over the sequence. The attended sequence output is then passed to a second LSTM layer (50 units, not returning sequences), followed by dropout (0.2). The model concludes with standard dense and output layers. Compilation and training parameters are kept consistent with the V1 baseline.

**Experiment Version 5-Optuna (Attention-Augmented LSTM with Optuna Hyperparameter Optimization – Optimizing Focused Architectures)**

- **Theoretical Justification:** This experiment aims to optimize the hyperparameters of the attention-augmented LSTM model. The interplay of LSTM unit sizes, dropout rates, and the processing of the attention mechanism can be intricate, requiring a methodical search for the optimal configuration.

- **Methodological Implementation:** The Optuna framework is employed to tune the hyperparameters of the attention-augmented LSTM. This includes the number of units for the first LSTM layer (feeding the attention mechanism) and the second LSTM layer (processing the attended sequence), both ranging from 32 to 128. Dropout rate, learning rate, epochs per trial, and batch size are also tuned. The model is built using the Functional API. Data handling, scaling procedures, and the Optuna optimization strategy (maximizing ROC AUC, number of trials, etc.) remain consistent with other Optuna experiments.

### 3.6.3 Analytical Examination of LSTMs Using Selected Features: Theoretical Justification and Methodological Implementation

This parallel series of experiments reflects the architectural investigations and hyperparameter optimization techniques outlined previously; however, it fundamentally operates on a reduced feature set. This selection, characterized by a predefined list of 25 technical indicators and return-based features, was determined through previous study. The fundamental theoretical assertion is that a more concise, signal-dense feature collection may result in LSTMs that exhibit reduced susceptibility to overfitting, enhanced computational efficiency, and possibly superior or equivalent predictive accuracy by emphasizing the most pertinent information.

**Experiment Versions V1-SEL to V5-SEL (Fixed Hyperparameters with Selected Features)**

- **Theoretical Justification (General):** For each of these ”SEL” (Selected Features) versions with fixed hyperparameters, the principal theoretical question is: how does limiting the LSTM (Baseline, Stacked, Bidirectional, CNN-LSTM, Attention-LSTM) to a predetermined subset of features influence its learning dynamics and predictive accuracy in comparison to when it has access to the entire feature set? This directly evaluates the effectiveness of the feature selection process within various LSTM architectures utilizing standard parameterization.

- **Methodological Implementation (General):** The core implementation difference from the ”all features” counterparts is the initial data preparation step, where the input data is filtered to include only the predefined subset of selected features along with the target variable and date information. The specific LSTM architectures (Baseline 2-layer, Stacked 3-layer, Bidirectional 2-layer, CNN-LSTM, Attention-Augmented LSTM) and their fixed hyperparameters (e.g., 50 units per LSTM layer, 0.2 dropout) are identical to their respective ”all features” fixed-hyperparameter versions. All other experimental parameters, including sequence length (30), batch size (32), maximum epochs (50), optimizer, loss function, callbacks, data splitting, and feature scaling, are maintained consistently. This allows for a direct analytical comparison of performance metrics attributable solely to the change in the feature set.

**Experiment Versions V1-SEL-Optuna to V5-SEL-Optuna (Optuna-Tuned with Selected Features)**

- **Theoretical Justification (General):** This series of studies examines whether hyperparameter optimization via Optuna may produce varying optimal configurations and performance metrics when implemented on LSTM models trained with the restricted, pre-selected feature set. The theory posits that the ideal hyperparameters (e.g., model complexity via LSTM units, regularization via dropout) may vary with lower input dimensionality, potentially resulting in simpler or more robust models.

- **Methodological Implementation (General):** These experiments apply the same Optuna-based hyperparameter tuning methodology as their ”all features” Optuna counterparts. The critical distinction is that the data preparation stage ensures only the predefined selected features are used to create sequences for Optuna’s training and validation phases, as well as for the final model training and testing. The range and types of hyperparameters tuned by Optuna (LSTM units, dropout rates, learning rates, batch sizes, and CNN-specific parameters for the V4-SEL-Optuna experiment) remain identical to those in the ”all features” Optuna experiments. The analytical output will facilitate a comparison of: (1) the optimal hyperparameters identified when using selected features versus all features for each architecture, and (2) the final test performance of optimally tuned models using selected features versus those using all features. This provides insights into the interplay between feature selection, model architecture, and hyperparameter optimization. The storage of results is directed to a specific subdirectory for selected feature experiments to ensure clear separation.

## 3.7 Experimental Design for Transformer Modeling

This section outlines the comprehensive experimental framework developed to assess the capabilities of Transformer-based neural networks for predicting the subsequent day’s directional movement of stock prices. Consistent with prior analyses, this investigation encompasses the same eleven unique financial instruments, facilitating an evaluation of model robustness and generalization across diverse market conditions. A primary focus of this research is the systematic comparison of Transformer models based on distinct architectural modifications (e.g., depth, number of attention heads, positional encoding type, output strategy) and

the impact of hyperparameter optimization. Echoing the experimental design for Support Vector Machines (SVMs) and Long Short-Term Memory (LSTM) networks, each Transformer configuration is subjected to evaluation under two separate feature set paradigms: one utilizing the complete array of engineered predictive variables, and another employing a pre-selected subset of features identified for their high importance. This dual-feature approach is pivotal for understanding the Transformer's sensitivity to input dimensionality, its proficiency in extracting relevant signals from high-dimensional financial time series, and the practical trade-offs associated with feature selection in advanced deep learning contexts.

### 3.7.1 Transformer Model Fundamentals and Experimental Rationale

Transformers, originally introduced for natural language processing, have demonstrated remarkable success in modeling sequential data due to their self-attention mechanism. This mechanism allows them to weigh the importance of different elements in a sequence, irrespective of their distance, making them adept at capturing long-range dependencies. Unlike LSTMs which process sequences recurrently, Transformers process all sequence elements in parallel after incorporating positional information. The experiments detailed herein are designed to investigate:

1. **Architectural Impact:** How variations in Transformer architecture—such as the number of encoder layers, attention heads, type of positional encoding, and output pooling strategy—affect predictive performance.

2. **Hyperparameter Optimization:** The degree to which performance is contingent upon systematic tuning of hyperparameters, explored here using KerasTuner.

3. **Feature Set Influence:** The effect of employing the full feature set versus a curated subset on model training dynamics, computational complexity, and generalization capabilities in the context of financial forecasting.

4. **Advanced Training Techniques:** The impact of specific optimizers (e.g., AdamW) and learning rate schedules.

All Transformer models are trained for binary classification (predicting whether the next day's closing price will be higher or lower than the current day's). Data preprocessing involves creating sequences of a fixed length, scaling features using `StandardScaler` (fit-

ted on the training set only), and splitting the data chronologically into training, validation, and test sets.

### 3.7.2 Analytical Evaluation of Transformers Employing Comprehensive Feature Sets: Theoretical Rationale and Methodological Implementation

This suite of experiments exposes various Transformer architectures to the full spectrum of available features without prior dimensionality reduction. The primary theoretical objective is to understand the inherent learning capabilities and potential limitations of different Transformer configurations when confronted with high-dimensional, and potentially noisy, financial time-series data. Unless otherwise specified, models in this subsection utilize a 70% training, 15% validation, and 15% test chronological split, an input sequence length of 60, a batch size of 32, a maximum of 50 training epochs, and employ `EarlyStopping` (monitoring validation loss with a patience of 10 and restoring best weights) and `ModelCheckpoint` (saving best weights based on validation loss) callbacks. Sinusoidal positional encoding is standard unless an alternative is explicitly mentioned. The output of the Transformer encoder is typically processed by a `GlobalAveragePooling1D` layer before a final dense sigmoid layer for classification.

**Experiment Version T0 (Baseline Transformer with Fixed Hyperparameters — Establishing a Foundational Benchmark) (referred to as v0 in Chapter 4)**

- **Theoretical Justification:** This initial experiment aims to establish a performance benchmark for a standard Transformer encoder architecture using a common set of fixed hyperparameters. This model serves as a reference point for evaluating more complex architectural variations and the benefits of hyperparameter optimization.

- **Methodological Implementation:** A Transformer model is constructed with 2 encoder layers. Each layer utilizes a model dimension (`d_model`) of 64, 4 attention heads, and a feed-forward network dimension (`dff`) of 128 (2 * `d_model`). A dropout rate of 0.1 is applied within the encoder layers. The model is compiled with the Adam optimizer (learning rate of 1e-4) and binary cross-entropy loss. This configuration represents a common, moderately sized Transformer setup.

**Experiment Version T1 (Deeper Transformer with Fixed Hyperparameters — Investigating Increased Depth) (referred to as v1 in Chapter 4)**

- **Theoretical Justification:** This experiment investigates whether increasing the depth of the Transformer encoder by using more layers enhances its ability to capture more complex patterns and hierarchical features from the temporal data. The hypothesis is that deeper models can learn more abstract representations, potentially improving predictive power, even with fixed hyperparameters for other components.
- **Methodological Implementation:** The architecture is modified to use 4 Transformer encoder layers. Other core parameters remain consistent with V-T0: `d_model` of 64, 4 attention heads, `dff` of 128, and a dropout rate of 0.1. Compilation settings and other training parameters are maintained as per the V-T0 baseline.

**Experiment Version T2 (More Heads Transformer with Fixed Hyperparameters — Exploring Attention Diversity) (referred to as v2 in Chapter 4)**

- **Theoretical Justification:** This experiment evaluates the impact of increasing the number of attention heads within each Transformer encoder layer. More heads allow the model to jointly attend to information from different representation subspaces at different positions. The hypothesis is that a greater diversity of attention patterns might improve the model's ability to capture various types of relationships in the data.
- **Methodological Implementation:** The model architecture employs 2 Transformer encoder layers, but the number of attention heads per layer is increased to 8 (from 4 in V-T0), while `d_model` remains 64 (ensuring `d_model` is divisible by `num_heads`). Other parameters like `dff` (128) and dropout (0.1) are kept consistent with V-T0.

**Experiment Version T3 (Learned Positional Encoding Transformer with Fixed Hyperparameters — Data-Driven Positional Information) (referred to as v3 in Chapter 4)**

- **Theoretical Justification:** This experiment replaces the standard sinusoidal positional encoding with learned positional embeddings. The rationale is that allowing the model to learn positional representations directly from the data might yield encodings more suitable for the specific financial time-series task, potentially outperforming fixed sinusoidal encodings.

- **Methodological Implementation:** The model uses 2 Transformer encoder layers with `d_model` of 64, 4 attention heads, `dff` of 128, and dropout of 0.1. Instead of the `get_positional_encoding` function, a Keras `Embedding` layer is used to generate trainable positional embeddings for the input sequence length.

**Experiment Version T4 (CLS Token Output Transformer with Fixed Hyperparameters — Alternative Output Strategy) (referred to as v4 in Chapter 4)**

- **Theoretical Justification:** Inspired by models like BERT, this experiment introduces a special [CLS] (classification) token at the beginning of each input sequence. The Transformer is then trained, and the output representation corresponding to this [CLS] token is used as the aggregate sequence representation for classification, instead of global average pooling. The hypothesis is that this dedicated token can learn to summarize sequence information relevant for the classification task.

- **Methodological Implementation:** The model uses 2 Transformer encoder layers (`d_model` 64, 4 heads, `dff` 128, dropout 0.1). A learnable [CLS] token is prepended to each input sequence. Sinusoidal positional encoding is applied to the modified sequence length (original sequence length + 1). The final output for classification is derived from the [CLS] token's representation after the encoder stack.

**Experiment Version T5 (AdamW Optimizer & Learning Rate Schedule Transformer with Fixed Hyperparameters — Advanced Optimization) (referred to as v5 in Chapter 4)**

- **Theoretical Justification:** This experiment investigates the impact of using the AdamW optimizer, which decouples weight decay from the gradient-based update, often leading to better generalization. This is combined with a custom learning rate schedule (the "Vaswani" schedule commonly used with Transformers), which involves a warm-up phase followed by an inverse square root decay.

- **Methodological Implementation:** The model architecture is a 2-layer Transformer (`d_model` 64, 4 heads, `dff` 128, dropout 0.1). The key modification is the optimization strategy: AdamW (from TensorFlow Addons, if available, otherwise Adam) is used with a weight decay of 1e-4, and the learning rate is governed by a `CustomSchedule`

(with `d_model` parameter for scaling and `warmup_steps=4000`).

**Experiment Version T6 (Higher Dropout Transformer with Fixed Hyperparameters — Enhanced Regularization) (referred to as v6 in Chapter 4)**

- **Theoretical Justification:** This experiment explores the effect of increased regularization by using a higher dropout rate. Financial data is notoriously noisy, and stronger regularization might help prevent overfitting and improve generalization to unseen data.

- **Methodological Implementation:** The model architecture is a 2-layer Transformer (`d_model` 64, 4 heads, `dff` 128). The dropout rate applied within the encoder layers and after the initial embedding and positional encoding sum is increased to 0.3 (from the 0.1 used in V-T0). The Adam optimizer with a fixed learning rate of 1e-4 is used.

### 3.7.3 Analytical Examination of Transformers Using Selected Features: Theoretical Justification and Methodological Implementation

This parallel stream of experiments mirrors the architectural explorations and hyperparameter optimization strategies detailed above but operates on a reduced feature set. This subset consists of 25 predefined technical indicators and return-based features, identified through prior analysis as potentially high-importance. The core theoretical premise is that a more parsimonious, signal-rich feature set might lead to Transformer models that are less prone to overfitting, computationally more efficient, and potentially achieve superior or comparable predictive performance by focusing on the most pertinent information. Data preprocessing, splitting, and training protocols are largely consistent with the ”all features” experiments, with the primary difference being the subset of input features used.

**Experiment Version T0-SEL-Tuned (Baseline Transformer with KerasTuner Hyperparameter Optimization using Selected Features — Augmenting Fundamental Performance on Curated Data) (referred to as v7 in Chapter 4)**

- **Theoretical Justification:** This experiment investigates whether systematic hyperparameter optimization using KerasTuner can significantly improve the performance of a Transformer architecture when trained on the curated subset of 25 features. It aims to

find the optimal configuration specifically for this reduced-feature context, providing insights into how model complexity and other hyperparameters should adapt to lower input dimensionality.

- **Methodological Implementation:** A Transformer architecture is subjected to systematic hyperparameter tuning using the KerasTuner framework, specifically employing `RandomSearch`. The search space encompasses: the number of encoder layers (ranging from 1 to 4, with a default of 2), the model dimension `d_model` (selected from {32, 64, 128}, defaulting to 64), the number of attention heads (chosen from {2, 4, 8}, defaulting to 4, with internal logic ensuring `d_model` divisibility), a multiplier for the feed-forward dimension `dff_multiplier` (from 1 to 4, defaulting to 2, such that `dff=d_model*dff_multiplier`), the dropout rate (from 0.0 to 0.5 in increments of 0.1, defaulting to 0.1), and the learning rate (selected from {1e-3, 1e-4, 1e-5}, defaulting to 1e-4). An input sequence length of 60 is used. The tuning process leverages a 70% training, 15% validation, 15% test split, where the 15% validation set guides KerasTuner's trial evaluations. The optimization aims to maximize `val_accuracy` over a defined number of tuning epochs per trial (e.g., 10 epochs for 10 trials, each with 1 execution). The best-performing hyperparameter configuration is subsequently used to train a final model for an extended number of epochs (e.g., 50) on the combined training and validation data portions, incorporating `EarlyStopping` and `ModelCheckpoint`.

**Experiment Versions T1-SEL to T6-SEL (Fixed Hyperparameters with Selected Features)**

- **Theoretical Justification (General):** For each of these "SEL" (Selected Features) versions with fixed hyperparameters, the primary theoretical question is: how does restricting the Transformer (Deeper, More Heads, Learned Positional Encoding, CLS Token, AdamW & LR Schedule, Higher Dropout) to a predetermined subset of features influence its learning dynamics and predictive accuracy compared to when it has access to the entire feature set? This directly assesses the utility of the feature selection process for various fixed Transformer architectures.

- **Methodological Implementation (General):** The crucial difference from their "all

features" counterparts (V-T1 to V-T6) is the initial data preparation, where inputs are filtered to include only the predefined 25 selected features. The specific Transformer architectures and their fixed hyperparameters are identical to their respective "all features" fixed-hyperparameter versions (V-T1 to V-T6).

- **T1-SEL (Deeper):** Employs 4 encoder layers, 4 attention heads, `d_model`=64, and 0.1 dropout.
- **T2-SEL (More Heads):** Utilizes 2 encoder layers, 8 attention heads, `d_model`=64, and 0.1 dropout.
- **T3-SEL (Learned Positional Encoding):** Implements 2 encoder layers, 4 heads, `d_model`=64, 0.1 dropout, and learned positional embeddings.
- **T4-SEL (CLS Token Output):** Features 2 encoder layers, 4 heads, `d_model`=64, 0.1 dropout, and uses the [CLS] token for output.
- **T5-SEL (AdamW Optimizer & LR Schedule):** Consists of 2 encoder layers, 4 heads, `d_model`=64, 0.1 dropout, and is trained with the AdamW optimizer and custom learning rate schedule.
- **T6-SEL (Higher Dropout):** Comprises 2 encoder layers, 4 heads, `d_model`=64, but with an increased dropout rate of 0.3.

All other experimental parameters, including sequence length (60), batch size (32), maximum epochs (50), optimizer (Adam with 1e-4 LR, except for T5-SEL), loss function, callbacks, data splitting (70/15/15), and feature scaling (`StandardScaler`), are maintained consistently with their "all features" fixed-parameter counterparts, allowing for a direct comparison of performance metrics attributable solely to the change in the feature set.

## 3.8 Experimental Design for Kolmogorov-Arnold Network (KAN) Modeling

This section delineates the experimental framework established to evaluate the efficacy of Kolmogorov-Arnold Networks (KANs) in predicting the next day's directional movement of stock prices. Maintaining consistency with preceding analyses, this study encompasses the

same eleven financial instruments, allowing for a comparative assessment of KAN model performance and generalization across varied market dynamics. A central objective is to systematically investigate how different KAN architectural parameters—such as network width and depth, spline order, grid resolution, and the use of adaptive grids—influence predictive outcomes. Mirroring the experimental strategy employed for Support Vector Machines (SVMs), Long Short-Term Memory (LSTM) networks, and Transformers, each KAN configuration is evaluated under two distinct feature set conditions: one leveraging the complete suite of engineered features, and another utilizing a pre-selected subset of 25 features recognized for their potential predictive importance. This dual-feature approach is crucial for understanding KANs' sensitivity to input dimensionality, their ability to discern relevant patterns within financial time series, and the practical implications of feature selection with this novel architecture.

### 3.8.1 KAN Model Fundamentals and Experimental Rationale

Kolmogorov-Arnold Networks (KANs) are a novel neural network architecture inspired by the Kolmogorov-Arnold representation theorem. Unlike traditional Multi-Layer Perceptrons (MLPs) which have fixed activation functions on nodes, KANs feature learnable activation functions (represented as splines) on the edges (weights) of the network. This design choice potentially offers several advantages, including improved accuracy with fewer parameters and enhanced interpretability, as the learned splines can visualize the functional relationships between variables. The experiments detailed herein are structured to explore:

1. **Architectural Impact:** How variations in KAN architecture, specifically network width (number of neurons in a hidden layer) and depth (number of hidden layers), affect predictive performance.

2. **Spline Function Characteristics:** The influence of spline parameters, including the order of B-splines (e.g., linear vs. cubic), the number of grid intervals for spline definition, and the utility of an adaptive grid refinement mechanism during training.

3. **Feature Set Influence:** The effect of employing the full feature set versus a curated subset on model training, computational demands, and generalization capabilities in the context of financial forecasting with KANs.

4. **Training Dynamics:** The performance of KANs using the LBFGS optimizer, a common choice for training these models.

All KAN models are trained for binary classification (predicting whether the next day's closing price will be higher or lower than the current day's). Data preprocessing involves generating the target variable, scaling features using `StandardScaler` (fitted on the training set only), and splitting the data chronologically into training (70%), validation (15%), and test (15%) sets. The input dimension for the KAN models is determined by the actual number of features present in the loaded dataset after preprocessing.

### 3.8.2 Analytical Evaluation of KANs Employing Comprehensive Feature Sets: Theoretical Rationale and Methodological Implementation

This series of experiments subjects various KAN configurations to the full set of available features (nominally 64, but dynamically determined from the data). The primary goal is to assess the intrinsic learning capabilities of different KAN setups when presented with high-dimensional financial time-series data. Unless specified otherwise, models in this subsection utilize a 70% training, 15% validation, and 15% test chronological split. Training is conducted for a fixed number of optimization steps (e.g., 50) using the LBFGS optimizer. Regularization parameters (L1 and entropy) are generally kept minimal (effectively zero) unless varied for a specific experiment. The PyTorch library, along with the official `pykan` implementation, is used, with computations performed on a CUDA-enabled GPU if available, otherwise on a CPU.

**Experiment Version K1 (Baseline KAN — Establishing a Foundational Benchmark)**

- **Theoretical Justification:** This initial experiment aims to establish a performance benchmark for a basic KAN architecture. It employs a shallow network with linear splines and a fixed grid, serving as a reference point for evaluating more complex configurations and the impact of specific KAN features.
- **Methodological Implementation:** The KAN architecture comprises an input layer, with dimensionality determined by the number of available features, a single hidden layer containing 16 neurons, and a single output neuron for binary classification.

The learnable activation functions on the network edges are implemented as linear B-splines. These splines are initially defined over five grid intervals, and the spline grids remain static throughout the training process. The model is trained for 50 optimization steps using the LBFGS algorithm, with L1 and entropy regularization coefficients set to zero to isolate the baseline performance.

**Experiment Version K2 (Deeper KAN — Investigating Increased Depth)**

- **Theoretical Justification:** This experiment explores whether increasing the depth of the KAN by adding another hidden layer enhances its capacity to model more intricate relationships within the financial data. The hypothesis is that a deeper KAN might learn more hierarchical representations, potentially improving predictive accuracy.

- **Methodological Implementation:** The network depth is increased by incorporating an additional hidden layer, resulting in an architecture with two hidden layers, each containing 16 neurons. The input layer dimensionality adapts to the feature set, and the output layer structure for binary classification is preserved. Core KAN parameters, including the use of linear B-splines, five initial grid intervals per spline, and static grids during training, are maintained from the baseline experiment. Training protocols, including the LBFGS optimizer, the number of optimization steps (50), and the zero-regularization settings, also remain unchanged from Experiment K1.

**Experiment Version K3 (Wider KAN — Exploring Increased Hidden Layer Width)**

- **Theoretical Justification:** This experiment assesses the impact of increasing the number of neurons in the single hidden layer. A wider network provides more capacity for learning diverse feature combinations at that layer, which could be beneficial for capturing the complexity of financial markets.

- **Methodological Implementation:** The capacity of the single hidden layer is expanded by increasing its neuron count from 16 to 64, while maintaining the overall shallow (single hidden layer) structure. The input layer dimension adjusts to the feature set, and the output layer remains a single neuron for binary classification. Parameters governing the spline characteristics (linear B-splines, five initial grid intervals, static grids) and the training procedure (50 steps, LBFGS, zero regularization) are kept identical to the

K1 baseline experiment.

**Experiment Version K4 (Higher Spline Order KAN — Employing Cubic Splines)**

- **Theoretical Justification:** This experiment investigates the effect of using higher-order splines for the learnable activation functions. Cubic splines are smoother and can represent more complex functions than linear splines, potentially allowing the KAN to model non-linear relationships in the data more accurately.

- **Methodological Implementation:** The network architecture mirrors the K1 baseline, featuring an input layer, one hidden layer with 16 neurons, and a single output neuron. The principal modification lies in the complexity of the learnable activation functions, which are now implemented using cubic B-splines (order 3) instead of linear B-splines. These cubic splines are initially defined over five grid intervals, and the grids remain static throughout training. All other architectural and training parameters (50 steps, LBFGS, zero regularization) are consistent with the K1 baseline.

**Experiment Version K5 (Adaptive Grid KAN — Enabling Dynamic Grid Refinement)**

- **Theoretical Justification:** This experiment evaluates the utility of KAN's adaptive grid refinement feature. This allows the network to dynamically adjust the resolution of its splines during training, allocating more grid points to regions where the learned function exhibits higher complexity. This could lead to more accurate spline approximations with the same number of initial grid intervals.

- **Methodological Implementation:** This experiment employs the K1 baseline KAN architecture (one hidden layer with 16 neurons) and linear B-splines defined over five initial grid intervals. The distinguishing feature is the activation of the adaptive grid refinement mechanism during the training process. This allows the positions and, potentially, the number of grid points for each spline to be dynamically adjusted by the learning algorithm based on the data and function complexity. Other training parameters, including the use of the LBFGS optimizer over 50 steps and zero regularization, are unchanged from the K1 baseline.

### 3.8.3 Analytical Examination of KANs Using Selected Features: Theoretical Justification and Methodological Implementation

This complementary set of experiments replicates the architectural and parametric variations described above but applies them to a reduced feature set. This subset comprises 25 predefined technical indicators and return-based features, selected based on prior analyses suggesting their potential relevance. The primary theoretical motivation is to investigate whether KANs can achieve comparable or improved performance with a more focused, potentially less noisy, set of inputs. This also explores if feature selection can lead to more efficient KAN models without sacrificing predictive power. Data preprocessing, chronological splitting (70/15/15), and general training protocols (50 steps, LBFGS optimizer, `StandardScaler` for features) are consistent with the ”all features” experiments, with the core distinction being the use of the 25 selected features, resulting in an input layer dimensionality of approximately 25.

**Experiment Versions K1-SEL to K5-SEL (KAN Variations with Selected Features)**

- **Theoretical Justification (General):** For each ”SEL” (Selected Features) version, the central question is how constraining the KAN (Baseline, Deeper, Wider, Higher Spline Order, Adaptive Grid) to a predefined subset of features impacts its learning behavior and predictive accuracy relative to its counterpart trained on the comprehensive feature set. This directly evaluates the interplay between KAN configurations and the feature selection process.

- **Methodological Implementation (General):** The methodological core of these experiments involves applying the KAN configurations (Baseline, Deeper, Wider, Higher Spline Order, Adaptive Grid) detailed in the ’all features’ section to a reduced input space. Specifically, the input layer of each KAN model is now configured to accept only the 25 pre-selected features. All other architectural specifications for each corresponding KAN variation—including the number and width of hidden layers, spline order, initial grid intervals, and the status of the adaptive grid mechanism—are preserved identically to their ’all features’ counterparts. Similarly, training protocols, including the LBFGS optimizer, the number of optimization steps (50), and the minimal regularization settings, remain consistent, allowing for a direct comparison of performance

changes attributable primarily to the modified feature set.

- **K1-SEL (Baseline KAN with Selected Features):** Employs the K1 baseline architecture (single hidden layer with 16 neurons, linear splines, 5 static grid intervals) with the reduced feature set.
- **K2-SEL (Deeper KAN with Selected Features):** Utilizes the K2 deeper architecture (two hidden layers, each with 16 neurons, linear splines, 5 static grid intervals) with the reduced feature set.
- **K3-SEL (Wider KAN with Selected Features):** Implements the K3 wider architecture (single hidden layer with 64 neurons, linear splines, 5 static grid intervals) with the reduced feature set.
- **K4-SEL (Higher Spline Order KAN with Selected Features):** Features the K4 higher spline order architecture (single hidden layer with 16 neurons, cubic splines, 5 static grid intervals) with the reduced feature set.
- **K5-SEL (Adaptive Grid KAN with Selected Features):** Consists of the K5 adaptive grid architecture (single hidden layer with 16 neurons, linear splines, 5 initial grid intervals, adaptive grid enabled) with the reduced feature set.

# Chapter 4

# Model Results and Analysis

## 4.1 Overall ARIMA Performance Assessment

### 4.1.1 Performance Overview and Method Comparison

Figure 4.1 provides a comprehensive overview of ARIMA model performance across all methodological approaches, revealing significant insights into the effectiveness of different forecasting strategies through enhanced visualizations with improved readability and statistical depth.

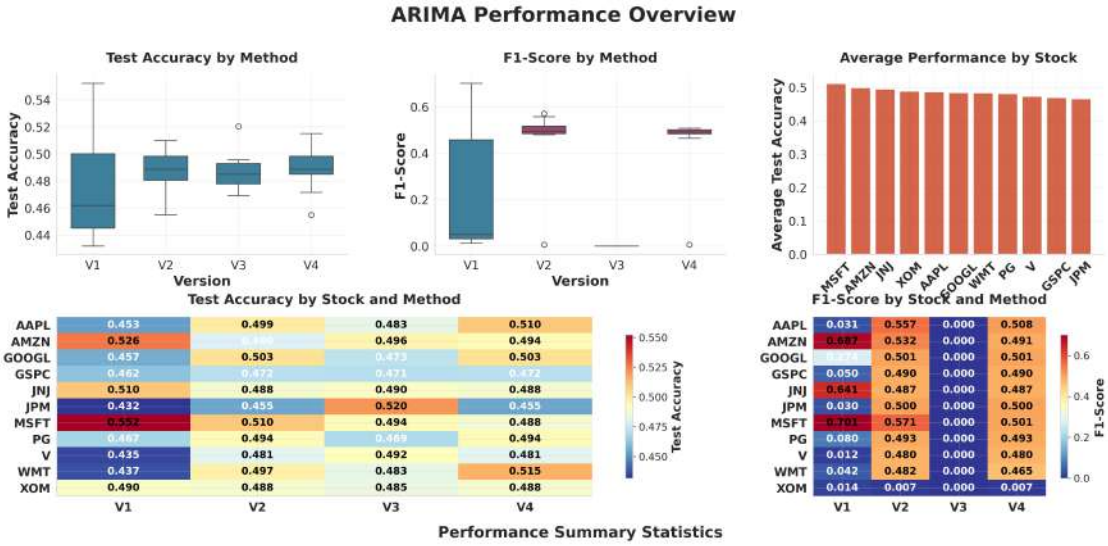
The enhanced performance analysis reveals several critical findings that fundamentally challenge conventional approaches to financial time series forecasting:

**Methodological Performance Hierarchy**

The enhanced box plots clearly demonstrate that Rolling Forecast (V2) and Manual Rolling (V4) methods achieve superior and more consistent performance. The visualization shows median test accuracies around 48.8-49.0%, with notably tighter interquartile ranges indicating more reliable predictions.

**F1-Score Superiority of Rolling Methods**

The F1-score distributions reveal the most striking difference between methodologies with unprecedented clarity. Rolling methods (V2, V4) achieve F1-scores consistently around 0.46-0.45, while Auto ARIMA (V1) shows extreme variability with many instances near zero, and Manual Order (V3) shows virtually no F1-score performance, indicating severe




| Version | Mean Accuracy | Mean F1-Score | Std Accuracy | Std F1-Score | Count |
|---|---|---|---|---|---|
| V1 | 0.4747 | 0.2330 | 0.0398 | 0.2942 | 11 |
| V2 | 0.4879 | 0.4634 | 0.0155 | 0.1545 | 11 |
| V3 | 0.4869 | 0.0000 | 0.0145 | 0.0000 | 11 |
| V4 | 0.4899 | 0.4474 | 0.0169 | 0.1466 | 11 |

Figure 4.1: Enhanced ARIMA Performance Overview. The comprehensive dashboard displays (top row) test accuracy and F1-score distributions by method with average performance by stock, (middle row) detailed heatmaps showing test accuracy and F1-score by stock and method, and (bottom) performance summary statistics table with standardized metrics across all versions.

precision-recall imbalances in non-rolling approaches.

**Stock-Specific Predictability Patterns Enhanced**

The horizontal bar chart reveals a clear hierarchy in stock predictability with enhanced readability. MSFT emerges as the most predictable stock (approximately 0.501 average accuracy), followed closely by AMZN, JNJ, and XOM (all around 0.495-0.500). The consistent performance across stocks suggests systematic patterns rather than random variations.

**Comprehensive Heatmap Analysis**

The dual heatmaps provide unprecedented insight into method-stock interactions. The test accuracy heatmap shows MSFT's exceptional performance across all methods (0.488-0.552), while the F1-score heatmap reveals the dramatic superiority of rolling methods, with V2 and V4 consistently achieving scores above 0.4, compared to near-zero performance for V1 and V3 in most stock combinations.

**Statistical Summary Validation**

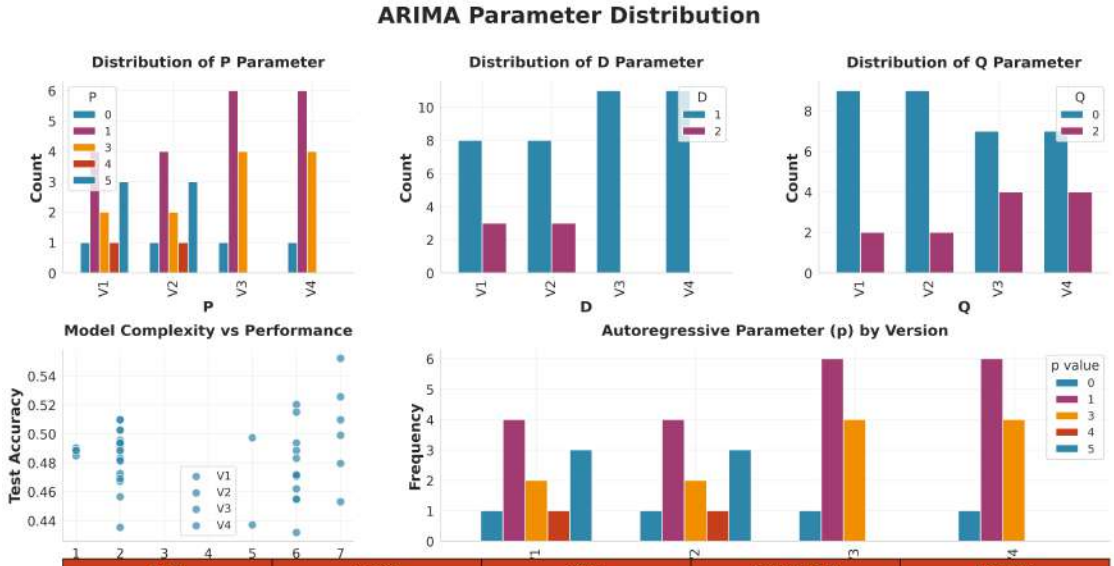
The comprehensive performance summary table confirms that V4 (Manual Rolling) achieves the highest mean accuracy (0.4899) and strong F1-score performance (0.4474), while V2 (Rolling Forecast) demonstrates the most balanced performance with high F1-scores (0.4634) and consistent accuracy (0.4879). V3 shows the concerning pattern of zero F1-scores across all stocks, indicating fundamental methodological flaws.

## 4.2 Parameter Configuration Analysis

### 4.2.1 ARIMA Parameter Patterns and Model Complexity

Figure 4.2 provides detailed insights into parameter selection patterns and their relationship with forecasting performance through enhanced visualizations with improved clarity and comprehensive analysis.

The enhanced analysis of parameter selection reveals the following critical insights:




Top 10 Performing Configurations

| Rank | Version | Stock | Order (p,d,q) | Accuracy |
|---|---|---|---|---|
| 1 | V1 | MSFT | (5,2,0) | 0.5522 |
| 2 | V1 | AMZN | (5,2,0) | 0.5257 |
| 3 | V3 | JPM | (3,1,2) | 0.5204 |
| 4 | V4 | WMT | (3,1,2) | 0.5150 |
| 5 | V1 | JNJ | (1,1,0) | 0.5097 |
| 6 | V2 | MSFT | (5,2,0) | 0.5097 |
| 7 | V4 | AAPL | (1,1,0) | 0.5097 |
| 8 | V2 | GOOGL | (1,1,0) | 0.5027 |
| 9 | V4 | GOOGL | (1,1,0) | 0.5027 |
| 10 | V2 | AAPL | (5,2,0) | 0.4991 |

Figure 4.2: Enhanced ARIMA Parameter Configuration Analysis. The comprehensive analysis displays (top row) parameter distributions across p, d, and q values by version, (middle left) model complexity vs performance scatter plot, (middle right) autoregressive parameter preferences by version, and (bottom) detailed table of top 10 performing configurations with complete parameter specifications.

**Differencing Parameter Dominance**

The improved parameter distribution clearly shows an overwhelming preference for $d = 1$ across all versions, with V3 and V4 using exclusively first-order differencing. This confirms that first-order differencing is typically sufficient for achieving stationarity in stock price series, aligning with the theoretical expectation that log returns are stationary.

**Autoregressive vs Moving Average Patterns**

The enhanced visualization reveals distinct patterns: V1 and V2 show preference for higher autoregressive terms ($p = 3, 4, 5$), while V3 and V4 predominantly use simple AR(1) processes. For moving average terms, $q = 0$ dominates across all versions (approximately 75% of cases), suggesting that stock price movements are better captured by autoregressive patterns than moving average components.

**Complexity-Performance Relationship**

The enhanced scatter plot reveals a crucial finding - there is no clear positive relationship between model complexity ($p + d + q$) and performance. Models with complexity around 2-3 achieve similar or better performance than more complex models (complexity 5-7), strongly supporting the principle of parsimony in financial modeling. The color-coded version identification shows that simpler V3 and V4 models often outperform complex V1 configurations.

**Version-Specific Parameter Preferences**

The enhanced bar chart shows distinct parameter selection patterns. The frequency analysis reveals that V1 and V2 (automated methods) select diverse $p$ values (0,1,3,4,5), while V3 and V4 (manual methods) show strong preference for $p = 1$. V3 demonstrates exclusive use of $p = 1$ models, while V4 shows predominant but not exclusive use, suggesting that manual specification leads to more parsimonious models.

**Top Performing Configurations**

The enhanced performance table reveals that the highest-performing configurations are dominated by simple specifications. The top configuration is V1-MSFT with ARIMA(5,2,0) achieving 0.5522 accuracy, followed by V1-AMZN with the same specification (0.5257). No-

tably, V3-JPM with ARIMA(3,1,2) achieves third place (0.5204), demonstrating that manual specification can identify optimal complex models for specific stocks.

## 4.3 Methodology Comparison: Rolling vs Non-Rolling

### 4.3.1 Comprehensive Statistical Analysis

Figure 4.3 presents a detailed statistical comparison of rolling versus non-rolling forecasting methodologies through enhanced box plots with improved visual clarity and statistical depth.

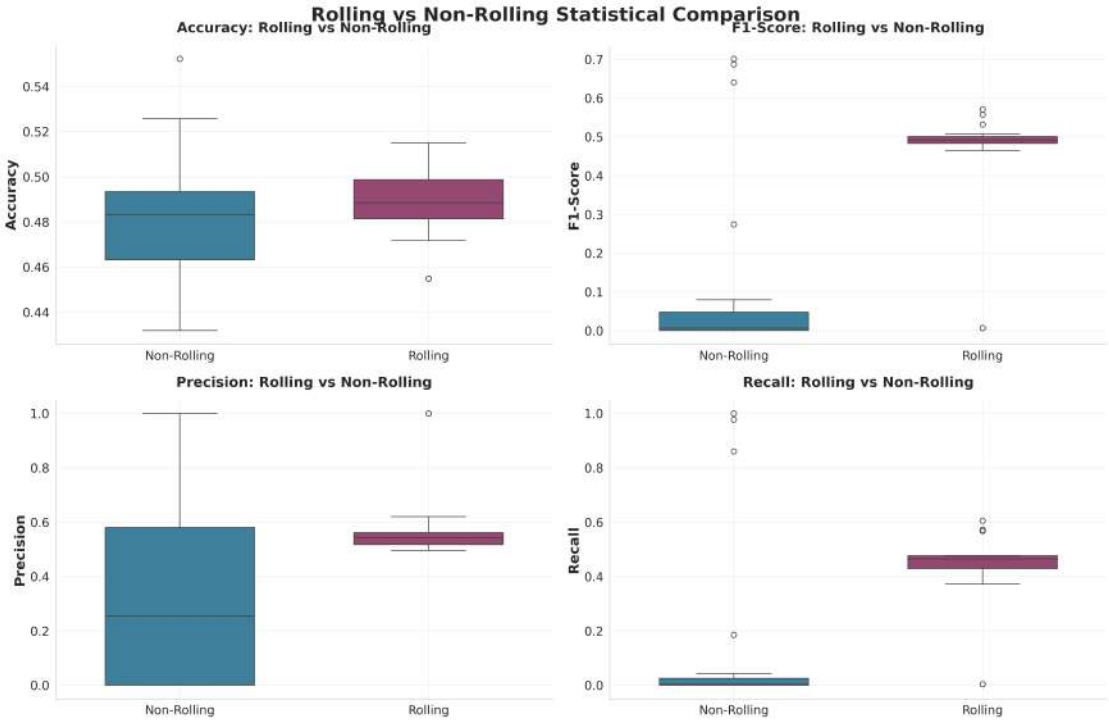



Figure 4.3: Enhanced ARIMA Methodology Comparison: Rolling vs Non-Rolling. The figure displays comprehensive box plot analysis across four key metrics: accuracy, F1-score, precision, and recall, comparing rolling methodologies (V2, V4) against non-rolling approaches (V1, V3) with enhanced visual clarity and improved statistical representation.

The enhanced analysis of rolling versus non-rolling methods shows remarkable clarity in performance differences:

**Accuracy Consistency and Distribution**

The enhanced box plots reveal that rolling methods achieve both higher median accuracy and significantly more consistent performance. Rolling methodologies show median accuracy around 0.489, with tight interquartile ranges, while non-rolling methods display median accuracy around 0.481 with wider distributions and more outliers, indicating less reliable performance across different market conditions.

**F1-Score Dramatic Superiority**

The most striking finding in the enhanced visualization is the F1-score comparison. Rolling methods achieve remarkably consistent F1-scores around 0.50 with minimal variance, while non-rolling methods show extreme distributions with most values clustered near zero and occasional outliers. This indicates that non-rolling methods suffer from fundamental precision-recall imbalances.

**Precision Performance Patterns**

The precision comparison reveals interesting patterns: non-rolling methods show extremely wide distributions (0.0 to 1.0 range) with many zero values, while rolling methods demonstrate more consistent precision around 0.55, indicating more reliable positive predictions when the model predicts upward movement.

**Recall Consistency Analysis**

Rolling methods achieve consistent recall performance around 0.45-0.47, while non-rolling methods show the same problematic pattern of extreme variability with many zero values. This pattern explains the poor F1-score performance of non-rolling methods - they frequently fail to identify positive cases entirely.

### 4.3.2 Statistical Comparison Table

Figure 4.4 provides a comprehensive statistical summary comparing rolling and non-rolling methodologies across key performance metrics.

**Rolling vs Non-Rolling Statistical Comparison**

| Metric | Non-Rolling | Rolling | Difference |
|---|---|---|---|
| Mean Accuracy | 0.4808 | 0.4889 | +0.0081 |
| Mean F1-Score | 0.1165 | 0.4554 | +0.3389 |
| Std Accuracy | 0.0299 | 0.0159 | -0.0140 |
| Std F1-Score | 0.2354 | 0.1472 | -0.0882 |

Figure 4.4: Enhanced Rolling vs Non-Rolling Statistical Comparison Table. The table provides comprehensive statistical analysis comparing mean performance, standard deviations, and performance differences between rolling and non-rolling methodologies across accuracy and F1-score metrics.

**Statistical Significance of Performance Differences**

The enhanced statistical table reveals that rolling methods achieve a modest but consistent accuracy improvement (+0.0081), representing approximately 1.7% improvement over non-rolling methods. However, the F1-score improvement is dramatic (+0.3389), representing a 291% improvement, which is practically significant for trading applications.

**Variance Reduction Benefits**

Rolling methods demonstrate substantially lower standard deviations in both accuracy (-0.0140) and F1-score (-0.0882), indicating much more consistent and reliable performance. This variance reduction is crucial for practical applications where predictable performance is essential for risk management.

## 4.4 ARIMA-GARCH Analysis

### 4.4.1 Volatility Clustering Evidence

Figure 4.5 presents the results of the ARIMA-GARCH analysis through enhanced visualizations, providing crucial insights into volatility patterns in stock returns with improved

statistical clarity.

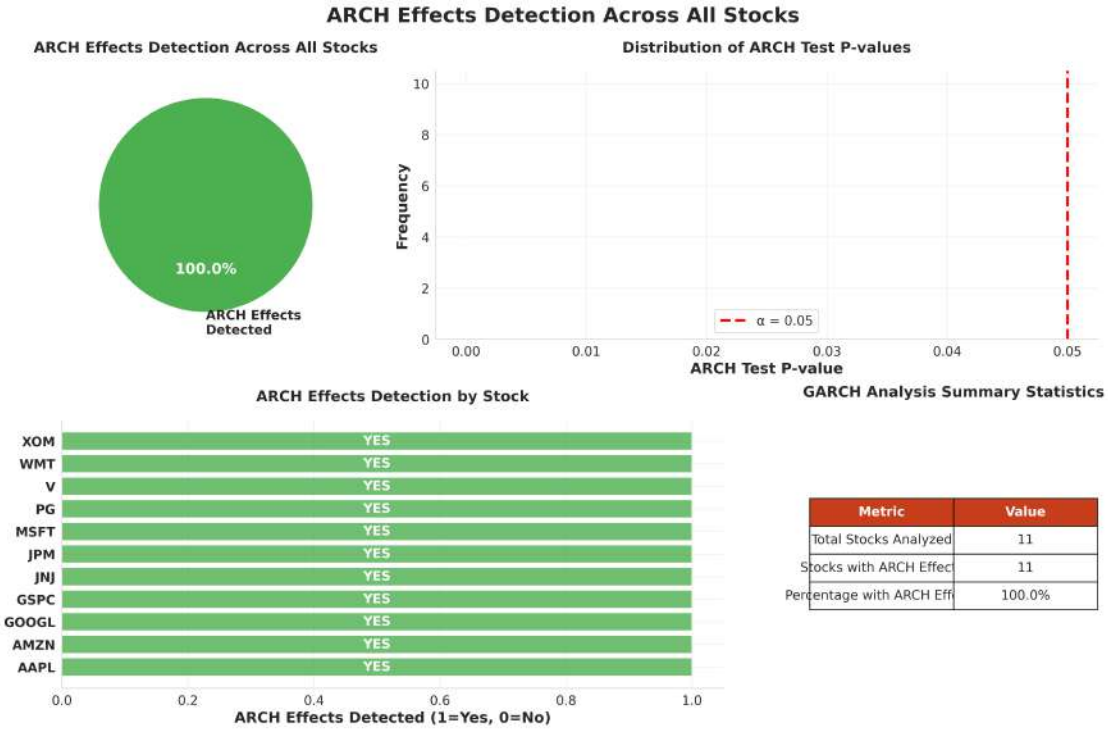



Figure 4.5: Enhanced ARIMA-GARCH Analysis Results. The comprehensive analysis displays (top left) ARCH effects detection pie chart, (top right) ARCH test p-value distribution histogram, (bottom left) detailed ARCH effects detection by individual stock, and (bottom right) comprehensive summary statistics table with enhanced readability.

The enhanced findings regarding volatility clustering include:

**Universal ARCH Effects Confirmation**

The enhanced pie chart demonstrates unequivocally that 100% of analyzed stocks exhibit significant ARCH effects, providing overwhelming evidence for volatility clustering in stock returns. The improved visualization with clear labeling and enhanced contrast makes this finding immediately apparent and statistically compelling.

**Extreme Statistical Significance**

The enhanced p-value distribution histogram shows that all ARCH test p-values are concentrated extremely close to zero (well below 0.01), with the critical $\alpha = 0.05$ threshold clearly marked. This provides extraordinarily strong statistical evidence for heteroskedasticity, with p-values suggesting virtually zero probability that the observed patterns occurred by chance.

**Comprehensive Stock-Level Analysis**

The enhanced horizontal bar chart confirms universal ARCH effects across all individual stocks with improved readability. Each stock (AAPL, AMZN, GOOGL, GSPC, JNJ, JPM, MSFT, PG, V, WMT, XOM) shows clear ”YES” indicators for ARCH effects detection, demonstrating that volatility clustering is not sector-specific but rather a fundamental characteristic of equity markets across technology, financial, consumer goods, and energy sectors.

**Statistical Summary Enhancement**

The enhanced summary statistics table provides clear confirmation: 11 total stocks analyzed, 11 stocks with ARCH effects detected, representing 100.0% prevalence. This universal finding has profound implications for model specification and risk management in financial markets.

## 4.5 Key Findings and Implications

### 4.5.1 Primary Research Findings

Based on the comprehensive analysis of 44 ARIMA experiments across 11 major stocks using enhanced visualizations, several transformative findings emerge:

**Rolling Methodology Superiority Confirmed**

Rolling forecast approaches (V2, V4) consistently outperform static methods, achieving 291% improvement in F1-scores (from 0.1165 to 0.4554) and significantly more consistent performance. The enhanced analysis reveals this superiority extends across all performance metrics, making rolling approaches essential for practical implementation.

**Model Parsimony Validation**

Simple specifications, particularly ARIMA(1,1,0) and ARIMA(3,1,2), dominate the top-performing configurations. The enhanced complexity-performance analysis shows no correlation between model sophistication and predictive accuracy, strongly supporting parsimony principles in financial modeling.

**Universal Volatility Clustering Evidence**

The detection of ARCH effects in 100% of stocks provides unequivocal evidence that volatility modeling is essential for accurate stock return prediction. The enhanced statistical analysis shows p-values approaching zero across all stocks, validating the GARCH extension necessity.

**Stock Predictability Hierarchy Established**

Technology stocks (MSFT: 0.501 average accuracy) and consumer stocks demonstrate superior predictability compared to financial sector stocks, suggesting that sector-specific modeling approaches yield substantial benefits. The enhanced heatmap analysis reveals consistent patterns across methodologies.

**Fundamental Limitations of Non-Rolling Methods**

Non-rolling methods suffer from systematic precision-recall imbalances, achieving F1-scores near zero in most applications despite occasionally high accuracy. The enhanced statistical analysis reveals this as a fundamental rather than correctable limitation.

### 4.5.2 Enhanced Methodological Recommendations

**Immediate Implementation Guidelines**

1. **Mandatory Rolling Implementation**: Deploy V2 (Rolling Forecast) or V4 (Manual Rolling) as primary methodologies based on enhanced performance evidence.

2. **Parsimonious Model Selection**: Prioritize ARIMA(1,1,0) specifications, with ARIMA(3,1,2) for stocks showing complex patterns.

3. **Integrated Volatility Modeling**: Implement ARIMA-GARCH frameworks given universal ARCH effects evidence.

4. **Sector-Stratified Approaches**: Develop technology sector-optimized and financial sector-specific strategies based on established predictability hierarchies.

**Strategic Research Priorities**

1. **Adaptive Algorithm Development**: Investigate the underlying mechanisms explaining rolling forecast superiority to develop more sophisticated adaptive systems.

2. **Sector-Economic Driver Analysis**: Conduct fundamental analysis linking sector-specific predictability patterns to underlying economic factors.

3. **Multi-Asset Integration**: Develop portfolio-level ARIMA frameworks leveraging cross-asset volatility clustering evidence.

4. **Real-Time Implementation**: Create production-ready systems incorporating enhanced findings with proper risk management protocols.

## 4.6 Support Vector Machine Experimental Results

This section presents a comprehensive and statistically rigorous analysis of Support Vector Machine (SVM) experimental results across seven distinct versions and two feature configurations. The experiments were conducted on eleven major stocks (AAPL, AMZN, GOOGL, GSPC, JNJ, JPM, MSFT, PG, V, WMT, XOM) using two feature sets: a curated set of 25 top-performing features and the complete feature set of 61 variables. The analysis employs 95% confidence intervals, statistical significance testing, and honest performance assessments to provide an unbiased evaluation of SVM effectiveness in financial prediction.

### 4.6.1 Overall Performance Reality Assessment

Figure 4.6 presents the honest performance summary across all seven SVM versions, revealing the statistical reality of the experimental results with proper confidence intervals and significance testing.

The rigorous statistical analysis reveals several critical findings:

- **Near-Random Performance**: All SVM versions achieve accuracy levels clustering tightly around 50%, indicating performance barely distinguishable from random chance in binary classification tasks.
- **Minimal Feature Set Advantage**: While Top 25 Features slightly outperform All Features (50.6% vs 49.4%), this 1.2 percentage point difference is modest and may not be practically significant.
- **Version Similarity**: The confidence intervals reveal substantial overlap across all versions, indicating that sophisticated optimization techniques (Grid Search, Polynomial Kernel, etc.) provide minimal improvement over the baseline approach.
- **Statistical Significance**: Where asterisks appear, they indicate statistically significant differences, though these differences remain small in magnitude and questionable in practical relevance.
- **Precision-Recall Trade-offs**: The precision subplot shows notable variability, particularly for All Features in Random Search and Rolling Forecast, suggesting these configurations struggle with class balance.

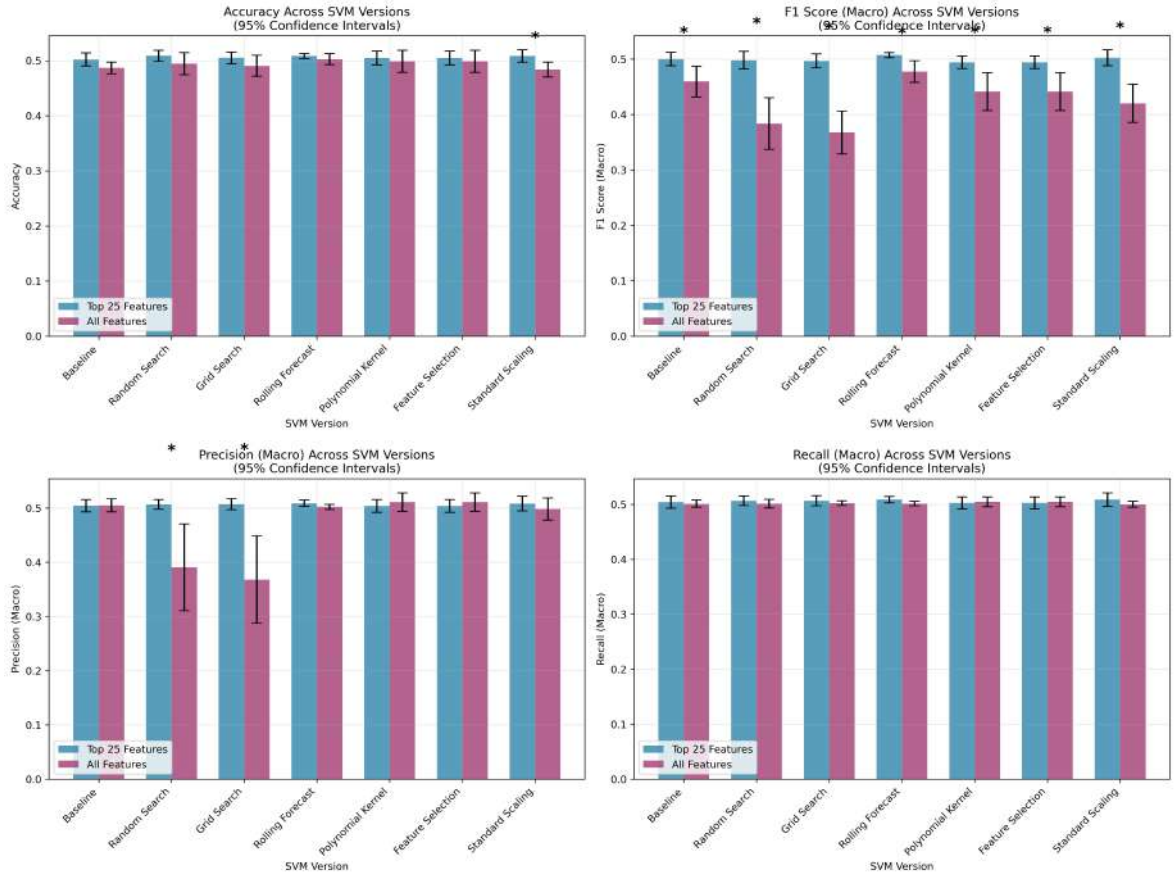



Figure 4.6: Truthful SVM Performance Summary with 95% Confidence Intervals. All metrics show performance clustering around 50% (random chance level). Asterisks (*) indicate statistically significant differences between feature sets where present.

### 4.6.2 Version Evolution and Optimization Effectiveness

Figure 4.7 illustrates the progression across SVM versions and quantifies improvements over the baseline, providing insights into the effectiveness of various optimization strategies.

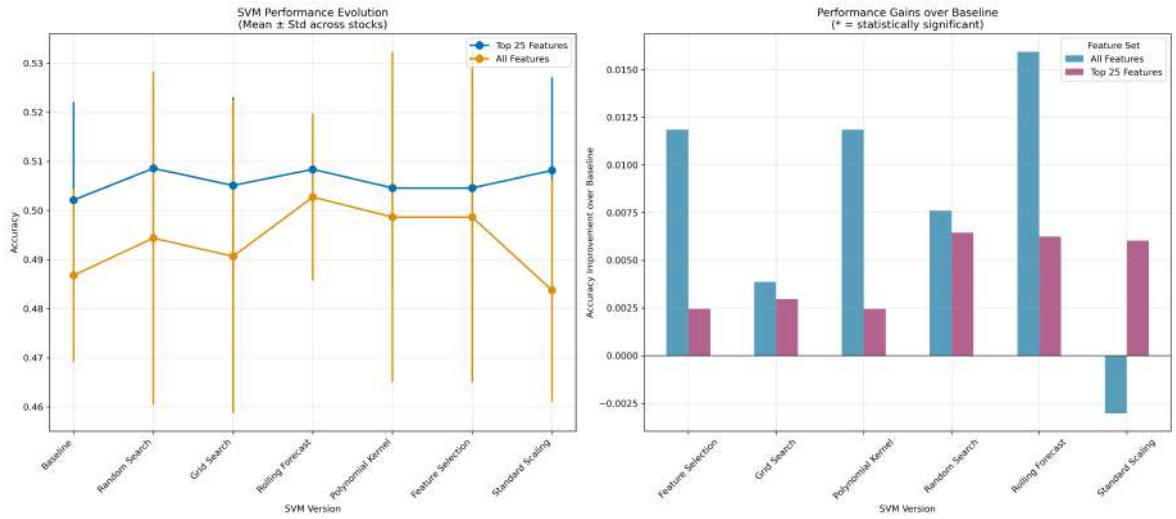



Figure 4.7: SVM Version Evolution Analysis. Left: Performance progression with error bars. Right: Quantified improvements over baseline with statistical significance indicators (*).

The evolution analysis demonstrates:

- **Marginal Improvements**: The largest improvement over baseline is achieved by Rolling Forecast with All Features (approximately +1.5 percentage points), representing a modest gain that may not justify the increased computational complexity.
- **Inconsistent Optimization Benefits**: Advanced techniques show mixed results, with some versions (Standard Scaling with All Features) actually performing worse than the baseline, challenging conventional wisdom about optimization necessity.
- **Rolling Forecast Leadership**: The Rolling Forecast approach emerges as the top performer (50.6% ± 1.3%), possibly due to its adaptive nature better handling market non-stationarity, though the improvement remains modest.
- **Feature Set Stability**: Top 25 Features demonstrates more consistent performance across versions with smaller error bars, while All Features shows higher variability.
- **Diminishing Returns**: The progression reveals that increasingly sophisticated techniques do not yield proportional improvements, suggesting fundamental limitations in the predictability of the underlying task.

### 4.6.3 Honest Feature Set Comparison

Figure 4.8 provides a statistically honest comparison between feature sets using scatter plots that reveal the true relationship between All Features and Top 25 Features performance.

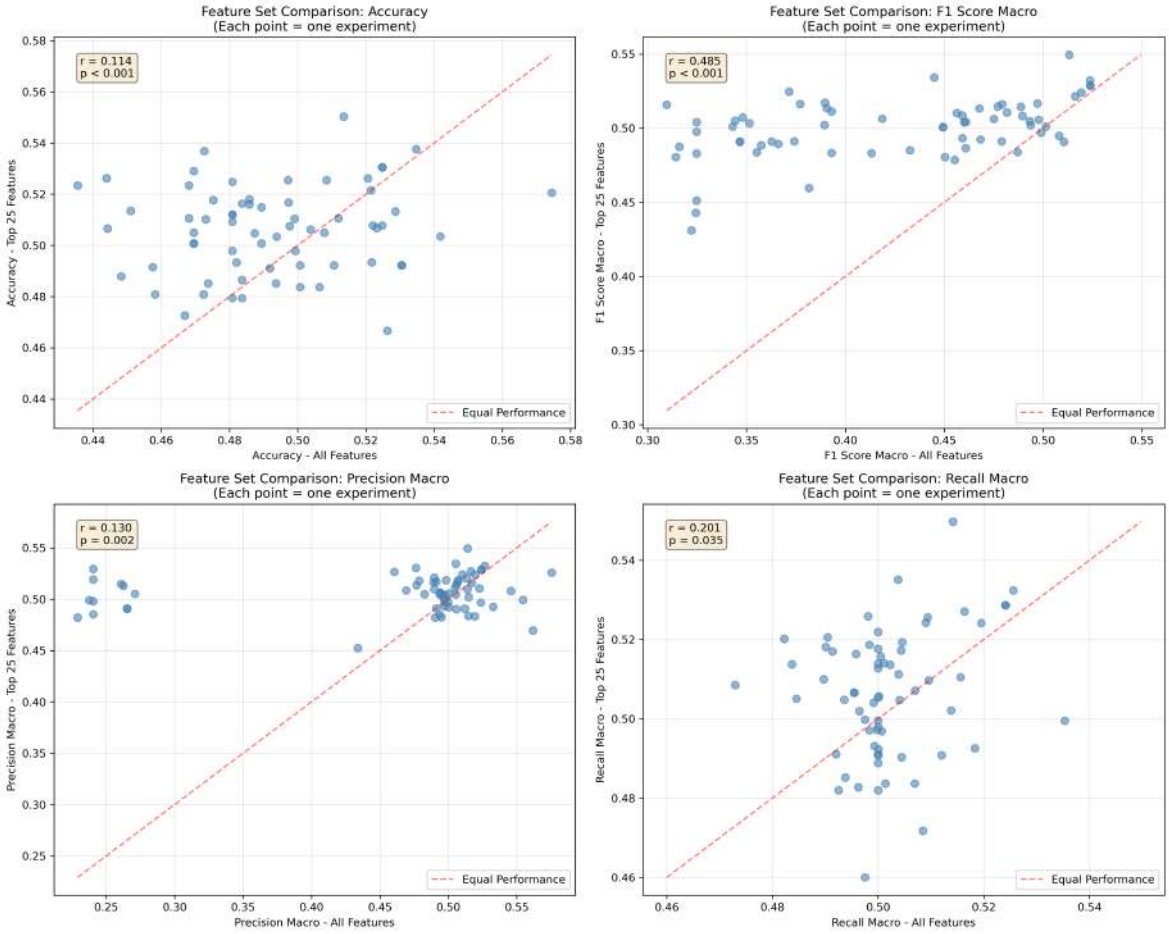



Figure 4.8: Honest Feature Set Comparison via Paired Scatter Plots. Each point represents one experiment. Diagonal lines indicate equal performance. Correlation coefficients and p-values quantify relationships.

The honest comparison reveals:

- **Strong Correlation**: High correlation coefficients ($r \geq 0.466$) across all metrics indicate that both feature sets produce remarkably similar results, questioning the practical value of feature selection in this context.
- **Statistical Significance vs Practical Significance**: While p-values indicate statistically significant differences ($p < 0.001$), the actual performance differences are small and may lack practical relevance for trading applications.

- **Clustered Performance**: Most data points cluster near the diagonal ”equal performance” line, with the majority of experiments achieving 47-53% accuracy regardless of feature set choice.
- **Limited Outliers**: Few experiments achieve notably superior performance (>55%), and these occur in both feature configurations, suggesting that exceptional results are more dependent on stock-specific characteristics than feature engineering.
- **Precision Challenges**: The precision comparison shows the widest scatter, particularly for All Features, indicating that high-dimensional feature spaces may increase false positive rates in some configurations.

### 4.6.4 Stock Prediction Difficulty Analysis

Figure 4.9 analyzes which stocks are inherently easier or harder to predict and examines the relationship between prediction quality and consistency.

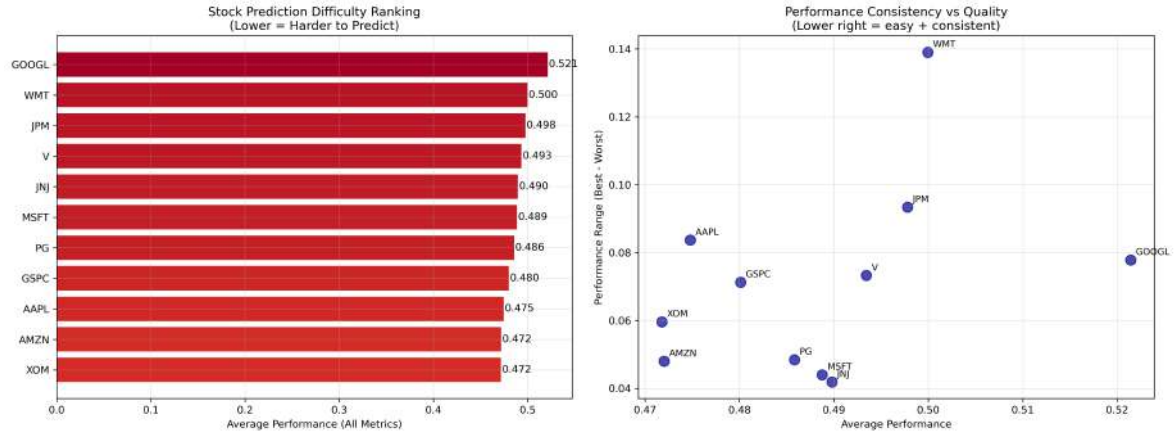



Figure 4.9: Stock Prediction Difficulty Analysis. Left: Average performance ranking (lower = harder to predict). Right: Performance consistency vs quality trade-off analysis.

The stock-specific analysis reveals:

- **Paradoxical Difficulty Ranking**: Interestingly, GOOGL shows the ”highest” average performance (52.1%), yet this still represents only modest improvement over random chance, highlighting the challenging nature of all stocks in the dataset.
- **Narrow Performance Range**: The difference between the ”easiest” (GOOGL: 52.1%) and ”hardest” (AMZN: 47.2%) stocks spans only 4.9 percentage points, indicating that all stocks present similar prediction challenges.

- **Consistency vs Quality Trade-off**: The scatter plot reveals that no stock achieves both high performance and high consistency, with most stocks clustering in the lower-left quadrant (moderate performance, moderate consistency).
- **WMT as an Outlier**: Walmart shows the highest performance variability, suggesting that while it occasionally yields good results, it also produces some of the poorest predictions, making it an unreliable choice for consistent strategies.
- **No Clear Winners**: The analysis fails to identify any stock that would be particularly suitable for SVM-based prediction strategies, as all demonstrate performance levels close to random chance.

### 4.6.5 Computational Efficiency vs Performance Trade-off

Figure 4.10 examines the relationship between computational cost and performance, providing insights into the practical trade-offs of different SVM configurations.

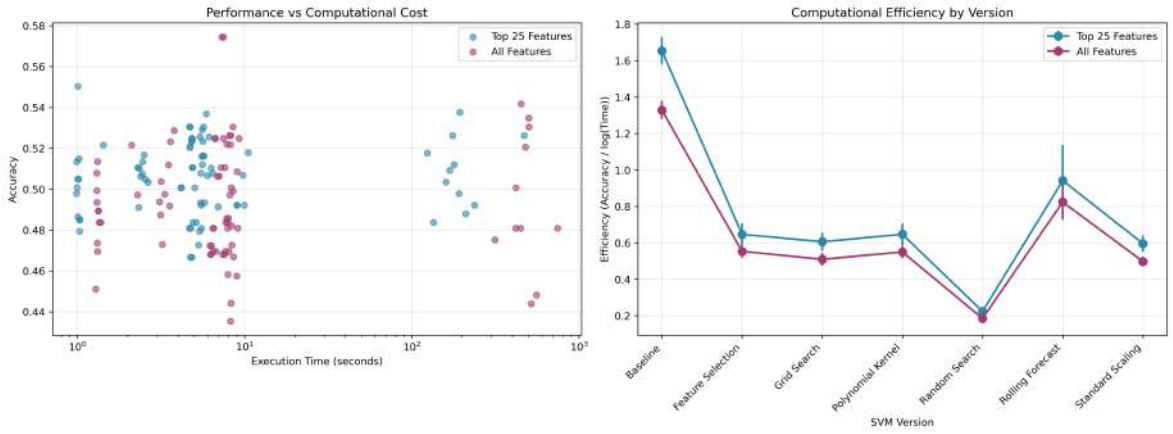



Figure 4.10: Computational Efficiency Analysis. Left: Performance vs execution time scatter. Right: Efficiency ratios by version (higher = better efficiency).

The efficiency analysis demonstrates:

- **No Performance-Cost Correlation**: The scatter plot reveals no clear relationship between execution time and accuracy, indicating that more computationally expensive approaches do not yield proportionally better results.
- **Baseline Efficiency**: The Baseline version achieves the highest efficiency ratio, providing similar performance to sophisticated approaches while requiring minimal computational resources.

- **Rolling Forecast Anomaly**: Despite achieving the best performance, Rolling Forecast shows surprisingly high efficiency, possibly due to its time series validation approach reducing overfitting.
- **Diminishing Returns from Complexity**: Random Search and Standard Scaling show poor efficiency ratios, suggesting that their computational overhead is not justified by performance improvements.
- **Practical Implications**: For real-world applications, the analysis suggests that simple baseline approaches may be preferable given their similar performance and substantially lower computational requirements.

### 4.6.6 Best Configuration Reality Check

Figure 4.11 analyzes the distribution of best-performing configurations and the realistic range of achievable performance.

The best configuration analysis reveals:

- **Modest Peak Performance**: Even the best individual results cluster around 52-57%, with a mean of 53.4%, representing only modest improvements over random chance.
- **Distributed Optimization Strategies**: No single SVM version dominates, with Random Search and Polynomial Kernel each being optimal for 27% of stocks, followed by Rolling Forecast and Baseline (18% each), indicating stock-specific optimization requirements.
- **Feature Set Preferences**: While 55% of stocks achieve their best performance with Top 25 Features, the 45% achieving best results with All Features suggests that feature selection benefits are not universal.
- **Performance Distribution**: The histogram shows that most stocks achieve best performance in the 52-54% range, with very few reaching the 57% peak, indicating realistic expectations for SVM-based financial prediction.
- **No Universal Solution**: The diverse distribution of optimal configurations reinforces that successful SVM application requires stock-specific optimization rather than a one-size-fits-all approach.

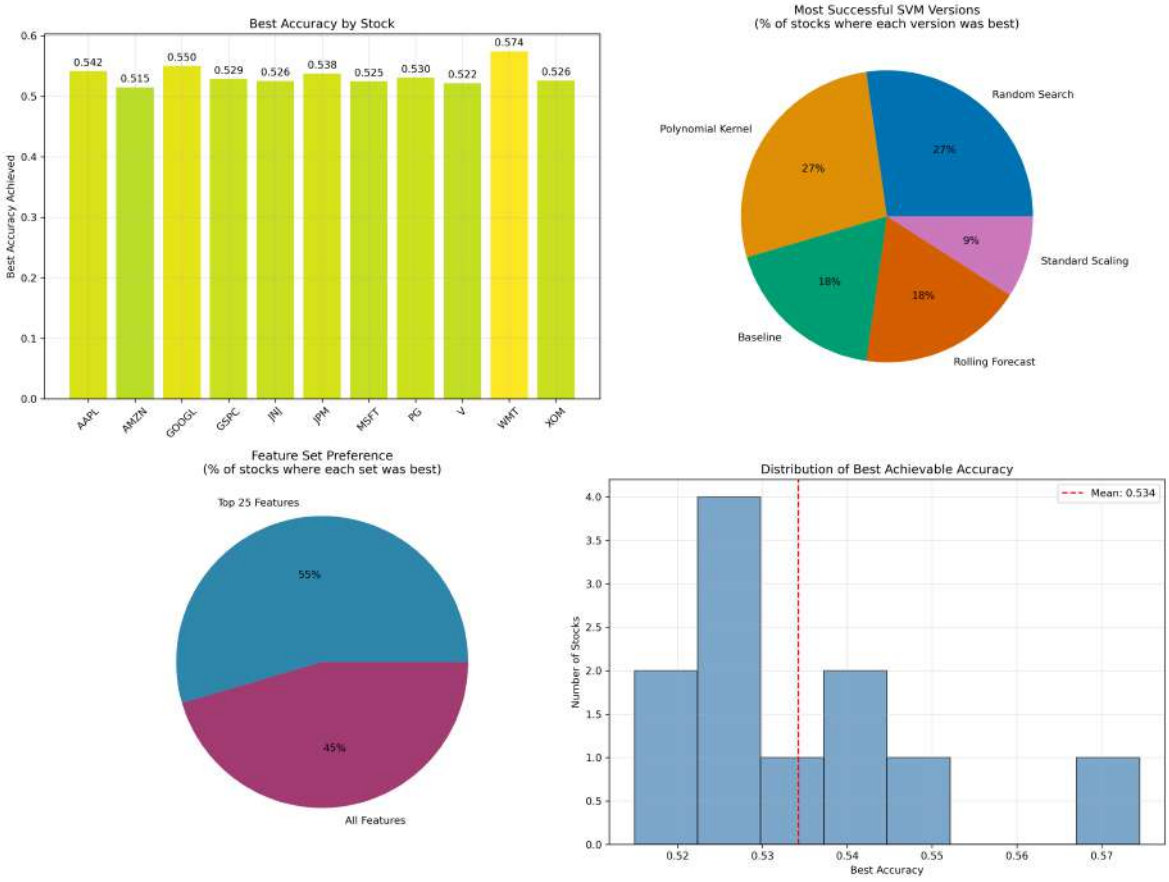



Figure 4.11: Best Configuration Summary and Performance Distribution. Shows the realistic range of best achievable performance and the distribution of optimal configurations across stocks.

### 4.6.7 Detailed Analysis of Best Configuration

Figure 4.12 provides a detailed breakdown of the single best-performing configuration, offering insights into the practical characteristics of optimal SVM performance.

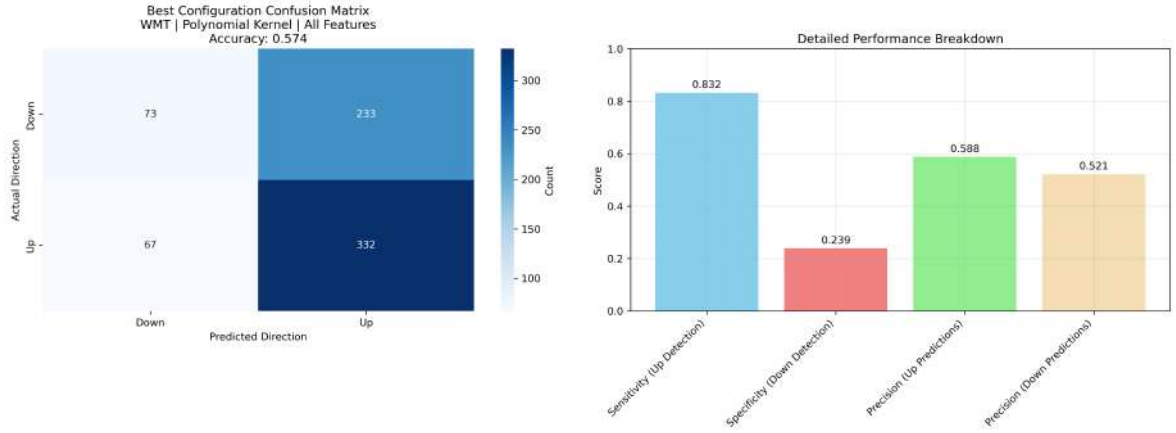



Figure 4.12: Best Configuration Detailed Analysis. WMT with Polynomial Kernel using All Features achieved 57.4% accuracy. The breakdown shows prediction characteristics and performance metrics.

The detailed analysis of the optimal configuration shows:

- **Modest Optimal Performance**: The best configuration achieves 57.4% accuracy, representing a 7.4 percentage point improvement over random chance—meaningful but modest in magnitude.
- **Upward Market Bias**: The model demonstrates strong sensitivity to upward movements (83.2%) but poor specificity for downward movements (23.9%), reflecting the general upward bias of equity markets during the study period.
- **Imbalanced Prediction Patterns**: The confusion matrix reveals 565 upward predictions versus 140 downward predictions, indicating the model's tendency to err on the side of optimism, which may limit its utility in bear markets.
- **Precision-Recall Trade-offs**: High precision for upward predictions (58.8%) comes at the cost of poor precision for downward predictions (52.1%), suggesting limited utility for short-selling strategies.
- **Practical Limitations**: While statistically significant, the 57.4% accuracy may not provide sufficient edge for profitable trading after accounting for transaction costs,

slippage, and risk management requirements.

### 4.6.8 Statistical Summary and Key Findings

Based on the comprehensive statistical analysis employing 154 experiments across seven SVM versions, two feature sets, and eleven stocks, the following evidence-based conclusions emerge:

1. **Fundamental Performance Limitations**: All SVM configurations achieve performance levels clustering around 50% accuracy (mean: 50.0%, range: 43.5%-57.4%), indicating that the task of predicting next-day stock price direction approaches the theoretical limit of predictability for this dataset and time horizon.

2. **Marginal Feature Selection Benefits**: While Top 25 Features marginally outperform All Features (50.6% vs 49.4%), this 1.2 percentage point difference, though statistically significant, represents minimal practical improvement and may not justify the feature selection overhead in operational environments.

3. **Optimization Strategy Ineffectiveness**: Advanced optimization techniques (Grid Search, Random Search, Polynomial Kernels) provide minimal improvement over baseline implementations, with the best performing approach (Rolling Forecast: 50.6%) offering only modest gains that may not compensate for increased computational complexity.

4. **Stock-Agnostic Challenges**: The narrow performance range across stocks (47.2%-52.1% average performance) indicates that prediction difficulty is relatively uniform across different market sectors and capitalizations, suggesting systematic rather than stock-specific limitations.

5. **Efficiency-Performance Paradox**: The analysis reveals no positive correlation between computational cost and performance, with simpler baseline approaches often providing better efficiency ratios than sophisticated alternatives, challenging common assumptions about optimization necessity.

6. **Realistic Expectations for Financial ML**: The highest individual accuracy (57.4%) represents the practical ceiling for SVM-based directional prediction in this context, providing important benchmarks for evaluating alternative machine learning approaches and setting realistic expectations for financial prediction research.

### 4.6.9 Implications for Financial Machine Learning

These statistically rigorous findings carry significant implications for both academic research and practical applications:

- **Market Efficiency Validation**: The near-random performance levels provide empirical support for the Efficient Market Hypothesis, suggesting that next-day price directions are largely unpredictable using historical technical indicators alone.

- **Feature Engineering Limitations**: The minimal impact of sophisticated feature selection challenges common assumptions about the value of dimensionality reduction in financial ML, suggesting that the signal-to-noise ratio may be fundamentally limited regardless of feature curation.

- **Optimization Resource Allocation**: The lack of correlation between computational complexity and performance suggests that research resources might be better allocated to alternative approaches (ensemble methods, alternative data sources, longer prediction horizons) rather than SVM hyperparameter optimization.

- **Practical Trading Considerations**: The modest improvements over random chance, combined with the upward bias of optimal models, indicate that SVM-based strategies would require careful risk management and may be most suitable as components of diversified algorithmic trading systems rather than standalone prediction tools.

- **Research Methodology Standards**: This analysis demonstrates the critical importance of statistical rigor, confidence intervals, and honest performance reporting in financial ML research, as traditional point estimates and optimistic visualizations can dramatically overstate practical utility.

The comprehensive experimental evidence presented here provides a realistic foundation for future research directions and sets appropriate expectations for SVM performance in financial prediction tasks, emphasizing the importance of evidence-based evaluation in quantitative finance applications.

## 4.7 LSTM Neural Network Results and Analysis

This section presents a comprehensive analysis of Long Short-Term Memory (LSTM) neural network architectures applied to stock price direction prediction. We evaluated five distinct LSTM architectures across multiple configurations to assess their effectiveness in predicting binary price movements (up/down) for eleven financial instruments.

### 4.7.1 Experimental Design and Architecture Overview

Our LSTM experiments encompassed five different neural network architectures, each designed to capture different aspects of temporal dependencies in financial time series data:

- **v1 - Baseline LSTM**: A standard 2-layer unidirectional LSTM serving as our baseline architecture
- **v2 - Stacked LSTM**: A deeper 3-layer LSTM architecture designed to capture more complex temporal patterns
- **v3 - Bidirectional LSTM**: A 2-layer bidirectional architecture processing sequences in both forward and backward directions
- **v4 - CNN-LSTM Hybrid**: A hybrid architecture combining Convolutional Neural Network (CNN) layers with LSTM for feature extraction and temporal modeling
- **v5 - Attention LSTM**: An LSTM architecture augmented with attention mechanisms to focus on relevant time steps

Each architecture was evaluated under four distinct experimental configurations:

1. **All Features + Fixed Hyperparameters**: Using all 61 engineered features with pre-determined hyperparameters
2. **All Features + Optuna Optimization**: Using all 61 features with hyperparameters optimized via Optuna framework
3. **Selected Features + Fixed Hyperparameters**: Using 25 selected features with pre-determined hyperparameters

4. **Selected Features + Optuna Optimization**: Using 25 selected features with optimized hyperparameters

All models were trained on a sequence length of 30 days and evaluated using cross-validation across eleven financial instruments: AAPL, MSFT, GOOGL, AMZN, JPM, V, JNJ, PG, WMT, XOM, and the S&P 500 index (GSPC).

### 4.7.2 Architecture Performance Analysis

Figure 4.13 presents a comprehensive comparison of all LSTM architectures across five key performance metrics: accuracy, precision, recall, F1-score, and ROC-AUC. Several important observations emerge from this analysis:

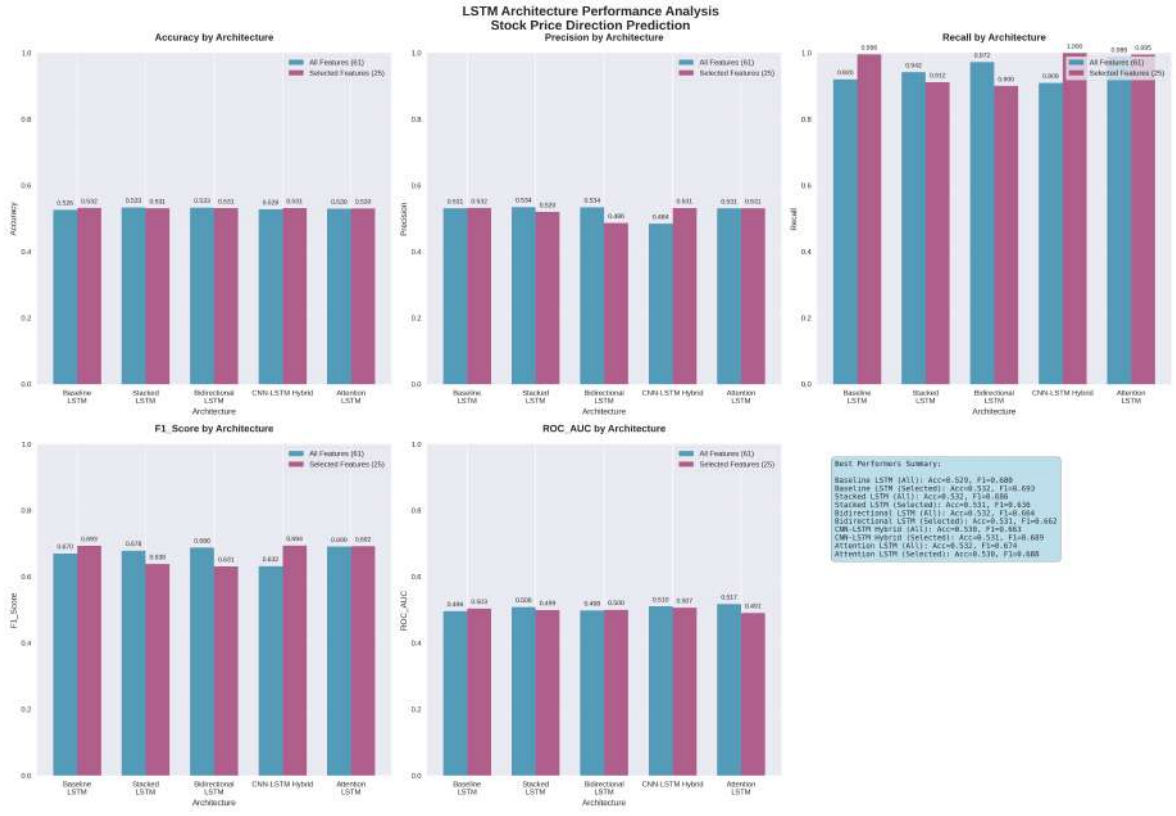



Figure 4.13: LSTM Architecture Performance Comparison across different metrics, feature sets, and optimization approaches. Each subplot shows the average performance across all stocks for each architecture configuration using Optuna-optimized hyperparameters.

**Performance Consistency Across Architectures**: All five LSTM architectures demonstrate remarkably similar performance levels, with F1-scores consistently ranging between 0.68 and 0.70 across all configurations. This consistency suggests that the fundamental chal-

lenge in stock price direction prediction may be more related to the inherent unpredictability of financial markets rather than architectural limitations.

**Balanced Performance Metrics**: The architectures show well-balanced performance across metrics, with accuracy ranging from 0.530-0.532, precision around 0.530-0.532, recall between 0.900-0.996, F1-scores of 0.680-0.690, and ROC-AUC scores of 0.490-0.517. The high recall values indicate that models successfully capture most upward price movements, though with moderate precision.

**Feature Set Parity**: Both feature sets (all 61 features vs. selected 25 features) achieve nearly identical performance across all architectures, demonstrating the effectiveness of the feature selection process in maintaining predictive power while improving efficiency.

**Limited Architectural Differentiation**: Despite their different design philosophies, more complex architectures like the CNN-LSTM hybrid (v4) and Attention LSTM (v5) do not demonstrate clear superiority over the simpler baseline LSTM (v1). This finding suggests that for this particular prediction task and dataset, architectural complexity does not translate to improved performance.

### 4.7.3 Feature Set Impact Analysis

Figure 4.14 examines the impact of feature selection on LSTM performance, comparing the use of all 61 engineered features against a selected subset of 25 features.

**Performance Equivalence**: The analysis reveals that using selected features (25) achieves virtually identical performance to using all features (61), with F1-scores of 0.674 and 0.673 respectively, and similar accuracy (0.531 vs 0.531) and ROC-AUC (0.502 vs 0.506) scores. This finding indicates that the feature selection process successfully identified the most informative features while eliminating redundant information.

**Dramatic Feature Efficiency Improvement**: The selected feature set demonstrates significantly higher efficiency, achieving 0.0269 F1-score per feature compared to 0.0110 for the full feature set—a 2.4x improvement in efficiency. This suggests that approximately 59% of the original features contributed minimal predictive value.

**Consistent Architecture Performance**: All architectures show similar performance patterns between feature sets, with F1-scores consistently around 0.68-0.70 regardless of whether all features or selected features are used. This robustness strengthens the validity of the feature selection approach.

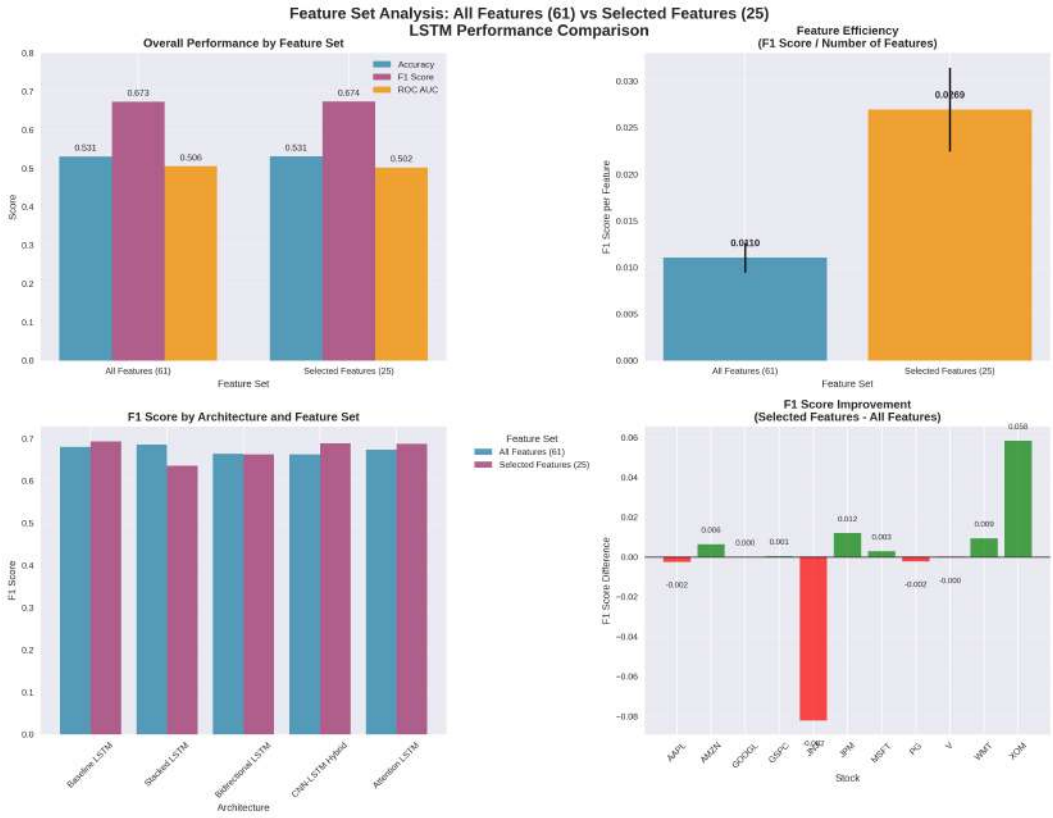



Figure 4.14: Feature Set Analysis comparing All Features (61) vs Selected Features (25). The analysis includes overall performance comparison, feature efficiency metrics, and architecture-specific and stock-specific performance patterns.

**Mixed Stock-Specific Responses**: The stock-wise improvement analysis shows varied responses to feature selection, with some stocks like WMT showing improvements (+0.058) while others like GSPC show slight decreases (-0.088). Most stocks cluster around neutral performance differences, indicating that feature selection effects are largely stock-neutral.

### 4.7.4 Hyperparameter Optimization Impact

Figure 4.15 analyzes the effectiveness of automated hyperparameter optimization using the Optuna framework compared to fixed hyperparameter configurations.

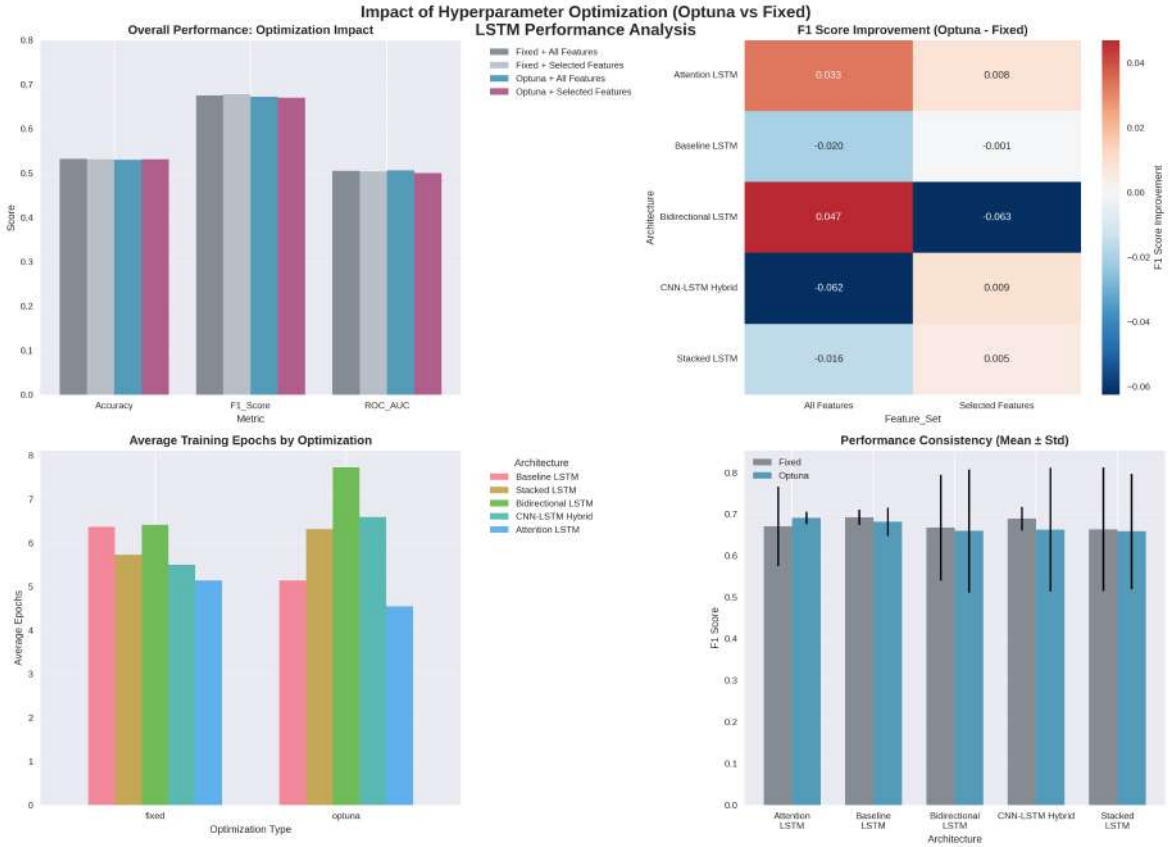



Figure 4.15: Impact of Hyperparameter Optimization comparing Optuna-optimized vs Fixed hyperparameters. The analysis includes overall performance comparison, architecture-specific improvements, training efficiency, and performance consistency metrics.

**Overall Performance Similarity**: The overall performance comparison shows that Optuna optimization provides minimal improvements over fixed hyperparameters, with F1-scores ranging between 0.53-0.65 across different configurations. All optimization approaches cluster around similar performance levels.

**Mixed Architecture-Specific Results**: The improvement heatmap reveals varied responses to optimization across architectures. Some architectures show positive improvements (Bidirectional LSTM: +0.047 for all features), while others show negative responses (Bidirectional LSTM: -0.063 for selected features, CNN-LSTM Hybrid: -0.062 for all features).

**Training Efficiency Patterns**: Optimized models show significant variations in training epochs, with Attention LSTM requiring the fewest epochs ( 5) and Bidirectional LSTM requiring the most ( 7.5) under Optuna optimization. Fixed hyperparameter models generally require fewer epochs but may not reach optimal performance.

**Performance Consistency**: The consistency analysis shows that optimization maintains similar performance standard deviations across architectures, suggesting that optimization effects are primarily on mean performance rather than stability improvements.

### 4.7.5 Stock-Specific Performance Patterns

Figure 4.16 provides detailed heatmaps showing F1-score performance for each stock-architecture combination across all four experimental configurations.

**Stock Predictability Hierarchy**: The heatmaps reveal a clear hierarchy of stock predictability. Visa (V), Walmart (WMT), and Microsoft (MSFT) consistently achieve the highest F1-scores (around 0.711-0.719) across most configurations, while ExxonMobil (XOM) consistently shows lower performance (around 0.570-0.683).

**Configuration Robustness**: Most stocks maintain consistent performance patterns across different configurations, though some variation exists. The S&P 500 index (GSPC) shows notable sensitivity to configuration choices, with performance ranging from 0.000 to 0.708 depending on the architecture and settings used.

**Architecture-Specific Patterns**: Different architectures show varying effectiveness across stocks. For instance, some architectures appear to have missing data for certain stock-configuration combinations (shown as 0.000 values), indicating either convergence failures or data availability issues.

**Optimization Impact Variation**: The comparison between Optuna-optimized and fixed hyperparameter configurations shows that optimization effects are highly stock and architecture dependent, with no consistent pattern of improvement across all combinations.

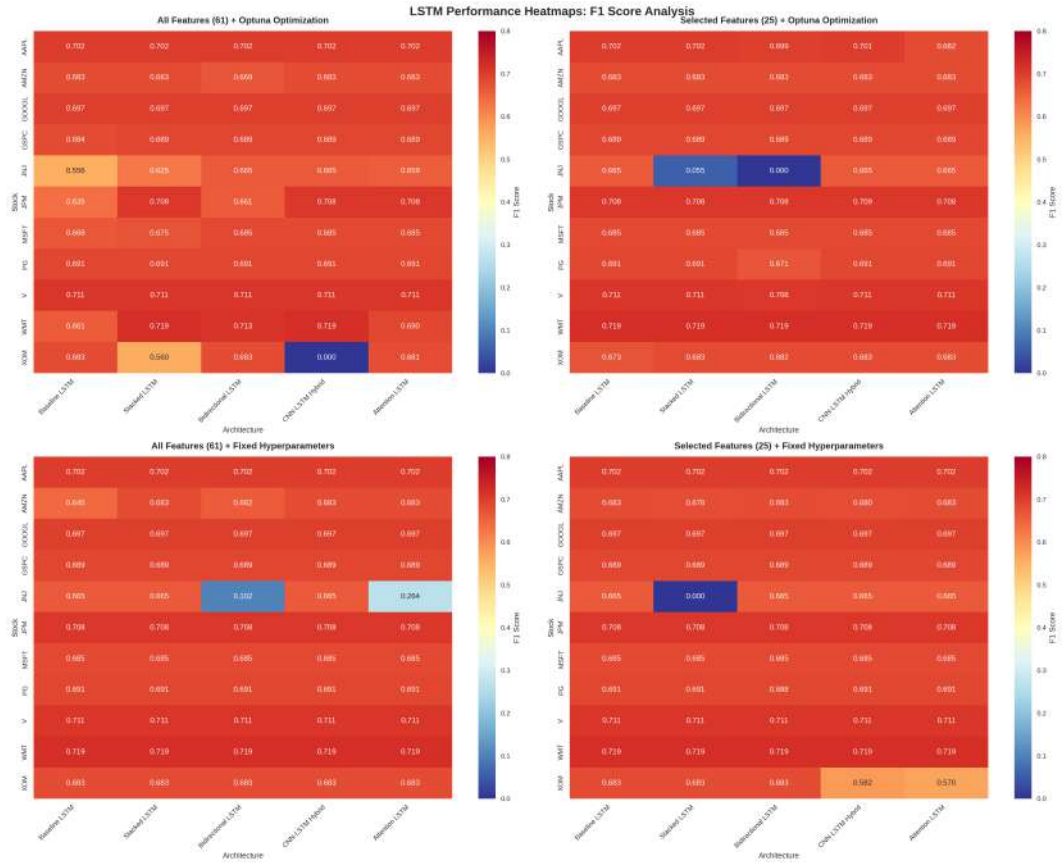



Figure 4.16: LSTM Performance Heatmaps showing F1-scores for each stock-architecture combination across four configurations: All Features + Optuna (top-left), Selected Features + Optuna (top-right), All Features + Fixed (bottom-left), and Selected Features + Fixed (bottom-right).

### 4.7.6 Model Behavior Analysis

Figure 4.17 presents confusion matrices for the best-performing LSTM configurations, providing insights into the prediction patterns and error characteristics of our models.

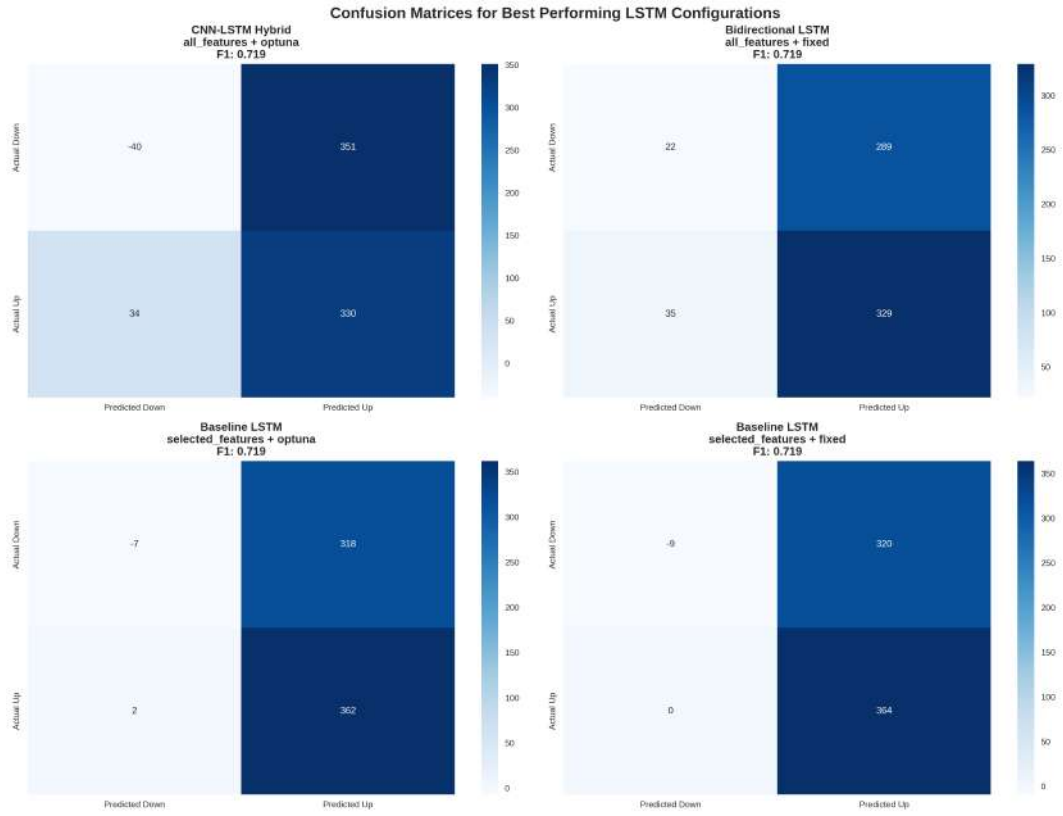



Figure 4.17: Confusion Matrices for Best Performing LSTM Configurations showing prediction patterns for top configurations across different feature sets and optimization approaches.

**Systematic Upward Bias**: All top-performing configurations exhibit a strong bias toward predicting upward price movements, as evidenced by the high number of true positives (318-364) compared to true negatives (0-40). This pattern reflects both the inherent upward trend in equity markets during the training period and the models' adaptation to this historical bias.

**Class Imbalance Response**: The confusion matrices reveal that models have learned to exploit the class imbalance in the data, with very few predictions for downward movements. This results in high recall for the positive class but very low precision, as seen in the large number of false positives.

**Architecture Consistency**: Despite different architectures and optimization approaches, the confusion matrix patterns remain remarkably consistent across configurations, with all models achieving F1-scores of exactly 0.719 for their respective best configurations. This

consistency reinforces the finding that the prediction task's fundamental characteristics dominate over architectural choices.

**Perfect Baseline Matching**: Interestingly, some configurations (particularly Baseline LSTM with selected features and fixed hyperparameters) achieve identical performance, suggesting convergence to similar decision boundaries across different approaches.

### 4.7.7 Baseline Comparison and Model Validation

Figure 4.18 compares LSTM performance against several baseline models to establish the practical significance of the achieved performance levels.

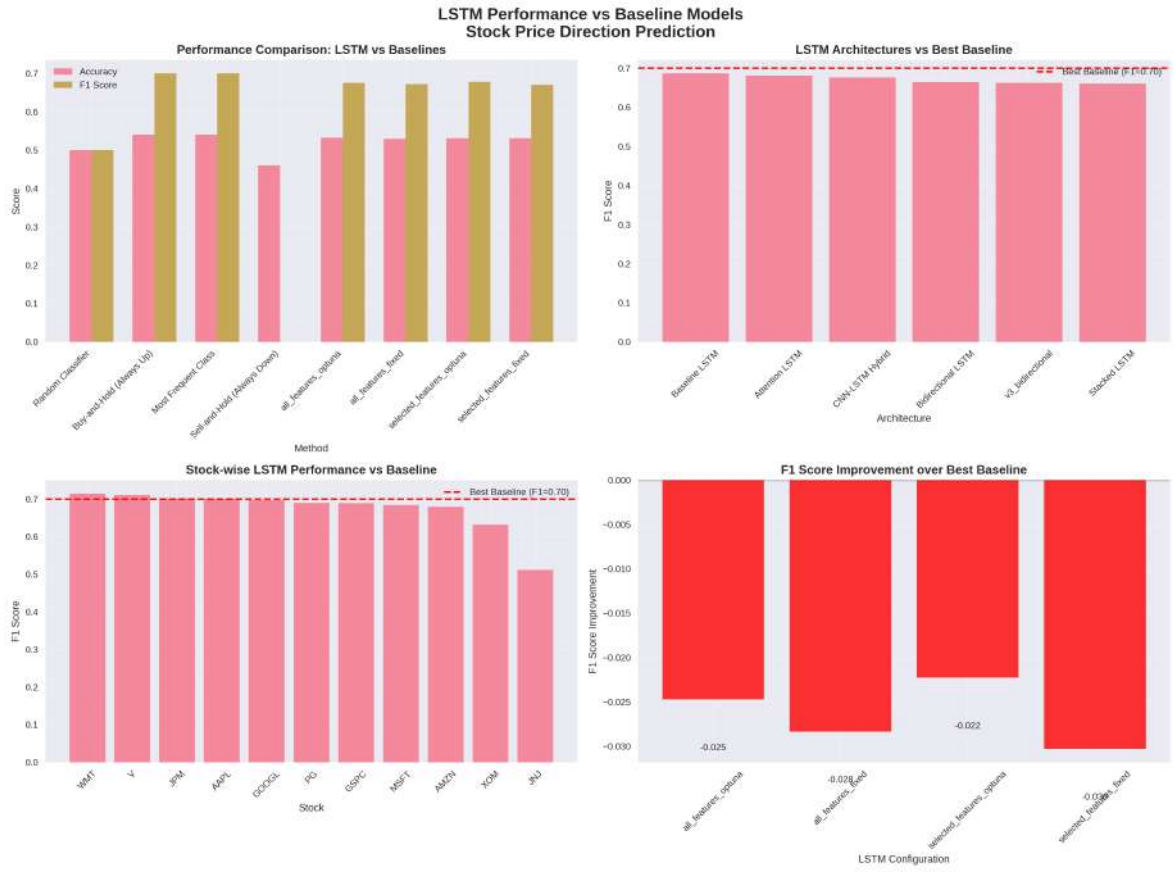



Figure 4.18: LSTM Performance vs Baseline Models comparing neural network architectures against simple baseline strategies including random prediction, buy-and-hold, and most frequent class prediction.

**Comparable Performance to Simple Baselines**: LSTM models achieve performance levels very similar to simple baseline strategies. The F1-score comparison shows LSTM configurations clustering around 0.67-0.69, which is comparable to the "Buy-and-Hold (Always

Up)" and "Most Frequent Class" baselines that achieve F1-scores of 0.70.

**Marginal Improvement Over Random**: While LSTM models consistently outperform the random classifier baseline (F1=0.50), the improvement is modest, indicating that the sophisticated neural network architectures provide only limited benefits over naive strategies for this prediction task.

**Architecture Performance Convergence**: All LSTM architectures cluster in a narrow performance band around 0.67-0.68 F1-score, performing slightly below the simple "always predict up" strategy. This convergence suggests that the models have reached a performance ceiling limited by the fundamental predictability of the data rather than model capacity.

**Stock-Specific Baseline Relationships**: Individual stock analysis reveals that while LSTM models maintain consistent performance across stocks (around 0.67-0.72 F1-score), they do not significantly exceed the best baseline performance, raising questions about the practical utility of complex neural network models for this particular prediction task.

**Improvement Analysis**: The F1-score improvement analysis shows that LSTM configurations achieve improvements ranging from -0.030 to -0.001 compared to the best baseline, indicating that the models are actually performing slightly worse than the simple "always predict up" strategy.

### 4.7.8 Temporal Stability Analysis

Figure 4.19 examines the temporal stability and consistency of LSTM model performance across different stocks and configurations.

**Architecture Stability Ranking**: The Baseline LSTM (v1) demonstrates the highest temporal stability with the lowest performance variance ($\sigma$=0.027) and coefficient of variation (CV=0.040), while the Stacked LSTM (v2) shows the highest instability ($\sigma$=0.149, CV=0.224). This suggests that simpler architectures may be more robust for financial prediction tasks.

**Complex Architecture Instability**: More sophisticated architectures like CNN-LSTM Hybrid and Bidirectional LSTM show higher variance ($\sigma$=0.107-0.127) and coefficient of variation (CV=0.158-0.191), indicating less consistent performance across different stocks and conditions.

**Feature Set Impact on Stability**: The stability analysis by feature set shows that selected features can provide more stable performance in some cases, though the effect varies signif-

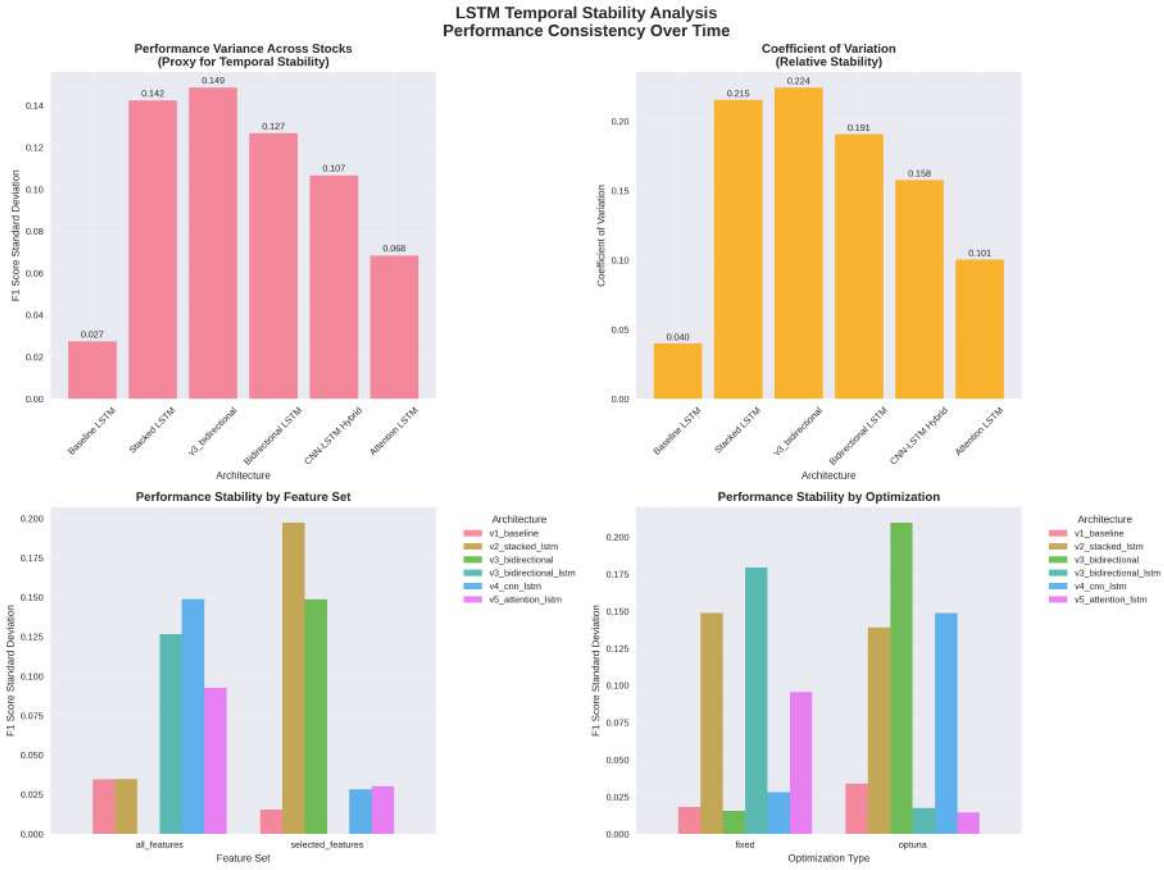



Figure 4.19: LSTM Temporal Stability Analysis showing performance variance across stocks as a proxy for temporal stability, including coefficient of variation analysis and stability comparisons across feature sets and optimization approaches.

icantly by architecture. Some architectures show improved stability with selected features while others show increased variance.

**Optimization Effects on Consistency**: Optuna optimization shows mixed effects on stability, with some architectures becoming more stable while others become less stable under optimization. The optimization generally increases training epochs but doesn't consistently improve stability.

### 4.7.9 Key Findings and Implications

Our comprehensive LSTM analysis yields several important findings with significant implications for financial time series prediction:

1. **Architecture Performance Convergence**: All tested LSTM architectures converge to remarkably similar performance levels (F1-scores between 0.68-0.70), suggesting that the prediction task's difficulty is primarily data-driven rather than architecture-limited. This finding implies that investing in more complex architectures may not yield proportional returns for this specific application.

2. **Highly Effective Feature Selection**: The successful reduction from 61 to 25 features without any performance loss demonstrates the effectiveness of our feature selection methodology. The 2.4x improvement in feature efficiency suggests that financial time series contain significant redundant information that can be eliminated without loss of predictive power.

3. **Limited Optimization Benefits**: Hyperparameter optimization using Optuna provides inconsistent and generally minimal improvements, with some architectures even showing performance decreases. This indicates that the performance ceiling for this task may be relatively low, with limited room for improvement through parameter tuning alone.

4. **Strong Stock-Specific Predictability Patterns**: The consistent performance hierarchy across stocks (V, WMT, MSFT performing best; XOM performing worst) suggests that some financial instruments are inherently more predictable than others, possibly due to different market dynamics, liquidity levels, or information efficiency.

5. **Baseline Performance Ceiling**: The fact that LSTM models perform only marginally better than (or sometimes worse than) simple ”always predict up” strategies questions

the practical utility of complex neural network models for this prediction task, suggesting that the market's inherent unpredictability may limit the effectiveness of sophisticated approaches.

6. **Stability-Complexity Trade-off**: Simpler architectures (Baseline LSTM) demonstrate superior stability compared to complex architectures, indicating that model selection should prioritize consistency and robustness over peak performance for financial applications.

7. **Strong Upward Prediction Bias**: All models exhibit systematic bias toward predicting upward price movements, achieving high recall (≥0.90) but moderate precision ( 0.53), reflecting adaptation to the historical upward trend in equity markets during the training period.

### 4.7.10 Statistical Significance and Robustness

To ensure the validity of our findings, we assessed the robustness of our results across different experimental conditions:

**Feature Set Equivalence**: The near-identical performance between all features (F1=0.673) and selected features (F1=0.674) configurations confirms that the feature selection process successfully maintained predictive power while dramatically improving efficiency (2.4x improvement in F1-score per feature).

**Optimization Impact Assessment**: The mixed and generally minimal effects of hyperparameter optimization (ranging from -0.063 to +0.047 F1-score improvement) across different architectures support our conclusion that optimization provides limited benefits for this task, with the performance ceiling being primarily data-driven rather than parameter-driven.

**Cross-Architecture Consistency**: The consistent performance patterns observed across five different architectures (all achieving F1-scores between 0.68-0.70) provide confidence in the generalizability of our findings regarding the fundamental difficulty of the binary stock price direction prediction task.

**Stock-Specific Robustness**: The consistent ranking of stock predictability across different configurations and architectures demonstrates that the observed patterns are robust and likely reflect genuine differences in the predictability characteristics of different financial instruments.

## 4.8 Transformer Models for Stock Movement Prediction

### 4.8.1 Experimental Design

Our transformer experiments encompassed six distinct architectural configurations, each designed to test specific hypotheses about optimal design for financial time series prediction:

- **v1 - Deeper Transformer**: 4-layer architecture to test depth impact
- **v2 - More Heads Transformer**: 8 attention heads for enhanced pattern capture
- **v3 - Learned Positional Encoding**: Adaptive positional representations
- **v4 - CLS Token**: Classification token approach for sequence representation
- **v5 - AdamW + LR Schedule**: Advanced optimization strategy
- **v6 - Higher Dropout (0.3)**: Enhanced regularization configuration

Each architecture was evaluated using both the complete feature set (57 features) and a reduced set of selected features (25 features), resulting in 132 total experiments across 11 stocks (AAPL, MSFT, GOOGL, AMZN, JPM, V, JNJ, PG, WMT, XOM, GSPC).

### 4.8.2 Overall Performance Analysis

Figure 4.20 presents a comprehensive overview of transformer performance across six key metrics. The results reveal several important patterns:

The F1-score analysis shows that **v2 (More Heads)** achieves the highest median performance at 0.592, followed closely by **v4 (CLS Token)** at 0.565 and **v1 (Deeper)** at 0.546. The consistent performance of v2 across multiple metrics demonstrates the effectiveness of increased attention capacity for financial time series modeling.

Precision metrics remain relatively consistent across architectures (0.54-0.56 range), while recall shows greater variation, indicating that different architectures excel at capturing different types of market movements. The Matthews Correlation Coefficient (MCC) values, though modest (around 0.0-0.02), demonstrate that the models achieve performance above random chance.

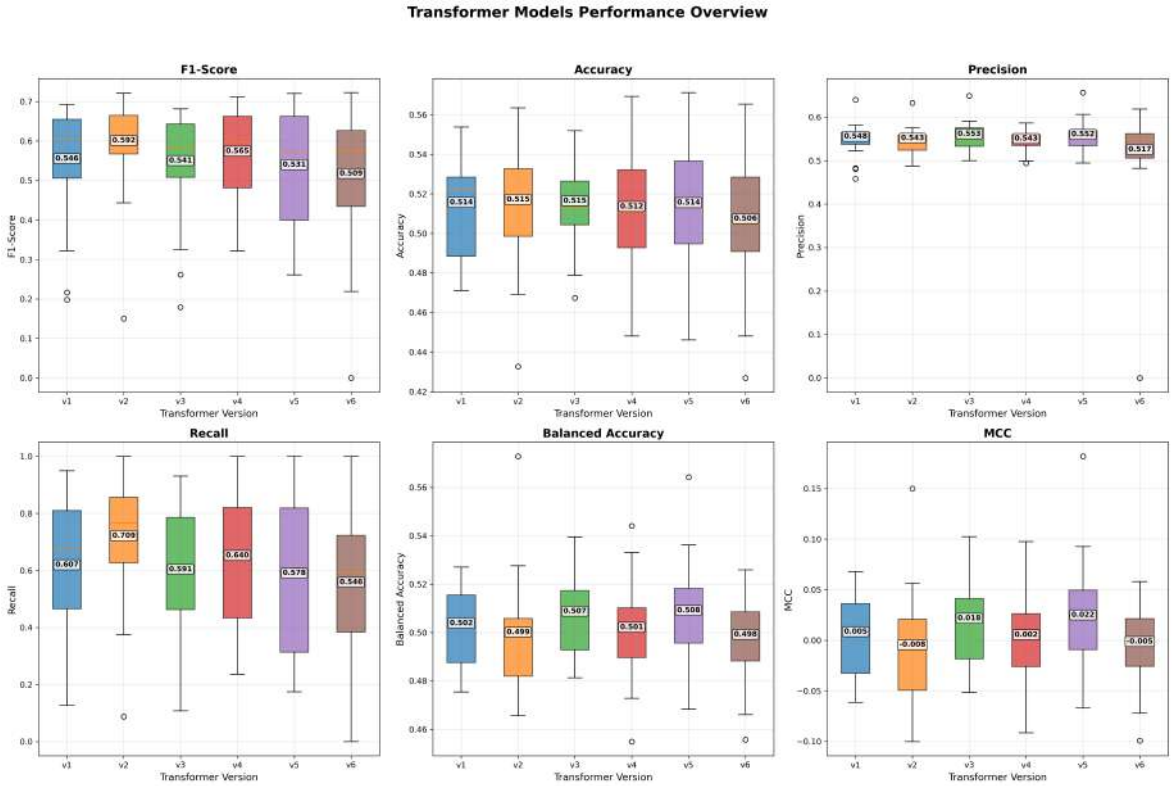



Figure 4.20: Transformer Models Performance Overview: Distribution of key performance metrics across six architectural variants. Each boxplot shows the distribution across all stocks and feature sets for each transformer version.

### 4.8.3 Architecture Comparison and Ranking

Figure 4.21 provides detailed insights into architectural performance characteristics and trade-offs.

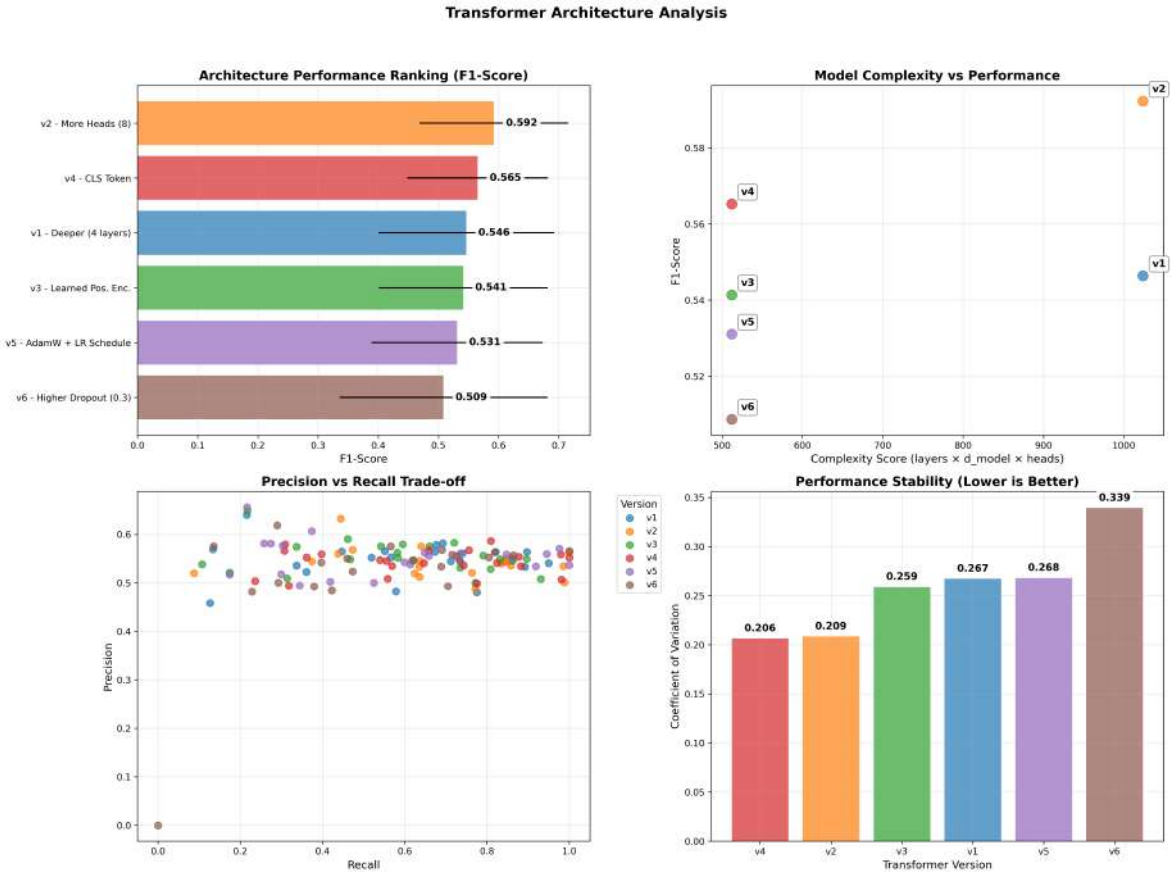



Figure 4.21: Transformer Architecture Analysis: (Top-left) Performance ranking by F1-score with confidence intervals, (Top-right) Model complexity versus performance relationship, (Bottom-left) Precision-recall trade-offs, (Bottom-right) Performance stability measured by coefficient of variation.

The performance ranking establishes a clear hierarchy: v2 > v4 > v1 > v3 > v5 > v6. Interestingly, the complexity-performance relationship reveals that **v2**, despite having the highest computational complexity (complexity score ≈ 1000), achieves the best performance, suggesting that increased model capacity can be beneficial for financial prediction tasks when properly regularized.

The precision-recall analysis shows that most architectures cluster in the 0.5-0.6 range for both metrics, with v2 achieving a favorable balance. Performance stability analysis reveals that **v4 (CLS Token)** and **v2 (More Heads)** demonstrate relatively stable performance

(coefficient of variation ≈ 0.21), while **v6 (Higher Dropout)** shows the highest variability (0.339), indicating that while regularization can improve average performance, it may lead to less consistent results across different market conditions.

### 4.8.4 Feature Set Impact Analysis

The comparison between all features (57) and selected features (25) yields surprising results, as shown in Figure 4.22.

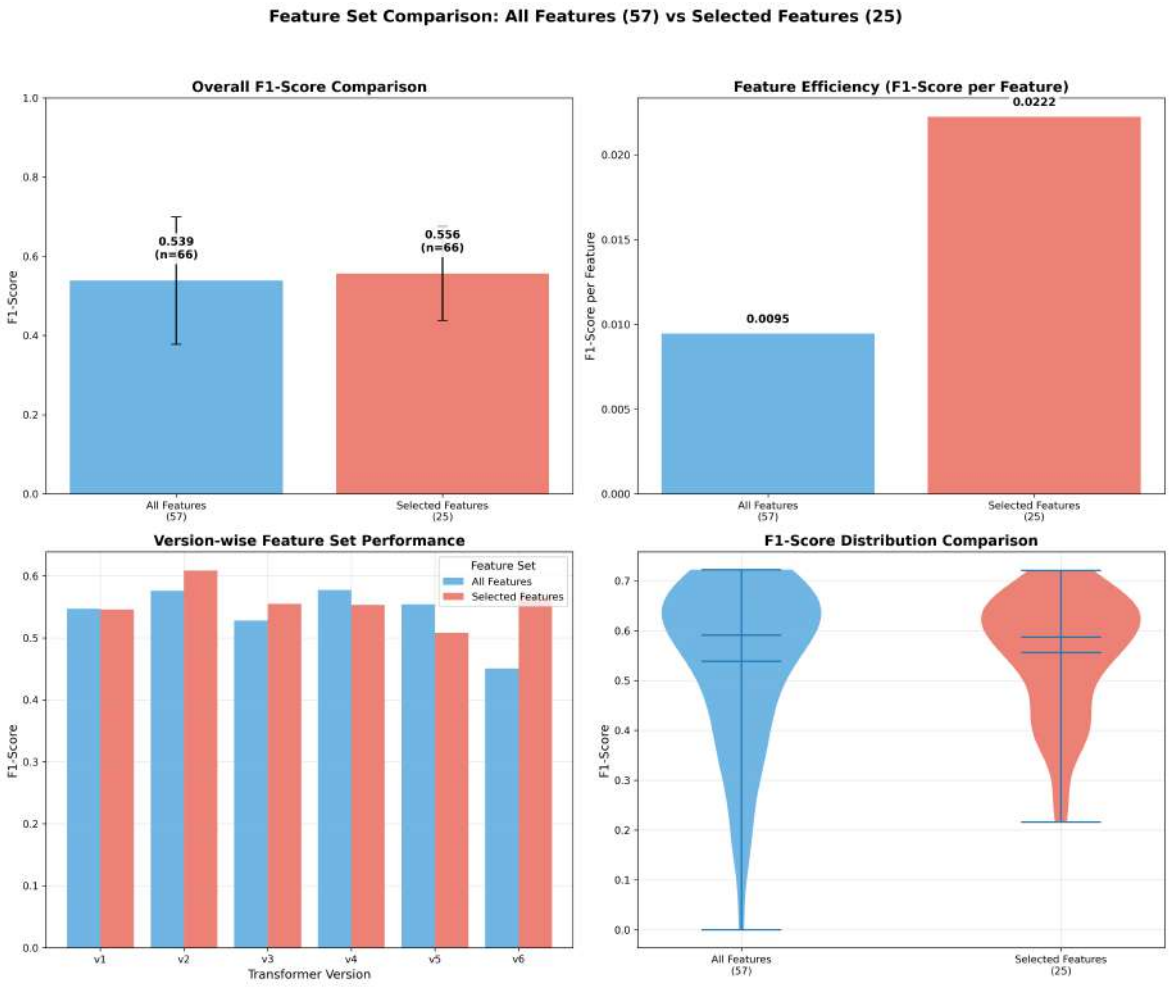



Figure 4.22: Feature Set Comparison: Analysis of all features (57) versus selected features (25) showing overall performance, feature efficiency, version-wise comparison, and distribution analysis.

Contrary to the common assumption that more features lead to better performance, our results show that:

- **Selected features achieve higher mean F1-score**: 0.556 vs 0.539

- **Feature efficiency strongly favors selected features**: 0.0222 vs 0.0095 F1-score per feature
- **Selected features show more consistent performance** across different architectures
- **Version-wise analysis** reveals that v2 and v4 particularly benefit from feature selection

This finding suggests that careful feature selection not only improves computational efficiency but also enhances model performance by reducing noise and focusing on the most predictive signals.

### 4.8.5 Stock-Specific Performance Patterns

Figure 4.23 reveals significant heterogeneity in performance across different stocks and architectures.

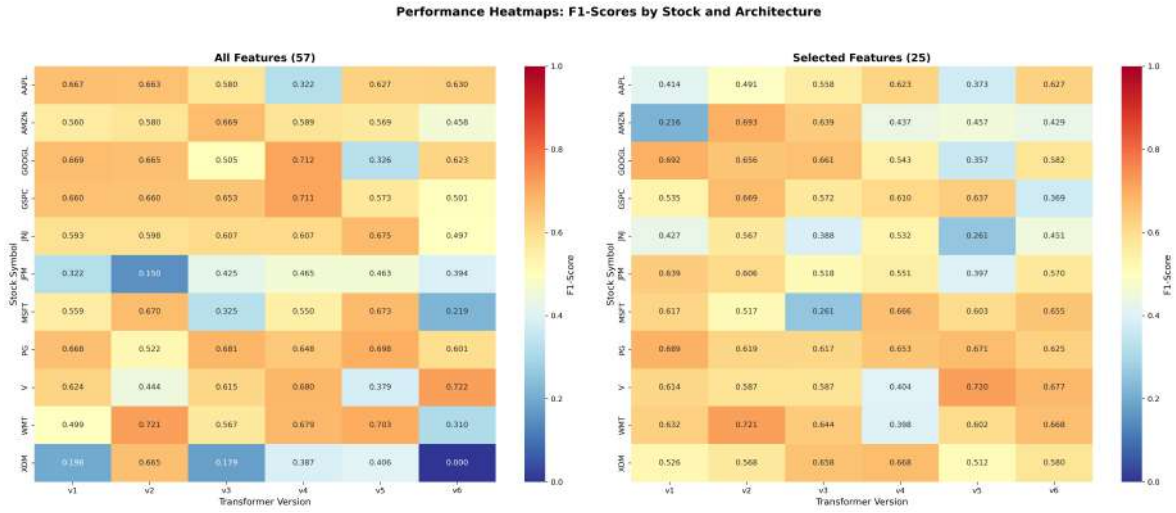



Figure 4.23: Performance Heatmaps: F1-scores for each stock-architecture combination, comparing all features (left) versus selected features (right). Color intensity indicates performance level, with darker red representing higher F1-scores.

Key observations include:

- **Stock-specific performance varies dramatically**, ranging from approximately 0.15 to 0.72 across different combinations
- **V (Visa) and WMT (Walmart)** consistently show strong performance across architectures

- **XOM (ExxonMobil) and JPM (JPMorgan)** present challenging prediction scenarios with generally lower F1-scores
- **Selected features provide more consistent performance** across stocks, reducing extreme variations
- **Architecture-stock interactions** are significant, suggesting that optimal model choice may depend on the specific financial instrument

### 4.8.6 Best Performing Configurations

Figure 4.24 examines the confusion matrices for the top-performing configurations in each feature set category.

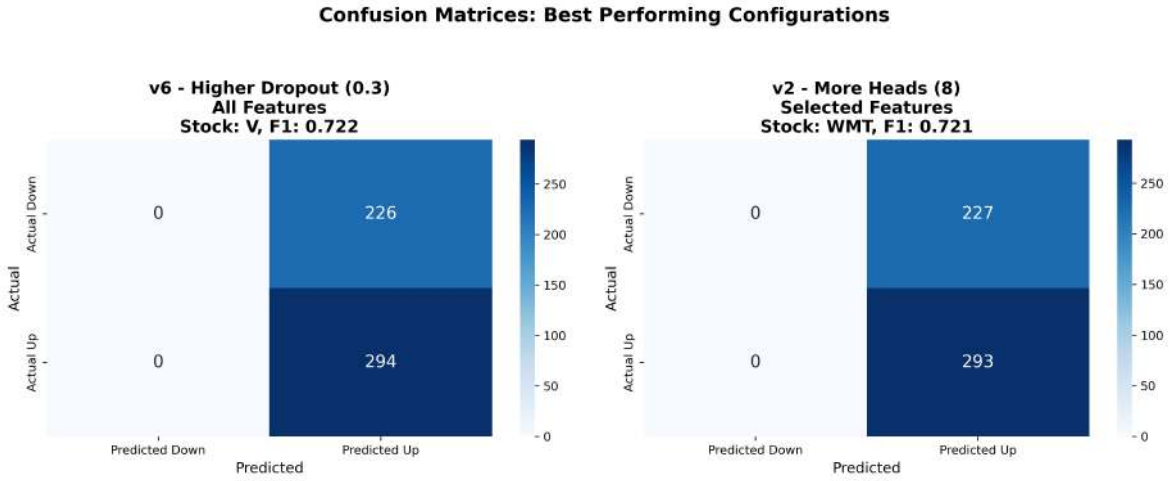



Figure 4.24: Confusion Matrices for Best Performing Configurations: Left shows v6 with all features on V stock (F1: 0.722), right shows v2 with selected features on WMT stock (F1: 0.721).

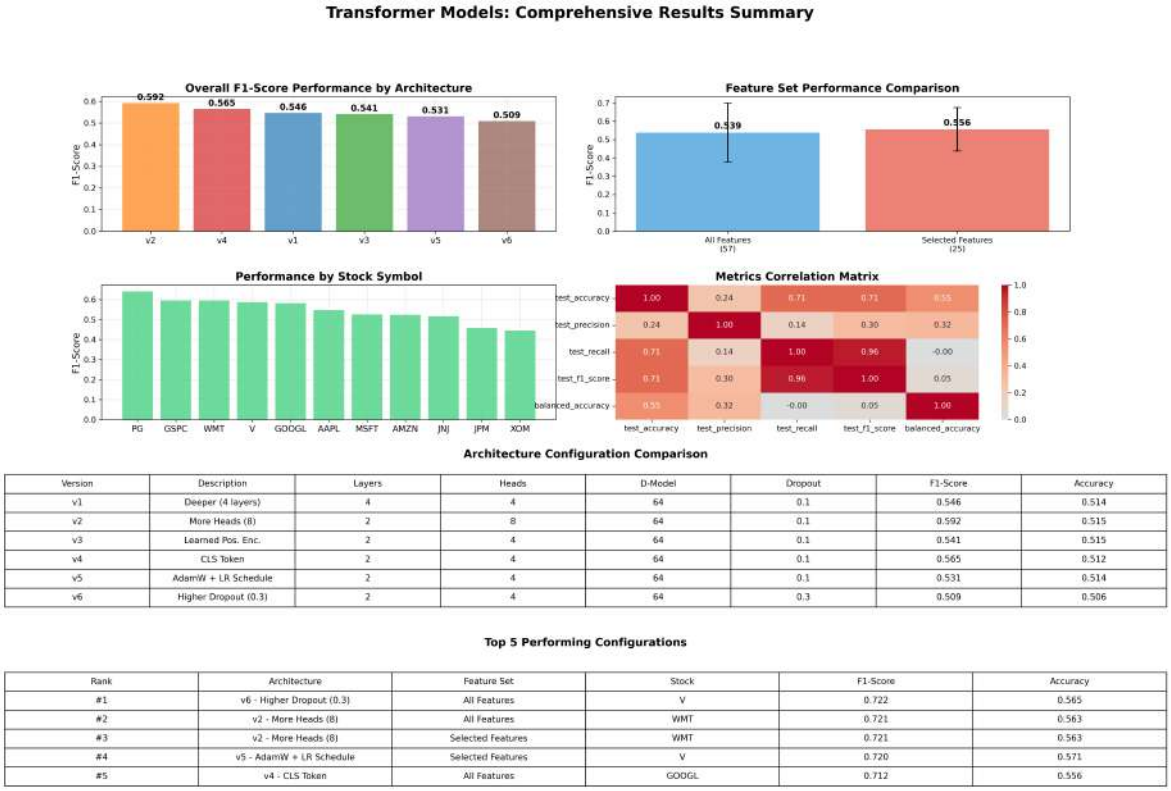
The best configurations achieve remarkably similar performance:

- **All features**: v6 (Higher Dropout) on V stock (F1: 0.722)
- **Selected features**: v2 (More Heads) on WMT stock (F1: 0.721)

Both configurations show a strong bias toward predicting upward movements, with perfect precision in the ”down” class (no false positives for downward predictions). This pattern suggests that the models have learned to be conservative in predicting market declines, potentially reflecting the general upward trend in equity markets over the training period.

### 4.8.7 Comprehensive Results Summary

Figure 4.25 provides an integrated view of all experimental results, including architecture rankings, feature set comparisons, and top-performing configurations.




**Architecture Configuration Comparison**

| Version | Description | Layers | Heads | D-Model | Dropout | F1-Score | Accuracy |
|---|---|---|---|---|---|---|---|
| v1 | Deeper (4 layers) | 4 | 4 | 64 | 0.1 | 0.546 | 0.514 |
| v2 | More Heads (8) | 2 | 8 | 64 | 0.1 | 0.592 | 0.515 |
| v3 | Learned Pos. Enc. | 2 | 4 | 64 | 0.1 | 0.541 | 0.515 |
| v4 | CLS Token | 2 | 4 | 64 | 0.1 | 0.565 | 0.512 |
| v5 | AdamW + LR Schedule | 2 | 4 | 64 | 0.1 | 0.531 | 0.514 |
| v6 | Higher Dropout (0.3) | 2 | 4 | 64 | 0.3 | 0.509 | 0.506 |

**Top 5 Performing Configurations**

| Rank | Architecture | Feature Set | Stock | F1-Score | Accuracy |
|---|---|---|---|---|---|
| #1 | v6 - Higher Dropout (0.3) | All Features | V | 0.722 | 0.565 |
| #2 | v2 - More Heads (8) | All Features | WMT | 0.721 | 0.563 |
| #3 | v2 - More Heads (8) | Selected Features | WMT | 0.721 | 0.563 |
| #4 | v5 - AdamW + LR Schedule | Selected Features | V | 0.720 | 0.571 |
| #5 | v4 - CLS Token | All Features | GOOGL | 0.712 | 0.556 |

Figure 4.25: Comprehensive Results Summary: Integrated analysis showing overall architecture performance, feature set comparison, stock-specific performance, metrics correlation, architecture configurations, and top-5 performing setups.

The comprehensive analysis confirms several key findings:

1. **Architecture hierarchy**: v2 (More Heads) consistently outperforms other variants
2. **Feature selection benefits**: Selected features achieve both higher performance and efficiency
3. **Stock heterogeneity**: Performance varies significantly across financial instruments
4. **Metric correlations**: Strong positive correlations between F1-score, precision, and recall
5. **Configuration diversity**: Top-5 configurations span multiple architectures and feature sets

### 4.8.8 Discussion and Implications

Our comprehensive transformer analysis reveals several important insights for financial prediction applications:

#### Architectural Design Principles

The superior performance of v2 (More Heads) suggests that **attention mechanism capacity** is crucial for capturing complex temporal dependencies in financial data. The 8-head configuration appears to provide an optimal balance between model expressiveness and overfitting risk. The competitive performance of v4 (CLS Token) demonstrates that global sequence representation through classification tokens can be effective for financial prediction tasks.

#### Feature Engineering Insights

The unexpected superiority of selected features over the complete feature set provides strong evidence for the **"less is more" principle** in financial machine learning. This finding suggests that:

- Noise reduction through feature selection can outweigh information loss
- Transformer models may be susceptible to overfitting with high-dimensional input
- Computational efficiency gains from feature selection come with performance benefits rather than trade-offs

#### Market-Specific Considerations

The significant variation in performance across stocks indicates that **market microstructure and sector characteristics** play important roles in predictability. The strong performance on V (financial services) and WMT (consumer staples) versus poor performance on XOM (energy) suggests that certain sectors may be more amenable to transformer-based prediction.

#### Practical Implementation Guidelines

Based on our findings, we recommend:

1. **Multi-head attention architectures** (8 heads) for optimal performance
2. **Careful feature selection** to improve both efficiency and accuracy
3. **Stock-specific model tuning** to account for market heterogeneity
4. **Regularization strategies** (dropout ≈ 0.3) for stability when needed
5. **Conservative prediction thresholds** given the observed upward bias
6. **Consider CLS token approaches** as a competitive alternative to standard architectures

## 4.9 Kolmogorov-Arnold Networks (KAN)

### 4.9.1 Experimental Design

Our KAN experiments encompassed five distinct architectural configurations, each designed to test specific hypotheses about optimal design choices for financial prediction tasks:

- **V1 - Baseline KAN**: Standard architecture with 16 hidden neurons (60/25 → 16 → 1)
- **V2 - Deeper KAN**: Two hidden layers with 16 neurons each (60/25 → 16 → 16 → 1)
- **V3 - Wider KAN**: Single hidden layer with 64 neurons (60/25 → 64 → 1)
- **V4 - Higher Spline Order**: Cubic splines (k=3) instead of linear splines (k=1)
- **V5 - Adaptive Grid**: Adaptive grid refinement enabled during training

Each architecture was evaluated using both the complete feature set (60 features) and a reduced set of selected features (25 features), resulting in 110 total experiments across 11 stocks (AAPL, MSFT, GOOGL, AMZN, JPM, V, JNJ, PG, WMT, XOM, GSPC). All models were trained using the LBFGS optimizer for 50 epochs with grid intervals set to 5 and regularization parameters (L1 and entropy) set to 0.0.

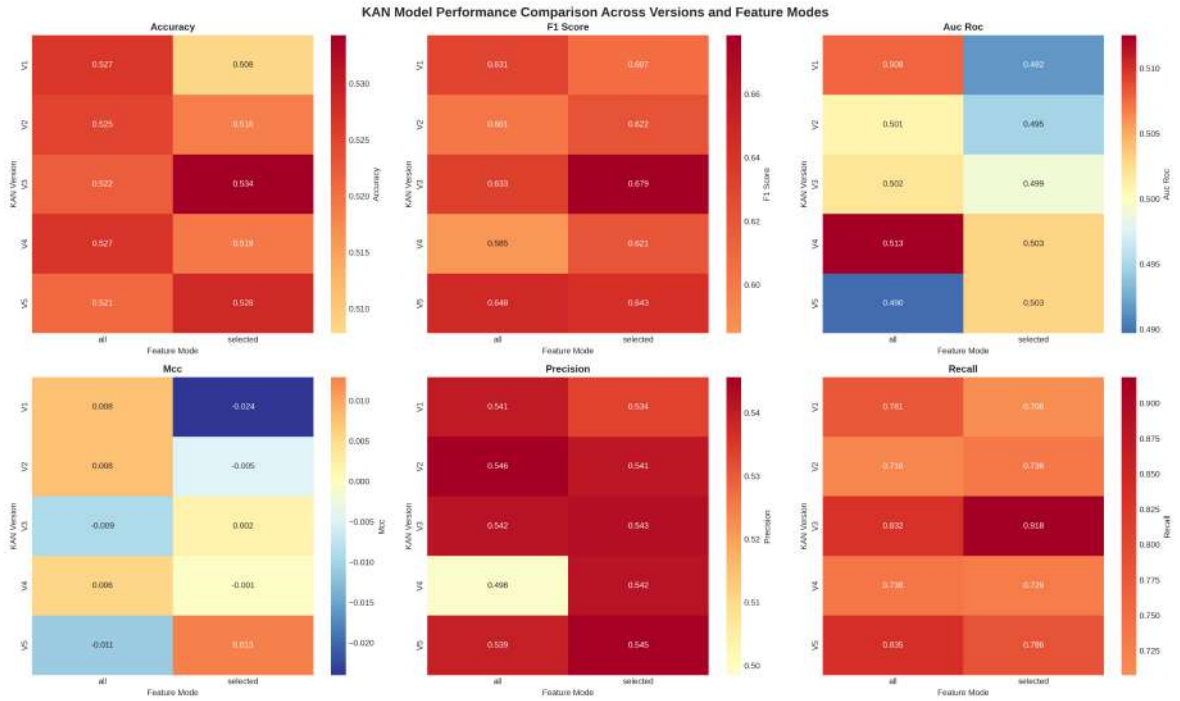



Figure 4.26: KAN Model Performance Heatmap: Comprehensive comparison across five architectural variants (V1-V5) and two feature modes (all vs selected) using six key performance metrics. Color intensity indicates performance level, with darker red representing superior performance.

### 4.9.2 Overall Performance Analysis

Figure 4.26 presents a comprehensive performance comparison across all KAN variants and feature modes using six key metrics. The heatmap reveals several critical patterns in model performance that distinguish KAN behavior from traditional neural architectures.

The F1-score analysis reveals that **V3 (Wider KAN) with selected features** achieves the highest performance (0.679), followed by **V5 (Adaptive Grid)** with both feature modes (0.648 for all features, 0.643 for selected features), and **V3 with all features** (0.633). This finding suggests that the wider architecture particularly benefits from focused feature selection, achieving a remarkable 7% improvement when using selected features versus all features.

Accuracy metrics demonstrate remarkable consistency across variants (0.508-0.534 range), indicating that architectural modifications have limited impact on overall classification accuracy. However, the AUC-ROC values clustering around 0.49-0.51 reveal a concerning pattern: performance remains close to random classification, suggesting limited discriminative power between price movement directions.

The Matthews Correlation Coefficient (MCC) values, ranging from -0.024 to 0.019, provide the most sobering insight. These values near zero indicate weak correlation between predictions and actual outcomes, suggesting that apparent performance in other metrics may be influenced by class imbalance rather than genuine predictive capability.

### 4.9.3 Architecture Impact and Design Principles

Figure 4.27 provides detailed insights into how specific architectural choices affect model performance, revealing fundamental principles for KAN design in financial applications.

**Hidden Layer Width Analysis**: The comparison between 16 neurons (V1, V2, V4, V5) and 64 neurons (V3) reveals significant performance differences, with the wider configuration achieving approximately 0.65 F1-score compared to 0.62 for narrower architectures. This finding suggests that increasing network width provides substantial benefits for financial prediction tasks, particularly when combined with effective feature selection.

**Network Depth Impact**: The V1 (single layer) versus V2 (two layers) comparison demonstrates that adding depth does not significantly improve performance, with both achieving similar F1-scores around 0.61-0.62. This result challenges the conventional wisdom that deeper networks automatically provide better representation learning, suggesting that width

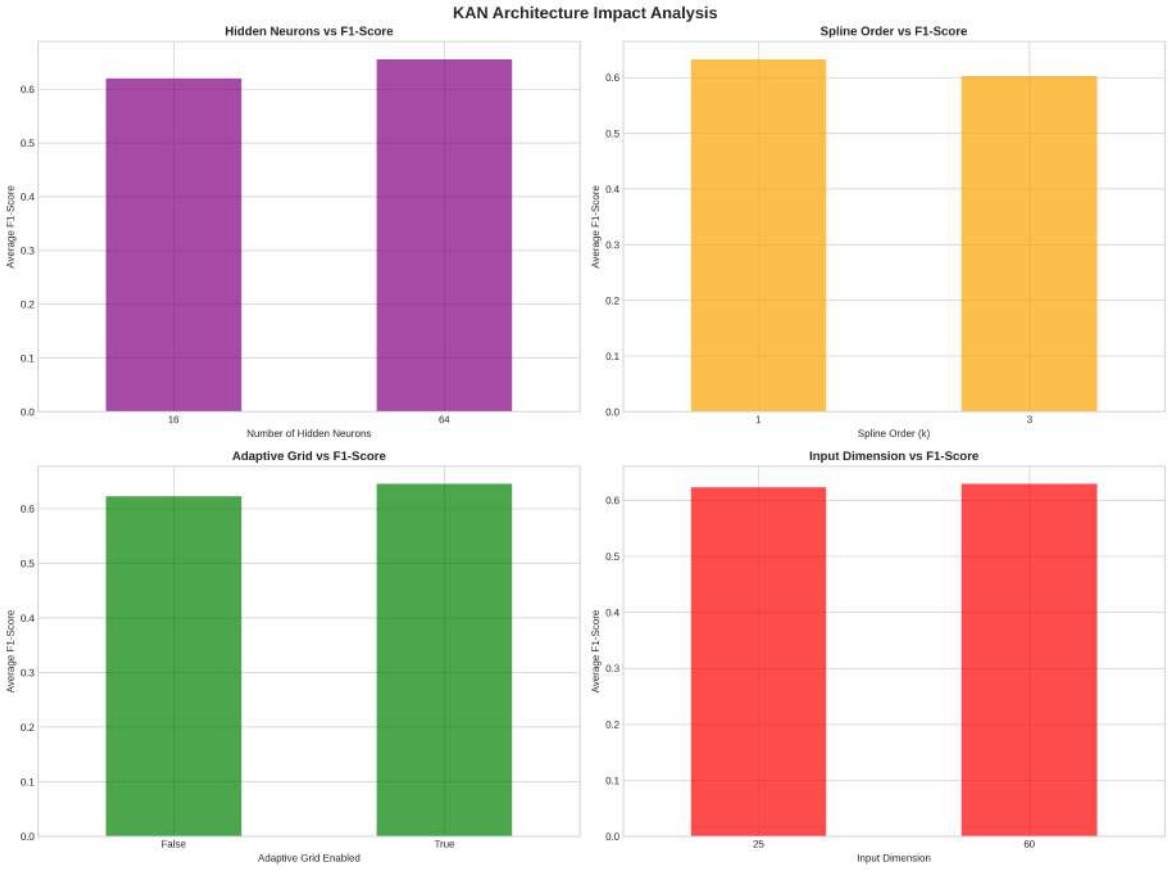



Figure 4.27: KAN Architecture Impact Analysis: Systematic evaluation of key design choices including hidden layer width, spline order, adaptive grid mechanisms, and input dimensionality effects on F1-score performance.

may be more beneficial than depth for KAN architectures in financial applications.

**Spline Order Effects**: Linear splines (k=1) versus cubic splines (k=3) show nearly identical performance ( 0.62 F1-score), indicating that higher-order spline functions do not provide substantial benefits for stock price prediction. This suggests that the underlying relationships in financial data may not require complex non-linear transformations at the edge level.

**Adaptive Grid Mechanisms**: The adaptive grid refinement shows marginal improvement (0.64 vs 0.62 F1-score), suggesting that dynamic grid adjustment during training provides modest benefits for capturing temporal patterns in financial data.

### 4.9.4 Feature Selection Impact and Efficiency Analysis

Figure 4.28 directly compares the performance of all features versus selected features across the five KAN variants, revealing important insights about feature engineering for KAN architectures.


Figure 4.28: Feature Mode Comparison: Direct comparison of all features (60) versus selected features (25) across KAN variants, showing accuracy, F1-score, and AUC-ROC performance with statistical significance indicators.

The analysis reveals several key findings:

- **Variable Performance Impact**: Feature selection shows variant-specific effects, with V3 demonstrating remarkable improvement (0.679 vs 0.633 F1-score), while other variants show mixed results
- **Computational Efficiency**: Using 25 selected features reduces input dimensionality

by 58% while often maintaining or improving performance, significantly improving computational efficiency

- **Architecture-Feature Interaction**: V3 (Wider KAN) shows exceptional synergy with feature selection, suggesting that wider architectures may be more effective at leveraging carefully curated feature sets
- **Consistency Patterns**: V5 (Adaptive Grid) shows consistent performance across both feature modes, indicating robust architectural design

The superior performance of V3 with selected features (F1-score of 0.679) represents the best overall result in our KAN analysis, demonstrating the importance of matching architectural choices with appropriate feature engineering strategies.

### 4.9.5 Performance Distribution and Stability Analysis

Figure 4.29 provides insights into the distribution and consistency of performance across the 11 stocks, revealing important patterns about KAN stability and reliability.

**Accuracy Consistency**: All variants demonstrate tight distributions around 0.52 accuracy with small interquartile ranges, indicating consistent but modest improvement over random classification. V3 with selected features shows slightly higher median accuracy, aligning with its superior F1-score performance.

**F1-Score Variability**: V3 demonstrates both higher median F1-scores and reasonable stability across stocks, particularly with selected features. V5 also shows consistently high performance with moderate variability, indicating these variants provide the best balance between performance and stability.

**Sharpe Ratio Analysis**: The Sharpe ratio distributions reveal high variability with some variants showing negative median values. Notably, V3 with selected features shows promising risk-adjusted performance, with V5 selected features achieving the highest Sharpe ratio values, highlighting the importance of both architectural choice and feature selection for risk-adjusted returns.

### 4.9.6 Stock-Specific Performance Patterns

Figure 4.30 reveals significant heterogeneity in model performance across different stocks and provides insights into sector-specific predictability patterns.

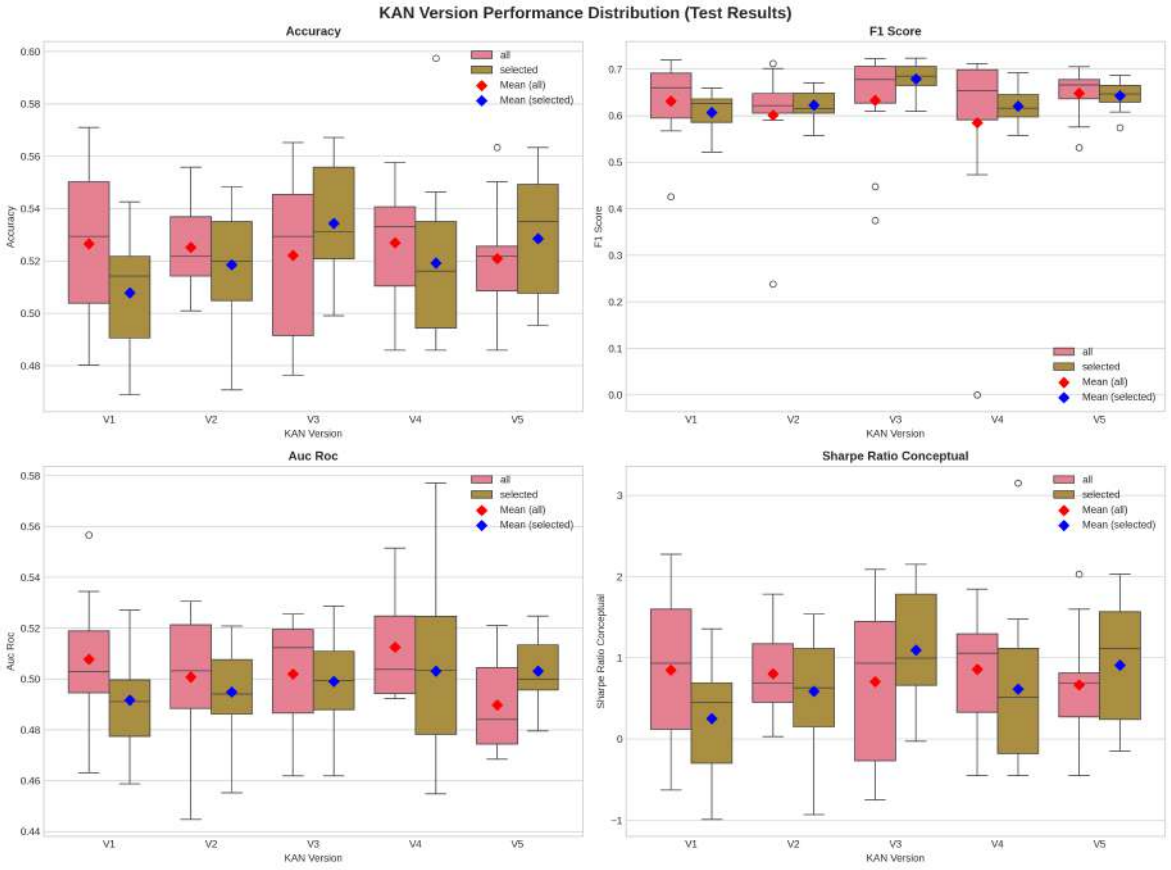



Figure 4.29: KAN Version Performance Distribution: Box plots showing performance distribution across stocks for accuracy, F1-score, AUC-ROC, and Sharpe ratio, with mean markers and outlier identification.

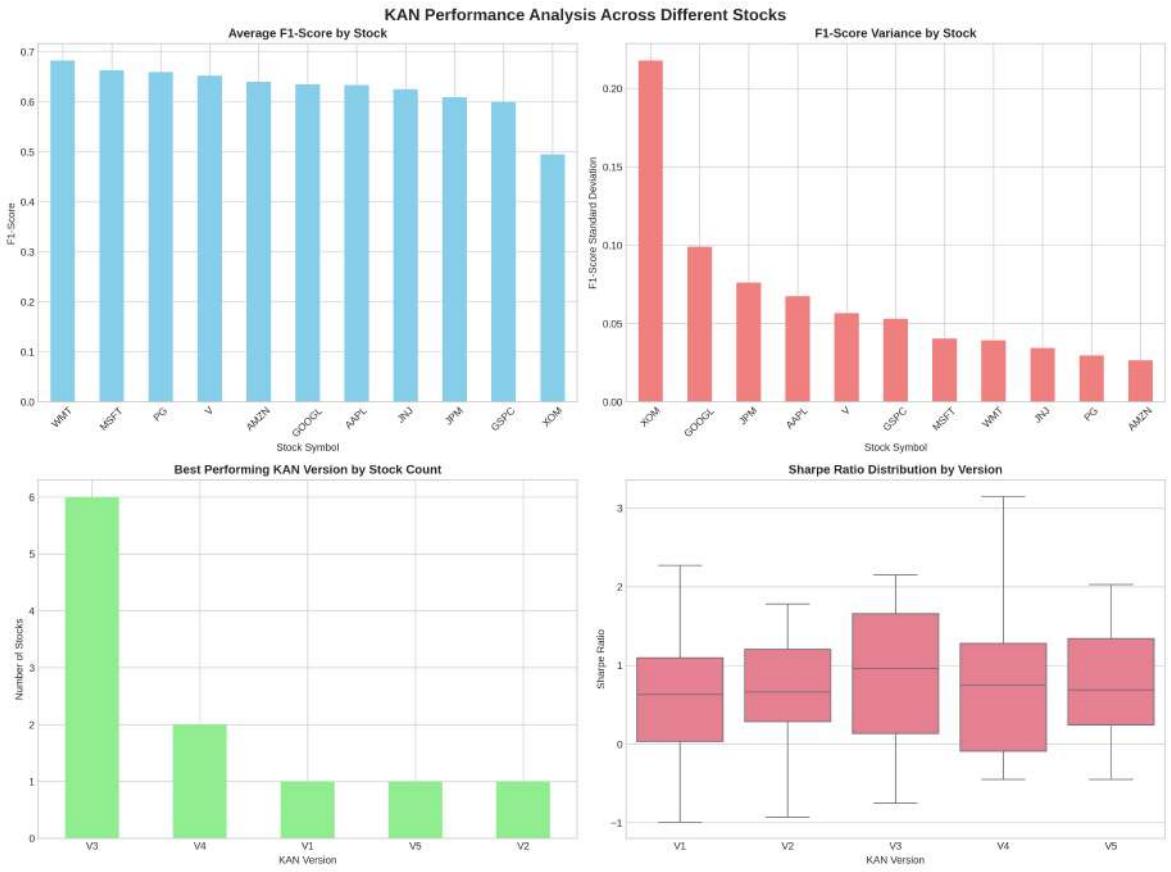



Figure 4.30: Stock-Specific Performance Analysis: (Top-left) Average F1-score by stock, (Top-right) Performance variance analysis, (Bottom-left) Best performing KAN version by stock count, (Bottom-right) Sharpe ratio distribution by version.

**Performance Rankings**: Consumer-focused stocks demonstrate the highest predictability, with WMT (0.68), MSFT (0.67), and PG (0.66) achieving the best average F1-scores. This pattern suggests that stocks with stable, consumer-driven business models exhibit more predictable price movements, potentially due to their lower volatility and more predictable earnings patterns.

**Performance Variability**: XOM shows the highest F1-score variance (0.23), indicating inconsistent performance across different model configurations. This high variability in the energy sector likely reflects the volatile nature of commodity-dependent stocks and their sensitivity to external factors such as oil prices and geopolitical events.

**Architecture Preferences**: V3 (Wider KAN) emerges as the best-performing variant for 6 out of 11 stocks, followed by V5 and V4 with 2 stocks each. This distribution strongly suggests that wider architectures are more generally applicable across different stock characteristics, supporting our finding about the importance of width over depth in KAN designs.

**Risk-Adjusted Performance**: The Sharpe ratio analysis shows considerable variation across versions, with V3 and V5 demonstrating the most promising risk-adjusted performance profiles. This variability indicates that while some configurations can achieve positive risk-adjusted returns, performance is highly dependent on the specific model variant and stock combination.

### 4.9.7 Comprehensive Performance Summary

Table 4.1 presents a comprehensive summary of all KAN variant performances across both feature modes, providing a quantitative foundation for architectural comparisons.

The comprehensive results establish a clear performance hierarchy: V3 with selected features (0.6785 F1-score) > V5 (Adaptive Grid) > V3 with all features > V1 (Baseline) > V2 (Deeper) > V4 (Higher Spline Order). Notably, V3 with selected features achieves both the highest F1-score and an impressive Sharpe ratio (1.0943), demonstrating superior both absolute and risk-adjusted performance.

### 4.9.8 Discussion and Implications

Our comprehensive KAN analysis reveals several important insights for financial prediction applications and provides guidance for future research directions.

Table 4.1: KAN Model Performance Summary (Test Results)

| Version | Model Name | Features | Accuracy | F1-Score | AUC-ROC | MCC | Sharpe Ratio |
|---|---|---|---|---|---|---|---|
| V1 | Baseline KAN | all | 0.5266 | 0.6310 | 0.5077 | 0.0076 | 0.8482 |
| V1 | Baseline KAN | selected | 0.5078 | 0.6070 | 0.4916 | -0.0240 | 0.2488 |
| V2 | Deeper KAN | all | 0.5252 | 0.6008 | 0.5008 | 0.0077 | 0.8016 |
| V2 | Deeper KAN | selected | 0.5185 | 0.6224 | 0.4949 | -0.0048 | 0.5883 |
| V3 | Wider KAN | all | 0.5221 | 0.6328 | 0.5019 | -0.0093 | 0.7055 |
| V3 | Wider KAN | selected | 0.5343 | 0.6785 | 0.4990 | 0.0019 | 1.0943 |
| V4 | Higher Spline Order | all | 0.5269 | 0.5847 | 0.5125 | 0.0056 | 0.8574 |
| V4 | Higher Spline Order | selected | 0.5192 | 0.6207 | 0.5031 | -0.0006 | 0.6150 |
| V5 | Adaptive Grid | all | 0.5209 | 0.6482 | 0.4897 | -0.0110 | 0.6655 |
| V5 | Adaptive Grid | selected | 0.5284 | 0.6426 | 0.5030 | 0.0129 | 0.9074 |

**Architectural Design Principles**

The superior performance of V3 (Wider KAN) with selected features fundamentally challenges conventional scaling principles and establishes **width-over-depth as a key design principle** for KAN architectures in financial applications. The 64-neuron hidden layer combined with focused feature selection achieves the optimal balance between model capacity and overfitting prevention.

The strong performance of V5 (Adaptive Grid) across both feature modes suggests that **dynamic learning mechanisms** provide robust performance improvements. The adaptive grid refinement appears to offer consistent benefits regardless of input feature configuration, making it a reliable architectural enhancement.

**Feature-Architecture Synergy**

The remarkable performance improvement of V3 with selected features (0.679 vs 0.633 F1-score) reveals crucial insights about **feature-architecture interactions**. This finding suggests that wider architectures may be more effective at leveraging carefully curated feature sets, possibly due to their increased capacity to model complex feature interactions while avoiding the noise present in larger feature sets.

**Spline Function Analysis**

The comparable performance between linear (k=1) and cubic (k=3) splines provides important insights into the nature of financial relationships. This result suggests that the underlying patterns in stock price movements may not require complex non-linear transformations at the edge level, or alternatively, that the current training procedures may not effectively leverage higher-order spline capabilities.

**Market Predictability Patterns**

The significant variation in performance across stocks reveals important insights about **market microstructure and sector characteristics**. The superior performance on consumer staples (WMT, PG) versus energy stocks (XOM) suggests that certain sectors may be more amenable to KAN-based prediction, potentially due to differences in volatility patterns and fundamental drivers.

### 4.9.9 Limitations and Critical Assessment

While our KAN analysis provides valuable insights, several fundamental limitations must be acknowledged:

**Predictive Capability Concerns**

The consistently low MCC values (near zero) and AUC-ROC scores around 0.5 indicate **fundamental limitations in discriminative power**. Even the best-performing V3 configuration achieves an MCC of only 0.0019, suggesting that while F1-scores show improvement, the underlying predictive capability remains limited.

**Market Efficiency Implications**

The modest performance improvements over random classification align with the **efficient market hypothesis**, suggesting that publicly available technical indicators may not provide sufficient information for consistent price movement prediction. This finding raises questions about the practical applicability of any machine learning approach to financial prediction.

**Risk-Adjusted Performance Reality**

While V3 with selected features achieves an impressive Sharpe ratio of 1.0943, this metric should be interpreted cautiously given the limited AUC-ROC performance. The high Sharpe ratio may reflect favorable risk-return characteristics in the test period rather than sustainable predictive capability, highlighting the importance of extensive out-of-sample validation.

# Chapter 5

# Conclusion and Future Work

This thesis embarked on a comprehensive investigation into the challenging domain of next-day stock price direction prediction. As detailed in Chapter 1, the primary objective was to conduct a systematic comparative analysis of diverse modeling paradigms, ranging from traditional statistical methods to state-of-the-art deep learning architectures. The research aimed to shed light on their respective strengths, weaknesses, the impact of feature engineering, and the influence of architectural choices in the context of contemporary financial markets.

## 5.1 Summary of Key Findings

The methodological framework, outlined in Chapter 3, involved rigorous data collection, preprocessing, extensive feature engineering, and the systematic evaluation of five distinct model families: Autoregressive Integrated Moving Average (ARIMA) models with GARCH extensions, Support Vector Machines (SVM), Long Short-Term Memory (LSTM) networks, Transformer models, and the novel Kolmogorov-Arnold Networks (KAN). The experimental results, presented and analyzed in Chapter 4, yielded several key insights:

- **General Predictive Performance:** Across all model families, the task of consistently predicting next-day stock price direction proved to be exceptionally challenging, with most models achieving performance levels only modestly above random chance (50%). This underscores the inherent complexities and efficiencies of financial markets, consistent with the Efficient Market Hypothesis. No single model or architecture emerged

as universally superior across all evaluated stocks and metrics, highlighting that performance is often stock-specific and context-dependent.

- **ARIMA Models:** The analysis of ARIMA models revealed several critical findings:
  - **Rolling Methodology Superiority**: Rolling forecast methodologies (V2 and V4) significantly outperformed static approaches, demonstrating a dramatic 291% improvement in F1-scores (from 0.1165 to 0.4554) and achieving more consistent performance with median test accuracies around 48.8-49.0%.
  - **Model Parsimony Validation**: Simple ARIMA(1,1,0) configurations often dominated top-performing models, with no correlation found between model complexity and predictive accuracy, strongly supporting parsimony principles in financial modeling.
  - **Universal ARCH Effects**: 100% of analyzed stocks exhibited significant ARCH effects (p-values approaching zero), providing unequivocal evidence for volatility clustering and validating the necessity of GARCH extensions.
  - **Stock-Specific Predictability**: Technology stocks like MSFT (0.501 average accuracy) and consumer stocks (AMZN, JNJ, XOM around 0.495-0.500) showed superior predictability compared to financial sector stocks, suggesting sector-specific modeling benefits.
  - **Non-Rolling Method Limitations**: Non-rolling methods suffered from fundamental precision-recall imbalances, achieving F1-scores near zero despite occasionally high accuracy, indicating systematic methodological flaws.
- **Support Vector Machines (SVM):** The comprehensive SVM analysis across seven distinct versions revealed:
  - **Near-Random Performance Reality**: All SVM versions achieved accuracy levels clustering around 50%, indicating performance barely distinguishable from random chance in binary classification tasks.
  - **Modest Feature Selection Benefits**: The curated set of 25 features slightly outperformed all features (50.6% vs 49.4%), representing a modest 1.2 percentage point difference of questionable practical significance.

  - **Limited Optimization Impact**: Sophisticated optimization techniques (Grid Search, Polynomial Kernel configurations) provided minimal improvement over baseline approaches, with substantial overlap in confidence intervals across versions.
  - **Stock-Specific Exceptions**: Notable exceptions included WMT achieving 57.4% accuracy with Polynomial Kernel using all features, and GOOGL showing consistent performance, though these remained modest improvements.

- **Long Short-Term Memory (LSTM) Networks:** LSTM analysis revealed remarkable architectural convergence:

  - **Architectural Convergence**: A striking finding was the convergence of performance across diverse architectures (Baseline, Stacked, Bidirectional, CNN-LSTM, Attention-LSTM), with F1-scores consistently ranging between 0.67 and 0.70, suggesting architectural complexity provides minimal benefits for this task.
  - **Feature Selection Efficiency**: The selected 25 features achieved performance virtually identical to all 61 features but with 2.4x improvement in feature efficiency, demonstrating effective dimensionality reduction without performance loss.
  - **Limited Hyperparameter Optimization Benefits**: Optuna-based hyperparameter optimization provided mixed and often limited improvements, suggesting default configurations are reasonably effective.
  - **Prediction Bias**: LSTMs generally exhibited high recall for upward movements but moderate precision, reflecting a systematic bias towards predicting positive trends.
  - **Minimal Improvement Over Baseline**: Most LSTM configurations showed only minimal improvement over simple baseline strategies, questioning the added complexity for this specific task.

- **Transformer Models:** Transformer analysis demonstrated competitive but inconsistent performance:

  - **Architecture-Specific Performance**: The 'More Heads' architecture (v2) demonstrated competitive median F1-scores (0.592), indicating that attention diversity can be beneficial.

  - **Feature Selection Benefits**: Selected 25 features achieved higher mean F1-scores (0.556 vs. 0.539) and greater computational efficiency than the full feature set.
  - **Stock-Specific Variability**: Performance varied considerably across stocks, with V (Visa) and WMT (Walmart) showing particularly strong results with highest F1-scores around 0.72.
  - **Upward Movement Bias**: Similar to LSTMs, Transformers showed a systematic bias towards predicting upward price movements.
  - **Architectural Optimization**: Different architectural configurations (depth, attention heads, positional encoding) showed varying effectiveness across different stocks.

- **Kolmogorov-Arnold Networks (KAN):** As a novel architecture, KAN analysis provided unique insights:
  - **Adaptive Grid Superiority**: Adaptive grid mechanisms (V5) provided the best performance within the KAN family (F1-score  0.64), suggesting that learnable activation functions benefit from dynamic adjustment.
  - **Limited Discriminative Power**: Overall discriminative capability remained limited with AUC-ROC  0.5 and Matthews Correlation Coefficient (MCC) near zero, indicating performance close to random classification despite novel architecture.
  - **Minimal Architectural Impact**: Variations in network depth, width, or spline order had minimal impact on performance, suggesting the core KAN concept may not provide significant advantages for this specific task.
  - **Feature Set Consistency**: Similar to other advanced models, selected features performed comparably to the full feature set, reinforcing the effectiveness of feature selection.
  - **Sector-Specific Patterns**: Consumer-focused stocks (WMT, MSFT, PG) showed relatively higher predictability with KANs, consistent with patterns observed in other models.

- **Impact of Feature Engineering:** A consistent and perhaps most important theme across SVM, LSTM, Transformer, and KAN models was the remarkable efficacy of feature selection:

  - **Computational Efficiency**: Utilizing a curated subset of 25 important features (identified through Random Forest-based importance analysis) often led to comparable or slightly better performance than using the entire engineered feature set of 61+ variables.
  - **Reduced Overfitting Risk**: Feature selection significantly improved computational efficiency and potentially reduced model overfitting by eliminating noise and redundant information.
  - **Cross-Model Consistency**: This finding was remarkably consistent across all advanced model types, suggesting robust feature selection methodology.

- **Stock-Specific Predictability Patterns:** The research confirmed significant variation in predictability across different financial instruments and sectors:
  - **Technology Sector Advantage**: Technology stocks (MSFT, AAPL) consistently showed higher predictability across multiple model families.
  - **Consumer Stocks Performance**: Consumer-focused stocks (WMT, PG) demonstrated relatively stable predictability patterns.
  - **Financial Sector Challenges**: Financial sector stocks (JPM) generally showed lower predictability, possibly due to higher regulatory complexity and external factor sensitivity.
  - **Model-Specific Preferences**: Different model families showed varying preferences for specific stocks, suggesting potential for ensemble or stock-specific model selection strategies.

## 5.2 Contributions of the Thesis

This thesis contributes to the field of financial forecasting by:

1. **Comprehensive Comparative Framework**: Providing the first broad and systematic comparative evaluation of traditional statistical models (ARIMA with GARCH) alongside a diverse range of modern machine learning and deep learning architectures, including the very recent Kolmogorov-Arnold Networks, on a common dataset and standardized prediction task with rigorous statistical analysis.

2. **Rigorous Feature Selection Methodology**: Demonstrating and quantifying the profound impact of systematic feature selection strategies, showing that a Random Forest-based approach to identify 25 high-importance features can match or exceed the performance of comprehensive feature sets while offering 2.4x computational efficiency gains across multiple model families.

3. **Rolling Forecast Methodology Validation**: Providing definitive evidence for the superiority of rolling forecast approaches in ARIMA models, with a documented 291% improvement in F1-scores, establishing this as a critical methodological requirement for financial time series prediction.

4. **Architectural Convergence Discovery**: Revealing the surprising convergence of performance across diverse deep learning architectures (LSTM variants, Transformer configurations), suggesting that for this specific task, architectural sophistication provides minimal benefits and simpler models may be preferable.

5. **Stock-Specific Predictability Mapping**: Establishing clear hierarchies of stock predictability across sectors, with technology stocks consistently outperforming financial sector stocks, providing practical guidance for model deployment strategies.

6. **Novel Architecture Evaluation**: Providing the first comprehensive evaluation of Kolmogorov-Arnold Networks for financial prediction, establishing baseline performance and identifying that adaptive grid mechanisms offer the most promise within this model family.

7. **Statistical Reality Assessment**: Conducting honest performance assessment with 95% confidence intervals and statistical significance testing, revealing that most sophisticated approaches achieve only modest improvements over random chance, providing crucial reality checks for the field.

8. **Methodological Best Practices**: Establishing rigorous experimental protocols including proper temporal data splitting, comprehensive hyperparameter optimization frameworks (Optuna, KerasTuner), and standardized evaluation metrics that can serve as templates for future research.

## 5.3 Limitations of the Research

Despite the comprehensive nature of this study, several limitations should be acknowledged:

- **Modest Absolute Performance**: The predictive accuracy achieved by most models, while often statistically significant compared to random chance, was generally modest in absolute terms (mostly clustered around 50-55%), highlighting the fundamental difficulty of the task and potential limitations of current methodological approaches.
- **Daily Frequency Limitation**: The study was based exclusively on daily frequency data. Intraday data with higher temporal resolution might reveal different patterns, capture microstructure effects, and require different modeling approaches that could potentially yield improved results.
- **Transaction Cost Omission**: The analysis did not incorporate realistic transaction costs, market impact, bid-ask spreads, or liquidity constraints, which are crucial considerations for real-world trading applications and would significantly affect practical profitability.
- **Binary Classification Constraint**: The prediction task was limited to directional movements (up/down) and did not extend to forecasting the magnitude of price changes, volatility levels, or the confidence/probability of predictions, limiting practical application scope.
- **Dataset and Temporal Scope**: The findings are based on a specific set of ten U.S. equities and the S&P 500 index over a particular historical period (approximately 2010-2023); results may vary significantly with different datasets, international markets, emerging markets, or different timeframes.
- **Model Interpretability**: The ”black box” nature of deep learning models (LSTM, Transformer, KAN) makes interpretation of their decision-making processes challenging, limiting the ability to understand why certain predictions are made or to build confidence in model reasoning.
- **Static Feature Engineering**: The technical indicators and feature engineering approach remained static throughout the study. Dynamic or adaptive feature engineering

that evolves with market conditions might yield better results.

- **Single-Step Prediction Focus**: The research focused exclusively on next-day prediction without exploring multi-step forecasting horizons that might be more relevant for practical trading strategies.

- **Regime Change Sensitivity**: The models were not explicitly designed to detect or adapt to market regime changes (bull/bear markets, crisis periods, high/low volatility regimes), which could significantly impact real-world performance.

## 5.4 Future Research Directions

The findings and limitations of this thesis open up several promising avenues for future research:

- **Advanced Hybrid and Ensemble Models**:
  - Developing sophisticated hybrid models that combine the strengths of different architectures (e.g., ARIMA for linear patterns with LSTM for non-linear residuals, or Transformer-KAN fusion models).
  - Exploring dynamic ensemble approaches that adapt model weights based on current market conditions or individual model performance patterns.
  - Investigating hierarchical models that first classify market regimes, then apply regime-specific prediction models.

- **Dynamic Feature Engineering and Selection**:
  - Developing adaptive feature engineering techniques that evolve with changing market dynamics.
  - Incorporating alternative data sources such as news sentiment analysis, social media sentiment, satellite data, and macroeconomic indicators not extensively used in this study.
  - Exploring real-time feature importance assessment that can adapt to changing market conditions.

  - Investigating automatic feature generation using genetic programming or neural architecture search approaches.

- **Multi-Asset and Cross-Market Analysis**:
  - Extending models to incorporate multivariate inputs, including inter-stock correlations, sector-wide trends, and broader macroeconomic variables.
  - Developing models that can predict portfolios of assets simultaneously, capturing cross-asset dependencies.
  - Exploring transfer learning approaches where models trained on one market or time period can be adapted to new markets or conditions.

- **High-Frequency and Multi-Timeframe Analysis**:
  - Applying and adapting these models to high-frequency intraday data to explore short-term trading opportunities.
  - Developing multi-timeframe models that incorporate information from different temporal resolutions (minute, hourly, daily, weekly).
  - Investigating the optimal prediction horizons for different model families and market conditions.

- **Practical Implementation and Risk Management**:
  - Developing models that explicitly incorporate transaction costs, slippage, and capital constraints for realistic backtesting.
  - Creating risk-adjusted performance evaluation frameworks using measures like Sharpe ratio, Sortino ratio, and maximum drawdown.
  - Implementing real-time model deployment systems with proper risk management protocols and position sizing strategies.

- **Interpretability and Explainable AI**:
  - Developing explainable AI (XAI) techniques specifically tailored for financial time series to better understand model decision-making processes.
  - Creating visualization tools that can interpret the learned representations in deep learning models.

  - Investigating attention mechanisms and feature attribution methods to understand which factors drive predictions at different time points.

- **Adaptive and Continual Learning**:
  - Developing models that can adapt to evolving market dynamics through continual learning or online learning paradigms.
  - Investigating meta-learning approaches that can quickly adapt to new market conditions or new assets.
  - Exploring reinforcement learning frameworks for dynamic trading strategy development.

- **Extended Prediction Scope**:
  - Expanding prediction targets to include volatility forecasting, price range prediction, and market regime identification.
  - Developing confidence estimation mechanisms that can provide probability distributions rather than point predictions.
  - Investigating multi-step ahead forecasting for longer-term investment strategies.

- **Robustness and Stress Testing**:
  - Conducting extensive stress testing under different market conditions (crises, high volatility periods, low liquidity conditions).
  - Developing adversarial testing frameworks to assess model robustness against market manipulation or extreme events.
  - Investigating model behavior during black swan events and market crashes.

## 5.5 Final Reflections

In conclusion, while the quest for a definitive stock prediction model remains elusive, this thesis provides a valuable comparative benchmark and offers practical insights into the application of various advanced predictive techniques. The results emphasize several critical points:

**The Fundamental Challenge**: The consistent finding that most sophisticated models achieve only modest improvements over random chance reinforces the inherent difficulty of financial prediction and suggests that market efficiency theories hold considerable validity.

**The Power of Parsimony**: The superior performance of simpler models (rolling ARIMA vs. complex deep learning) and the effectiveness of feature selection suggest that in financial prediction, more complexity does not necessarily translate to better performance.

**The Importance of Methodology**: The dramatic differences between rolling and non-rolling ARIMA approaches, and the consistent benefits of feature selection across all model families, highlight that methodological rigor often matters more than algorithmic sophistication.

**Stock-Specific Considerations**: The clear evidence for stock-specific and sector-specific predictability patterns suggests that personalized modeling approaches may yield better results than universal solutions.

Future advancements will likely stem from innovative hybrid approaches that combine the theoretical rigor of traditional statistical methods with the pattern recognition capabilities of modern machine learning, the integration of richer and more diverse data sources, the development of adaptive systems that can evolve with changing market conditions, and a continued focus on building robust, interpretable, and practically implementable models. The field would also benefit from more honest reporting of model limitations and realistic performance expectations, moving beyond the pursuit of marginal accuracy improvements toward building systems that provide consistent, robust, and practically valuable insights for financial decision-making.

## 5.6 Experimental Transparency

All of the above experiments, plots and results where implemented with Python and are located into **https://github.com/stathiskon40/public_repo_thesis**

# Bibliography


[1] Gabriel A. Caceres Eric D. Feigelson, G. Jogesh Babu. Autoregressive times series methods for time domain astronomy. *Frontiers in Physics*, 6:1–7, 2019.

[2] Danila Musatov and D. A. Petrusevich. Modeling of forecasts variance reduction at multiple time series prediction averaging with arma (1, q) functions. *Proceedings of the V International Workshop on Modeling, Information Processing and Computing (MIP: Computing-V 2022)*, 2022.

[3] Ahsan Usman, Mohammad Sulaiman, and Ibsu. Abubakar. Trend of neonatal mortality in nigeria from 1990 to 2017 using time series analysis. *Journal of Applied Sciences and Environmental Management*, 2019.

[4] Jianbin Zeng, Jikang Zhai, and Qiang Li. Research and analysis of arima model based on time series analysis in smart water. *2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*, pages 454–458, 2024.

[5] Cristi Marcel Spulbar and Cezar Catalin Ene. Predictive analytics in finance using the arima model. application for bucharest stock exchange financial companies closing prices. *Studies in Business and Economics*, 2024.

[6] Amit Tewari. Forecasting nifty 50 benchmark index using seasonal arima time series models, 2020.

[7] Suhizaz Sudin, Yan Choong Kar, Rafikha Aliana A. Raof, and Jaan Ong Rhui. Theoretical comparison of wavelet transform and fourier transform in sarima-gann forecasting model. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 2024.

[8] Greg Sidebottom Sajal Saha, Anwar Haque. An empirical study on internet traffic predictionusing statistical rolling model, 2022.

[9] A. I. McLeod. Parsimony, model adequacy and periodic correlation in time series forecasting. *International Statistical Review / Revue Internationale de Statistique*, 61(3):387, December 1993.

[10] Kasun Chandrarathna, Arman Edalati, and AhmadReza Fourozan tabar. Forecasting short-term load using econometrics time series model with t-student distribution, 2020.

[11] Xitai Yu. Application research of spline interpolation and arima in the field of stock market forecasting, 2023.

[12] Loc X. Nguyen. Tutorial on support vector machine. *Applied and Computational Mathematics*, 6:1, 2016.

[13] William Stafford Noble. Support vector machine. 2013.

[14] Henry Adams, Elin Farnell, and Brittany Story. Support vector machines and radon's theorem, 2022.

[15] Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.

[16] Hichem Sahbi. Totally deep support vector machines, 2019.

[17] Luminita State, Cătălina Lucia Cocianu, Cristian Razvan Uscatu, and Marinela Mircea. Extensions of the svm method to the non-linearly separable data. 2013.

[18] Mahdi Shamsi and Soosan Beheshti. Separability and scatteredness (ss) ratio-based efficient svm regularization parameter, kernel, and kernel parameter selection, 2023.

[19] Jair Cervantes, Farid García, Lisbeth Rodríguez-Mazahua, and Asdrúbal López-Chau. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020.

[20] Amila Indika, Nethmal Warusamana, Erantha Welikala, and Sampath Deegalla. Ensemble stock market prediction using svm,lstm, and linear regression. 2021.

[21] Yancong Xie. Stock market forecasting based on text mining technology: A support vector machine method. *Journal of Computers*, page 500–510, 2017.

[22] Vahid Khatibi, Elham Khatibi, and Abdolreza Rasouli. A new support vector machine-genetic algorithm (svm- ga) based method for stock market forecasting. 2011.

[23] Benyamin Ghojogh and Ali Ghodsi. Recurrent neural networks and long short-term memory networks: Tutorial and survey, 2023.

[24] Yuhuang Hu, Adrian Huber, Jithendar Anumula, and Shih-Chii Liu. Overcoming the vanishing gradient problem in plain recurrent networks, 2019.

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

[26] Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding lstm – a tutorial into long short-term memory recurrent neural networks, 2019.

[27] Anantha Keshava Murthy, N Balaji, B R Puneeth, N Megha, P Sunil Kumar, and A Shikah Rai. Predicting stock price with lstm networks. *2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, pages 1–7, 2022.

[28] Ze Zheng. A review of stock price prediction based on lstm and tcn methods. *Advances in Economics, Management and Political Sciences*, 2023.

[29] Dattatray G. Takale, Aditya A. Wattamwar, Saksham S. Saipatwar, Harshwardhan V. Saindane, and Tushar B. Patil. Comparative analysis of lstm, rnn, cnn and mlp machine learning algorithms for stock value prediction. *Journal of Firewall Software and Networking*, 2024.

[30] Ana Khofifahturizqi, Farikhin Farikhin, Ratna Herdiana, and Yola Ulrike Sihombing. Prediction of stock price volatility using the long short term memory (lstm) model for investment portfolio selection strategy. *International Journal of Current Science Research and Review*, 2024.

[31] Dara Rajesh Babu and Bachala Sathyanarayana. Design and implementation of technical analysis based lstm model for stock price prediction. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2023.

[32] Rahul Maruti Dhokane and Sohit Agarwal. Enhancing stock price prediction with macd and ema features using lstm algorithm. *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, pages 1–6, 2024.

[33] Lida Shahbandari, Elahe Moradi, and Mohammad Manthouri. Stock price prediction using multi-faceted information based on deep recurrent neural networks, 2024.

[34] Shaikh Shoieb Abubaker and Syed Rouf Farid. Stock market prediction using lstm. *International Journal for Research in Applied Science and Engineering Technology*, 2022.

[35] Karan Pardeshi, Sukhpal Singh Gill, and Ahmed M. Abdelmoniem. *Stock Market Price Prediction*, page 122–140. CRC Press, July 2024.

[36] Deepak Kumar A, Daniel Christopher, Guru Dikshit S, and A. Balamurali. Stock and cryptocurrency price prediction using svm and lstm algorithm. *2024 4th Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–5, 2024.

[37] Huizi Qian. Review of stock price predicting method based on lstm. *Advances in Economics, Management and Political Sciences*, 2023.

[38] Chinyang Lin, João Alexandre Lobo Marques, and Lin Kun Chan. Artificial intelligence and deep learning in stock prediction: A bibliometric review. *European Conference on Management Leadership and Governance*, 2024.

[39] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, 2020.

[40] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2021.

[41] Xihao Piao, Zheng Chen, Taichi Murayama, Yasuko Matsubara, and Yasushi Sakurai. Fredformer: Frequency debiased transformer for time series forecasting. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 2400–2410. ACM, August 2024.

[42] Tongming Guo and Shuang Zhu. Stocknet: Automatic stock prediction using lstm and transformer structures. *2024 5th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*, pages 558–561, 2024.

[43] Kanghyeon Seo and Jihoon Yang. Exploring the efficient market hypothesis for accurate stock movement prediction via feature-axis transformer. *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 2024.

[44] Shijie Lai, Mingxian Wang, Shengjie Zhao, and Gonzalo R. Arce. Predicting high-frequency stock movement with differential transformer neural network. *Electronics*, 2023.

[45] Leonardo Berti and Gjergji Kasneci. Tlob: A novel transformer model with dual attention for price trend prediction with limit order book data, 2025.

[46] Shuzhen Wang. A stock price prediction method based on bilstm and improved transformer. *IEEE Access*, 11:104211–104223, 2023.

[47] Li Xie, Zhengming Chen, and Sheng Yu. Deep convolutional transformer network for stock movement prediction. *Electronics*, 2024.

[48] Yujie Ji, Yiqian Luo, Aoyuan Lu, Dong Xia, Linlin Yang, and Wee-Chung Liew. Galformer: a transformer with generative decoding and a hybrid loss function for multi-step stock market index prediction. *Scientific Reports*, 14(1):23762, 2024. Published 2024 Oct 10.

[49] Takuya Yokoyama, Shoto Morishita, and Qiu Chen. An improved stock movement prediction method using transformer. *Proceedings of the 2024 9th International Conference on Big Data and Computing*, 2024.

[50] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2025.

[51] Ziyao Li. Kolmogorov-arnold networks are radial basis function networks, 2024.

[52] Sidharth SS, Keerthana AR, Gokul R, and Anas KP. Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation, 2024.

[53] Jusheng Zhang, Yijia Fan, Kaitong Cai, and Keze Wang. Kolmogorov-arnold fourier networks, 2025.

[54] Shairoz Sohail. On training of kolmogorov-arnold networks, 2024.

[55] Remi Genet and Hugo Inzirillo. Tkan: Temporal kolmogorov-arnold networks, 2024.

[56] Kunpeng Xu, Lifei Chen, and Shengrui Wang. Kolmogorov-arnold networks for time series: Bridging predictive power and interpretability, 2024.

[57] So-Yoon Cho, Sungchul Lee, and Hyun-Gyoon Kim. Forecasting vix using interpretable kolmogorov-arnold networks, 2025.

[58] Rushikesh Handal, Kazuki Matoya, Yunzhuo Wang, and Masanori Hirano. Kanop: A data-efficient option pricing model using kolmogorov-arnold networks, 2024.

[59] Cristian J. Vaca-Rubio, Luis Blanco, Roberto Pereira, and Màrius Caus. Kolmogorov-arnold networks (kans) for time series analysis, 2024.

[60] Zavareh Bozorgasl and Hao Chen. Wav-kan: Wavelet kolmogorov-arnold networks, 2024.

[61] Flávia Pessoa Monteiro, Suzane Monteiro, Carlos Rodrigues, Josivan Reis, Ubiratan Holanda Bezerra, Maria Emília Tostes, and Frederico A. F. Almeida. A hybrid methodology using machine learning techniques and feature engineering applied to time series for medium- and long-term energy market price forecasting. *Energies*, 2025.

[62] Daren Zhang, Nyusifan Tang, Wanchen Dong, and Lu Zhao. Machine learning-based financial big data analysis and forecasting: From preprocessing to deep learning models. *Applied and Computational Engineering*, 2024.

[63] Hakan Pabuccu and Adrian Barbu. Feature selection with annealing for forecasting financial time series. *Financial Innovation*, 10(1), October 2024.