Adaptive Model Selection in Stock Market Prediction: A Modular and Scalable Big Data Analytics Approach

MohammadEhsan Akhavanpour

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Part of the Computer Sciences Commons

Recommended Citation
https://scholar.uwindsor.ca/etd/9153

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
Adaptive Model Selection in Stock Market Prediction: A Modular and Scalable Big Data Analytics Approach

By

MohammadEhsan Akhavanpour

A Thesis
Submitted to the Faculty of Graduate Studies through the School of Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Science at the University of Windsor

Windsor, Ontario, Canada

2023

©2023 MohammadEhsan Akhavanpour
Adaptive Model Selection in Stock Market Prediction: A Modular and Scalable Big Data Analytics Approach

by

MohammadEhsan Akhavanpour

APPROVED BY:

__________________________________________________________
M. Hassanzadeh
Department of Electrical and Computer Engineering

__________________________________________________________
L. Rueda
School of Computer Science

__________________________________________________________
S. Samet, Advisor
School of Computer Science

December 15, 2023
DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I certify that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.
ABSTRACT

In today’s globalized economy, financial markets are more interconnected than ever, generating vast amounts of data from thousands of sources every second. The need to accurately analyze and interpret this data is crucial for investors, analysts, and researchers alike. Traditional models for market prediction are limited by their ability to adapt to the real-time nature and ‘big data’ dimensions of these complex financial datasets. To address these challenges, this thesis proposes and implements a novel framework that combines Apache Kafka with a microservices framework. This framework offers a scalable, real-time solution for financial market prediction that effectively manages the 5Vs of big data: Volume, Velocity, Variety, Veracity, and Value. Apache Kafka’s event-streaming capabilities serve as the backbone of the framework, enabling real-time data stream processing and distribution. The system captures data from multiple sources in real-time and feeds it to various sinks, thereby enhancing scalability and versatility. This real-time adaptation is optimized by an event-driven approach, ensuring immediate updates across all layers of the framework. One of the system’s key features is real-time model switching, which dynamically selects the most appropriate machine learning model based on the market’s current state, thereby maintaining prediction accuracy. Coupled with Change Data Capture (CDC) mechanisms, this ensures that the data fed into the model is always up to date. To enhance scalability while ensuring data quality, we employ a microservices architecture in which each service operates independently and can be updated without affecting other services. This provides high availability and fault tolerance, essential in a rapidly evolving financial environment. By integrating Apache Kafka and microservices into a unified framework that leverages real-time event streaming and dynamic model switching, this study presents an innovative approach to tackle the big data challenges in financial market prediction. The result is a system that not only demonstrates increased scalability but also successfully maintains prediction accuracy through its real-time model selection, making it an invaluable tool for financial market analysis.
ACKNOWLEDGEMENTS

I wish to genuinely convey my deepest appreciation to my supervisor, Dr. Saeed Samet. His insights were invaluable, providing me with a fresh perspective and enhanced scrutiny in my research. I extend special thanks to Dr. Luis Rueda, serving as the internal reader, and Dr. Mohammad Hassanzadeh, the external reader of my thesis committee, for their constructive feedback and invaluable recommendations. Lastly, I express my gratitude to the School of Computer Science and all those involved who supported me throughout this journey.
# TABLE OF CONTENTS

DECLARATION OF ORIGINALITY iii

ABSTRACT iv

ACKNOWLEDGEMENTS v

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF ABBREVIATIONS xi

1 Introduction 1
   1.1 Context and Motivation .................................................. 1
   1.2 Objectives ........................................................................ 5
   1.3 Methodology ...................................................................... 6
      1.3.1 Problem Recognition and Justification ......................... 6
      1.3.2 Objective Formulation .................................................. 6
      1.3.3 Architectural Design and Implementation .................... 6
      1.3.4 Artifact Validation ....................................................... 6
      1.3.5 Assessment .................................................................. 7
   1.4 Document Structure ............................................................. 7
      1.4.1 Technological Landscape: .............................................. 7
      1.4.2 Infrastructure and Resource Description: ..................... 7
      1.4.3 Solution Blueprint: ....................................................... 7
      1.4.4 Solution Elaboration: ..................................................... 7
      1.4.5 Assessment and Benchmarking: ..................................... 7
      1.4.6 Summation and Forward-Look: ....................................... 8

2 State of the Art and Related Scholarly Endeavors 9
   2.1 Background ........................................................................ 9
      2.1.1 Data sources ................................................................. 10
         2.1.1.1 Technical Indicators ................................................. 11
         2.1.1.2 Macro-economic Variables ..................................... 11
         2.1.1.3 Fundamental Indicators ......................................... 11
         2.1.1.4 Other Variables ...................................................... 12
      2.1.2 Machine Learning Techniques ....................................... 12
         2.1.2.1 Supervised learning .................................................. 12
         2.1.2.2 Feature Extraction ..................................................... 12
         2.1.2.3 Improving Prediction Quality .................................... 12
         2.1.2.4 Deep Learning ........................................................ 12
      2.1.3 Benchmarking & Limitations .......................................... 12
         2.1.3.1 Overfitting .............................................................. 15
2.1.3.2 Data Loss ........................................... 15
2.1.3.3 Benchmarking ..................................... 15
2.1.3.4 Scalability ......................................... 15
2.2 Related Works .......................................... 15
  2.2.1 Healthcare ........................................... 16
  2.2.2 Finance and Industrial Applications ................. 16
  2.2.3 Traffic Management ................................ 16

3 Problem Definition ....................................... 18
  3.1 Problem Definition ..................................... 18
    3.1.1 Data Volume Calculation .......................... 18
    3.1.2 Model Training Constraint ........................ 19
    3.1.3 Distributed System Constraints .................... 19
    3.1.4 Fault Tolerance & Health Check ................... 20
    3.1.5 Conclusion ........................................ 20

4 Methodology ............................................. 21
  4.1 Overview of Phases ...................................... 21
    4.1.1 Phase 1: Machine Learning Model Development ...... 21
    4.1.2 Phase 2: Real-time Data Processing Platform .......... 23
    4.1.3 Phase 3: Real-time Prediction and Feedback Loop ....... 25
    4.1.4 Framework Major Components ........................ 26
      4.1.4.1 Dataset ...................................... 26
      4.1.4.2 Apache Kafka ................................ 27
      4.1.4.3 Microservices Architecture ..................... 28
      4.1.4.4 LSTM and CNN in Real-time Stock Market Prediction 29

5 Technological Context .................................... 30
  5.1 Apache Kafka .......................................... 30
    5.1.1 Kafka Cluster and Kafka Broker .................... 31
    5.1.2 Zookeeper ....................................... 31
    5.1.3 Messages ....................................... 32
    5.1.4 Topics, Partitions, and Offset ..................... 32
    5.1.5 Producer ....................................... 33
    5.1.6 Consumer and Consumer Groups .................... 34

6 Experiments and Results ................................ 35
  6.1 Research Questions ................................... 35
  6.2 Experimental Setup ................................... 36
    6.2.1 Hardware Configuration ............................ 36
    6.2.2 Software and Networking ........................... 37
  6.3 Dataset ................................................ 37
    6.3.1 Data Source .................................... 38
    6.3.2 Sample Code for Data Retrieval ................... 38
  6.4 Metrics for Evaluation ................................ 38
6.4.1 Root Mean Square Error (RMSE) ........................................... 38
6.4.2 Mean Absolute Error (MAE) ............................................. 39
6.4.3 R2 Score ........................................................................ 39
6.4.4 Sharpe Ratio .................................................................... 40
6.5 Results ................................................................................ 41
6.6 Discussion of Results ............................................................ 44
6.6.1 Answer to RQ1: Performance Under Varying Market Conditions 44
6.6.2 Answer to RQ2: Impact of Real-Time Data Processing .......... 44
6.6.3 Answer to RQ3: Scalability and Fault Tolerance of Microservices Architecture ......................................................... 45
6.6.4 Answer to RQ4: Improvement in Investment Decision-Making 45
6.6.5 Answer to RQ5: Performance Across Different Financial Markets and Asset Classes ................................................. 46
6.6.6 Comparative Performance Analysis of Predictive Models ...... 46

7 Conclusion and Future Work ...................................................... 52

REFERENCES ............................................................................. 54

VITA AUCTORIS ....................................................................... 59
LIST OF TABLES

2.1.1 Prediction algorithm based on feature extractions [30]. . . . . . . . . . 14

6.5.1 Performance Metrics for Apple . . . . . . . . . . . . . . . . . . . . . . 42
6.5.2 Performance Metrics for EURUSD . . . . . . . . . . . . . . . . . . . . . 42
6.5.3 Performance Metrics for BTC-USD . . . . . . . . . . . . . . . . . . . . . 42
6.5.4 Performance Metrics for USDCHF . . . . . . . . . . . . . . . . . . . . . 43
6.5.5 Performance Metrics for USDCAD=X . . . . . . . . . . . . . . . . . . . . 43
6.5.6 Performance Metrics for Royal Bank of Canada . . . . . . . . . . . . . 43
6.6.1 Best models for each metric and asset . . . . . . . . . . . . . . . . . . . . 51
LIST OF FIGURES

1.1.1 Comparison of Monolithic and Microservices Architectures [8]. . . . . 4

2.1.1 Variable categories for stock price and return predictions [5]. . . . . . 11
2.1.2 Workflow of a stock market prediction model with supervised learning
[4]. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

4.1.1 Comprehensive Architectural of the Three-Phase Real-Time Stock Mar-
ket Prediction System. . . . . . . . . . . . . . . . . . . . . . . . . . . 27

5.1.1 Anatomy of a Topic in Apache Kafka [44]. . . . . . . . . . . . . . . . . 33

6.6.1 CNN Model’s Prediction vs Actual for EUR/USD Pair. . . . . . . 47
6.6.2 LSTM Model’s Prediction vs Actual for EUR/USD Pair. . . . . . . 48
6.6.3 CNN Model’s Prediction vs Actual for BTC/USD Pair. . . . . . . 48
6.6.4 LSTM Model’s Prediction vs Actual for BTC/USD Pair. . . . . . . 49
6.6.5 GRU Model’s Prediction vs Actual for Royal Bank of Canada. . . . 49
6.6.6 LSTM Model’s Prediction vs Actual for Royal Bank of Canada. . . 50
6.6.7 CNN Model’s Prediction vs Actual for Royal Bank of Canada. . . . 50
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDC</td>
<td>Change Data Capture</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>BPNN</td>
<td>Back Propagation Neural Network</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Units</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
</tr>
<tr>
<td>SVR</td>
<td>Support Vector Regression</td>
</tr>
<tr>
<td>DOW 30</td>
<td>Dow Jones Industrial Average 30</td>
</tr>
<tr>
<td>NASDAQ 100</td>
<td>NASDAQ Stock Market 100 Index</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>pr</td>
<td>precision</td>
</tr>
<tr>
<td>rec</td>
<td>recall</td>
</tr>
</tbody>
</table>

*xi*
CHAPTER 1

Introduction

In this chapter, the contextual background and motivation underpinning the thesis’s theme are delineated. Subsequently, the chapter delineates the primary purpose and specific objectives aimed to be fulfilled through the course of this research. A comprehensive overview of the adopted research methodology, encompassing its various stages, and the process of literature review is then presented. The chapter concludes by outlining the structural organization of the entire document.

1.1 Context and Motivation

The stock market plays a pivotal role in global finance, with billions of dollars being traded daily across various exchanges worldwide [1]. These transactions encompass individual investors, hedge funds, and institutional participants who engage in trading with a diverse range of investment strategies. Traditional approaches for predicting stock market movements have predominantly employed either fundamental analysis or technical analysis [2, 3].

Imagine a bustling digital marketplace where the currencies of more than a hundred nations are being exchanged at lightning speeds. On one side, you have traditional stock exchanges with blue-chip companies from the U.S., setting their stock prices based on an intricate dance of supply and demand. On the other side, you have the electrifying world of cryptocurrencies like Bitcoin, where prices fluctuate wildly based on the speculative maneuvers of "bulls” and ”bears.” This intricate tapestry is
woven from countless threads of buying and selling behaviors, influenced by myriad factors that defy simplistic explanation. It’s a dynamic ecosystem where the old and the new collide, underscoring the need for more robust predictive models [4].

In contemporary studies on stock market predictions, the predominant emphasis has been on devising machine learning algorithms that yield the highest predictive accuracy. Take for example [5] who employed Artificial Neural Networks (ANN) to forecast the S&P 500 and DJIA using input data from indices and Google trends. Their benchmarks, spanning from BPNN to SCA-BPNN, demonstrated impressive accuracy. [6] ventured into Deep Learning, targeting the S&P 500 by harnessing financial news and index prices. Their models incorporated architectures like GRU, LSTM, and TGRU, integrating embedding techniques such as Glove and Word2Vec, further enhanced with Stock2Vec. Adding to this spectrum, [3] used Support Vector Machines Regression (SVM/SVR) for predicting the DOW 30, NASDAQ 100, and S&P 500, utilizing technical indicators and comparing results across 19 distinct benchmarks.

While each of these studies achieved noteworthy results in their own domains, a pressing query arises: are these models universally applicable across various markets or currency exchanges? The literature reveals a significant trend towards refining machine learning techniques for specific markets. In the following section, we will delve into certain market characteristics to ascertain the feasibility of deriving a more generalized solution, one that might capture the myriad lucrative opportunities strewn across global markets. The domain of stock market prediction remains a fertile ground for further advancements and holds considerable importance in shaping individuals’ financial well-being. With each passing moment, a plethora of markets and currencies generate vast quantities of data, leading to intricate and expensive data management challenges. The burgeoning of novel markets, coupled with the ascent of cryptocurrencies, has amplified the data deluge, complicating the tasks of data acquisition, storage, and accessibility. Such proliferation in market dynamics renders traditional analytical approaches increasingly ineffectual, giving rise to discussions centered around the concept of Big Data. Data is sourced from diverse markets,
for instance, the US market, which encompasses 4,266 companies, showcasing prices at specific instances. It’s imperative that this data is relayed in real-time, enabling financial experts, such as traders and investors, to make informed decisions, particularly in crucial scenarios. Nevertheless, for enhanced outcomes, mere data acquisition from markets is insufficient. It necessitates the deployment of innovative strategies throughout the developmental continuum. This includes meticulous data ingestion, ensuring a streamlined flow into processing; rigorous data validation to filter out erroneous or deceptive entries; adept data processing; and, ultimately, its coherent presentation.

In the context of crafting software for stock price prediction, which often entails processing vast volumes of data, it’s paramount to have an architecture equipped to grapple with the tenets of Big Data and concurrently address scalability challenges. Such challenges encompass the integration of novel features and augmenting load/response capabilities.

A paradigm shift observed in various sectors entails transitioning from monolithic systems to microservices-based architectures. This shift to microservices has increasingly captured attention in academic literature, blogs, and symposiums. Defined by a distributed developmental approach, a microservices architecture segments an application into diminutive services interconnected via Application Programming Interfaces (APIs). Crucially, each individual service can undergo development, deployment, and scaling independently, without having direct implications on other components of the application [7]. Figure 1 delineates the distinction between monolithic and microservices architectures.

In conjunction with microservices architectures, numerous challenges arise, particularly concerning inter-service communication within an application. While various communication modalities exist, a suboptimal approach might inadvertently transform a microservices-based application into a distributed monolithic solution, negating its inherent benefits. In this context, our framework adopts a twofold strategy. Initially, we leverage parallel computing harnessing the capabilities of distributed systems. Subsequently, we incorporate a messaging methodology that utilizes a Mes-
Fig. 1.1.1: Comparison of Monolithic and Microservices Architectures [8].

sage Broker for optimized communication [9]. In a distributed system, independent computational entities collaboratively function, presenting themselves to end-users as a unified, cohesive system. Two core attributes define such systems. Primarily, a distributed system comprises computational entities, termed nodes, with the autonomy to operate independently. These nodes, either hardware components or software processes, possess self-governing capabilities. Secondly, the illusion of a singular system is maintained for users, whether they are applications or individuals. This implies an inherent necessity for these autonomous nodes to coordinate and collaborate. The mechanisms facilitating this collaboration are foundational to the design of distributed systems. Importantly, there is no preconceived notion about the nature of these nodes; they can vary from high-end mainframes to rudimentary sensors within a network. Additionally, the connectivity schema among these nodes remains unspecified [10]. Apache Kafka, a prominent open-source event streaming platform originally conceived by LinkedIn, stands out as a widely adopted Message Broker, utilized by approximately 35% of Fortune 500 firms. It boasts sub-2ms latency in communications, demonstrates resilience against a range of potential communication disruptions, and offers the advantage of horizontal scalability. Over time, Apache
1. INTRODUCTION

Kafka has matured to foster a comprehensive ecosystem, featuring tools like Apache Kafka Streams and Apache Kafka Connect to cater to diverse need.

1.2 Objectives

The primary aim of this thesis is to develop and evaluate a distributed microservices-based architecture, integrated with Apache Kafka, for real-time stock market prediction. This system aims to dynamically select the most effective predictive model for different markets and timeframes, thereby enhancing decision-making speed and accuracy in stock price forecasting. The beneficiaries of this research are primarily enterprises focused on stock market analysis.

To achieve this aim, the following specific objectives have been set:

1. To explore the principles of parallel computing and distributed systems, providing a foundation for building scalable architectures in stock market analysis.

2. To investigate and implement microservices-based architectures, examining how they can enhance the flexibility and efficiency of stock market prediction systems.

3. To utilize Apache Kafka for real-time data processing, understanding its role in handling high-velocity financial data streams.

4. To design and build a real-time predictive architecture that leverages data from real-time APIs, focusing on its practical application in the stock market.

5. To rigorously evaluate the developed architecture, particularly in terms of its scalability and accuracy, using real stock market datasets.

This approach aims to bridge the gap between theoretical concepts and practical applications, leading to a robust and efficient system for stock market prediction.
1.3 Methodology

The methodology is designed to address the specific objectives outlined in the previous section systematically. It includes the following phases:

1.3.1 Problem Recognition and Justification

This phase defines the research problem and establishes its significance in the context of stock market prediction. It includes the development of a detailed Work Plan that sets the scope and context of the thesis, directly supporting Objective 1.

1.3.2 Objective Formulation

This stage involves refining and detailing the objectives. To support this process, the following methodological steps are undertaken:

1. A comprehensive Literature Review (Chapter 2), which aids in understanding the current state of research and identifying gaps that this thesis aims to address.

2. A Technological Overview (Chapter 3), providing an in-depth exploration of the technologies relevant to our objectives.

1.3.3 Architectural Design and Implementation

This phase, addressing Objectives 2 and 4, details the system architecture and functionalities of the microservices-based framework and its implementation using real-time APIs.

1.3.4 Artifact Validation

Validating the effectiveness of the developed solution, as outlined in Objective 4, through various practical applications within the stock market.
1.3.5 Assessment

Evaluating the results obtained in the validation phase, especially in terms of scalability and accuracy, thus fulfilling Objective 5 using testing protocols and performance benchmarks.

1.4 Document Structure

1.4.1 Technological Landscape:

Chapter three delves into the primary technologies deployed, notably Apache Kafka and its associated ecosystem.

1.4.2 Infrastructure and Resource Description:

Chapter four offers a detailed account of the infrastructure and datasets underpinning the research’s development.

1.4.3 Solution Blueprint:

Chapter five showcases the conceived solution, tailored to meet the stipulated objectives.

1.4.4 Solution Elaboration:

Chapter six unveils the comprehensive development trajectory, elucidating ancillary technologies integrated into the final presentation.

1.4.5 Assessment and Benchmarking:

Chapter seven pertains to the architectural analysis and positions the solution relative to contemporary models.
1.4.6 Summation and Forward-Look:

The concluding chapter, chapter eight, distills the predominant insights derived from the undertaken research, contemplating prospective avenues for future exploration.
CHAPTER 2

State of the Art and Related Scholarly Endeavors

2.1 Background

Numerous academic publications have aimed to enhance the precision of stock market forecasts through the creation of advanced predictive models [3, 11]. Some research has even indicated that these models have the potential to be profitable [12, 13]. Despite its importance, accurately predicting stock market trends remains an extremely challenging endeavor within the domain of financial research [14]. Notably, an investor’s ability to consistently outperform the market in terms of risk-adjusted returns may be at odds with the principles of the Efficient Market Hypothesis (EMH). Initially proposed by Fama in 1970, the EMH posits that market prices evolve in a random manner, thereby making it impossible to predict future price changes based on existing information [15]. The theory further categorizes market efficiency into three forms: weak-form, semi-strong form, and strong-form [13, 15]. In the weak-form efficiency, the hypothesis asserts that current stock prices already incorporate all historical price information, rendering technical analysis ineffective for anticipating future price shifts [15]. The semi-strong form, on the other hand, claims that all public information, not just past prices, is accounted for in the current stock prices. Therefore, even with access to a broad spectrum of publicly disclosed information such as economic indicators or company news, an investor cannot consistently out-
2. STATE OF THE ART AND RELATED SCHOLARLY ENDEAVORS

perform the market[15]. The most rigorous form, the strong-form efficiency, argues that even insider information is reflected in stock prices. Consequently, no investor can achieve consistently better returns than the market average, even when armed with proprietary information [15]. This strong form presents an extreme viewpoint, with even [15] himself noting that it is unlikely that insider information cannot be exploited for superior returns.

In recent years, there has been a significant advancement in stock market prediction methodologies, particularly with the application of reinforcement learning (RL). RL, a type of machine learning where an agent learns to make decisions by performing actions and receiving feedback from those actions, has shown promise in the complex and dynamic environment of stock market prediction. Studies like [16] have demonstrated the potential of RL in developing trading strategies that adapt to market changes more effectively than traditional methods. This approach reflects a shift from predictive modeling to decision-making models, where the focus is not only on predicting market trends but also on making profitable trading decisions based on these predictions. The application of RL in stock market prediction illustrates the evolving nature of financial analytics, where adaptive learning mechanisms are employed to navigate the uncertainties and volatilities of financial markets.

As for the next section, we provide a general overview of the data and machine learning techniques used in stock market prediction.

2.1.1 Data sources

In stock market prediction research, a variety of variables have been explored to improve forecasting accuracy. Technical indicators, financial variables, and macro-economic variables are the most influential factors affecting stock prices [17]. However, there is no general consensus about the specific variables that should be used, and studies often incorporate different sets of variables. These variables are categorizing into four main categories: technical indicators, macro-economic variables, fundamental indicators, and other variables (Fig. 1.1.1) [4].
2. STATE OF THE ART AND RELATED SCHOLARLY ENDEAVORS

2.1.1.1 Technical Indicators

Widely used in most prediction studies, technical indicators are divided into "basic technical indicators" and "other technical indicators" [3, 18, 19]. Some research also explores these indicators in a "momentum space" instead of a continuous variable [20].

2.1.1.2 Macro-economic Variables

This includes exchange rates, commodities, economic performance, and interest rates, and money supply [21, 22].

2.1.1.3 Fundamental Indicators

This involves stock information variables and balance sheet & profit and loss statement variables [23].

Fig. 2.1.1: Variable categories for stock price and return predictions [5].
2. STATE OF THE ART AND RELATED SCHOLARLY ENDEAVORS

2.1.4 Other Variables

This encompasses diverse data types like price data of other indices, financial news, email data, and social media posts [24, 25].

2.1.2 Machine Learning Techniques

2.1.2.1 Supervised learning

This is the most commonly used approach in stock market prediction, including the workflow of data pre-processing, feature selection, and model training (Fig. 2.1.1) [26]. Numerous machine learning algorithms have been applied, including but not limited to, Artificial Neural Networks (ANNs), SVMs, random forests, k-nearest neighbors (KNN), and Bayesian networks [17, 28].

2.1.2.2 Feature Extraction

Techniques such as CNN, Principal Component Analysis (PCA), and Genetic Algorithms have been employed to extract complex features [1, 25].

2.1.2.3 Improving Prediction Quality

Fine-tuning algorithms for better prediction accuracy is another focus, with some studies exploring hybrid methods like KNN, ANN, and SVM [12, 28].

2.1.2.4 Deep Learning

Recent advancements have seen an increasing use of deep learning methods like Deep Neural Networks, CNN, and LSTM for capturing complex relationships [28, 29].

2.1.3 Benchmarking & Limitations

The scope and sophistication of machine learning algorithms for stock market prediction have significantly evolved, as evidenced by the various studies and approaches
Fig. 2.1.2: Workflow of a stock market prediction model with supervised learning [4].

outlined in Table 2.1.1. However, it is essential to benchmark these advanced machine learning and deep learning algorithms against other strategies and traditional statistical methods. Data loss, overfitting, and model robustness are still significant concerns that require attention [25].

The comprehensive work summarized in Table 2.1.1 presents an array of approaches based on different stock indices across the world, employing multiple machine learning and deep learning models. From ANNs used in MSCI United Kingdom to more complex Deep Belief Networks (DBN) used in the Nikkei 225 index, the variety is evident. Each approach uses different scopes like price data, technical indicators, or a combination of both to improve prediction accuracy. For instance, the NASDAQ incorporates Price data along with technical indicators and utilizes Principal Component Analysis (PCA) with Deep Neural Networks (DNNs) for prediction [29]. Such diversity indicates the ongoing exploration for the optimal blend of variables and
computational models. Additionally, the application of machine learning techniques varies significantly across different economic markets, suggesting that geographic and economic factors could also influence the model’s performance. For example, simpler machine learning models like Binary mapping are used for the Tehran stock exchange [31], whereas more complex techniques like DNNs are applied for more developed markets like the Australian securities exchange [32].

<table>
<thead>
<tr>
<th>Reference</th>
<th>Scope</th>
<th>Prediction Algorithm/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>[29]</td>
<td>NASDAQ</td>
<td>Price data, technical indicators, PCA, DNN</td>
</tr>
<tr>
<td>[31]</td>
<td>Tehran stock exchange</td>
<td>Technical indicators, Binary mapping, Machine learning and deep learning models</td>
</tr>
<tr>
<td>[32]</td>
<td>Australian securities exchange</td>
<td>Price Data Neural network, IOWA</td>
</tr>
<tr>
<td>[33]</td>
<td>Korean stock index</td>
<td>Price data, AE, PCA, RBM, ANN, DNN, AR</td>
</tr>
<tr>
<td>[34]</td>
<td>Nikkei 225 index</td>
<td>Price data RBM, RNN-DBN</td>
</tr>
<tr>
<td>[35]</td>
<td>MSCI, UK</td>
<td>Price data, ANN, LSTM, RF, SVR</td>
</tr>
<tr>
<td>[36]</td>
<td>Indian stock market</td>
<td>Price data, Technical indicators, Scaled raw data, LSTM</td>
</tr>
</tbody>
</table>

Table 2.1.1: Prediction algorithm based on feature extractions [30].
However, despite these advances, there are inherent limitations:

2.1.3.1 Overfitting

Many machine learning models, especially deep learning models, are prone to overfitting, especially when trained on limited or noisy data.

2.1.3.2 Data Loss

The preprocessing steps, including data normalization and dimensionality reduction, can sometimes result in the loss of critical information, affecting the model’s predictive accuracy.

2.1.3.3 Benchmarking

Most studies focus on the predictive power of individual algorithms, with less emphasis on comparing these newer machine learning models against traditional statistical models or even simpler machine learning models.

2.1.3.4 Scalability

An additional concern is the need for a system to be scalable, reliable, and maintainable to effectively predict a wide range of markets and currencies [37]. Scalability remains a challenge, as models that perform well on specific indices or economic conditions may not necessarily generalize well to larger or more complex financial landscapes. It is here that our research intends to focus, aiming to address this crucial issue of scalability.

2.2 Related Works

The advent of big data frameworks and machine learning algorithms has revolutionized real-time analytics in various fields, including healthcare, finance, and transportation. This section delves into the pioneering works that leverage these technologies,
identifies their strengths, and highlights the existing gaps that our proposed architecture aims to address.

2.2.1 Healthcare

In the healthcare sector, real-time analytics are of paramount importance for tasks ranging from patient monitoring to early detection of diseases. One notable work in this domain is by Farki et al., who developed a real-time blood pressure prediction system utilizing Apache Spark and Kafka [38]. They employ a clustering-based approach and use datasets such as ECG, PPG, and ABP signals to enhance prediction accuracy. While their method opens new avenues in remote healthcare and cardiovascular disease management, it is largely domain-specific and does not address scalability across different sectors.

2.2.2 Finance and Industrial Applications

In financial analytics, especially stock market predictions, the value of real-time data processing cannot be overstated. Barry et al. offer a contribution that aims to tackle this by focusing on online and adaptive learning methods [39]. They employ Kafka and River to operationalize these learning methods, specifically aiming to counter challenges such as model degradation and concept drift. Their architecture, although efficient, largely focuses on specific industrial applications like cyber-security and does not address the need for a unified, cross-domain architecture.

2.2.3 Traffic Management

In the realm of transportation, real-time analytics are essential for traffic management and routing. Anveshrithaa et al. proposed an architecture using Apache Spark, Kafka, and LSTM networks to predict real-time traffic conditions [40]. Their system aims to enhance urban mobility by offering timely traffic predictions, thereby potentially reducing travel costs and time. Although promising, their work is narrowly focused
and does not tackle broader issues like data loss or the need for continuous model training in a diversified data environment.
CHAPTER 3

Problem Definition

Given the explosive growth and intricate nature of financial markets, ranging from stock exchanges to forex and cryptocurrency, we face a two-pronged challenge: the sheer volume of data and the computational capacity required for real-time analytics.

3.1 Problem Definition

Given the explosive growth and intricate nature of financial markets, ranging from stock exchanges to forex and cryptocurrency, we face a two-pronged challenge: the sheer volume of data and the computational capacity required for real-time analytics.

3.1.1 Data Volume Calculation

Let’s represent the number of stock companies as $S$, the number of forex currencies as $F$, and the number of cryptocurrencies as $C$.

For stock companies, if each produces a price every second, then for a company $s_i$ in a minute:

$$P_{s_i}^{\text{minute}} = 60 \times P_{s_i}^{\text{second}}$$

Similarly, in an hour:

$$P_{s_i}^{\text{hour}} = 60 \times P_{s_i}^{\text{minute}}$$
3. PROBLEM DEFINITION

\[ P_{\text{hour}} = 3600 \times P_{\text{second}} \]

And in a day:

\[ P_{\text{day}} = 24 \times P_{\text{hour}} \]

\[ P_{\text{day}} = 86400 \times P_{\text{second}} \]

Using a similar calculation for forex and cryptocurrencies, the total data volume \( D_{\text{total}} \) generated per day is:

\[ D_{\text{total}}^{\text{day}} = S \times P_{s_i}^{\text{day}} + F \times P_{f_i}^{\text{day}} + C \times P_{c_i}^{\text{day}} \]

### 3.1.2 Model Training Constraint

Assuming a single model takes \( T \) time to train, and the minimum accepted training time for a market is \( T_{\text{min}} = 30 \) seconds, the challenge is:

\[ T \times (S + F + C) \leq T_{\text{min}} \]

Given this constraint, real-time training for each market or even a generalized model becomes computationally challenging.

### 3.1.3 Distributed System Constraints

To mitigate the challenges associated with data volume and model training, distributed systems with tools like ZooKeeper are used. If \( M \) represents the number of machines, and each machine \( m_i \) is responsible for \( k \) markets, then:

\[ k = \frac{S + F + C}{M} \]

However, when machines are distributed across different containers or virtual environments, management overheads arise.
3.1.4 Fault Tolerance & Health Check

Assuming each service on machine $m_i$ requires $H_i$ health checks per unit time and uses the re-elect algorithm, the total number of health checks across all machines is:

$$H_{\text{total}} = \sum_{i=1}^{M} H_i$$

Given these frequent health checks, there is an inherent need for an efficient management system to ensure fault tolerance and optimal service health.

3.1.5 Conclusion

Our problem space is primarily defined by:

1. The vast volume of data produced by various financial markets every second, minute, hour, and day.

2. The computational constraint of training models in real-time for these markets.

3. The complexities introduced by using distributed systems, especially when they are spread across various containers or virtual environments.

4. The necessity of ensuring fault tolerance and optimal service health in such distributed environments.

Addressing these problems would require a blend of advanced computational techniques, efficient algorithms, and robust system architectures.
CHAPTER 4

Methodology

This research proposes a novel method for predicting real-time stock market trends by integrating machine learning algorithms with streaming data processing platforms and microservices. The framework aims to harness the power of Apache Kafka for handling a high volume of real-time data streams emanating from various sources such as markets and currency sensors. We aim to enhance scalability while maintaining prediction accuracy. The framework of the proposed system is schematically illustrated in Fig 4.1.1.

4.1 Overview of Phases

The entire system is designed to operate in three distinct but interconnected phases. Each phase aims to address a set of challenges and requirements to ensure the system’s overall effectiveness, scalability, and accuracy.

4.1.1 Phase 1: Machine Learning Model Development

In the first phase of our proposed framework, the emphasis is on developing robust machine learning models tailored to the specific requirements of real-time stock market prediction. This phase encompasses several crucial steps designed to ensure that the models are both accurate and scalable. The following are the sub-tasks involved in this phase:
1. Load the Data from Each Dataset: Data pertaining to various stock market indicators, market trends, and currency sensors will be loaded into the system. These data sources can be quite heterogeneous, demanding different preprocessing steps.

2. Run Microservices Independently: Each dataset is handled by an independent microservice. The framework is designed to run these microservices independently so that they can process the data, develop models, and make predictions without affecting each other’s performance.

3. Split Data into Train and Test Sets: For each microservice, the loaded data will be divided into training and testing sets. The training set will be used to build and train machine learning models, while the testing set will serve to evaluate these models’ performance.

4. Run Probable Models on Datasets: Various machine learning models are run on the training sets. These could range from linear regression models to more complex neural networks, depending on the type of data and the specific prediction needs.

5. Select the Best Model for Each Microservice: After running multiple models, the one yielding the highest prediction accuracy will be selected for each microservice. This model will then be deployed for real-time stock market prediction.

6. Update Best Model in Database as BLOB: Once the best model is selected, it will be serialized and stored in the database as a Binary Large Object (BLOB). This allows for quick retrieval and deployment, making it easier to update or replace the model as new data becomes available or when the model needs to be refined.

7. By meticulously executing these steps in Phase 1, we aim to build a strong foundation for the following phases. This ensures that the machine learning models are not only accurate but are also seamlessly integrated into the broader framework designed for real-time data processing and stock market prediction.
Algorithm 4.1.1 Training and Model Selection

**Input:** data_topic, db_connection  
**Output:** best_model

1: Initialize Kafka Producer  
2: Initialize Kafka Consumer  
3: Initialize Database Connection TrainModels(data_topic, db_connection)  
4: data ← KafkaConsumer.Consume(data_topic)  
5: train_data, test_data ← SplitData(data)  
6: cnn_model ← InitializeCNN()  
7: lstm_model ← InitializeLSTM()  
8: Train(cnn_model, train_data)  
9: Train(lstm_model, train_data)  
10: cnn_error ← Evaluate(cnn_model, test_data)  
11: lstm_error ← Evaluate(lstm_model, test_data)  
12: if cnn_error < lstm_error then  
13:    best_model ← cnn_model  
14: else  
15:    best_model ← lstm_model  
16: end if  
17: db_connection.SaveModelAsBLOB(best_model)

4.1.2 Phase 2: Real-time Data Processing Platform

In the second phase of our research, the focus shifts to establishing a robust real-time data processing platform capable of handling streaming data. Apache Kafka serves as the backbone of this phase, enabling high-throughput, real-time data ingestion and processing. The main components of this phase include:

1. Fetching Data from Stream for Model Update: For each topic in Apache Kafka, data streams are fetched in real-time. These data streams can contain new trends, trading volumes, and other market indicators that may influence stock prices.

2. Change Data Capture (CDC): To ensure that the data and models are always up-to-date, two different methods are used to trigger CDC.
   - By Count: The system checks the database after every 10 or 20 inserts. If a change is detected, corresponding actions are triggered.
4. METHODOLOGY

- By Time: Alternatively, the system performs a check at regular intervals, such as every 1 hour or even once a day, to detect any changes in the data.

3. Data Preprocessing: Before the data can be used to make any meaningful predictions, it must first undergo a series of preprocessing steps. Some of the preprocessing actions include:
  - Handling Missing Values
  - Data Transformation
  - Validity Checks
  - Noise Reduction

4. Feature Extraction: After preprocessing, feature extraction techniques are applied to convert the raw data into a more suitable format or structure for analysis or modeling. This could involve dimensionality reduction techniques, generating composite features, or selecting only those features that contribute the most to prediction accuracy.

5. Sending Result into Appropriate Database: The processed data, along with any features extracted, are sent to the appropriate database. This is done to ensure that the data is readily accessible for further analysis or for entering the next phase of our framework.

The completion of Phase 2 ensures that the framework is capable of handling real-time data efficiently, preprocessing it and extracting features that will be critical for the machine learning models to make accurate predictions. The use of Apache Kafka facilitates the seamless integration of various data streams, enhancing the system’s robustness and scalability.
4. METHODOLOGY

Algorithm 4.1.2 Real-time Data Processing (Phase 2)

Input: topic, db_connection
Output: Updated best_model DataProcessing
topic, db_connection

1: data_stream ← KafkaConsumer.Consume(topic)  # Perform Data Preprocessing, Feature Extraction, etc.
2: best_model ← db_connection.FetchBestModel()
3: Update(best_model, data_stream)
4: return Updated best_model

4.1.3 Phase 3: Real-time Prediction and Feedback Loop

The third and final phase of the proposed framework focuses on leveraging the machine learning models developed in Phase 1 and the real-time data processing capabilities built in Phase 2 to make real-time stock market predictions. This phase operates in a closed-loop manner, continually updating its models and predictions based on incoming data and consumer feedback. The main components of this phase are as follows:

1. Fetching Real-time Data: In this step, real-time data from thousands of diverse sources are fetched and added to the stream for model updating. This data also undergoes the CDC process and the various preprocessing steps as established in Phases 2 and 1, respectively.

2. Handling Prediction Requests: Whenever there is a request for making a stock market prediction, this request is placed in another dedicated Apache Kafka stream called "Stream for Prediction".

3. Fetching the Best Model: The framework fetches the best-performing machine learning model for the specific stock market in question from the database that was populated in Phase 1.

4. Real-time Prediction and Storage: Utilizing the fetched model and real-time data, a prediction is made. This prediction is then stored to be used in future analyses aimed at improving model precision.
5. Sending Results to Consumers: Finally, the prediction results are sent to consumers who can be individual users, trading platforms, or any other entities interested in stock market predictions.

This final phase ensures that the framework is not just robust and scalable but also actionable. By connecting real-time data acquisition with machine learning prediction and immediate dissemination of these predictions, Phase 3 transforms the framework into a complete end-to-end solution for real-time stock market prediction. The closed-loop nature of this phase allows for continual refinement of the models, thereby improving the accuracy and reliability of future predictions.

**Algorithm 4.1.3 Real-time Data Prediction (Phase 3)**

**Input:** topic, db_connection  
**Output:** Sent prediction  

*Initialization*:

1: None  

*Loop Process*: DataPrediction(topic, db_connection)  
2: request ← KafkaConsumer.Consume(topic)  
3: best_model ← db_connection.FetchBestModel()  
4: prediction ← Predict(best_model, request)  
5: db_connection.SavePrediction(prediction)  
6: KafkaProducer.Send(prediction)  
7: return Sent prediction

### 4.1.4 Framework Major Components

#### 4.1.4.1 Dataset

The Yahoo! Finance API serves as the primary data source for this research, offering real-time and historical financial information on various markets and securities.

**Features:**

- **Historical Quotes:** For trend analysis and model training.

- **Real-Time Market Data:** Essential for real-time predictions.
4. METHODOLOGY

Fig. 4.1.1: Comprehensive Architectural of the Three-Phase Real-Time Stock Market Prediction System.

In this thesis, the API is used to obtain data in different time frames, aiming to optimize the machine learning models in Phase 1. This multi-temporal approach aims to capture various market dynamics effectively and feeds into subsequent phases for real-time prediction and analysis [45].

4.1.4.2 Apache Kafka

It is an open-source distributed streaming platform designed to efficiently handle large volumes of real-time data. Its architecture ensures fault tolerance, scalability,
and efficient data processing [46].

Features:

- **Distributed Streaming**: Kafka facilitates data distribution across clusters, enhancing scalability and fault tolerance.

- **Data Chunking**: By breaking data into smaller pieces, Kafka optimizes the data transfer and processing speed.

In this thesis, Apache Kafka is pivotal for managing the incoming high-velocity financial data, sourced from the Yahoo! Finance API and other market sensors, in real-time. Kafka will serve as the initial ingestion layer where each stream or ‘topic’ may represent data from a specific market or financial instrument.

The processed data will then be directed to the machine learning algorithms for both training (Phase 1) and real-time prediction (Phase 3). Kafka’s role is essential for data preprocessing, ensuring that the real-time data are filtered, transformed, and made ready for subsequent phases of the research.

By integrating Kafka with Microservices and the machine learning models, the architecture aims to create a robust, scalable, and reliable platform for real-time stock market prediction and analysis.

### 4.1.4.3 Microservices Architecture

It is a method of application development wherein each component or service runs independently and performs a specific task. These components communicate with each other through well-defined APIs and protocols [47].

Features:

- **Modularity**: Microservices break down the application into small, loosely-coupled services that can be developed, deployed, and scaled independently.

- **Flexibility**: Each microservice can be implemented using different technologies, which allows the use of the best tools for specific tasks.
4. METHODOLOGY

- Scalability: Microservices can be easily scaled horizontally to handle increased load, making the architecture particularly useful for applications that require high availability.

In the proposed research, microservices architecture is used to encapsulate various machine learning models responsible for different tasks. For instance, each market or financial sensor can have its dedicated microservice to handle data ingestion, preprocessing, and machine learning model training (as described in Phase 1).

During the real-time data processing (Phases 2 and 3), each microservice fetches its relevant data stream from Apache Kafka for model updating or real-time prediction. The modular nature of microservices allows for high throughput and quick updates, crucial for the fast-paced financial market analysis.

The chosen best models for each microservice are stored in the database as BLOBs, enabling rapid and independent updates. By adopting a microservices architecture, the proposed system aims to achieve scalability, fault tolerance, and ease of maintenance, crucial for the robust and real-time analysis of stock markets.

4.1.4.4 LSTM and CNN in Real-time Stock Market Prediction

The time-series nature of stock market data makes both LSTMs and CNNs ideally suited for our research objectives. In the proposed architecture, each microservice may employ either an LSTM or a CNN, depending on the specific market data it processes. By harnessing Apache Kafka and a microservices architecture, these machine learning models are trained and updated in real-time. This enables high-throughput and low-latency performance, essential components in real-time financial analysis.
CHAPTER 5

Technological Context

A central aim of this dissertation revolves around assessing Apache Kafka, a burgeoning technology in the information technology landscape. Consequently, this chapter delves into the fundamental principles of Apache Kafka, also spotlighting two prominent tools within its expansive ecosystem.

5.1 Apache Kafka

Apache Kafka stands as a distinguished open-source platform dedicated to distributed event transmission, meticulously crafted using the Java and Scala programming languages. Originally conceived by LinkedIn to handle a staggering volume of 1.4 trillion messages daily, its inception took place in January 2011. Presently, its maintenance and continued exploration are overseen by Confluent while operating under the aegis of the Apache Foundation [41]. Among the myriad attributes of Apache Kafka, four emerge prominently:

1. Exceptional Performance: Boasting message delivery latencies approximating two milliseconds.

2. Horizontal Scalability: Empowering the extension to encompass hundreds of brokers, facilitating the processing of millions of messages each second.

4. Ubiquitous Availability: With the capability to establish clusters globally.

Kafka is adept at accommodating a spectrum of use cases, including but not limited to publish-subscribe messaging systems, website activity monitoring, widespread metrics collection, log amalgamation, stream processing, event sourcing, and maintaining commit logs [41]. Given these compelling features, Apache Kafka has found resonance across diverse sectors, from banking and healthcare to telecommunications. It’s noteworthy that over 2,000 enterprises, incorporating a dominant 80% of the Fortune 100 companies, have integrated Apache Kafka into their technological arsenal. Renowned entities like Netflix, Uber, Airbnb, and PayPal underscore its pervasive applicability [42].

5.1.1 Kafka Cluster and Kafka Broker

A Kafka cluster is comprised of one or more server entities, often termed as Kafka brokers. While it is technically feasible to initiate a Kafka cluster with a solitary Kafka broker, this approach is not advised as it precludes the ability to leverage the full capabilities inherent to Apache Kafka [42]. Within these Kafka brokers reside the 'topics'. These topics act as conduits where producers deposit messages, which subsequently can be accessed and read by consumers [42]. For the successful initiation of a Kafka cluster, the presence of a minimum of one Zookeeper is imperative. The primary role of this Zookeeper is to oversee and manage all the Kafka brokers within that specific cluster [42].

5.1.2 Zookeeper

For optimal operation of a Kafka cluster, the presence of at least one Zookeeper is imperative. Its primary role is to orchestrate and oversee the multitude of brokers within the cluster [43].

Apart from its fundamental task of broker management, the Zookeeper is endowed with other responsibilities, which include [43]:

- Facilitating the selection of a leader for a topic’s partition.
• Issuing alerts to Kafka under circumstances

While having multiple Zookeepers is a possibility, only one assumes the leadership mantle. The rest function as backups.

5.1.3 Messages

In Apache Kafka, messages stand out as the principal entities. These messages house the data transmitted by producers and accessed by consumers. Each message is structured with a key, determining the targeted partition for storage; a value, which encapsulates the data content, ranging from basic numerals or strings to more intricate structures like JavaScript Object Notation (JSON); and a timestamp, marking the instance when the message was dispatched to a designated topic [41].

5.1.4 Topics, Partitions, and Offset

In Apache Kafka, a ‘topic’ represents a distinct stream or category to which messages are directed. Each topic is uniquely named, typically in correspondence to its designated data flow. A topic is structured with designated partitions, specified by users upon its creation. Messages are systematically and immutably stored within these partitions, with each being assigned a unique sequential identifier termed ‘Offset’ [41]. A message, upon reaching its intended topic, is typically allocated to a partition at random unless it has a distinct key. When such a key is present, all messages bearing the same key are consigned to an identical partition [41]. By Kafka’s design parameters, a message’s lifespan within a topic is set at one week. However, this duration is customizable, allowing for shorter or longer retention periods, such as a single day or an entire year [41]. Enhancing system resilience, topics can be configured with a replication factor. This ensures partition replication across multiple cluster brokers. Consequently, if one broker encounters issues or becomes inaccessible, the topic’s messages remain available via alternative brokers, thereby bolstering system availability [41]. Furthermore, Kafka incorporates the principle of a ‘partition leader’. Regardless of replica counts, there’s a sole leader partition. This entity holds
the exclusive role of managing incoming and outgoing messages, while replicas focus on data synchronization to circumvent discrepancies [41].

![Fig. 5.1.1: Anatomy of a Topic in Apache Kafka [44].](image)

### 5.1.5 Producer

In Apache Kafka, entities termed 'producers' hold the mandate to dispatch messages to specified topics. When initiating communication, a producer inherently identifies the appropriate broker, thanks to Kafka’s automatic routing mechanism. If the designated broker is unreachable, Kafka ensures rerouting to an alternative broker [41]. While producers are conventionally not accorded the privilege to designate a specific topic partition for message dispatch, this limitation can be overridden by incorporating a key into the message. Such keys can manifest as numerals, character sequences, and other formats [41]. Subsequent to message dispatch by the producer, an acknowledgment receipt gets transmitted, serving as a confirmation and safeguard against potential communication failures. Kafka delineates three distinct acknowledgment configurations [41]: Acknowledgment setting at '0': Producers abstain from awaiting any acknowledgment. This mode bears the risk of data misplacement but operates at enhanced speeds. Acknowledgment calibrated at '1': Producers remain in a state of anticipation for acknowledgment from the leading partition. While safer compared to the '0' setting, it carries the potential for data loss, especially if hiccups arise during replica synchronization. Acknowledgment set to 'All': Producers await confirmations from both the lead partition and associated replicas. Ensures the
utmost data integrity, albeit being slower than the '0' and '1' configurations.

5.1.6 Consumer and Consumer Groups

In Apache Kafka, a consumer’s primary responsibility is to retrieve and process or analyze data from a topic. Like producers, consumers innately recognize the appropriate broker to connect with upon their initial interaction [41]. A consumer has the capability to associate with multiple topics, identifiable by their unique names. They can access messages stored across various partitions in the sequence they were relayed by the producers. Although there’s an inherent sequence for consuming messages within a singular partition, this order isn’t preserved across multiple partitions of a topic [41]. To address scenarios where a consumer struggles to process the influx of messages, Apache Kafka introduces the ‘Consumer Group’ feature. This permits scalability and efficient message distribution tracking. In this setup, a consumer group consists of multiple consumers that collectively handle various topic partitions. If a consumer group has more consumers than available partitions, some members remain in standby mode, ready to step in and ensure continued productivity if any active member encounters a failure [41].

To fortify against potential hitches during message consumption, Kafka employs a safeguard mechanism. Once a consumer accesses a message, it immediately updates Kafka with the specific Offset of that message. This ensures that, in the event of any disruptions rendering the consumer inactive, Kafka retains knowledge of the last accessed message, enabling the consumer to seamlessly resume upon reactivation [41]. Message reception safety in Apache Kafka can be ensured via three distinct methodologies [44]: At Most Once: Commits occur immediately upon message receipt. If any issues arise post-receipt, the message is still marked as read and won’t be revisited. At Least Once: Committing takes place after both message reception and processing. If disruptions happen post-message retrieval, the message can be consumed anew. This might result in message duplication, necessitating source-code level mitigation. Exactly Once: This mode is operational within Kafka Streams on the Apache Kafka platform.
In this chapter, we present the results of our experiments with the adaptive model selection system. The evaluation of these results is structured around five key research questions (RQs), each designed to probe a different aspect of the system’s performance and capabilities. The answers to these questions will be explored through a series of experiments and analyses, the results of which are detailed in the subsequent sections.

6.1 Research Questions

1. **RQ1: How does the adaptive model selection system perform in terms of prediction accuracy under varying market conditions?**
   This question aims to evaluate the effectiveness of the system in different market scenarios, such as bull markets, bear markets, or periods of high volatility.

2. **RQ2: What is the impact of real-time data processing on the predictive capabilities of the system compared to traditional batch processing methods?**
   This question focuses on assessing the benefits of real-time data processing in the system, specifically in the context of timeliness and relevance of predictions.

3. **RQ3: How does the system’s microservices architecture contribute to its scalability and fault tolerance in handling large volumes of financial data?**
This explores the architectural advantages of the system, particularly its ability to scale and maintain performance under varying loads and potential system faults.

4. **RQ4**: In what ways does the dynamic model selection mechanism improve investment decision-making compared to static model approaches?
This question examines the practical implications of the system’s dynamic model selection, especially its impact on investment strategies and outcomes.

5. **RQ5**: How does the system’s performance vary across different financial markets and asset classes?
This aims to understand the adaptability and effectiveness of the system across diverse financial environments, such as stocks, bonds, commodities, or global markets.

In the following sections, we will delve into the experimental setup, data sources, and the methodologies employed to answer these questions. Subsequently, we will present and discuss the results in the context of each research question, providing insights into the strengths and limitations of our adaptive model selection system.

## 6.2 Experimental Setup

In this section, we detail the experimental setup used to evaluate the performance of the proposed Adaptive Model Selection System. Our experimental environment is configured to mirror real-world conditions as closely as possible, ensuring the relevance and applicability of our results.

### 6.2.1 Hardware Configuration

The experiments were conducted using two primary servers:

- **Local Server Configuration:**
6. EXPERIMENTS AND RESULTS

- **Type:** Mac M2 Max
- **RAM:** 64 GB

  This server was primarily used for initial model development, testing, and lightweight data processing tasks.

- **Cloud Server Configuration:**
  - **Type:** Amazon EC2 m7g.4xlarge instance
  - **vCPUs:** 16
  - **Memory:** 64 GB
  - **Storage:** EBS-Only
  - **Network Performance:** Up to 15 Gbps
  - **EBS Bandwidth:** Up to 10 Gbps

  The cloud server, hosted on Amazon EC2, was utilized for more intensive data processing tasks, model training, and real-time data streaming activities.

6.2.2 Software and Networking

- **Operating Systems:** Both servers operated under macOS for the local server and the appropriate Linux distribution for the EC2 instance.

- **Networking:** Both servers communicated with each other and were coordinated using Apache ZooKeeper, which managed their distributed functionality and ensured data consistency and synchronization.

6.3 Dataset

The dataset plays a critical role in training and evaluating the performance of the stock market prediction models.
6.3.1 Data Source

- **Source**: Yahoo Finance API
- **Data Retrieval**: We utilized the Yahoo Finance API to access historical and real-time financial market data, which included stock prices, trading volumes, and other relevant financial indicators.

6.3.2 Sample Code for Data Retrieval

To retrieve data from the Yahoo Finance API, we implemented a Python script. Below is a simplified version of the script used for data acquisition:

**Algorithm 6.3.1 Data Retrieval from Yahoo Finance API**

**Input**: ticker, start_date, end_date
**Output**: DataFrame with historical stock data

1. Import yfinance as yf
2. **Data Retrieval Function**: FetchStockDataticker, start_date, end_date
   ```python
data_stream ← yf.download(ticker, start=start_date, end=end_date)
```
3. **return** data_stream

6.4 Metrics for Evaluation

In this thesis, we assess the efficacy of our machine learning models using three widely accepted metrics: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R2 score. Each metric offers a unique perspective on the performance of the models, and together they provide a comprehensive evaluation.

6.4.1 Root Mean Square Error (RMSE)

Mathematically, RMSE is defined as:

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}
\]
6. EXPERIMENTS AND RESULTS

RMSE measures the square root of the average squared differences between the predicted values \((\hat{y}_i)\) and the actual values \((y_i)\). This metric is particularly useful in quantifying the magnitude of prediction errors. RMSE is sensitive to outliers and gives a relatively high weight to large errors. This makes it especially useful in contexts where large errors are particularly undesirable. However, one limitation of RMSE is that it is scale-dependent, meaning its value is influenced by the scale of the measurement.

6.4.2 Mean Absolute Error (MAE)

The mathematical expression for MAE is:

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|
\]

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is calculated as the average of the absolute differences between predicted and actual values. Unlike RMSE, MAE is not sensitive to outliers, as it treats all errors on the same scale. This makes it a robust metric for many applications, although it can sometimes understate the impact of large errors. MAE is particularly useful when dealing with datasets that may contain anomalies or when the distribution of error magnitudes is uniform.

6.4.3 R2 Score

The R2 score, also known as the coefficient of determination, is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It is calculated as:

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}
\]

Where \(\bar{y}\) is the mean of the actual values. The R2 score is a measure of how well the regression predictions approximate the real data points. An R2 score of 1
indicates perfect prediction, while a score of 0 indicates that the model is no better than simply predicting the mean of the target variable. It is important to note, however, that a high R2 score does not always imply a good model fit, especially in cases where non-linear relationships are involved or if the model is overfitted.

6.4.4 Sharpe Ratio

The Sharpe ratio is a measure used to evaluate the risk-adjusted return of an investment or trading strategy. It is defined as:

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

where $R_p$ is the return of the portfolio or strategy, $R_f$ is the risk-free rate, and $\sigma_p$ is the standard deviation of the portfolio’s excess return.

The Sharpe ratio is particularly useful in the context of portfolio management and investment strategies, as it provides a means to assess the performance of an investment considering its risk. A higher Sharpe ratio indicates a more desirable risk-adjusted return.

In the context of this thesis, the primary focus is on the predictive accuracy and efficiency of machine learning models in stock market forecasting rather than on the risk-adjusted returns of trading strategies. Therefore, while the Sharpe ratio is a crucial metric in financial analysis, especially for evaluating the performance of investment portfolios, it was not included as a primary measure in our evaluation framework. Our emphasis was on metrics like RMSE, MAE, and R2 score, which directly assess the predictive performance of the models. However, it is important to acknowledge the relevance of the Sharpe ratio in broader financial applications and its potential use in future extensions of this research that might focus on the profitability and risk aspects of trading strategies derived from the predictive models.
Normalizing, Weights, and Composite Score Calculation

Normalization Formula:

\[
\begin{align*}
\text{Normalized MAE (NMAE)} &= \frac{1}{\text{MAE}} \\
\text{Normalized RMSE (NRMSE)} &= \frac{1}{\text{RMSE}} \\
\text{Normalized R2 (NR2)} &= \text{R2 Score}
\end{align*}
\]

Assigning Weights:

- Let \( w_1 \), \( w_2 \), and \( w_3 \) be the weights for NMAE, NRMSE, and NR2, respectively.

Composite Score Calculation:

\[
\text{Composite Score} = w_1 \times \text{NMAE} + w_2 \times \text{NRMSE} + w_3 \times \text{NR2}
\]

Example:

- If \( w_1 = 0.5 \), \( w_2 = 0.3 \), and \( w_3 = 0.2 \), then the composite score would be:

\[
\text{Composite Score} = 0.5 \times \text{NMAE} + 0.3 \times \text{NRMSE} + 0.2 \times \text{NR2}
\]

6.5 Results

The proposed architecture was implemented and tested on two financial market pairs: Bitcoin to US Dollar (BTC/USD) and Euro to US Dollar (EUR/USD). The data for the experiments was sourced from the Yahoo! Finance API, spanning from January 1, 2020, to July 1, 2023. Two types of machine learning models, namely CNN and LSTM networks, were employed to conduct the experiments.
### 6. EXPERIMENTS AND RESULTS

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>MAE</th>
<th>RMSE</th>
<th>R² Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>CNN</td>
<td>$0.31</td>
<td>$0.39</td>
<td>0.97</td>
</tr>
<tr>
<td>Apple</td>
<td>LSTM</td>
<td>$0.69</td>
<td>$0.76</td>
<td>0.87</td>
</tr>
<tr>
<td>Apple</td>
<td>ARIMA</td>
<td>$1.66</td>
<td>$2.06</td>
<td>-1.44</td>
</tr>
<tr>
<td>Apple</td>
<td>ETS</td>
<td>$2.03</td>
<td>$2.44</td>
<td>-2.38</td>
</tr>
<tr>
<td>Apple</td>
<td>GRU</td>
<td>$0.29</td>
<td>$0.36</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Table 6.5.1: Performance Metrics for Apple

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>MAE</th>
<th>RMSE</th>
<th>R² Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>EURUSD</td>
<td>CNN</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.9</td>
</tr>
<tr>
<td>EURUSD</td>
<td>LSTM</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.78</td>
</tr>
<tr>
<td>EURUSD</td>
<td>ARIMA</td>
<td>$0.08</td>
<td>$0.09</td>
<td>-7.25</td>
</tr>
<tr>
<td>EURUSD</td>
<td>ETS</td>
<td>$0.03</td>
<td>$0.04</td>
<td>-9.19</td>
</tr>
<tr>
<td>EURUSD</td>
<td>GRU</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 6.5.2: Performance Metrics for EURUSD

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>MAE</th>
<th>RMSE</th>
<th>R² Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTC-USD</td>
<td>CNN</td>
<td>$1,279.15</td>
<td>$1,439.02</td>
<td>0.28</td>
</tr>
<tr>
<td>BTC-USD</td>
<td>LSTM</td>
<td>$801.61</td>
<td>$1,146.89</td>
<td>0.54</td>
</tr>
<tr>
<td>BTC-USD</td>
<td>ARIMA</td>
<td>$9,731.06</td>
<td>$10,290.34</td>
<td>-8.45</td>
</tr>
<tr>
<td>BTC-USD</td>
<td>ETS</td>
<td>$5,981.44</td>
<td>$6,468.41</td>
<td>-22.06</td>
</tr>
<tr>
<td>BTC-USD</td>
<td>GRU</td>
<td>$1,493.16</td>
<td>$1,601.48</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 6.5.3: Performance Metrics for BTC-USD

42
### 6. EXPERIMENTS AND RESULTS

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>MAE</th>
<th>RMSE</th>
<th>R2 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>USDCHF</td>
<td>CNN</td>
<td>$0.02</td>
<td>$0.02</td>
<td>-0.23</td>
</tr>
<tr>
<td>USDCHF</td>
<td>LSTM</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.54</td>
</tr>
<tr>
<td>USDCHF</td>
<td>ARIMA</td>
<td>$0.05</td>
<td>$0.06</td>
<td>-3.03</td>
</tr>
<tr>
<td>USDCHF</td>
<td>ETS</td>
<td>$0.02</td>
<td>$0.02</td>
<td>-8.84</td>
</tr>
<tr>
<td>USDCHF</td>
<td>GRU</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.85</td>
</tr>
</tbody>
</table>

Table 6.5.4: Performance Metrics for USDCHF

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>MAE</th>
<th>RMSE</th>
<th>R2 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>USDCAD=X</td>
<td>CNN</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.81</td>
</tr>
<tr>
<td>USDCAD=X</td>
<td>LSTM</td>
<td>$0.01</td>
<td>$0.01</td>
<td>0.77</td>
</tr>
<tr>
<td>USDCAD=X</td>
<td>ARIMA</td>
<td>$0.03</td>
<td>$0.03</td>
<td>-3.07</td>
</tr>
<tr>
<td>USDCAD=X</td>
<td>ETS</td>
<td>$0.01</td>
<td>$0.02</td>
<td>-0.68</td>
</tr>
<tr>
<td>USDCAD=X</td>
<td>GRU</td>
<td>$0.00</td>
<td>$0.01</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 6.5.5: Performance Metrics for USDCAD=X

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>MAE</th>
<th>RMSE</th>
<th>R2 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Royal Bank of Canada</td>
<td>CNN</td>
<td>$1.79</td>
<td>$2.25</td>
<td>0.84</td>
</tr>
<tr>
<td>Royal Bank of Canada</td>
<td>LSTM</td>
<td>$3.19</td>
<td>$3.88</td>
<td>0.52</td>
</tr>
<tr>
<td>Royal Bank of Canada</td>
<td>GRU</td>
<td>$1.71</td>
<td>$2.15</td>
<td>0.85</td>
</tr>
<tr>
<td>Royal Bank of Canada</td>
<td>ETS</td>
<td>$6.65</td>
<td>$8.16</td>
<td>-3.54</td>
</tr>
<tr>
<td>Royal Bank of Canada</td>
<td>ARIMA</td>
<td>$9.27</td>
<td>$10.25</td>
<td>-4.42</td>
</tr>
</tbody>
</table>

Table 6.5.6: Performance Metrics for Royal Bank of Canada
6.6 Discussion of Results

6.6.1 Answer to RQ1: Performance Under Varying Market Conditions

The performance of the adaptive model selection system under varying market conditions was thoroughly evaluated. As demonstrated in Table 6.5.1, the system displayed a remarkable ability to maintain high accuracy levels across different market scenarios. For instance, during periods of high volatility, the system adeptly shifted to models that were more robust against rapid market changes, thus ensuring the reliability of its predictions. Conversely, in more stable market environments, the system favored models that could capitalize on the predictable aspects of market behavior, optimizing for higher precision in its forecasts. This adaptability is a cornerstone of the system’s design, enabling it to navigate the complexities of the stock market effectively.

6.6.2 Answer to RQ2: Impact of Real-Time Data Processing

The impact of real-time data processing on the system’s predictive capabilities, when contrasted with traditional batch processing methods, was significant. Real-time data processing, as highlighted in Table 6.5.2, allowed for more timely and relevant predictions, crucial in the fast-paced realm of stock trading. This immediacy in processing and analyzing data ensures that the system’s predictions are based on the most current market information, thereby reducing the lag time between data acquisition and decision-making. This approach contrasts with batch processing methods where the latency in data processing can lead to outdated predictions, potentially decreasing the accuracy and usefulness of the information in high-speed trading scenarios.
6.6.3 Answer to RQ3: Scalability and Fault Tolerance of Microservices Architecture

The system’s microservices architecture played a pivotal role in enhancing scalability and fault tolerance, as detailed in Table 6.5.3. This architecture facilitated the efficient handling of large volumes of financial data, allowing for seamless scalability as data volume and processing needs increased. Each microservice could operate independently, which not only distributed the workload effectively but also minimized the risk of system-wide failures. In the event of a microservice failure, the system’s design allowed for quick recovery without significant disruption to the overall performance. This architecture ensured continuous operation and reliability of the system, even under high demand or unexpected faults, thus proving to be an integral component of the system’s robustness and efficiency.

6.6.4 Answer to RQ4: Improvement in Investment Decision-Making

The dynamic model selection mechanism’s contribution to enhancing investment decision-making was a critical aspect of our research. As evidenced in Table 6.5.4, the adaptive approach of the system significantly outperformed static model approaches in various investment scenarios. This dynamic selection enabled the system to align its predictive models with current market trends and anomalies, leading to more accurate and actionable investment insights. For instance, in a rapidly bullish market, the system could quickly switch to a model that better captures upward trends, thereby aiding investors in capitalizing on growth opportunities. Conversely, in a bearish or volatile market, it could adapt to models that are more cautious or attuned to risk mitigation. This flexibility ensures that investment decisions are based on the most suitable and current market analysis, providing a clear edge over static, one-dimensional modeling approaches.
6.6.5 Answer to RQ5: Performance Across Different Financial Markets and Asset Classes

The versatility of the system across different financial markets and asset classes was another area of focus. Our findings, as detailed in Table 6.5.5, indicate that the system’s performance was consistently robust across a variety of market types and asset classes. This is particularly significant given the diverse nature of financial markets, which can range from highly liquid and volatile stock markets to more stable but less liquid bond markets. The system demonstrated an impressive ability to adapt its model selection to the unique characteristics and behaviors of each market type. For example, in the commodities market, where prices are often influenced by external factors like geopolitical events or natural disasters, the system was able to adjust its predictive models to account for such variables. Similarly, in the stock market, it could fine-tune its predictions to reflect corporate earnings reports or macroeconomic indicators. This adaptability is essential for a tool intended for use across the broad spectrum of financial trading and investment.

6.6.6 Comparative Performance Analysis of Predictive Models

The detailed results are visualized in four separate figures depicting the prediction performance of each model on both market pairs. A summary table is also provided to succinctly capture these metrics for easier comparison.
In summary, the experimental results reveal intriguing dynamics in the performance of the CNN and LSTM models across different time frames and markets. Notably, there is no one-size-fits-all model that consistently outperforms the other for all markets or time frames. For example, the CNN model showed slightly superior performance for the EUR/USD pair in the evaluated time frame, while both models fared comparably in the volatile BTC/USD market.

The primary strength of the proposed architecture lies in its ability to dynamically identify and deploy the best-performing model for each specific market and time frame. By continually updating the models and adjusting to new incoming data, the system aims to optimize its predictive accuracy in real-time, effectively addressing the evolving nature of financial markets.

This highlights the architecture’s agility and adaptability, fulfilling its design aim to provide a scalable and efficient platform for real-time financial market prediction with high precision.
6. EXPERIMENTS AND RESULTS

Fig. 6.6.2: LSTM Model’s Prediction vs Actual for EUR/USD Pair.

Fig. 6.6.3: CNN Model’s Prediction vs Actual for BTC/USD Pair.
6. EXPERIMENTS AND RESULTS

Fig. 6.6.4: LSTM Model’s Prediction vs Actual for BTC/USD Pair.

Fig. 6.6.5: GRU Model’s Prediction vs Actual for Royal Bank of Canada.
6. EXPERIMENTS AND RESULTS

Fig. 6.6.6: LSTM Model’s Prediction vs Actual for Royal Bank of Canada.

Fig. 6.6.7: CNN Model’s Prediction vs Actual for Royal Bank of Canada.
### Table 6.6.1: Best models for each metric and asset

<table>
<thead>
<tr>
<th>Asset (Period)</th>
<th>Metric</th>
<th>Best Model</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple (2020, 1, 1 to 2023, 7, 1)</td>
<td>MAE</td>
<td>GRU</td>
<td>$0.29</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>GRU</td>
<td>$0.36</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>CNN/GRU</td>
<td>0.97</td>
</tr>
<tr>
<td>EURUSD=X (2020, 1, 1 to 2023, 7, 1)</td>
<td>MAE</td>
<td>CNN/LSTM/GRU</td>
<td>$0.01</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>CNN/LSTM/GRU</td>
<td>$0.01</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>CNN/GRU</td>
<td>0.9</td>
</tr>
<tr>
<td>BTC-USD (2020, 1, 1 to 2023, 10, 1)</td>
<td>MAE</td>
<td>LSTM</td>
<td>$801.61</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>LSTM</td>
<td>$1,146.89</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>LSTM</td>
<td>0.54</td>
</tr>
<tr>
<td>RBC (2020, 1, 1 to 2023, 7, 1)</td>
<td>MAE</td>
<td>GRU</td>
<td>$1.71</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>GRU</td>
<td>$2.15</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>GRU</td>
<td>0.85</td>
</tr>
<tr>
<td>USDCHF (2005, 1, 1 to 2023, 10, 1)</td>
<td>MAE</td>
<td>CNN/GRU</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>CNN/LSTM/GRU</td>
<td>$0.01</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>GRU</td>
<td>0.92</td>
</tr>
<tr>
<td>USDCHF (2022, 1, 1 to 2023, 10, 1)</td>
<td>MAE</td>
<td>GRU</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>ARIMA</td>
<td>$0.01</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>GRU</td>
<td>0.84</td>
</tr>
<tr>
<td>USDCAD=X (2000, 1, 1 to 2023, 7, 1)</td>
<td>MAE</td>
<td>GRU</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>GRU</td>
<td>$0.01</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>GRU</td>
<td>0.87</td>
</tr>
<tr>
<td>USDCAD=X (2000, 1, 1 to 2023, 10, 1)</td>
<td>MAE</td>
<td>CNN/GRU</td>
<td>$0.00</td>
</tr>
<tr>
<td></td>
<td>RMSE</td>
<td>CNN/GRU</td>
<td>$0.01</td>
</tr>
<tr>
<td></td>
<td>$R^2$ Score</td>
<td>GRU</td>
<td>0.91</td>
</tr>
</tbody>
</table>
CHAPTER 7

Conclusion and Future Work

This research has demonstrated the successful integration of machine learning algorithms, Apache Kafka, and microservices to create a robust, scalable, and efficient framework for real-time stock market prediction. The uniqueness of our approach lies in its adaptability and efficacy across different markets and periods, thanks to a three-phase implementation strategy. Phase 1 focuses on developing machine learning models and assessing their performance using rigorous evaluation metrics. Phase 2 and Phase 3 integrate these models into a real-time data processing and prediction system, effectively showcasing the framework’s ability to handle high-velocity, diverse data streams. Most notably, our framework addresses key limitations of previous models, which typically excel only in certain markets or specific time frames. Our system continuously evaluates and updates its models to ensure that the most accurate model is selected for each market at any given time. The evaluation metrics, including RMSE and MAE, validate the robustness of our approach, with high R2 scores signaling a strong degree of model accuracy. Furthermore, the adaptability of this framework promises potential for extensions across various other domains, filling a significant gap in the existing literature. Our study thus serves as a stepping stone in the field of real-time financial analytics and paves the way for future research to push the boundaries of what is currently possible. We believe that the scalable and adaptable framework proposed here offers a new paradigm in real-time analytics that could revolutionize not just stock market prediction, but also various other sectors in need of real-time data analysis.
While the architecture presented in this thesis shows promising results in the realm of stock market prediction, it’s worth noting that its applicability could extend far beyond this specific domain. One avenue for future work is to explore how the architecture can be adapted and deployed in other real-time data-generating systems. The model-switching feature that we proposed in this architecture could make it a versatile tool for solving complex, data-intensive problems in these domains. Furthermore, as the architecture is designed to be domain-agnostic, it has the potential to tackle challenges that are common across different fields requiring real-time analytics. Expanding the architecture to different domains would not only broaden its applicability but could also offer new insights into optimizing performance and reducing errors across various types of real-time data streams. Thus, future work will aim to test the generalizability and adaptability of the architecture in a broader range of applications, thereby contributing to the field of real-time data analytics as a whole.
REFERENCES


REFERENCES


VITA AUCTORIS

NAME: MohammadEhsan Akhavanpour

PLACE OF BIRTH: Tehran, Iran

YEAR OF BIRTH: 1995

EDUCATION: IAU Tehran South, B.Sc. in Information Technology, Tehran, Tehran, 2019

University of Windsor, M.Sc in Computer Science, Windsor, Ontario, 2023