ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ÉTUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

Order N°: 3974

THESIS

To obtain the: **DOCTORATE**

Research Center: Center of Research Mathematics and Applications in Rabat (CEREMAR)

Research Structure : Laboratoire de Mathématiques, Statistique et Applications (LMSA)

Discipline : Mathematics

Speciality: Applied Mathematics and Statistics

Presented and defended on: 27/07/2024

By:

Hassan OUKHOUYA

Advanced Predictive Modeling and Analysis of Financial Markets: A Comparative Study using Machine Learning and Statistical Methods in Moroccan and International Contexts

JURY

Abdelhak ZOGLAT	PES	Faculty of Science, Mohammed V University, Rabat, Morocco	President
Abdelhadi AKHARIF	PES	Faculty of Science and Technology, Abdelmalek Essaâdi University, Tangier, Morocco	Reviewer/ Examiner
Abdellatif EL AFIA	PES	High National School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University, Rabat, Morocco	Reviewer/ Examiner
Mohamed FIHRI	PH	Faculty of Science, Mohammed V University, Rabat, Morocco	Reviewer/ Examiner
Raby GUERBAZ	PH	Faculty of Law, Economics and Social Sciences - Aïn Sebaâ, Hassan II University, Casablanca, Morocco	Examiner
Khalid BELKHOUTOUT	PES	Faculty of Law, Economics and Social Sciences – Souissi, Mohammed V University, Rabat, Morocco	Invited
Aziz LMAKRI	PA	National Higher School of Arts and Crafts (ENSAM), Hassan II University, Casablanca, Morocco	Invited
Khalid EL HIMDI	PES	Faculty of Science, Mohammed V University, Rabat, Morocco	Thesis Supervisor

Academic year: 2023-2024

Dedication

My dear parents, Idir and Khadija, have given me the love and tenderness I always needed. I can never express how grateful I am to you for your trust and support in all my life choices. May God bless you and grant you long lives filled with joy.

To my dear brothers, Salem, Ismail, Yassine, and my dear sister, Malika, may God protect you all.

To my dear cousins and all the members of my family who have supported me at every moment.

To my dear friends and colleagues from FSR, FSJES Souissi, FSJES Aîn Sebaâ, ENSAM Casa, especially Ahmed, Mohamed, Khalid, Hamza, and Aziz, your support and friendship have made my university and doctoral journey a joyous and fulfilling experience. I am deeply grateful for your presence in my life.

Acknowledgement

This thesis was only possible with the help of several people. I want to express my gratitude to everyone who supported me in some way throughout this research, whether directly or indirectly. This thesis was conducted at the Laboratory of Applied Mathematics and Statistics (LMSA), Department of Mathematics at the Faculty of Sciences, Mohammed V University in Rabat, under the guidance of Mr. **Khalid El Himdi**, a PES at the Faculty of Sciences, Mohammed V University in Rabat, I am indebted to Professor El Himdi for his invaluable mentorship, scientific expertise, and unwavering support throughout my research. His exceptional human qualities have provided me with an excellent working environment, allowing me to learn and grow as a researcher. I extend my deepest gratitude and appreciation to him.

I want to express my sincere thanks to Mr. **Abdelhak Zoglat**, a PES at the Faculty of Sciences, Mohammed V University in Rabat, for graciously accepting the thesis defence committee president role. He honoured me by undertaking this important role despite his commitments and responsibilities. His expertise and experience as a distinguished professor, renowned for his significant achievements, have added immense value to the evaluation process. I am truly honoured to have him as the committee president. Please find here, Professor, an expression of my sincere thanks.

I would like to thank Mr. **Abdelhadi Akharif**, PES at the Faculty of Sciences and Technology at Abdelmalek Essaâdi University in Tangier, who rigorously evaluated and examined this thesis. His expertise, thorough examination, and valuable feedback have been in-

strumental in strengthening this work's academic quality and credibility. I sincerely thank him for his time, dedication, and contribution to the evaluation process.

I would also like to express my gratitude to Mr. **Abdellatif EL AFIA**, PES at the High National School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University in Rabat, for his rigorous evaluation and examination of this thesis. His expertise, thorough examination, and valuable feedback have greatly enhanced this work's academic quality and credibility. I sincerely appreciate his time, dedication, and contribution to the evaluation process.

I am grateful to Mr. **Mohamed Fihri**, PH at the Faculty of Sciences, Mohammed V University in Rabat, for serving as a thesis reviewer. His valuable insights, critical feedback, and expertise in the field have greatly enhanced the quality and rigour of this work.

I am grateful to Mr. **Raby Guerbaz**, PH at the Faculty of Law, Economics and Social Sciences Aïn Sebaâ of Hassan II University in Casablanca, for being a thesis reviewer. His valuable insights, critical feedback, and expertise in the field have greatly enhanced the quality and rigour of this work.

I am expressing my sincere appreciation to Mr. **Khalid Belkhoutout**, PES at the Faculty of Law, Economics, and Social Sciences Souissi, Mohammed V University in Rabat, for his guidance and continuous assistance during my time as a PhD candidate. His expertise and support have been invaluable in advancing my research.

I appreciate Mr. **Aziz Lmakri**, PA at the Higher National School of Arts and Crafts (ENSAM), Hassan II University in Casablanca, for his guidance, insightful discussions, and expertise in nonlinear time series. His contributions have greatly shaped my understanding and contributed to this research's success. I am thankful for the knowledge he shared.

Thank you all! H. OUKHOUYA.

Abstract

This thesis utilizes classical and machine learning (ML) techniques to investigate forecasting models for the Moroccan stock market. We evaluate the predictive performance of models including ARIMA, Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and hybrid models that combine these approaches. Our research comprises multiple studies, each offering distinct insights into financial time series forecasting. The first study compares ARIMA, SVR, and LSTM models for forecasting the Moroccan All Shares Index (MASI) index, revealing that nonlinear models like LSTM and SVR slightly outperform the linear ARIMA model for log closing prices over 37 days. The second study evaluates SVR, XGBoost, MLP, and LSTM models on the MSI 20 index, finding that SVR and MLP optimized via Grid Search (GS) achieve the highest accuracy.

In the third study, we forecast international stock indices (MASI, CAC 40, DAX, FTSE 250, NASDAQ, and HKEX), with the hybrid LSTM-XGBoost model optimized using GS providing superior accuracy. The fourth study focuses on the MASI, demonstrating the effectiveness of hybrid models like SVR-XGBoost and LSTM-XGBoost. The fifth study uses post-COVID historical financial data to compare ARIMA, Bilinear models, and LSTM networks, highlighting the advantages of hybrid models that combine linear and nonlinear elements. The final study introduces a robust hybrid forecasting model that integrates traditional ML techniques with deep learning (DL) methods, showing enhanced predictive accuracy for multivariate stock market series.

Our findings have significant implications for financial forecasting in the Moroccan stock market and international contexts, emphasizing the potential of hybrid models. This research offers valuable insights for investors, financial analysts, and policymakers, contributing to understanding predictive modeling in financial markets. It also aids the Moroccan financial ecosystem's portfolio management, risk assessment, and decision support systems. Future work will focus on refining hybrid models and exploring their application in algorithmic trading and realized volatility forecasting.

Keywords: Time Series; Modelling; Forecasting; Moroccan Stock Market; ML Models; ARIMA; SVR; XGBoost; MLP; LSTM; Hybrid Models; Bilinear Models; Stock Price Prediction.

Résumé

Cette thèse utilise des méthodes classiques et des techniques d'apprentissage automatique pour étudier les modèles de prévision pour le marché boursier marocain. Nous évaluons la performance prédictive de modèles incluant ARIMA, Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM) et des modèles hybrides qui combinent ces approches. Notre recherche comprend plusieurs études, chacune offrant des aperçus distincts sur la prévision des séries temporelles financières. La première étude compare les modèles ARIMA, SVR et LSTM pour la prévision de l'indice Moroccan All Shares Index (MASI), révélant que les modèles non linéaires tels que LSTM et SVR sont légèrement plus performants que le modèle linéaire ARIMA pour le logarithme des prix de clôture sur 37 jours. La deuxième étude évalue les modèles SVR, XGBoost, MLP et LSTM sur l'indice MSI 20 et révèle que les modèles SVR et MLP optimisés via Grid Search (GS) atteignent la plus grande précision.
Dans la troisième étude, nous prévoyons les indices boursiers internationaux (MASI, CAC 40, DAX, FTSE 250, NASDAQ et HKEX) à l'aide du modèle hybride LSTM-XGBoost optimisé grâce à GS, offrant une précision supérieure. La quatrième étude se concentre sur le MASI, démontrant l'efficacité des modèles hybrides tels que SVR-XGBoost et LSTM-XGBoost. La cinquième étude utilise des données financières historiques post-COVID pour comparer les modèles ARIMA, bilinéaires et les réseaux LSTM, en soulignant les avantages des modèles hybrides qui combinent des éléments linéaires et non linéaires. La dernière étude présente un modèle de prévision hybride robuste qui intègre des techniques ML traditionnelles avec des méthodes d'apprentissage profond (DL), montrant une précision prédictive améliorée pour les séries boursières multivariées.
Nos résultats ont des implications significatives pour les prévisions financières sur le marché boursier marocain et dans les contextes internationaux, soulignant le potentiel des modèles hybrides. Cette recherche offre des informations précieuses pour les investisseurs, les analystes financiers et les décideurs politiques, contribuant à la compréhension de la modélisation prédictive sur les marchés financiers. Elle aide également l'écosystème financier marocain à gérer les portefeuilles, évaluer les risques et mettre en place des systèmes d'aide à la décision. Les travaux futurs se concentreront sur l'affinement des modèles hybrides et l'exploration de leur application dans le trading algorithmique et la prévision de la volatilité réalisée.

Keywords: Séries Chronologiques ; Modélisation ; Prévision ; Marché Boursier Marocain; Modèles ML; ARIMA; SVR; XGBoost; MLP; LSTM; Modèles Hybrides; Modèles Bilinéaires; Prédiction des Cours de Bourse.

Detailed Summary

This thesis aims to enhance the predictive accuracy of forecasting models for the Moroccan stock market by leveraging both classical and machine learning (ML) techniques. It encompasses a comprehensive evaluation of various models, including ARIMA, Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and their hybrid counterparts. The research provides detailed insights into financial time series forecasting through studies focusing on the Moroccan All Shares Index (MASI) and other international stock indices.

The first study examines the predictive capabilities of ARIMA, SVR, and LSTM models for forecasting the log closing prices of the MASI index over 37 days. The findings indicate that nonlinear models such as SVR and LSTM slightly outperform the linear ARIMA model. This highlights the advantage of nonlinear approaches in capturing the complexities of financial time series data.

The second study focuses on the MSI20 index and evaluates the performance of SVR, XGBoost, MLP, and LSTM models. SVR and MLP models optimized through Grid Search (GS) demonstrate the highest accuracy, underscoring the importance of parameter optimization in enhancing model performance.

Expanding the scope to international stock indices (MASI, CAC 40, DAX, FTSE 250, NASDAQ, and HKEX), the third study identifies the hybrid LSTM-XGBoost model, optimized using GS, as the most accurate forecasting method. This study emphasizes the potential of hybrid models to deliver superior predictive performance across diverse financial markets.

Focusing again on the MASI index, the fourth study demonstrates the effectiveness of hybrid models such as SVR-XGBoost and LSTM-XGBoost. These models exhibit enhanced accuracy, validating the strategy of combining linear and nonlinear techniques to leverage their complementary strengths.

The fifth study utilizes post-COVID historical financial data to compare the performance of ARIMA, Bilinear models, and LSTM networks. The results highlight the advantages of hybrid models that integrate linear (ARIMA, Bilinear) and nonlinear (LSTM) elements, providing robust forecasts in the volatile post-pandemic financial environment.

The final study introduces a robust hybrid forecasting model that combines traditional ML techniques with deep learning (DL) methods. This model demonstrates enhanced predictive accuracy for multivariate stock market series, showcasing its potential for complex financial forecasting tasks.

The research findings significantly affect financial forecasting in the Moroccan stock market and international contexts. The demonstrated potential of hybrid models offers valuable insights for investors, financial analysts, and policymakers. These models can significantly contribute to the Moroccan financial ecosystem's portfolio management, risk assessment, and decision-support systems. Future work will refine hybrid models and explore their application in algorithmic trading and realized volatility forecasting.

This thesis underscores the efficacy of combining classical and ML approaches in financial forecasting. The hybrid models, particularly those optimized through GS, offer superior predictive accuracy, making them invaluable tools for forecasting stock market indices. This research enhances the understanding of predictive modelling in financial markets and provides practical solutions for improving investment strategies and financial decision-making processes.

Résumé détaillé

Cette thèse vise à améliorer la précision prédictive des modèles de prévision pour le marché boursier marocain en tirant parti à la fois des techniques classiques et des techniques d'apprentissage automatique. Elle comprend une évaluation complète de divers modèles, y compris ARIMA, Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), et leurs équivalents hybrides. La recherche fournit des informations détaillées sur la prévision des séries temporelles financières par le biais d'études portant sur l'indice Moroccan All Shares Index (MASI) et d'autres indices boursiers internationaux.

La première étude examine les capacités prédictives des modèles ARIMA, SVR et LSTM pour prévoir le logarithme des prix de clôture de l'indice MASI sur 37 jours. Les résultats indiquent que les modèles non linéaires, tels que SVR et LSTM, sont légèrement plus performants que le modèle linéaire ARIMA. Cela met en évidence l'avantage des approches non linéaires dans la capture des complexités des données de séries temporelles financières.

La deuxième étude se concentre sur l'indice MSI 20 et évalue les performances des modèles SVR, XGBoost, MLP et LSTM. Les modèles SVR et MLP optimisés par Grid Search (GS) démontrent la plus grande précision, soulignant l'importance de l'optimisation des paramètres dans l'amélioration des performances du modèle.

En élargissant le champ d'application aux indices boursiers internationaux (MASI, CAC 40, DAX, FTSE 250, NASDAQ et HKEX), la troisième étude identifie le modèle hybride LSTM-XGBoost, optimisé à l'aide de GS, comme la méthode de prévision la plus précise. Cette étude souligne le potentiel des modèles hybrides pour fournir des performances prédictives supérieures sur divers marchés financiers.

En se concentrant à nouveau sur l'indice MASI, la quatrième étude démontre l'efficacité des modèles hybrides tels que SVR-XGBoost et LSTM-XGBoost. Ces modèles font preuve d'une précision accrue, ce qui valide la stratégie consistant à combiner des techniques linéaires et non linéaires pour tirer parti de leurs forces complémentaires.

La cinquième étude utilise des données financières historiques post-COVID pour comparer les performances des modèles ARIMA, bilinéaires et des réseaux LSTM. Les résultats mettent en évidence les avantages des modèles hybrides qui intègrent des éléments linéaires (ARIMA, bilinéaire) et non linéaires (LSTM), fournissant des prévisions robustes dans l'environnement financier volatile post-pandémique.

La dernière étude présente un modèle de prévision hybride robuste qui combine des

techniques traditionnelles de ML avec des méthodes d'apprentissage profond (DL). Ce modèle démontre une précision prédictive améliorée pour les séries boursières multivariées, mettant en évidence son potentiel pour les tâches de prévision financière complexes.

Les résultats de la recherche ont une influence significative sur les prévisions financières du marché boursier marocain et dans les contextes internationaux. Le potentiel démontré des modèles hybrides offre des informations précieuses pour les investisseurs, les analystes financiers et les décideurs politiques. Ces modèles peuvent contribuer de manière significative à la gestion de portefeuille, à l'évaluation des risques et aux systèmes d'aide à la décision de l'écosystème financier marocain. Les travaux futurs permettront d'affiner les modèles hybrides et d'explorer leur application dans le trading algorithmique et la prévision de la volatilité réalisée.

Cette thèse souligne l'efficacité de la combinaison des approches classiques et de l'apprentissage automatique (ML) dans les prévisions financières. Les modèles hybrides, en particulier ceux optimisés par GS, offrent une précision prédictive supérieure, ce qui en fait des outils précieux pour prédire les indices boursiers. Cette recherche améliore la compréhension de la modélisation prédictive dans les marchés financiers et fournit des solutions pratiques pour améliorer les stratégies d'investissement et les processus de prise de décision financière.

Contents

List of Figures

List of Tables

List of Abbreviations

ACF Autocorrelation Function

ACVF Autocovariance Function

Adaboost Adaptive Boosting

ADAM Adaptive Moment Estimation

ADF Augmented Dickey-Fuller

AI Artificial Intelligence

AIC Akaike Information Criterion

ANNs Artificial Neural Networks

AR Autoregressive

AR-NN Autoregressive Neural Network

ARCH Autoregressive Conditional Heteroskedasticity

ARFIMA Autoregressive Fractionally Integrated Moving Average

ARIMA Autoregressive Integrated Moving Average

ARIMAX ARIMA with Exogenous Factor

ARMA Autoregressive Moving Average

ARMR Autoregressive Moving Reference

BIC Bayesian Information Criterion

BPTT Back Propagation Through Time

CMLE Conditional Maximum Likelihood Estimation

CNNs Convolutional Neural Networks

CPU Central Processing Unit

CSE Casablanca Stock Exchange

CSI China Securities Index

CSS Conditional-Sum-of-Squares

CV Cross-Validation

DBNs Deep Belief Networks

DL Deep Learning

DMLP Deep Multilayer Perceptron

DNN Deep Neural Network

DT Decision Tree

EKF Extended Kalman Filter

EM Expectation Maximization

EMD Empirical Mode Decomposition

EMH Efficient Market Hypothesis

ERM Empirical Risk Minimization

ES Exponential Smoothing

FCNN Fully Connected Neural Network

FFNN Feed Forward Neural Network

GAN Generative Adversarial Network

GARCH Generalised Autoregressive Conditional Heteroskedasticity

GD Gradient descent

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

GS Grid Search

HPM Hybrid Prediction Model

i.i.d. independent and identically distributed

IACF Inverse Autocorrelation Function

IoT Internet of Things

IPSO Improved Particle Swarm Optimization

ISE Istanbul Stock Exchange

KKT Karush-Kuhn-Tucker

KLD Kullback Leibler Divergence

KNN K-Nearest Neighbors

LM Lagrange Multiplier

LSTM Long Short-Term Memory

MA Moving Average

MADEX Moroccan Most Active Shares Index

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MARS Multivariate Adaptive Regression Spline

MASI Moroccan All Shares Index

ME Mean Error

MENA Middle East and North Africa

MFE Mean Forecast Error

ML Machine Learning

MLE Maximum Likelihood Estimation

MLP Multi-Layer Perceptron

MLR Multiple Linear Regression

MM Method of Moments

MNN Multilayer Neural Networks

MPE Mean Percentage Error

MSE Mean Square Error

MSI20 Morocco Stock Index 20

NLP Natural Language Processing

NMSE Normalized Mean Square Error

NNs Neural Networks

NSE National Stock Exchange

NSE Nigeria Stock Exchange

NYSE New York Stock Exchange

PACF Partial Autocorrelation Function

PCA Principal Component Analysis

QPP Quadratic Optimization Problem

RBF Radial Basis Function

ReLU Rectified Linear Unit

RF Random Forests

RMSE Root Mean Squared Error

RNN Recurrent Neural Networks

S&P 500 Standard & Poor's 500

SARIMA Seasonal ARIMA

SE Standard Errors

SGD Stochastic Gradient Descent

SRM Structural Risk Minimization

SRNN Simple Recurrent Neural Network

SSE Shanghai Stock Exchange

SVMs Support Vector Machines

SVR Support Vector Regression

SVs Support Vectors

TBATS Trigonometric Box-Cox ARMA Trend Seasonality

TSR Time Series Regression

U Statistics Theil's U Statistics

U.S. United States

VC Vapnik-Chervonenkis

XGBF-DNN eXtreme Gradient Boosting Deep Neural Networks

XGBoost eXtreme Gradient Boosting

Chapter 1

General Introduction

1.1 Literature Review

Time series forecasting is a crucial tool that combines statistics and Machine Learning (ML) to project future trends using historical data. Its applications are wide-ranging, including finance, marketing, and governmental agencies. Models are customized to predict stock prices for informed investment decisions and to forecast environmental conditions for proactive disaster management. By carefully analyzing past observations, time series forecasting provides valuable guidance for decision-making and helps mitigate future uncertainties (Tamatta, 2018). This dynamic field not only influences sectors such as finance and econometrics but also extends its reach to diverse areas like medicine, earth sciences, and signal processing. In this thesis, we will explore the intricate world of time series forecasting, with a focus on its critical role in understanding the complexities of stock market indexes, which are fundamental to economic stability.

Many research efforts have investigated the forecast accuracy of financial time series utilizing diverse modelling approaches. These models range from traditional linear frameworks like the *Autoregressive Integrated Moving Average (ARIMA)* model to more complex nonlinear methods. While ARIMA and its variations (*Autoregressive (AR)*, *Moving Average (MA)*, and *Autoregressive Moving Average (ARMA)* models) (G. E. Box et al., 1970; Cochrane, 1997; K.W. Hipel, 1994; J. Lee, 2018) are effective at capturing linear relationships and seasonal patterns (*Seasonal ARIMA (SARIMA)* model), their inflexibility limits their applicability to situations where data display nonlinear behavior. To address this limitation, researchers have explored nonlinear models such as the *Autoregressive Conditional Heteroskedasticity* (ARCH) (Engle, 1982) and *Generalised Autoregressive Conditional Heteroskedasticity (GARCH)* models (Bollerslev, 1986). These models are well-known for their ability to capture changing variances (heteroscedasticity) present in financial time

series. However, such models may not always be suitable for forecasting non-financial data, such as aviation demand (Xu et al., 2019), as they focus primarily on volatility in stock market returns. In contrast, Artificial Intelligence (AI) models offer promising opportunities for forecasting across a wide range of domains. These models effortlessly capture nonlinear patterns in data without imposing distributional assumptions or requiring data transformations, making them a compelling alternative.

In the stock market, accurate forecasts of future stock values are crucial for making profitable investment decisions. This idea was famously expressed by Gordon Gekko in the movie "Wall Street" in 1987. However, it challenges the traditional Efficient Market Hypothesis (EMH), which states that market prices instantly reflect all available information, making stock prices unpredictable and following a random walk pattern (Fama, 1970). Despite the assertion of EMH that "the market is unbeatable," recent decades have seen a rise in studies that challenge this notion. These studies show that prediction is possible through robust algorithms and methods, even in the complex and volatile stock market (Fama, 1991), which is influenced by various external factors such as political events, global economic conditions, corporate performance, investor sentiment, and unforeseen occurrences. This complex interplay of elements makes forecasting stock prices a challenging task for both academics and experts. Volatility, which is synonymous with unpredictability and risk, is an essential aspect of stock market dynamics. It significantly influences investment decisions and portfolio strategies. Therefore, it is crucial to understand, analyze, and forecast volatility in order to provide investors with informed financial guidance. However, empirical studies have shown that volatility is inherently unstable over time, revealing certain patterns. The COVID-19 pandemic has further intensified this volatility, leading to unprecedented levels of uncertainty and fluctuations in global financial markets, including the Moroccan stock market.

The conventional methods used to model financial time series often struggle to capture their inherent complexities, which are characterized by strong non-linear patterns. However, with the rapid development of computational power, attention has shifted towards leveraging AI and ML techniques, such as *Artificial Neural Networks (ANNs)* , *Support Vector Machines (SVMs)* , and *eXtreme Gradient Boosting (XGBoost)*, to overcome these limitations. These methods aim to uncover the elusive and ever-evolving patterns within financial data, thus paving the way for more advanced forecasting capabilities (M. Kumar & M., 2014). Unlike traditional approaches like Box and Jenkins ARIMA models,

which offer simplicity but may struggle with nonlinearity and nonstationarity, SVMs provide a compelling alternative. By harnessing SVMs, forecasters can effectively address the challenges posed by nonlinear time series, ultimately improving predictive performance. The growing popularity of SVMs can be attributed to their ability to train complex nonlinear systems using sample data, as demonstrated by their successful application in economic forecasting, including predicting stock market indicators based on the economic growth of various countries. In contrast to ARIMA models, which may exhibit bias and convergence to the series mean over forecast horizons, SVMs offer a promising avenue for more robust and accurate predictions in the realm of financial and economic forecasting. To overcome the challenges posed by nonlinear and non-stationary time series (Okasha, 2014), alternative forecasting methods like SVMs are essential. Studies comparing ARIMA, SVM, and ANN models often focus on economic data. Ping-Feng Pai, 2005 suggests a hybrid approach combining ARIMA and SVMs for stock price forecasting, emphasizing parameter optimization. SVM, based on Vapnik's Structural Risk Minimization (SRM) principle (V. N. Vapnik, 1999), identifies support vectors for robust generalization (L. J. Cao & Tay, 2003; T. Raicharoen, 2003).

Neural Networks (NNs) excel in learning hierarchical representations, unlike traditional methods like ARIMA. With the rise of affordable storage and advanced computing, Deep Learning (DL) methods like *Deep Belief Networks* (DBNs) and *Convolutional Neural Networks* (CNNs) have achieved breakthroughs by exploiting local data dependencies (Lipton et al., 2015). ML techniques, including DL algorithms such as *Multi-Layer Perceptron (MLP)*, *Recurrent Neural Networks* (RNNs) and *Long Short-Term Memory* (LSTM), have brought about a revolution in prediction by capturing complex relationships between variables. These methods, along with ML-based approaches such as SVM, XGBoost and *Random Forests* (RF), are being applied in various fields, including finance J. Cao et al., 2019; Siami-Namini et al., 2018. LSTM, in particular, is renowned for its effectiveness in forecasting time-series data K. Chen et al., 2015; Rhanoui et al., 2019, particularly in predicting S&P 500 volatility. Although stock market prediction remains challenging, ANNs hold promise, as demonstrated by early successes in forecasting Moroccan All Shares Index (MASI) returns. Current research is focused on combining AI techniques and integrating new factors to enhance forecasting accuracy. However, there is limited empirical evidence comparing traditional methods like ARIMA with DL-based approaches like LSTM in economic and financial time series prediction (Siami-Namini et al., 2018).

The performance of stock market prediction models is crucial. DL models can be optimized using algorithms like Grid Search (GS) to improve their predictive capabilities. Afterward, these models should be evaluated using appropriate metrics to determine how accurate and reliable their forecasts are. Performance measures play a crucial role in objectively quantifying the effectiveness of models and enabling comparisons. The main purpose of these measures is to provide quantitative indicators that show the accuracy and quality of the predictions generated by the models. Researchers, investors, and financial decision-makers can use these measures to develop well-informed investment strategies and effectively manage portfolios. By employing performance metrics such as Mean Forecast Error (MFE) (G. Zhang et al., 1998), Mean Absolute Error (MAE) (G. P. Zhang, 2007), Mean Percentage Error (MPE) (Flaherty & Lombardo, 2000), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Normalized Mean Square Error (NMSE) (L. J. Cao & Tay, 2003), Coefficient of Determination (R^2) (Di Bucchianico, 2008), and Theil's U Statistics (U) (Park, 1999), stakeholders can determine the suitability of a model for specific stock price prediction requirements.

The primary aim of this thesis is to conduct a comprehensive comparative analysis of various forecasting models. The goal is to determine the most effective approaches for minimizing forecast errors and enhancing prediction accuracy. In particular, the study focuses on forecasting the daily closing prices of the Moroccan stock market index (MASI), Morocco Stock Index 20 (MSI20) and an international stock market index. The analysis includes classical stochastic models such as univariate ARIMA, as well as modern ML models like SVR, XGBoost, MLP, and LSTM. Additionally, the research explores the development of novel hybrid models that integrate elements from both classical and ML-based approaches. The evaluation of these models will be based on historical data, with a specific emphasis on predicting the MASI with MSI 20 indices and other selected stock market indices, such as DAX, HKEX, CAC 40, NASDAQ, and FTSE 250. The overall objective is to identify the most accurate forecasting methods that enable investors to make informed decisions, mitigate risks, and maximize profits, thereby contributing to the economic advancement of Morocco. This investigation will also address the impact of the COVID-19 pandemic on stock market dynamics and explore strategies to mitigate potential losses during periods of market instability. The research will adopt three distinct approaches: univariate stochastic modeling, multivariate machine learning, and deep learning, with a particular focus on hybrid models that integrate elements from both traditional and

modern forecasting techniques. Through this comparative study, the thesis aims to provide valuable insights into effective forecasting methodologies for stock market indices. This will facilitate informed decision-making and foster economic growth.

The thesis makes multifaceted contributions by analyzing and modeling the dynamics of stock market indices, with a focus on the MASI index. Firstly, the thesis examines the closing price dynamics of the MASI index using a range of methodologies, including the traditional Box Jenkins methodology and advanced ML and DL models. The analysis aims to uncover underlying patterns and trends in the MASI index, facilitating more accurate forecasting and decision-making in the Moroccan financial market. Additionally, the thesis extends its scope to model daily prices for various stock market indices, such as MSI20, CAC 40, DAX, FTSE 250, NASDAQ, and HKEX. By leveraging ML and DL approaches in both univariate and multivariate scenarios, the research provides comprehensive insights into the dynamics of these indices, enhancing our understanding of global financial markets.

Moreover, the thesis rigorously compares the performance and precision of traditional forecasting models with ML techniques in minimizing prediction errors. By developing and evaluating models such as ARIMA, SVR, XGBoost, MLP, LSTM, and hybrid approaches, the research identifies the most effective methodologies for predicting stock market indices. This comparative analysis employs the GS algorithm for hyperparameter optimization and evaluates model accuracy using performance measures. Furthermore, the thesis utilizes the skforecast library to conduct backtesting with the XGBoost model, demonstrating its forecasting accuracy for daily stock prices. Through this comprehensive approach, the thesis advances our understanding of stock market dynamics, providing valuable insights for investors, financial analysts, and policymakers. By elucidating the strengths and limitations of various forecasting methodologies, the research aims to improve the efficiency and effectiveness of decision-making processes in financial markets, contributing to the stability and growth of the economy.

1.2 Related Works

Recent studies have highlighted the importance of forecasting financial time series data. These data are known for their volatility and vulnerability to economic factors (Mustapa & Ismail, 2019). While traditional linear models like ARIMA and GARCH have been com-

monly employed, their limitations have prompted researchers to explore modern ML-DL single and hybrid techniques such as SVMs, XGBoost, MLP, and LSTMs. These techniques have demonstrated superior performance in capturing nonlinear patterns. Comparative studies in quantitative finance research have substantiated the effectiveness of these modern approaches compared to traditional methods. Moreover, nonlinear stochastic models like GARCH have emerged as viable alternatives to ARIMA, providing increased flexibility in capturing the complexities of financial data (Cocianu & Grigoryan, 2016; Z. Li & Tam, 2018; Rundo et al., 2019; Sezer et al., 2020).

Adebiyi et al., 2014 conducted a study comparing the performance of ARIMA and ANNs models using New York Stock Exchange (NYSE) data. Their findings demonstrated that the NN model outperformed ARIMA in terms of accuracy in forecasting. Similarly, Wijaya et al., 2010 investigated stock price forecasting in Indonesia, comparing the ANN and ARIMA methods. Their results showed that the ANN approach resulted in smaller errors compared to ARIMA. Falloul and Mansouri, 2014 aimed to model the performance of the MASI. Through statistical analysis, they identified the GARCH process as the most suitable model for this purpose. Various studies have been conducted to explore forecasting techniques in financial markets. Tang et al., 2003 extended the ARMA-GARCH model for stock price prediction, which demonstrated improved results compared to AR-GARCH models. Hossain and Nasser, 2008; Hossain and Nasser, 2011 compared SVM with ARMA-GARCH and GARCH models, and found that SVM's long memory property was beneficial for forecasting financial returns. Xu et al., 2019 applied SARIMA-SVR to aviation industry indicators, which resulted in enhanced accuracy. Ping-Feng Pai, 2005 proposed a hybrid ARIMA-SVM methodology for stock price forecasting, with a focus on parameter optimization for optimal results. M. Kumar and M., 2014 identified ARIMA-SVM as the best model for stock index return prediction, achieving high accuracy and returns. The empirical studies have substantiated the efficacy of XGBoost in stock price prediction. Yuan et al., 2021 have introduced a predictive model that utilizes XGBoost and Grid SearchCV algorithms. Their findings demonstrate superior performance compared to other ML methods, with a significant decrease in RMSE of 1.39%, 2.43%, and 8.33% compared to SVM, NN, and LightGBM algorithms, respectively. Moreover, their analysis provides valuable insights into the key features that influence closing stock prices, thereby offering valuable insights for predictive modeling and decision-making processes in financial markets. Siami-Namini et al., 2018 analyzed historical financial time series data

from January 1985 to August 2018. Their study found that LSTM, a DL-based algorithm, performed better than the traditional ARIMA model. LSTM demonstrated an average reduction in error rates ranging from 84% to 87% compared to ARIMA, indicating its superior predictive capability in financial forecasting. In a related study by Rhanoui et al., 2019, a comparative analysis was conducted to forecast financial budget time series. The findings of their study demonstrated that the LSTM model outperformed the ARIMA model, showcasing the effectiveness of DL approaches in financial forecasting tasks. Y. Wang et al., 2023 employed the XGBoost machine learning algorithm to forecast the closing price of the China Securities Index (CSI) 300 futures contract, confirming its suitability in predicting financial time series. Their research showcased substantial improvements in predictive efficiency by optimizing empirical modal decomposition. The results highlighted the accuracy of both LSTM and XGBoost models, with empirical modal decomposition particularly enhancing the predictive accuracy of XGBoost. K.-j. Kim, 2003 conducted a comparative study evaluating SVM, backpropagation NNs, and case-based reasoning for financial forecasting. The research highlighted SVM as a promising alternative for stock market prediction, showcasing its potential efficacy in financial markets. In their study, Kara et al., 2011 aimed to establish and compare two models for predicting the daily direction of movement of the Istanbul Stock Exchange (ISE) National 100 Index. The experiments conducted by the authors revealed that the ANN model (75.74%) outperformed the SVM model (71.52%) in terms of average performance. This indicates that the ANN model has superior predictive capability in this specific context.

Misra et al., 2018 discovered that applying Principal Component Analysis (PCA) improved the accuracy of predictions made using the Linear Regression Model. While SVM excelled with non-linear classification stock market, Linear Regression performed well with linear data due to its high confidence values. The RF approach achieved high accuracy rates in binary classification models, while MLP showed the lowest margin of error for predictions. Hiransha et al., 2018 conducted a study on price forecasting for five companies listed on the National Stock Exchange (NSE) and NYSE. The study used a model trained on a single NSE company and employed four ML models: RNNs, MLP, CNN, and LSTM. The results showed that NNs outperformed the linear model (ARIMA) (Ariyo et al., 2014). Among the neural models, CNN demonstrated superior performance. This was attributed to its heightened sensitivity in detecting sudden changes in the data structure by utilizing specific time windows for predictions. Ji et al., 2021 proposed a novel hybrid

model that combines special improved particle swarm optimization (IPSO) with LSTM for stock price forecasting. Their model outperformed baseline models, such as SVR, LSTM, and PSO-LSTM, in experimental results, specifically on the Australian stock market index. K. Chen et al., 2015 conducted a study comparing the effectiveness of LSTM with a random prediction method in modeling and forecasting the China stock market. The research findings revealed that the LSTM model greatly improved the accuracy of stock return forecasts, increasing it from 14.3% to 27.2%. These results highlight the effectiveness of LSTM in predicting the volatile and unpredictable nature of the Chinese stock market.

Table 1.1: Summary of Selected State-of-the-Art Studies

Reference	Method	Stock Market	Objective
Nabipour et al., 2020	Decision Tree, RF, Adaptive Boosting (Adaboost), XGBoost, SVM, Naïve Bayes, K-Nearest Neighbors (KNN), Logistic Regression, ANN, RNN, and LSTM	NYSE & Petroleum, diversified financials, basic metals and non-metallic minerals	Predicting stock market trends
Elmsili and Outtaj, 2021	Ridge regression, LASSO regression, SVM, KNN, RF, and Adaboost	Casablanca Stock Exchange (CSE)	Stock returns prediction
Samarawickrama and Fernando, 2017	RNN, Simple Recurrent Neural Network (SRNN), Gated Recurrent Unit (GRU) and LSTM	Sri Lankan stock market	Predicting daily stock prices
Hussein et al., 2015	ANN, MLP and Kullback Leibler Divergence (KLD)	Stock Market's Daily Data	Stock market prediction

Continued on next page

Table 1.1: Summary of Selected State-of-the-Art Studies (Continued)

Reference	Method	Stock Market	Objective
Sheta et al., 2015	SVM, MLP and ANN	S&P 500 Dataset	Predicting stock market index
Yu et al., 2021	LSTM, XGBoost, RNN and LSTM-XGBoost	S&P 500 time series	Prediction of stock prices (Open, High, Low, Close, Volume, Adj-Close)
Bhamidipati and Saisanthiya, 2023	RF and Backtesting	NSE	Stock price prediction
Fazeli and Houghten, 2019	LSTM and Technical indicators	Apple, Microsoft, Google and Intel stocks	Prediction of stock market trends
Zheng et al., 2021	SVR and BA-SVR	Chinese stock market	Stock index prediction
Ariyo et al., 2014	ARIMA	NYSE & Nigeria Stock Exchange (NSE)	Stock price prediction
Y. Wang and Guo, 2020	ARIMA, XGBoost, GSXGB and DWT-ARI-MA-XGBoost	10 stock data sets	Forecasting of stock market volatility

The core objective of this thesis is to compare and discuss three prominent forecasting methods for Moroccan and international stock markets: stochastic, NNs, and ML approaches. We specifically aim to evaluate ARIMA's modelling and forecasting capabilities compared to SVR, XGBoost, MLP, and LSTM models, individually and within novel hybrid frameworks. The thesis consists of five chapters, each with the following structure:

- *Chapter 2*, introduces the stochastic approach and time series analysis. It first provides an overview of basic financial time series analysis concepts, including properties like stationarity and parsimony. Then, It Discusses various stochastic time series models, such as Box-Jenkins ARIMA linear models and non-linear models like GARCH. Finally, the presentation of statistical learning theory will provide a foun-

dation for understanding subsequent chapters.

- Chapter 3, provides ML approaches for time series forecasting: first, an Introduction to ML approaches and an in-depth discussion on SVR and XGBoost, including mathematical models and graphical representations. Second, NN are explored in time series forecasting, covering models such as MLP, RNN, and LSTM. Finally, a hybrid model combining LSTM and XGBoost is described, and XGBoost Models are backtested.
- Chapter 4, presents experimental forecasting of the Moroccan Stock Market. It starts with an analysis and discussion of experimental results from forecasting the MASI using ARIMA, SVMs, and LSTM models. Then, the model's performance is synthesized and compared, leading to general conclusions and suggestions for future research directions.
- Chapter 5, presents a comparative analysis of ML methods, specifically SVR, XGBoost, LSTM, and MLP. The analysis includes discussing experimental results and their application for forecasting the Morocco Stock Index 20. The performance of each model is summarized and compared, leading to general conclusions and future research prospects.
- Chapter 6, explores the application of XGBoost, LSTM, and the hybrid LSTM-XGBoost model for predicting trends in international stock markets. It includes an analysis of experimental results obtained from forecasting various international stock market indices using these models. Additionally, it examines the predictive performance of XGBoost models through backtesting. The chapter concludes by discussing implications for future research in international stock market forecasting.
- Chapter 7, presents a comparative study of predictive modelling techniques applied to the Moroccan financial market. Specifically, it focuses on three techniques: ARIMA, Bilinear models, and LSTM networks. We use post-COVID historical financial data to evaluate these models and employ preprocessing techniques, such as normalization, to ensure compatibility. The models are then trained and assessed using MAE, RMSE, and directional forecast accuracy metrics.
- Chapter 8, examines the modelling and forecasting of daily prices for the Moroccan All Shares Index (MASI) using different machine learning methods, such as XGBoost, SVR, LSTM, MLP, and their combinations. We utilize GS to optimize model

parameters, and our comprehensive comparative study shows that hybrid models like SVR-XGBoost and LSTM-XGBoost achieve higher accuracy in predicting daily prices.

- *Chapter 9*, summarizes the main findings from the previous chapters, highlighting the effectiveness of various predictive models and the superior performance of hybrid approaches. It discusses the implications for portfolio management, risk assessment, and decision-support systems in the Moroccan financial market. The research's practical applications for investors and policymakers are explored, and future research directions are suggested, including improvements to hybrid models and their use in algorithmic trading and volatility forecasting. The chapter concludes with reflections on the contributions and potential impact of the research on financial predictive modelling.

Chapter 2

Stochastic approach for time series analysis

" All models are wrong, some are useful. "

– GEORGE E. P. BOX, British statistician

Contents

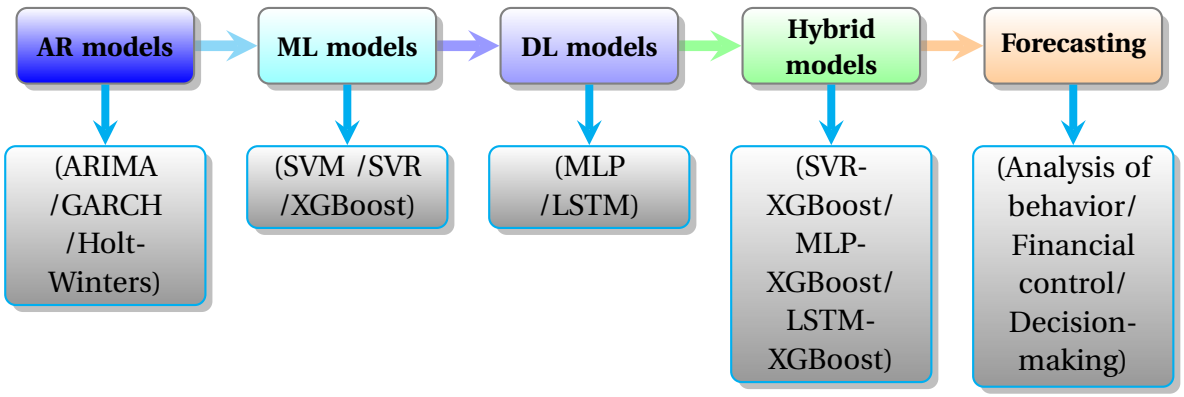
2.1 Introduction to time series analysis

The introduction to time series analysis in the first chapter discusses the significance and applications of dynamic modelling in various fields such as business, economics, finance, and computer science (Siami-Namini et al., 2018). Time series analysis aims to understand the patterns in sequential data, develop suitable models, and forecast future values. Literature showcases numerous forecasting models, with parameters estimated from available data (K.W. Hipel, 1994). Time series analysis involves fitting a model to the data, allowing for in-depth analysis, forecasting, and simulation.

2.1.1 Timeline of time series analysis and forecasts

Time series forecasting involves collecting past observations to construct a mathematical model that reflects the underlying data generation process (G. Zhang, 2003, 2007). This is particularly useful when the statistical pattern of repeated observations is poorly understood or when an explanatory model is lacking. Forecasting plays a crucial role in decision-making processes, necessitating the development of accurate models (Adhikari & Agrawal, 2013).
Over time, time-series analysis and forecasting have seen significant advancements, with researchers and practitioners exploring various methodologies. Early approaches like AR models focused on capturing linear dependencies, while hybrid models combined statistical techniques with ML algorithms to improve accuracy. ML models such as SVM and XGBoost introduced non-linearity, while DL models like LSTM addressed complex patterns. The choice of method now depends on data characteristics, and ongoing research aims to refine our understanding further. A timeline of strategies adopted for financial time-series modelling is presented in Figure 2.1.

Figure 2.1 – Timeline of adopted approaches for time-series analysis and forecasting.



2.2 Stochastic process and Time series

A stochastic process is a collection of random variables indexed by time $Y(\omega, t)$, where ω belongs to a sample space, and t belongs to an index set for a fixed value of t. $Y(\omega, t)$ represents a random variable. When considering a specific ω, $Y(\omega, t)$ as a function of t is referred to as a sample function or realization (Wei, 2006).

Definition 1. *Consider the probability space* $(\Omega, \mathcal{F}, \mathbb{P})$. *Where* Ω *is the sample space,* $\mathcal{F}$ *is a* σ*-algebra, and* $\mathbb{P}$ *is a probability measure. In addition to an index set that we note* T. *Through the rest of this work, we will suppose that the index set* T *represents time. A*

stochastic (or random) process is a set of time-indexed random variables $\mathrm{Y}(\omega, t)$. *Where* $\omega \in \Omega$ *and* $t \in \mathrm{T}$.

- $\mathrm{Y}(\omega, t)$ *is said to be a discrete-time stochastic process if the time set T is finite.*
- $\mathrm{Y}(\omega, t)$ *is said to be a continuous-time stochastic process if the time set* T *is infinite. For a fixed value of* t, $\mathrm{Y}(\omega, t)$ *is a random variable.*

Definition 2. *A time series is a set of observations arranged in chronological order. It is represented as* $\{\mathrm{Y}_t\}$, *where* t *denotes time and belongs to a specific time set. Each value of* t *corresponds to a random variable,* Y_t.

A time series is non-deterministic, i.e. what will happen in the future is impossible to predict with precision. Generally, a time series $\{\mathrm{Y}_t, t \in \mathbb{Z}\}$ is assumed to follow a certain probability model (Cochrane, 1997), which describes the joint distribution of the random variable Y_t. A stochastic process is a mathematical way of describing the probability structure of a time series (K.W. Hipel, 1994). Thus, the sequence of observations of the series is, in fact, a realization example of the stochastic process which produced it. The n-dimensional distribution function is defined for all $\mathrm{Y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ by:

$$\mathrm{F}_{\mathrm{Y}_{t_1},\ldots,\mathrm{Y}_{t_n}}(y_1, \ldots, y_n) = \mathbb{P}\{\omega \in \Omega : \mathrm{Y}_{t_1} \leq y_1, \ldots, \mathrm{Y}_{t_n} \leq y_n\}. \tag{2.1}$$

The assumption often made in time series analysis is that the variables Y_t follow an independent and identically distributed (i.i.d.) normal distribution. However, as noted by Cochrane, 1997, time series data typically exhibit certain regular patterns over the long term, deviating from strict i.i.d. behaviour. For instance, if today's temperature in a city is unusually high, it is reasonable to expect tomorrow's temperature also to be high. Properly applied time series forecasting considers such patterns, leading to forecasts that closely approximate the true values.

2.3 Concept of stationarity and Model parsimony

Stationarity in a stochastic process refers to a state of statistical equilibrium (K.W. Hipel, 1994). In these processes, key statistical properties such as the mean (μ) and variance (σ^2) remain constant over time, essential for constructing time series models to forecast future values. This assumption also helps simplify the mathematical complexity of the model. There are two stationary processes, each distinguished by its unique characteristics, which are presented in the following theorem:

Theorem 1. *A process* $\{\mathrm{Y}_t, t \in \mathbb{Z}\}$ *is said to be* ***Strongly Stationary, Strictly Stationary*** *or* ***Strict-sense Stationary*** *if the joint probability distribution function of* $\{\mathrm{Y}_{t_1+k}, \ldots, \mathrm{Y}_{t_n+k}\}$ *is independent* $\forall\, t_1, \ldots, t_n$ *and* $\forall\, n \in \mathbb{N}$. *Thus, for a strong stationary process, the joint distribution of any possible set of random variables from the process is independent of time (Cochrane, 1997) i.e.:*

$$\mathrm{F}_{\mathrm{Y}_{t_1+k},\ldots,\mathrm{Y}_{t_n+k}}(y_1, \ldots, y_n) = \mathrm{F}_{\mathrm{Y}_{t_1},\ldots,\mathrm{Y}_{t_n}}(y_1, \ldots, y_n). \tag{2.2}$$

In practical applications, it is not always necessary to have strong stationarity. This has led to the study of a weaker form known as **weakly stationary of order** k (J. Lee, 2018). In this form, the statistical moments of a stochastic process up to a certain order depend only on the differences in time, regardless of when the data used for moment estimation occurred. For example, a stochastic process $\{\mathrm{Y}_t, t \in \mathbb{Z}\}$ is considered second-order stationary if it has a mean and variance that do not change over time, and its covariance

values $\mathbb{C}ov(x_t, x_{t-k})$ depend only on k. For simplicity, we will use "stationary" to refer to "second-order stationarity".
Stationarity is important in time series analysis. Both strong and weak forms of stationarity are significant, with weak stationarity often detected through mathematical tests such as the Dickey-Fuller test (Dickey & Fuller, 1979). As G. E. Box et al., 1970 discussed, stationarity simplifies the modelling process, although longer historical data may introduce non-stationarity. However, short-term stationary processes can still effectively model short-time series. Non-stationary series often show trends or seasonal patterns, requiring differentiation and power transformations to achieve stationarity (K.W. Hipel, 1994).

2.3.1 Model Parsimony in Time-Series Analysis

The principle of **parsimony**, also known as the "Occam's razor" principle, plays a crucial role when constructing a time-series model. Parsimony suggests that among several suitable models, the one with the fewest parameters should be chosen to adequately represent the underlying time-series data. In other words, simplicity is preferred when explaining complex phenomena.

1. **Choosing Simplicity**: Chatfield, 1996; G. Zhang, 2003 emphasize that the simplest model capable of capturing the essential characteristics of the time series should be favoured. This aligns with Occam's razor, which encourages selecting the most basic explanation when faced with competing options.
2. **Risk of Overfitting**: As model complexity increases, so does the risk of overfitting. Overfitting occurs when a model fits the training data too closely but fails to generalize well to unseen data. Srivastava et al., 2014 highlight this concern. An overfitted model may perform well on historical data but not provide reliable forecasts.
3. **Guiding Principle**: Parsimony acts as a guiding principle to mitigate overfitting. By favouring simpler models, we reduce the chances of introducing unnecessary complexity. A parsimonious model strikes a balance between capturing essential patterns and avoiding overfitting.

In conclusion, when establishing time-series predictions, practitioners should carefully consider model efficiency and select the most appropriate model from the available options.

2.4 Autocorrelation and partial autocorrelation functions

The *Autocorrelation Function* (ACF) and *Partial Autocorrelation Function* (PACF) analyses are essential for identifying suitable models for time series data. They quantify the relationship between observations and are plotted against time lags to inform modelling and forecasting decisions, aiding in determining AR and MA orders.

2.4.1 Autocovariance and autocorrelation functions

For a stationary process $\{Y_t\}$, we have the mean $\mathbb{E}(Y_t) = \mu$ and variance $\mathbb{V}ar(Y_t) = \mathbb{E}(Y_t - \mu)^2 = \sigma^2$ of the time series, which are constant, and the covariances $\mathbb{C}ov(Y_t, Y_s)$, which are functions only of the time difference $|t-s|$. Hence, in this case, we write the covariance between Y_t and Y_{t+k} at lag k is defined as (Cochrane, 1997) :

$$\gamma_k = \mathbb{C}ov(Y_t, Y_{t+k}) = \mathbb{E}\big[(Y_t - \mu)(Y_{t+k} - \mu)\big], \tag{2.3}$$

and the correlation (Wei, 2006) between Y_t and Y_{t+k} at lag k is defined as:

$$\rho_k = \frac{\mathbb{C}ov(Y_t, Y_{t+k})}{\sqrt{\mathbb{V}ar(Y_t)}\sqrt{\mathbb{V}ar(Y_{t+k})}} = \frac{\gamma_k}{\gamma_0}, \tag{2.4}$$

where we note the autocovariance at lag zero i.e. $\mathbb{V}ar(Y_t) = \mathbb{V}ar(Y_{t+k}) = \gamma_0$ is the variance of the time series. As functions of k, γ_k is called the Autocovariance Function and ρ_k is called the Autocorrelation Function in time series analysis because they represent the covariance and correlation between Y_t and Y_{t+k} from the same process, separated only by k time lags. The definition clearly shows that the autocorrelation coefficient ρ_k is dimensionless and independent of the measurement scale. Also, $-1 \leq \rho_k \leq 1$. Statisticians G. E. Box et al., 1970 termed γ_k as the theoretical Autocovariance Function (ACVF) and ρ_k as the theoretical ACF.

2.4.2 Partial autocorrelation function

In addition to the autocorrelation between Y_t and Y_{t+k}, another measure, known as the Partial Autocorrelation Function is used to measure the correlation between an observation k period ago and after taking account for observations with intermediate lags, the current observation (i.e. at lags $< k$) (J. Lee, 2018). PACF(1) and ACF(1) are the same at lag 1. The detailed formulae for calculating PACF are given in K.W. Hipel, 1994. Mathematically, the PACF at lag k is defined as follows:

$$\phi_{kk} = \begin{cases} \rho_1 & \text{if } k = 1 \\ Corr(Y_t, Y_{t+k} | Y_{k+1}, \ldots, Y_{t+k-1}) & \text{otherwise.} \end{cases} \tag{2.5}$$

Because the stochastic process generating a time series is usually unknown, the real or theoretical ACF and PACF values cannot be determined. However, these values should be estimated using the training data, i.e. the currently available time series. The ACF and PACF values estimated from the training data are referred to as sample $\hat{\boldsymbol{\rho}}_{\boldsymbol{k}}$ and $\hat{\boldsymbol{\phi}}_{\boldsymbol{k}}$, respectively. With only one realization $(Y_1, Y_2, \ldots, Y_n)$. A natural estimator for the mean μ of a stationary process is the empirical mean.

$$\mu = \frac{1}{n}\sum_{t=1}^{n} Y_t. \tag{2.6}$$

Similarly, with one realization, we use the following estimators using the mean estimate for empirical autocovariance function and autocorrelation at lag k is given by:

$$\hat{\gamma}_k = \frac{1}{n}\sum_{t=1}^{n-k} (Y_t - \mu)(Y_{t+k} - \mu) \quad \forall\ k = 0, 1, \ldots, \tag{2.7}$$

and

$$\hat{\rho}_k = \frac{\hat{\gamma}_k}{\hat{\gamma}_0} \quad \forall\ k = 0, 1, \ldots, \tag{2.8}$$

where $\{Y(t), t = 0, 1, 2, \ldots\}$ is the training series of size n with mean μ. Durbin, 1960 gave a systematic estimator for the PACF, starting with $\hat{\phi}_{11} = \hat{\rho}_1$ and the other terms are defined

iteratively as follows:

$$\hat{\phi}_{k+1,k+1} = \frac{\hat{\rho}_{k+1} - \sum_{j=1}^{k} \hat{\phi}_{kj}\hat{\rho}_{k+1-j}}{1 - \sum_{j=1}^{k} \hat{\phi}_{kj}\hat{\rho}_{j}}, \tag{2.9}$$

and

$$\hat{\phi}_{k+1,j} = \hat{\phi}_{k,j} - \hat{\phi}_{k+1,k+1}\hat{\phi}_{k,k+1-j}. \tag{2.10}$$

The sample ACF plot is important in identifying the model type to fit a time series of length n, as discussed by G. E. Box et al., 1970. Since ACF is symmetrical about lag zero, it is only required to plot the sample ACF for positive lags, from lag one onwards to a maximum lag of about $\frac{n}{4}$ (Brockwell & Davis, 2002). The sample PACF graphic aids in determining an AR process' maximum order. The methods for calculating ACF and PACF for ARMA models are described in K.W. Hipel, 1994. We shall demonstrate the use of these plots for our practical dataset in Chapter 4 and in Chapter 7.

2.4.3 White noise processes

A process $\{a_t\}$ is called a white noise process if it is a sequence of uncorrelated random variables from a fixed distribution with constant mean $\mathbb{E}(a_t) = \mu_a$, usually assumed to be 0, constant variance $\mathbb{V}ar(a_t) = \sigma_a^2$ and $\gamma_k = \mathbb{C}ov(a_t, a_{t+k}) = 0$ for all $k \neq 0$. By definition, it immediately follows that a white noise process $\{a_t\}$ is stationary with the autocovariance function (Wei, 2006):

$$\gamma_k = \begin{cases} \sigma_a^2 & \text{if } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases} \tag{2.11}$$

$$\rho_k = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases} \tag{2.12}$$

$$\phi_{kk} = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases} \tag{2.13}$$

A white process is Gaussian if its joint distribution is normal. In the following discussion, unless mentioned otherwise, $\{a_t\}$ is always referred to as a zero-mean Gaussian white noise process.

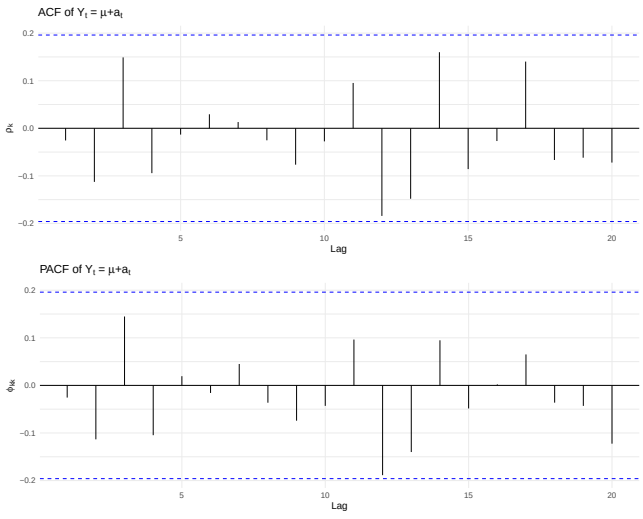


Figure 2.2 – ACF and PACF of a white noise processes: $Y_t = \mu + a_t$.

2.5 Financial time series forecasting: State of the art

Financial time series forecasting, particularly asset price prediction, is a widely explored area, focusing primarily on predicting the next movement of underlying assets. Although various subtopics exist, such as stock price, index, forex, commodity, bond, volatility, and cryptocurrency forecasting, they share similar underlying dynamics. Research in this domain typically falls into price prediction and price movement (trend) prediction. While price forecasting is fundamentally a regression problem, the emphasis is often placed on correctly identifying directional movements rather than precise price prediction. Consequently, trend prediction, which involves forecasting price direction changes, is considered more significant and often treated as a classification issue. Some studies focus solely on binary movement prediction (up or down), while others consider neutral movements, presenting a three-class classification problem.

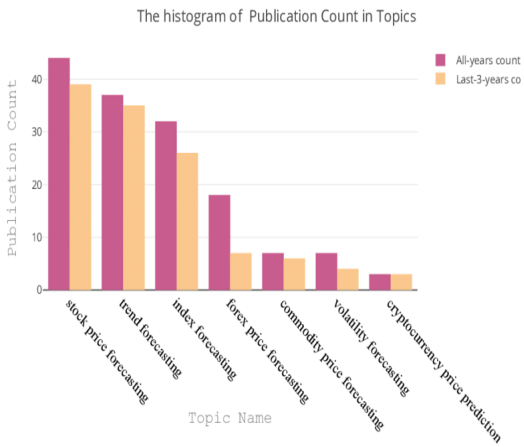


Figure 2.3 – Histogram of number publications in articles (Sezer et al., 2020).

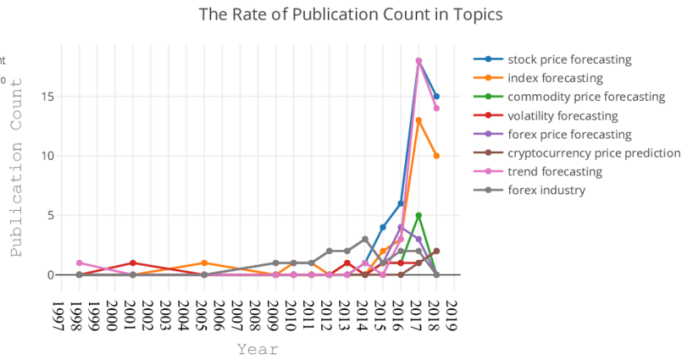


Figure 2.4 – Rate of publication count in topics (Sezer et al., 2020).

Figure 2.3 overviews the asset classes researchers target when developing forecasting models. Notably, there is a significant emphasis on predictions related to the stock market. The main focus areas include forecasting stock prices, trends, and indexes, accounting for more than 70% of the research. Specifically, there are 46 publications on stock price forecasting, 38 on trend forecasting, and 33 on index forecasting. There is also a growing interest in other areas, such as foreign exchange prediction, volatility forecasting, and cryptocurrency forecasting, although there have been fewer publications in these areas. Figure 2.4 shows the temporal trend in publication counts across various implementation domains.

2.5.1 Financial time series and their characteristics

Financial time series analysis focuses on the theory and practice of valuing assets over time, combining empirical observation with theoretical underpinnings. Unlike other time-series analyses, financial time-series analysis is characterized by inherent uncertainty, particularly regarding asset volatility. Various definitions of asset volatility exist, adding complexity to the analysis, and the volatility of the stock return series may not be readily apparent. As a result, statistical theory and methods play a crucial role in navigating this uncertainty (M.-Y. Chen, 2013).

Asset returns

Financial research often prioritizes asset returns over asset prices for two critical reasons outlined by Campbell et al., 1998. Firstly, asset returns offer a comprehensive and scale-free assessment of investment potential, making them more accessible to average investors. Secondly, return series exhibit more desirable statistical properties than price series, simplifying analysis. Asset returns can be defined in various ways, typically about the asset's price P_t at a given time index t. Throughout the thesis, we will explore several definitions of return, assuming the absence of dividends in the interim (Tsay, 2010).

One-Period Simple Return

Holding the asset for one period from date $t-1$ to date t would result in a simple gross return:

$$1 + R_t = \frac{P_t}{P_{t-1}} \quad \text{or} \quad P_t = P_{t-1}(1 + R_t). \tag{2.14}$$

The one-period simple net return or simple return is as follows:

$$R_t = \frac{P_t}{P_{t-1}} - 1 = \frac{P_t - P_{t-1}}{P_{t-1}}. \tag{2.15}$$

Multiperiod Simple Return

Holding the asset for k periods between dates $t-k$ and t gives a k-period simple gross return:

$$\begin{aligned} 1 + R_t[k] &= \frac{P_t}{P_{t-k}} = \frac{P_t}{P_{t-1}} \times \frac{P_{t-1}}{P_{t-2}} \times \cdots \times \frac{P_{t-k+1}}{P_{t-k}} \\ &= (1 + R_t)(1 + R_{t-1}) \cdots (1 + R_{t-k+1}) \\ &= \prod_{j=0}^{k-1} (1 + R_{t-j}). \end{aligned}$$

As a result, the k-period simple gross return is simply the sum of the k one-period simple gross returns. This is referred to as a compound return. The simple net return over k-periods is:

$$R_t[k] = \frac{P_t - P_{t-k}}{P_{t-k}}.$$

Real-time intervals are essential when discussing and comparing returns (e.g., monthly or annual). When the time interval is not specified, it is presumed to be one year. The annualized (average) return on an asset held for k years is defined as:

$$\text{Annualized}\{R_t[k]\} = \left[\prod_{j=0}^{k-1} (1 + R_{t-j}) \right]^{1/k} - 1.$$

This is a geometric mean of the k one-period simple gross returns involved and can be computed by:

$$\text{Annualized}\{R_t[k]\} = \exp\left[\frac{1}{k} \sum_{j=0}^{k-1} \ln\big(1 + R_{t-j}\big) \right] - 1,$$

where $\exp(x)$ denotes the exponential function and $\ln(x)$ is the natural logarithm of the positive number x. Because arithmetic average is easier to compute than geometric mean, and one-period returns are typically small, a first-order Taylor expansion can be used to estimate the annualized return, giving:

$$\text{Annualized}\{\mathrm{R}_t[k]\} \approx \frac{1}{k}\sum_{j=0}^{k-1}\mathrm{R}_{t-j}. \tag{2.16}$$

Accuracy of the approximation in Eq. (2.16) may not be sufficient in some applications.

Continuous Compounding

In general, the net asset value A of continuous compounding is

$$\mathrm{A} = \mathrm{C}\ \exp(r \times n), \tag{2.17}$$

where r is the interest rate per annul, C is the initial capital, and n is the number of years. From Eq. (2.17), we have

$$\mathrm{C} = \mathrm{A}\ \exp(-r \times n), \tag{2.18}$$

which is the *present value* of an asset that will be worth A dollars n years from now, assuming a r per annul continuously compounded interest rate.

Continuously Compounded Return

The natural logarithm of the simple gross return of an asset is called the continuously compounded return or log *return* :

$$r_t = \ln(1+\mathrm{R}_t) = \ln\frac{\mathrm{P}_t}{\mathrm{P}_{t-1}} = p_t - p_{t-1}, \tag{2.19}$$

where $p_t = ln(\mathrm{P}_t)$. Continuously compounded returns r_t enjoy some advantages over the simple net returns R_t. First, consider multiperiod returns. We have:

$$\begin{aligned} r_t[k] &= \ln(1+\mathrm{R}_t[k]) = \ln[(1+\mathrm{R}_t)(1+\mathrm{R}_{t-1})\ldots(1+\mathrm{R}_{t-k+1})] \\ &= \ln(1+\mathrm{R}_t) + \ln(1+\mathrm{R}_{t-1}) + \cdots + \ln(1+\mathrm{R}_{t-k+1}) \\ &= r_t + r_{t-1} + \cdots + r_{t-k+1}. \end{aligned}$$

As a consequence, the continuously compounded multiperiod return is like the sum of the continuously compounded one-period returns. Second, log returns have more tractable statistical properties.

Portfolio Return

The weighted average of the simple net returns of the assets engaged in a portfolio of N assets is the simple net return of the portfolio as a whole, where the weight on each asset equals the percentage of the portfolio's value invested in that asset. Let p be a portfolio in which asset i weights w_i. Then, at time t, the simple return of p is:

$$\mathrm{R}_{p,t} = \sum_{i=1}^{\mathrm{N}} w_i \mathrm{R}_{it}, \quad \text{where } \mathrm{R}_{it} \text{ is the simple return of asset } i.$$

A portfolio's continuously compounded returns, on the other hand, have little of the above useful property. If all of the simple returns R_{it} are small, we get $r_{p,t} \approx \sum_{i=1}^{N} w_i r_{it}$, where $r_{p,t}$ is the portfolio's continuously compounded return at time t. Portfolio results are frequently studied using this approximation.

Dividend Payment

We must change the definitions of asset returns if an asset pays dividends frequently. Let D_t be the dividend payment of an asset between dates $t-1$ and t and P_t be the price of the asset at the end of period t. Thus, the dividend is not included in P_t. Then, at time t, the simple net return and continuously compounded return are (Tsay, 2010):

$$R_t = \frac{P_t + D_t}{P_{t-1}} - 1, \quad r_t = \ln(P_t + D_t) - \ln(P_{t-1}).$$

Excess Return

The difference between the asset's return and the return on some reference asset at time t is the asset's excess return. The reference asset is frequently assumed to be riskless, such as the return on a short-term United States (U.S.) Treasury bill. The asset's simple excess return and log excess return are therefore defined as follows:

$$Z_t = R_t - R_{0t}, \quad z_t = r_t - r_{0t}, \tag{2.20}$$

where R_{0t} and r_{0t} are the reference asset's simple and log returns, respectively. The payoff on an arbitrage portfolio that goes long in one asset and short in the other with no net initial investment is referred to as the excess return in finance.

2.6 Time series forecasting using stochastic models (ARIMA)

2.6.1 Introduction

The introduction to this section emphasizes the importance of selecting an appropriate model to reflect the underlying structure of the series for future forecasting. Linear and non-linear time series models are discussed, focusing on ARIMA models developed by G. E. Box et al., 1970. ARIMA models are versatile and capable of handling both stationary and non-stationary series. The Box-Jenkins methodology for ARIMA model implementation involves three phases: identification, estimation, and diagnostic or validation. The section also mentions the significance of understanding the phases of the process for effective forecasting. Linear ARIMA and non-linear GARCH stochastic time series models, along with their characteristics, will be elaborated upon in this section, laying the groundwork for subsequent chapters covering other time series forecasting models (Adhikari & Agrawal, 2013; Cochrane, 1997; Ettayyebi & El Himdi, 2018b; Galbraith, Zinde-Walsh, et al., 2001; Hamzaçebi, 2008; V. Kumar, 2016; Park, 1999; Perrelli, 2001; Tong, 1983; G. Zhang, 2003, 2007) .

2.6.2 ARMA models

An ARMA(p,q) model is a mix of AR(p) and MA(q) models that can be used to model univariate time series.

The AR representations of time series processes

The future value of a variable in an AR(p) model is considered to be a linear combination of p past observations, a random error, and a constant term. The AR(p) model can be stated mathematically (K.W. Hipel, 1994; J. Lee, 2018):

$$Y_t = c + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t = c + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \varepsilon_t, \tag{2.21}$$

Y_t represents the actual value at period t, and ε_t denotes the random error, assumed to follow a **white noise process** characterized by uncorrelated variables with zero mean and constant variance σ^2. The model incorporates parameters φ_i $(i = 1, 2, \ldots, p)$ and a constant c, with the model order represented by the integer constant p. Typically, the constant term may be omitted for simplicity. Estimating parameters for an AR process often involves formulating *Yule-Walker* equations, which aid in computing the PACF and determining efficient moment estimates for the AR model parameters. By introducing the backshift operator $B^j x_t = x_{t-j}$, we can write equation (2.21) in the compact form:

$$\varphi_p(B) Y_t = c + \varepsilon_t, \tag{2.22}$$

where $\varphi_p(B) = 1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p = 1 - \sum_{i=1}^{p} \varphi_i B^i$.

The MA representations of time series processes

An MA(q) model utilizes previous errors as the explanatory variables, similar to how a AR(p) model regresses against the series' past values. The MA(q) model is defined as follows (Cochrane, 1997; K.W. Hipel, 1994; J. Lee, 2018):

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} = \mu + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t, \tag{2.23}$$

or, equivalently,

$$Y_t = \mu + \theta_q(B) \varepsilon_t, \tag{2.24}$$

where $\theta_q(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q = 1 + \sum_{j=1}^{q} \theta_j B^j$, and μ denotes the series mean and $\theta_j (j = 1, 2, \cdots, q)$ are the model parameters with q representing the model order. It assumes random shocks follow a white noise process with zero mean and constant variance σ^2. The MA model involves regressing current observations against past random shocks, posing a greater challenge in fitting compared to AR models (Adhikari & Agrawal, 2013).

☞ The AR and MA models can be efficiently mixed to generate the ARMA models, a general and consequential class of time series models. Mathematically an ARMA(p, q) model is represented as :

$$\begin{aligned} Y_t &= c + \varphi_1 Y_{t-1} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \\ &= c + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}. \end{aligned} \tag{2.25}$$

Or,

$$\varphi_p(B) Y_t = \theta_q(B) \varepsilon_t. \tag{2.26}$$

Here, the model orders p and q refer to p autoregressive and q moving average terms. An essential property of the AR(p) process is **invertibility**, i.e. an AR(p) process can always be written in terms of an AR(∞) process with $\sum_{i=0}^{\infty} |\varphi_i| < \infty$. Whereas for an MA($q$) process to be invertible, all the roots of the equation $\theta_q(B) = 0$ must lie outside the unit circle. This condition is known as the *Invertibility Condition* for an MA process.

2.6.3 Stationarity analysis

When an AR(p) process is represented as $\varphi(B)Y_t = \varepsilon_t$, $\varphi(B) = 0$ is known as the characteristic equation for the process. G. E. Box et al., 1970 proved that a necessary and sufficient condition for the AR(p) process to be stationary is that all the roots of the characteristic equation must fall outside the unit circle. Another simple algorithm for determining the stationarity of an AR process is mentioned K.W. Hipel, 1994, known as the Pagano, 1972 algorithm. For example, as shown in J. Lee, 2018, the AR(1) model $Y_t = c + \varphi_1 Y_{t-1} + \varepsilon_t$ is stationary when $|\varphi_1| < 1$, with a constant mean $\mu = \frac{c}{1-\varphi_1}$ and constant variance $\gamma_0 = \frac{\sigma^2}{1-\varphi_1^2}$. An MA(q) process is always stationary, depending on the MA parameter values. The conditions regarding stationarity and invertibility that relate to AR and MA processes also apply to an ARMA process. If all the roots of the characteristic equation $\varphi(B) = 0$ are outside the unit circle, the ARMA(p,q) process is stationary. The ARMA(p,q) process is invertible and can be expressed as a pure AR process if all the roots of the lag equation $\theta(B) = 0$ lie outside the unit circle.

2.6.4 ARIMA models

The ARMA models are limited to stationary time series data, yet many real-world time series exhibit non-stationary behaviour, particularly those in socio-economic and business domains. The ARIMA model, an extension of ARMA, is introduced to address this. ARIMA incorporates differencing to transform non-stationary series into stationary ones. Widely used in econometrics for time series forecasting, ARIMA has been a standard approach for a long time, despite its limitations in specific scenarios (Faraway & Chatfield, 1998; Flaherty & Lombardo, 2000; Hamzaçebi, 2008). The ARIMA(p,d,q) model, expressed using lag polynomials, offers a flexible framework for modelling various time series phenomena:

$$\varphi_p(B)(1-B)^d Y_t = \theta_q(B)\varepsilon_t, \; i.e. \left(1 - \sum_{i=1}^{p} \varphi_i B^i\right)(1-B)^d Y_t = \left(1 + \sum_{j=1}^{q} \theta_j B^j\right)\varepsilon_t, \quad (2.27)$$

where

$$\varphi_p(B) = (1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p) \text{ and } \theta_q(B) = (1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q).$$

- The orders of the model's autoregressive, integrated, and moving average parts are denoted by p, d, and q, respectively, which are all integers higher than or equal to zero.
- The integer d controls the level of differencing. Generally, $d = 1$ is enough in most cases. When $d = 0$, it reduces to an ARMA(p,q) model.
- ARIMA($p,0,0$) represents the AR(p) model, while ARIMA($0,0,q$) represents the MA(q) model.

- ⊛ ARIMA$(0,1,0)$, i.e. $Y_t = Y_{t-1} + \varepsilon_t$ is a special one known as the *Random Walk* model (Cochrane, 1997; J. Lee, 2018; G. Zhang, 2003). Non-stationary data, such as economic and stock price series, are commonly used (ARIMA(0,1,0)).

A valid generalization of ARIMA models is the Autoregressive Fractionally Integrated Moving Average (ARFIMA) model, which allows noninteger values of the differencing parameter d. ARFIMA can model time series with a large memory (Galbraith, Zinde-Walsh, et al., 2001). In this model, the term $(1-B)^d$ is to be expanded using the general binomial theorem. Researchers have made a variety of contributions to the estimate of the general ARFIMA parameters.

2.6.5 Some nonlinear time series models

We have already discussed linear time series models, but it is essential to consider nonlinear models for more comprehensive analysis and forecasting. The pioneering work of Campbell et al., 1998 has emphasized the significance of exploring nonlinearities in time series. Nonlinear models can generally be divided into two categories: those with nonlinearities in the mean and those with nonlinearities in the variance (heteroskedastic). To illustrate this, we will present two nonlinear models from Perrelli, 2001:

- ♣ *NMA Model:* $Y_t = \varepsilon_t + \alpha\varepsilon_{t-1}^2$. This model has a nonlinear, meanwhile maintaining a linear variance.
- ♣ Engle, 1982 *ARCH Model*: $Y_t = \varepsilon_t + \alpha\sqrt{\varepsilon_t^2}$. In this model, the variance is nonlinear (heteroskedastic), while the mean remains linear. Variants such as GARCH and EGARCH build upon this model.
- ♣ Bilinear Model: can be expressed as $Y_t = \varphi Y_{t-1} + \theta\varepsilon_{t-1} + \gamma Y_{t-1}\varepsilon_{t-1} + \varepsilon_t$. In this model, the presence of the product term $\gamma Y_{t-1}\varepsilon_{t-1}$ introduces nonlinearity into both the mean and variance. This provides a flexible framework for capturing various types of nonlinear dependencies and interactions within the time series data. The Bilinear Model is particularly useful in cases where linear models fail to capture the underlying dynamics due to the interaction between past values and past shocks (C. W. J. Granger & Andersen, 1978).

Introduction to ARCH and GARCH modeling

The ARIMA model may struggle when faced with heteroskedastic [1] behavior within the process, which can decrease efficiency, especially in predictive capacity. Engle, 1982 and later Engle and Bollerslev, 1986 developed methodologies to address this issue, ultimately leading to the creation of ARCH and GARCH models (Perrelli, 2001). These models aim to account for heteroskedasticity, where the variance of the series changes over time. It is important to note that these models assume the series to be stationary. In contrast to classical time series models like ARIMA (7.1), which assume a constant variance, financial series often exhibit varying conditional variances over time. This variability needs to be appropriately captured by the residual models. For example, if ε represents white noise, but its square shows autocorrelation, ε becomes a sequence of uncorrelated but not independent random variables, indicating a deviation from Gaussian white noise (Ettayyebi, 2017). It is said that ε_t following an ARCH(p) it follows (Falloul & Mansouri, 2014; Hansen & Lunde, 2005; Hossain & Nasser, 2008; Perrelli, 2001):

1. EMQ $= \mathbb{E}[Y_{n+h} - \hat{Y}_n(h)]^2$.

$$\varepsilon_t = \sigma_t z_t, \tag{2.28}$$

$$z_t \sim \text{i.i.d.}\, \mathcal{N}(0,1) \text{ independent of the past } \varepsilon_{t-1}, \varepsilon_{t-2}, \cdots \forall\, t$$

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2. \tag{2.29}$$

where $\omega > 0,\ \alpha_i \geq 0,\ i = 1, \ldots, p,$ and

$$\sum_{i}^{p} \alpha_i < 1.$$

This last condition ensures the stationarity of ε_t. But observe that the conditional variances $\sigma_t^2, \sigma_{t-1}^2, \ldots,$ are themselves summaries of past variability, hence the idea of economic models where σ_t^2 is a function of $\varepsilon_t^2, \varepsilon_{t-1}^2, \ldots,$ past and also past conditional variances: $\sigma_t^2, \sigma_{t-1}^2, \ldots,$. This is the point of view of the GARCH models. We say that ε_t follows a GARCH(p, q) if it follows:

$$\varepsilon_t = \sigma_t z_t.$$

$$z_t \sim \text{i.i.d.}\, \mathcal{N}(0,1) \text{ independent of the past } \varepsilon_{t-1}, \varepsilon_{t-2}, \ldots \forall\, t$$

Bollerslev, 1986 presented an essential extension of the ARCH model by replacing the AR(p) representation in equation (2.29) with an ARMA(p, q) formulation:

$$\begin{aligned} \sigma_t^2 &= \omega + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_q \sigma_{t-q}^2 \\ &= \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2. \end{aligned} \tag{2.30}$$

Where $\omega > 0,\ \alpha_i, \beta_j \geq 0,\ i = 1, \ldots, p, j = 1, \ldots, q$ and

$$\sum_{i}^{p} \alpha_i + \sum_{j}^{q} \beta_j < 1.$$

The model in equation (2.30), together with equations (2.28) and (2.30), is known as the generalized ARCH or GARCH(p,q) model. A more parsimonious GARCH representation of a high-order ARCH model may exist, making it considerably easier to identify and estimate. A GARCH$(1,1)$ model with only three parameters in the conditional variance equation is usually sufficient to establish a satisfactory model fit for financial time series. Indeed, Hansen and Lunde, 2005 demonstrated that finding a volatility model that performs as well as the simple GARCH$(1,1)$ is challenging. As a result, GARCH$(1,1)$ is used in this thesis to analyze the fit, test ARCH/GARCH effects, and perform diagnostic checks for model adequacy and estimates (Enders, 2004).

Finite mixture of ARMA-GARCH for time series forecasting

Linear time series models like AR and ARMA can be integrated with GARCH to model the dynamics of stock indexes and their volatilities. The mixture of the ARMA − GARCH model is similar to the AR-GARCH model proposed by Tang et al., 2003. Specifically, each

component of the mixture model can be denoted as a normal ARMA series (Hossain & Nasser, 2008; Hossain & Nasser, 2011):

$$y_{t,k} = \sum_{r=1}^{R} b_{rk} y_{t-r,k} + \sum_{s=1}^{S} a_{sk} \varepsilon_{t-s,k} + \varepsilon_{t,k}.$$

Furthermore, each residual term $\varepsilon_{t,k}$ is assumed Gaussian white noise with variance denoted by the GARCH model:

$$\sigma_{t,k}^2 = \omega_k + \sum_{i=1}^{p} \alpha_{ik} \varepsilon_{t-i,k}^2 + \sum_{j=1}^{q} \beta_{jk} \sigma_{t-j,k}^2,$$

where $\alpha_{ik} > 0$ for $i = 1, \ldots, p$ and $\beta_{jk} > 0$ for $j = 1, \ldots, q$. The finite mixture of the ARMA − GARCH model can be expressed mathematically as a K-component Gaussian mixture model:

$$P(y_t) = \sum_{k=1}^{K} \delta_k G(y_t; \hat{y}_{t,k}, \sigma_{t,k}^2),$$

$$\hat{y}_{t,k} = \sum_{r=1}^{R} b_{rk} y_{t-r,k} + \sum_{s=1}^{S} a_{sk} \varepsilon_{t-s,k},$$

where $\delta_k > 0$ and $\sum_{k=1}^{K} \delta_k = 1$. One-step ahead prediction can be done after the model has been learned by taking the expectation of y_t as follows:

$$\mathbb{E}(y_t) = \delta_1 \hat{y}_{t,1} + \cdots + \delta_K \hat{y}_{t,K}.$$

The mixture model is learned using an extended expectation-maximization approach, see Tang et al., 2003 for details.

2.6.6 Box-Jenkins methodology

After discussing different time series models, the next important step is to select a suitable model that can accurately generate forecasts based on historical data patterns and determine the optimal model order. G. E. Box et al., 1970 introduced a practical method called the Box-Jenkins methodology for constructing ARIMA or SARIMA models that best fit a given time series while following the principle of parsimony. Their methodology is highly significant in time series analysis and forecasting (Flaherty & Lombardo, 2000; G. Zhang, 2003). The Box-Jenkins methodology does not assume any specific pattern in the historical data but instead employs a three-step iterative approach (see diagram 2.5 below) - model identification, parameter estimation, and diagnostic verification - to choose the most parsimonious ARIMA model from a wide range of possibilities. This iterative process continues until an appropriate model is identified. Then, the model is used to forecast future values of the time series.

Model identification

In time series analysis, the **first step** involves plotting the data to assess trends, seasonality, outliers, and other non-stationary phenomena. This examination often leads to considering data transformations, such as variance-stabilizing transformations and differencing. Logarithmic transformations are suitable for addressing nonconstant variance, while Box-Cox's power transformation (Sakia, 1992) is frequently used to stabilize

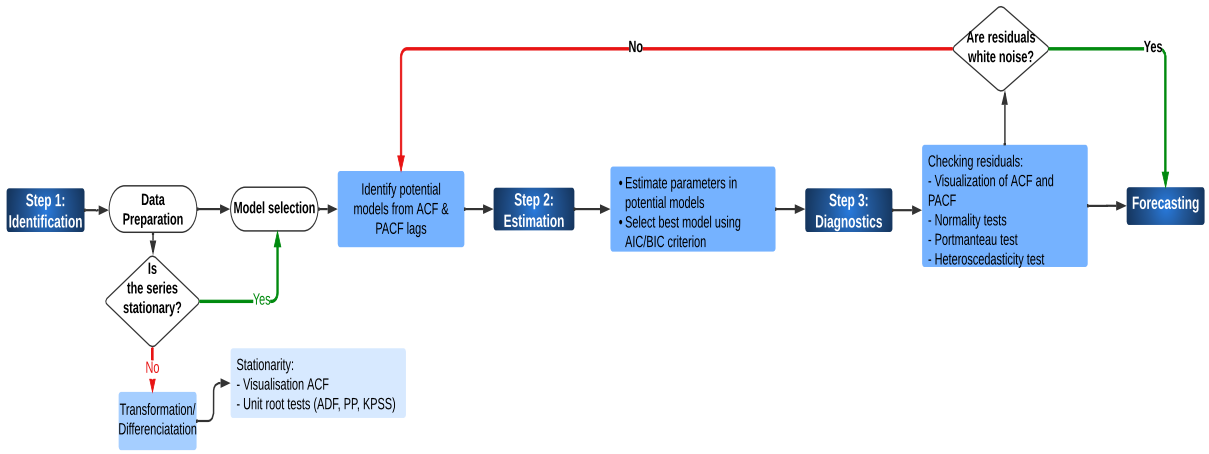


Figure 2.5 – Box-Jenkins methodology for optimal model selection.

variance. The **second step** involves computing and inspecting the sample ACF and PACF of the original series to determine the appropriate degree of differencing for stationarity. Then, the sample ACF and PACF of the transformed and differenced series are computed and analyzed to identify the orders of AR and MA components. Typically, these orders (denoted by p and q) are limited to 3 or less. Theoretical guidelines for selecting p and q in ARMA models are summarized in Tables (2.1) and (2.2), providing insights into ACF and PACF's expected behavior.

Table 2.1 – Characteristics of theoretical ACF and PACF for stationary processes (Wei, 2006).

Process	**ACF**	**PACF**
AR(p)	Tails off * as exponential decay or damped sine wave.	Cuts off after lag p.
MA(q)	Cuts off after lag q.	Tails off as exponential decay or damped sine wave.
ARMA(p,q)	Tails off * after lag $(q-p)$.	Tails off after lag $(p-q)$.

* Tails off as exponential decay or damped sine wave.

The **third step** is to test the deterministic trend term μ when $d > 0$. Other widely used measures for model identification are *Akaike Information Criterion* (AIC) as defined in the following proposition:

Proposition 1 (Bozdogan, 1987)**.** *Consider a collection of competing models* $\{M_p : p = 1,2,\ldots,n\}$*, each indexed by* $p = 1,2,\ldots,n$*. The criterion is given by:*

$$AIC(p) = -2\log L(\hat{\theta}_p) + 2p, \tag{2.31}$$

where $\hat{\theta}_p$ *represents the estimated parameters for model* M_p*. This criterion is designed to minimize the discrepancy between model complexity and goodness of fit, making it a practical estimator of either twice the negentropy,* $2\mathbb{E}[I(0^*;\theta)]$*, or negative twice the expected log-likelihood,* $-2\mathbb{E}[\log f(Y|\theta_i)]$*, concerning the true distribution, as determined by the method of maximum likelihood (Akaike, 1998).*

However, *Bayesian Information Criterion (*BIC*)* which are defined below (Faraway & Chatfield, 1998; Kihoro et al., 2004):

$$\mathrm{BIC}(p) = n\ln\left(\frac{\hat{\sigma}_e^2}{n}\right) + p + p\ln(n), \tag{2.32}$$

where n is the number of effective observations used to fit the model, p is the number of parameters in the model, and $\hat{\sigma}_e^2$ is the sum of sample squared residuals. The number of model parameters that minimize either AIC or BIC determines the best model order. Other similar criteria for optimal model identification have also been widely discussed in the literature.

Table 2.2 – Properties of the ACF and PACF for various ARMA models (Wei, 2006).

Model	**ACF**	**PACF**
$(1,d,0)$ AR(1)	Exponential or oscillatory decay.	$\phi_{kk} = 0$ for $k > 1$.
$(2,d,0)$ AR(2)	Exponential or sine wave decay.	$\phi_{kk} = 0$ for $k > 2$.
$(p,d,0)$ AR(p)	Exponential and /or sine wave decay.	$\phi_{kk} = 0$ for $k > p$.
$(0,d,1)$ MA(1)	$\rho_k = 0$ for $k > 1$.	Dominated by damped exponential.
$(0,d,2)$ MA(2)	$\rho_k = 0$ for $k > 2$.	Dominated by damped exponential or sine wave.
$(0,d,q)$ MA(q)	$\rho_k = 0$ for $k > q$.	Dominated by linear combination of damped exponentials and / or sine waves.
$(1,d,1)$ ARMA$(1,1)$	Tails off. Exponential decay from lag 1.	Tails off. Dominated by exponential decay from lag 1.
(p,d,q) ARMA(p,q)	Tails off after $q-p$ lags. Exponential and/or sine wave decay after $q-p$ lags.	Tails off after $p-q$ lags. Dominated by damped exponentials and/or sine waves after $p-q$ lags.

Parameters estimation

After identifying a tentative model, the next step is to estimate its parameters and then verify its adequacy for the series. It is common for a given series to be represented by multiple models. Therefore, various model selection criteria are employed in time series simulation models after discussing diagnostic checks. Several parameter estimation methods, such as the Method of Moments (MM), Maximum Likelihood Method and Least Squares, are used for a tentatively identified model. Maximum Likelihood Estimators (MLE) are preferred when the series is assumed to follow a Gaussian stationary process (i.e., $\varepsilon_t \sim \mathcal{N}(0,\sigma^2)$ i.i.d.). For MLE, initial values must be provided as inputs to the optimization algorithm. In our study, we consistently use the method of **Conditional Maxi-**

mum Likelihood Estimation (CMLE) (Rochester et al., 1956; Wei, 2006). We start by recalling the general form of a stationary ARMA(p,q) model:

$$\varphi_p(\mathrm{B})\mathrm{Y}_t = \theta_q(\mathrm{B})\varepsilon_t. \tag{2.33}$$

With ε_t is white noise. Let $\theta = (\varphi_1,\ldots,\varphi_p,\theta_1,\ldots,\theta_q,\sigma^2)$ be the vector of the model parameters. Suppose we observed a sample of size n $(y_1,y_2,\ldots,y_n)$. The approach will be to calculate the joint probability density, which could be considered as the probability of that particular sample being observed.

$$f_{\mathrm{Y}_n,\mathrm{Y}_{n-1},\ldots,\mathrm{Y}_1}(y_n,y_{n-1},\ldots,y_1;\theta). \tag{2.34}$$

The MLE of θ is the value for which this sample is most likely to have been observed, i.e., θ maximizes (2.34). This approach requires specifying a particular distribution for the ε white noise process. Generally, we assume that ε is Gaussian white noise:

$$\varepsilon_t \sim \mathcal{N}(0,\sigma^2) \text{ i.i.d.} \tag{2.35}$$

The $(p+1)$th observation is:

$$\mathrm{Y}_{p+1} = \varphi_1\mathrm{Y}_p + \varphi_2\mathrm{Y}_{p-1},\cdots,\varphi_p\mathrm{Y}_1 + \varepsilon_{p+1} + \theta_1\varepsilon_p + \theta_2\varepsilon_{p-2} + \cdots + \theta_q\varepsilon_{p-q+1}.$$

Conditional on: $\mathrm{Y}_1 = y_1$, $\mathrm{Y}_2 = y_2,\cdots,\mathrm{Y}_p = y_p$ and $\varepsilon_p = \varepsilon_{p-1} = \cdots = \varepsilon_{p-q+1} = 0$. We have :

$$\mathrm{Y}_t \backsim \mathcal{N}((\varphi_1\mathrm{Y}_p + \varphi_2\mathrm{Y}_{p-1},\ldots,\varphi_p\mathrm{Y}_1),\sigma^2). \tag{2.36}$$

Then, the logarithm of the conditional likelihood function calculated from $t = p+1,\ldots,n$ is:

$$\begin{aligned}\mathcal{L}^*(\theta) &= \log f(y_n,y_{n-1},\ldots,y_{p+1}/y_p,\ldots,y_1,\varepsilon_p = \varepsilon_{p-1} = \cdots = \varepsilon_{p-q+1} = 0;\theta)\\ &= -\frac{n-p}{2}\log(2\pi) - \frac{n-p}{2}\log\left(\sigma^2\right) - \sum_{t=p+1}^{n}\frac{\varepsilon_t^2}{2\sigma^2},\end{aligned} \tag{2.37}$$

where the sequence $\{\varepsilon_{p+1},\varepsilon_{p+2},\ldots,\varepsilon_n\}$ can be calculated from $\{y_1,y_2,\ldots,y_n\}$ by iterating over $t = p+1,p+2,\ldots,n$.

$$\varepsilon_t = \mathrm{Y}_t - \varphi_1\mathrm{Y}_{t-1} - \varphi_2\mathrm{Y}_{t-2} - \cdots - \varphi_p\mathrm{Y}_{t-p} - \theta_1\varepsilon_{t-1} - \theta_2\varepsilon_{t-2} - \cdots - \theta_q\varepsilon_{t-q}.$$

The estimator θ is given by:

$$\hat{\theta} = \underset{\theta}{argmin}\,\mathcal{L}^*(\theta). \tag{2.38}$$

It is important to note that if we realize size n to estimate an ARMA(p,q) model by the CMLE method, we will only use $n-p$ observations of this realization.

Diagnostic checking

The time series model development process is iterative and begins with model identification and parameter estimation. Once the parameters are estimated, the next crucial step is assessing the model's adequacy and ensuring the assumptions are satisfied. One of the fundamental assumptions is that the values of $\{\varepsilon_t\}$ represent white noise, which means they are uncorrelated random shocks with a zero mean and constant variance. In an estimated model, the residuals $\hat{\varepsilon}_t$'s are used to estimate these unobserved white noise

ε_t's values. Therefore, model diagnostic checking involves carefully examining the residual series $\{\hat{\varepsilon}_t\}$. Since these residuals result from parameter estimation, diagnostic checking typically occurs within the estimation phase of a time series package. The provided equation can be used to derive residuals for an ARMA(p,q) process:

$$\hat{\varepsilon}_t = Y_t - \sum_{i=1}^{p} \hat{\varphi}_i Y_{t-i} - \sum_{j=1}^{q} \hat{\theta}_j \hat{\varepsilon}_{t-j}. \tag{2.39}$$

If the specified model is adequate and where appropriate orders p and q are identified, the residues in the Eq. (2.39) should behave like a Gaussian white noise process with constant variance. To evaluate the normality of errors, it is recommended to construct a histogram of the standardized residuals $\frac{\hat{\varepsilon}_t}{\hat{\sigma}_\varepsilon}$ and compare it with a standard normal distribution. This can be done using the chi-square goodness-of-fit test or Tukey's simple five-number summary. Additionally, examining a plot of residuals can help determine the constancy of variance.
Further analysis involves calculating the sample ACF and PACF, or the inverse autocorrelation function (IACF), of residuals. This is done to detect discernible patterns and ensure that all patterns are statistically insignificant, typically within two standard deviations of $\alpha = 0.05$. Another valuable tool is the portmanteau lack-of-fit test, which scrutinizes the joint null hypothesis using all residual sample ACFs as a cohesive unit:

$$H_0 : \rho_1 = \rho_2 = \cdots = \rho_k = 0,$$

With the test statistic

$$Q = n(n+2) \sum_{k=1}^{K} (n-k)^{-1} \hat{\rho}_k^2. \tag{2.40}$$

This test statistic is the modified Q statistic originally proposed by G. E. Box and Pierce, 1970. Under the null hypothesis of model adequacy, Ljung and Box, 1978 show that the Q statistic approximately follows the $\chi^2(K-m)$ distribution, where $m = p+q$. This process of identification, estimation, and diagnosis is repeated several times until a satisfactory model is finally selected (see Figure 2.5). This model is then used to predict future values for the time series. In the next section, we briefly discuss the prediction of time series based on ARIMA models.

2.6.7 Forecasting

The ARIMA model class is used to predict future values of a time series. Consider the general class of the ARIMA(p,d,q) model:

$$\varphi_p(B)(1-B)^d Y_t = \theta_q(B)\varepsilon_t, \tag{2.41}$$

where $\varphi_p(B) = (1 - \varphi_1 B - \varphi_2 B^2 - \cdots - \varphi_p B^p)$ is a stationary AR operator, $\theta_q(B) = (1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q)$ is an invertible MA operator, and the process $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ $i.i.d.$ is a Gaussian white noise. For clarity, the deterministic trend parameter (constant) c is omitted, but no loss of generality results. Eq. (2.41) is one of the most commonly used models in forecasting applications. In the general case for this process with $d \neq 0$, the mean, variance, and autocorrelation function vary with time. A finite number of fixed parameters essentially determine the process's whole evolution. Our goal in forecasting is to produce an optimal forecast with no errors or the slightest possible error, leading us

to the mean error quadratic forecast [2] (Wei, 2006).
We can view the forecast as an estimation of parameters to calculate the RMSE for the optimal forecast. When the series is known up to time n, the optimal forecast for Y_{t+h} (with h being the forecast horizon) is determined by its conditional expectation based on the observations $(Y_1, Y_2, \ldots, Y_n)$. In other words:

$$\hat{Y}_n(h) = \mathbb{E}[Y_{n+h}|Y_n, Y_{n-1}, \ldots, Y_1]. \tag{2.42}$$

2.7 Overview of statistical learning theory

In the late 1960s, statistical learning theory emerged as a theoretical pursuit focused on the challenge of estimating functions from given data sets. However, it remained primarily theoretical until the mid-1990s when learning algorithms, like SVM, were developed based on these theoretical foundations. This transition elevated statistical learning theory from pure theory to a practical tool for developing algorithms that can estimate multidimensional functions. For a detailed theoretical overview, V. N. Vapnik, 1998, V. N. Vapnik, 1999 provides a comprehensive exposition, both with and without proofs. This section will explore a learning model and demonstrate its analysis within the broader statistical framework of minimizing expected loss with observed data. This framework includes practical applications such as **pattern recognition**, **regression estimation**, and **density estimation**, all of which are specific instances of this general model.

2.7.1 Function estimation model

Three components can be used to characterize the model of learning from examples:

1. A random vector generator x, that chooses randomly from a fixed but unknown distribution $P(x)$;
2. A supervisor that returns an output vector $y \in D \subseteq \mathbb{R}$ for every input vector $x \in X \subseteq \mathbb{R}^N$ where $D = \{-1, 1\}$ or $\{0, 1\}$ for binary classification problem and $D = \mathbb{R}$ for regression problem, according to a conditional distribution function [3] $P(y|x)$, also fixed but unknown;
3. A learning machine capable of implementing a set of functions $\{f(x, w),\ w \in \Lambda\}$ [4]. Here w is a parameter from the set Λ such that the value $w = w^*$ defines the specific function $f(x, w^*)$ in the set $f(x, w)$. Finding the required function means determining the corresponding value of the parameter $w \in \Lambda$. The set Λ, to which the parameter w belongs, is arbitrary: It can be a set of scalar quantities, a set of vectors, or a set of abstract elements (V. N. Vapnik, 1998).

The learning problem is to select the optimal function from a set of functions that best predicts the supervisor's response. The selection is based on l random i.i.d. observations chosen using the $P(x, y) = P(x)P(y|x)$ equation (V. N. Vapnik, 1999)

$$\text{Data:} \quad \{(x_1, y_1), \ldots, (x_l, y_l), x \in X, y \in Y, l \in [1, \ldots, N]\}. \tag{2.43}$$

2. The variances of the residuals of the variables examined are different.
3. This is the general case which includes a case where the supervisor uses a function $y = f(x)$.
4. We shall always omit the braces when writing a set of functions. A single function can be separated from a cluster of functions by whether or not the parameter a is fixed (V. N. Vapnik, 1998).

2.7.2 Problem of risk minimization

To choose the best available approximation to the supervisor's response, one measures the loss or discrepancy $\mathrm{L}(y, f(x,w))$ between the response y of the supervisor to a given input x and the response $f(x,w)$ provided by the learning machine. Consider the expected value of the loss, given by the *risk functional*:

$$\mathrm{R}(w) = \int \mathrm{L}(y, f(x,w)) d\mathrm{P}(x,y). \tag{2.44}$$

The goal is to find the function $f(x,w_0)$, which minimizes the risk functional $\mathrm{R}(w)$. That is to say, over the class of functions $f(x,w),\ w \in \Lambda$ in the situation where the joint probability distribution $\mathrm{P}(x,y)$ is unknown, and the only available information is contained in the training set (2.43) (V. N. Vapnik, 1999).

Three main learning problems

This is a clear and concise formulation of the learning problem, addressing several key issues. The three main issues are pattern recognition, regression estimation, and density estimation, which will be discussed in further detail.

The problem of pattern recognition: Let us start with the problem of learning how to recognize patterns. Formally, we want to estimate a function $f : \mathbb{R}^N \to \{\pm 1\}$ using input-output training data (2.43) (V. N. Vapnik, 1998). Consider the following *Loss-Function*:

$$\mathrm{L}(y, f(x,w)) = \begin{cases} 0\,, & \text{if } y = f(x,w) \\ 1\,, & \text{if } y \neq f(x,w). \end{cases} \tag{2.45}$$

For this Loss Function, the functional Eq. (2.44) provides the probability of classification error (i.e., when the answers y given by the supervisor and the answers given by indicator function $f(x,w)$ differ). The problem, therefore, is to find the function that minimizes the probability of classification errors when probability measure $\mathrm{P}(x,y)$ is unknown, but the data (2.43) are given (V. N. Vapnik, 1999).

The problem of regression estimation: Let the supervisor's answer $y \in \mathbb{R}$ be a real value, and let $f(x,w)$, $w \in \Lambda$, be a set of real functions which contains the *regression function*:

$$f(x,w_0) = \int y d\mathrm{P}(y|x).$$

It is known that if $f(x,w) \in \mathcal{L}_2(\mathrm{P})$, then the regression function is the one which minimizes the functional Eq. (2.44) with the following Loss-Function:

$$\mathrm{L}(y, f(x,w)) = (y - f(x,w))^2. \tag{2.46}$$

Thus, regression estimation involves minimizing the risk function Eq. (2.44) with the loss function Eq. (2.46) in the situation where the probability measure $\mathrm{P}(x,y)$ is unknown but the data (2.43) is given (V. N. Vapnik, 1999).

The problem of density estimation: Finally, consider the problem of density estimation from the set of densities $p(x,w),\ w \in \Lambda$. The following Function is used to solve this problem:

$$\mathrm{L}(p(x,w)) = -\log p(x,w). \tag{2.47}$$

It is known that the density minimizes the risk functional Eq. (2.44) with the Loss-Function Eq. (2.47). Thus, again, to estimate the density from the data, one has to minimize the

risk-functional under the condition where the corresponding probability measure $P(x)$ is unknown but i.i.d. data are given (V. N. Vapnik, 1999):

$$x_1, \ldots, x_n$$

The general setting of the learning problem: The following is a general description of the learning problem's setting. Let the probability measure $P(z)$ be defined on the space Z. Consider the set of functions $Q(z, w)$, $w \in \Lambda$. The goal is to minimize the risk of functional:

$$R(w) = \int Q(z, w) dP(z), \quad w \in \Lambda, \tag{2.48}$$

if probability measure $P(z)$ is unknown but an i.i.d. sample is given:

$$z_1, \ldots, z_l. \tag{2.49}$$

The learning problems considered above are particular cases of this general problem of *minimizing the risk functional* Eq. (2.48) based on empirical data (2.49), where z describes a pair (x, y) and $Q(z, w)$ is the specific loss function [For example, one of Eq. (2.45), (2.46), or (2.47)]. Below, we will describe the results to provide a general problem statement. To apply it to specific problems, the corresponding loss functions must be substituted in the formulas obtained (V. N. Vapnik, 1999).

Empirical risk minimization induction principle

In order to minimize the risk functional Eq. (2.48), for an unknown probability measure $P(z)$ the following induction principle is usually used. The *empirical risk* functional formed on the basis of the training set (2.49) replaces the predicted risk functional $R(w)$.

$$R_{emp}(w) = \frac{1}{l} \sum_{i=1}^{l} Q(z, w). \tag{2.50}$$

The principle is to approximate the function that minimizes risk Eq. (2.48) by the function that minimizes empirical risk Eq. (2.50). This principle is called the **Empirical Risk Minimization induction principle (ERM) principle** (V. N. Vapnik, 1999).

ERM principle and the classical methods

The ERM principle is quite general. The least-squares technique in regression estimation and the maximum likelihood method in density estimation are examples of traditional methods for solving a specific learning problem. Their realizations of the ERM principle for the specific loss functions are considered above. Indeed, to specify the regression problem, one introduces an $n+1$-dimensional variable $z = (x, y) = (x^1, \ldots, x^n, y)$ and uses the loss function Eq. (2.46). Using this loss function in Eq. (2.50), we obtain the function :

$$R_{emp}(w) = \frac{1}{l} \sum_{i=1}^{l} (y_i - f(x, w))^2.$$

This function should be reduced to zero to obtain the regression estimate (i.e., the least square method). Although the ERM principle is a general explanation of these traditional estimate problems, any theory based on it also applies to the classical approaches (V. N. Vapnik, 1999).

2.7.3 Vapnik-Chervonenkis (VC) Dimension

The VC *dimension*, defined as the highest number h of points that can be separated in all feasible ways using functions of the given class, is the most well-known capacity notion in VC theory. The following is an example of a VC bound:

$$R(w) \leqslant R_{emp}(w) + \phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right). \tag{2.51}$$

If $h < l$ is the VC dimension of the function class that the learning machine can implement, then the inequality obtains for all functions in that class with a probability of at least $1 - \eta$ (V. N. Vapnik, 1998, 1999), where the term ϕ indicates the level of confidence, defined as follows:

$$\phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right) = \sqrt{\frac{h(\log\left(\frac{2l}{h}\right) + 1) - \log\left(\frac{\eta}{4}\right)}{l}}. \tag{2.52}$$

Other ideas, such as the *annealed VC entropy*or the *Growth function*, can be used to formulate tighter boundaries. These are typically seen to be more difficult to model, although they are critical to the conceptual aspect of VC theory (V. N. Vapnik, 1998). The *fat-splitting dimension* is an alternative capacity concept that can be used to create boundaries. The bound Eq. (2.51) requires some additional explanation. Suppose we wanted to learn a "dependency." where $P(x, y) = P(x).P(y)$, i.e., where the pattern x contains no information about the label y, with uniform $P(y)$. Given a fixed-size training sample, we can develop a ML that achieves zero training error (provided we have no examples contradicting each other).
On the other hand, this machine will need a high VC dimension, h, to reconstruct the random labelings. As a result, the expression Eq. (2.52), increasing monotonically with h, will be large, and the bound Eq. (2.51) will not support possible hopes that we should expect a small test error due to the small training error. This makes it understandable how Eq. (2.51) can hold independent assumptions about the underlying distribution $P(x, y)$: it always holds (provided that $h < l$), but it does not always make a nontrivial prediction $-a$ bound on an error rate becomes void if it is larger than the maximum error rate. To obtain nontrivial predictions from Eq. (2.51), the function space must be constrained, assuming the capacity (e.g., VC dimension) is small sufficient (about the available amount of data) (V. N. Vapnik, 1998).

2.8 Performance Evaluation of Various Forecasting Measurements

Evaluating the forecasting performance of different models is challenging because asset volatility is not directly observable (Tsay, 2010). Traditional econometric literature has focused on statistical accuracy measures of point forecasts, often overlooking the economic significance of these forecasts. Forecasts are designed to aid decision-making, so consideration of the decision-maker is crucial. Biases may arise if a model is tailored to specific users, such as a think-tank building a macro model for a particular political party (C. W. Granger & Pesaran, 2000). Typically, comparisons between models are made by examining the errors when forecast values are measured against actual outcomes. Metrics used include Mean Error (ME), MAE, MFE, MPE, MAPE, Theil's U Statistics, R-squared

(R^2), and RMSE. Other comparisons may involve the correlation between one model's predictions and another's realizations or utility and profit-based measures of predictive ability (West, 2006). For financial applications, statistical performance measurements may need to be increased, as the real efficacy of a trading strategy should be assessed based on its performance in a simulated trading environment. A virtual trader can buy or sell stock index funds, taking short and long positions to test the model's performance in a simulated market (M. Kumar & M., 2014).
This section examines various evaluations of time series forecasting approaches and their implementation. The raw data's is divided into training and test sets, with a validation set often used for model tuning. Preprocessing techniques, such as normalization and transformations like the Box-Cox Transformation, are applied. Models are constructed using the training set, and their forecasts are evaluated against the test set. Inverse transformations may be applied to forecasts if necessary. Multiple models are compared on the test dataset to assess prediction accuracy. Key performance metrics used by researchers are reviewed in this section.

2.8.1 Description of various forecasting evaluations

Frequently used as benchmarks, statistical loss functions are important for evaluating forecast accuracy. Accuracy measures are typically defined based on forecast errors, $e_i = y_i - \hat{y}_i$, where y_i is the actual daily closing price and $\hat{y}_t$ is the forecasted value. In financial time series, the decision-maker may prioritize the sign of stock predictions over the exact price value. Out-of-sample forecasts can be assessed by comparing the prediction signs with actual outcomes (M. Kumar & M., 2014). The mean closing price of a set (training or test) is given by

$$\bar{y} = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} y_i,$$

The variance is

$$\sigma^2 = \frac{1}{N_{set} - 1} \sum_{i=1}^{N_{set}} (y_i - \hat{y}_i)^2,$$

where N_{set} is the size of the set. Several accuracy measures exist in the forecasting literature, each with its strengths and weaknesses (G. Zhang et al., 1998). The most commonly used measures include:

$$ME = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} (y_i - \hat{y}_i). \tag{2.53}$$

$$MAE = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} |y_i - \hat{y}_i|. \tag{2.54}$$

- MAE represents the mean of absolute differences between actual and predicted values and more accurate evaluation results as reference Kao et al., 2020.

$$MSE = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} (y_i - \hat{y}_i)^2. \tag{2.55}$$

- Mean Square Error (MSE) measures the deviation of the prediction from the actual value by calculating the mean of the squared residuals (Yaslan & Bican, 2017).

$$\text{RMSE} = \sqrt{\frac{1}{\text{N}_{\text{set}}} \sum_{i=1}^{\text{N}_{\text{set}}} (y_i - \hat{y}_i)^2}. \tag{2.56}$$

- RMSE measures the standard deviation of the residuals between predicted and actual values (Okasha, 2014).

$$\text{MPE}\,(\%) = \frac{1}{\text{N}_{\text{set}}} \sum_{i=1}^{\text{N}_{\text{set}}} \left(\frac{y_i - \hat{y}_i}{y_i} \right) \times 100. \tag{2.57}$$

- MPE represents the mean of the percentage differences between actual and predicted values.

$$\text{MAPE}\,(\%) = \frac{1}{\text{N}_{\text{set}}} \sum_{i=1}^{\text{N}_{\text{set}}} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100. \tag{2.58}$$

- MAPE represents the mean of the absolute percentage differences between actual and predicted values. It is useful for evaluating the performance of forecasting models.

$$\text{R}^2 = 1 - \frac{\sum_{i=1}^{\text{N}_{\text{set}}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{\text{N}_{\text{set}}} (y_i - \bar{y})^2}. \tag{2.59}$$

- R-squared is a statistical measure representing the proportion of the variance in the dependent variable that is predictable from the independent variables.

$$\text{Theil's U Statistics} = \sqrt{\frac{\sum_{i=1}^{\text{N}_{\text{set}}} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{\text{N}_{\text{set}}} (y_i - \bar{y})^2}}. \tag{2.60}$$

- Theil's U statistic measures the accuracy of predictions relative to a naive benchmark model. Its properties are (Park, 1999):
  - It's a weighted average of cumulative predicted error.
  - $0 \leq \text{U} \leq 1$; U = 0 means perfect fit.
  - Changes in scale and data transformations have an impact on this measure.
  - For evaluating good forecast accuracy, the U-statistic should be near zero.

We have examined different crucial metrics for forecasting accuracy, each with unique properties. Utilizing multiple performance criteria provides a comprehensive understanding of forecast errors' magnitude and direction. Therefore, financial time series analysts typically employ several measures for robust evaluation (Adhikari & Agrawal, 2013). Among these, MAPE, RMSE, and R^2 are particularly favoured due to their sensitivity to large errors, making them ideal for assessing the performance of stock return predictors (Ettayyebi, 2017).

Chapter 3

Machine Learning approach

" Machine intelligence is the last invention that humanity will ever need to make. "

–Nick Bostrom

Contents

3.1 Introduction

3.1.1 Overview of Machine Learning

ML is a subset of AI that uses algorithms to discover and analyze complex relationships within data. For example, ML systems like Siri, which is an automatic speech recognition tool, can convert acoustic signals into meaningful sentences (Awad & Khanna, 2015). ML is applied in various fields, including web search, big data analytics, medical research, and financial forecasting. The core of ML lies in its ability to make predictions about future events or outcomes based on past data. Samuel, 1959b defined ML as the process through which computers can learn on their own from their experiences, gradually reducing the need for explicit programming. Similarly, Mitchell et al., 1986 describes ML as the improvement of task performance through experience. Alan Turing's influential work in 1950 formed the basis for assessing machine intelligence (Turing, 2009), with a focus on a machine's ability to simulate human-like responsiveness. These definitions highlight the transformative potential of ML and its role in driving user-centered innovations.

Financial markets are frequently subject to unpredictable and irrational behavior, which events like the Brexit vote or previous US elections can influence. This inherent unpredictability makes it difficult to identify consistent trends in financial data. To effectively model these turbulent structures, ML algorithms are needed to uncover hidden patterns and forecast future impacts. ML, along with DL, is considered the most effective approach for this task. DL, in particular, is highly skilled at handling complex structures and extracting intricate relationships, thereby improving the accuracy of derived insights [1].

3.1.2 Literature review of support vector machines

Financial time series prediction remains a key focus in the finance industry, with ML models demonstrating better performance than traditional forecasting techniques. Ongoing research for improved models is driven by the advent of automated electronic trading systems and the desire for higher yields. As a result, new studies in finance and computational intelligence literature continuously emerge, with DL recently emerging as a leading predictive approach (Sezer et al., 2020). However, this section focuses on SVM, sparse kernel decision machines rooted in statistical learning theory. SVM offers a rational approach to ML challenges by constructing its learning model with only a subset of the training data. It is widely applied in classification, regression, novelty detection, and feature reduction, and its principles also extend to *support vector regression (SVR)* for regression tasks (Géron, 2019). SVR, a supervised learning method, uses an asymmetrical loss function that equally penalizes high and low misestimates. Notably, SVR exhibits computational efficiency regardless of input space dimensionality, high prediction accuracy and robust generalization capabilities (Awad & Khanna, 2015). This section will provide concise insights, mathematical models, and graphical representations of SVM and SVR.

1. Source: www.kdnuggets.com

3.2 Support vector machines

3.2.1 Hyperplane classifiers

Determining a set of functions with quantifiable capabilities is crucial to constructing effective learning algorithms. Notably, V. Vapnik and Lerner, 1963, and V. Vapnik and Chervonenkis, 1964 focused on hyperplanes as a fundamental concept. In binary classification problems, SVM aim to identify a canonical hyperplane that efficiently separates two classes of training samples (M. Kumar & M., 2014; Suykens & Vandewalle, 1999; T. Raicharoen, 2003; V. N. Vapnik, 1998).

$$(w \cdot x) + b = 0, \quad \text{where } w \in \mathbb{R}^N, b \in \mathbb{R}, \tag{3.1}$$

corresponding to decision functions:

$$f(x) = sgn((w \cdot x) + b), \tag{3.2}$$

and then introduced the Generalized Portrait, a learning approach for separable situations to produce f from actual data. Two facts support it. First, there is a single hyperplane that yields the highest margin of separation between the classes among all hyperplanes that separate the data (V. N. Vapnik, 1998),

$$max_{w,b} min\{||x - x_i|| : x \in \mathbb{R}^N, (w \cdot x) + b = 0, i = 1, \ldots, l\}. \tag{3.3}$$

Secondly, as shown in Figure 3.1, the discriminative capacity of the SVM decreases as the margin increases. In this graphical representation, each data point occupies an n-dimensional space, and classification is accomplished by utilizing a hyperplane. The hyperplane effectively separates the data points into separate classes in a well-separable manner (refer to Figure 3.1), which characterizes the SVM as a discriminative classifier. The SVMs functionality relies on identifying the hyperplane that maximizes the minimum distance to the training samples (Raju et al., 2018).

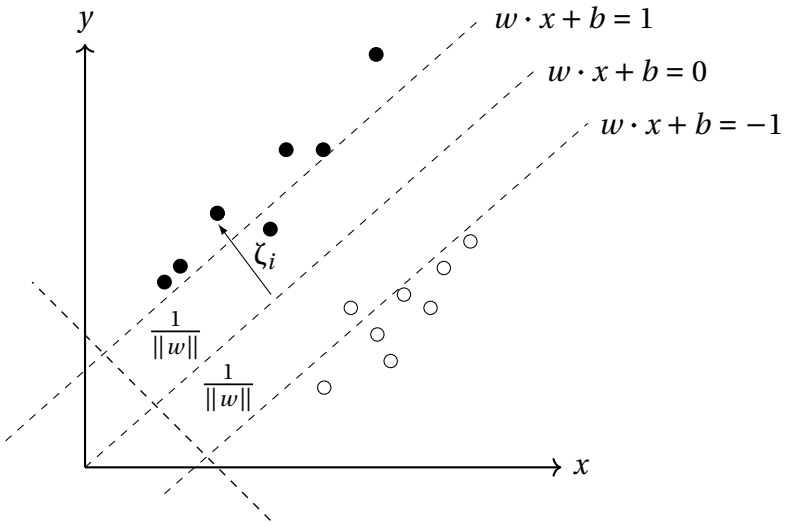


Figure 3.1 – Support vectors for linearly separable data points.

Optimal hyperplane

The *optimum hyperplane* is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted) and intersects it halfway between them. Based on the separability of the problem, there is a weight vector w and a threshold b such that

$$y_i \cdot ((w \cdot x_i) + b) > 0 \quad (i = 1, \ldots, l).$$

w and b are rescaled so that the closest point(s) to the hyperplane satisfy:

$$|(w \cdot x_i) + b| = 1,$$

We obtain a *canonical* form (w, b) of the hyperplane, satisfying

$$y_i \cdot ((w \cdot x_i) + b) \geq 1.$$

It's important to note that the margin in this situation is $\frac{2}{||w||}$ when measured perpendicularly to the hyperplane. This can be seen by considering two points x_1, x_2 on opposite sides of the margin, i.e.

$$\begin{array}{l}(w \cdot x_i) + b = +1 \\ (w \cdot x_i) + b = -1\end{array} \quad \Rightarrow (w \cdot (x_1 - x_2)) = 2 \Rightarrow \left(\frac{w}{||w||} \cdot (x_1 - x_2)\right) = \frac{2}{||w||},$$

then projecting them onto the normal vector of the hyperplane $\frac{w}{||w||}$ (Raju et al., 2018; V. N. Vapnik, 1998).

3.2.2 SVMs for linearly separable data

Lagrangian

To construct this *Optimal Hyperplane* (Figure 3.1), one solves the following quadratic optimization problem (QPP):

$$\begin{array}{ll}\text{Minimize} & \mathrm{J}(\boldsymbol{w}) = \frac{1}{2} w \cdot w = \frac{1}{2} ||w||^2 \\ \text{Subject to} & y_i \cdot ((w \cdot x_i) + b) \geq 1\,, \qquad i = 1, \ldots, l\end{array} \tag{3.4}$$

To solve the QPP Eq. (3.4) it is conveniently transformed to the dual space. Then, introducing Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian:

$$\mathrm{L}(w, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^{l} \alpha_i (y_i \cdot ((x_i \cdot w) + b) - 1). \tag{3.5}$$

The Lagrangian L must be minimized in terms of the *primal variables* w and b, while being maximized in respect of *dual variables* α_i (i.e. a saddle point has to be found). Let us try to get a sense of what was going on with this. If a constraint Eq. (3.4) is violated, then $y_i \cdot ((x_i \cdot w) + b) - 1 < 1$, in which case L can be increased by increasing the corresponding α_i. At the same time, w and b should change in order for L to decrease. To prevent $-\alpha_i (y_i \cdot ((x_i \cdot w) + b) - 1)$ from becoming arbitrarily large, the change in w and b will ensure that, provided the problem is separable, the constraint to be eventually satisfied. Similarly, all limitations that are not perfectly met can be understood as equalities, i.e. for which $y_i \cdot ((x_i \cdot w) + b) - 1 > 1$, the corresponding α_i must be 0: this is the value of α_i that maximizes L (V. N. Vapnik, 1998).

Karush-Kuhn-Tucker conditions

The Lagrange multipliers and *Karush-Kuhn-Tucker* (KKT) complementarity conditions of optimization theory (Bertsekas, 1997; Karush, 1939; Kuhn & Tucker, 1951) are used to find the optimal solution. The condition that at the saddle point, the derivatives of L with respect to the primal variables must vanish, the gradient of the Lagrangian (zero, with respect to primal variables)

$$\nabla \mathrm{L}(w, b, \alpha) = \begin{bmatrix} \frac{\partial}{\partial b}\mathrm{L}(w, b, \alpha) \\ \frac{\partial}{\partial w}\mathrm{L}(w, b, \alpha) \end{bmatrix} = \begin{bmatrix} -\sum_{i=1}^{l} \alpha_i y_i \\ w - \sum_{i=1}^{l} \alpha_i y_i x_i \end{bmatrix} = 0. \tag{3.6}$$

Based on the KKT conditions leads to,

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{3.7}$$

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i. \tag{3.8}$$

Note: These conditions appear in a study by Kuhn and Tucker, 1951 and also in Karush, 1939 unpublished master's thesis Awad and Khanna, 2015.

Support vector

As a result, the solution vector has an expansion in terms of a subset of the training patterns, especially those with a non-zero α_i, which are known to as Support Vectors (SVs). By the KKT complementarity conditions:

$$\alpha_i \cdot [y_i((x_i \cdot w) + b) - 1] = 0, \quad i = 1, \ldots, l, \tag{3.9}$$

The SVs lie on the margin (Figure 3.1). All remaining examples of the training set are irrelevant: their constraint Eq. (3.4) does not play a role in the optimization, and they do not appear in the expansion Eq. (3.8). The hyperplane (Figure 3.1) is totally controlled by the patterns closest to it, moreover the solution should not be dependent on the other cases (V. N. Vapnik, 1998).

Dual optimization problem

By substituting Eq. (3.7) and Eq. (3.8) into L, one eliminates the primal variables and arrives at the Wolfe dual of the optimization problem (e.g. Bertsekas, 1997): find multipliers α_i for the dual problem of SVM optimization which

$$\text{Minimize} \quad \mathrm{W}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot y_j) \tag{3.10}$$

$$\text{Subject to} \quad \sum_{i=1}^{l} \alpha_i y_i = 0, \ \text{and} \ \alpha_i \geq 0, \quad i = 1, \ldots, l. \tag{3.11}$$

As a result, the hyperplane decision function can be written as follows (Adhikari & Agrawal, 2013; V. N. Vapnik, 1998):

$$f(x) = sgn\left(\sum_{i=1}^{l} y_i \alpha_i \cdot (x \cdot x_i) + b\right), \tag{3.12}$$

where b is computed using Eq. (3.9) called the optimal bias. An unknown data is classified in either of the two classes according to the sign of $f(x)$ (Adhikari & Agrawal, 2013), i.e. $sign()$ is the signum function. It gives $+1$ (stable slope) if the element is greater than or equal to zero, and -1 (failed slope) if it is less than zero (Samui & Kothari, 2011).
There are various theoretical arguments in support of the optimal hyperplane's high generalization performance. It is also computationally appealing because it may be built by solving a quadratic programming problem. But how can this be generalized to the case of decision functions which, unlike Eq. (3.2), are nonlinear in the data (V. N. Vapnik, 1998).

3.2.3 SVMs for non-linearly separable data

Soft-Margin SVMs

When the data are not completely separable (e.g., due to noise), as with the points misclassified marked in Figure 3.2, **slack** variables ξ_i are introduced to the SVM objective function to allow error in the misclassification. In this context, SVM is not looking for a hard margin that will perfectly categorize all data. Consequently, SVM is now a soft-margin classifier, which means it correctly classifies most of the data while allowing a few points near the separating boundary to be misclassified (Awad & Khanna, 2015).

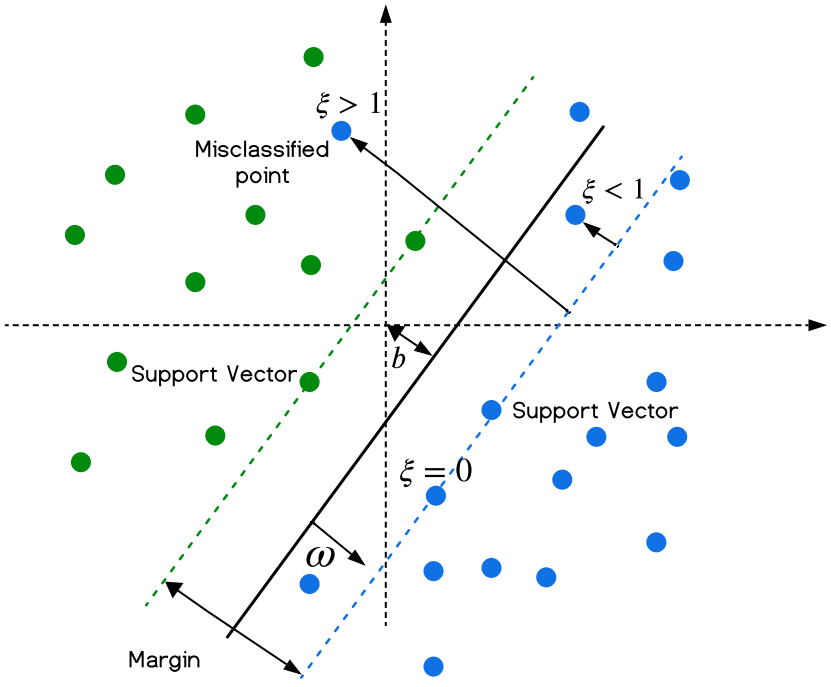


Figure 3.2 – A few misclassifications, as part of soft-margin SVM.

In primal form

A *Soft Margin Hyperplane* classifier is constructed in such cases. The corresponding QPP is given by Awad and Khanna, 2015; Suykens and Vandewalle, 1999; V. N. Vapnik,

1998:

$$\begin{aligned} &\text{Minimize} && \mathrm{J}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{\xi}) = \tfrac{1}{2}\|w\|^2 + \mathrm{C}\sum_{i=1}^{l}\xi_i \\ &\text{Subject to:} && y_i \cdot ((w \cdot x_i) + b) \geq 1 - \xi_i\,; \qquad i = 1,\ldots,l \\ & && \xi_i \geq 0\,; \qquad\qquad\qquad\quad i = 1,\ldots,l \end{aligned} \tag{3.13}$$

here the slack variables ξ_i are introduced to relax the hard-margin constraints and $\mathrm{C} > 0$ is the *regularization constant*[2] which assigns a penalty to misclassification (see Figure 3.2). Again, Eq. (3.13) represents a QPP, and its solution can be found by applying Lagrange multipliers and Kühn-Tucker conditions. The optimal weight vector and decision function are similar to those in the linearly separable case (Adhikari & Agrawal, 2013).

In dual form

In dual form the soft margin SVMs formulation is:

$$\begin{aligned} &\text{Minimize} && \mathrm{W}(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i - \tfrac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) \\ &\text{Subject to} && \sum_{i=1}^{l}\alpha_i y_i = 0, \text{ and } 0 \leq \alpha_i \leq \mathrm{C}, \qquad i = 1,\ldots,l \end{aligned} \tag{3.14}$$

The upper bound C on the Lagrange multipliers α_i is the only difference from the separable case. This way, the influence of the individual patterns (which could always be outliers) gets limited (V. N. Vapnik, 1998). As above, the solution takes Eq. (3.12). The threshold b can be calculated by making use of the fact that the slack variable x_i is zero for all SVs x_i with $\alpha_i < \mathrm{C}$ (again, as a result of the KKT complementarity criteria), and thus:

$$y_i = \sum_{j=1}^{l} y_j \alpha_j \cdot (x_i \cdot x_j) + b. \tag{3.15}$$

If one uses an optimizer that works with the double dual, one can recover the primal variable b value directly from the corresponding double dual variable (V. N. Vapnik, 1998).

3.2.4 Feature spaces and kernels

When a problem cannot be separated linearly in the input space, soft-margin SVM cannot find a robust hyperplane that minimizes the number of misclassified data points and generalizes well (see Figure 3.2). To achieve this, a kernel can be used to transform the data into a higher-dimensional space called the *kernel space*, where the data can be linearly separated (see Figure 3.3). Instead of solving for a high-order separating hypersurface in the input space, a linear hyperplane can be obtained in the kernel space to separate the multiple classes involved in the classification task. This method is appealing because the overhead of moving to the kernel space is relatively lower compared to learning a nonlinear surface (Awad & Khanna, 2015). The main idea is to use a nonlinear map to project the data into another dot product space (called the feature space) F:

2. The *regularization term* often known as the *box constraint*, or C, is a variable that depends on the optimization objective. As C increases, the margin becomes tighter, and the number of misclassifications decreases. Because the SVM's goal is to maximize the margin between the two classes, as C decreases, more violations are permitted (Awad & Khanna, 2015).

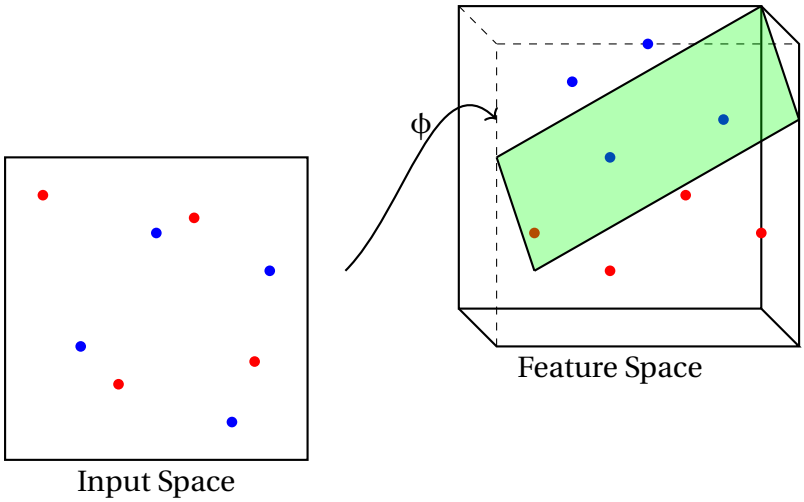


Figure 3.3 – Kernel trick in support vector classification.

$$\Phi : \mathbb{R}^N \longrightarrow F, \tag{3.16}$$

and use F to carry out the linear algorithm described above. This problem can be solved by indicating that both the building of the optimal hyperplane in F (cf. Eq. (3.10)) and the evaluation of the corresponding decision function Eq. (3.12) only require the evaluation of dot products $(\Phi(x)\cdot\Phi(y))$, and never the mapped patterns $\Phi(x)$ in explicit form (Kara et al., 2011; V. N. Vapnik, 1998).

Mercer kernel

A kernel must be a Hermitian and positive semidefinite matrix, and it must satisfy Mercer's theorem, which states that the kernel or Gram matrix must be **positive** on all pairs of data points, and **semidefinite** (Awad & Khanna, 2015), forming:

$$K(x,y) = \sum_r \Phi_r(x)\Phi_r(y), \tag{3.17}$$

where $\Phi(x)$ belongs to the Hilbert space.

Theorem 2. ***(Mercer)***
If K *is a continuous symmetric kernel of a positive integral operator* T*, i.e.*

$$(Tf)(y) = \int_C K(x,y)f(x)dx, \tag{3.18}$$

with

$$\int_{CC} K(x,y)f(x)f(y)dxdy \geq 0, \tag{3.19}$$

for all $f \in L_2(C)$ *(*C *being a compact subset of* $\mathbb{R}^N$*), it can be expanded in a uniformly convergent series (on* $C \times C$*) in terme of* T*'s is eigenfunctions*[3] Ψ_j *and positive eigenvalues* λ_j *,*

$$K(x,y) = \sum_{j=1}^{N_F} \lambda_j \Psi(x)\Psi(y), \tag{3.20}$$

3. In mathematics, an eigenfunction of a linear operator T defined on some function space is any non-zero function f in that space that, when acted upon by T, is only multiplied by some scaling factor called an eigenvalue. As an equation, this condition can be written as $Tf = \lambda f$.

where $N_F \leq \infty$ *is the number of positive eigenvalues.*

Note that originally proven for the case where $\mathcal{C} = [a, b]$ $(a < b \in \mathbb{R})$ this theorem also holds true for general compact spaces (Dunford & Schwartz, 1963; V. N. Vapnik, 1998).

Types of SVM-Kernels

The primal formulation of the kernel SVM is

$$\begin{array}{lll} \text{Minimize} & J(\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\xi}) = \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i & \\ \text{Subject to:} & y_i \cdot ((w \cdot \Phi(x_i)) + b) \geq 1 - \xi_i \,; & i = 1, \ldots, l \\ & \xi_i \geq 0 \,; & i = 1, \ldots, l \end{array} \tag{3.21}$$

where $\Phi(x_i)$ is such that $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$.

The SVM solution must again answer the KKT constraints, which are as follows:

1. $w = \sum_{i=1}^{l} \alpha_i y_i \Phi(x_i)$
2. $\sum_{i=1}^{l} \alpha_i y_i = 0$
3. $C - \mu_i - \alpha_i = 0; \quad i = 1, \ldots, l$
4. $\alpha_i (y_i (w \cdot \Phi(x_i)) + b) - 1 + \xi_i) = 0, \quad i = 1, \ldots, l$
5. $\mu_i, \xi_i = 0, \quad i = 1, \ldots, l$
6. $\mu_i, \xi_i > 0, \quad i = 1, \ldots, l$

Some well-known kernels used in SVM literature are shown in the following Table 3.1.

Table 3.1 – Some popular kernel functions (Adhikari & Agrawal, 2013; Awad & Khanna, 2015; Ranković et al., 2014).

Name of kernel	$\mathbf{K(x, y)}$
Polynomial function	$(ax \cdot y + c)^q, q > 0$
Linear kernel	$x \cdot y$
Hyperbolic tangent (sigmoid)	$\tanh(\beta x \cdot y + \gamma)$ where β, γ are constants.
Affine	$(x \cdot y + \sigma)^p$
Correlation	$\exp\left(\frac{x \cdot y}{\|x\|\|y\|} - \sigma\right)$
Gaussian radial basis function (RBF)	$\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right), \sigma > 0$
Laplacian (RBF)	$\exp\left(-\frac{\|x-y\|}{\sigma}\right)$
Rationnel (RBF)	$1 - \frac{\|x-y\|^2}{\|x-y\|^2 + \sigma}$
χ^2	$\exp\left(-\frac{\left(\sum_k \frac{(x_k - y_k)^2}{x_k + y_k}\right)}{\sigma}\right)$

3.3 Support vector regression

3.3.1 SVR overview

A Support Vector Regression is a type of SVM and supervised learning algorithm that analyzes data for regression analysis. Drucker et al., 1996 proposed this version of SVM for regression in 1996. A regression model estimates a continuous-valued multivariate function, a generalization of the classification issue. Binary classification issues are solved with SVMs by transforming them into convex optimization problems (V. N. Vapnik, 1998). The optimization issue is finding the largest margin separating the hyperplane while correctly categorizing as many training points as possible. With support vectors, SVMs describe this ideal hyperplane. The SVM's sparse solution and high generalization suit regression issues well. SVM to SVR generalization is achieved by attaching the function with a ε-insensitive band known as the ε-tube. The optimization problem is reformulated in this tube to identify the tube that best approximates the continuous-valued function while balancing model complexity and prediction error (Awad & Khanna, 2015).
SVR is characterized as an optimization problem in which the convex ε-insensitive loss function is optimized, and then the flattest tube contains the majority of the training cases. As a result, a multiobjective function is formed by combining the loss function with the geometrical characteristics of the tube. Then, using appropriate numerical optimization algorithms, convex optimization, which has a unique solution, is solved. The hyperplane is represented using support vectors and training samples outside the tube's limits. In SVR, the most influential cases that determine the form of the tube are the support vectors, and the training and test data are considered to be i.i.d., obtained from the same fixed but unknown probability distribution function in a supervised learning setting (Awad & Khanna, 2015). The performance of an SVR model depends on a proper set of parameters. The SVR parameters, the kernel function, the regularization parameter, and the tube size of the ε-insensitive loss function are specified carefully by the trial-and-error method (see subsection 3.3.3) (Ranković et al., 2014).

3.3.2 SVR: Concepts, mathematical model, and graphical representation

The term "margin" refers to a notion used in pattern recognition. Using Vapnik's ε-insensitive loss function Figure 3.4, an analogue of the margin is generated in the space of the target values y (keep in mind that in regression, we have $y \in \mathbb{R}$) to generalize the SV procedure to regression estimation (Ranković et al., 2014; V. N. Vapnik, 1998)

$$L_\varepsilon(y, f(x,w)) = |y - f(x)|_\varepsilon := \begin{cases} 0\,, & \text{if } |y - f(x,w)| \le \varepsilon. \\ |y - f(x,w)| - \varepsilon\,, & \text{otherwise.} \end{cases} \tag{3.22}$$

Then, as usual, the empirical risk to be minimized is given by Adhikari and Agrawal, 2013:

$$R_{emp}(f) = \frac{1}{l}\sum_{i=1}^{l} L_\varepsilon(y_i, f(x_i, w)). \tag{3.23}$$

To estimate a linear regression

$$f(x,w) = (w \cdot x) + b, \tag{3.24}$$

with precision ε, one minimizes

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}|y_i - f(x_i, w)|_\varepsilon. \tag{3.25}$$

Written as a constrained optimization problem this reads (Cortes & Vapnik, 1995):

$$\text{Minimize} \quad J(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \tag{3.26}$$

$$\text{Subject to:} \quad ((w \cdot x_i) + b) - y_i \le \varepsilon + \xi_i\,; \qquad i = 1, \ldots, l \tag{3.27}$$

$$y_i - ((w \cdot x_i) + b) \le \varepsilon + \xi_i^*\,; \qquad i = 1, \ldots, l \tag{3.28}$$

$$\xi_i, \xi_i^* \ge 0\,; \qquad i = 1, \ldots, l \tag{3.29}$$

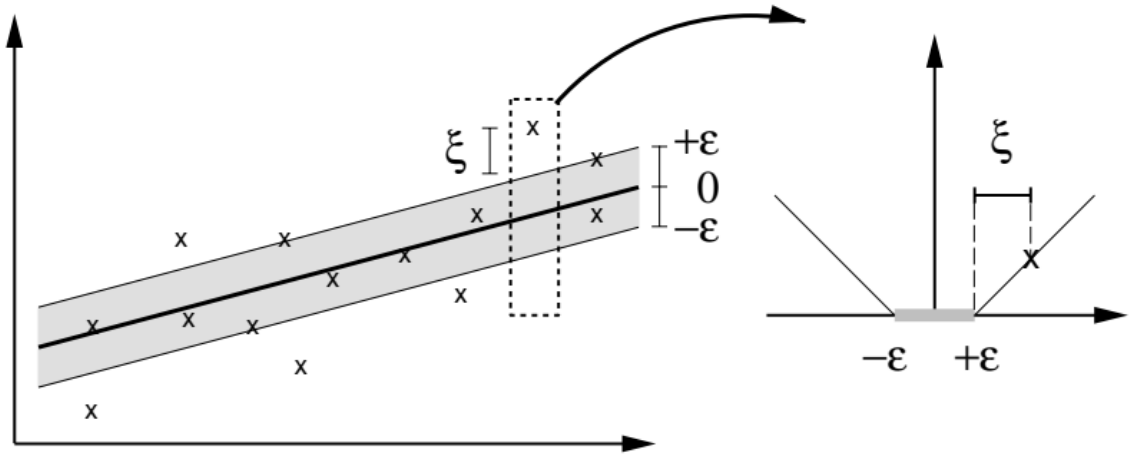


Figure 3.4 – Nonlinear SVR with ε-insensitive loss function (Ranković et al., 2014; Smola & Schölkopf, 2004).

In SVs regression, the desired accuracy ε is specified a priori. After that, one tries to fit a tube with a radius of ε to the data. The trade-off between model complexity and points lying outside of the tube (with positive slack variables ξ_i is shown in Figure 3.4) is determined by minimizing Eq. (3.26). For all $i = 1, \ldots, l$. Note that according to Eq. (3.27) and (3.28) any error smaller than ε does not require a nonzero ξ_i or ξ_i^*, and hence does not enter the objective function Eq. (3.26).

Similarly to pattern recognition, generalization to nonlinear regression estimation is accomplished by utilizing kernel functions. Introducing Lagrange multipliers, one thus arrives at the following optimization problem: for $C > 0$, $\varepsilon \ge 0$ chosen a priori,

$$\begin{aligned}\text{Maximize} \quad W(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = &-\varepsilon\sum_{i=1}^{l}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)y_i \\ &-\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_i^* - \alpha_i)k(x_i, x_j)\end{aligned} \tag{3.30}$$

$$\text{Subject to:} \quad 0 \le \alpha_i, \alpha_i^* \le C,\ i = 1, \ldots, l, \text{ and } \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0. \tag{3.31}$$

Regression function

Finally, the great decision hyperplane is found to be (Adhikari & Agrawal, 2013; Ranković et al., 2014; V. N. Vapnik, 1998):

$$f(x) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) k(x_i, x) + b, \tag{3.32}$$

where b is determined by the fact that Eq. (3.27) becomes equality with $\xi_i = 0$ if $0 < \alpha_i < C$, and Eq. (3.28) become equality with $\xi_i = 0$ if $0 < \alpha_i < C$, and Eq. (3.28) becomes an equality with $\xi_i^* = 0$ if $0 < \alpha_i^* < C$. From an abstract perspective, several expansions of this method are possible; all we need is a goal function that depends on the vector (w, ξ) (cf. Eq. (3.26)).

Finally, the procedure can be adjusted so that ε does not need to be defined ahead of time. Instead, one specifies an upper bound $0 \leq \nu \leq 1$ on the fraction of points allowed to lie outside the tube (asymptotically, the number of SVs) and the corresponding ε is computed automatically (Figure 3.4). This is achieved by using a primal objective function:

$$\frac{1}{2}\|w\|^2 + C\left(\nu l\varepsilon + \sum_{i=1}^{l} |y_i - f(x_i)|_\varepsilon\right), \tag{3.33}$$

instead of Eq. (3.25) and treating $\varepsilon \geq 0$ as a parameter that we minimize over (V. N. Vapnik, 1998).

3.3.3 Bigger picture

Let us briefly examine the important characteristics of the SVs method for regression, as explained so far, before going into the algorithmic aspects of the implementation. Figure 3.5 contains a graphical overview of the different steps in the regression stage. A map Φ the input pattern (for which a prediction is to be formed) onto feature space. The images of the training patterns are then used to compute dot products under the map Φ (Smola & Schölkopf, 2004).

The input x and the SVs x_i are nonlinearly mapped (by Φ) into a feature space F (see Figure 3.5), where dot products are computed. Using the kernel k, these two layers are, in practice, computed in one single step. The results are linearly combined by weights ν_i, found by solving a quadratic program (in pattern recognition, $\nu_i = y_i\alpha_i$; in regression estimation Eq. (3.32), $\nu_i = \alpha_i^* - \alpha_i$) (Ranković et al., 2014). The linear combination is fed into the function σ (in pattern recognition, $\sigma(x) = sgn(x + b)$; in regression estimation, $\sigma(x) = x + b$) (V. N. Vapnik, 1998). Like other multivariate statistical models, the quality and performance of SVR models depend on a good set of three parameters – the regularization constant C, the tube size ε, and the type of kernel function. From the results reported in the literature, it can be observed that the Gaussian kernel (see Table 3.1) is commonly used for function approximation problems. There are no standard guidelines for selecting C and ε. Selection is usually a heuristic trial-and-error process. [4] The stability between model complexity and model performance is determined by the parameter

4. The process of trial and error is a problem-solving technique that involves many tries to find a solution. Almost all statisticians employ a fundamental learning strategy when learning new behaviours. Trying a method, seeing if it works, and if it doesn't, trying a different method. This process is repeated until the goal is achieved or a solution is found.

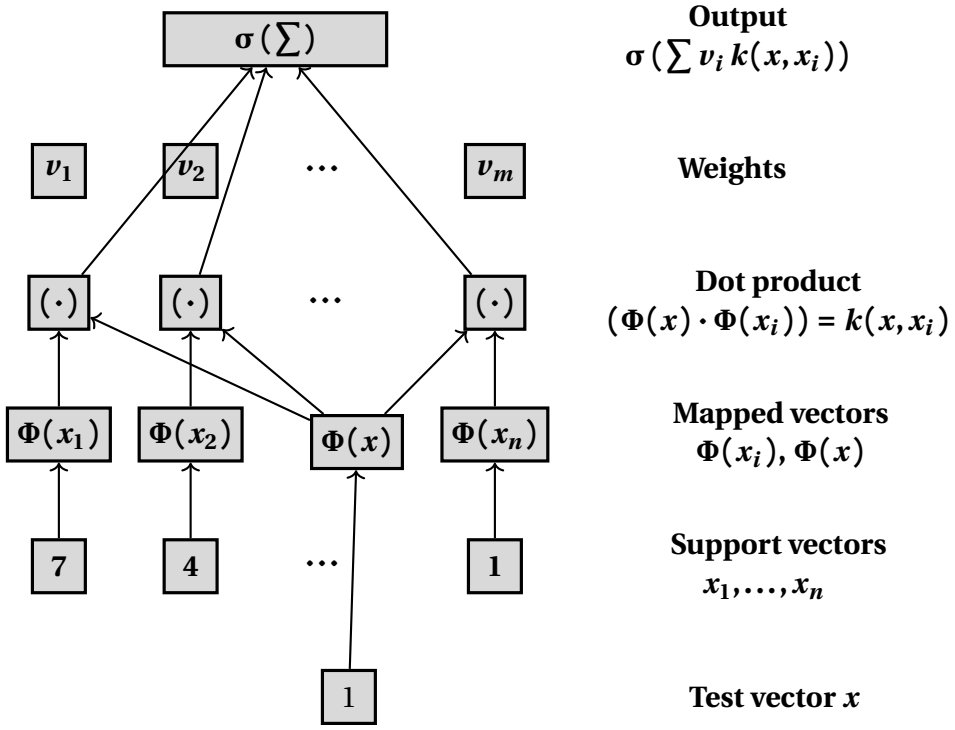


Figure 3.5 – Architecture of a regression machine constructed by the SVs algorithm.

C. The algorithm will overfit the training data if the C value is too large. The value of ε can affect the number of support vectors. By choosing a larger ε, the number of support vectors can be lowered (Ranković et al., 2014).

3.4 An overview of ANNs

In 1942, McCulloch and Pitts experimented with simple logical operation nets for a basic nonlinear model of real neurons to simulate bio-systems. In the field of computational calculation, this experiment opens a new perspective. Moreover, Table 3.2 summarizes the developments of ANNs from the early stages. In the last decade, ANNs have received much attention from researchers and are considered a powerful computational tool for complex problems. The ANNs also have the potential to make decisions and display adaptive behavior for complex, noisy sets of data, like a human brain (Raza & Khosravi, 2015).
Neurons, which are deeply connected little processing units of the network, are used in ANN models to attain the greatest performance. For improved results, the artificially attached neuron network processes massive data to investigate numerous conflicting ideas. ANNs-based models have been successfully implemented in several fields of daily life such as Biomedical applications, aerospace, automotive, electronics, finance industry, etc. (Lai, 1998). The ANNs received great attention from researchers since the mid-1980s for time series forecasting problems and attracted researchers as powerful computational tools for prediction problems. The ANNs perform much better than previously implemented techniques for non-linear input variables. The neural network can address complicated problems, including adaptive control, decision-making under uncertainty, and prediction patterns (Patterson, 1998). The ANNs have more advantages than a statistical model, as they can map the input and output relationship without making complex dependencies. The ANNs extract the nonlinear relationship between input variables using

the training process of the network. The network can learn the future output behaviour using the pattern reorganization function. On the other hand, the output patterns were learned by training the network with the input training data patterns. The appropriate learning algorithm, suitable training data, and optimized network structure may increase the network's overall performance and reduce the complexity (Ho et al., 1992).

Table 3.2 – Summary of major ANNs developments.

#	Year	Authors	References	ANNs developments
1	1942	McCulloch & Pitts	McCulloch and Pitts, 1943	Proposed the concept of first artificial neurons
2	1942	Hebb & Pitts	Hebb, 1968	Proposed first learning algorithm to memorize the adapting weight values
3	1958	Rosenblatt	Rosenblatt, 1958	Developed a first form of artificial network with Perceptron
4	1959	Lee	R. Lee, 1959	Proposed the Artron
5	1960	Widrow & Hoff	Widrow and Hoff, 1960	Proposed LMS training method and Adaline (Adaptive Linear Neuron)
6	1982	Hopfield	Hopfield, 1982	Design Hopfield neural network
7	1988	Widrow & Winter	Widrow and Winter, 1988	Design a network using Adeline neurons which is called Madaline
8	1986	Rumelhart et al.	McClelland and Rumelhart, 1981	Proposed multilayer perceptron base neural network with backpropagation algorithm
9	1987	Hecht-Nielsen	Hecht-Nielsen, 1987	Proposed the concept of self organizing mapping using counter propagation network
10	1988	Chua & Yang	Chua and Yang, 1988	Design cellular neural network

3.5 Biological neuron

In the last few decades, studies have shed light on the human brain and nervous system construction and operation (Principe et al., 2000). Neurons are the primary building blocks of the human nervous system. The central body, dendrites, synapses, and axon are the major components of a neuron, as shown in Figure 3.6. The information signal flows in a neuron from left to right, from the dendrites, through the cell body, and out the axon. The signal information is transferred from the axon to the dendrite in a second using a connection between axons. A synapse is a connection between two dendrites. Many processing components or neurons in the human brain process the data from the senses. The human brain has an estimated 10500 billion neurons to process all information (Rumelhart, McClelland, Group, et al., 1986). The biological neuron inspired the artificial neural network notion. A biological neuron has memory capability between the interconnection of neurons, and these connections are named synaptic weights.

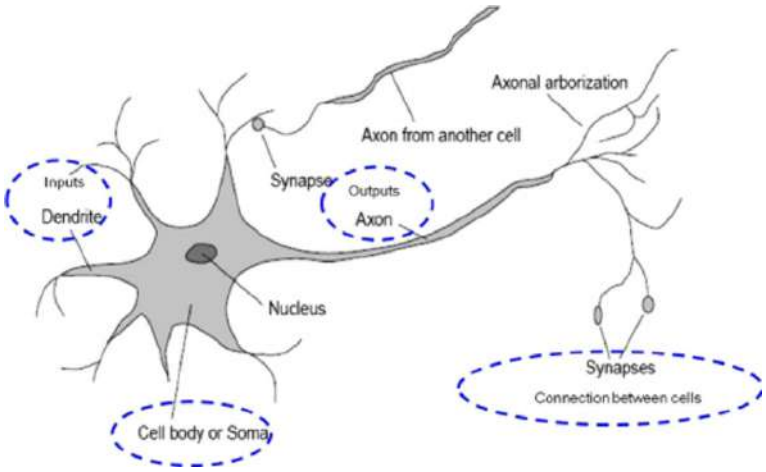


Figure 3.6 – Structural diagram of neuron (Jain et al., 2009).

3.6 Artificial neural networks

The neural network consists of input, hidden, and output layers connected by the processing elements that are called neurons, as shown in the diagram of Figure 3.7 neurons in the synaptic weights connect network layers. The neural network learning algorithm changes weights to map the input/output relationship. The sum of the weighted inputs is processed and applied to the activation function, which generates the output. To reduce the error between the network's generated output and the desired output, the weights and biases are changed (Raza & Khosravi, 2015). The *architecture* used to arrange the connections among nodes is crucial to the efficacy of the neural network. From simple single-layer *perceptrons* to complicated multilayer networks, there is a large range of topologies to choose from (Aggarwal, 2015; Kara et al., 2011). The simple mathematical expression of the output neuron can be calculated as follows:

$$y_j = f\Big(\sum_{i=1}^{n} w_{ij}x_i + w_{0j}\Big), \tag{3.34}$$

where y_j the output of network is, w_{ij} is the connection weight of Jth neuron to Ith layer neuron and x_i is the input of the neuron (Ettayyebi, 2017). The fundamental unit of a neural network may have a signal or many inputs and one output. Training and testing are two basic neural network operations: training and testing (Raza & Khosravi, 2015).

3.6.1 Types of activation function

The activation function of the neural network acts as a squeezing function to transfer weighted inputs to generate the network outputs. Various activation functions are present, such as step function, linear function, logistic function, sigmoid functions, and tangent-hyperbolic. However, the activation may vary according to the neural network's architecture, the number of inputs, and the nature of the problem. However, there is no clear rule for choosing the best activation function to produce the best network output. The literature review shows that the activation function change may affect the network output (G. Zhang et al., 1998). The activation function is a two-step process that is a linear combination of weighted inputs and transfer functions. In addition, the transfer function converts the weighted total inputs to the target unit. Several transfer functions

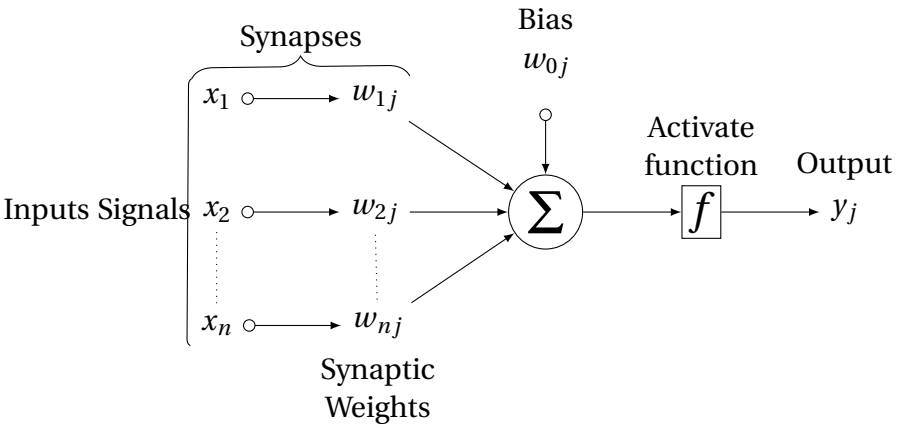


Figure 3.7 – Diagram of an ANNs.

are available, and they can be selected based on the problem's nature (Mellit & Kalogirou, 2008). This function resembles mapping training examples to binary class labels learned in SVMs. The following are three of the most commonly used activation functions:

- The linear function.
- The sigmoid function.
- The hyperbolic tangent function.

The sigmoid transfer function shown in Figure 3.8 takes the input x that $x \in (-\infty, +\infty)$, and produces an output σ that $\sigma \in [0,1]$. This transfer function is commonly used in Backpropagation Networks (cf. Figure 3.12). The general form of sigmoid's function is (Ettayyebi, 2017; Sezer et al., 2020):

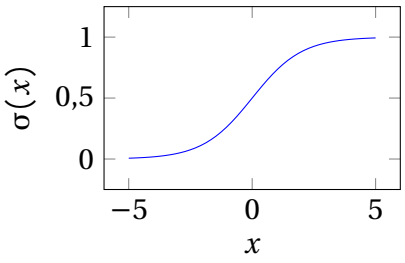


Figure 3.8 – Diagram of the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

The linear transfer function or Rectified Linear Unit (ReLU) shown in Figure 3.9 calculates the output of the neuron by simply returning the value that it was passed to it. This neuron can be developed to find a linear approximation to a nonlinear function or to learn an affine function of its inputs. A linear network cannot perform a nonlinear computation (G. Zhang et al., 1998).

$$\text{ReLU}(x) = \max(0, x)$$

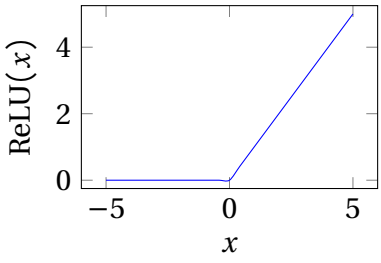


Figure 3.9 – Diagram of ReLU.

The hyperbolic tangent transfer function shown in Figure 3.10 is used to form neurons that make classification decisions (Ettayyebi, 2017).

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

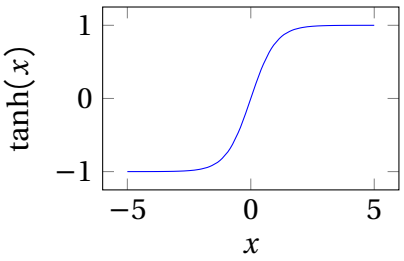


Figure 3.10 – Diagram of the hyperbolic tangent function.

Such a neuron is mentioned in the literature as the McCulloch-Pitts model in recognition of the work done by McCulloch and Pitts, 1943. In this model, the output of a neuron assumes the value 1 if the signal strength arriving at the neuron is positive and the value -1 otherwise.

3.6.2 Stochastic model of a neuron

Figure 3.7 shows a deterministic neural model with input-output behaviour that is precisely defined for all inputs. Using a stochastic neural model to analyze various neural network applications is preferable. The activation function of the McCulloch and Pitts, 1943 model is given a probabilistic interpretation in an analytically tractable method. For example, a neuron can only exist in one of two states: $+1$ or -1. A neuron's decision to *fire* (or change its state from "off" to "on") is probabilistic. Let x denote the neuron's state and $\mathbb{P}(v)$ signify the *probability* of firing, with v being the neuron's induced local field (Haykin et al., 2009). Then we can write:

$$x = \begin{cases} +1\,, & \text{with probability } \mathbb{P}(v) \\ -1\,, & \text{with probability } 1 - \mathbb{P}(v) \end{cases} \tag{3.35}$$

The sigmoid-shaped function is a popular choice for $\mathbb{P}(v)$:

$$\mathbb{P}(v) = \frac{1}{1 + e^{-v/\mathrm{T}}}, \tag{3.36}$$

where T is a *pseudo temperature* used to adjust the noise level and, consequently, the firing uncertainty (Little, 1974). However, it is important to realize that T is not the physical temperature of a neural network, be it a biological or an ANN. However, as previously suggested, we should consider T as a control parameter for the thermal fluctuations that represent the effects of synaptic noise. Note that when $\mathrm{T} \to 0$, the stochastic neuron is described by Eqs. (3.35) and (3.36) simplifies to the McCulloch-Pitts model, which is noiseless (i.e., deterministic) (Haykin et al., 2009).

3.7 Multilayer Neural Networks

The perceptron model is a simple type of neural network with only one input layer and one output layer. Because the input layers only transmit the attribute values without actually applying any mathematical function on the inputs, the perceptron model learns a simple linear model with a single output node as its function. In practice, more complex models may need to be learned with *Multilayer Neural Networks* (MNN) (Aggarwal, 2015). MNN has a *hidden layer*, in addition to the input and output layers. In principle, the nodes in the hidden layer can be connected using several topologies. The hidden layer, for example, could be typically composed of nodes from one layer feeding into nodes from the next. The *multilayer feed-forward network* is what this is termed. *Feed Forward Neural Network* (FFNN) is the most common form of Neural Networks architecture used in time

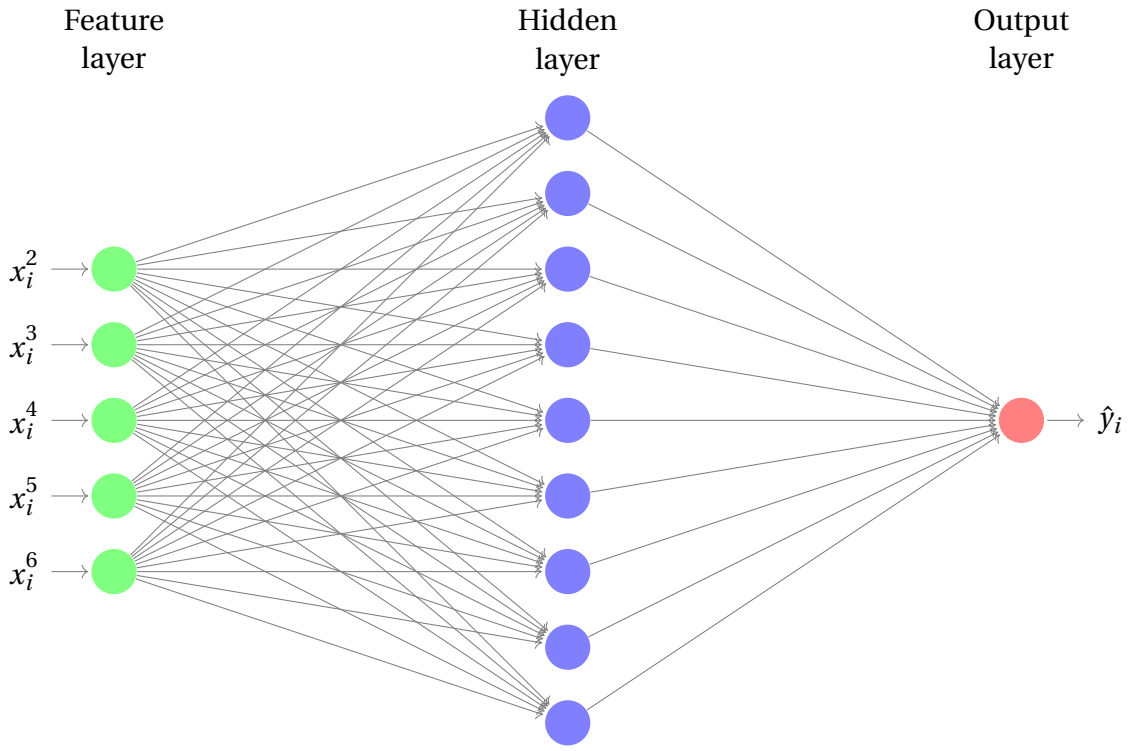


Figure 3.11 – Three-layer feed forward ANN architecture.

series modeling (Ettayyebi, 2017; Kara et al., 2011). The nodes in one layer are considered fully connected to the nodes in the second hidden layer. As a result, when the analyst specifies the number of layers and nodes in each layer, the topology of the multilayer feed-forward network is automatically determined. A single-layer feed-forward network can be analyzed to a basic perceptron. A multilayer feed-forward network with only one hidden layer is a commonly used model. Such a network may be considered a two-layer feed-forward network. An example of a three-layer feed-forward network is illustrated in Figure 3.11 and a Deep feed-forward neural network with two hidden layers (blue balls). In addition, the input layer (green balls) and the output layer (red points) are illustrated in Figure 3.12. Another property of the multilayer feed-forward network is that it is not limited to using linearly signed input functions. Arbitrary functions such as the logistic, sigmoid (cf. Figure 3.8), or hyperbolic tangents (cf. Figure 3.10) may be used in different nodes of the hidden layer and output layer. An example of such a function, when applied to the training tuple $\overline{X}_i = (x_i^1, \ldots, x_i^p)$, $\overline{W} = (w_1, \ldots, w_p)$ to yield an output value of $\hat{y}_i$, is as follows (Aggarwal, 2015):

$$\hat{y}_i = \sum_{j=1}^{p} w_i \frac{1}{1 + e^{-x_i^j}} + b. \tag{3.37}$$

The value of $\hat{y}_i$ is no longer a predicted output of the final class label in $\{-1, +1\}$ if it refers to a function computed at the hidden layer nodes. This output is then propagated forward to the next layer. In general, the architectures FFNN is shown in Figure 3.12 describes the output value $(\hat{y}_t)$ and the p inputs $(x_{t-1}, x_{t-2}, \ldots, x_{t-p})$ of the model is computed using the following mathematical expression (Sulistyowati et al., 2018):

$$\hat{y}_t = f^0 \left[\sum_{j=1}^{q} \left\{ w_j^0 f_j^h \left[\sum_{i=1}^{p} w_{ij}^h x_{t-i} + b_j^h \right] + b^0 \right\} \right], \tag{3.38}$$

where w_{ij}^h is the weight of the i th neuron input layer to the j th neuron hidden layer, b_j^h is the bias of the j th neuron in the hidden layer $(j = 1,2,\ldots,q)$, w_j^0 is the weight of the j th neuron from hidden layer to the neuron in the output layer, and b^0 is the bias of the neuron in the output layer. f_j^h is the activation function in the hidden layer using a sigmoid function (3.8). f^0 is the activation function in the output layer with the linear function (3.9).

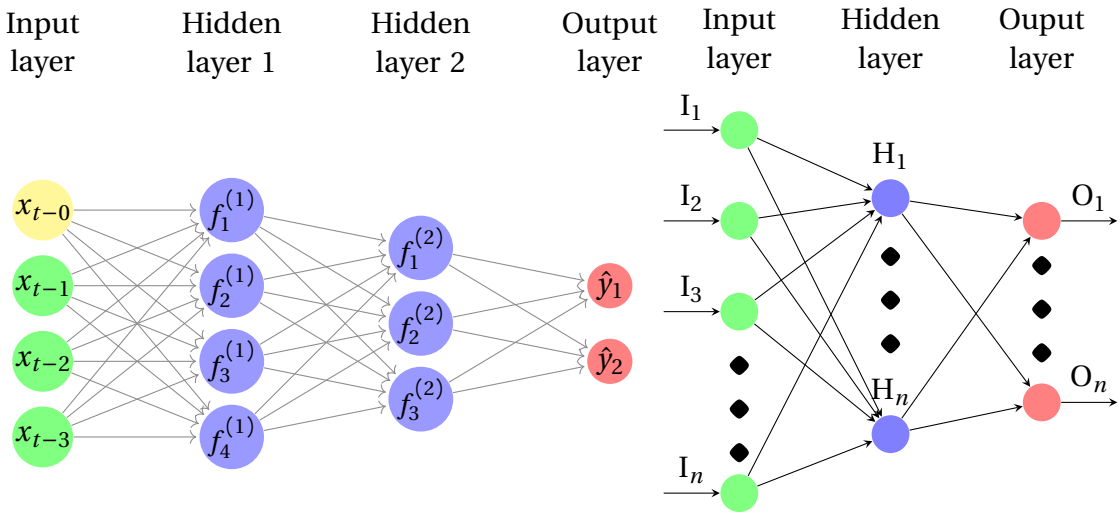


Figure 3.12 – A Multi-layer Feed-Forward Neural Network.

3.8 Comparing with the SVMs

In terms of capturing arbitrary functions, an MNN outperforms a kernel SVM. An MNN can record noncontiguous class distributions with distinct decision borders in various data areas and decision boundaries of arbitrary shapes. Various nodes in the hidden layer can logically capture different choice boundaries in different data areas, and the node in the output layer can integrate the findings from these different decision boundaries. For example, the three different nodes in the hidden layer of Figure 3.11 could conceivably capture three different nonlinear decision boundaries of different shapes in different localities of the data (Aggarwal, 2015). Furthermore, the generalization power of neural networks can be improved by using a (weighted) regularization penalty term $\lambda\frac{\|w\|^2}{2}$ in the objective function. The maximum margin term in SVMs is identical to this regularization term. For SVMs, the maximum margin term is often called the regularization term. Variations of SVMs exist, in which the maximum margin term is replaced with a penalty $\sum_{i=1}^{p}|w_i|$.

Additionally, certain forms of the slack term in SVMs (e.g., quadratic slack) are similar to the main objective function in other linear models (e.g., least-squares models). The slack term in SVMs is computed from the margin separators rather than the decision boundary, which is the essential difference. This is consistent with the philosophy of SVMs that discourages training data points from not only being on the wrong side of the decision boundary but also from being close to the decision boundary. Consequently, while different linear models share some fundamental similarities, they focus on distinct

optimization elements. This is why maximum margin models are generally more robust to noise than linear models that use only distance-based penalties to reduce the number of data points on the wrong side of the separating hyperplanes. Neural networks are susceptible to noise in experiments. On the other hand, MNN can approximate virtually any complex function in principle (Aggarwal, 2015).

3.9 Recurrent neural networks (RNNs)

ANNs with recurrent connections are called *Recurrent Neural Networks*, which are another type of DL network that is capable of modeling sequential data (e.g., time series) for sequence recognition and prediction, which was initially introduced by *Hochreiter* and *Schmidhuber* (Salehinejad et al., 2017; Siami-Namini et al., 2018). Some designs of RNNs were used to predict the stock market (K. Chen et al., 2015). In principle, RNNs can learn to map one variable sequence to another (Figure 3.13). Traditional ML models (*Back Propagation Through Time (BPTT)*, *Jordan-Elman networks*, and so on), also use RNNs, but the time lengths in these models are often shorter than in deep RNN models. RNNs are equivalent to very *Deep Neural Networks (DNN)*, that share model parameters and receive input at each time step; we can say that the RNN is a generalization of FFNN that has internal memory. Deep RNNs are preferred because they can cover longer periods. Unlike *Fully Connected Neural Networks (FCNNs)*. Therefore, according to FFNN, we can deduce that RNNs use internal *memory* to process incoming inputs. RNNs are used in different fields to analyze time-series data (handwriting recognition, translation machine, speech recognition, etc...). RNNs are strong for predicting the next character in a text, language translation applications, and sequential data processing, according to the literature (Sezer et al., 2020).

RNNs are a class of supervised ML models; RNN model architecture consists of a different number of layers and different types of units (neurons) in each layer; the units are connected by arcs (synapses) that have weight. A neuron's output is a non-linear mixture of its inputs. The main distinction between RNN and FCNN is that each RNN unit processes current and historical input data simultaneously. In the RNN model, the output is dependent on the preceding data. During their functioning, RNNs process input sequences individually at any given time. The "state vector" in the units on the hidden layer contains information about the input's history. The RNNs are turned into a *Deep Multilayer Perceptron (DMLP)* when the output of the units in the hidden layer is separated into discrete time steps (LeCun et al., 2015; Sezer et al., 2020). In Figure 3.13, the dynamics of the network depicted across time steps can be visualized by unfolding it as in. Given this figure (Figure 3.13), the network can be interpreted not as cyclic but rather as a deep network with one layer per time step and shared weights across time steps. Backpropagation can then train the unfolded network across many time steps. This algorithm, called BPTT, was introduced by Werbos in 1990. All recurrent networks in common current use apply it (Lipton et al., 2015); the information flow in the RNN's hidden layer is divided into discrete times. As shown in Figure 3.13. A simple RNN includes input, recurrent hidden, and output layers. N input units constitute the input layer. The inputs to this layer are a sequence of vectors through time t such as $\{\ldots, x_{t-1}, x_t, x_{t+1}, \ldots\}$, where $x_t = (x_1, x_2, \ldots, x_N)$. The input units in a fully connected RNN are connected to the hidden units in the hidden layer, where the connections are defined with a weight matrix $\mathbf{W}_{IH}$. Figure 3.13 shows that the hidden layer has M hidden units $\mathbf{h}_t = (h_1, h_2, \ldots, h_M)$, that are connected during time via recurrent connections. Using small non-zero elements to initialize hidden units can improve the network's overall performance and stability (Salehinejad et al., 2017). The

system's state space, or "memory", is defined by the hidden layer as follows:

$$\mathbf{h}_t = f_{\mathrm{H}}(\mathbf{o}_t), \tag{3.39}$$

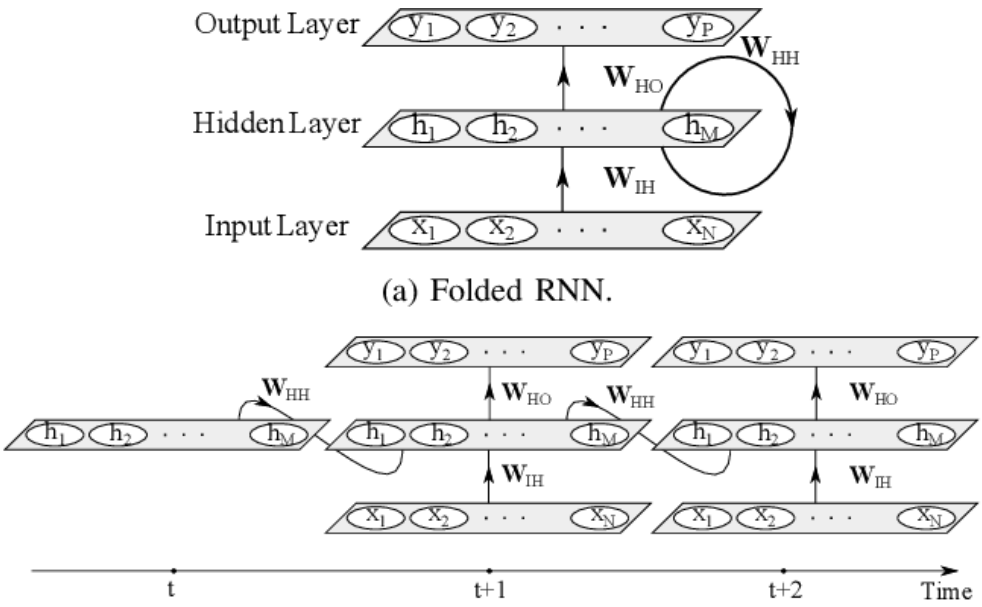
where

$$\mathbf{o}_t = \mathbf{W}_{\mathrm{IH}} x_t + \mathbf{W}_{\mathrm{HH}} \mathbf{h}_{t-1} + \mathbf{b}_h, \tag{3.40}$$

where the hidden layer activation function is $f_{\mathrm{H}}(.)$, while the hidden unit bias vector is $\mathbf{b}_h$. Weighted connections $\mathbf{W}_{\mathrm{HO}}$ connect the hidden units to the output layer. The output layer has P units $\mathbf{y}_t = (y_1, y_2, \ldots, y_{\mathrm{P}})$ that are computed as:

$$\mathbf{y}_t = f_o(\mathbf{W}_{\mathrm{HO}} \mathbf{h}_t + \mathbf{b}_o), \tag{3.41}$$

where $f_o(.)$ is the activation functions and $\mathbf{b}_o$ is the bias vector in the output layer. Because the input target pairs are instructed in time, the preceding processes are repeated over time $t = (1, \ldots, \mathrm{T})$. The Eqs. (3.40) and (3.41) indicate that an RNN is constituted of nonlinear state equations that can be iterated over time. The hidden states offer a prediction at the output layer depending on the input vector at each timestep. The hidden state of an RNN is a set of values that contains all of the unique necessary information about the network's historical states over many timesteps, independent of any external factors. This combined data can forecast the network's future behavior and make accurate predictions at the output layer (Pascanu et al., 2013; Salehinejad et al., 2017). An RNN uses a simple nonlinear activation function (3.6.1) in every unit. However, if effectively trained by timesteps, even a simple structure can model complex dynamics (Mikolov et al., 2014; Salehinejad et al., 2017).



(a) Folded RNN.

(b) Unfolded RNN through time.

Figure 3.13 – Unfolding structure of a simple RNN over time t. Each arrow represents a complete unit connected between the layers. Biases are not shown in the figure to keep it simple (Salehinejad et al., 2017).

3.9.1 Training RNNs

A training dataset of input-target pairs is required to train an RNN in a supervised fashion. By optimizing the network's weights, the goal is to minimize the difference between the output and target pairs (i.e., the loss value). A literature review reveals that the main focus is on reducing the complexity of training algorithms while increasing speed convergence. Some of the approaches for training RNNs are multi-grid random search, time-weighted pseudo-newton optimization, Gradient descent (GD), Hessian-free, Expectation Maximization (EM), Extended Kalman Filter (EKF), approximated Levenberg-Marquardt, and global optimization algorithms (Salehinejad et al., 2017). In this section, we discuss Gradient-based Learning Methods, detail for other methods is available in (Pascanu et al., 2013; Salehinejad et al., 2017).

Loss function

The loss function measures the network's performance by comparing the output y_t to the corresponding target z_t, which is defined as:

$$\mathcal{L}(y,z) = \sum_{t=1}^{\mathrm{T}} \mathcal{L}_t(y_t, z_t). \tag{3.42}$$

This is the sum of all losses in each timestep. The loss function (or the cost function) selection is problem-dependent. Some popular loss functions are *Euclidean distance* and *Hamming distance*[5] for forecasting of real-values and cross-entropy over the probability distribution of outputs for classification problems (Salehinejad et al., 2017).

Gradient-based Learning Methods

GD is a simple and popular optimization method in DL (Bengio et al., 1994). The GD is also called batch GD because, in each optimization iteration, it computes the gradient for the entire dataset to conduct a single update as:

$$\theta_{t+1} = \theta_t - \frac{\lambda}{\mathrm{U}} \sum_{k=1}^{\mathrm{U}} \frac{\partial \mathcal{L}_k}{\partial \theta}, \tag{3.43}$$

where U is size of training set, λ is the learning rate, and θ is set of parameters. This method is computationally intensive for big datasets and thus is not ideal for online training (i.e., training the models as inputs arrive) (Salehinejad et al., 2017).

Back-propagation through time

Back-propagation for feed-forward networks is referred to as BPTT. For learning RNNs, the conventional BPTT approach "unfolds" the network in time and propagates error signals backwards in time. By considering the network parameters in Figure 3.13 as the set $\theta = \{\mathbf{W}_{\mathrm{HH}}, \mathbf{W}_{\mathrm{IH}}, \mathbf{W}_{\mathrm{HO}}, \mathbf{b}_{\mathrm{H}}, \mathbf{b}_{\mathrm{I}}, \mathbf{b}_{\mathrm{O}}\}$ and $\mathbf{h}_t$ as the hidden state of network at time t, we can write the gradients as

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{t=1}^{\mathrm{T}} \frac{\partial \mathcal{L}_t}{\partial \theta}. \tag{3.44}$$

5. The Hamming norm $\| \mathrm{v} \|_{\mathrm{H}}$ is defined as the number of non-zero entries of vector v.

At time t, the expansion of loss function gradients is:

$$\frac{\partial \mathcal{L}_t}{\partial \theta} = \sum_{k=1}^{t} \left(\frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_t} \cdot \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \cdot \frac{\partial \mathbf{h}_k^+}{\partial \theta} \right), \tag{3.45}$$

$\frac{\partial \mathbf{h}_k^+}{\partial \theta}$ is the partial derivative (i.e., "immediate" partial derivative). It describes how the set θ parameters affect the loss function at the previous timesteps (i.e., $k < t$). To transport the error through time from timestep t back to timestep k, we can have the following:

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}. \tag{3.46}$$

We can consider Eq. (3.46) as a Jacobian matrix for the hidden state parameters in Eq. (3.39) as

$$\prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \mathbf{W}_{\mathrm{HH}}^{\mathrm{T}} diag|f_{\mathrm{H}}'(\mathbf{h}_{i-1})|, \tag{3.47}$$

where $f'(\cdot)$ is the element-wise derivate of function $f(\cdot)$ and $diag(\cdot)$ is the diagonal matrix.

The long-term and short-term contributions of hidden states in the network can generally be differentiated over time. The long-term dependency refers to the contribution of inputs and corresponding hidden states at time $k << t$, and short-term dependencies refer to other times Figure 3.14 shows that as the network makes progress over time, the contribution of the inputs x_{t-1} at discrete time $t-1$ vanishes through time to the timestep $t+1$ (the dark grey in the layers decays to higher grey). On the other hand, the contribution of the loss function value $\mathcal{L}_{t+1}$ concerning the hidden state $\mathbf{h}_{t+1}$ at time $t+1$ in BPTT is more than the previous timesteps (Pascanu et al., 2013; Salehinejad et al., 2017).

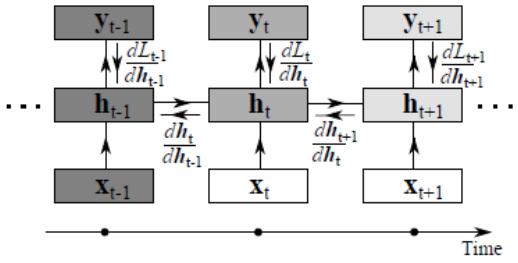


Figure 3.14 – As the network receives new inputs over time, the sensitivity of units decay (lighter shades in layers), and the BPTT overwrites the activation in hidden units. As a result, the first inputs visited are ignored (Pascanu et al., 2013; Salehinejad et al., 2017).

3.9.2 Memory problem

Unjustified amplification of weights and the model's inability to learn training data are two of the RNN's key issues. This is termed the "**Explodings Gradients**" actual problem. The second issue with simple RNNs is that they do not store information for long. Thus, the neural network will eventually lose the ability to connect the relationships between the data, finding it challenging to learn long-term addictions. The "**Vanishing gradient problem**" is the name given to this subject (Mikolov et al., 2014). To overcome exploding/vanishing gradient problems, a new concept has been introduced: the "**LSTM**" abbreviation of Long Short-Term Memory (Gers et al., 1999; Hochreiter & Schmidhuber, 1997; Mikolov et al., 2014; Rhanoui et al., 2019).

Vanishing gradient problem

According to the literature, capturing complex data patterns in the real world is possible using a strong nonlinearity (Mikolov et al., 2014). However, this may cause RNNs to suffer from the vanishing gradient problem (Bengio et al., 1994). The exponential shrinking of gradient magnitudes as they are propagated back through time is the subject of this problem. This phenomenon causes the memory of the network to ignore long-term dependencies and hardly learn the correlation between temporally distant events. There are two reasons for that:

1. Standard nonlinear functions such as the sigmoid function have a gradient almost everywhere close to zero (Mikolov et al., 2014).
2. The magnitude of the gradient is multiplied over and over by the recurrent matrix as it is backpropagated through time. When the eigenvalues of the recurrent matrix are less than one, the **gradient rapidly converges to zero in this situation**. This happens normally after 5 ~ 10 steps of back-propagation (Mikolov et al., 2014).

The gradients decrease when the weights are small when training RNNs on long sequences (e.g., 100 timesteps). A set of real numbers' products can shrink or explode to 0 or infinity, respectively. A similar idea applies to matrices, except shrinkage/explosion occurs in some directions. In Pascanu et al., 2013, it is shown that by considering ρ as the spectral radius of the recurrent weight matrix $\mathbf{W}_{\text{HH}}$, it is necessary at $\rho > 1$ for the long term components to explode as $t \to \infty$. It is possible to use singular values to generalize it to the nonlinear function $f'_{\text{H}}(\cdot)$ in Eq. (3.39) by bounding it with $\gamma \in \mathbb{R}$ such as

$$|| \, diag(f'_{\text{H}}(\mathbf{h}_k)) \, || \leq \gamma. \tag{3.48}$$

Using the Eq. (3.47), the Jacobian matrix $\frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}$, and the bound in Eq. (3.48), we can have

$$\left\| \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k} \right\| \leq || \, \mathbf{W}_{\text{HH}}^{\text{T}} \, || \cdot || \, diag(f'_{\text{H}}(\mathbf{h}_k)) \, || \leq 1. \tag{3.49}$$

We can consider $\left\| \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k} \right\| \leq \delta < 1$ such as $\delta \in \mathbb{R}$ for each step k. By continuing it over different timesteps and adding the loss function component, we can have the following:

$$\left\| \frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_t} \Big(\prod_{i=k}^{t-1} \frac{\partial \mathbf{h}_{i+1}}{\partial \mathbf{h}_i} \Big) \right\| \leq \delta^{t-k} \left\| \frac{\partial \mathcal{L}_t}{\partial \mathbf{h}_t} \right\|. \tag{3.50}$$

This equation shows that as $t - k$ gets larger, the long-term dependencies move toward zero, and the vanishing problem happens. Finally, we can see that the sufficient condition for the gradient vanishing problem to appear is that the largest singular value of the recurrent weights matrix $\mathbf{W}_{\text{HH}}$ (i.e., λ_1) satisfies $\lambda_1 < \frac{1}{\gamma}$ (Pascanu et al., 2013).

Exploding gradient problem

The exploding gradient problem is one of the most challenging aspects of employing BPTT to train RNNs (Bengio et al., 1994). Gradients for training RNNs on long sequences may expand when the weights increase and the gradient's norm rises dramatically. As it is stated in Pascanu et al., 2013, the necessary condition for this situation to happen is $\lambda_1 > \frac{1}{\gamma}$. Many techniques have recently been developed to solve the increasing gradient problem. Mikolov developed a gradient norm-clipping approach in 2012 to overcome the

growing gradient problem when training RNNs on large data with simple tools like BPTT and *Stochastic Gradient Descent* (SGD) (Mikolov & Zweig, 2012). Pascanu et al., 2013 has proposed a method identical to Mikolov's by using a hyper-parameter as the threshold for norm-clipping the gradients. Heuristic algorithms can set this parameter; however, the training technique is not very sensitive to it and works well for smaller thresholds (Salehinejad et al., 2017).

3.10 Long short-term memory

Recurrent connections can enhance neural networks' performance by leveraging their ability to understand sequential relationships. However, one drawback is that the memory provided by these connections can be limited due to the training methods used for RNNs. In the training phase, all models encountered challenges with either exploding or vanishing gradients, leading to the network's inability to learn long-term sequential correlations. To solve this problem, several models have been developed, with LSTMs being the most widely used (Salehinejad et al., 2017).

Hochreiter and Schmidhuber, 1997, introduced the LSTM model primarily to overcome the vanishing gradients problem. One of the most common and effective ways for reducing the impacts of vanishing and exploding gradients is the LSTM algorithm (K. Chen et al., 2015). This approach changes the structure of hidden units from "tanh" (3.10) or "sigmoid" (3.8) to memory cells, in which gates control their inputs and outputs. These gates control the flow of information to hidden neurons and preserve extracted features from previous timesteps (Le et al., 2015). Time-series data is the most popular type of data for LSTM models. *Natural Language Processing* (NLP), language translation, language modeling, sentiment analysis, predictive analysis, speech recognition, and financial time series analysis are just a few of their uses. LSTM networks can be more successful in time series data analysis, such as language translation, using attention modules and AEs [6] structures, in time series data research, such as language translation, LSTM networks can be more successful (Y. Wu et al., 2016).
It is demonstrated that the internal values of the LSTM model can expand indefinitely for a continuous sequence (Gers et al., 1999). The network cannot detect which information is no longer important, even when continuous sequences have naturally recurrent characteristics. The forget gate learns weights that determine the rate at which the memory cell's value decays. A memory cell keeps its value across time when the input and output gates are off, and the forget gate is not producing decay, assuring that the error gradient remains constant during back-propagation (Le et al., 2015). Because of this structure, the network may be able to remember information for extended periods (Salehinejad et al., 2017). The hidden layer of LSTM has a high level of complexity. For the identical sizes of hidden layers, a standard LSTM contains around four times more parameters than a standard RNN for the same size hidden layers (Mikolov et al., 2014). The goal of the LSTM approach, when it was first proposed, was to create a scheme that could improve learning long-term relationships, not to find the smallest or best scheme (Le et al., 2015).

3.10.1 Standard LSTM

As shown in Figure 3.15, a standard LSTM cell, consists of input, forget, and output gates and a cell activation component. These devices accept activation signals from vari-

6. Unsupervised learning models use networks, which are ANN types (Sezer et al., 2020).

ous sources and govern cell activation using predefined multipliers. For many timesteps, the LSTM gates can prevent most of the network from changing the contents of the memory cells. Signals are preserved, and errors are transmitted far longer in LSTM networks than in RNNs. Because of these characteristics, LSTM networks can process data with complicated and isolated inter-dependencies and perform in many sequence learning fields (Sezer et al., 2020; Van Houdt et al., 2020; Yao et al., 2015).

- The block input: This step is devoted to updating the block input component, which combines the current input x_t and the output of that LSTM unit $\mathbf{h}_{t-1}$ in the last iteration. This can be accomplished as shown below:

$$z_t = \tanh\left(\mathbf{W}_{\mathrm{I}g^c} x_t + \mathbf{W}_{\mathrm{H}g^c} \mathbf{h}_{t-1} + \mathbf{b}_g^c\right), \tag{3.51}$$

  $\mathbf{W}_{\mathrm{I}g^c}$ and $\mathbf{W}_{\mathrm{H}g^c}$ are the weights associated with x_t and $\mathbf{h}_{t-1}$, respectively, while $\mathbf{b}_g^c$ stands for the bias weight vector.

- The input gate: In this step, we update the input gate that combines the current input x_t, the output of that LSTM unit $\mathbf{h}_{t-1}$ and the cell value g_{t-1}^c in the last iteration. The following equation shows this procedure:

$$g_t^i = \sigma(\mathbf{W}_{\mathrm{I}g^i} x_t + \mathbf{W}_{\mathrm{H}g^i} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^i} g_{t-1}^c + \mathbf{b}_g^i), \tag{3.52}$$

  where $\mathbf{W}_{\mathrm{H}g^i}$ is the weight matrix from hidden state to the input gate, $\mathbf{W}_{\mathrm{I}g^i}$ represents the weight matrix from the input layer to the input gate, $\mathbf{W}_{g^c g^i}$ is the weight matrix from cell activation to the input gate, and $\mathbf{b}_g^i$ is the bias of the input gate.

- The forget gate: In this step, the LSTM unit determines which information should be removed from its previous cell states g_{t-1}^c. Therefore, the activation values g_t^f of the forget gates at time step t are calculated based on the current input x_t, the outputs $\mathbf{h}_{t-1}$ and the state g_{t-1}^c of the memory cells at the previous time step $t-1$, the peep-hole connections, and the bias terms $\mathbf{b}_g^f$ of the forget gates. This can be done as follows:

$$g_t^f = \sigma(\mathbf{W}_{\mathrm{I}g^f} x_t + \mathbf{W}_{\mathrm{H}g^f} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^f} g_{t-1}^c + \mathbf{b}_g^f), \tag{3.53}$$

  where $\mathbf{W}_{\mathrm{H}g^f}$ represents the weight matrix from hidden state to the forget gate, $\mathbf{W}_{\mathrm{I}g^f}$ is the weight matrix from the input layer to the forget gate, $\mathbf{W}_{g^c g^f}$ represents the weight matrix from cell activation to the forget gate, and $\mathbf{b}_g^f$ is the bias of the forget gate.

- The cell gate: This step computes the cell value, which combines the block input z_t, the input gate g_t^i and the forget gate g_t^f values, with the previous cell value. This can be performed as seen below:

$$g_t^c = g_t^i \tanh\left(\mathbf{W}_{\mathrm{I}g^c} x_t + \mathbf{W}_{\mathrm{H}g^c} \mathbf{h}_{t-1} + \mathbf{b}_g^c\right) + g_t^f g_{t-1}^c = g_t^i z_t + g_t^f g_{t-1}^c, \tag{3.54}$$

  where $\mathbf{W}_{\mathrm{H}g^c}$ represents the weight matrix from the hidden state to the cell gate, $\mathbf{W}_{\mathrm{I}g^c}$ represents the weight matrix from the input layer to the cell gate, and $\mathbf{b}_g^c$ is the bias of the cell gate.

- The output gate: This step calculates the output gate, which combines the current input x_t, the output of that LSTM unit $\mathbf{h}_{t-1}$ and the cell value g_{t-1}^c in the last iteration. This could be done in the following form:

$$g_t^o = \sigma(\mathbf{W}_{\mathrm{I}g^o} x_t + \mathbf{W}_{\mathrm{H}g^o} \mathbf{h}_{t-1} + \mathbf{W}_{g^c g^o} g_{t-1}^c + \mathbf{b}_g^o), \tag{3.55}$$

where $\mathbf{W}_{\mathrm{H}g^o}$ denotes the weight matrix from hidden state to the output gate, $\mathbf{W}_{\mathrm{I}g^o}$ is the weight matrix from the input layer to the output gate, $\mathbf{W}_{g^c g^o}$ denotes the weight matrix from cell activation to the output gate, and $\mathbf{b}_g^o$ is the bias of the output gate.

Finally, the hidden state (Block output), we calculate the block output, which combines the current cell value g_t^c with the current output gate value g_t^o as follows:

$$h_t = g_t^o \tanh\left(g_t^c\right). \tag{3.56}$$

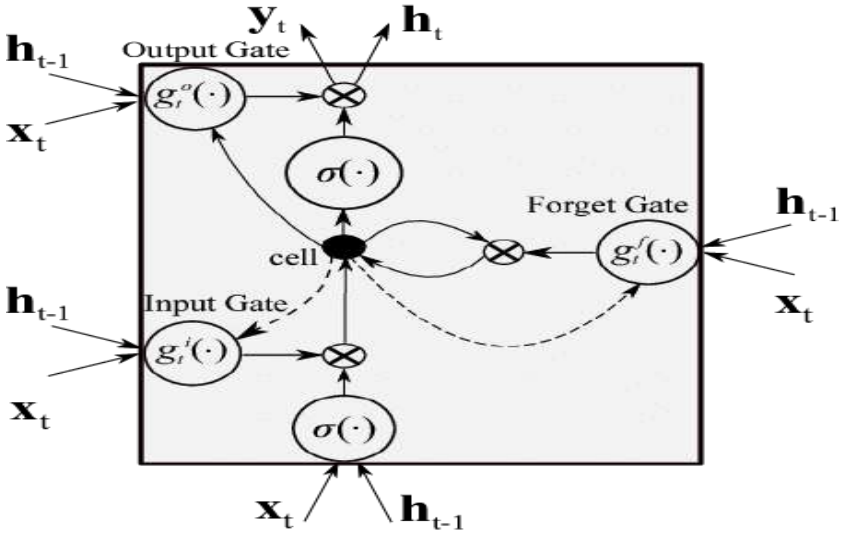


Figure 3.15 – One cell of the LSTM memory block. Time lag is represented by the dashed line.

3.10.2 Stack LSTMs

The idea of depth in ANNs also applies to LSTMs by stacking different hidden layers with LSTM cells in space to increase the network capacity (Kalchbrenner et al., 2015; Salehinejad et al., 2017). A hidden layer l in a stack of L LSTMs using the hidden layer in Eq. (3.39) is defined as:

$$h_t^l = f_{\mathrm{H}}(\mathbf{W}_{\mathrm{IH}}\mathbf{h}_t^{l-1} + \mathbf{W}_{\mathrm{HH}}\mathbf{h}_{t-1}^l + \mathbf{b}_h^l), \tag{3.57}$$

where the hidden vector sequence $\mathbf{h}_t^l$ is computed over time $t = (1,\ldots,\mathrm{T})$ for $l = (1,\ldots,\mathrm{L})$. The initial hidden vector sequence is defined using the input sequence $\mathbf{h}^o = (x_1,\ldots,x_{\mathrm{T}})$ (Salehinejad et al., 2017). Then, the output of the network is:

$$y_t = f_o(\mathbf{W}_{\mathrm{HO}}\mathbf{h}_t^{\mathrm{L}} + \mathbf{b}_o). \tag{3.58}$$

A stack pointer in a stacked LSTM can determine which cell in the LSTM contains the preceding timestep's state and memory cell (Yao et al., 2015). Not only can the controller push to and pop from the top of the stack in real-time, but an LSTM can also keep the contents of the stack in a continuous space embedding (Ballesteros et al., 2015; Yao et al., 2015). Combining stacked LSTM with different RNN structures for different applications requires investigation. One example is a combination of stacked LSTM with frequency domain CNN for speech processing (Salehinejad et al., 2017).

3.10.3 Advantages & disadvantages of standard LSTM and stacked LSTM

We can summarize the major advantages and disadvantages of the two different LSTM cells in the following Table 3.3:

Table 3.3 – Advantages and disadvantages of the Standard & Stacked LSTMs methods (Salehinejad et al., 2017)

Method	Advantages	Disadvantages
Standard LSTM	- They are able to model long-term sequence dependencies better than a simple RNN. - They are more robust to the problem of short memory than 'Vanilla' RNNs (vanishing gradients) since the definition of the internal memory is changed from (cf.(3.39) and (4.12)): $h_t = f_{\mathrm{H}}(\mathbf{o}_t) \longrightarrow h_t = g_t^o \tanh\left(g_t^c\right)$	- They add more parameters to learn, which increases the computing complexity as compared to RNN. - Because there are more memory cells than in 'Vanilla' RNNs, the amount of memory required is higher.
Stacked LSTM	- According to the deeper architecture, it models long-term sequential dependencies.	- Based on a stack of LSTM cells, it has a higher memory need and computational complexity than LSTM.

Chapter 4

A comparative study of ARIMA, SVMs, and LSTM models in forecasting the Moroccan stock market

"Prediction is very difficult, especially if it's about the future."

- Niels Bohr

Contents

4.1 Introduction

After the COVID-19 crisis had an impact on different sectors such as industrial, finance, health, aviation, etc. Investors boosted the stock market due to the economic recession; see, for example, Foroni et al., 2020. Modeling and forecasting the future price of a stock is important for investors as it helps them reduce the risk of making the wrong decision by successfully predicting price values or movement (Baek & Kim, 2018). Forecasting stock price trends is a challenging task because it constantly deals with a dynamic market that is influenced over time by numerous factors, including political, reserve events, movement of other financial markets, general economic conditions, investor expectations, as well as market volatility, which is difficult to model, and needs further study (Ayala et al., 2021; M. Kumar & M., 2014).
Recently, many researchers in financial econometrics have attempted to analyze the problem of stock price prediction as a problem of time series forecasting (Challa et al., 2020; Sezer et al., 2020). This raised their curiosity about improving forecasting models over time. Thus, it would be worthwhile to model financial time series as several approaches have been developed based on linear and non-linear models. During the past decades, much research has contributed to improving the quality of index prediction using classical and ML approaches (Sheta et al., 2015). These methods have captured the interest of researchers and investors since they can provide interesting ideas and insight about stock prices and trends (Ji et al., 2021).
Financial time series modeling uses linear statistical models such as regression, moving average, Holt-Winters Exponential Smoothing (ES), and Box-Jenkins' ARIMA (Khedmati et al., 2020; Rather et al., 2015). These methods are mostly used because they are prevalent for time series analysis. The results obtained are simple to understand and interpret but relatively powerless (Okasha, 2014; Qu & Zhang, 2016) when compared to classical nonlinear approaches, specifically the GARCH model, which measures and forecasts volatility in the behavior of financial time series (Alshammari et al., 2020; Catania & Grassi, 2022; El Ghini & Saidi, 2017) While examining historical data, the ARIMA model assumes a linear correlation between the values of a time series (Khashei & Hajirahimi, 2019). ARIMA models can predict short-term stock prices, which could help make investment decisions (Ariyo et al., 2014; Siami-Namini et al., 2018) and optimization of problem cloud resource orchestration (Qin et al., 2019).
ML is a branch of AI that uses algorithms to synthesize links between data and information in a systematic way Awad and Khanna, 2015. In recent years, the scientific community has become more interested in the development of two approaches: ML and DL, such as SVM, ANN (John et al., 2022), and LSTM for financial time series forecasting research. These approaches have attracted the attention of academics because they can provide useful findings and insight into market trends (Ji et al., 2021). The SVM is a rapidly growing technique often used for pattern classification and function estimation issues in various areas (Ranković et al., 2014; Xie et al., 2018). SVM has achieved extensive utilization regression estimation, which is called the SVR, and has thus been applied to time series forecasting challenges (Qu & Zhang, 2016). SVR concerns the divide of data and minimization of the distance between the hyperplane and the observations and tries to find the optimal hyperplane using the principle of SRM (Al-Musaylh et al., 2018; Ayala et al., 2021; Yaslan & Bican, 2017). The kernel function utilized in SVR is essential in capturing the nonlinear relationship between the input and output data in the high-dimensional feature space. It is thus regarded as a useful regression analysis approach (Khedmati et al., 2020; Sulistyowati et al., 2018).

The LSTM neural networks are developed by RNN and have significant application value in many real-world applications (Alizadeh & Ma, 2021; Casado-Vara et al., 2021; Nasirtafreshi, 2022; Van Houdt et al., 2020). LSTM is an advanced DL architecture that learns from non-linear relationships in sequential and time-series data (Baek & Kim, 2018). LSTM can remember both short-term and long-term values, and there has been a lot of recent research on the stock market's application (Hollis et al., 2018; McNally et al., 2018; Rundo et al., 2019). The "Vanishing Gradient Problem" is solved using the LSTM model. That is, the neurons of the hidden layers have no connection between them. Otherwise, the networks have difficulties retaining information over time. Therefore, LSTM uses memory cells to retain long-term and short-term memory separately (Chhajer et al., 2022; Z. Li & Tam, 2018; Qiu et al., 2020; Rhanoui et al., 2019). Different types of LSTM architectures exist for time series forecastings, such as vanilla LSTM, stacked LSTM, and bidirectional LSTM (Salehinejad et al., 2017).

Though models based on ARIMA, SVR, and LSTM have succeeded in time series forecasting in various real-world applications. For example, in the forecasting of COVID-19 cases (Sabry et al., 2022) in predicting primary energy consumption (Kao et al., 2020), they have several limitations. First, the ML techniques necessitate sufficient data to construct a good approximation. Second, when utilizing LSTM models, parameter selection, uncertainty in weight derivations, and the risk of overfitting must all be evaluated (McNally et al., 2018). As a result of these limitations, more studies have started to use SVR in forecasting, as it can minimize the disadvantages of LSTM models (Chhajer et al., 2022; Kao et al., 2020).

Many authors have conducted a comparison between the traditional methods, such as the ARIMA model, and ML methods, such as SVR and LSTM models, in forecasting financial time series (Elsaraiti & Merabet, 2021; Sheta et al., 2015; Siami-Namini et al., 2018). The examination of previous and recent related works of classical and ML approaches had been developed for stock price prediction (Salehinejad et al., 2017; Sezer et al., 2020). These studies were carried out to determine the performance between single and hybrid models; Rundo et al., 2019, Ettayyebi and El Himdi, 2018a. For example, Adebiyi et al., 2014 obtained the superiority of the single ANN model over the ARIMA model, with the published Dell stock index from the NYSE. M. Kumar and M., 2014 developed and identified four different models: ARIMA, SVM, ANN, RF, and their hybrids using the daily closing price values of the stock markets from the National Stock Exchange of India. The results found that the hybrid ARIMA-SVM model is the best to achieve high forecast accuracy and better returns. Okasha, 2014 investigated the Al-Quds daily stock price index of the Palestinian Stock Exchange using SVM, ARIMA, and ANN models. The results of SVM provide a more accurate model and more efficient forecasts for the stock index than both ANN and ARIMA models.

Rather et al., 2015 proposed RNN, AR, ES, Autoregressive Moving Reference (ARMR), and Hybrid Prediction model (HPM) for prediction of stock returns. The authors suggested that HPM outperforms RNNs, as expected, and achieves excellent prediction performance. Sheta et al., 2015 explored the use of ANNs, Multiple Linear Regression (MLR), and SVM to build prediction models for the Standard & Poor's 500 (S&P 500) stock index. Compared to MLR and ANN models, the developed SVM model with the RBF kernel model provided good prediction capabilities. Yaslan and Bican, 2017 proposed to predict the 3 different electricity load demands using a hybrid method that incorporates Empirical Mode Decomposition (EMD) and SVR algorithms. According to the results, the proposed algorithm outperforms the Single SVR and Denoised-SVR methods on electricity load forecasting. A similar study was done by Al-Musaylh et al., 2018 to forecast electricity demand

in Queensland, Australia, based on the Multivariate Adaptive Regression Spline (MARS), SVR, and ARIMA models. For the daily (24-hour) forecasting horizon, the SVR model outperforms the MARS and ARIMA models, with a higher WI (0.890) and MAE (162.363 MW). Hollis et al., 2018 examined the LSTM and the LSTM with attention using five stocks from Kaggle's Two Sigma and found that both had a reasonable performance of up to 60%. The comparison shows that an LSTM with attention can outperform LSTMs, but more research is needed because there are difficulties with such model architectures. Tamatta, 2018 used the ARIMA, Autoregressive Neural Network (AR-NN), and Trigonometric Box-Cox ARMA Trend Seasonality (TBATS) models to forecast patients waiting lists in different time bands over all of Ireland's Hospitals. According to the comparison of the three models, ARIMA is the best model with the lowest RMSE value. Sulistyowati et al., 2018 combined linear methods such as the Time Series Regression (TSR) model and ARIMA with Exogenous Factor (ARIMAX), and nonlinear models such as NN and SVR to predict Air Passenger and Cargo in Indonesia. The results indicated that the ARIMAX-NN and TSR-NN hybrids gave more accurate predictions than the TSR-SVR and ARIMAX-SVR hybrids. Siami-Namini et al., 2018 compared ARIMA and LSTM models in forecasting stock market indexes. The comparison result indicated that LSTM was superior to ARIMA, with an improvement obtained by LSTM between 84% and 87%. Z. Li and Tam, 2018 attempted to adopt a Decision Tree (DT), MLP, SVM, and LSTM for investigating the momentum and reversal effects using China Securities Index 300 (CSI 300). The experimental results demonstrated that the SVM is beneficial for capturing the relevant momentum and reversal effects and possibly building profitable trading strategies. McNally et al., 2018 used ARIMA, RNN, and LSTM to predict the price of Bitcoin. The ARIMA forecast based on errors performed worse than the LSTM and RNN models, when LSTM outperformed the RNN marginally, with the Graphics Processing Unit (GPU) training time surpassing the Central Processing Unit (CPU) implementation by 67.7%. Rhanoui et al., 2019 proposed an ARIMA random walk and LSTM neural network to forecast the financial budget. The analysis showed that the LSTM model performs better than the ARIMA model. Khashei and Hajirahimi, 2019 have proposed hybrid ARIMA and MLP models for stock price forecasting. Empirical results indicate that the conventional hybrid ARIMA-ANN and ANN-ARIMA models can achieve more accurate results. Khedmati et al., 2020 applied ARIMA, ANN, SVM, RF, Bayesian, and Kriging methods for modeling and forecasting the daily Bitcoin exchange rate. The SVM outperforms all univariate and multivariate models. It is chosen as the best model when the RMSE and MAPE are much smaller than the value of all other models. Qiu et al., 2020 compared the prediction of three stock indices using LSTM, Gated Recurrent Unit (GRU), and LSTM model with wavelet denoising. The results show that the coefficient of determination of the attention-based LSTM model is higher than 0.94, and the MSE of the model is lower than 0.05. Ji et al., 2021 employed a Novel Improved Particle Swarm Optimization (IPSO), SVR, and LSTM hybrid models on the Australian stock market index forecasting. They showed that the proposed IPSO-LSTM model performs best and generalizes well to different stock market indices. Zhou et al., 2021 demonstrated no difference for these specific times series forecasting accuracy using the classical statistics such as ARIMA and ES methods and the deep Generative Adversarial Network Model (GAN) with LSTM.

From the gaps identified in the related works of comparative studies between traditional ARIMA, LSTM, and SVR models, these methods are currently accepted as very promising approaches to many problems in the real world, especially in the stock markets. However, there is a rarity in the application of these methods on the Moroccan stock market. We seek to overcome the drawback by examining the applicability of the three promis-

ing methods for predicting short-term stock of the daily closing price and comparing the performance of the three models. Because of its extensive focus on three algorithms: ARIMA, SVR, and LSTM models, the study stands out from other research articles on a similar topic. The main objective of this chapter is to compare econometric models such as ARIMA and soft computing-based models, including SVR and LSTM, in the Moroccan stock market to reduce error rates in performance measures. Our fundamental contributions to this work are summarized as follows:

- The investigation of general steps of ARIMA modeling using the daily log closing price.

Table 4.1 – A sample of the MASI stock data, from 07^{th} April 2020 to 13^{th} October 2021.

Date	Close	Open	High	Low	Variation %
2020-04-07*	9056.04	9240.00	9279.12	9051.22	-1.99
2020-04-08	9183.80	9056.04	9196.91	9056.04	1.41
⋮	⋮	⋮	⋮	⋮	⋮
2021-10-12	13160.56	13145.58	13165.08	13145.58	0.11
2021-10-13	13187.73	13160.56	13188.80	13142.72	0.21

*Format of date: yyyy-mm-dd

- Building the optimal learning models using the LSTM network and SVR models. The LSTM hyperparameters are updated adaptively by our series. Also, test the capabilities of SVR kernels based on their parameters.
- Comparing the predictive capacities of traditional forecasting techniques and ML-based algorithms to see which sequence of models outperforms others in forecasting log closing price behaviour.

To achieve these objectives, this chapter is organized into the following sections: Section 4.2 describes the stock index and the theoretical background of the methods used in this study, which include ARIMA, SVR, and LSTM models. Section 4.3 presents the empirical results and discussions of each model obtained in this research. Finally, Section 4.4 outlines conclusions and recommendations for future works.

4.2 Data and Methodology

This section describes the dataset and the mathematical background of the traditional and ML algorithms. The basic knowledge of the ARIMA, SVR, and LSTM models is presented. In addition, the performance measures for the comparison between the three models.

4.2.1 Dataset description and preprocessing

MASI Dataset

The Moroccan stock market is one of the most efficient exchanges in the Middle East and North Africa (MENA) region. In 1929, the CSE [1] was created. The CSE calculates two

1. https://www.casablanca-bourse.com

major indices to measure the performance of listed companies: the MASI and the Moroccan Most Active Shares Index (MADEX). We chose MASI as a proxy for the CSE since it represents all listed shares. The MASI is a free-float capitalization-weighted index generally considered the Maghreb region's most important stock market index. The MASI index comprises the 77 most liquid values negotiated in the continuous system, with the highest cash trading volume during the control period. It accounts for 22 sectors of the economy (Belcaid & El Ghini, 2019; El Ghini & Saidi, 2017). The data used in this work are the historical daily prices of the MASI index. Each financial time series data set includes the following variables: Open price, Low price, High price, Close price, and Variation % or Change. The Open price is the opening price of the index at the beginning of the trading day, the Low price represents the minimum MASI during the trading day, the High price represents the maximum MASI during the trading day, and the closing price indicates the MASI when the market closes (see Table 4.1).
In this work, the daily closing price is chosen to represent the MASI. This is because the closing prices reflect all the activities of the day's index. We use the daily closing price of the MASI index, from 07 April 2020 to 13 October 2021, from Investing Website [2], with a total of 377 daily observations (5 days per week) in Moroccan Dirham (MAD). The CSE is open for trading Monday through Friday from 9:00 AM to 3:30 PM (GMT/GMT+1 in the summer). We checked that the original MASI time series closing price data was free of missing values and outliers. Figure 4.1 represents the natural logarithms of the series and indicates that the time series show peaks and swings, i.e., large fluctuations (volatil-

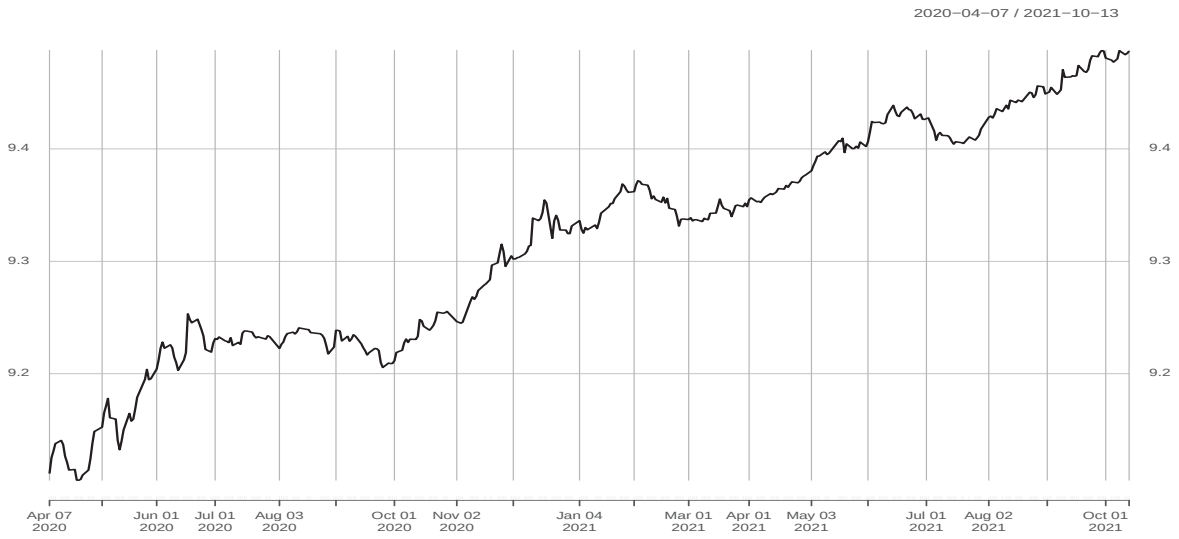


Figure 4.1 – Logarithms of MASI Index daily closing price.

ity) vary over time. Using natural logarithms has effectively controlled specific statistical characteristics and data attributes, including variance stabilisation and improved linear relationships. We can also see that the logarithm of closing prices has a generally upward trend over time. There is no seasonality (no repeating patterns in the series). The observed increasing trend shows that the series is not stationary. Considering our series is non-stationary, the first difference is sufficient to make it stationary. To fit an ARIMA model for log-transformed data, see Figure 4.2. This transformation helps achieve more reliable results, especially when dealing with volatile financial data.

2. https://fr.investing.com/indices/masi-historical-data

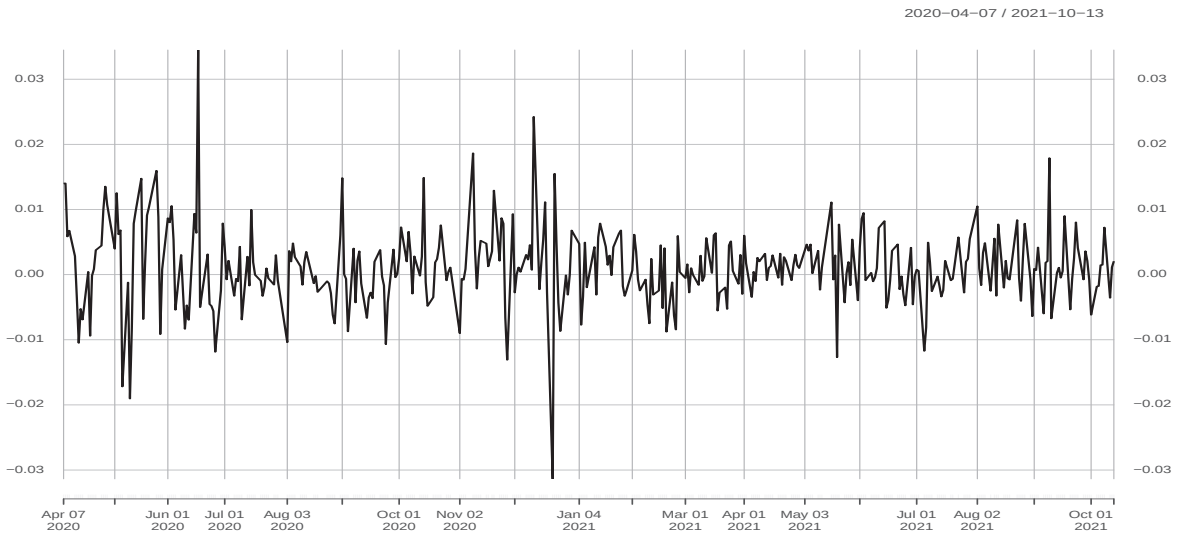


Figure 4.2 – 1^{st} Differences of the Logarithms of MASI Index daily closing price.

Data Preprocessing

This phase is necessary as it allows data preparation and makes them to our needs. Our research used different libraries for developing our models on **R** software (R Core Team, 2018). We opted for R in our chapter due to its specialized time series analysis and financial forecasting packages. Its robust tools like "`forecast`", "`xts`", and a supportive community of experts enhance its suitability for implementing diverse forecasting econometric models. The natural logarithms of the series will be modeled and predicted using the ARIMA, SVR, and LSTM models. Data preprocessing is a crucial step in ML. Models and forecasts benefit from high-quality data (Awad & Khanna, 2015; Géron, 2019). The SVR method was fed with one-day lagged values so that they can be used to predict the next days of logarithms closing price. Otherwise, in the LSTM method, we have normalized the logarithms of the MASI to real numbers in the range $[0,1]$ by :

$$x^{'}(t) = \frac{x(t) - x_{min}(t)}{x_{max}(t) - x_{min}(t)}, \tag{4.1}$$

where t represents each trading day, $x(t)$ represents raw input data that are the log closing prices of the daily data, $x^{'}(t)$ represents the normalized input data that is normalized from $x(t)$, x_{min} and x_{max} represent the minimum and maximum value of the whole time series, respectively (Ji et al., 2021). Employing a normalised series in the LSTM model enhances convergence, stabilises gradient updates, and facilitates capturing intricate temporal patterns in financial time series data. After data preprocessing, the literature review suggests that each financial time series data set was split into two subsets: training (in-sample) and test (out-of-sample) datasets. Earlier studies have used in-sample periods of 60%, 70%, 80%, 90%, and the test of 40%, 30%, 20%, and 10% for out-of-sample periods respectively (Okasha, 2014; Qu & Zhang, 2016; Siami-Namini et al., 2018). In this study, we used about 90% of the data as a training set. Thus, data from the 7th of April 2020 to the 19th of August 2021 was used for model estimation, and the remaining 10% as a test set for the period from the 23rd of August 2021 to the 13th of October 2021, as listed in Table 4.2.

Table 4.2 – Ranges of the training and testing data.

Training set	Testing set
2020-04-07 to 2021-08-19	2021-08-23 to 2021-10-13

4.2.2 ARIMA model for stock price forecasting

Classically, ARIMA models have been widely used for work on stock market forecasting (Adebiyi et al., 2014; Ariyo et al., 2014; Khedmati et al., 2020; M. Kumar & M., 2014). The ARIMA model is a process that forecasts future values of time series using historical observations and white noise or random errors. The ARIMA model addresses non-stationary data by employing differencing to achieve stationarity. Financial time series are not always stationary, as we have seen in Figure 4.1. After the first difference of our log series stationary, as shown in Figure 4.2, we have a confirmation of the Augmented Dickey-Fuller test (ADF), which means that the process has no unit root. As a result, the time series is stationary or has no time-dependent structure. The general form of the ARIMA model of order (p,d,q) combines two models: AR and MA, as follows:

$$\left(1-\sum_{i=1}^{p}\phi_i \mathrm{B}^i\right)(1-\mathrm{B})^d x_t = \delta + \left(1-\sum_{j=1}^{q}\theta_j \mathrm{B}^j\right)\varepsilon_t, \tag{4.2}$$

where x_t is the time series, δ is related to the mean of the x_t however, δ is called the deterministic trend term, and is often omitted from the model unless it is really needed, B is the lag operator, ϕ_i, θ_j are the parameters of the AR, MA, respectively, and ε_t error terms which are i.i.d. from a normal distribution with zero mean and constant variance of σ^2 (i.e. $\varepsilon_t \sim \mathcal{N}(0,\sigma^2)$). The process of modeling ARIMA models based on the Box-Jenkins methodology always contains three iterative steps: Model-identification, Parameter estimation, and Diagnostic checking, see for example (Alizadeh & Ma, 2021; Kao et al., 2020).

4.2.3 SVR model for stock price forecasting

The SVR, one of the powerful supervised ML algorithms, has succeeded in various disciplines (Al-Musaylh et al., 2018; Qu & Zhang, 2016). Consider a given n data points $\{x_i, y_i\}_{i=1}^{n}$ where x_i is the input data ($x_i \in \mathrm{X}$), and y_i is the corresponding output value ($y_i \in \mathrm{X}'$). The idea of SVR is to build a mapping of the high-dimensional-feature space $\phi(x)$, which is nonlinearly transformed from the input space $\mathrm{X} = \mathbb{R}^n$ to a new feature space $\mathrm{X}' = \mathbb{R}^n$, whose dimension depends on the mapping and is not necessarily finite. The SVR algorithm aims to find a function $f(x_i)$ that has the most ε-sensitive errors (see Eq. (4.5)) from the obtained targets y_i for all the data set and their minimization (Kao et al., 2020; Ranković et al., 2014; Yaslan & Bican, 2017). The formulation of an SVR problem is often best described from a geometric perspective in Figure 4.3. From the concepts of the SVR model, a nonlinear regression problem is represented as follows:

$$\begin{aligned} y = f(x,w) &= < w, \phi(x) > + b = w^{\mathsf{T}}\phi(x) + b \\ &= \sum_{i=1}^{n} w_i \phi(x_i) + b, \quad y, b \in \mathbb{R}, x, w \in \mathbb{R}^{n+1}, \end{aligned} \tag{4.3}$$

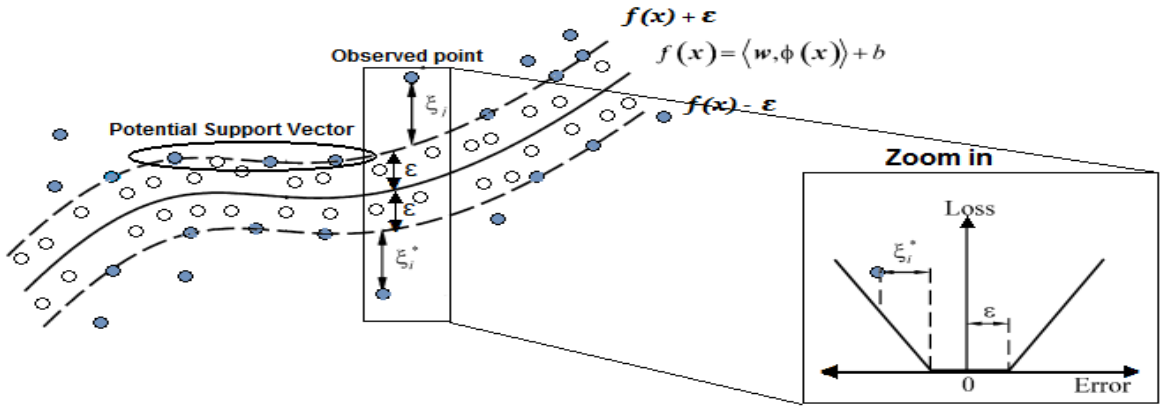


Figure 4.3 – SVR boundary with slack variables and ε-insensitive loss function.

where w^{T} represents the transpose of w weight vector and b is the bias term (or constant), $< \cdot, \cdot >$ represents the dot product in the new feature space. SVR formulates this function approximation $f(x)$ problem as an optimization problem, the weight (w) and the bias (b) are estimated by minimizing the regularized risk function (Sheta et al., 2015) as shown in the following equations (Awad & Khanna, 2015):

$$\mathrm{R(C)} = \frac{1}{2}\|w\|^2 + \frac{\mathrm{C}}{n}\sum_{i=1}^{n} \mathrm{L}_\varepsilon(f(x_i), y_i), \tag{4.4}$$

where the second term $\mathrm{L}_\varepsilon(f(x_i), y_i)$ is known as $\varepsilon-$intensive loss function or the empirical error and given by the following explicit form:

$$\mathrm{L}_\varepsilon(f(x_i), y_i) = \begin{cases} |f(x_i) - y_i| - \varepsilon, & |f(x_i) - y_i| \geq \varepsilon \\ 0\,, & \text{otherwise.} \end{cases} \tag{4.5}$$

Minimizing of Eq. (4.4) is equivalent as a constrained minimum primal optimization problem:

$$\begin{aligned} &\text{Minimize} && \tfrac{1}{2}\|w\|^2 + \mathrm{C}\sum_{i=1}^{n}(\zeta_i + \zeta_i^*), \\ &\text{Subject to} && y_i - w^{\mathsf{T}}\phi(x_i) \leq \zeta_i^* + \varepsilon, \\ & && w^{\mathsf{T}}\phi(x_i) - y_i \leq \zeta_i + \varepsilon, \\ & && \zeta_i^*, \zeta_i \geq 0, \qquad i = 1, \ldots, n \end{aligned} \tag{4.6}$$

where the first term $\frac{1}{2}\|w\|^2$ is called the regularised term that measures the smoothness of the function, C represents the penalty parameter that determines the trade-off between the regularisation term and the empirical risk (Al-Musaylh et al., 2018). ζ_i^* and ζ_i are positive slack variables that represent the distances from the observed point to the corresponding boundary values of ε-tube size of SVRs are tolerated as shown in Figure 4.3. To solve the above optimization problem in Eq. (4.6), we use the Lagrange multiplier and KKT conditions (Awad & Khanna, 2015). The derivation of the SVR function Eq. (4.3) can

be expressed in its general form as:

$$f(x) = f(x, \alpha, \alpha^*) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \mathrm{K}(x_i, x) + b,$$

Subject to:

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) = 0, \qquad 0 \leq \alpha_i, \alpha_i^* \leq \mathrm{C}, \qquad i = 1, \ldots, n \tag{4.7}$$

where α_i and α_i^* are Lagrangian multipliers (Ranković et al., 2014). Therefore, $\mathrm{K}(x_i, x) = \phi(x_i) \cdot \phi(x)$, $\mathrm{K}(.,.)$ is the kernel function that satisfies Mercer's condition (Sheta et al., 2015). The most common kernel functions used in the literature, (Qu & Zhang, 2016; Yaslan & Bican, 2017) are presented in the Table 4.3. Where d is the degree of the polynomial kernel, σ^2 is the bandwidth of the RBF kernel, and κ and δ are kernel parameters that can be tuned. In our case, considering a time series $\{x_t\}_{t=1}^{n}$, the SVR method was fed with

Table 4.3 – Some popular kernel functions.

Kernel function name	$\mathrm{K}(x_i, x)$
Linear	$x_i^\top x + 1$
Polynomial	$(x_i^\top x + 1)^d$
Hyperbolic tangent (Sigmoid)	$\tanh\left(\kappa x_i^\top x + \delta\right)$
Gaussian RBF	$\exp\left(\frac{-\|x_i - x\|}{2\sigma^2}\right)$

values lagged by one day, and the time series had to be transferred into an autocorrelated data set. The decision to use a one-day lagged value in the SVR model is grounded in capturing immediate market dependencies and momentum. This choice balances theoretical relevance, model complexity, and the practical trade-off between capturing short-term effects and avoiding overfitting. In other words, if $\{x_t\}$ is the target value of the forecast, the previous values $\{x_{t-1}, x_{t-2}, \ldots, x_{t-p}\}$ must be the correlated variables of the input, where p is an integration dimension. Training the model and setting the parameters are two elements that streamline the model-building and evaluation process for the SVR model. The three essential steps to fit the model are:

— Evaluate the effect of model tuning parameters on performance via resampling.

— Select the optimal parameters set.

— Estimate model performance from the training set.

The two performance metrics are used to determine the best settings. By default, RMSE and the MAE are computed for regression, and the parameter values of the SVR model are chosen using both metrics, with their expressions in (2.54) and (2.56).

The k-fold Cross-Validation (CV) approach (once or repeatedly) calculates the average prediction error rate after evaluating the SVR model on a different subset of the training data. K-fold CV is a robust method for estimating each kernel's accuracy in the SVR

model. A k-fold CV is often performed with $k = 5$ or $k = 10$. However, we use a 10-fold CV repeated 10 times to estimate the prediction error for each kernel to determine the best model and parameter appropriate for the SVR model, as illustrated in Figures 4.4, 4.5, 4.6.

In these figures, the RMSE and MAE were used to select the optimal model using the

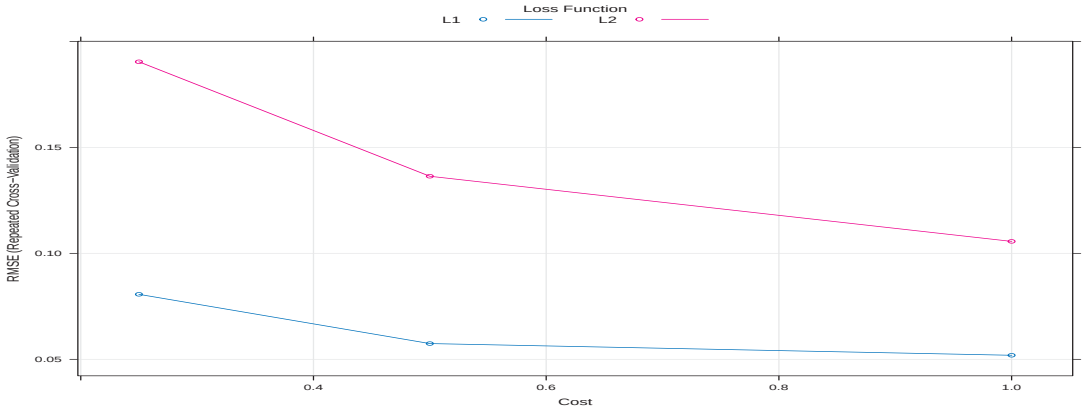


Figure 4.4 – L_1 and L_2 regularized SVR (dual) with linear kernel.

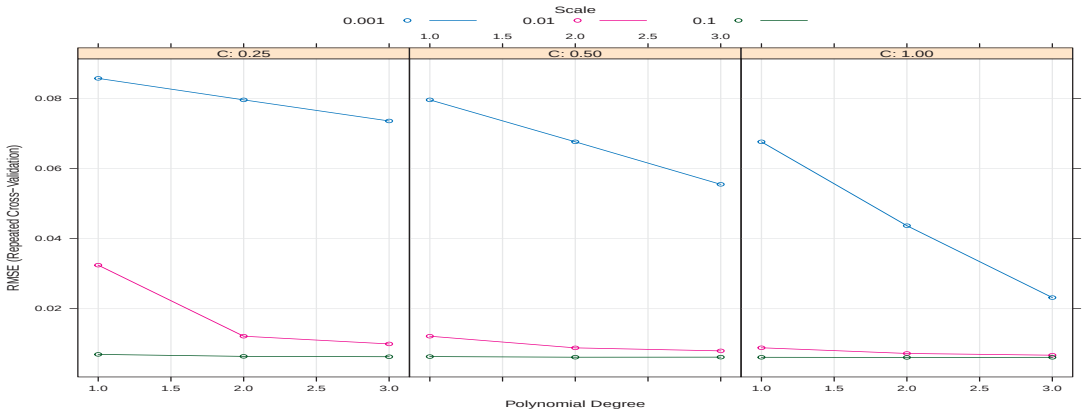


Figure 4.5 – SVR with polynomial kernel fitting parameters (degree, scale, C).

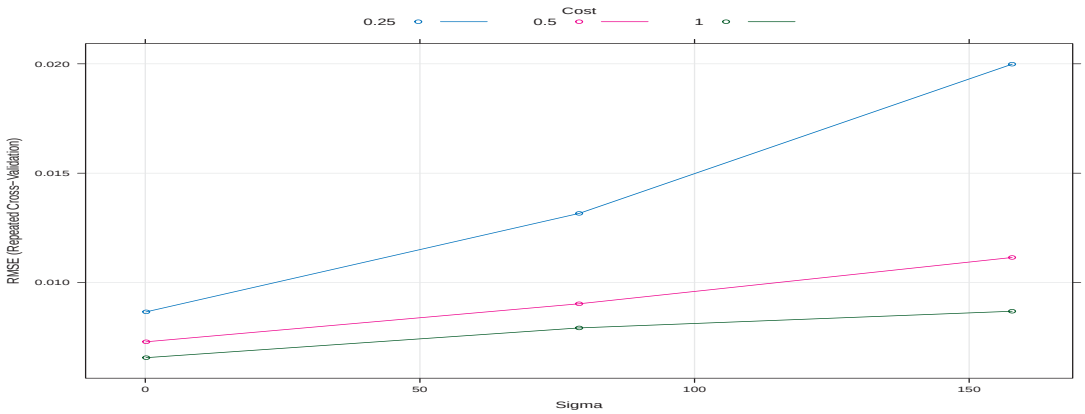


Figure 4.6 – SVR tuning parameters (sigma, C) with RBF kernel.

smallest value. From resampling results across tuning parameters, and pre-processing

was scaled to $[0,1]$, which will be useful when we compare resampling levels between models, we can see in Figure 4.4 the final values used for the SVR model with linear kernel were C(cost) = 1 and Loss = L_1. In Figure 4.5, the final values used for the SVR model with a polynomial kernel with degree = 2, scale = 0.1, and C = 1. In Figure 4.6, the final values of the tuning parameters used for the SVR model with RBF kernel were sigma = 0.196 and C = 1. Based on the summary results of each model, we found that the SVR model with a polynomial kernel is significant. As it has a small RMSE = 0.006, MAE = 0.004. In addition, the degree, scale, and C tuning parameters are optimal. However, we chose the SVR (dual) with linear kernel for model validation on test data and comparison with other ARIMA and LSTM models since it is the most explanatory in terms of accuracy.

4.2.4 LSTM model for stock price forecasting

An RNN refers to a neural network of the most advanced DL with recurrent connections. The recurrent connections learn the dependencies between input data that is sequential or time-series. The main problem of an RNN is the exploding/vanishing gradient during the training phase (Elsaraiti & Merabet, 2021), which typically arises when the model cannot learn long-term dependencies. LSTM networks are explicitly designed to learn order dependence in sequence prediction problems, which helps in stock price forecasting, as past prices play an important role in predicting future stock market prices. This structure potentially facilitates the network to remember information for longer pe-

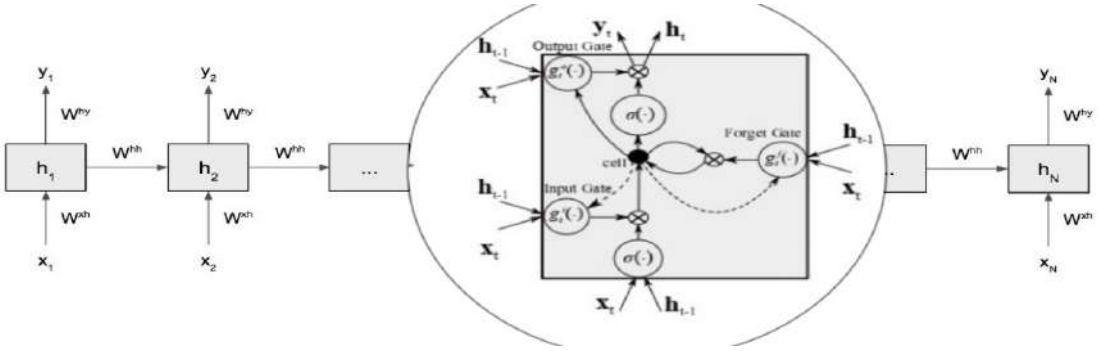


Figure 4.7 – Sequences of the LSTM memory block with dashed line indicating time lag (Salehinejad et al., 2017).

riods using the "memory". A diagram illustrating the sequence structure of this model is shown in Figure 4.7 above. As can be seen, the typical LSTM cell consists of memory and gates, the input and hidden state correspond to x_t and h_t, respectively, at time t. The input, forget, cell, and output gates correspond to g_t^i, g_t^f, g_t^c, and g_t^o, respectively, at time t. The calculations for each gate are performed using the following formulas (Baek & Kim, 2018; Van Houdt et al., 2020):

The input gate of LSTM is defined as follows:

$$g_t^i = \sigma(W_{Ig^i} x_t + W_{Hg^i} h_{t-1} + W_{g^c g^i} g_{t-1}^c + b_{g^i}), \tag{4.8}$$

where W_{Ig^i} and W_{Hg^i} are the weights matrix associated with x_t and h_{t-1}, respectively, $W_{g^c g^i}$ is the weight matrix from cell activation to the input gate, while b_{g^i} is the bias of the input gate.

The forget gate of LSTM is defined as follows:

$$g_t^f = \sigma(W_{Ig^f} x_t + W_{Hg^f} h_{t-1} + W_{g^c g^f} g_{t-1}^c + b_{g^f}), \tag{4.9}$$

where W_{Ig^f}, W_{Hg^f}, and $W_{g^c g^f}$ are the weights associated with x_t, h_{t-1}, g^c_{t-1}, respectively, while b_{g^f} stands for the bias of the forget gate.

The output gate of LSTM is defined as follows:

$$g^o_t = \sigma(W_{Ig^o} x_t + W_{Hg^o} h_{t-1} + W_{g^c g^o} g^c_{t-1} + b_{g^o}), \tag{4.10}$$

where W_{Ig^o}, W_{Hg^o} and $W_{g^c g^o}$ are the weights matrix associated with x_t, h_{t-1} and g^c_{t-1}, respectively, while b_{g^o} represents for the bias of the output gate.

The cell gate of LSTM is defined as follows:

$$g^c_t = g^i_t \tanh\left(W_{Ig^c} x_t + W_{Hg^c} h_{t-1} + b_{g^c}\right) + g^f_t g^c_{t-1}, \tag{4.11}$$

where W_{Ig^c} and W_{Hg^c} are the weights associated with x_t and h_{t-1}, while b_{g^c} denotes for the bias of the cell gate.

The hidden state of LSTM is defined as follows:

$$h_t = g^o_t \tanh\left(g^c_t\right). \tag{4.12}$$

The logistic sigmoid activation function for gating is denoted as σ in Figure 4.7 and computed by $\sigma(x) = \frac{1}{1+e^{-x}}$, where the output is in the range of $[0,1]$. If the output value is "0" it means "no information can enter". Otherwise, if the value increases to "1" this means "everything can be fully reflected"; the LSTMs can learn longer temporal patterns (Ji et al., 2021; Siami-Namini et al., 2018). Our research uses the hyperbolic tangent activation function tanh, often used as the input and output block activation function. The MASI log closing price series should be in a supervised learning mode. LSTM shows a target variable Y and a predictor X exist. For a data set lagged by k, we transform the series by shifting it and using the value at a time $(t-k)$ as input and the value at time t as output, then a normalization by Eq. (4.1). The loss function is MSE, the optimization algorithm is Adaptive Moment Estimation (ADAM), and learning rate decay is specified for each update. Finally, we evaluated the model's performance using accuracy as a metric. Table 4.4 shows the structure of our LSTM model in more detail. `Keras` (F. Chollet et al., 2015) and `Tensorflow` (Abadi et al., 2016) were used to implement the model suggested in this chapter. Our LSTM model was trained based on the optimizer ADAM, the learning rate was 0.05, and the decay rate was 10^{-6}. During the training process, the data set was divided into several batches sized at 1.

Table 4.4 – Parameter settings of the LSTM neural network

Layer	Type	Output Shape	# Param
1	LSTM	(1,1)	12
2	Dense	(1,1)	2
3	Dropout	(1,1)	0
Total params: 14			
Trainable params: 14			
Non-trainable params: 0			

We adjust the LSTM model and reset the network states after 50 epochs. Then, the validation data was divided from the training data, and the proportion was set to 10%. To avoid overfitting, we adopted the early stopping and dropout strategy for a 0.02 rate in our model. As illustrated in Figure 4.8, the LSTM training and validation losses through epochs. It can be noted that training and validation losses intersect over a certain number of epochs, so we can confirm that our model is satisfactory.

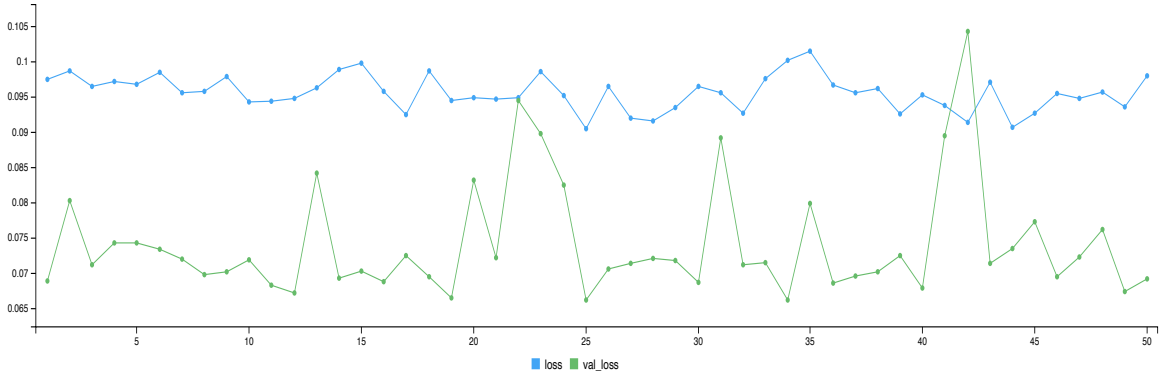


Figure 4.8 – LSTM model fitting for training and validation loss (50 epochs).

4.2.5 Performance measures for comparison

We explore the new approaches of ARIMA, SVR, and LSTM in predicting stock prices. We will then present the methodology we followed in Figure 4.9 to use the three models to model and predict the closing price of the MASI log series. Finally, we will use the predicted results to compare the models with them. To assess the performance of the

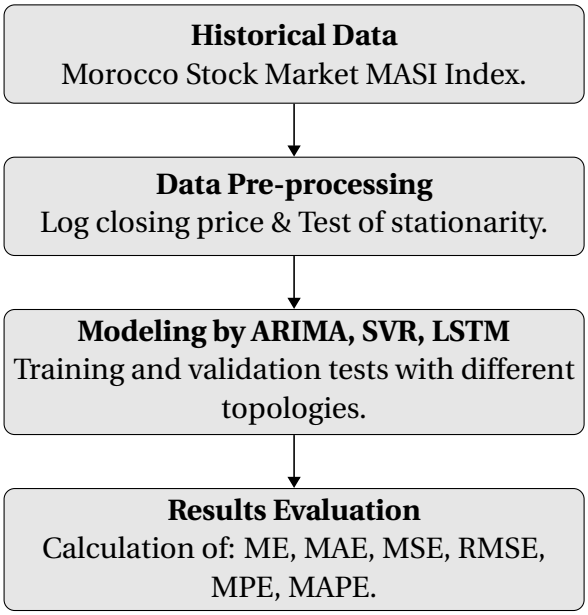


Figure 4.9 – Block diagram representing the proposed methodology.

developed financial forecasting model, several evaluation metrics are used to evaluate the accuracy of the stock market prediction models (M. Kumar & M., 2014; Sheta et al., 2015). These different error criteria are adopted to measure the deviation between the predicted time series values to that of the actual value. This chapter uses six evaluation metrics: ME, MPE, MAE, MSE, RMSE, and MAPE to compare the performances of our proposed methods on stock price prediction of series log MASI closing price. The previous equations (2.53), (2.57), (2.54), (2.55), (2.56), (2.58) are the formulas for calculating the six performance metrics (Al-Musaylh et al., 2018; Ji et al., 2021; Khashei & Hajirahimi, 2019).

Based on these accuracy measures, the lower values of the performance measures correspond to a robust prediction for the models' comparison. The major advantage of using RMSE is that it penalized large errors (Siami-Namini et al., 2018). In addition, the MAPE

value should be less than 10% (Kao et al., 2020). The advantages and disadvantages of the three different methods in this study are summarized in Table 4.5.

Table 4.5 – Comparison of the main advantages and limits of LSTM, ARIMA, and SVR models.

Method	Advantages	Limitations
LSTM	- It can model long-term sequence dependencies better than a simple RNN. - It is more robust to the problem of short memory than "Vanilla" RNNs (vanishing gradients) Salehinejad et al., 2017.	- It adds more parameters to learn, which increases the complexity of the calculation compared to RNN. - Since there are more memory cells than in the "Vanilla" RNNs, the amount of memory required is higher Salehinejad et al., 2017.
SVR	- SVR is good at remembering data since it employs Support Vectors, a subset of training points. - It performs effectively when the number of samples exceeds the number of dimensions. - It is most robust in high dimensions and productive when there is a clear margin of separation.	- SVR has two weaknesses. Firstly, it takes more time to process large amounts of data because of the longer training periods. Secondly, we don't have a specific rule to choose the model parameters; rather, we use the trial and error process Ranković et al., 2014.
ARIMA	- ARIMA modeling is used for its stability and simplicity, especially when building univariate time series models. - Given linear time series, the ARIMA model performs better. - It can improve forecast accuracy while limiting parameter numbers to a minimum.	- Dedicated specifically to time series. Otherwise, the model does not work. - ARIMA is useful for short-term forecasting and sensitive to changes that have occurred over time. - Used for small amounts and linear data, it does not have many techniques for many variables and complex relationships.

4.3 Experimental results and discussions

In this section, we present the results obtained by running different models on the time series of ARIMA, SVR, and LSTM models in subsections 4.2.2, 4.2.3, and 4.2.4. In the next three subsections (4.3.1, 4.3.2, 4.3.3), we report the prediction results for the Moroccan stock market. In the first one, we present a prediction for each model. In the second, we compare the MASI log closing price forecast between three models for 37 trading days of the test set. In our experiments of methods, as mentioned in Table 4.2, the log closing price of MASI between April 7, 2020, and August 19, 2021, is used as the training set, while the log closing price between August 23, 2021, and October 13, 2021, is used as the test set. As our series period is after the COVID-19 crisis, we investigated the effect of a crisis on the series log closing price of the proposed three models. Results of the linear model

are shown initially, followed by the results of ARIMA, and finally, results of the non-linear models (SVR, LSTM) are discussed in the subsections below.

4.3.1 Result of the developed ARIMA model

We look to identify the orders of the ARIMA model of the training data. Since the stationary series is essential for the ARIMA model, the ACF and the PACF are used to determine if a specific series is stationary or non-stationary and how many AR, MA terms p, and q, respectively are necessary. Based on Figure 4.10, we can see the training ACF decays rapidly from its initial value of unity at zero lag, and the training PACF tails off as exponential decay after lag 1. The candidates selected from the visualization of the ACF and

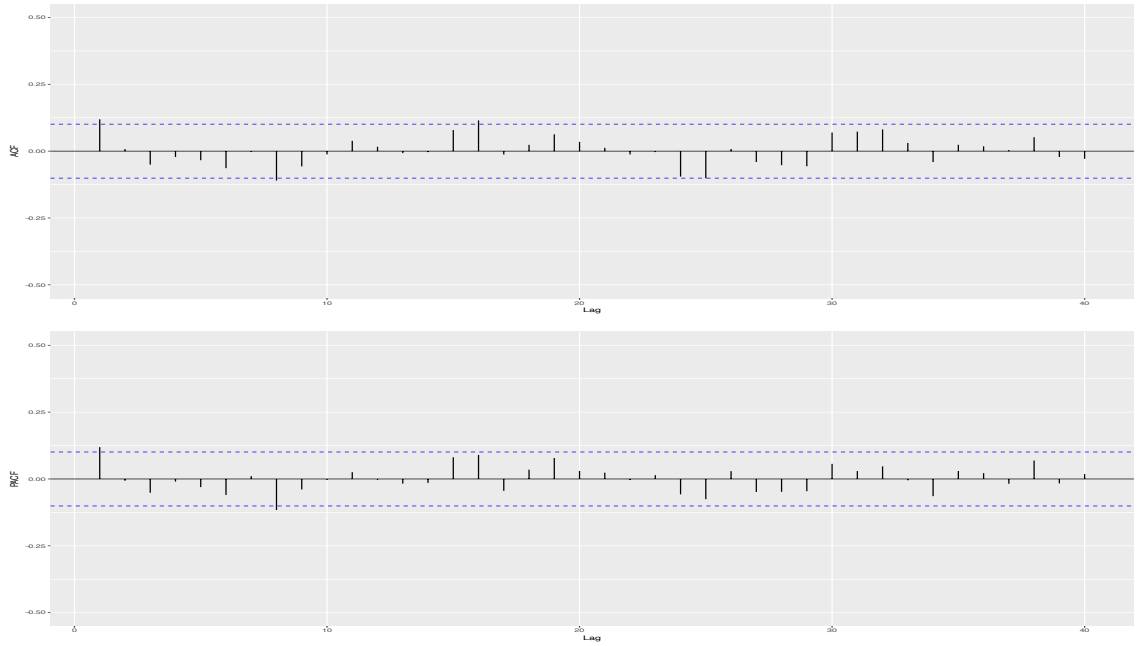


Figure 4.10 – The ACF (top) and PACF (bottom) plots of log series difference $d = 1$.

PACF plots are ARIMA(1,1,1), ARIMA(1,1,0), and ARIMA(0,1,1). After specifying the model ARIMA(p,d,q) in the previous step, the fitting method used to estimate the parameters of our model is the Maximum Likelihood Estimation (MLE) or minimizing Conditional-Sum-of-Squares (CSS). The best ARIMA model is identified by considering the following information criteria: BIC and AIC. For more detail, see Zhou et al., 2021. Based on the diagnosis of the smallest BIC and AIC, the ARIMA(0,1,1) and ARIMA(1,1,0) models are better for the log MASI stock index than the ARIMA(1,1,1) model. As a result, we found the ARIMA(1,1,0) model as the best model to fit the log closing prices of MASI stock. The outcomes of the parameter estimations for both ARIMA models in Figure 4.6 indicate a favorable alignment with the training series, as demonstrated by the statistical significance of all coefficient p-values. The standard errors (SE) of the estimated parameters are in (.), and the following models were obtained:

— **ARIMA(0,1,1) model:**

$$(1-\mathrm{B})\hat{x}_t = (1 - \underset{(0.053)}{0.124}\mathrm{B})\varepsilon_t, \tag{4.13}$$

— **ARIMA(1,1,0) model:**

$$(1 - \underset{(0.054)}{0.13}\ \mathrm{B})(1-\mathrm{B})\hat{x}_t = \varepsilon_t, \tag{4.14}$$

Table 4.6 – Estimation results of the ARIMA(1,1,0) and ARIMA(0,1,1) models.

Coefficients	**Estimate**	**SE**	**t.value**	**p.value**
ar1	0.1285	0.0543	2.3674	0.0185***
constant	0.0010	0.0004	2.6476	0.0085***
ma1	0.1242	0.0526	2.3603	0.0188***
constant	0.0010	0.0004	2.6968	0.0074***

where $\varepsilon_t = x_t - \hat{x}_t$ is the difference between the actual value, which is the differenced logarithm of the MASI index of the closing price and the fitted values of the training series, with AIC = −2488.56, BIC = −2480.91, and RMSE = 0.006 for ARIMA(0,1,1) model, and AIC = −2489.05, BIC = −2481.39, and RMSE = 0.006 for ARIMA(1,1,0) model. The third step of the Box-Jenkins methodology is the diagnosis to confirm the adequacy of the ARIMA model developed.

Table 4.7 – Residual normality tests ARIMA models.

	Jarque Bera Test, H_0: Normal distribution		
Models	**df**[3]	**X-squared**	**p-value**
ARIMA(0,1,1)	2	3.633	0.163***
ARIMA(1,1,0)	2	3.362	0.186***
	Shapiro-Wilk test, H_0: Normal distribution		
Models		**W**	**p-value**
ARIMA(0,1,1)		0.988	0.103***
ARIMA(1,1,0)		0.989	0.167***
	Anderson-Darling test, H_0: Normal distribution		
Models		**A**	**p-value**
ARIMA(0,1,1)		0.454	0.268***
ARIMA(1,1,0)		0.378	0.405***

Note: *** indicates significance at the 5% level.

To evaluate the adequacy of the models, we will study the distribution of the residuals. From the results of Table 4.7, all p-values greater than 5% indicate no significant evidence to reject the H_0. Based on the Jarque-Bera, Shapiro-Wilk, and Anderson-Darling tests, the residuals are consistent with a normal distribution. We have already seen a small gap between the distribution of the residuals and the normal distribution, as shown in Figure 4.11, which means the existence of kurtosis that comes from the extreme positive and negative values in the residuals. But overall, satisfies the normality assumption. From

3. Degrees of Freedom.

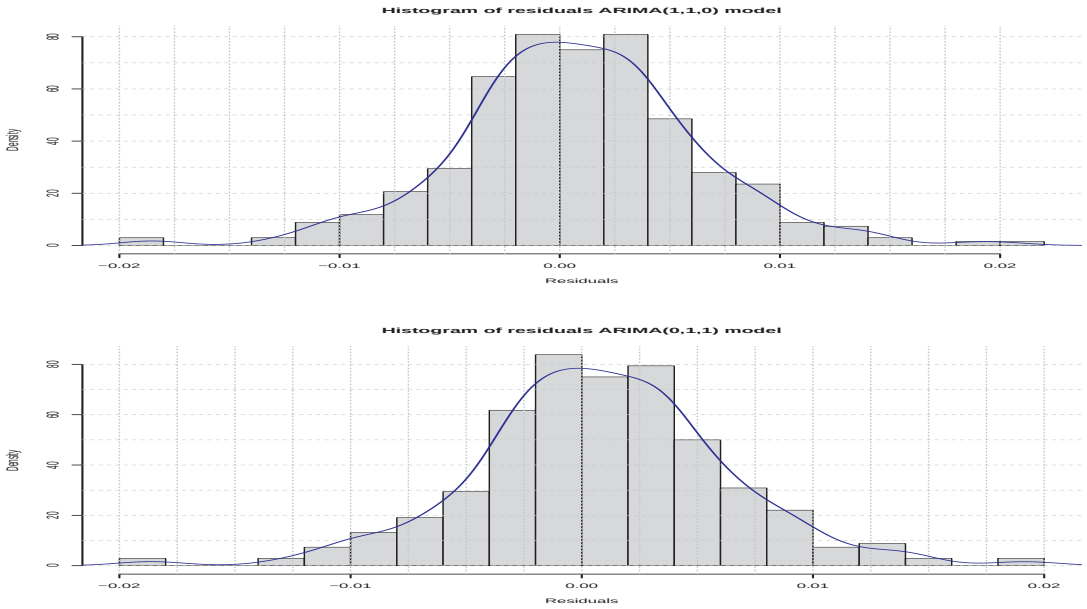


Figure 4.11 – Diagnostic plots for the standardized residuals of the ARIMA(0,1,1) (top) and ARIMA(1,1,0) (bottom).

the results in Table 4.8 of the ARCH Lagrange Multiplier (LM) test and the Box-Pierce test or Ljung-Box test. So, there is no ARCH effect, and the residuals have no autocorrelation. These plots and the significance test of the coefficients suggest that the two ARIMA models adequately fit the time series of the MASI closing price index's logarithms. In predicting the results, we will only use the ARIMA(0,1,1) because they have very close values to the actual values. Using the ARIMA(0,1,1) model, log closing price forecasts were made

Table 4.8 – ARCH test and Box-Pierce test.

	Box-Pierce test, H_0: No Autocorrelation		
Models	**Chi-squared**	**p-value**	**df**
ARIMA(0,1,1)	2.741	0.098***	1
ARIMA(1,1,0)	2.299	0.129***	1
	ARCH LM-test, H_0: No ARCH effects		
Models	**Chi-squared**	**p-value**	**df**
ARIMA(0,1,1)	0.389	0.533***	1
ARIMA(1,1,0)	0.347	0.556***	1

Note: *** indicates significance at the 1% level.

for all 37 days of the test set. Figure 4.12 shows the log closing price series (blue line) and forecast values (black line) using the ARIMA model. From the graph, the predicted values are close to the actual values and move in the direction of the forecast values in many instances, as shown in the performance of the selected ARIMA model in Figure 4.12. Our model cannot predict the largest peaks in the test set series, especially in the last days of August 2021. This means that the model prediction is unsatisfactory because there is a gap between the actual and predicted values of the non-linear patterns.

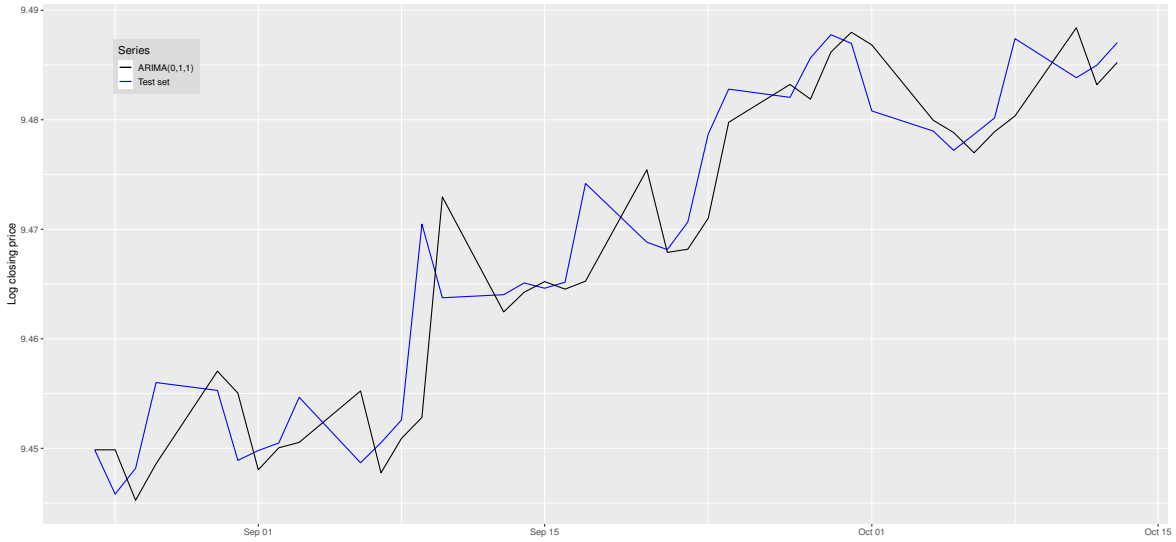


Figure 4.12 – Forecasting of MASI using the ARIMA model (37 days).

4.3.2 Result of the developed SVR model

Figure 4.13 represents the actual and predicted series of log closing prices for the testing set of the developed SVR model. The SVR (dual) with the linear kernel is used to develop the training set model and the forecast test set compared to the SVR model with the polynomial kernel. As a result, we see good precision in predicting the SVR (dual) model with a linear kernel. Therefore, we used a GS (Hollis et al., 2018; Sheta et al., 2015) in the package `caret` to obtain the values of the parameters cost and Loss that have a high influence on the accuracy of the SVR model. It was found that the best performance for the training set model can be obtained with cost = 1, Loss = L1, RMSE = 0.1056, and MAE=0.0877. The results shown in Figure 4.13 indicate that the tendencies of the pre-

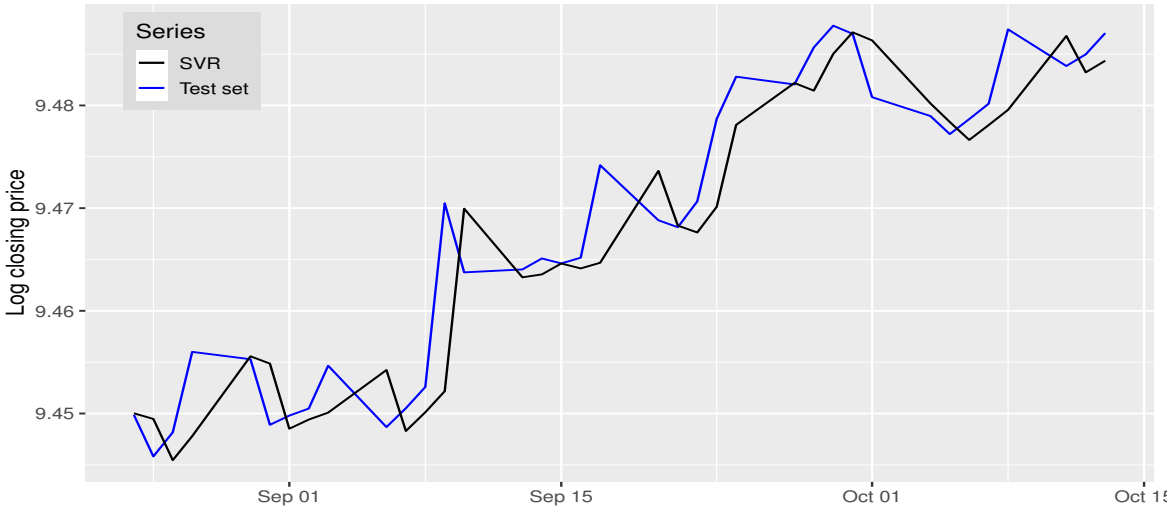


Figure 4.13 – Forecasting of MASI using the SVR model (37 days).

dicted value curve are identical to those of the actual value curve. As can be observed, the SVR model cannot forecast peaks or sudden changes in the series, which shows that the predictions are unsatisfactory.

4.3.3 Result of the developed LSTM model

Technically, when predicting the series at the time $(t+1)$, the prediction LSTM model feeds the daily log closing price of the MASI index for 37 trading days, from time $(t-37)$ to time t, into the input nodes. This window moves with the all dataset. The 37 first days are fed into the network during training, and the 38^{th} is set as the target value for supervised learning. In our experiment to build the forecasting model for MASI, we chose the number of LSTM output units as 1 for each of the three input layers (based on Table 4.4). Then after, the output forecast is calculated. It's important to note that the number of output units in each layer is determined by trial and error Baek and Kim, 2018. Figure 4.14 gives the graph of the prediction against the test data of the actual log closing price to see the performance of the LSTM model. As can be seen from the figure, the LSTM model has a minimal magnitude to the test set. But, the prediction of the model keeps the tendency of the original test set. The model can capture non-linear fluctuations or patterns, which implies that the predictions are satisfactory. However, LSTM can predict MASI's log closing price trend very well. Figure 4.15 assembles all the curves so that we

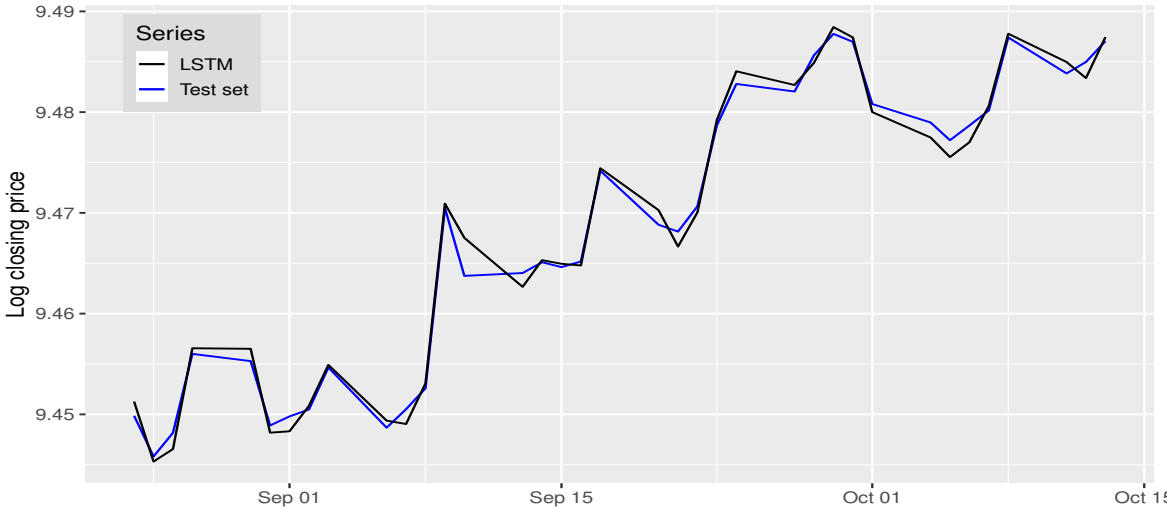


Figure 4.14 – Forecasting of MASI using the LSTM model (37 days).

can make comparisons between the best model produced of the three prediction models on the MASI test set. The green, red, and blue lines, respectively, represent the ARIMA model, the LSTM model, and the SVR model. The black line indicates the actual closing price of the 37 days. As can be seen, the forecast curve of LSTM is very close and identical to the true value curve for the MASI index in the trend compared to other models (ARIMA, SVR), which indicates the robustness of the LSTM model for the stock market.

4.3.4 Evaluation criteria comparison of methods

For concluding the results section, we make a discussion of performance criteria comparison of methods to test the accuracy of each model. The ARIMA, SVR, and LSTM models are estimated for the training log of the closing price of the MASI index. The model estimation selection process is followed by an empirical evaluation based on the out-of-sample test set. Table 4.9 shows the prediction results of the log closing price (37 days) for each model based on the performance of the models as measured by six statistical

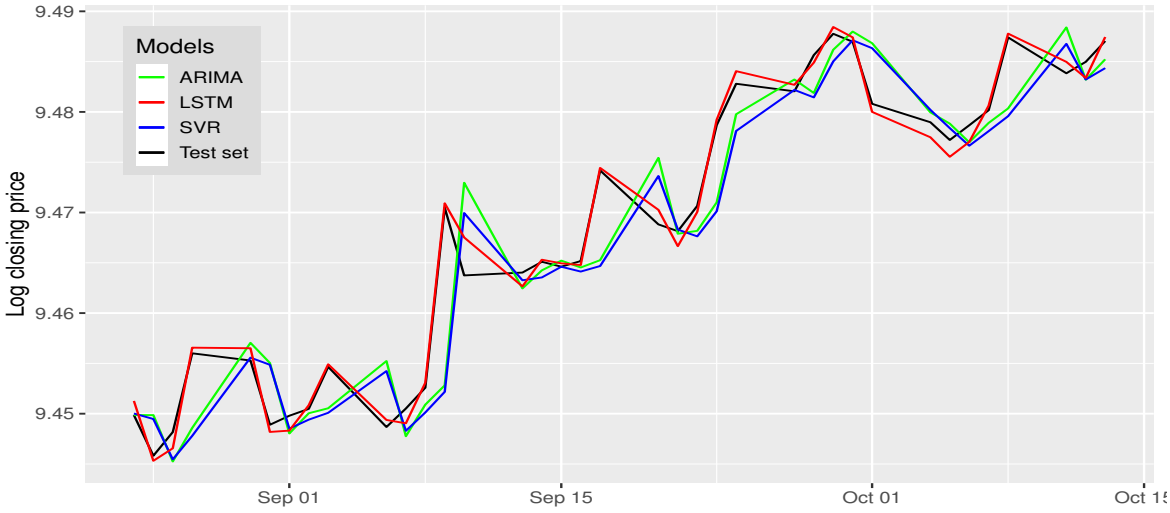


Figure 4.15 – Comparison of forecasting MASI using all models (37 days).

evaluation criteria. These values indicate that the LSTM model is more efficient than the other models. This is because the values of ME, MAE, RMSE, MPE, MAPE, and MSE of the LSTM model are lower than other models, such as ARIMA and SVR. The SVR performs better than the other two models, i.e., SVR and ARIMA. It is clear from the above results that, on average, the LSTM model is the best model for forecasting the daily log closing price of the MASI index.

Table 4.9 – Performance metrics of forecasting for log closing price from Aug. 24^{th} - Oct. 13^{th} 2021.

Models	**ARIMA(0,1,1)**		**SVR**		**LSTM**	
Error measure	Train	Test	Train	Test	Train	Test
ME	0.0009	0.0167	0.0006	0.0015	0.0011	0.0012
MAE	0.0044	0.0178	0.0007	0.0035	0.0011	0.0012
RMSE	0.0061	0.0218	0.0009	0.0050	0.0015	0.0016
MPE (%)	0.9732	17.6595	0.6729	1.5792	1.2659	1.3000
MAPE (%)	4.7436	18.8381	0.7513	3.7468	1.2836	1.3074
MSE*	37.6547	475.6664	0.7531	25.1960	2.3714	2.4619

* $\times 10^{-6}$.

4.4 Conclusions

In this research work, we clarify the opinions reported in the literature on the superiority of ML models over the classical model in financial time series forecasting problems. We investigate the predictive capability of ML-based algorithms, such as SVR and LSTM

models, and traditional algorithms, such as the ARIMA model. The performance of these models is compared mutually for modeling the behaviour and predicting the trend of one of the most known indexes in the Moroccan stock market. We fit the linear ARIMA model and the non-linear LSTM and SVR models to the MASI index of the log closing price time series data and used these models to forecast future observations (37 days). The results were validated and compared using a variety of evaluation criteria. These results confirmed our hypothesis that the LSTM model outperforms the ARIMA and SVR models. This type of study has been few up until now. Our research opens a new window for comparing these models on the Moroccan stock market. The limitation of this work is that the optimal parameters for the LSTM model are uncertain, as the parameters must be chosen by trial and error. In future work, we will investigate the results of LSTM forecasts as input to the Markov decision model in algorithmic trading for individual investors. This can help construct strategies for the main Moroccan financial companies on the CSE.

Chapter 5

Comparing Machine Learning Methods-SVR, XGBoost, LSTM, and MLP-For Forecasting the Moroccan Stock Market

" He who forecasts the weather by the moon, will often be right and often wrong. "

–Anyonomy

Contents

5.1 Introduction

The 2008 recession, the stock market crash of 2015, the COVID-19 Pandemic, and the Russian Invasion of Ukraine are some of the most recent crises which have had an immense impact on the financial markets and the destruction of wealth worldwide. Modeling and forecasting the stock market is a challenge that many engineers and financial researchers face. The literature review examined studies on stock market prediction using ML models. It concluded that DL was the most commonly utilized model for forecasting stock price trends (Mintarya et al., 2023; Ozbayoglu et al., 2020). Traditional econometric methods might require improved performance in relevant nonlinear time series and may not be appropriate for directly forecasting stock prices because of their volatility (R. Li et al., 2022). However, for complex nonlinear financial time series, methods such as SVR, XGBoost, MLP, and LSTM can detect nonlinear relationships in the forecasting of stock prices (T. Kim & Kim, 2019) and achieve better fitting results by tuning multiple parameters (Xu et al., 2022). Hyperparameter optimization or tuning in ML refers to selecting the most appropriate parameters for a particular learning model (Yang & Shami, 2020). Some studies use GS optimization Rubio and Alba, 2022, while others use Bayesian (Liwei et al., 2021; Xia et al., 2017) or pigeon-inspired optimization algorithms (Al-Thanoon et al., 2022). In this chapter, the GS algorithm is used to optimize the parameters of each model, such as SVR, XGBoost, MLP, and LSTM models. Then, we compare them using seven measures: ME, MPE, MSE, MAE, RMSE, MAPE, and R^2. In (Sai Krishna & McCrae, 2019), it was discovered that using LSTM with MA yields superior results for predicting stock prices compared to SVR, as measured by different performance criteria such as MAE, MSE, MAPE, RMSE, and R^2 values. In their research (Al-Nefaie & Aldhyani, 2022), Al-Nefaie et al. proposed using LSTM and MLP models to forecast the fluctuations in the Saudi stock market. They found that the correlation coefficient for the two models was higher than 0.995. The LSTM model proved to have the highest accuracy and best model fit. In (Goverdhan et al., 2022), the authors compared performance measures such as MSE, MAE, RMSE, and MAPE to forecast stock market trends based on ARIMA, XGBoost, and LSTM. Their tests found that XGBoost performed the best. In a different context, ML methods such as ANN and MLP (Ettayyebi & El Himdi, 2018a) were used to forecast solar irradiance. The results indicate that the MLP model with exogenous variables performs better than the other models. Similarly, in another study (Kumari & Toshniwal, 2021), the predictive performance and stability of eXtreme Gradient Boosting Deep Neural Networks (XGBF-DNN) made it an optimal and reliable model for forecasting hourly global horizontal irradiance using the GS algorithm. The main objective of this investigation is to improve the GS optimization algorithm by optimizing the hyperparameters of ML models. Furthermore, this study compares two ML methods and two DL methods. To achieve this objective, the research includes an in-depth literature review of 20 studies on ML models for stock market prediction. Overall, the chapter aims to contribute to the stock market prediction field by improving the performance of ML models through hyperparameter optimization. Our contribution is highlighted in Table 8.8, which offers a more accurate prediction than other studies. The rest of this chapter is organized as follows: Section 5.2 outlines the suggested methodology for modeling and forecasting financial time series, specifically stock prices. Section 5.3 details the implementation of the study, results, and discussion. Lastly, Section 5.4 offers conclusions and suggests future research directions.

Table 5.1 – Some previous work using ML models for stock market forecasting.

Reference	Location	Year	Benchmark Model	MAPE (%)	RMSE	R^2
Sai Krishna and McCrae, 2019	Irlande	2019	LSTM, SVR, MA	1.03	347.46	0.83
Lu et al., 2020	China	2020	MLP, CNN, RNN, LSTM, CNN-RNN	—	39.688	0.965
Goverdhan et al., 2022	American	2022	ARIMA, XGBoost, LSTM	3.8	6.101	0.961
This study	Morocco	2023	SVR, XGBoost, MLP, LSTM	0.368	3.993	0.989

5.2 Materials and Methods

This section provides an overview of the methodology employed in this research.

5.2.1 Data Collection

This research focuses on modeling and forecasting a new Moroccan Stock Index 20 (MSI 20) composed of the most liquid companies listed on the CSE from various sectors, including Attijariwafa Bank, Itissalat Al-Maghrib, Banque Populaire, LafargeHolcim Morocco, … etc. The MSI 20 is calculated in real time during the business hours of the CSE from Monday to Friday, and from 9:00 a.m. to 3:30 p.m. local time (Limited on weekends and holidays). The research was conducted using the Python software. In addition, several libraries were used, including Matplotlib, Pandas, NumPy, Sklearn, Tensorflow, and Keras. We use daily MSI 20 data to train each model and predict closing prices. Since the launch of the index of length N = 541, we use prices as Input = (Open, High, Low, and Closing prices), see the Table 5.2 below: To train model parameters, we use historical data

Table 5.2 – Sample data (First five days) and descriptive statistics of MSI 20 index daily prices from 18 December 2020 to 09 February 2023.

Trade date	Open (MAD)	High (MAD)	Low (MAD)	Close (MAD)
2020-12-18	946.25	950.56	943.85	944.58
2020-12-21	944.58	944.58	911.51	912.17
2020-12-22	912.17	928.51	906.77	927.79
2020-12-23	927.79	935.71	927.07	932.43
2020-12-24	932.43	932.81	926.58	927.16
Statistics	Open	High	Low	Close
Count	541	541	541	541
Mean	988.86	993.28	984.91	988.84
Std	78.01	77.05	78.35	78.15
Min	775.38	797.01	775.38	775.38
Max	1140.69	1142.56	1135.86	1140.69

of length N,

$$\mathrm{X} = \{\mathrm{X}_t^{(1)}, \mathrm{X}_t^{(2)}, \mathrm{X}_t^{(3)}, \ \ t = 1, 2, \ldots, \mathrm{N}\},$$

where $\mathrm{X}^{(i)}$ with $i = 1, 2, 3$ represents the Open, High, and Low prices, respectively. For the output with a single observation sequence, we use only closing price,

$$\mathrm{Y} = \mathrm{Y}_t, \ \ t = 1, 2, \ldots, \mathrm{N}.$$

5.2.2 Preprocessing

The dataset was divided into 90% of the observations used for training and 10% for model evaluation during testing. Data preprocessing is an essential step in ML that helps achieve competitive results and eliminate metric unit effects. In this case, we normalized using a min-max scale, which scales all variables to a range of $[0, 1]$:

$$\tilde{\mathrm{X}}_t = \frac{\mathrm{X}_t - \mathrm{X}_t^{min}}{\mathrm{X}_t^{max} - \mathrm{X}_t^{min}}, \tag{5.1}$$

where X_t represents the historical data for each feature variable in the time series (Open, High, and Low prices) and the X_t^{max} and X_t^{min} values are the sample's maximum and minimum values.

5.2.3 SVR Model

The SVR model, a new financial time series prediction method, is used to address the challenges of nonlinear regression. We assume a linear relationship exists between X_t and Y_t as in the left side of the Eq. (5.2). To perform nonlinear regression using SVR, the concept consists of creating an $x \longrightarrow \varphi(x)$ transformation that maps the original feature space X, which has N dimensions, onto the new feature space X'. Mathematically, this can be explained by the equation shown below:

$$\mathrm{Y}_t = \mathrm{W} \cdot \mathrm{X}_t + b = \sum_{i=1}^{\mathrm{N}} w_i x_{it} + b_i, \ \xrightarrow{\text{Non-linear}} \ \mathrm{Y}_t = \sum_k \beta_k y_k \mathcal{K}(x_k, x) + b, \tag{5.2}$$

where w_i is the vector of weights and b_i is a bias, β_k is the coefficient of the Lagrange multipliers, and $\mathcal{K}(x_k, x) = \varphi(x_k) \cdot \varphi(x)$ is a kernel function. The most commonly used kernels include linear, Gaussian, and polynomial functions (Ranković et al., 2014).

5.2.4 XGBoost Model

XGBoost is an ML model used for stock market time series forecasting that uses a set of decision trees (Dezhkam & Manzuri, 2023). A gradient descent algorithm guides the process of preparing subsequent trees to minimize the loss function of the last tree Yang and Shami, 2020.

$$\mathcal{L}_{\mathrm{T}}(\mathrm{F}(x_i)) = \sum_{i=1}^{\mathrm{N}} \chi(y_i, \mathrm{F}_{\mathrm{T}}(x_i)) + \sum_{t=1}^{\mathrm{T}} \Pi(f_t), \tag{5.3}$$

where $\chi(\cdot)$ is a specified loss function that quantifies the deviations of the predicted and actual target values, and $\mathrm{F}_{\mathrm{T}}(x_i) = \sum_{t=1}^{\mathrm{T}} f_t(x_i)$ denotes the forecast on the i-th sample at the T-th boost and $\Pi(f) = \alpha \mathrm{K} + 0.5 \times \kappa \|\varpi\|^2$, where K is the number of leaves. For

the regularization term, α is the parameter of complexity. $||\omega||^2$ is the L2 norm of weight regularization, κ is a constant coefficient, and $\Pi(\cdot)$ represents the term of regularization, which penalizes the model complexity (Xia et al., 2017).

5.2.5 MLP Model

The MLP is a frequently used ANN consisting of three layers of neurons: an input layer, one or more hidden layers, and an output layer. The inputs (x_i) are multiplied by their weights (w_i), and the resulting products are combined. This sum and a bias term (b) are fed into an activation function to produce the neuron's output (Y_t) (Koukaras et al., 2022). Eq. (5.2) can be used to express this process in mathematical words:

$$\zeta_t = \sum_{i=1}^{N} w_i x_{it} + b \Longrightarrow Y_t = \sigma(\zeta_t), \tag{5.4}$$

$$\sigma(\zeta_t) = \frac{1}{1+e^{-\zeta_t}}, \tag{5.5}$$

where $\sigma(\cdot)$, the activation function, is frequently employed as a function, either continuous or discontinuous, that maps real numbers to a specific interval. Alternatively, the sigmoidal activation function can also be utilized (Al-Nefaie & Aldhyani, 2022).

5.2.6 LSTM Model

LSTM models are RNNs that excel at learning and retaining long-term dependencies, making them successful in various applications such as financial time series forecasting. The principle of an LSTM cell consists of the following four equations (Oukhouya & El Himdi, 2023a; J. M.-T. Wu et al., 2023):

♣ **Forget gate:**

$$f_t = \delta(W_{ef} \cdot (X_t, h_{t-1}) + W_{cf} C_{t-1} + b_f), \tag{5.6}$$

♣ **Input gate:**

$$i_t = \delta(W_{ki} \cdot (X_t, h_{t-1}, C_{t-1}) + b_i), \quad k = x, h, c$$

♣ **Memory cell:**

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_{lc}(X_t, h_{t-1}) + b_c), \quad l = x, h$$

♣ **Output gate:**

$$o_t = \delta(W_{ji} \cdot (X_t, h_{t-1}, C_{t-1}) + b_o), \quad j = x, h, c$$

where W_{e*}, W_{c*}, W_{k*}, W_{l*}, W_{j*}, and b_f, b_i, b_c, b_o are the weight and bias of each layer respectively, and $h_t = o_t \cdot \tanh(C_t)$ represents the hidden layer output. The forget gate (f_t) determines if the information should be kept or discarded, while the input gate (i_t) integrates new data into the cell (C_t). Eventually, the output gate (o_t) governs the selection of relevant data to transmit to the succeeding cell (C_{t-1}).

5.2.7 Grid Search

In ML, GS is commonly used to fine-tune parameters such as regularization strength, learning rate, several hidden layers ..., etc. By using the GS algorithm, we can identify the optimal set of hyperparameters for the models, resulting in better predictions and improved performance. In our case, we first defined the hyperparameters and their search space, as shown in the Table 5.3. The optimal hyperparameters for each model are shown below in bold blue. The design of the research study is illustrated in Figure 5.1.

Table 5.3: The GS hyper-parameter sets for the SVR, XGBoost, MLP, and LSTM models.

ML Models	Parameters	Type	Search Space
SVR	Epsilon in the SVR loss function (`epsilon`)	Continuous	[0.0001, **0.001**, 0.01]
	Regularization parameter (`C`)	Discrete	[100, **1000**, 1100]
	The kernel type (`kernel`)	Categorical	['`linear`', '`poly`', '`rbf`', '`sigmoid`']
	Kernel coefficient (`gamma`)	Continuous	[**1e-5**, 1e-4, 1e-3]
	Tolerance for stopping criterion (`tol`)	Continuous	default = 1e-3
XGBoost	# of regression trees (`n_estimators`)	Discrete	[100, **200**, 1000]
	Maximum regression tree depth (`max_depth`)	Discrete	[5,10, **15**, 20]
	Boosting the rate of learning (`learning_rate`)	Continuous	[0.01, **0.06**, 0.09]
	Minimum reduction of loss (`gamma`)	Continuous	[0.001, **0.01**, 0.1]
	Regularization term L1 on weights (`reg_alpha`)	Continuous	[0.001, **0.01**, 0.1]
	Regularization term L2 on weights (`reg_lambda`)	Continuous	[0.001, **0.01**, 0.1]
	Objective learning function (`objective`)	Categorical	['`reg:squarederror`', '`reg:linear`']
MLP	# of neurons in the ith hidden layer (`hidden_layer_sizes`)	Discrete	[(50,50,50), (50,100,50), **(100,)**]
	Activation function for hidden layer (`activation`)	Categorical	['`relu`', '`tanh`', '`logistic`', '`identity`']
	The solver for weight optimization (`solver`)	Categorical	['`sgd`','`adam`','`lbfgs`']
	Strength of the regularization term L2 (`alpha`)	Continuous	[0.001, **0.01**, 0.1]

Continued on next page

Table 5.3: The GS hyper-parameter sets for the SVR, XGBoost, MLP, and LSTM models. (Continued)

ML Models	Parameters	Type	Search Space
	Learning rate for weight update program (`learning_rate`)	Categorical	['constant', 'invscaling', 'adaptive']
LSTM	# of epochs to train the model (`epochs`)	Discrete	[20, **90**, 100]
	# of hidden layer	Discrete	[1, **2**, 3, 4]
	The function that tries to optimize (optimizer)	Categorical	['adam', 'rmsprop', 'sgd']
	Learning rate (learning_rate)	Continuous	[0.1, **0.01**, 0.001]
	Activation functions (`activation`)	Categorical	['tanh', 'sigmoid', 'relu']
	# of hidden layer neurons (`neurons_1`, `neurons_2`)	Discrete	[150, **250**, 350], [200, **200**, 300],
	Loss functions to be minimized during model training (loss)	Categorical	['mae', 'mse', 'mape']

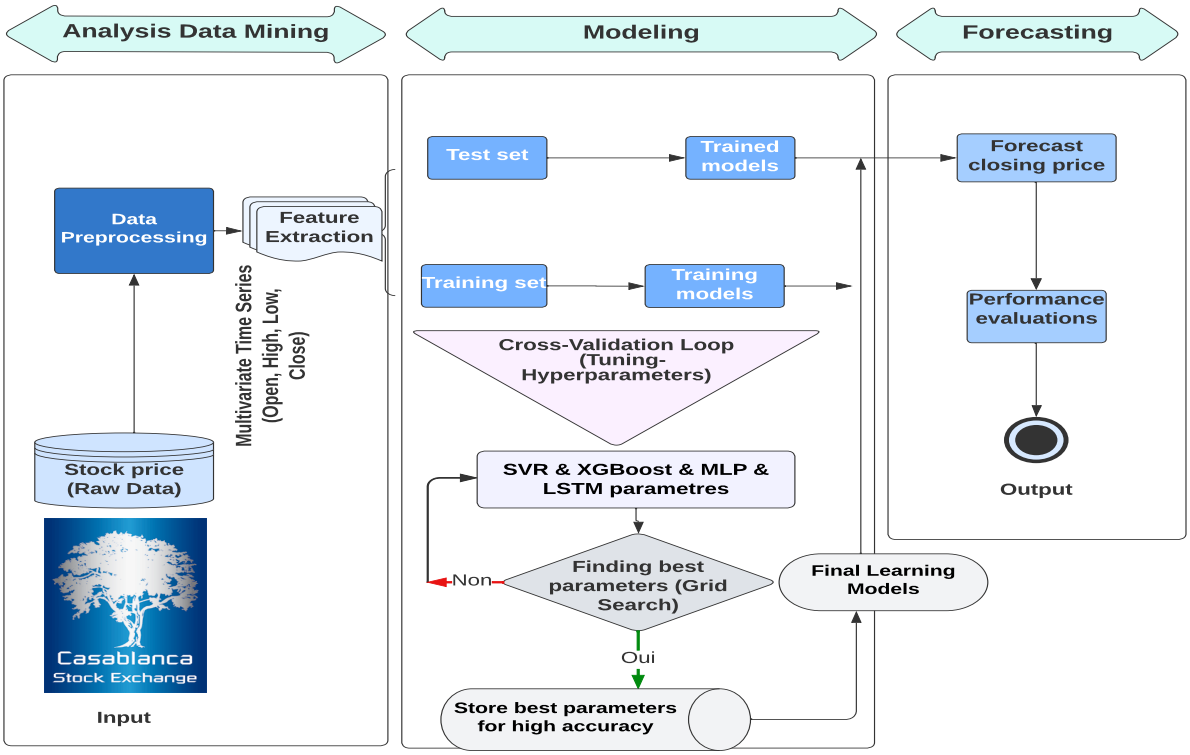


Figure 5.1 – Proposed research design architecture for analyzing, modeling, and predicting MSI 20.

5.3 Results and Discussion

In the section on results and discussion, we report and compare the performance of the SVR, XGBoost, MLP, and LSTM models for forecasting the MSI 20 stock market. Various evaluation measures such as ME, MPE, MSE, MAE, RMSE, MAPE, and R^2 scores are obtained to assess the models' accuracy. The general formula for evaluation measures is presented in subsection 2.8.1. Table 5.4 shows the evaluation values of various forecasting models for the MSI 20 price. XGBoost had the highest error among the four models. LSTM had the second-worst performance. SVR and MLP models had lower performance metrics than XGBoost and LSTM models. However, the SVR model had the best evaluation values for forecasting MSI 20 prices, achieving optimal ME, MAE, RMSE, MPE, MAPE, and MSE values of 0.674, 3.092, 3.993, 0.082%, 0.368%, and 15.941, respectively. The MLP model also showed excellent performance metrics. However, the predicted results of the XGBoost model in Figure 5.2 on the right showed a specific error. They predicted values of MSI 20, especially during the fall period (6 January 2023 – 10 January 2023), indicating that the model needed more data to improve its performance. Based on the results presented in Table 5.4 and Figure 5.2, it can be concluded that the SVR model followed by MLP models had the best performance for forecasting MSI 20 compared to other models.

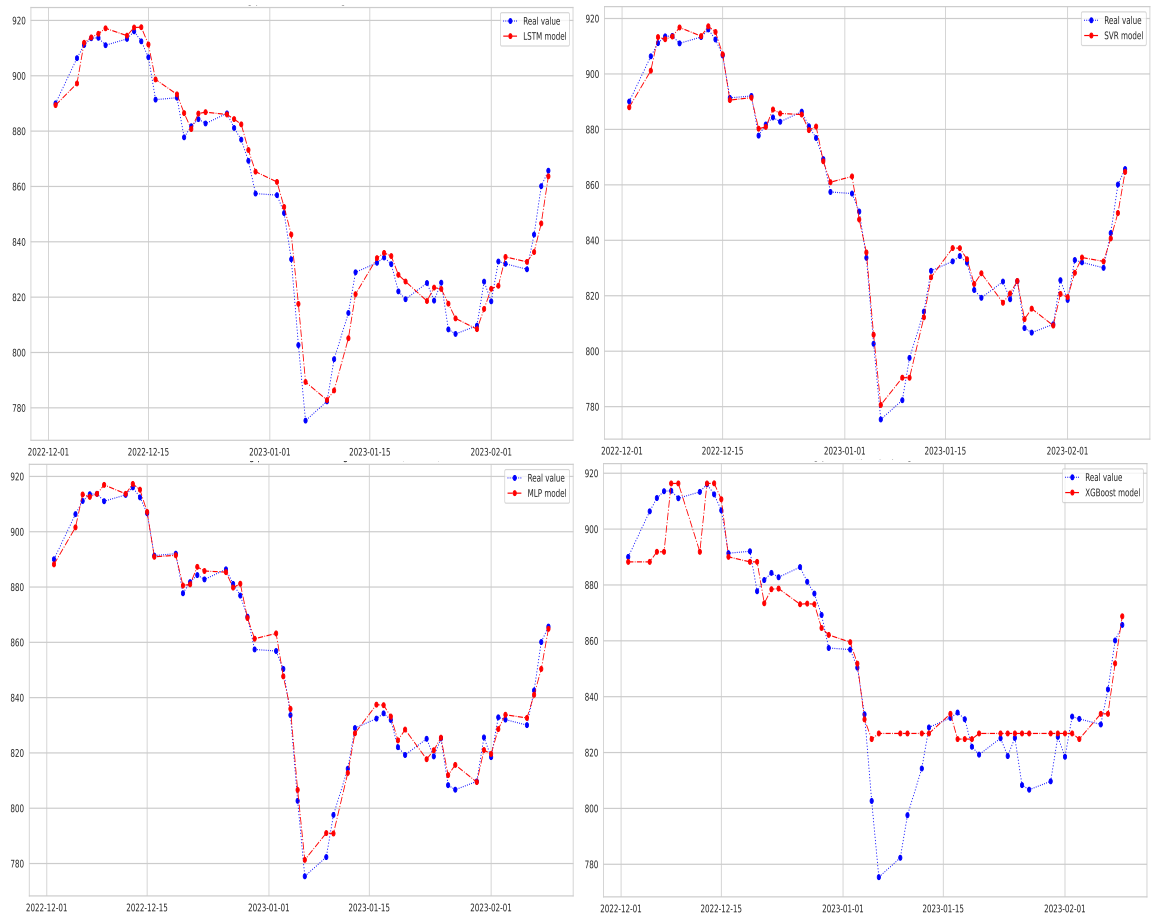


Figure 5.2 – MSI 20 stock price prediction: results of LSTM, SVR, MLP, and XGBoost models.

Table 5.4 – Performance analysis measures metrics of each model.

Models	**SVR**		**XGBoost**		**MLP**		**LSTM**	
Error Measure	Train	Test	Train	Test	Train	Test	Train	Test
ME	0.0899	0.674	2.277	0.558	5.406e-6	0.883	2.339	1.4
MAE	2.058	3.092	2.460	9.165	2.059	3.101	3.554	5.065
RMSE	2.646	3.993	3.141	13.515	2.642	4.018	4.496	6.322
MPE (%)	9.134×10^{-3}	0.082	0.226	0.116	7.603e-4	0.107	0.164	0.234
MAPE (%)	0.207	0.368	0.244	1.095	0.207	0.370	0.357	0.603
MSE	7.003	15.941	9.864	182.651	6.978	16.286	20.215	39.97
R^2	0.998	0.989	0.997	0.882	0.998	0.989	0.995	0.974

5.4 Conclusions and Future Work

This study applied four ML models-SVR, XGBoost, MLP, and LSTM-to model and forecast MSI 20 prices using multivariate time series data. The results showed that SVR outperformed the other models with lower errors and higher accuracy 98.9%. Future research could focus on improving the performance of XGBoost and exploring the potential of other models, such as CNN-LSTM, for stock market forecasting.

Chapter 6

Forecasting International Stock Market Trends: XGBoost, LSTM, LSTM-XGBoost, and Backtesting XGBoost Models

" The goal of forecasting is not to predict the future but to tell you what you need to know to take meaningful action in the present. "

–Paul Saffo

Contents

6.1 Introduction

In a laissez-faire world economy, the stock market is a central platform, enabling the exchange of funds between individuals and companies. Renowned for its accessibility and the potential for significant profits, it has long been a magnet for investors searching for lucrative returns and businesses needing capital (Yun et al., 2021). Forecasting is crucial in this dynamic landscape, utilising historical data to anticipate forthcoming events. This practice spans diverse disciplines such as environmental science, economics, business, and finance, demonstrating its broad-ranging impact and relevance. The stock market occupies a central position within financial markets, captivating the interest of researchers from both financial and technical backgrounds. The prediction of stock market price trends has emerged as a prominent research area due to its relevance to investors (Guresen et al., 2011). A significant portion of forecasting challenges revolves around examining time. A time series is a sequential arrangement of observations related to a variable. In this context, the variable under consideration is the stock price. Using sentiment analysis for predicting stock price trends within the context of financial time series helps improve the forecasting of closing prices (Thormann et al., 2021). In exploring the accuracy of classical, ML, and DL methods in time series forecasting, recent researchers delve into these challenges (Jouilil et al., 2023).

The stock price forecasting methods can be divided into two categories: Linear Models such as AR, MA, ARMA, ARIMA (Ariyo et al., 2014), and Nonlinear Models such as ARCH, GARCH (Bezerra & Albuquerque, 2017), ML algorithms (Shah et al., 2019). Linear models cannot capture the dynamics present in the data, while nonlinear models offer alternative approaches to address this limitation. To address the challenges of accurately predicting stock price direction, we have developed a hybrid model that combines the strengths of both ML and DL methods. The model's framework involves sequential workflow steps, including preprocessing, feature engineering, optimization, prediction, validation, and evaluation. We aim to overcome the complexities involved in stock price prediction and enhance the accuracy of our model. We have thoroughly reviewed the existing research on stock price prediction using ML models, conducting a comprehensive analysis of relevant studies in the field. Ding and Qin, 2020 introduced an Associated and Deep Recurrent Neural Network (DRNN) model, leveraging the LSTM network model with multiple inputs and outputs. The experimental results demonstrate the superior accuracy of the associated model compared to the other two models in simultaneously predicting multiple values, achieving a prediction accuracy of over 95%. Shen and Shafiq, 2020 studied Chinese stock market data, employing comprehensive preprocessing and feature engineering techniques. They developed a customized LSTM-based system that outperformed other approaches in accurately predicting stock market price trends. The study emphasized the importance of effective feature engineering for achieving high accuracy in stock market trend prediction. Vuong et al., 2022, XGBoost was used for feature selection in high-dimensional time-series data, followed by LSTM for stock price forecasting. Results showed that this approach outperformed the baseline ARIMA model regarding MSE, RMSE, and MAE on a Forex dataset from 2008-2018. In a comparative conducted by Oukhouya and El Himdi, 2023b, various ML methods, including XGBoost, SVR, MLP, and LSTM, were evaluated for forecasting a daily closing price of the MSI 20. The results demonstrated that the SVR and MLP models, optimized with GS, have outperformed other models and achieved outstanding accuracy in price prediction.

Some investigators have explored the use of hybrid models to surpass the limitations of single models, with promising results. Various forecasting techniques were explored

in a study conducted by the authors Ettayyebi and El Himdi, 2018a. The initial approach employed linear modeling techniques, including classical ARIMA-GARCH and exponential smoothing models. The subsequent approach employed non-linear modeling techniques, specifically ANNs, focusing on the MLP model. The results suggest that the MLP model with exogenous variables outperforms the alternative models, highlighting its effectiveness in renewable energy and time series forecasting. The study presented by Semmelmann et al., 2022 proposed a hybrid bi-directional LSTM-XGBoost model for accurate energy community load forecasting. By leveraging separate forecasts for general load patterns and peak loads and comprehensively integrating them, the hybrid model demonstrated superior performance compared to conventional methods relying on standard load profiles and LSTM-based forecasts. A separate study of Gutmann et al., 2021 introduces a hybrid ML model that combines XGBoost and LSTM to predict advanced truck parking occupancy at 30, 60, 90, and 120-minute intervals. The model demonstrates high accuracy in its predictions, utilizing historical occupancy data and ensuring compatibility with existing truck occupancy detection systems for future forecasting. This research presents the LSTM-XGBoost, a novel hybrid model that combines LSTM residuals within the XGBoost framework. The proposed model is compared to individual XGBoost and LSTM models to improve the accuracy of predicting fluctuations in six international stock prices. Additionally, by utilizing the skforecast library (Amat Rodrigo & Escobar Ortiz, 2023), we enhance the interval prediction of Bootstrap based on XGBoost through backtesting. Our findings demonstrate that the proposed model effectively tracks both stocks' upward and downward movements with its optimized GS approach and enhanced capture ability. This leads to improved prediction performance compared to using a single XGBoost or LSTM model for forecasting multivariate stock prices.

The rest of this chapter is organized as follows. Section 6.2 presents the experimental data set of six international stock markets and the data preprocessing techniques employed. The design methodology, including the algorithms utilized and the implementation details of the models, is also described in this section. Section 6.3 of this work thoroughly analyzes and evaluates the proposed model, including a comparison with alternative models. The findings and their implications are discussed in detail. Finally, Section 6.4 concludes the chapter by summarizing the main findings and suggesting potential avenues for future research.

6.2 Methods

This section describes the methodology for modeling and forecasting six international stock market trends. It includes time series pre-processing for multivariate data, the implementation of XGBoost, LSTM, and hybrid LSTM-XGBoost models, and the architectural design of the research process.

6.2.1 Handling Multivariate Time Series Data

The historical data used in the experiments consists of multiple time series of daily prices (Open, Low, High, Close prices, and Volume) for six international stock markets across different continents, such as DAX, MASI, HKEX, CAC 40, NASDAQ, and FTSE 250. The series covers the period from March 1st, 2018, to March 1st, 2023, with 1,259 trading days for each stock market series. Several reasons justify the choice of 5 years and including six international stock markets: (1) investors commonly analyze stock market trends using recent data for more relevant insights, (2) assessing the models' performance

during and before crisis periods, such as the COVID-19 pandemic, and (3) evaluating our model's performance across different continents and contexts. The data was collected from investing.com. Let $\mathrm{X}^{\mathrm{stock}}(t) = \{x_t^{\mathrm{stock}_{i,j}}; t \in \mathrm{T}\}$ represent a multivariate time series for each stock, where i represents the DAX, MASI, HKEX, CAC 40, NASDAQ, FTSE 250 respectively, and j represents the Open, Low, and High prices, Log(volume) respectively. We used the closing price of each share as a forecasting target with a single observation sequence,

$$\mathrm{Y}^{\mathrm{stock}_i} = \hat{y}_{t+1}^{\mathrm{stock}_i}. \tag{6.1}$$

The preprocessing step is crucial in time series analysis, where all multiple input series is transformed into a homogeneous numerical array to be compatible with the XGBoost and LSTM models. Data normalization is applied using the min-max formula (Eq. (6.2)) to ensure that features with different scales are processed together.

$$\tilde{x}_t = \frac{x_t^{\mathrm{stock}_{i,j}} - \min(x_t^{\mathrm{stock}_{i,j}})}{\max(x_t^{\mathrm{stock}_{i,j}}) - \min(x_t^{\mathrm{stock}_{i,j}})}. \tag{6.2}$$

This normalization technique facilitates faster convergence of the gradient descent algorithm and maintains the integrity of relationships within the data without introducing bias (Chung & Shin, 2020; Devan & Khare, 2020).

6.2.2 XGBoost Model

The XGBoost model, developed by T. Chen and Guestrin, 2016; S. Li and Zhang, 2020, is a robust ML algorithm known for its practicality and speed of execution. It employs gradient-boosted decision trees for efficient feature selection and has shown promising results in stock market forecasting due to its parallelization and scalability capabilities. The equation to predict the target series for sample i can be expressed as:

$$\hat{y}_{t+j} = \sum_{i=1}^{\mathrm{I}} f_i(x_j), \quad f_i \in \mathbf{F}, \ (j = 1, 2, \ldots, \mathrm{N}), \tag{6.3}$$

where **F** represents the space of regression trees. The idea of XGBoost is to minimise the generalised objective function (or loss function) can be written as:

$$\mathrm{L}_k = \sum_{j=1}^{\mathrm{N}} (y_{t+j} - \hat{y}_{t+j})^2 + \sum_{k=1}^{\mathrm{K}} \left(\gamma \mathrm{T} + \frac{1}{2} \alpha \sum_{m=1}^{\mathrm{T}} \omega_m^2 \right), \tag{6.4}$$

where T consists of the number of leaf nodes, N is the set of all samples in leaf m. The score of leaf m is measured by ω_m. α and γ are parameters of the tree.

6.2.3 LSTM Model

LSTMs, a subtype of RNNs, effectively address the challenge of exploding and vanishing gradients (Pascanu et al., 2013; Schmidhuber, Hochreiter, et al., 1997) by incorporating a memory cell and gate mechanism Y. Wang and Ni, 2019. The LSTM model consists of blocks with input (i_t), forget (f_t), and output gates (o_t). These blocks form a recurrent structure where the output of one block is fed into the next (Gutmann et al., 2021). Using LSTM models in stock market forecasting enables the effective capture of long-term

dependencies, resulting in precise predictions. The LSTM cell operates based on four equations, which govern its underlying principle (Oukhouya & El Himdi, 2023b):

$$\begin{aligned} f_t &= \delta(\mathcal{W}_{ef} \cdot (\tilde{x}_t, h_{t-1}) + \mathcal{W}_{cf} \mathrm{C}_{t-1} + b_f), \\ i_t &= \delta(\mathcal{W}_{ki} \cdot (\tilde{x}_t, h_{t-1}, \mathrm{C}_{t-1}) + b_i), && k = x, h, c \\ \mathrm{C}_t &= f_t \cdot \mathrm{C}_{t-1} + i_t \cdot \tanh(\mathcal{W}_{lc}(\tilde{x}_t, h_{t-1}) + b_c), && l = x, h \\ o_t &= \delta(\mathcal{W}_{ji} \cdot (\tilde{x}_t, h_{t-1}, \mathrm{C}_{t-1}) + b_o), && j = x, h, c \end{aligned}$$

The notation used in the equation includes the weights ($\mathcal{W}_{e*}$, $\mathcal{W}_{k*}$, $\mathcal{W}_{c*}$, $\mathcal{W}_{j*}$, $\mathcal{W}_{l*}$) and biases (b_f, b_c, b_i, b_o) for each layer. The hidden layer output (h_t) is calculated as the element-wise multiplication of the output gate (o_t) and the hyperbolic tangent of the cell state (C_t).

6.2.4 Hybrid LSTM-XGBoost Model

After forecasting the closing prices using the LSTM model, we calculated the residuals by taking the difference between the actual and predicted values at each time step for each stock, as expressed in Eq. (6.5). To develop our hybrid LSTM-XGBoost model, we utilized the GS algorithm for both training and evaluation processes. The GS algorithm allowed us to systematically explore different combinations of hyperparameters to find the optimal configuration for our hybrid model. This approach enhances the accuracy and performance of our model in predicting stock market trends.

$$\text{Residual}_{t+1}^{\text{stock}_i} = y_{t+1} - \hat{y}_{t+1}. \tag{6.5}$$

The optimal parameters of the hybrid LSTM-XGBoost model are determined using the GS optimization algorithm. The search space for these parameters is detailed in subsection 6.3.1 of the chapter.

6.2.5 Proposed method

In this study, the forecasting of the six stocks is divided into three main sections: 1) Data Collection and Preprocessing, 2) Model Learning and Testing, and 3) Forecasting and Backtesting with Bootstrap. The flowchart in Figure 6.1 provides an overview of the proposed approach, highlighting the critical functions of each section. These sections collectively form the framework for our research, enabling us to effectively collect, preprocess, train, test, and forecast stock market data. Our research used backtesting with prediction interval bootstrap to forecast multiple series for each stock, including Open, High, and Low prices and log_volume. Using a recursive multi-step approach, we relied on the previous predictions to forecast the Close price series. The skforecast library's Forecaster object facilitated model training and future prediction generation. By conducting backtesting, we assessed the accuracy and reliability of the predictive XGBoost model by applying it retrospectively to historical data, ensuring its effectiveness in forecasting future time series values.

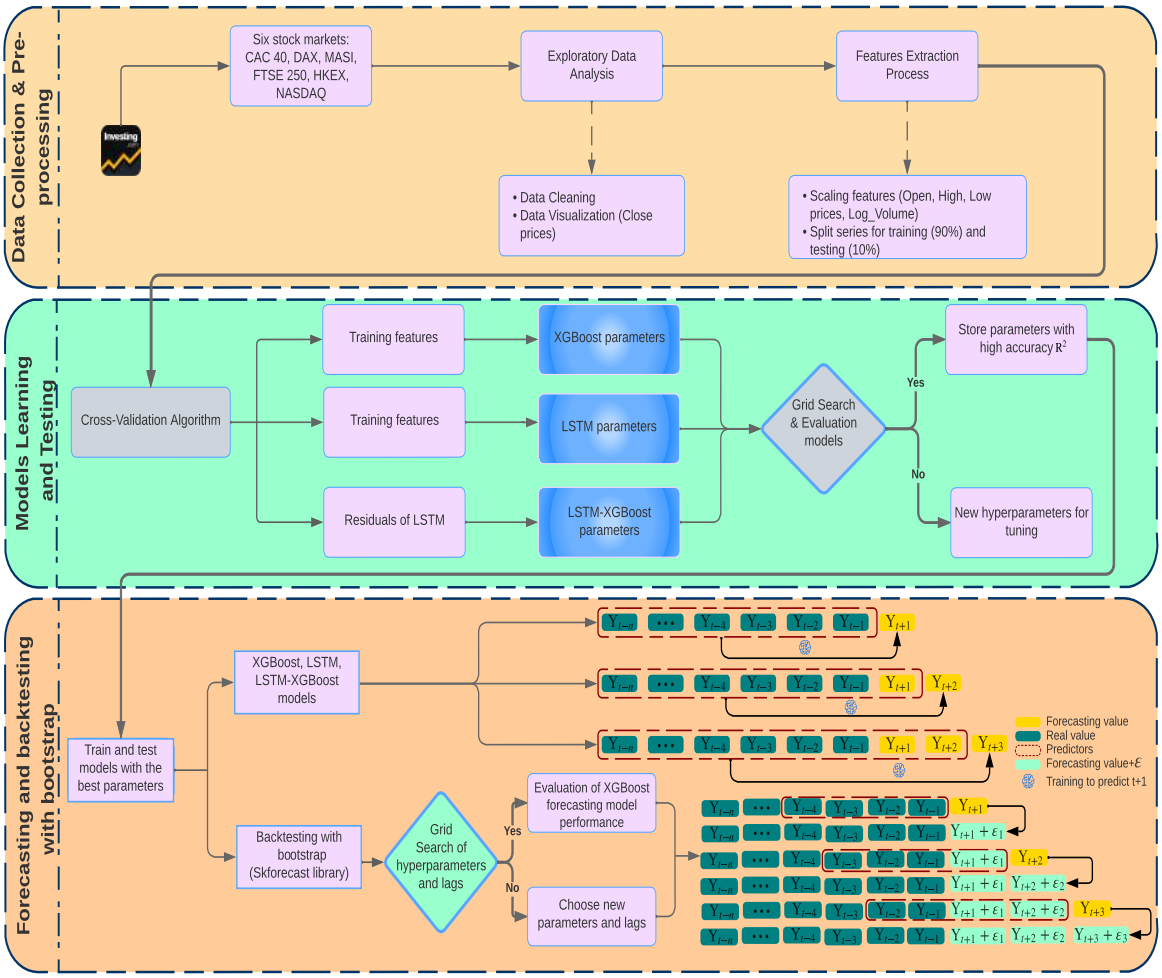


Figure 6.1 – Workflow of this study

6.3 Result and Discussion

This section discusses the results of forecasting six international stock markets using three different models. We conduct a thorough comparative analysis to evaluate their performance and assess the XGBoost model's backtesting capabilities. Additionally, we offer detailed insights into the performance measures achieved by each model. All experiments were conducted on a macOS system featuring a 2.4 GHz Intel Core i5 processor and 16 GB RAM. Furthermore, model evaluation and training took place on a Google Colab Pro+ virtual machine utilising eight TPUs and High-RAM. The LSTM model was implemented using Keras with TensorFlow (Géron, 2022) and XGBoost (v. 1.7.5) with the Scikit-Learn library as the underlying framework (Abadi et al., 2016) within the Anaconda3 environment (« Anaconda Software Distribution », 2020) using Python.

6.3.1 Models Grid Search Results

The GS procedure utilized a 5-fold blocked time series split cross-validation technique on the training set to discover the optimal hyperparameters for each model. The performance of the models was assessed using MSE as the evaluation metric for multi-step output. The GS results provided the optimal hyperparameters for the XGBoost, LSTM, LSTM-XGBoost, and Backtesting-XGBoost models, summarized in Table 6.1. Subsequently, the

Table 6.1 – Grid Search Hyperparameter Results for the XGBoost, LSTM, LSTM-XGBoost, and Backtesting-XGBoost Models

Models	Parameters	CAC 40	DAX	MASI	FTSE 250	HKEX	NASDAQ
XGBoost	# of regression trees	1000	100	300	500	500	800
	Maximum regression tree depth	60	30	20	30	10	10
	Boosting the rate of learning	0.03	0.1	0.02	0.02	0.03	0.05
	Minimum reduction of loss	0.00001	0.001	0.01	0.1	0.1	0.001
	Regularization term L1 on weights	0.5	0.3	0.1	0.1	0.09	0.3
	Objective learning function	Squarederror	Squarederror	Squarederror	Squarederror	Squarederror	Squarederror
LSTM	Batch size	32	16	4	8	8	16
	Neurons in the 1st hidden layer	250	900	300	400	150	450
	Neurons in the 2nd hidden layer	400	900	300	500	300	200
	Number of epochs	100	100	100	100	100	100
	Patience	3	7	4	5	4	4
LSTM-XGBoost	# of regression trees	1000	300	300	1000	600	600
	Maximum regression tree depth	2	4	2	5	2	3
	Boosting the rate of learning	0.1	0.03	0.01	0.008	0.02	0.04
	Minimum reduction of loss	0	0	0	0	0.001	0
	Objective learning function	Squarederror	Squarederror	Squarederror	Squarederror	Squarederror	Squarederror
Backtesting	# of regression trees	900	300	700	1000	500	300
	Maximum regression tree depth	1	2	3	3	1	2
	Boosting the rate of learning	0.03	0.04	0.09	0.09	0.03	0.03

best parameters were applied to train the respective models, and the close pricing of the six stocks was predicted for both the testing and training series.

6.3.2 Comparative Analysis of Models Performance

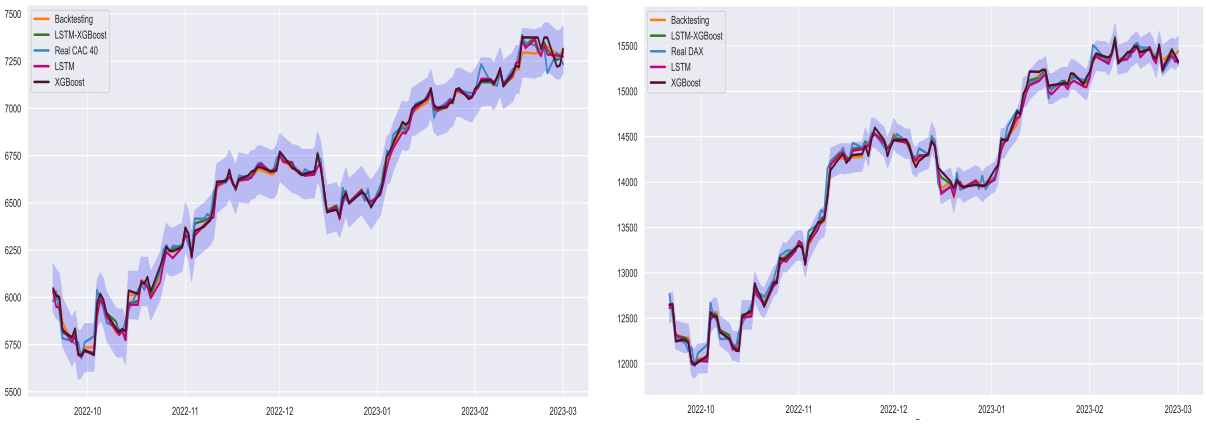
The predictive performance of the XGBoost, LSTM, and proposed LSTM-XGBoost hybrid models in forecasting the directional movements of six stock prices (DAX, MASI, HKEX, CAC 40, NASDAQ, and FTSE 250) was evaluated using several performance metrics (see subsection 2.8.1), including MAE, MAPE, RMSE, Theil's U Statistics, and R^2. Each model estimates one of the interval limits to construct a prediction interval using the Backtesting XGBoost model. In our study, the models obtained for Q = 0.1 and Q = 0.9 create an 80% prediction interval (90% − 10% = 80%). The findings presented in Table 6.2 equivocally establish the superior performance of the hybrid LSTM-XGBoost model across all error measures compared to the benchmark model. These results affirm that the LSTM-XGBoost model exhibits exceptional accuracy when applied to the six stock market indices under scrutiny. Notably, this model delivers precise predictions while effectively managing the trade-off between the training and testing datasets. Additionally, the XGBoost model emerges as a viable and commendable alternative for investors.

Theil's U Statistics values close to zero for all models further confirm the excellent forecasting ability of the hybrid model in predicting the trends of the six stock markets compared to the benchmark. Figures 6.2, 6.3 provides a visual representation of the performance of the hybrid model and backtesting, highlighting the accuracy and quality of the predictions. Our results reveal that model effectiveness varies by index, with XGBoost excelling with the DAX index and LSTM performing optimally with NASDAQ. Overall, the LSTM-XGBoost hybrid model consistently demonstrates high accuracy, showing promise in stock price prediction across diverse indices.

Table 6.2 – Performance comparison of forecasting six stock markets and benchmark models in out-of-sample periods

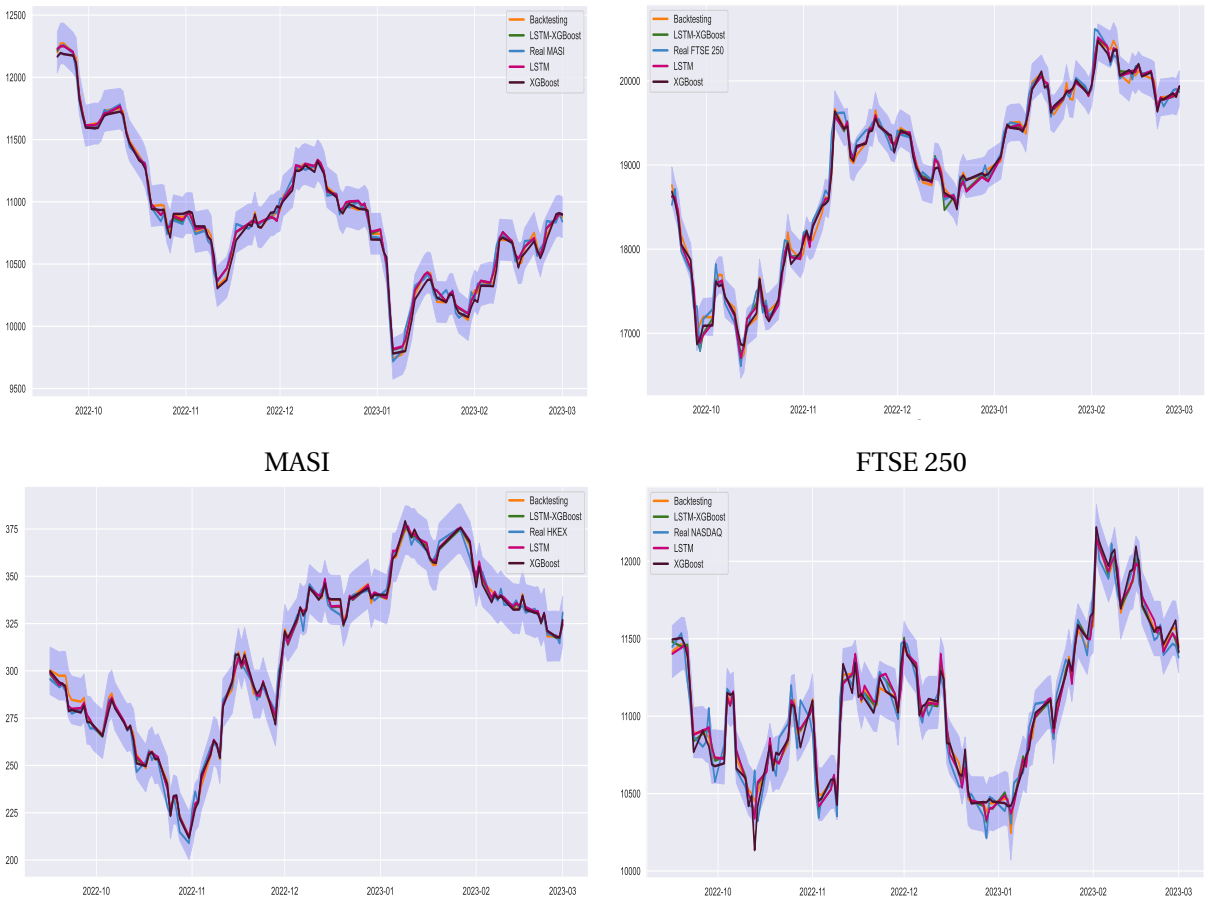
Models	MAE		MAPE (%)		RMSE		Theil's U Statistics*		R^2	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
XGBoost										
DAX	1.724	62.153	0.013	0.449	3.043	81.732	0.002	0.041	0.999	0.994
MASI	30.76	49.853	0.261	0.464	37.947	64.703	0.027	0.055	0.998	0.984
HKEX	0.381	2.871	0.124	0.970	0.504	3.730	0.420	3.857	0.999	0.992
CAC 40	0.069	31.341	0.001	0.479	0.103	42.393	0.0003	0.096	0.999	0.991
NASDAQ	1.080	75.528	0.011	0.686	1.611	102.150	0.001	0.083	0.999	0.951
FTSE 250	3.578	82.499	0.018	0.443	6.036	118.972	0.001	0.033	0.999	0.986
LSTM										
DAX	54.176	59.731	0.420	0.433	73.096	80.322	0.042	0.040	0.997	0.994
MASI	26.002	32.709	0.225	0.306	34.602	40.162	0.025	0.034	0.998	0.994
HKEX	2.1180	2.787	0.616	0.939	2.898	3.503	2.407	3.617	0.999	0.993
CAC 40	24.561	30.386	0.439	0.469	34.528	40.852	0.104	0.093	0.997	0.992
NASDAQ	60.264	83.205	0.570	0.754	85.667	101.700	0.072	0.083	0.999	0.952
FTSE 250	67.561	74.758	0.348	0.310	91.031	101.420	0.022	0.028	0.998	0.989
LSTM-XGBoost										
DAX	34.952	49.958	0.269	0.365	45.676	67.062	0.026	0.034	0.999	0.996
MASI	24.651	31.706	0.213	0.296	32.584	39.120	0.023	0.033	0.999	0.995
HKEX	1.7360	2.569	0.523	0.867	2.288	3.347	1.904	3.463	0.999	0.994
CAC 40	10.513	25.513	0.186	0.393	13.427	34.349	0.040	0.078	0.999	0.994
NASDAQ	32.437	77.660	0.316	0.705	43.465	95.260	0.037	0.078	0.999	0.957
FTSE 250	45.795	73.350	0.230	0.392	57.837	99.300	0.014	0.028	0.999	0.990

* $\times 10^{-5}$.



CAC 40 DAX

Figure 6.2 – Comparative of Stock Price Forecasting for CAC 40, and DAX: XGBoost, LSTM, LSTM-XGBoost, and Backtesting Models Results.



MASI FTSE 250

HKEX NASDAQ

Figure 6.3 – Comparative of Stock Price Forecasting for MASI, FTSE 250, NASDAQ, and HKEX: XGBoost, LSTM, LSTM-XGBoost, and Backtesting Models Results.

6.4 Conclusion

This study aimed to assess the performance of different algorithms in predicting stock markets across various continents. Three models, namely XGBoost, LSTM, and LSTM-XGBoost, were employed to evaluate the effectiveness of backtesting performance, specifically in the prediction interval. The experimental findings demonstrated that the proposed hybrid model significantly enhanced the accuracy of forecasting the trends in six international stock markets, surpassing the performance of the individual XGBoost and LSTM models. This chapter provides valuable insights into the comprehensive exploration of the impact of Backtesting with refit and fixed training size (rolling origin) as well as GS hyper-parameter optimization on the performance of the models. The hybrid model, while showing promise, manifests limitations associated with market volatility, hyperparameter sensitivity, time lag, and offset. These elements necessitate consideration in real-world applications, emphasizing the need for continuous research to enhance its robustness. In future research, we aim to enhance the stock market forecasting model by adding more variables and exploring the potential of other hybrid models, such as CNN-LSTM. We also plan to improve the performance of Backtesting by incorporating exogenous variables. These advancements will improve accuracy and reliability in predicting stock market trends.

Chapter 7

New Predictive Modeling in Financial Markets using Nonlinear Time Series and Deep Learning Models

" Fortune Favors the Prepared Mind "

–Louis Pasteur

Contents

7.1 Introduction

Time series forecasting integrates statistical methods and ML to anticipate future patterns using past data. Its applications span finance, marketing, and government sectors, playing a crucial role in stock price prediction and disaster management. This dynamic field extends its reach to domains like medicine and earth sciences. In this chapter, we explore the utilization of classical models and DL in time series forecasting to comprehend stock market indexes and their influence on residual models.

Several studies have examined the predictability of financial time series using various models. These models range from traditional linear frameworks like the ARIMA model to more complex nonlinear methods. While ARIMA and its variations AR, MA, and ARMA models (G. E. Box et al., 1970; Cochrane, 1997; K.W. Hipel, 1994; J. Lee, 2018) are effective at capturing linear relationships and SARIMA model, their inflexibility limits their applicability to situations where data display nonlinear behaviour. To address this limitation, researchers have explored nonlinear time series models such as the ARCH (Engle, 1982) and GARCH models (Bollerslev, 1986). These models are well-known for their ability to capture changing variances (heteroscedasticity) present in financial time series. Furthermore, various real-life problems, such as economics (C. W. J. Granger & Andersen, 1978) and demography (Subba Rao & Gabr, 1987), have seen the successful application of bilinear time series models. These models were originally introduced by C. W. J. Granger and Andersen, 1978 and further developed by Subba Rao, 1981 and Liu and Brockwell, 1988. In this study, we specifically focus on the application of first-order superdiagonal bilinear models, also known as $\mathrm{BL}(0,0,2,1)$. The practical utility of these models in multiple fields has been demonstrated through their applications in studies conducted by Maravall, 1983, Weiss, 1986, and Bouzaâchane, 2006. However, it should be noted that such models may not always be suitable for forecasting non-financial data, such as aviation demand (Xu et al., 2019), as they primarily focus on volatility in stock market returns. In contrast, AI models offer promising opportunities for forecasting across a wide range of domains. These models effortlessly capture nonlinear patterns in data without imposing distributional assumptions or requiring data transformations, making them a compelling alternative.

NNs excel in learning hierarchical representations, unlike traditional methods like ARIMA. With the rise of affordable storage and advanced computing, DL methods like DBNs and CNNs have achieved breakthroughs by exploiting local data dependencies (Lipton et al., 2015). ML techniques, including DL algorithms such as MLP, RNNs and LSTM, have brought about a revolution in prediction by capturing complex relationships between variables (Oukhouya et al., 2024; Sen & Dutta Choudhury, 2024). These methods, along with ML-based approaches such as SVMs, XGBoost and RF, are being applied in various fields, including finance (J. Cao et al., 2019; Siami-Namini et al., 2018). LSTM, in particular, is renowned for its effectiveness in forecasting time series data (K. Chen et al., 2015; Rhanoui et al., 2019), particularly in predicting S&P 500 volatility. Although stock market prediction remains challenging, ANNs hold promise, as demonstrated by early successes in forecasting MASI returns. Current research is focused on combining AI techniques and integrating new factors to enhance forecasting accuracy. However, there is limited empirical evidence comparing traditional methods like ARIMA with DL-based approaches like LSTM in economic and financial time series prediction (Siami-Namini et al., 2018).

The performance of stock market prediction models is extremely important. DL models can be enhanced by utilizing algorithms such as GS to improve their predictive capa-

bilities. Additionally, it is essential to evaluate these models using appropriate metrics in order to assess the accuracy and reliability of their forecasts. Performance measures are crucial in objectively quantifying the effectiveness of the models and facilitating comparisons. These measures serve the main purpose of providing quantitative indicators that demonstrate the accuracy and quality of the predictions generated by the models. Researchers, investors, and financial decision-makers can use these measures to develop well-informed investment strategies and effectively manage portfolios. By utilizing performance metrics such as MFE (G. Zhang et al., 1998), MAE (G. P. Zhang, 2007), MPE (Flaherty & Lombardo, 2000), MAPE, RMSE, NMSE (L. J. Cao & Tay, 2003), and Theil's U Statistics (U) (Park, 1999), stakeholders can determine the suitability of a model for specific stock price prediction requirements.

This research aims to find the best model for forecasting the Moroccan stock market index (MSI 20) by comparing different models. We will examine both traditional methods (ARIMA, bilinear) and newer approaches (LSTM, hybrid models) to determine which is the most accurate, particularly during periods of market volatility. The findings of this study will assist investors in making informed decisions and potentially minimize losses. This chapter is divided into several sections to comprehensively address our research objectives. Section 7.2 provides a comprehensive review of recent literature in the subject area, which contextualizes our research and identifies gaps that need to be addressed. In Section 7.3, we describe the MSI 20 stock index and explain the theoretical background of the methodologies used. Specifically, we discuss the development of the ARIMA, bilinear models, LSTM, and their proposed hybrids. This section helps readers understand the analytical framework. Section 7.4 presents the empirical findings from applying each model discussed in the previous section. We methodically analyze and discuss these results, contributing to a better understanding of their effectiveness in predicting residuals of each model for stock market behaviour. Finally, in Section 7.5, we summarize our conclusions drawn from the empirical findings and provide recommendations for future research. This section offers valuable insights for both academics and practitioners in the field of financial forecasting and investment analysis.

7.2 Literature Review

Recent studies have highlighted the importance of forecasting financial time series data because of its volatility and susceptibility to economic factors (Mustapa & Ismail, 2019). While traditional linear models like ARIMA and GARCH have been widely used, their limitations have prompted researchers to explore modern ML-DL single and hybrid techniques such as SVR, XGBoost, MLP, and LSTM. These models have demonstrated superior performance in capturing nonlinear patterns. Comparative studies in quantitative finance research have confirmed the effectiveness of these modern approaches in comparison to traditional methods. Additionally, nonlinear stochastic models like GARCH have emerged as alternatives to ARIMA, providing greater flexibility in capturing the complexities of financial data (Cocianu & Grigoryan, 2016; Z. Li & Tam, 2018; Rundo et al., 2019; Sezer et al., 2020).

Adebiyi et al., 2014 conducted a study comparing the performance of ARIMA and ANNs models using New York Stock Exchange (NYSE) data. The findings of their study demonstrated that the neural network model outperformed ARIMA in terms of forecasting accuracy. Similarly, Wijaya et al., 2010 conducted a study on stock price forecasting in Indonesia, comparing the ANN and ARIMA methods. The results of their study indicated that the ANN approach had smaller errors in comparison to ARIMA. Falloul

and Mansouri, 2014 aimed to model the performance of the MASI. They identified the GARCH process as the most suitable model for this purpose through statistical analysis. Many studies have been conducted to explore forecasting techniques in financial markets. Tang et al., 2003 extended the ARMA-GARCH model for predicting stock prices, and their study showed better results when compared to AR-GARCH models. Hossain and Nasser, 2008; Hossain and Nasser, 2011 compared SVM with ARMA-GARCH and GARCH models, and found that SVM's long memory property was beneficial for forecasting financial returns. Xu et al., 2019 applied SARIMA-SVR to aviation industry indicators, leading to improved accuracy. Ping-Feng Pai, 2005 proposed a hybrid ARIMA-SVM methodology for stock price forecasting, with a specific emphasis on optimizing parameters to achieve the best possible results. M. Kumar and M., 2014 identified ARIMA-SVM as the best model for predicting stock index returns, achieving high accuracy and returns. Empirical studies have also substantiated the effectiveness of XGBoost in predicting stock prices.

Weiss, 1986 introduced extensions to the ARMA model, such as bilinearity and ARCH errors, and investigated their combination. Tests were conducted for each extension, and parameter estimation methods were applied. The results showed that incorporating bilinearity and ARCH errors is essential for precise modelling. The study also discussed the implications of misspecification when only considering the ARMA model and applied the methods to real-world time series data. Maravall, 1983 utilized real-world data to demonstrate the advantages of addressing nonlinearities in ARIMA forecasts. The nonlinearity detection was based on squared residuals' ACF. The inclusion of parsimonious bilinear processes proved to be effective in modelling nonlinearity. Moreover, the detection of nonlinearity and improved forecasts were consistent across linear and bilinear specifications. Bilinear models were particularly successful in capturing unusual patterns or outlier sequences. Bibi and Ghezal, 2018 introduce and explore a new class of models called Markov-switching Bilinear-GARCH processes (MS-BLGARCH). These models extend Markov-switching GARCH models by incorporating interaction components between the observed series and its volatility process. The objective of these models is to capture the complex dynamics and behaviour in financial time-series data that exhibit structural changes, with a particular focus on asymmetric behaviour like the leverage effect. The main goals of this study are to establish the structural properties of the model, including stationary and causal solutions, and to derive expressions for the covariance function and higher-order structure to facilitate model identification. Additionally, the authors aim to develop parameter estimation techniques using the Generalized Method of Moments (GMM) and apply them to real-world financial data, such as the foreign exchange rate of the Algerian Dinar against the single European currency. Lmakri et al., 2020 derives pseudo-Gaussian and rank-based tests for testing white noise against panel super-diagonal bilinear dependence, while Lmakri et al., 2024 develops optimal rank-based testing procedures for randomness against panel general super-diagonal bilinear dependence.

Yuan et al., 2021 have introduced a predictive model that utilizes XGBoost and Grid SearchCV algorithms. Their findings demonstrate superior performance compared to other ML methods, with a significant decrease in RMSE of 1.39%, 2.43%, and 8.33% compared to SVM, NN, and LightGBM algorithms, respectively. Moreover, their analysis provides valuable insights into the key features that influence closing stock prices, thereby offering valuable insights for predictive modelling and decision-making processes in financial markets. Siami-Namini et al., 2018 analyzed historical financial time series data from January 1985 to August 2018. Their study found that LSTM, a DL-based algorithm, performed better than the traditional ARIMA model. LSTM demonstrated an average re-

duction in error rates ranging from 84% to 87% compared to ARIMA, indicating its superior predictive capability in financial forecasting. In a related study by Rhanoui et al., 2019, a comparative analysis was conducted to forecast financial budget time series. The findings of their study demonstrated that the LSTM model outperformed the ARIMA model, showcasing the effectiveness of DL approaches in financial forecasting tasks. Y. Wang et al., 2023 employed the XGBoost ML algorithm to forecast the closing price of the China Securities Index (CSI) 300 futures contract, confirming its suitability in predicting financial time series. Their research showcased substantial improvements in predictive efficiency by optimizing empirical modal decomposition. The results highlighted the accuracy of both LSTM and XGBoost models, with empirical modal decomposition particularly enhancing the predictive accuracy of XGBoost. K.-j. Kim, 2003 conducted a comparative study evaluating SVM, backpropagation NNs, and case-based reasoning for financial forecasting. The research highlighted SVM as a promising alternative for stock market prediction, showcasing its potential efficacy in financial markets. In their study, Kara et al., 2011 aimed to establish and compare two models for predicting the daily direction of movement of the Istanbul Stock Exchange (ISE) National 100 Index. The experiments conducted by the authors revealed that the ANN model (75.74%) outperformed the SVM model (71.52%) in terms of average performance. This indicates that the ANN model has superior predictive capability in this specific context.

K. Chen et al., 2015 conducted a study comparing the effectiveness of LSTM with a random prediction method in modelling and forecasting the China stock market. The research findings revealed that the LSTM model greatly improved the accuracy of stock return forecasts, increasing it from 14.3% to 27.2%. These results highlight the effectiveness of LSTM in predicting the volatile and unpredictable nature of the Chinese stock market. X. Wu and Lu, 2021 introduces a hybrid forecasting system for financial asset prices that integrates fuzzy techniques, statistical tools, and neural networks. This system consists of three stages: fitting a fuzzy autoregression model, developing a fuzzy bilinear regression model, and combining fuzzy bilinear regression forecasting with probabilistic neural network classifiers. This approach aims to enhance accuracy in predicting complex financial systems by addressing outliers and incorporating risk perspectives. Validation using historical data from the Shanghai Stock Exchange (SSE) composite index confirms the system's superiority over other methods. Theoharidis et al., 2023 introduced a new hybrid DL model called VAE-ConvLSTM. This model combines Variational Autoencoders and Convolutional LSTM Networks. The study's main focus was on forecasting inflation and it showed how DL approaches can effectively handle the nonlinearities and nonstationarity present in macroeconomic data. Table 7.1 summarizes the most critical and up-to-date financial time series forecasting in different stock market studies and highlights the different methodologies and approaches used in recent research.

7.2.1 Timeline of time series analysis and forecasts

Time series forecasting involves collecting past observations to construct a mathematical model that reflects the underlying data generation process (G. P. Zhang, 2007; G. Zhang, 2003). This is particularly useful when the statistical pattern of repeated observations is poorly understood or when an explanatory model is lacking. Forecasting plays a crucial role in decision-making processes, necessitating the development of accurate models (Adhikari & Agrawal, 2013). Over time, time series analysis and forecasting have seen significant advancements, with researchers and practitioners exploring various methodologies. Early approaches like AR models focused on capturing linear de-

Table 7.1 – Summary of Selected State-of-the-Art Studies

References	Method	Stock Market	Objective
Nabipour et al., 2020	Decision Tree, RF, Adaptive Boosting (Adaboost), XGBoost, SVM, Naïve Bayes, K-Nearest Neighbors (KNN), Logistic Regression, ANN, RNN, and LSTM	NYSE & Petroleum, diversified financials, basic metals and non-metallic minerals	Predicting stock market trends
Elmsili and Outtaj, 2021	Ridge regression, LASSO regression, SVM, KNN, RF, and Adaboost	CSE	Stock return prediction
Samarawickrama and Fernando, 2017	RNN, SRNN, GRU and LSTM	Sri Lankan stock market	Predicting daily stock prices
Hussein et al., 2015	ANN, MLP and KLD	Stock Market's Daily Data	Stock market prediction
Sheta et al., 2015	SVM, MLP and ANN	S&P 500 Dataset	Predicting stock market index
Yu et al., 2021	LSTM, XGBoost, RNN and LSTM-XGBoost	S&P 500 time series	Prediction of stock prices (Open, High, Low, Close, Volume, Adj-Close)
Fazeli and Houghten, 2019	LSTM and Technical indicators	Apple, Microsoft, Google and Intel stocks	Prediction of stock market trends
X. Wu and Lu, 2021	Neutral fuzzy bilinear regression	SSE	Develop a hybrid forecasting system for financial asset prices
Sornette and Pisarenko, 2008	Bilinear stochastic model	Stock Market	Financial time series modeling
Ariyo et al., 2014	ARIMA	NYSE & NSE	Stock price prediction
Y. Wang and Guo, 2020	ARIMA, XGBoost, GSXGB and DWT-ARI-MA-XGBoost	10 stock data sets	Forecasting of stock market volatility

pendencies, while hybrid models combined statistical techniques with bilinear models to improve accuracy. DL models such as LSTM introduced non-linearity and addressed complex patterns. The choice of method now depends on data characteristics, and ongoing research aims to refine our understanding further. A timeline of strategies adopted for financial time-series modelling is presented in Figure 7.1.

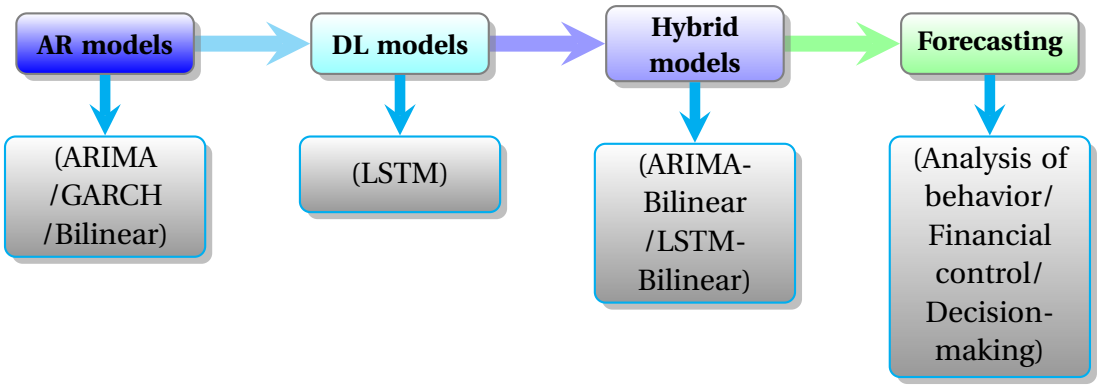


Figure 7.1 – Timeline of adopted approaches for time series analysis and forecasting.

7.3 Data and Methodology

The CSE was established in 1929 and is considered a cornerstone of the Moroccan financial landscape [1]. It is renowned for its efficiency within the MENA region. Two key indices, the MASI and the MADEX, form the foundation of the CSE's performance metrics (Oukhouya & El Himdi, 2023a). For our study, we focused on the MSI 20 index, which represents the broader CSE as it includes the 20 most prominent listed shares. As a free-float capitalisation-weighted index, the MSI 20 holds significant stature within the financial ecosystem of the Maghreb region. It represents the market's most critical stocks in terms of liquidity and trading volume (Oukhouya & El Himdi, 2023b). Our analysis examines the historical daily prices of the MSI 20 index, including important variables such as the open price, low price, high price, close price, and transaction volume. The open price indicates the index's value at the beginning of the trading day, while the low and high prices represent the minimum and maximum values of the MSI 20 observed during the trading session. The closing price reflects the index's value at the end of the trading day.

Figure 7.3 illustrates the profound impact of the post-COVID-19 era on the Moroccan stock market, particularly the significant downturn in the MSI 20 index. December 2022 marked a historic plunge, with the index experiencing its largest single-day loss of -8.82%. This unprecedented decline, unmatched since the index's inception in 2002, highlights the severe market volatility caused by the pandemic. The subsequent period was characterized by heightened investor anxiety and uncertainty, as evidenced by the market's erratic behaviour. However, amidst the turmoil, the market demonstrated resilience and embarked on a remarkable recovery trajectory in early 2023. Various factors contributed to this resurgence, including government-led financial support for different economic sectors and advancements in COVID-19 vaccine development. These positive developments restored investor confidence and optimism, leading to increased market activity and signalling a hopeful outlook for the future.

7.3.1 Time series forecasting using ARIMA model

Various generalized models are available for analyzing time series data, representing a wide range of stochastic processes. Among these models, the AR and MA models, extensively discussed by G. E. Box et al., 1970 and K.W. Hipel, 1994, respectively, are particularly notable. The combination of these models gives rise to the ARMA model, as proposed by G. Box and Jenkins, 1970, and the ARIMA model, introduced by Cochrane, 1997.

1. https://www.casablanca-bourse.com/en

Further advancements include the Autoregressive Fractionally Integrated Moving Average (ARFIMA) model (Bhardwaj & Swanson, 2006), which integrates components from both ARMA and ARIMA models, as outlined by Galbraith, Zinde-Walsh, et al., 2001 and Park, 1999. The SARIMA model, advocated by Hamzaçebi, 2008, is utilized to address seasonality. These models, often called Box-Jenkins models due to their foundation on the Box-Jenkins principle, aim to understand variations in time series data (M. Kumar & M., 2014; G. Zhang, 2003). ARIMA models try to minimize the discrepancy between observed and modeled values by adjusting the observed data. Known for their versatility in handling stationary and non-stationary series, ARIMA models have long been a standard tool for time series forecasting. However, it is worth noting that they have limitations, as pointed out by Siami-Namini et al., 2018. The mathematical expression for an ARIMA(p,d,q) model involves lag polynomials, as described by Flaherty and Lombardo, 2000 and K.W. Hipel, 1994:

$$\varphi_p(\mathrm{B})(1-\mathrm{B})^d \mathrm{Y}_t = \theta_q(\mathrm{B})\varepsilon_t, \tag{7.1}$$

where $\varphi_p(\mathrm{B}) = \left(1 - \sum_{i=1}^{p} \varphi_i \mathrm{B}^i\right)$ and $\theta_q(\mathrm{B}) = \left(1 + \sum_{j=1}^{q} \theta_j \mathrm{B}^j\right)$, Y_t and ε_t are respectively the actual value and random error (or random shock) at time t, hence, ε_t is assumed to be a white noise process, i.e. a sequence of uncorrelated variables with zero mean and constant variance σ^2.

Box-Jenkins Methodology

After exploring time series analysis, the next crucial step is to select an appropriate model. This model will generate accurate forecasts based on historical data patterns and determine the optimal model order. The Box-Jenkins methodology, introduced by G. E. Box et al., 1970, offers a practical approach for constructing ARIMA or SARIMA models that best fit a given time series while adhering to the principle of parsimony (see subsection 2.6.6). This methodology, highlighted by G. Zhang, 2003 and Flaherty and Lombardo, 2000, is essential in time series analysis and forecasting. It employs a three-step iterative process - model identification, parameter estimation, and diagnostic verification - to identify the most economical ARIMA model from various possibilities. As illustrated in Figure 2.5, this iterative process continues until an appropriate model is identified, enabling the forecasting of future values for the time series. The Box-Jenkins methodology comprises four iterative stages:

1. **Model identification**: Ensure stationarity of variables, detect seasonality, and determine the autoregressive or moving average components using ACF and PACF plots.
2. **Model estimation**: Utilize computing algorithms to find coefficients fitting the chosen ARIMA model, often employing MLE or non-linear least-squares estimation.
3. **Model checking**: Verify if the estimated model meets the criteria of a stationary univariate process, ensuring residuals' independence and constant mean and variance over time. Utilize ACF and PACF of residuals for misspecification identification. If necessary, iterate back to step one and repeat. Select the appropriate model based on AIC or BIC and validate with MSE, as defined in the proposition 1.
4. **Forecasting**: Once the ARIMA model meets stationary criteria, it is employed to forecast future values.

7.3.2 Time series forecasting using Bilinear model

Nonlinear time series have gained significant attention in recent years, with various models being proposed and successfully applied in areas such as economics, demography, and environmental studies. Among these models, the bilinear time series model, denoted as $\mathrm{BL}(p,q,\mathrm{P},\mathrm{Q})$, is particularly popular. This model is defined as follows:

$$\mathrm{Y}_t = \sum_{j=1}^{p} a_j \mathrm{Y}_{t-j} + \sum_{k=1}^{q} c_k \varepsilon_{t-k} + \sum_{j=1}^{\mathrm{P}} \sum_{k=1}^{\mathrm{Q}} b_{jk} \mathrm{Y}_{t-j} \varepsilon_{t-k} + \varepsilon_t. \tag{7.2}$$

The constants a_j, c_k, and b_{jk} are real-valued. The sequence of random variables ε_t is independent and identically distributed, with a mean of zero and a finite variance of σ^2. These models were first proposed and developed by C. W. J. Granger and Andersen, 1978 and later by Liu and Brockwell, 1988. An important model to consider is the first-order superdiagonal bilinear model, denoted as $\mathrm{BL}(0,0,2,1)$. This chapter specifically examines the presence of this model which is defined for $t = 1,2,\ldots,\mathrm{T}$.

$$\mathrm{Y}_t = b\mathrm{Y}_{t-2}\varepsilon_{t-1} + \varepsilon_t, \tag{7.3}$$

with Y_t is an observation at time t described by a nonlinear stochastic difference equation; $\varepsilon_t \overset{i.i.d}{\sim} (0,\sigma^2)$ and b is a constant in $\mathbb{R}$. The probabilistic properties of first-order superdiagonal time series models $\mathrm{BL}(0,0,2,1)$ (such as invertibility and stationarity) have been studied by Pham and Tran, 1981. Let $\mathcal{F}_t(\varepsilon)$ and $\mathcal{F}_t(\mathrm{Y})$ denote the $\sigma-$algebras generated by $\{\varepsilon_s | s \leqslant t\}$ and $\{\mathrm{Y}_s | s \leqslant t\}$, respectively. Then, Equation (7.3) admits a unique stationary solution (Y_t) (i.e., $\mathcal{F}_t(\varepsilon)$-measurable) if and only if $b^2\sigma^2 < 1$. In this case, one can write

$$\mathrm{Y}_t = \sum_{j=1}^{\infty} b^j \varepsilon_{t-2j} \prod_{k=1}^{j} \varepsilon_{t-2k+1} + \varepsilon_t.$$

In addition, equation (7.3) is invertible (i.e., ε_t is $\mathcal{F}_t(\mathrm{Y})$-measurable) if and only if $2b^2\sigma^2 < 1$. In this case, one can write

$$\varepsilon_t = \mathrm{Y}_t + \sum_{j=1}^{\infty} (-b)^j \mathrm{Y}_{t-j} \prod_{k=1}^{j} \mathrm{Y}_{t-k-1}.$$

Several methods, such as the method of moments, the least squares method, and the repeated residual method, have been established in the literature for estimating the parameters of bilinear models. For example, Pham and Tran, 1980, Sesay and Rao, 1991, Bouzaâchane, 2006, and Tan and Wang, 2016. Furthermore, the identification of a bilinear model was first addressed by C. W. J. Granger and Andersen, 1978. They proposed using the AIC to address this issue, even if it meant fitting multiple models. In a subsequent study, Berlinet and Francq, 1990 conducted a comprehensive investigation into identifying the superdiagonal bilinear model, incorporating moments up to the third order. Following a similar methodology, Subba Rao and Da Silva, 1992 examined the identification of bilinear models of the $\mathrm{BL}(p,0,p,1)$ type.

Using the method of moments does not provide us with general results regarding the existence of higher-order moments. However, there are specific cases for which results are available. It is important to note that moments may not exist for all orders, as demonstrated by C. W. J. Granger and Andersen, 1978. Furthermore, Guegan, 1981 provides the

expression of moments up to order 4 for the bilinear model BL$(0,0,2,1)$. We will proceed with this assumption:

$$c(h) = \mathbb{E}(Y_t Y_{t-h}) \quad \forall\ h \geq 0, \quad c(i,j) = \mathbb{E}(Y_t Y_{t-i} Y_{t-j}) \quad \forall\ i,j \geq 0,$$
$$\text{and} \quad c(i,j,k) = \mathbb{E}(Y_t Y_{t-i} Y_{t-j} Y_{t-k}) \quad \forall\ i,j,k \geq 0.$$

Proposition 2 (Guegan, 1981)**.** *Let $(Y_t)_{t\in\mathbb{Z}}$ be the process defined by equation (7.3), where $\varepsilon_t \overset{i.i.d}{\sim} (0,\sigma^2)$. Let $u = b^2\sigma^2$. If $u < 1$, then the fourth-order moments are finite. Under these conditions, we have the following results:*

— ***1st order moment*** *:* $\mathbb{E}(Y_t) = 0,\ \forall\, t \in \mathbb{Z}$

— ***2nd-order moment*** *:* $c(h) = \mathbb{E}(Y_t Y_{t-h})$

$$c(h) = \begin{cases} \mathbb{E}(Y_t^2) = \mathbb{V}ar(Y_t) = \dfrac{\sigma^2}{1-u}, & \text{if } h = 0 \\ 0, & \text{otherwise.} \end{cases}$$

— ***3rd-order moment*** *:* $c(i,j) = \mathbb{E}(Y_t Y_{t-i} Y_{t-j})$

$$c(i,j) = \begin{cases} \dfrac{b\sigma^4}{1-u}, & \text{if } i = 1, j = 2 \\ 0, & \text{otherwise.} \end{cases}$$

Proof.

$$\begin{aligned} c(1,2) &= \mathbb{E}(Y_t Y_{t-1} Y_{t-2}) \\ &= \mathbb{E}\left[(\varepsilon_t + b\varepsilon_{t-1}Y_{t-2})(\varepsilon_{t-1} + b\varepsilon_{t-2}Y_{t-3})Y_{t-2}\right] \\ &= \mathbb{E}\left[(\varepsilon_t\varepsilon_{t-1} + b\varepsilon_t\varepsilon_{t-2}Y_{t-3} + b\varepsilon_{t-1}^2 Y_{t-2} + b^2\varepsilon_{t-1}Y_{t-2}\varepsilon_{t-2}Y_{t-3})Y_{t-2}\right] \\ &= \mathbb{E}\left[\varepsilon_t\varepsilon_{t-1}Y_{t-2} + b\varepsilon_t\varepsilon_{t-2}Y_{t-3}Y_{t-2} + b\varepsilon_{t-1}^2 Y_{t-2}^2 + b^2\varepsilon_{t-1}Y_{t-2}^2\varepsilon_{t-2}Y_{t-3}\right] \\ &= b\sigma^2\mathbb{E}(Y_{t-2}^2) \\ &= b\sigma^2\frac{\sigma^2}{1-b^2\sigma^2} \end{aligned}$$

4th-order moment : $c(i,j,k) = \mathbb{E}(Y_t Y_{t-i} Y_{t-j} Y_{t-k})$

$$c(0,0,0) = \frac{3\sigma^4(1+u)}{(1-u)(1-3u^2)}, \quad c(0,1,1) = \frac{\sigma^4(1+2u)}{(1-u)^2}, \quad c(0,2,2) = \frac{\sigma^4(1+3u)}{(1-u)(1-3u^2)},$$
$$c(1,3,3) = \frac{2u\sigma^4(1+u)}{1-u}, \quad c(1,1,3) = \frac{2u\sigma^4}{b(1-u)}, \quad c(0,1,4) = \frac{2u^2\sigma^4}{b(1-u)}, \quad c(1,3,4) = \frac{u\sigma^4}{1-u}$$

and $c(i,j,k) = 0$ for all other values of i,j,k. □

The estimation of the parameters b and σ^2 in the bilinear model BL(0,0,2,1): $Y_t = bY_{t-2}\varepsilon_{t-1} + \varepsilon_t$ was discussed by Guegan, 1981. They proposed estimators for the parameters of this model using the method of moments. The expression for these estimators is as follows:

$$\hat{b} = \frac{\hat{c}(0)\hat{c}(0,1,4)}{\hat{c}(1,2)\hat{c}(1,1,3)} \quad \text{and} \quad \hat{\sigma}^2 = \frac{\hat{c}(1,1,3)\hat{c}^2(1,2)}{\hat{c}^2(0)\hat{c}(0,1,4)}. \tag{7.4}$$

7.3.3 Time series forecasting using LSTM model

DL, a subset of ML, utilizes ANNs that are designed to simulate the structure and functionality of the human brain in 1957 by Frank Rosenblat. Specifically, RNNs like LSTM and GRU have gained significant recognition in time series forecasting (Elman, 1990; Rumelhart, Hinton, & Williams, 1986; Werbos, 1988). This is mainly due to their ability to effectively handle noisy data and capture intricate temporal dependencies. While CNNs are also employed for noise reduction and extracting valuable information from data, they are not as commonly used for long and complex temporal sequences (Ghosh et al., 2020). By developing an LSTM model, we can enhance the accuracy of predictions by harnessing the strengths of these networks in capturing temporal patterns (Sen & Dutta Choudhury, 2024) and filtering noise in the stock market (Oukhouya & El Himdi, 2023a).

The Back-propagation training algorithm

The Back-propagation algorithm is a key training method for LSTM, aiming to minimize error by iteratively adjusting connection weights. It begins with input-output pairs, propagating inputs through layers to generate predictions, which are compared to actual outputs using MSE:

$$\mathcal{L}(\omega) = \frac{1}{2\mathrm{N}_{set}} \sum_{i=1}^{\mathrm{N}_{set}} (e_i)^2 = \frac{1}{2\mathrm{N}_{set}} \sum_{i=1}^{\mathrm{N}_{set}} (y_i - \hat{y}_i)^2. \tag{7.5}$$

Our goal is to find a set of weights (i.e. ω) that minimize the error function $\mathcal{L}$, such as N_{set} train or test set. Therefore, we calculate the gradient of the error function, as the optimal weight verifies the condition:

$$\nabla \mathcal{L}(\hat{\omega}) = 0. \tag{7.6}$$

Minimizing this equation is done through the gradient descent iterative method, where the weights are updated under the following formula:

$$\omega_{t+1} = \omega_t + \Delta\omega_t, \tag{7.7}$$

where

$$\Delta\omega_t = -\eta \frac{\partial \mathrm{E}_t}{\partial \omega} = -\eta \nabla \mathrm{E}_t. \tag{7.8}$$

The positive parameter η is called the learning rate, defining the iteration step in the negative gradient direction. After the forward learning process, the generated output is compared with the desired one, resulting in an error that is then backpropagated through the network while adjusting the weights. A momentum term δ can also be introduced to enhance learning stability and speed, resulting in the following learning rule:

$$\Delta\omega_{t+1} = -\eta \frac{\partial \omega}{\partial \mathcal{L}} + \delta\Delta\omega_t, \tag{7.9}$$

where $\delta < 1$.

LSTM model

Addressing the short-term memory constraints inherent in traditional RNNs, the LSTM model, introduced by Hochreiter and Schmidhuber, 1997, is considered a significant advancement (Gers & Schmidhuber, 2001; Gers et al., 2002). Unlike conventional RNNs, LSTMs are designed with a specialized architecture that can preserve important information over long sequences while discarding irrelevant data. This improved functionality is achieved by incorporating memory cells, each equipped with input, forget, and output gates, which allow for precise control over the flow of information (Hochreiter & Schmidhuber, 1996; Staudemeyer & Morris, 2019), given by the following mathematical formulas:

1. **Forget gate**: The forget gate, denoted by f_t, determines which information to retain or discard. It processes the information from the previous hidden state h_{t-1} and the current input x_t using the sigmoid function σ, where 1 indicates retention and 0 indicates forgetting. Mathematically, it is calculated as:

$$f_t = \sigma(\mathrm{W}_{fh} \cdot h_{t-1} + \mathrm{W}_{fx} \cdot x_t + b_f). \tag{7.10}$$

2. **Input gate**: The input gate, denoted by i_t, updates the cell state through a three-step process. First, x_t and h_{t-1} are inputted into a sigmoid function to regulate the values that will be added to the state. This is formulated as:

$$i_t = \sigma(\mathrm{W}_{ih} \cdot h_{t-1} + \mathrm{W}_{ix} \cdot x_t + b_i). \tag{7.11}$$

   Next, x_t and h_{t-1} are passed through a hyperbolic tangent function (tanh) to scale the values between -1 and 1, enhancing the network's fitting capability:

$$\tilde{\mathrm{C}}_t = \tanh(\mathrm{W}_{ch} \cdot h_{t-1} + \mathrm{W}_{cx} \cdot x_t + b_c). \tag{7.12}$$

   Finally, the old cell state C_{t-1} is updated by element-wise multiplication of the outputs from the sigmoid and tanh functions ($i_t \cdot \tilde{\mathrm{C}}_t$). The new cell state C_t is obtained by combining the results as follows:

$$\mathrm{C}_t = f_t \odot \mathrm{C}_{t-1} + i_t \odot \tilde{\mathrm{C}}_t. \tag{7.13}$$

3. **Output gate**: The output gate, denoted by o_t, determines the next hidden state h_t, which contains information about the previous inputs. First, x_t and h_{t-1} are passed through a sigmoid function to determine the portion of the LSTM cell to output:

$$o_t = \sigma(\mathrm{W}_{oh} \cdot h_{t-1} + \mathrm{W}_{ox} \cdot x_t + b_o). \tag{7.14}$$

   Next, the new cell state C_t is passed through the tanh function. The updated hidden state h_t, used for predictions at time $t+1$, is obtained by element-wise multiplication of the tanh output with the sigmoid output:

$$h_t = o_t \odot \tanh(\mathrm{C}_t), \tag{7.15}$$

   with W_{fh}, W_{ih}, W_{oh}, W_{ch} are the weight matrices that interconnect the gates and the cell state, respectively; W_{fx}, W_{ix}, W_{ox}, W_{cx} represent the input weight matrices for the gates and the cell state, respectively; b_f, b_i, b_o, b_c are the corresponding bias terms; $\odot$ denotes the element-wise product of matrices.

7.3.4 Time series forecasting using hybrid model

The hybrid approach of integrating ARIMA and Bilinear models consists of a two-phase methodology. In the first phase, the optimal ARIMA model is used to capture the linear characteristics of the time series data. In the second phase, the Bilinear model accounts for the nonlinear residuals obtained from the fitted ARIMA series. This process involves modeling the error term ε_t of the ARIMA model as a Bilinear process of order $(p, q, \mathrm{P}, \mathrm{Q})$. Figure 7.2 provides a visual representation of this hybrid procedure.

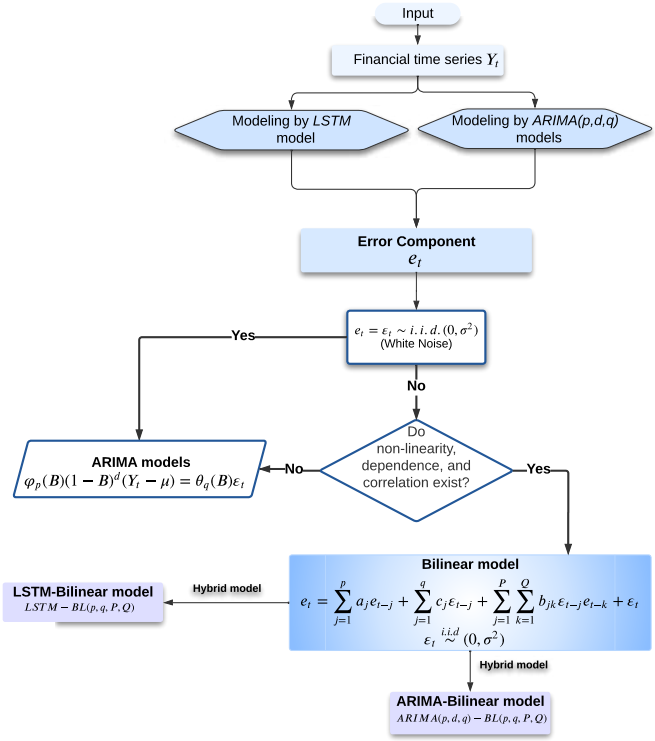


Figure 7.2 – Flowchart of the procedure of the ARIMA-Bilinear and LSTM-Bilinear models

Linear time series models such as AR, MA and ARMA can be combined with Bilinear models to model the dynamics of stock indexes and their volatilities. The ARMA-Bilinear model is similar to the combination of the AR-Bilinear and MA-Bilinear models proposed by Maravall, 1983. In particular, each component of the mixture model can be represented as a regular ARIMA series:

$$\mathrm{Y}_t = \sum_{j=1}^{p} a_j \mathrm{Y}_{t-j} + \sum_{j=1}^{q} c_j e_{t-j} + e_t. \tag{7.16}$$

Furthermore, each error term e_t is assumed serially dependent, denoted by the hybrid ARIMA$(p, d, q)-$BL$(0, 0, \mathrm{P}, \mathrm{Q})$ model:

$$e_t = \sum_{j=1}^{\mathrm{P}} \sum_{k=1}^{\mathrm{Q}} b_{jk} \varepsilon_{t-j} e_{t-k} + \varepsilon_t, \tag{7.17}$$

where $\varepsilon_t \overset{i.i.d}{\sim} (0, \sigma^2)$.

7.3.5 Performance Evaluation of Financial Time Series Models

Performance evaluation metrics are essential for assessing the effectiveness and reliability of forecasting models in the stock market, where the environment is dynamic and volatile. Various metrics, such as ME, MAE, RMSE, Theil's U, and R^2, are commonly used to measure the accuracy and predictive power of financial time series models (Leung & Zhao, 2021). These metrics offer quantitative measures of a model's ability to capture and predict stock market fluctuations, enabling investors and analysts to make well-informed decisions. We describe in subsection 2.8.1 mathematical formulations of these key performance evaluation metrics. These metrics provide valuable insights into the performance and accuracy of forecasting models, assisting market participants in making informed investment decisions.

7.4 Results and Discussions

In this section, we present the outcomes obtained from applying various models to analyze the time series data of ARIMA, Bilinear, LSTM, and their hybrid counterparts. The main focus is on predicting the MSI 20. We start by introducing the predictions generated by ARIMA and LSTM models using the Box-Jenkins Methodology. After that, we carefully select the best model and analyze its residuals, with a specific emphasis on the ARIMA model. Next, we compare the residuals of MSI 20 log closing price forecasts across four different models and their hybrid variants, using both training and test datasets. For our experiment, the series of residuals in MSI 20 prices from April 12, 2022, to January 30, 2024, serve as the training set, while those from January 31, 2024, to April 9, 2024, make up the test set. Since our dataset covers the period after the COVID-19 crisis, we thoroughly investigate the impact of the crisis on MSI 20 closing prices across the proposed models. We begin by presenting the results of the linear model, followed by the ARIMA model, and conclude with a detailed discussion on the outcomes of the nonlinear models (ARIMA-Bilinear, LSTM-Bilinear).

7.4.1 Model Identification

In this study, we focus on the daily closing price as the representative measure for the MSI 20. We chose this because daily closing prices capture the entire day's index activities for the 20 most liquid companies listed on the CSE. We obtained the daily closing price data for the MSI 20 index from April 12, 2022, to April 9, 2024, from the Investing website[2]. This data consists of 503 daily observations, denominated in MAD, with trading sessions occurring from Monday to Friday, 9:00 AM to 3:30 PM (GMT/GMT+1 during daylight saving time) on the CSE. Before conducting the analysis, we carefully checked the original MSI 20 time series closing price data to ensure there were no missing values or outliers. Figure 7.3 illustrates the original series, which exhibits noticeable peaks and swings, indicating significant fluctuations (volatility) over time. The use of natural logarithms has been effective in addressing specific statistical characteristics and improving series attributes, such as variance stabilization and enhancing linear relationships. Additionally, the logarithmic transformation reveals an evident upward trend in closing prices throughout the observed period. Although there are no recurring patterns indicating seasonality in the series, the increasing trend demonstrates that the series is non-stationary.

2. https://www.investing.com/indices/masi-20

To fit an ARIMA model to the log-transformed data, as shown in Figure 7.6, we apply the first difference, which makes the series stationary. This transformation helps produce more reliable results, especially when analyzing volatile financial data.

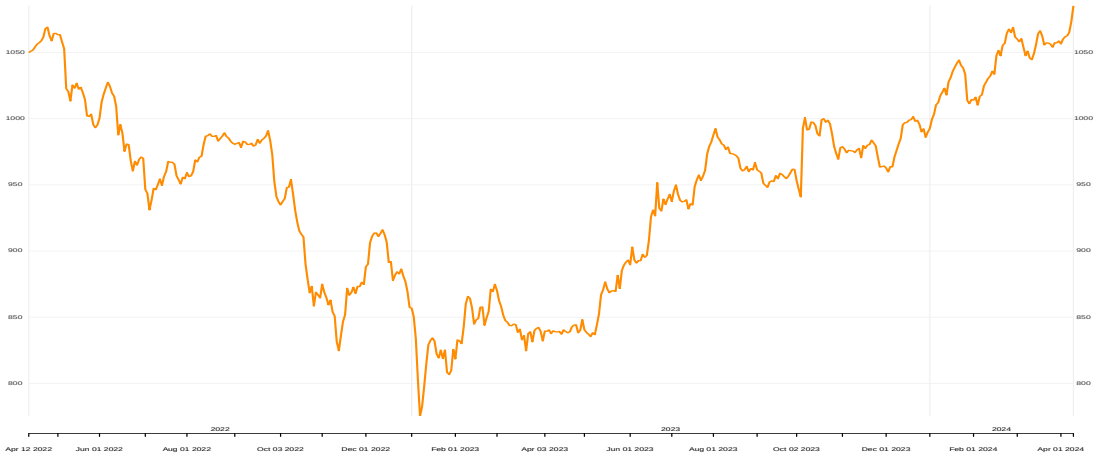


Figure 7.3 – Daily closing price of MSI 20

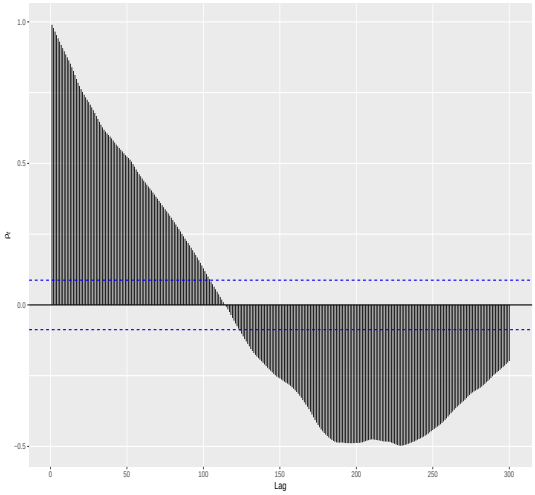


Figure 7.4 – ACF of the close price of MSI 20.

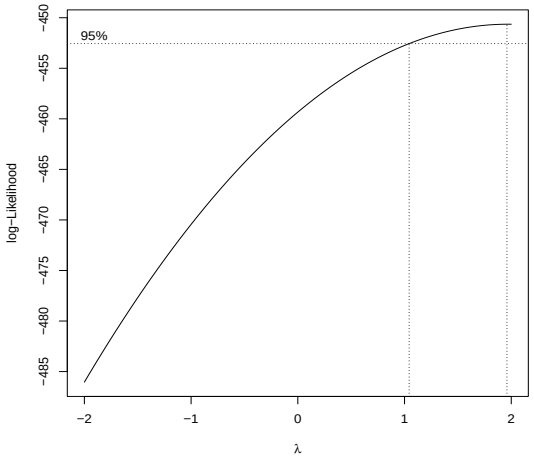


Figure 7.5 – Dashed vertical line represents the estimated parameter $\hat{\lambda} = 1.999$, while the other lines indicate the 95% confidence interval of the estimation.

The Box-Cox functions transformations are given for different values of λ by the following expression:

$$\begin{cases} \frac{Y_t^{\hat{\lambda}}-1}{\hat{\lambda}} & \text{if} \quad \hat{\lambda} \neq 0 \\ \log(Y_t) & \text{if} \quad \hat{\lambda} = 0 \end{cases} \tag{7.18}$$

In Figure 7.4, the ACF plots of the closing price series provide valuable insights into the MSI 20. Notably, the ACF plot depicts autocorrelation values that do not decrease exponentially towards zero, indicating significant autocorrelation and potential non-stationarity in the series. To address this issue, a common practice in financial time series analysis involves applying a log transformation or Box-Cox transformation to capture the stock's growth rate and stabilize variance. This is illustrated in Figure 7.5, where the optimal transformation parameter $\hat{\lambda} = 1.999$ is determined. This transformation is instrumental in scaling the data for enhanced analysis. While the log transformation yields a new series that may exhibit improved stationarity, it is imperative to consider the specific data

characteristics and analytical requirements when deciding on this transformation. Additionally, further investigation of stationarity using techniques such as the ADF test is advisable. The ADF test offers statistical evidence of stationarity and aids in selecting appropriate modeling techniques. Furthermore, to identify the orders of the ARIMA model for the training data, we rely on diagnostic tools such as the ACF and the PACF. These tools play a crucial role in discerning the stationarity status of the series and determining the necessary AR and MA terms, denoted as p and q, respectively. As depicted in Figure 7.7, our analysis of the training data's ACF and PACF reveals significant trends. Specifically, the training ACF displays rapid decay from its unity initial value at zero lag, suggesting a stationary series. Additionally, the training PACF exhibits exponential decay after lag 1, further supporting the stationary nature of the series. These observations guide us in determining suitable p and q values for our ARIMA modeling efforts.

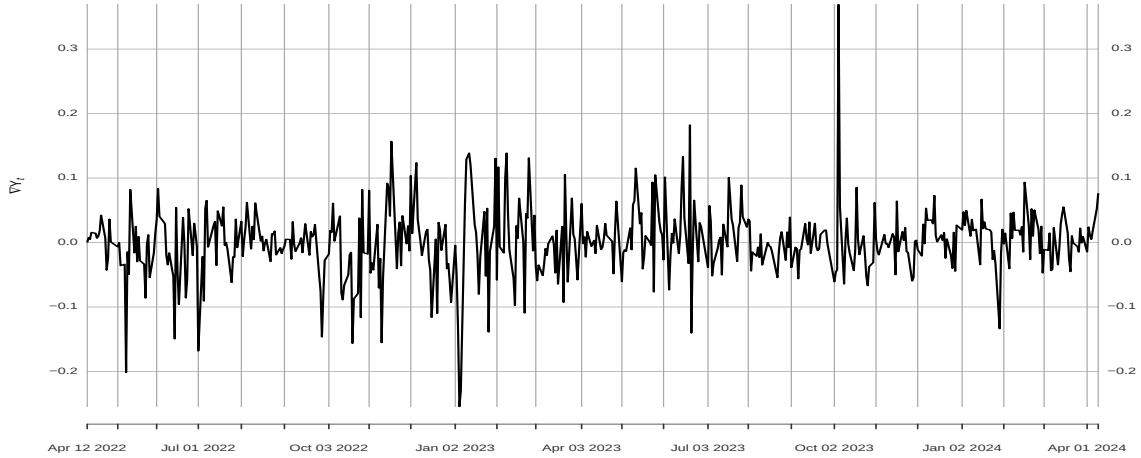


Figure 7.6 – Difference-Transformed Logarithm of Daily Closing Prices for MSI 20 Index.

7.4.2 Parameter estimation

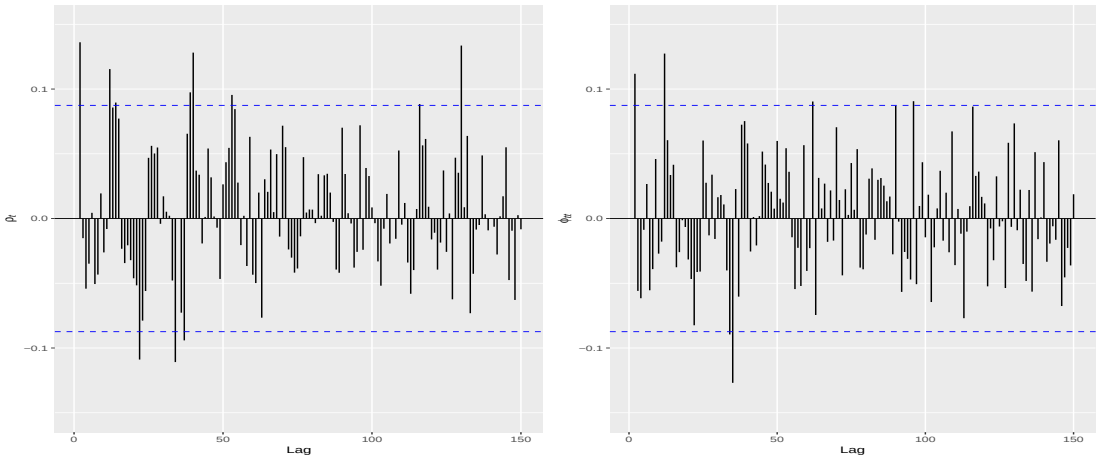


Figure 7.7 – ACF and PACF plots of log daily closing prices transformed by difference for the MSI 20 index.

ARIMA(0,0,2), ARIMA(2,0,0), and ARIMA(1,0,1). These models were selected based on their performance. Parameter estimation was conducted for each model, the SE of the

estimated parameters are in (.), and the resulting equations are presented below:
For the ARIMA(1,1,1) model:

$$\underset{(0.149)}{(1-0.458\mathrm{B})}\hat{y}_t = \underset{(0.157)}{(1+0.292\mathrm{B})}\varepsilon_t, \tag{7.19}$$

with $\hat{y}_t$ represents the forecasted values, and ε_t denotes the error term. Similarly, for the ARIMA(0,1,2) model:

$$\hat{y}_t = (1 - \underset{(0.046)}{0.149\mathrm{B}} - \underset{(0.049)}{0.162\mathrm{B}^2})\varepsilon_t. \tag{7.20}$$

Lastly, for the ARIMA(2,1,0) model:

$$(1 - \underset{(0.047)}{0.147\mathrm{B}} - \underset{(0.047)}{0.113\mathrm{B}^2})\hat{y}_t = \varepsilon_t. \tag{7.21}$$

The findings presented in Table 8.2 provide a comprehensive assessment of various ARIMA models applied to forecast the transformed log MSI 20 index time series. We can gain insights into their effectiveness by evaluating both the fit and complexity of each model. Log-likelihood values indicate how well each model explains the observed series, with higher values indicating a better fit. Meanwhile, AIC, AICc[3], and BIC values provide a framework for model selection, balancing fit quality with model complexity. Notably, among the evaluated models, the ARIMA(1,1,1) model emerges as the most favourable candidate, with the lowest log-likelihood and lowest AIC, AICc, and BIC values. This suggests that it balances explanatory power and simplicity, offering a concise representation of the series and promising potential for accurate forecasting. The estimation of

Table 7.2 – Performance Comparison of ARIMA Models on Training Set

Models	Log likelihood	AIC	AICc	BIC
ARIMA(1,1,1)	674.62	-1341.24	-1341.16	-1324.78
ARIMA(0,1,2)	677.17	-1346.33	-1346.24	-1329.87
ARIMA(2,1,0)	676.11	-1344.23	-1344.14	-1327.77

parameters for the LSTM model applied to the financial time series of MSI 20 closing prices involves transforming the residual series into a supervised learning model. This transformation involves establishing a target variable (Y) and a predictor (X) for the LSTM model. By shifting the dataset by a lag of k, we create input-output pairs where the input at time $(t-k)$ is used to predict the output at time t, followed by normalization. The loss function employed is MSE, optimized using the ADAM algorithm, with specified learning rate decay for each update. The architectural design of the LSTM model comprises three layers: an LSTM layer responsible for capturing long-term dependencies within the log-transformed MSI 20 series, a Dense layer tasked with feature extraction, and a Dropout layer to mitigate overfitting. With a total of 14 trainable parameters, the model maintains a relatively compact and efficient structure. The model was implemented using the `Keras` (F. K. Chollet, 2015) and `TensorFlow` (Abadi et al., 2016) frameworks, following the specifications outlined in this study. During the training phase, the LSTM model was optimized using the ADAM optimizer with a learning rate of 0.05 and a decay rate of 10^{-6}.

3. AICc is a correction of AIC for small sample sizes.

The dataset was partitioned into batches of size 1 to facilitate efficient training iterations. These parameters and strategies were meticulously selected to ensure robust model training and effective capture of the underlying dynamics within the MSI 20 closing price time series.

7.4.3 Diagnostic checking

The statistical tests presented in Table 7.3 reveal valuable information about the characteristics of the residuals generated by the ARIMA(1,1,1) model. The absence of significant autocorrelation, as detected by the Box-Ljung test, indicates that consecutive residuals are independent, strengthening the model's credibility. However, the Jarque Bera Test reveals a departure from the Gaussian distribution in the residuals, suggesting potential complexities that traditional parametric models do not capture. In addition, the Kendall's rank correlation tau test shows no significant correlation between residuals and their lagged values, confirming their independence. Furthermore, the Runs Test underscores the randomness of the residuals, reinforcing their suitability for modelling purposes. However, the BDS Test reveals substantial evidence of non-linear dependence in the residuals, highlighting the complexities inherent in the underlying generating process of the series.

Table 7.3 – Statistical Tests Results on Residuals of ARIMA(1,1,1) Model

Tests	**Statistics**	**p-value**	**Type**	**Decision of** H_0
Ljung-Box	10.025	0.438	Autocorrelation	Accepted
Jarque Bera	791.41	< 2.2e-16	Normality	Rejected
Kendall	-0.910	0.363	Correlation	Accepted
Runs	-0.418	0.676	Randomness	Accepted
Brock, Dechert and Scheinkman (BDS)	d=2	(0,0)	Non-linear dependence	Rejected

7.4.4 Forecasting model results

In our prediction process, we exclusively use the residuals of the ARIMA(1,1,1) and LSTM models as inputs in our hybrid BL(0,0,2,1) model. We make this idee choice because these models have the capability to capture both linear and nonlinear dynamics, resulting in residuals that closely align with the forecasted values of the hybrid model. Using the ARIMA(1,1,1) model, we generate residual forecasts for the entire test set, as shown in Figure 7.9. Figure 7.8 displays the residual series of the ARIMA model (blue line) alongside the forecast values of the hybrid Bilinear model (red line), derived using the ARIMA model trained on the training set (i.e., ARIMA(1,1,1)-BL(0,0,2,1)). Importantly, Figures 7.8 and 7.9 highlight the close correspondence between the predicted values of the ARIMA residuals and those of the proposed hybrid Bilinear model, observed across various instances in both the training and test datasets. This alignment is further supported by the performance metrics of the selected ARIMA(1,1,1)-BL(0,0,2,1) hybrid model, presented in Table 7.4, which demonstrate its superior performance compared to the standalone ARIMA(1,1,1) model. Moving forward, Figure 7.10 illustrates the residuals and predicted LSTM-BL(0,0,2,1) hybrid model for the training set of the developed LSTM model. However, the results shown in Figure 7.11 reveal limitations in the predictive capabilities of

the LSTM-BL(0,0,2,1) hybrid model for test set. While the predicted residual values of the LSTM exhibit a tendency to align with those of the hybrid model in the test set, the hybrid model fails to accurately forecast peaks or sudden changes in the residual series. This inadequacy highlights the unsatisfactory performance of the LSTM-BL(0,0,2,1) hybrid model in capturing the complex dynamics of the data.

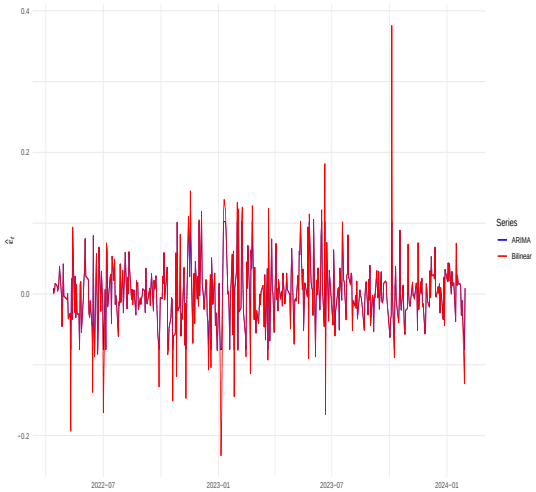


Figure 7.8 – Comparison of ARIMA(1,1,1) and ARIMA-BL(0,0,2,1) time series residuals (Training)

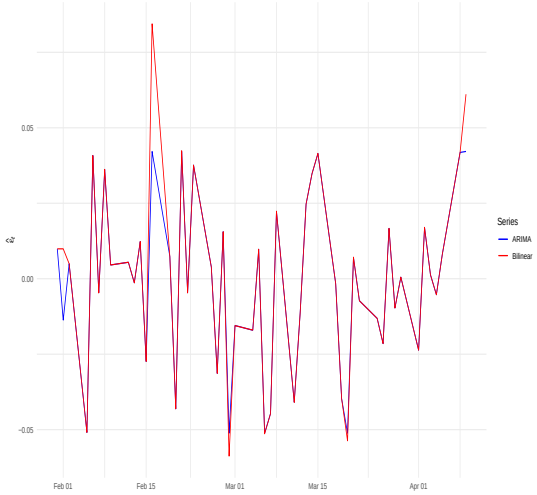


Figure 7.9 – Comparison of ARIMA(1,1,1) and ARIMA-BL(0,0,2,1) time series residuals (Testing)

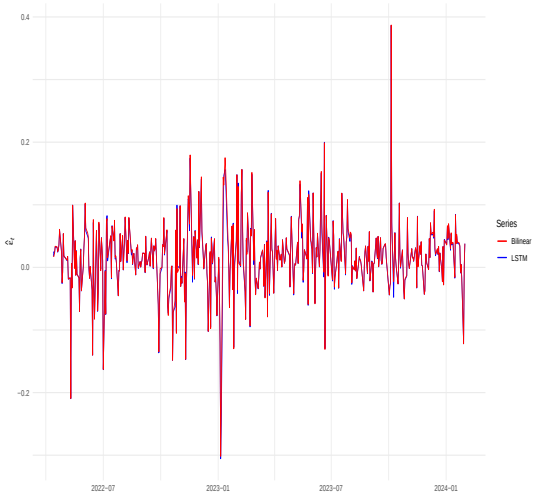


Figure 7.10 – Comparison of LSTM and LSTM-BL(0,0,2,1) time series residuals (Training)

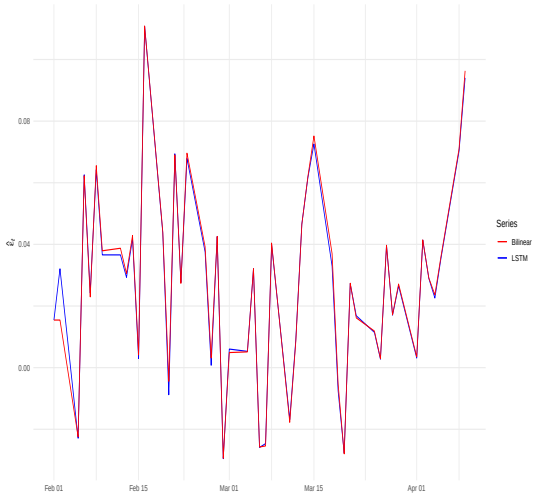


Figure 7.11 – Comparison of LSTM and LSTM-BL(0,0,2,1) time series residuals (Testing)

To conclude the results section, we compare the performance of various models using five statistical evaluation criteria on both the training and test sets. The ARIMA, LSTM, and their hybrid bilinear models were estimated using the training residuals. After estimating the models, we empirically evaluate their performance on out-of-sample test residuals. Table 7.4 presents the prediction results of residuals for each model, providing insights into their accuracy based on the evaluation criteria. The comparison reveals that the hybrid LSTM-BL(0,0,2,1) model demonstrates superior efficiency and accuracy compared to the ARIMA(1,1,1)-BL(0,0,2,1) model on the training set. This is evidenced

Table 7.4 – Performance metrics for forecasting residual hybrid models

Metrics	ARIMA(1,1,1) - BL(0,0,2,1)		LSTM-BL(0,0,2,1)	
	Training	Testing	Training	Testing
ME	0.001	0.011	0.001	0.018
MAE	0.007	0.011	0.002	0.018
RMSE	0.010	0.013	0.005	0.018
Theil's U	0.493	0.746	0.001	0.345
R^2	0.967	0.940	0.993	0.992

by the values of ME, MAE, RMSE, Theil's U, and R^2. However, for the test set, the performance of the ARIMA(1,1,1)-BL(0,0,2,1) model outperforms the others. These findings suggest that, on average, the hybrid LSTM-BL(0,0,2,1) model emerges as the most effective model for forecasting the daily closing price of the MSI 20 index. Nonetheless, the ARIMA(1,1,1)-BL(0,0,2,1) model demonstrates notable performance in specific scenarios. This underscores the importance of considering the characteristics of the dataset and the evaluation criteria when selecting the appropriate forecasting model. Figure 7.12 com-

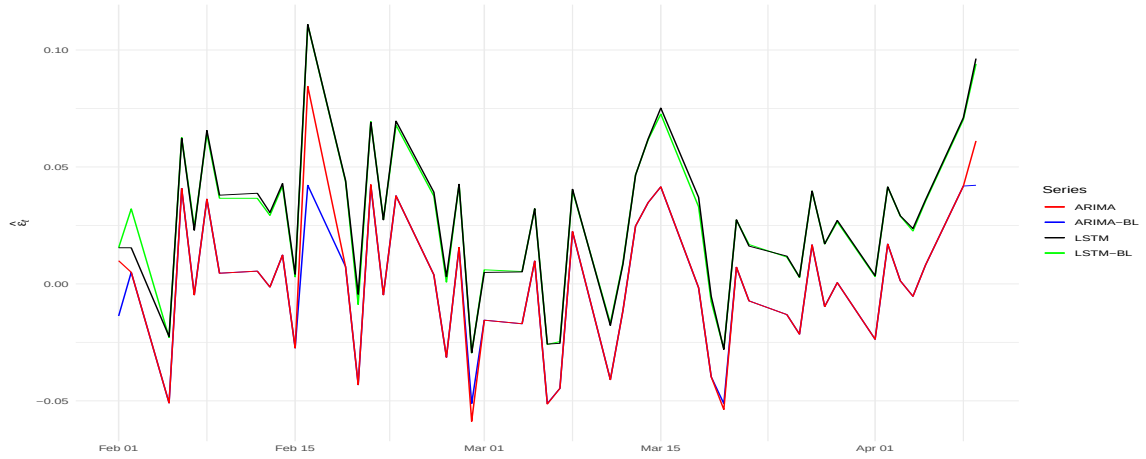


Figure 7.12 – Comparison of forecasting residuals MSI 20 using all models (Test set)

pares the residuals of four models used to forecast the Morocco Stock Index (MSI) 20: ARIMA, ARIMA-BL, LSTM, and LSTM-BL. Residuals indicate the difference between the actual values of MSI 20 and the values predicted by the models on the test set. In Figure 7.12, it is apparent that the residuals of the ARIMA and ARIMA-BL models are larger in magnitude compared to the residuals of the LSTM and LSTM-BL models. This suggests that the LSTM and LSTM-BL models are better fitting the test residuals compared to the ARIMA and ARIMA-BL models.

7.5 Conclusions

In this work, we examine the dominant literature on forecasting financial time series. Specifically, we compare the effectiveness of hybrid DL models to single models in both classical and ML methodologies. Our main objective is to assess the predictive performance of hybrid bilinear-based algorithms, such as the LSTM model, in comparison to

traditional approaches like the ARIMA model. For this investigation, we focus on modelling and predicting the residuals of linear and nonlinear models applied to the MSI 20 index, a prominent benchmark in the Moroccan stock market. We fit linear ARIMA and nonlinear LSTM models to log-transformed MSI 20 closing price data and generate forecasts for future residuals in both training and testing sets.

By evaluating our findings using various performance metrics, we confirm our hypothesis that the hybrid LSTM-BL(0,0,2,1) model outperforms the hybrid ARIMA-BL(0,0,2,1) model in terms of predictive accuracy. This comparative analysis contributes new insights into the effectiveness of hybrid models within the context of Moroccan stock market dynamics, addressing a research gap in the existing literature and the lack of performances for the single models. However, our study is limited by the uncertainty surrounding optimal parameters for the hybrid bilinear model, which requires a trial-and-error approach for parameter selection. In future endeavours, we plan to expand our investigation to include other hybrid model forecasts, particularly those integrated into realized volatility forecasting with GARCH multivariate LSTM models. Such research has the potential to inform algorithmic trading strategies tailored to individual investors and major financial entities within the Moroccan market landscape.

Chapter 8

Machine Learning Models-Based Forecasting Moroccan Stock Market

“ The best way to predict the future is to create it. ”

–Abraham Lincoln

Contents

8.1 Introduction

In recent years, the global financial markets and wealth have faced significant upheaval due to crises like the 2008 recession, the 2015 stock market crash, the COVID-19 pandemic, and the Russian invasion of Ukraine. These events have left an indelible mark on the financial landscape, causing widespread disruptions and substantial wealth destruction on a global scale. Inflation nearly doubled only in the last year, financial markets shifted from all-time highs in 2021 to new lows, and experts are warning that worse is yet to come. With the emergence of the 21st century with new technologies and methods, such as AI, Big Data, ML, Blockchain, Internet of Things (IoT), and Generative AI, researchers have turned to statistical algorithms and models that empower computers to learn from data and make predictions or decisions autonomously, marking a significant shift in various fields (Samuel, 1959a).

These ML methods are applied in various industries, including healthcare, finance, energy, computer vision, and NLP. Predicting and modeling stock market behaviour poses a formidable challenge encountered by engineers and financial researchers alike. It helps large financial companies to understand their investments better and to choose the right time to invest according to the projects, news, and concerns about economic conditions. Stock prices fluctuate constantly, so anticipating them may be highly profitable for a business company. The literature review analyzed ML and classical models for stock market prediction studies and found that the model of choice for trend prediction in stock prices was predominantly DL (Mintarya et al., 2023; Ozbayoglu et al., 2020). Some of these studies have utilized various methods, such as fundamental and technical indicators (AGGRAWAL & Dhawan, 2022) and sentiment analysis (Lin et al., 2022), to make precise forecasts about stock prices. Traditional econometric methods, such as ES, regression analysis, and ARIMA, may excel in capturing patterns within nonlinear time series data, but they may not be the most suitable choice for directly forecasting stock prices due to their limitations in handling stock price dynamics (R. Li et al., 2022).

We examine various methods for forecasting complex financial time series. Specifically, we explore the efficacy of individual models such as SVR, MLP, XGBoost, LSTM, and their hybrid models. Our results illustrate that through improvements in the GS optimization algorithm, we can effectively identify nonlinear relationships in stock price data (T. Kim & Kim, 2019), resulting in improved fitting outcomes through the tuning of multiple parameters (Xu et al., 2022). In ML, hyperparameter optimization or tuning involves selecting the most suitable hyperparameters for a specific learning model (Yang & Shami, 2020). Previous studies have employed various optimization algorithms, including GS (Rubio & Alba, 2022), even pigeon-inspired optimization algorithms (Al-Thanoon et al., 2022), and Bayesian approaches (Liwei et al., 2021; Xia et al., 2017). In this study, we developed the GS algorithm to fine-tune the parameters of each model. We also identified the variables for each sector that are important for the investment. Subsequently, the XGBoost, SVR, MLP, LSTM, and their hybrid models are employed to forecast the new test set. We assess and contrast the effectiveness of these models by employing four key metrics: RMSE, MAE, R^2, and MAPE.
The primary aim of this study is to optimize the performance of the GS optimization algorithm by optimizing hyperparameters in ML models. Additionally, the research involves a comparative analysis of one ML and two DL individual models and their hybrid models. To achieve this objective, an extensive examination encompassing 34 pertinent studies focusing on ML models for stock market prediction was undertaken. The pri-

mary aim of this chapter is to enrich the domain of stock market prediction by enhancing the precision of both individual and hybrid ML and DL models, through the refinement of hyperparameter settings. Our findings, presented in Table 8.1, demonstrate superior predictive capabilities compared to other studies in the field. The rest of this chapter is

Table 8.1 – Comparative Analysis of ML and DL Models in Previous Financial Time Series Forecasting Studies.

Markets	**Year**	**Comparative Models**	**RMSE**	**MAPE(%)**	**R^2(%)**	**Reference**
Europe	2017	LSTM, SVR	23.640	——	——	Chniti et al., 2017
Irlande	2019	MA, SVR, LSTM,	347.46	1.03	83.00	Sai Krishna and McCrae, 2019
China	2020	MLP, CNN, RNN, LSTM, CNN-RNN	39.688	——	96.50	Lu et al., 2020
Iran	2020	ARIMA, SVR, RF, ANN	142.16	1.147	——	Khedmati et al., 2020
Australian	2021	LSTM, IPSO, IPSO-LSTM	67.322	0.724	97.60	Ji et al., 2021
American	2022	ARIMA, XGBoost, LSTM	6.1010	3.800	96.10	Goverdhan et al., 2022
Jordan	2022	LSTM, CNN, Bi-LSTM, GRU, MLP	1.579	7.624	——	Alkhatib et al., 2022
Morocco	**2023**	**SVR, MLP, LSTM, SVR-XGBoost, MLP-XGBoost, LSTM-XGBoost**	**1.4974**	**1.89**	**99.82**	**Our Study**

structured as follows: Section 8.2 conducts a comprehensive review of prior literature that explores the applications of ML and DL models in finance. In Section 8.3, we delineate the methodology employed for modeling and forecasting financial time series, with a particular emphasis on stock prices. The presentation of our study's implementation, results, and subsequent discussion are presented in Sections 8.4. Lastly, in Section 8.5, we propose general conclusions and some limitations, as well as identify potential directions for future research in this area.

8.2 Review of Existing Research

In recent years, substantial research efforts have been directed toward the prediction of stock market prices. The financial market is a complex, non-parametric dynamic system, and there are two main approaches to forecasting stock prices: traditional analysis and ML methods. Traditional analysis methods rely on time series models, such as the Box Jenkins methodology, while ML methods utilize a variety of algorithms, such as DL models. A comparative study examined the efficiency of these methods for stock market forecasting. This section offers a concise overview of the existing studies.

Plakandaras et al., 2015 introduced a prediction model designed to anticipate daily and monthly spot prices for five exchange rates using a hybrid approach that combines autoregressive and structural methodologies. The authors implemented two versions of this approach, which included Ensemble Empirical Mode Decomposition (EEMD), SVR, NN, and Multivariate Adaptive Regression Splines (MARS). The study showed that the EEMD-MARS-SVR methodology consistently outperformed the reference random walk model for out-of-sample forecasts. In a recent study Nejad and Ebadzadeh, 2024, the authors introduced a novel framework based on Deep Regret Analytic Generative Adversarial Network (DRAGAN) and feature matching. The study found that this method outperformed the LSTM model, which was used as a baseline method when applied to several stocks. In contrast, a distinct investigation Sai Krishna and McCrae, 2019 revealed that the utilization of LSTM in conjunction with MA leads to superior outcomes in stock price prediction when compared to SVR, as evaluated through various performance metrics, including MAE, MAPE, MSE, RMSE, and R^2 scores. In a study Al-Nefaie and Aldhyani, 2022, investigated the application of AI techniques, specifically MLP and LSTM models, for predicting variations in the Saudi stock market. Their findings indicated a strong correlation coefficient exceeding 0.995 between the two models. Notably, the LSTM model exhibited superior accuracy and demonstrated the highest capacity for model fitting. In a study conducted by the authors Goverdhan et al., 2022, a comparison was made between LSTM, XGBoost, and ARIMA models in terms of their performance measures for forecasting trends in stock markets. Performance measures such as MSE, RMSE, MAE, and MAPE were utilized for evaluation. The results of their tests indicated that XGBoost outperformed the other models and exhibited the highest forecasting accuracy. Khedmati et al., 2020 proposed and analyzed various modeling and forecasting approaches for Bitcoin price, including classical methods like ARIMA and ML algorithms such as Kriging, Bayesian method, SVR, ANN, and RF. The study revealed that SVR outperformed all the other models with performance measures of RMSE = 142.158 and MAPE = 1.147%, significantly smaller than the values obtained by other models. In their paper Shi et al., 2022, the authors integrated a time series pattern, an XGBoost regressor, CNN with an attention mechanism, and an LSTM network in a nonlinear relationship. Their results indicate that the hybrid model is more effective, resulting in relatively high accuracy for stock prediction. In their article Song and Choi, 2023, the authors delve into applying hybrid models based on RNNs for predicting stock market indices. The proposed models, namely GRU-CNN, CNN-LSTM, and ensemble models, undergo comprehensive evaluation and comparison against conventional benchmarks across various periods. The study reveals that integrating diverse RNN-based models can enhance forecasting performance and facilitate the construction of portfolios using predicted stock market prices. The results of their investigation indicate that the hybrid model exhibits improved efficacy, resulting in a comparatively elevated level of precision in stock forecasting.
In a separate context, ML methods, such as MLP and ANNs, were examined for solar irradiance prediction Ettayyebi and El Himdi, 2018a, with results favouring the MLP model when incorporating exogenous variables. Another study Kumari and Toshniwal, 2021 utilized eXtreme Gradient Boosting Deep Neural Networks (XGBF-DNN) to forecast hourly global horizontal irradiance, utilizing the GS algorithm for its superior predictive capabilities and stability.

8.3 Methods

The methodology section of this chapter offers a summary of the methodology employed in this investigation, with data collection [1] in Table 8.2, preprocessing, and implementation of various ML models, in particular SVR, XGBoost, MLP, LSTM, as well as hybrid models such as MLP-XGBoost, SVR-XGBoost, and LSTM-XGBoost. The following subsections describe the procedures at each stage of the methodology and the use of the GS algorithm for hyperparameter tuning.

8.3.1 Data Preprocessing for Proposed Models

In preparation for modeling a multivariate time series, meticulous data processing is essential. This encompasses identifying predictor variables (all 29 index sectors & profitability) and the target variable (stock closing price).

Table 8.2 – Sector Indexes and the MASI Market Index.

Index code	Market & Sector indexes	Companies
ESGI	Environmental, Social and Governance	TOTALENERGIES MARKETING MAROC, TAQA MOROCCO, COSUMAR, CIH, SALAFIN, SODEP-MARSA MAROC
AGRO	Agri-Food Production	MUTANDIS SCA, COSUMAR, LESIEUR CRISTAL, UNIMER, DARI COUSPATE, CARTIER SAADA
ASSUR	Insurance	SANLAM MAROC, AFMA, WAFA ASSURANCE, AGMA, ATLANTASANAD
BANK	Banks	BMCI, BCP, ATTIJARIWAFA BANK, CIH, BANK OF AFRICA, CDM
P&G	Oil & Gas	TOTAL ENERGIES MARKETING MAROC, AFRIQUIA GAZ
⋮	⋮	⋮
MASIR	MASI Gross Profitability	Profitability index
MASRN	MASI Net Profitability	Profitability index
TRANS	Transport	TIMAR, CTM
MASI prices	Open, High, Low, Close, Volume	Market index

$$\text{Data set:} \quad \{(x^{(i)}, y^{(i)}) \in \mathcal{X}, \mathcal{Y}, i = 1, \ldots, 29\}.$$

We standardize the data and partition it into training (90%) and test (10%) datasets, and we use a block of length N = 1310 (over about five years).

$$\mathcal{X} = (x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(29)}, t = 1, \ldots, \mathrm{N}).$$

1. Download the sector indexes and the MASI market index from the website: casablanca-bourse.com

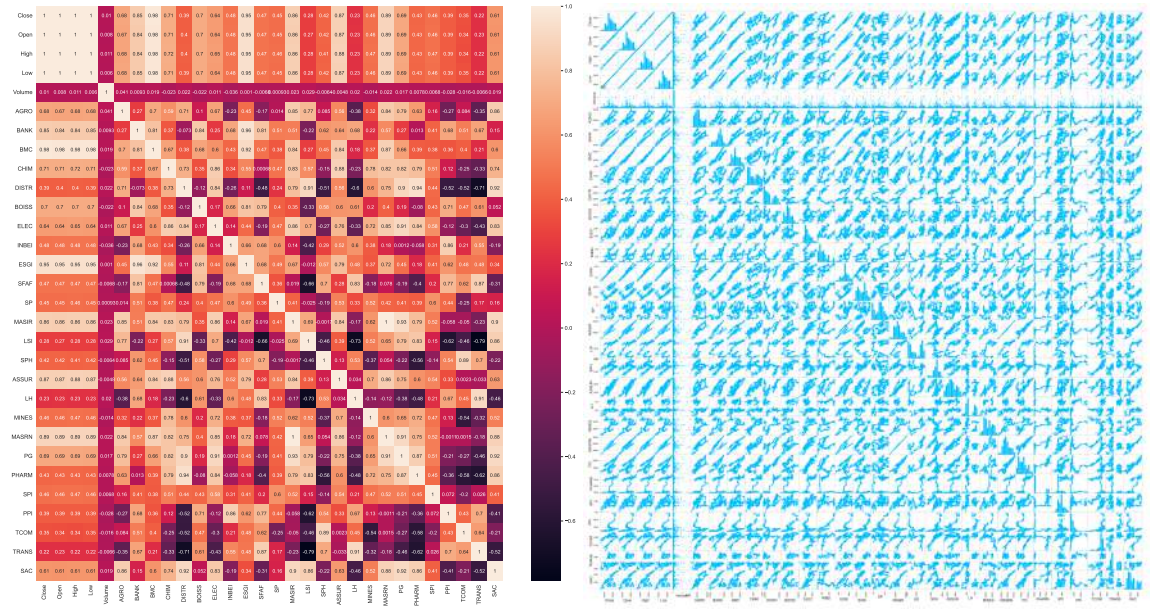


Figure 8.1 – Correlation & Scatterplots between Sector's index and MASI prices.

Our seven models are designed to learn a mapping function from a sequence of past observations (input variables) to the current observation (target variable):

$$\mathcal{Y} = (y_t, t = 1, \ldots, \mathrm{N}).$$

In regression, we use the squared loss function, which is computed as the average of the squared differences between observed and predicted values:

$$\mathcal{L}(\text{data}, \theta) = \frac{1}{\mathrm{N}} \sum_{t=1}^{\mathrm{N}} (y_t^{(i)} - \mathrm{F}(x_t^{(i)}; \theta))^2.$$

Each data point consists of independent features and a desired output, denoted as (x^i, y^i), compared with the models predicted, represented as $\mathrm{F}(x^i; \theta)$, where θ is the ML models parameter. Our analysis of Figure 8.1 reveals strong positive correlations, especially among MASI, BMC, BANK, ESGI, MASIR, and MASRN. Outliers and distinctive patterns in PPI, SPH, TCOM, and others hint at unusual market dynamics. Non-linear relationships predominate, prompting the exploration of advanced ML techniques. Variable distributions vary, with some resembling normal distributions and others displaying skewness, providing valuable insights into market behaviour.

8.3.2 Supervised ML Models

SVR model

SVR is a powerful tool for making predictions based on a dataset of training values and their corresponding target labels. SVR is a regression method to minimize prediction errors by determining the optimal hyperplane (Awad & Khanna, 2015). The following equation defines the SVR model:

$$f_{\mathrm{SVR}}^{(t)}(x) = \mathrm{W}^{\mathrm{T}} \phi(x_t) + b,$$

where, W^T signifies the weight vector, while b indicates the bias term, and $\phi(x_t)$ is the mapping function. The formulation continues as follows:

$$\arg\min \frac{1}{2} \left\| \vec{w}^2 \right\|.$$

In this equation, w signifies the normal vector. The main objective of SVR is to minimize the disparity between the predicted and observed values, thus ensuring robust predictive accuracy (Bathla, 2020). The algorithm of SVR is briefly summarized as mentioned in Algorithm 1.

Algorithm 1 SVR Algorithm

Require: Time series data set normalization $\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$
Ensure: SVR model parameters
1: Parameters = 'kernel': ['linear', 'rbf','poly'], 'C': [10, 100, 1000], 'γ': [0.0001, 0.001, 0.01], 'ε' : [0.1, 0.01, 0.001]
2: Svr = SVR(kernel='rbf')
3: **Procedure** GridSearchCV(Svr, Parameters, cv=5, scoring='MSE')
4: GridSearchCV.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
5: Best_parameters = SVR(kernel, $\hat{C}$, $\hat{\gamma}$, $\hat{\varepsilon}$)
6: **end procedure**
7: Svr.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
8: $\hat{y}_{train} \longleftarrow Svr.predict(\tilde{\mathcal{X}}_{train})$
9: $\hat{y}_{test} \longleftarrow Svr.predict(\tilde{\mathcal{X}}_{test})$
10: **return** kernel, $\hat{C}$, $\hat{\gamma}$, $\hat{\varepsilon}$, $\hat{y}_{test}$, $\hat{y}_{train}$

XGBoost model

XGBoost, a stock market time series forecasting model, utilizes decision trees in a gradient descent algorithm to minimize the loss function (Yun et al., 2021).

$$f^{(t)}_{\text{XGBoost}}(x) = \sum_t \overbrace{\left[h_i g_t(x_i) + \frac{1}{2} f_i g_t^2(x_i) \right]}^{\text{Loss function}} + \underbrace{\Omega(g_t)}_{\text{Regularization}},$$

where h_i and f_i represent statistical measures of the loss function's 1^{st}- and 2^{nd}-order gradients, respectively. Each g_t corresponds to a distinct tree structure with its respective leaf weights w. However, $\Omega(g) = \gamma N + \frac{1}{2}\lambda \sum_{j=1}^{N} w_j^2$ represents a model complexity term used for regularization. The XGBoost model can be summarized in Algorithm 2.

8.3.3 DL models

MLP model

The MLP, a widely used ANN, features an architecture with input (x_t), hidden (h_t), and output layers (y_t). Inputs are weighted (w_i), and their products are combined (Oukhouya & El Himdi, 2023b). The resulting element-wise MLP model is formulated as follows (Tank et al., 2021):

Algorithm 2 XGBoost Algorithm

Require: Time series data set normalization $\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$
Ensure: XGBoost model parameters
1: Parameters = 'max_depth': [30,40,50], 'learning_rate': [0.02,0.03,0.04], 'γ': [0,0.000001,0.00001], 'n_estimators': [300,400,1000], 'reg_alpha': [0.2,0.3,0.4]
2: Xgboost = xgb.XGBRegressor()
3: **Procedure** GridSearchCV(Xgboost, Parameters, cv=5, scoring='MSE')
4: GridSearchCV.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
5: Best_parameters = xgb.XGBRegressor((learning_rate, max_depth, γ, $n\_estimators$, reg_alpha)
6: **end procedure**
7: Xgboost.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
8: $\hat{y}_{train} \longleftarrow Xgboost.predict(\tilde{\mathcal{X}}_{train})$
9: $\hat{y}_{test} \longleftarrow Xgboost.predict(\tilde{\mathcal{X}}_{test})$
10: **return** learning_rate, max_depth, γ, $n\_estimators$, reg_alpha, $\hat{y}_{test}$, $\hat{y}_{train}$

↬ For the first hidden layer:

$$h_t^1 = \eta\Big(\sum_{k=1}^{N} w^{1k} x_{t-k} + b^1\Big),$$

↬ For hidden layers 2 to L − 1:

$$h_t^l = \eta(w^l h_t^{l-1} + b^l), \qquad l = 2, \ldots, L-1$$

↬ and for the output layer:

$$y_{ti} = w^L h_t^{L-1} + b^L + \varepsilon_{ti},$$

where $\eta(\cdot)$ represents an activation function, such as logistic or tanh, and ε_{ti} is a white noise term with zero mean. Algorithm 3, provided below, offers a detailed description of the MLP model's process.

Algorithm 3 MLP Algorithm

Require: Time series data set normalization $\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$
Ensure: MLP model parameters
1: Parameters = 'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,100,100)], 'activation': ['tanh', 'logistic', 'relu', 'identity'], 'α': [0.0001, 0.001, 0.01], 'learning_rate': ['invscaling', 'adaptive', 'constant'], 'solver': ['sgd', 'lbfgs', 'adam'],
2: mlp = MLPRegressor()
3: **Procedure** GridSearchCV(mlp, Parameters, cv=5, scoring='MSE')
4: GridSearchCV.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
5: Best_parameters = MLPRegressor(hidden_layer_sizes, learning_rate, α, activation, solver, adaptive)
6: **end procedure**
7: mlp.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
8: $\hat{y}_{train} \longleftarrow mlp.predict(\tilde{\mathcal{X}}_{train})$
9: $\hat{y}_{test} \longleftarrow mlp.predict(\tilde{\mathcal{X}}_{test})$
10: **return** hidden_layer_sizes, learning_rate, α, activation, solver,adaptive, $\hat{y}_{test}$, $\hat{y}_{train}$

LSTM model

LSTM, a specialized form of RNN, is ideal for sequential data with long-term dependencies. Despite their data requirements, we include LSTM for comparative analysis. LSTM's unique feature is introducing a "cell state" alongside the hidden state to capture long-term patterns, forming the basis for our forecasting approach (Jouilil et al., 2023; Tank et al., 2021).

$$\begin{aligned}
\mathbf{i}_t &= \eta(w^{xi}\mathbf{x}_t + w^{hi}\mathbf{h}_{t-1} + b_i),\\
\mathbf{f}_t &= \eta(w^{xf}\mathbf{x}_t + w^{hf}\mathbf{h}_{t-1} + b_f),\\
\mathbf{o}_t &= \eta(w^{xo}\mathbf{x}_t + w^{ho}\mathbf{h}_{t-1} + b_o),\\
\mathbf{c}_t &= \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * \mathbf{g}_t,\\
\mathbf{h}_t &= \mathbf{o}_t * \eta(\mathbf{c}_t).
\end{aligned}$$

Within the LSTM model, the activation function is represented by $*$ to signify element-wise multiplication. The input, forget, and output gates (represented as $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$) govern the modification of the cell state ($\mathbf{c}_t$) (Oukhouya & El Himdi, 2023b). This modified cell state is subsequently utilized to compute the hidden state ($\mathbf{h}_t$), which is critical for prediction. Algorithm 4, presented subsequently, provides an in-depth explanation of the LSTM model's functioning.

Algorithm 4 LSTM Algorithm

Require: Time series data set normalization $\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$
Ensure: LSTM model parameters
1: Lstm = Sequential()
2: Lstm.add(LSTM(200, input_shape=($\tilde{\mathcal{X}}_{\text{train}}$, 29), activation='relu'))
3: Lstm.add(Dense(100, activation='relu'))
4: Lstm.add(Dense(1))
5: Lstm.compile(loss='MSE', optimizer='adam')
6: early_stopping = EarlyStopping(monitor="loss", patience = 1)
7: history = Lstm.fit($\tilde{\mathcal{X}}_{\text{train}}$, $\tilde{\mathcal{Y}}_{\text{train}}$, validation_data=($\tilde{\mathcal{X}}_{\text{test}}$, $\tilde{\mathcal{Y}}_{\text{test}}$), epochs=30, batch_size=1, callbacks = [early_stopping], verbose=0)
8: Parameters = 'batch_size': [2,3,4], 'neurons_1': [250, 300, 350], 'neurons_2': [220, 250, 280], 'epochs': [75, 80, 85], 'patience': [1,2]
9: clf = KerasRegressor(build_fn=Lstm, verbose=0)
10: **Procedure** GridSearchCV(clf, Parameters, cv=5, scoring='MSE')
11: GridSearchCV.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
12: Best_parameters = KerasRegressor(batcĥ_size, neurôns_1, $neurôns\_2$, epôchs, patiêence)
13: **end procedure**
14: Lstm.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
15: $\hat{y}_{train} \longleftarrow Lstm.predict(\tilde{\mathcal{X}}_{train})$
16: $\hat{y}_{test} \longleftarrow Lstm.predict(\tilde{\mathcal{X}}_{test})$
17: **return** batch̃_size, neurôns_1, neurôns_2, epôchs, patiêence, $\hat{y}_{test}$, $\hat{y}_{train}$

8.3.4 Hybrid ML Models

In our study, we start by using individual models to predict MASI's close price time series. We train these models to minimize the difference between their predictions ($\hat{y}_t^{\text{models}}$)

and actual values (y_t).

$$\text{Res}_t = y_t - \hat{y}_t^{\text{models}}.$$

These predictions are not perfect, resulting in residuals. In the second part, we employ these residuals as targets for an XGBoost model to create hybrid ML models and enhance the accuracy of MASI close price predictions, such as identifying profitable sectors for investors. Mathematically, our hybrid model's final prediction at time $t+1$ is a sum of predictions from individual models, such as SVR, MLP, and LSTM, along with the prediction from the XGBoost model for the residual:

$$\hat{y}_{t+1}^{\text{Hybrid}} = \hat{y}_t^{\text{models}} + \hat{y}_t^{\text{Hybrid}}.$$

This combination, which includes the single model's predictions and the residuals predicted by the XGBoost model, results in the ultimate prediction of the hybrid model. Including these residuals significantly enhances the accuracy of our approach, capturing valuable information that the individual model may have overlooked. Algorithm 5, as proposed, illustrates the functioning of the hybrid model.

Algorithm 5 Hybrid Models Algorithm

Require: Residual time series $\tilde{\mathcal{X}}_{\text{train}}$, $\hat{\text{Res}}_{\text{train}}^{\text{models}}$
Ensure: Hybrid model parameters
1: Parameters = 'learning_rate': [0.0001,0.001,0.01], 'n_estimators': [780,790,795], 'max_depth': [1,2,3], 'γ': [0,0.00001,0.0001]
2: Hybrid_model = xgb.XGBRegressor()
3: **Procedure** GridSearchCV(Hybrid_model, Parameters, cv=5, scoring='MSE')
4: GridSearchCV.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$)
5: Best_parameters = KerasRegressor(n_estimators, learning_rate, max_depth, γ)
6: **end procedure**
7: Hybrid_model.fit($\tilde{\mathcal{X}}_{\text{train}}, \tilde{\mathcal{Y}}_{\text{train}}$, eval_set=[($\tilde{\mathcal{X}}_{\text{train}}$, $\hat{\text{Res}}_{\text{train}}^{\text{models}}$), ($\tilde{\mathcal{X}}_{\text{test}}$, $\hat{\text{Res}}_{\text{test}}^{\text{models}}$)], eval_metric='MAE', verbose=False)
8: $\hat{y}_{train} \longleftarrow \text{H}ybrid\_model.predict(\tilde{\mathcal{X}}_{train})$
9: $\hat{y}_{test} \longleftarrow \text{H}ybrid\_model.predict(\tilde{\mathcal{X}}_{test})$
10: **return** n_estimators, learning_rate, max_depth, γ, $\hat{y}_{test}$, $\hat{y}_{train}$

8.4 Results and Discussion

In this chapter, all experiments were conducted using Python (version 3.9.0) on two distinct computing servers: Ubuntu 18.04 running on an Intel(R) Xeon(R) Gold 5320 CPU and Windows 11 with an Intel Core i7-10875 processor running at 2.30 GHz, equipped with 32 GB of RAM and 32 cores. In the subsequent section, we provide the results and discussion of our analysis, focusing on the evaluation of six-time series forecasting models: SVR, MLP, LSTM, SVR-XGBoost, MLP-XGBoost, and LSTM-XGBoost. The efficacy of forecasting models is assessed through four accuracy metrics: MAE, MAPE, R^2, and RMSE. These metrics are represented in Equations (2.54), (2.58), (2.56), and (2.59). In this section, we highlight three main research outcomes: optimizing hyperparameters for single and hybrid models, identifying important feature sectors for each hybrid model with XGBoost, and comparing the performance and accuracy of our proposed models.

8.4.1 Analyzing Feature Importance in Hybrid Models

After fine-tuning our proposed models, we identified the top 29 feature contributions within three hybrid models: SVR-XGBoost, MLP-XGBoost, and LSTM-XGBoost, as illustrated in Figures 8.2, 8.3, 8.4. Among these, 10 sector features and 2 profitability indices, namely MASIR and MASRN, emerged as the most influential features. These important sectors in our hybrid models encompass distributors, environmental, banks, forestry, building and construction, pharmaceutical industry, hotels, transport, telecommunications, and beverages. This observation aligns with the findings presented in Figure 8.1. However, it is noteworthy that the LSTM-XGBoost model did not capture patterns associated with certain variables such as SPH, High, MASRN, BANK, TCOM, and SPI. It seems that the model needs to be improved.

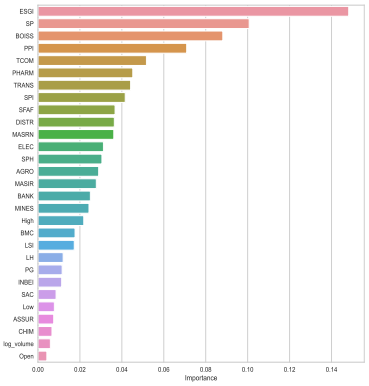


Figure 8.2 – SVR-XGBoost model

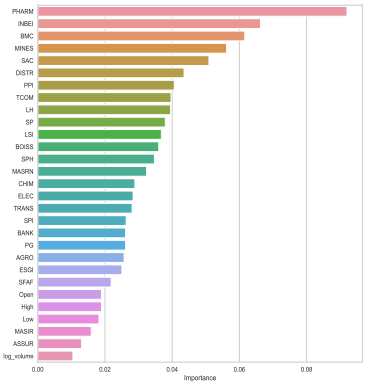


Figure 8.3 – MLP-XGBoost model

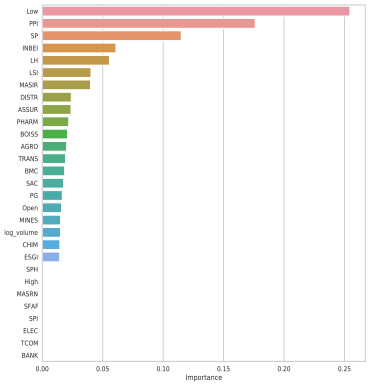


Figure 8.4 – LSTM-XGBoost model

8.4.2 Comparative Models Analysis

The comprehensive evaluation results for all individual and hybrid prediction models are presented in Figures 8.5, 8.6, 8.7. Our experimental results consistently reveal a pattern where metric evaluation values decrease with the extension of the prediction period (train, test), as demonstrated in Table 8.3. Moreover, Table 8.3 illustrates that the proposed models outperformed the benchmarks in various datasets, with the SVR-XGBoost model significantly exceeding the MLP-XGBoost model, closely following the LSTM model. Notably, Figures 8.5, 8.7 highlights an increase in prediction accuracy for LSTM, LSTM-XGBoost, and SVR models. In summary, the DL individual and hybrid models consistently provided accurate predictions across all models. However, in terms of performance, the SVR and SVR-XGBoost models outperformed others, demonstrating lower RMSE, MAE, and MAPE values. This performance pattern is further illustrated in Figure 8.8 when comparing boxplots within each train and test dataset. Boxplots provide valuable insights into typical information, error distributions, and the spread of error probabilities (Lin et al., 2022; Liwei et al., 2021; Xia et al., 2017). Figure 8.8 displays boxplots for RMSE, MAPE, and MAE errors corresponding to the forecasting of Moroccan stock market indices using different models. Specifically, the SVR-XGBoost and LSTM-XGBoost hybrid models consistently exhibit smaller and more concentrated RMSE, MAPE, and MAE errors across train and test sets.

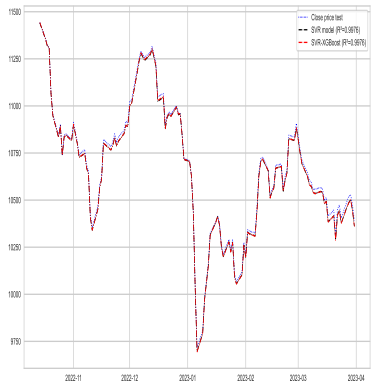


Figure 8.5 – SVR, Hybrid models

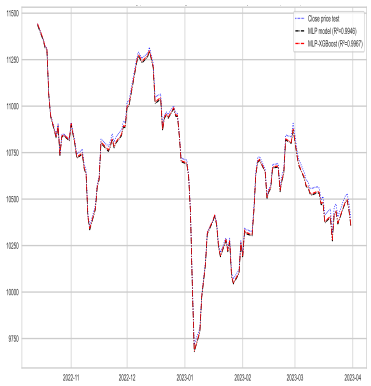


Figure 8.6 – MLP, Hybrid models

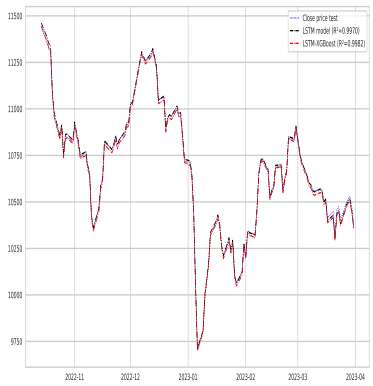


Figure 8.7 – LSTM, Hybrid models

Table 8.3 – Comparison of Performance Measures for All Models on the Train and Test Sets

Models	MAE		RMSE		MAPE (%)		Summary
	Train	Test	Train	Test	Train	Test	
SVR	√, +	√, +	√, +	√, +	√, +	√, +	√, +
MLP	−, +	−	−, +	−	−, +	−	−
LSTM	−, +	−, +	−, +	−, +	−, +	−	−, +
SVR-XGBoost	−, +	−	√, +	√	+	√	√
MLP-XGBoost	√, +	−	−, +	−	√, +	−	−
LSTM-XGBoost	−, +	√, +	−, +	+	−, +	+	+

Notes: '√', Perform well; '−', Underperform; '+', High Accuracy.

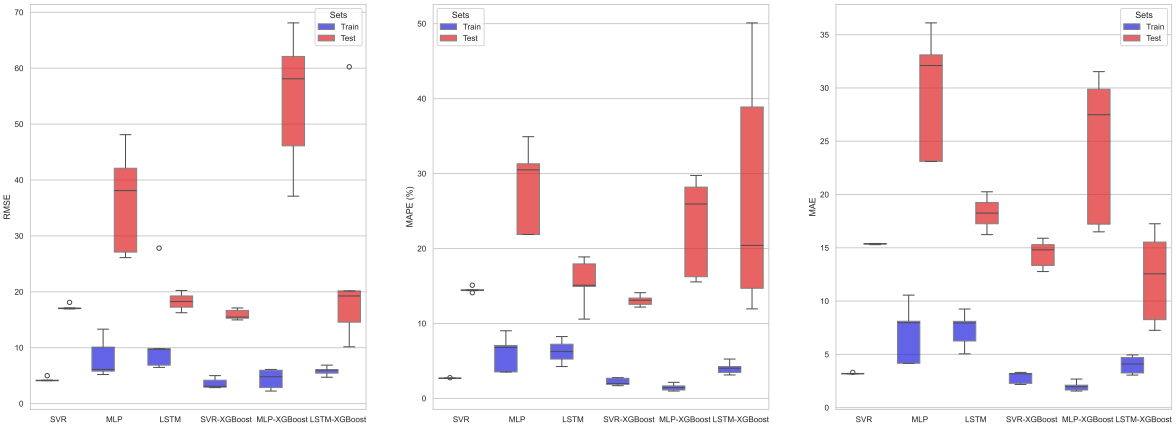


Figure 8.8 – Comparing Box Plots: RMSE, MAPE, and MAE for Single and Hybrid Models (Train and Test).

8.5 Conclusion

This research comprehensively compared prevalent methodologies and hybrid ML models for forecasting Moroccan stock market prices. The study incorporated significant feature sectors, such as Finance, Non-Energy Minerals, Communications, and others, to comprehensively assess their impact on stock market dynamics. Through our analysis and predictive work, we have derived a set of compelling findings that offer valuable implications. Furthermore, we extracted dependent variables based on their correlation

to ensure the significance of our models, thereby enhancing the reliability of our conclusions. The performance of the models was assessed using metrics including MAE, MAPE, R^2, and RMSE. Throughout our experiments, we observed that both the individual SVR model and the SVR-XGBoost hybrid model demonstrated the highest level of performance, achieving accuracy rates of 99.75% and 99.76%, respectively. The hybrid models proposed in this chapter demonstrate the ability to capture non-linear patterns within the stock market data. The results highlight the superior performance of the SVR-XGBoost and LSTM-XGBoost hybrid models, which can provide valuable insights for investors and institutions seeking information on promising sectors for investment, thereby optimizing returns and reducing risks.

Nonetheless, our model does have some limitations. It focuses exclusively on the impact of index sector price data on closing prices, overlooking the potential advantages of incorporating additional features like sentiment news. Additionally, the hyperparameter tuning process, particularly for the LSTM model, is time-intensive when utilizing the GS algorithm. In future research, we will address these limitations by integrating classical econometric methods with advanced ML models for enhanced financial time-series forecasting. Furthermore, we plan to compare the efficiency of hyperparameter tuning between the GS algorithm and a Bayesian approach to streamline computational time.

Chapter 9

General Conclusions and Outlook

In this thesis, we comprehensively analyzed various forecasting models for the Moroccan stock market, focusing on classical and ML approaches. Our research is encapsulated in six distinct articles, each contributing valuable insights into financial time series forecasting.

1. **Comparative Study of ARIMA, SVMs, and LSTM Models**:
   - This study evaluated the performance of ARIMA, SVR, and LSTM models in forecasting the MASI index.
   - Results showed that LSTM outperformed both ARIMA and SVR, highlighting the potential of deep learning models in capturing complex market patterns.
2. **Comparing ML Methods-SVR, XGBoost, LSTM, and MLP**:
   - We compared the effectiveness of SVR, XGBoost, LSTM, and MLP models in forecasting the MSI 20 index.
   - Optimized using Grid Search, SVR, and MLP models demonstrated superior accuracy, emphasizing the importance of hyperparameter tuning in ML models.
3. **Forecasting International Stock Market Trends**:
   - This study focused on forecasting key global stock indices using LSTM, XGBoost, and a hybrid LSTM-XGBoost model.
   - The hybrid model showed the best performance, offering precise forecasts crucial for global financial analysis.
4. **ML Models-Based Forecasting of the Moroccan Stock Market**:
   - We explored various ML models, including SVR-XGBoost and LSTM-XGBoost, for forecasting the MASI index.

— Results highlighted the superior performance of these hybrid models, demonstrating their efficacy in predicting stock prices across different sectors.

5. **New Predictive Modeling in Financial Markets**:
   — This research assessed ARIMA, Bilinear, and LSTM models for financial forecasting post-COVID.
   — Combining linear and non-linear elements, the hybrid models effectively captured market dynamics, offering significant advantages over traditional models.
6. **Data-driven Fortunes: Analysis of Multivariate Time Series Forecasting**:
   — We developed a robust hybrid model that integrates traditional ML techniques with deep learning methods for multivariate time series forecasting.
   — The hybrid models, particularly SVR-XGBoost and LSTM-XGBoost, outperformed benchmarks, providing high predictive accuracy and valuable insights for investors and policymakers.

Across these studies, we consistently found that hybrid models offer superior forecasting performance, particularly those integrating deep learning with traditional approaches. These models effectively capture the complexities of financial time series data, providing more accurate and reliable predictions. This research contributes to the expanding field of financial forecasting, offering practical insights for portfolio management, risk assessment, and economic policymaking.

Future work will focus on refining these hybrid models and exploring their application in algorithmic trading to develop robust strategies for individual investors and financial institutions in the Moroccan market. Additionally, we plan to investigate further other hybrid model forecasts, particularly those integrating realized volatility forecasting with GARCH multivariate LSTM models, to enhance predictive capabilities and decision-making processes in financial markets.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems [Software available from tensorflow.org]. *arXiv preprint arXiv:1603.04467*. https://www.tensorflow.org/

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics, 2014*.

Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer.

AGGRAWAL, T., & Dhawan, M. (2022). State-of-the-art vs prominent models: An empirical analysis of various neural networks on stock market prediction.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike* (pp. 199–213). Springer.

Alizadeh, M., & Ma, J. (2021). A comparative study of series hybrid approaches to model and predict the vehicle operating states. *Computers & Industrial Engineering, 162*, 107770.

Alkhatib, K., Khazaleh, H., Alkhazaleh, H. A., Alsoud, A. R., & Abualigah, L. (2022). A new stock price forecasting method using active deep learning approach. *Journal of Open Innovation: Technology, Market, and Complexity, 8*(2), 96.

Al-Musaylh, M. S., Deo, R. C., Adamowski, J. F., & Li, Y. (2018). Short-term electricity demand forecasting with MARS, SVR and ARIMA models using aggregated demand data in Queensland, Australia. *Advanced Engineering Informatics, 35*, 1–16.

Al-Nefaie, A. H., & Aldhyani, T. H. (2022). Predicting close price in emerging Saudi Stock Exchange: Time series models. *Electronics, 11*(21), 3443.

Alshammari, T. S., Ismail, M. T., Al-wadi, S., Saleh, M. H., & Jaber, J. J. (2020). Modeling and forecasting Saudi stock market volatility using wavelet methods. *The Journal of Asian Finance, Economics, and Business, 7*(11), 83–93.

Al-Thanoon, N. A., Algamal, Z. Y., & Qasim, O. S. (2022). Hyper parameters Optimization of Support Vector Regression based on a Chaotic Pigeon-Inspired Optimization Algorithm. *Mathematical Statistician and Engineering Applications, 71*(4), 4997–5008.

Amat Rodrigo, J., & Escobar Ortiz, J. (2023). *skforecast* (Version 0.9.0).

Anaconda software distribution. (2020). https://docs.anaconda.com/

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 106–112.

Awad, M., & Khanna, R. (2015). *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*. Springer nature.

Ayala, J., Garcıa-Torres, M., Noguera, J. L. V., Gómez-Vela, F., & Divina, F. (2021). Technical analysis strategy optimization using a machine learning approach in stock market indices. *Knowledge-Based Systems*, *225*, 107119.

Baek, Y., & Kim, H. Y. (2018). ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, *113*, 457–480.

Ballesteros, M., Dyer, C., & Smith, N. A. (2015). Improved transition-based parsing by modeling characters instead of words with LSTMs. *arXiv preprint arXiv:1508.00657*.

Bathla, G. (2020). Stock Price Prediction using LSTM and SVR, 211–214.

Belcaid, K., & El Ghini, A. (2019). US, European, Chinese economic policy uncertainty and Moroccan stock market volatility. *The Journal of Economic Asymmetries*, *20*, e00128.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Berlinet, A., & Francq, C. (1990). Stationnarité et identification d'un processes bilinéaire strictement superdiagonal. *Statistique et Analyse des Données*, *15*(1), 1–24.

Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society*, *48*(3), 334–334.

Bezerra, P. C. S., & Albuquerque, P. H. M. (2017). Volatility forecasting via SVR–GARCH with mixture of Gaussian kernels. *Computational Management Science*, *14*, 179–196.

Bhamidipati, V. S. P., & Saisanthiya, D. (2023). Stock price prediction using random forest classifier and backtesting. *2023 International Conference on Computer Communication and Informatics (ICCCI)*, 1–8.

Bhardwaj, G., & Swanson, N. R. (2006). An empirical investigation of the usefulness of ARFIMA models for predicting macroeconomic and financial time series. *Journal of econometrics*, *131*(1-2), 539–578.

Bibi, A., & Ghezal, A. (2018). Markov-switching BILINEAR-GARCH models: Structure and estimation. *Communications in Statistics-Theory and Methods*, *47*(2), 307–323.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, *31*(3), 307–327.

Bouzaâchane, K. (2006). *Modeles de séries chronologiques bilinéaires: Estimations, algorithmes et applications* [Doctoral dissertation, Ecole Mohammadia d'Ingénieurs].

Box, G. E., Jenkins, G. M., & Reinsel, G. (1970). Time series analysis: Forecasting and control Holden-day San Francisco. *BoxTime Series Analysis: Forecasting and Control Holden Day1970*.

Box, G. E., & Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, *65*(332), 1509–1526.

Box, G., & Jenkins, G. (1970). *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Bozdogan, H. (1987). Model selection and akaike's information criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, *52*(3), 345–370.

Brockwell, P. J., & Davis, R. A. (2002). *Introduction to time series and forecasting*. Springer.

Campbell, J. Y., Lo, A. W., MacKinlay, A. C., & Whitelaw, R. F. (1998). The econometrics of financial markets. *Macroeconomic Dynamics*, *2*(4), 559–562.

Cao, J., Li, Z., & Li, J. (2019). Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical Mechanics and its Applications*, *519*, 127–139.

Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, *14*(6), 1506–1518.

Casado-Vara, R., Martin del Rey, A., Pérez-Palau, D., de-la-Fuente-Valentín, L., & Corchado, J. M. (2021). Web Traffic Time Series Forecasting Using LSTM Neural Networks with Distributed Asynchronous Training. *Mathematics*, *9*(4).

Catania, L., & Grassi, S. (2022). Forecasting cryptocurrency volatility. *International Journal of Forecasting*, *38*(3), 878–894.

Challa, M. L., Malepati, V., & Kolusu, S. N. R. (2020). S&P BSE Sensex and S&P BSE IT return forecasting using ARIMA. *Financial Innovation*, *6*(1), 1–19.

Chatfield, C. (1996). Model uncertainty and forecast accuracy. *Journal of Forecasting*, *15*(7), 495–508.

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. *2015 IEEE international conference on big data (big data)*, 2823–2824.

Chen, M.-Y. (2013). Financial time series and their characteristics. *National Chung Hsing University, Available at http://web.nchu.edu.tw/˜finmyc/tim_serp. pdf*.

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

Chhajer, P., Shah, M., & Kshirsagar, A. (2022). The applications of artificial neural networks, support vector machines, and long–short term memory for stock market prediction. *Decision Analytics Journal*, *2*, 100015.

Chniti, G., Bakir, H., & Zaher, H. (2017). E-commerce Time Series Forecasting using LSTM Neural Network and Support Vector Regression. *Proceedings of the international conference on big data and Internet of Thing*, 80–84.

Chollet, F. K. (2015). Available online: Https://keras.io (accessed on 14 august 2019).© 2019 by the authors. *Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons. org/licenses/by/4.0/)*.

Chollet, F., et al. (2015). Keras.

Chua, L. O., & Yang, L. (1988). Cellular neural networks: Applications. *IEEE Transactions on circuits and systems*, *35*(10), 1273–1290.

Chung, H., & Shin, K.-s. (2020). Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Computing and Applications*, *32*, 7897–7914.

Cochrane, J. H. (1997). *Time Series for Macroeconomics and Finance* (G. S. of Business, Ed.). spring.

Cocianu, C.-L., & Grigoryan, H. (2016). MACHINE LEARNING TECHNIQUES FOR STOCK MARKET PREDICTION. A CASE STUDY OF OMV PETROM. *Economic Computation & Economic Cybernetics Studies & Research*, *50*(3).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

Devan, P., & Khare, N. (2020). An efficient XGBoost–DNN-based classification model for network intrusion detection system. *Neural Computing and Applications*, *32*, 12499–12514.

Dezhkam, A., & Manzuri, M. T. (2023). Forecasting stock market for an efficient portfolio by combining XGBoost and Hilbert Huang transform. *Engineering Applications of Artificial Intelligence*, *118*, 105626.

Di Bucchianico, A. (2008). Coefficient of determination (R^2). *Encyclopedia of statistics in quality and reliability*.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, *74*(366a), 427–431.

Ding, G., & Qin, L. (2020). Study on the prediction of stock price based on the associated network model of LSTM. *International Journal of Machine Learning and Cybernetics*, *11*.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1996). Support vector regression machines. *Advances in neural information processing systems*, *9*.

Dunford, N., & Schwartz, J. T. (1963). Linear operators ii: Spectral theory.

Durbin, J. (1960). The fitting of time-series models. *Revue de l'Institut International de Statistique*, 233–244.

El Ghini, A., & Saidi, Y. (2017). Return and volatility spillovers in the Moroccan stock market during the financial crisis. *Empirical Economics*, *52*(4), 1481–1504.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Elmsili, B., & Outtaj, B. (2021). Predicting Stock Market Movements Using Machine Learning Techniques. *International Journal of Accounting, Finance, Auditing, Management and Economics*, *2*(3), 390–405.

Elsaraiti, M., & Merabet, A. (2021). A comparative analysis of the ARIMA and LSTM predictive models and their effectiveness for predicting wind speed. *Energies*, *14*(20), 6782.

Enders, W. (2004). Applied Econometric Time Series. *Inc., New York*.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the Econometric Society*, 987–1007.

Engle, R. F., & Bollerslev, T. (1986). Modelling the persistence of conditional variances. *Econometric reviews*, *5*(1), 1–50.

Ettayyebi, H. (2017). *Time series forecasting with Artificial neural networks: Global solar irradiance* [Master's thesis, MOHAMMED V UNIVERSITY IN RABAT].

Ettayyebi, H., & El Himdi, K. (2018a). Artificial neural network for forecasting one day ahead of global solar irradiance. *Smart Application and Data Analysis for Smart Cities (SADASC'18)*.

Ettayyebi, H., & El Himdi, K. (2018b). Artificial neural networks for forecasting the 24 hours ahead of global solar irradiance. *AIP Conference Proceedings*, *2056*(1), 020010.

Falloul, M. E. M., & Mansouri, A. (2014). Indice MASI: Une tentative de modélisation par les modèles ARIMA et GARCH [MASI index: An attempt of modeling using ARIMA and GARCH models]. *International Journal of Innovation and Applied Studies*, *7*(4), 1560.

Fama, E. F. (1970). Efficient capital markets. *Journal of finance*, *25*(2), 383–417.

Fama, E. F. (1991). Efficient capital markets: Ii. *The journal of finance*, *46*(5), 1575–1617.

Faraway, J., & Chatfield, C. (1998). Time series forecasting with neural networks: A comparative study using the air line data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *47*(2), 231–250.

Fazeli, A., & Houghten, S. (2019). Deep learning for the prediction of stock market trends. *2019 IEEE International Conference on Big Data (Big Data)*, 5513–5521.

Flaherty, J., & Lombardo, R. (2000). Modelling private new housing starts in Australia. *A paper presented in the Pacific-Rim Real Estate Society conference*, 24–27.

Foroni, C., Marcellino, M., & Stevanovic, D. (2020). Forecasting the Covid-19 recession and recovery: Lessons from the financial crisis. *International Journal of Forecasting*.

Galbraith, J. W., Zinde-Walsh, V., et al. (2001). *Autoregression-based estimators for ARFIMA models* (tech. rep.). CIRANO.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. "O'Reilly Media, Inc."

Gers, F. A., & Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE transactions on neural networks*, *12*(6), 1333–1340.

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. *Journal of machine learning research*, *3*(Aug), 115–143.

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental concepts of convolutional neural network. *Recent trends and advances in artificial intelligence and Internet of Things*, 519–567.

Goverdhan, G., Khare, S., & Manoov, R. (2022). Time Series Prediction: Comparative Study of ML Models in the Stock Market.

Granger, C. W. J., & Andersen, A. P. (1978). An introduction to bilinear time series models.

Granger, C. W., & Pesaran, M. H. (2000). Economic and statistical measures of forecast accuracy. *Journal of Forecasting*, *19*(7), 537–560.

Guegan, D. (1981). Etude d'un modele non lineaire, le modele superdiagonal d'ordre 1.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert systems with Applications*, *38*(8), 10389–10397.

Gutmann, S., Maget, C., Spangler, M., & Bogenberger, K. (2021). Truck parking occupancy prediction: XGBoost-LSTM model fusion. *Frontiers in Future Transportation*, *2*, 693708.

Hamzaçebi, C. (2008). Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences*, *178*(23), 4550–4559.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a GARCH(1,1)? *Journal of applied econometrics*, *20*(7), 873–889.

Haykin, S. S., et al. (2009). Neural networks and learning machines.

Hebb, D. (1968). The organization of behavior.

Hecht-Nielsen, R. (1987). Counterpropagation networks. *Applied optics*, *26*(23), 4979–4984.

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. (2018). NSE stock market prediction using deep-learning models. *Procedia computer science*, *132*, 1351–1362.

Ho, K. L., Hsu, Y. .-., & Yang, C. .-. (1992). Short term load forecasting using a multilayer neural network with an adaptive learning algorithm. *IEEE Transactions on Power Systems*, *7*(1), 141–149.

Hochreiter, S., & Schmidhuber, J. (1996). LSTM can solve hard long time lag problems. *Advances in neural information processing systems*, *9*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hollis, T., Viscardi, A., & Yi, S. E. (2018). A comparison of LSTMs and attention mechanisms for forecasting financial time series. *arXiv preprint arXiv:1812.07699*.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

Hossain, A., & Nasser, M. (2008). Comparison of GARCH and neural network methods in financial time series prediction. *2008 11th International Conference on Computer and Information Technology*, 729–734.

Hossain, A., & Nasser, M. (2011). Comparison of the finite mixture of ARMA-GARCH, back propagation neural networks and support-vector machines in forecasting financial returns. *Journal of Applied Statistics*, *38*(3), 533–551.

Hussein, A. S., Hamed, I. M., & Tolba, M. F. (2015). An efficient system for stock market prediction. *Intelligent Systems' 2014: Proceedings of the 7th IEEE International Conference Intelligent Systems IS'2014, September 24-26, 2014, Warsaw, Poland, Volume 2: Tools, Architectures, Systems, Applications*, 871–882.

Jain, A., Srinivas, E., & Rauta, R. (2009). Short term load forecasting using fuzzy adaptive inference and similarity. *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 1743–1748.

Ji, Y., Liew, A. W.-C., & Yang, L. (2021). A Novel Improved Particle Swarm Optimization With Long-Short Term Memory Hybrid Model for Stock Indices Forecast. *IEEE Access*, *9*, 23660–23671.

John, M. M., Ganiny, S., & Hanief, M. (2022). Predictive modeling of pump noise using multi-linear regression and random-forest models-via optimal data splitting. *International Journal of Simulation and Process Modelling*, *18*(4), 267–283.

Jouilil, Y., et al. (2023). Comparing the Accuracy of Classical and Machine Learning Methods in Time Series Forecasting: A Case Study of USA Inflation. *Statistics, Optimization & Information Computing*, *11*(4), 1041–1050.

Kalchbrenner, N., Danihelka, I., & Graves, A. (2015). Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

Kao, Y.-S., Nawata, K., & Huang, C.-Y. (2020). Predicting Primary Energy Consumption Using Hybrid ARIMA and GA-SVR Based on EEMD Decomposition. *Mathematics*, *8*(10).

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert systems with Applications*, *38*(5), 5311–5319.

Karush, W. (1939). *Minima of Functions of Several Variables with Inequalities as Side Conditions* [Master's thesis, Department of Mathematics, University of Chicago].

Khashei, M., & Hajirahimi, Z. (2019). A comparative study of series ARIMA/MLP hybrid models for stock price forecasting. *Communications in Statistics-Simulation and Computation*, *48*(9), 2625–2640.

Khedmati, M., Seifi, F., & Azizi, M. (2020). Time Series Forecasting of Bitcoin Price Based on Autoregressive Integrated Moving Average and Machine Learning Approaches. *International Journal of Engineering*, *33*(7), 1293–1303.

Kihoro, J., Otieno, R., & Wafula, C. (2004). Seasonal time series forecasting: A comparative study of ARIMA and ANN models.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1-2), 307–319.

Kim, T., & Kim, H. Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PloS one*, *14*(2), e0212320.

Koukaras, P., Nousi, C., & Tjortjis, C. (2022). Stock market prediction using microblogging sentiment analysis and machine learning. *Telecom*, *3*(2), 358–378.

Kuhn, H., & Tucker, A. (1951). Nonlinear programming, paper presented at Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability.

Kumar, M., & M., T. (2014). Forecasting stock index returns using ARIMA-SVM, ARIMA-ANN, and ARIMA-random forest hybrid models. *International Journal of Banking, Accounting and Finance*, *5*(3), 284–308.

Kumar, V. (2016). TIME SERIES MODELING AND FORECASTING USING STOCHASTIC MODELS: A REVIEW. *International Journal of Engineering Sciences and Research Technology*, 585–589.

Kumari, P., & Toshniwal, D. (2021). Extreme gradient boosting and deep neural network based ensemble learning approach to forecast hourly solar irradiance. *Journal of Cleaner Production*, *279*, 123285.

K.W. Hipel, A. M. (1994). *Time Series Modelling of Water Resources and Environmental Systems*. Amsterdam, Elsevier.

Lai, L. L. (1998). *Intelligent system applications in power engineering: Evolutionary programming and neural networks*. John Wiley & Sons, Inc.

Le, Q. V., Jaitly, N., & Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lee, J. (2018). Univariate time series modeling and forecasting (Box-Jenkins method), Econ 413, Lecture 4. *Department of Economics, University of Illinois*.

Lee, R. (1959). Generalization of learning in a machine. *Preprints of papers presented at the 14th national meeting of the Association for Computing Machinery*, 1–4.

Leung, T., & Zhao, T. (2021). Financial time series analysis and forecasting with Hilbert–Huang transform feature generation and machine learning. *Applied Stochastic Models in Business and Industry*, *37*(6), 993–1016.

Li, R., Han, T., & Song, X. (2022). Stock price index forecasting using a multiscale modelling strategy based on frequency components analysis and intelligent optimization. *Applied Soft Computing*, *124*, 109089.

Li, S., & Zhang, X. (2020). Research on orthopedic auxiliary classification and prediction model based on XGBoost algorithm. *Neural Computing and Applications*, *32*, 1971–1979.

Li, Z., & Tam, V. (2018). A machine learning view on momentum and reversal trading. *Algorithms*, *11*(11), 170.

Lin, Y.-L., Lai, C.-J., & Pai, P.-F. (2022). Using deep learning techniques in forecasting stock markets by hybrid data with multilingual sentiment analysis. *Electronics*, *11*(21), 3513.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Little, W. A. (1974). The existence of persistent states in the brain. *Mathematical biosciences*, *19*(1-2), 101–120.

Liu, J., & Brockwell, P. J. (1988). On the general bilinear time series model. *Journal of Applied probability*, *25*(3), 553–564.

Liwei, T., Li, F., Yu, S., Yuankai, G., et al. (2021). Forecast of LSTM-XGBoost in stock price based on bayesian optimization. *Intell. Autom. Soft Comput*, *29*(3), 855–868.

Ljung, G. M., & Box, G. E. (1978). On a measure of lack of fit in time series models. *Biometrika*, *65*(2), 297–303.

Lmakri, A., Akharif, A., & Mellouk, A. (2020). Optimal Detection of Bilinear Dependence in Short Panels of Regression Data. *Revista Colombiana de Estadıstica*, *43*(2), 143–171.

Lmakri, A., Akharif, A., & Mellouk, A. (2024). Optimal Rank-based Procedures for Testing Randomness against Panel Bilinear Dependence.

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. *Complexity*, *2020*, 1–10.

Maravall, A. (1983). An application of nonlinear time series forecasting. *Journal of Business & Economic Statistics*, *1*(1), 66–74.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological review*, *88*(5), 375.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

McNally, S., Roche, J., & Caton, S. (2018). Predicting the price of bitcoin using machine learning. *2018 26th euromicro international conference on parallel, distributed and network-based processing (PDP)*, 339–343.

Mellit, A., & Kalogirou, S. A. (2008). Artificial intelligence techniques for photovoltaic applications: A review. *Progress in energy and combustion science*, *34*(5), 574–632.

Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., & Ranzato, M. (2014). Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*.

Mikolov, T., & Zweig, G. (2012). Context dependent recurrent neural network language model. *2012 IEEE Spoken Language Technology Workshop (SLT)*, 234–239.

Mintarya, L. N., Halim, J. N., Angie, C., Achmad, S., & Kurniawan, A. (2023). Machine learning approaches in stock market prediction: A systematic literature review. *Procedia Computer Science*, *216*, 96–102.

Misra, M., Yadav, A. P., & Kaur, H. (2018). Stock market prediction using machine learning algorithms: A classification study. *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, 2475–2478.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine learning*, *1*, 47–80.

Mustapa, F. H., & Ismail, M. T. (2019). Modelling and forecasting S&P 500 stock prices using hybrid ARIMA-GARCH model. *Journal of Physics: Conference Series*, *1366*(1), 012130.

Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S., & Mosavi, A. (2020). Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. *Ieee Access*, *8*, 150199–150212.

Nasirtafreshi, I. (2022). Forecasting cryptocurrency prices using Recurrent Neural Network and Long Short-term Memory. *Data & Knowledge Engineering*, *139*, 102009.

Nejad, F. S., & Ebadzadeh, M. M. (2024). Stock market forecasting using DRAGAN and feature matching. *Expert Systems with Applications*, *244*, 122952.

Okasha, M. K. (2014). Using support vector machines in financial time series forecasting. *International Journal of Statistics and Applications*, *4*(1), 28–39.

Oukhouya, H., & El Himdi, K. (2023a). A comparative study of ARIMA, SVMs, and LSTM models in forecasting the Moroccan stock market. *International Journal of Simulation and Process Modelling*, *20*(2), 125–143.

Oukhouya, H., & El Himdi, K. (2023b). Comparing Machine Learning Methods—SVR, XGBoost, LSTM, and MLP— for Forecasting the Moroccan Stock Market. *Computer Sciences & Mathematics Forum*, *7*(1), 39.

Oukhouya, H., Kadiri, H., El Himdi, K., & Guerbaz, R. (2024). Forecasting International Stock Market Trends: XGBoost, LSTM, LSTM-XGBoost, and Backtesting XGBoost Models. *Statistics, Optimization & Information Computing*, *12*(1), 200–209.

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied soft computing*, *93*, 106384.

Pagano, M. (1972). An algorithm for fitting autoregressive schemes. *Journal of the Royal Statistical Society Series C: Applied Statistics*, *21*(3), 274–281.

Park, H. (1999). Forecasting three-month treasury bills using ARIMA and GARCH models. *Econ930, Department of Economics, Kansas State University, Tech. Rep.*

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *International conference on machine learning*, 1310–1318.

Patterson, D. W. (1998). *Artificial neural networks: Theory and applications*. London : Prentice-Hall, 1996.

Perrelli, R. (2001). Introduction to ARCH & GARCH models. *University of Illinois Optional TA Handout*, 1–7.

Pham, T. D., & Tran, L. (1980). Quelques résultats sur les modeles bilinéaires de séries chronologiques.

Pham, T. D., & Tran, L. T. (1981). On the first-order bilinear time series model. *Journal of Applied Probability*, *18*(3), 617–627.

Ping-Feng Pai, C.-S. L. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, *33*, 497–505.

Plakandaras, V., Papadimitriou, T., & Gogas, P. (2015). Forecasting daily and monthly exchange rates with machine learning techniques. *Journal of Forecasting*, *34*(7), 560–573.

Principe, J. C., Euliano, N. R., & Lefebvre, W. C. (2000). *Neural and adaptive systems: Fundamentals through simulations* (Vol. 672). Wiley New York.

Qin, H., Yu, M., Lai, Y., Liu, Z., & Liu, J. (2019). Cloud resource orchestration optimisation based on ARIMA. *International Journal of Simulation and Process Modelling*, *14*(5), 420–430.

Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PloS one*, *15*(1), e0227222.

Qu, H., & Zhang, Y. (2016). A new kernel of support vector regression for forecasting high-frequency stock returns. *Mathematical Problems in Engineering*, *2016*.

R Core Team. (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Raju, K. S., Murty, M. R., Rao, M. V., & Satapathy, S. C. (2018). Support Vector Machine with k-fold cross validation model for software fault prediction. *Int. J. of Pure and Applied Maths*, 321–334.

Ranković, V., Grujović, N., Divac, D., & Milivojević, N. (2014). Development of support vector regression identification model for prediction of dam structural behaviour. *Structural Safety*, *48*, 33–39.

Rather, A. M., Agarwal, A., & Sastry, V. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, *42*(6), 3234–3241.

Raza, M. Q., & Khosravi, A. (2015). A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, *50*, 1352–1372.

Rhanoui, M., Yousfi, S., Mikram, M., & Merizak, H. (2019). Forecasting financial budget time series: ARIMA random walk vs LSTM neural network. *IAES International Journal of Artificial Intelligence*, *8*(4), 317.

Rochester, N., Holland, J., Haibt, L., & Duda, W. (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on information Theory*, *2*(3), 80–93.

Rosenblatt, F. (1958). Principle of Neurodynamics Spartan Books.

Rubio, L., & Alba, K. (2022). Forecasting selected colombian shares using a hybrid ARIMA-SVR model. *Mathematics*, *10*(13), 2181.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition.

Rundo, F., Trenta, F., di Stallo, A. L., & Battiato, S. (2019). Machine learning for quantitative finance applications: A survey. *Applied Sciences*, *9*(24), 5574.

Sabry, I., Mourad, A.-H. I., Idrisi, A. H., & ElWakil, M. (2022). ARIMA-based time-series analysis for forecasting of COVID-19 cases in Egypt. *International Journal of Simulation and Process Modelling*, *19*(1-2), 86–96.

Sai Krishna, L., & McCrae, J. P. (2019). A Comparative Study of SVM and LSTM Deep Learning Algorithms for Stock Market Prediction. *AICS*, 446–457.

Sakia, R. M. (1992). The Box-Cox transformation technique: A review. *Journal of the Royal Statistical Society Series D: The Statistician*, *41*(2), 169–178.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

Samarawickr- ama, A., & Fernando, T. (2017). A recurrent neural network approach in predicting daily stock prices an application to the sri lankan stock market. *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, 1–6.

Samuel, A. L. (1959a). Machine learning. *The Technology Review*, *62*(1), 42–45.

Samuel, A. L. (1959b). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210–229.

Samui, P., & Kothari, D. (2011). Utilization of a least square support vector machine (lssvm) for slope stability analysis. *Scientia Iranica*, *18*(1), 53–58.

Schmidhuber, J., Hochreiter, S., et al. (1997). Long short-term memory. *Neural Comput*, *9*(8), 1735–1780.

Semmelmann, L., Henni, S., & Weinhardt, C. (2022). Load forecasting for energy communities: A novel LSTM-XGBoost hybrid model based on smart meter data. *Energy Informatics*, *5*(1), 1–21.

Sen, A., & Dutta Choudhury, K. (2024). A case study of Gulf Securities Market in the last 20 years: A Long Short-Term Memory approach. *Statistica Neerlandica*, *78*(1), 136–166.

Sesay, S., & Rao, T. S. (1991). DIFFERENCE EQUATIONS FOR HIGHER-ORDER MOMENTS AND CUMULANTS FOR THE BILINEAR TIME SERIES MODEL BL(p, 0, p, 1). *Journal of Time Series Analysis*, *12*(2), 159–177.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181.

Shah, D., Isah, H., & Zulkernine, F. (2019). Stock Market Analysis: A Review and Taxonomy of Prediction Techniques. *International Journal of Financial Studies*, *7*(2).

Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of big Data*, *7*(1), 1–33.

Sheta, A. F., Ahmed, S. E. M., & Faris, H. (2015). A comparison between regression, artificial neural networks, and support vector machines for predicting stock market index. *Soft Computing*, *7*(8), 2.

Shi, Z., Hu, Y., Mo, G., & Wu, J. (2022). Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction. *arXiv preprint arXiv:2204.02623*.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018). A comparison of ARIMA and LSTM in forecasting time series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394–1401.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, *14*(3), 199–222.

Song, H., & Choi, H. (2023). Forecasting stock market indices using the recurrent neural network based hybrid models: CNN-LSTM, GRU-CNN, and ensemble models. *Applied Sciences*, *13*(7), 4644.

Sornette, D., & Pisarenko, V. (2008). Properties of a simple bilinear stochastic model: Estimation and predictability. *Physica D: Nonlinear Phenomena*, *237*(4), 429–445.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding LSTM–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.

Subba Rao, T., & Da Silva, M. E. (1992). Identification of bilinear time series models BL(p, 0, p, 1). *Statistica Sinica*, *2*(2), 465–478.

Subba Rao, T. (1981). On the theory of bilinear time series models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *43*(2), 244–255.

Subba Rao, T., & Gabr, M. (1987). An Introduction to Bispectral Analysis and Bilinear Time Series Models. *Journal of the Royal Statistical Society. Series A (General)*, *150*.

Sulistyowati, R., Kuswanto, H., Astuti, E. T., et al. (2018). Hybrid forecasting model to predict air passenger and cargo in Indonesia. *2018 international conference on information and communications technology (ICOIACT)*, 442–447.

Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, *9*(3), 293–300.

T. Raicharoen, P. S., C. Lursinsap. (2003). Application of critical support vector machine to time series prediction. *Circuits and Systems*, *5*, V-741-V–744.

Tamatta, R. K. (2018). *Time series forecasting of hospital Inpatients and Day case waiting list using ARIMA, TBATS and Neural Network Models* [Doctoral dissertation, Dublin, National College of Ireland].

Tan, L., & Wang, L. (2016). The LASSO method for bilinear time series models. *Communications in Statistics-Simulation and Computation*, *45*(3), 1072–1082.

Tang, H., Chiu, K.-C., & Xu, L. (2003). Finite mixture of ARMA-GARCH model for stock price prediction. *Proceedings of the Third International Workshop on Computational Intelligence in Economics and Finance (CIEF'2003), North Carolina, USA*, 1112–1119.

Tank, A., Covert, I., Foti, N., Shojaie, A., & Fox, E. B. (2021). Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(8), 4267–4279.

Theoharidis, A. F., Guillén, D. A., & Lopes, H. (2023). Deep learning models for inflation forecasting. *Applied Stochastic Models in Business and Industry*, *39*(3), 447–470.

Thormann, M.-L., Farchmin, J., Weisser, C., Kruse, R.-M., Säfken, B., & Silbersdorff, A. (2021). Stock price predictions with lstm neural networks and twitter sentiment. *Statistics, Optimization & Information Computing*, *9*(2), 268–287.

Tong, H. (1983). Threshold Models in Non-Linear Time Series Analysis. *Springer-Verlag , New York*.

Tsay, R. (2010). Analysis of Financial Time Series. *Michal Greguš, Doc., RNDr., PhD. Odbojárov*, *10*(820), 05.

Turing, A. M. (2009). *Computing machinery and intelligence*. Springer.

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A Review on the Long Short-Term Memory Model. *Artificial Intelligence Review*, *53*(8), 5929–5955.

Vapnik, V., & Chervonenkis, A. Y. (1964). A class of algorithms for pattern recognition learning. *Avtomat. i Telemekh*, *25*(6), 937–945.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Inc., New York, John Wiley & Sons.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, *10*(5), 988–999.

Vapnik, V., & Lerner, A. Y. (1963). Recognition of patterns with help of generalized portraits. *Avtomat. i Telemekh*, *24*(6), 774–780.

Vuong, P. H., Dat, T. T., Mai, T. K., Uyen, P. H., et al. (2022). Stock-price forecasting based on XGBoost and LSTM. *Computer Systems Science & Engineering*, *40*(1).

Wang, Y., & Guo, Y. (2020). Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost. *China Communications*, *17*(3), 205–221.

Wang, Y., & Ni, X. S. (2019). A XGBoost risk model via feature selection and Bayesian hyperparameter optimization. *arXiv preprint arXiv:1901.08433*.

Wang, Y., Guo, L., Zhang, Y., & Ma, X. (2023). Research on CSI 300 Stock Index Price Prediction Based On EMD-XGBoost. *Frontiers in Computing and Intelligent Systems*, *3*(1), 72–77.

Wei, W. W.-S. (2006). *Time Series Analysis: Univariate and Multivariate Methods* (R. H. Deirdre Lynch Sara Oliver, Ed.; Second, Vol. 10). Greg Tobin.

Weiss, A. A. (1986). Arch and bilinear time series models: Comparison and combination. *Journal of Business & Economic Statistics*, *4*(1), 59–70.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, *1*(4), 339–356.

West, K. D. (2006). Forecast evaluation. *Handbook of economic forecasting*, *1*, 99–134.

Widrow, B., & Hoff, M. E. (1960). *Adaptive switching circuits* (tech. rep.). Stanford Univ Ca Stanford Electronics Labs.

Widrow, B., & Winter, R. (1988). Neural nets for adaptive filtering and adaptive pattern recognition. *Computer*, *21*(3), 25–39.

Wijaya, Y. B., Kom, S., & Napitupulu, T. A. (2010). Stock price prediction: Comparison of ARIMA and artificial neural network methods-An Indonesia Stock's Case. *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*, 176–179.

Wu, J. M.-T., Li, Z., Herencsar, N., Vo, B., & Lin, J. C.-W. (2023). A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimedia Systems*, *29*(3), 1751–1770.

Wu, X., & Lu, Q. (2021). Financial asset yield series forecasting based on risk-neutral fuzzy bilinear regression and probabilistic neural network. *Journal of Intelligent & Fuzzy Systems*, *40*(6), 11829–11844.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv: 1609.08144*.

Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert systems with applications*, *78*, 225–241.

Xie, Z., Xu, Y., & Hu, Q. (2018). Uncertain data classification with additive kernel support vector machine. *Data & Knowledge Engineering*, *117*, 87–97.

Xu, S., Chan, H. K., & Zhang, T. (2019). Forecasting the demand of the aviation industry using hybrid time series SARIMA-SVR approach. *Transportation Research Part E: Logistics and Transportation Review*, *122*, 169–180.

Xu, S., Zou, S., Huang, J., Yang, W., & Zeng, F. (2022). Comparison of different approaches of machine learning methods with conventional approaches on container throughput forecasting. *Applied Sciences*, *12*(19), 9730.

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295–316.

Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. (2015). Depth-gated LSTM. *arXiv preprint arXiv:1508.03790*.

Yaslan, Y., & Bican, B. (2017). Empirical mode decomposition based denoising method with support vector regression for time series prediction: A case study for electricity load forecasting. *Measurement*, *103*, 52–61.

Yu, S., Tian, L., Liu, Y., & Guo, Y. (2021). LSTM-XGBoost application of the model to the prediction of stock price. *Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19–23, 2021, Proceedings, Part I 7*, 86–98.

Yuan, G., Zhang, T., Zhang, W., & Li, H. (2021). Analysis of stock price based on the XGBoost algorithm with EMA-19 and SMA-15 features. *2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE)*, 1–4.

Yun, K. K., Yoon, S. W., & Won, D. (2021). Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process. *Expert Systems with Applications*, *186*, 115716.

Zhang, G. P. (2007). A neural network ensemble method with jittered training data for time series forecasting. *Information Sciences*, *177*(23), 5329–5346.

Zhang, G. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175.

Zhang, G. (2007). A neural network ensemble method with jittered training data for time-series forecasting. *Information Sciences*, *177*, 5329–5346.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, *14*(1), 35–62.

Zheng, J., Wang, Y., Li, S., & Chen, H. (2021). The stock index prediction based on SVR model with bat optimization algorithm. *Algorithms*, *14*(10), 299.

Zhou, K., Wang, W., Huang, L., & Liu, B. (2021). Comparative study on the time series forecasting of web traffic based on statistical model and Generative Adversarial model. *Knowledge-Based Systems*, *213*, 106467.

List of publications and submission

List of publications

1. H. Oukhouya; K. El Himdi. A Comparative Study of ARIMA, SVMs, and LSTM Models in Forecasting the Moroccan Stock Market. International Journal of Simulation and Process Modelling, **2023**, 20(2), 125-143, https://doi.org/10.1504/IJSPM.2023.136481

2. H. Oukhouya; K. El Himdi. Comparing Machine Learning Methods-SVR, XGBoost, LSTM, and MLP-for Forecasting the Moroccan Stock Market. Computer Sciences & Mathematics Forum, **2023**, 7(1), MDPI, https://doi.org/10.3390/IOCMA2023-14409

3. H. Oukhouya; H. Kadiri; K. El Himdi; R. Guerbaz. Forecasting International Stock Market Trends: XGBoost, LSTM, LSTM-XGBoost, and Backtesting XGBoost Models. Statistics, Optimization & Information Computing, **2024**, 12(1), 200-209, https://doi.org/10.19139/soic-2310-5070-1822

4. H. Oukhouya; K. El Himdi. Machine Learning Models-Based Forecasting Moroccan Stock Market. Lecture Notes in Networks and Systems, **2024**, 1104, Springer, https://doi.org/10.1007/978-3-031-68628-3_6

List of submission

5. H. Oukhouya; A. Lmakri; R. Guerbaz; K. El Himdi. New Predictive Modeling in Financial Markets using Nonlinear Time Series and Deep Learning Models. Computational Intelligence. **2024**, Wiley, http://wiley.atyponrex.com/journal/coin

6. H. Oukhouya; S. Mahboub; R. Guerbaz; K. El Himdi. Data-driven Fortunes: Analysis of Multivariate Time Series Forecasting in Moroccan Stocks. Scientific African, **2024**, Elsevier, https://track.authorhub.elsevier.com

7. H. Oukhouya; A. El Rhiouane; T. Zari; R. Guerbaz; K. El Himdi. Balancing Returns and Responsibility: Markowitz Optimization for ESG-Integrated Portfolios in Morocco. Computational Economics, **2024**, Springer, Manuscript ID: CSEM-D-24-00210

8. H. Kadiri; **H. Oukhouya**; K. Belkhoutout; K. El Himdi. Dynamic Interconnections and Contagion Effects among Global Stock Markets: A VECM Analysis. ECONOMICS - INNOVATIVE AND ECONOMICS RESEARCH JOURNAL, **2024**

A comparative study of ARIMA, SVMs, and LSTM models in forecasting the Moroccan stock market

Hassan Oukhouya* and Khalid El Himdi

Laboratory LMSA,
Department of Mathematics,
Faculty of Sciences,
Mohammed V University of Rabat,
4 Avenue Ibn Batouta, BP 1014 RP, Rabat, Morocco
Email: hassan.oukhouya@um5r.ac.ma
Email: k.elhimdi@um5r.ac.ma
*Corresponding author

Abstract: Modelling and forecasting time series constitute important financial research areas for academics and practitioners. The stock markets significantly impact investment decisions in many different economic activities. Forecasting the direction of stock prices is essential in planning an investment strategy or determining the right time to trade. However, stock price movements (trends) are noisy, nonlinear, and chaotic. Predicting the evolution of price series to improve the return on investment is difficult. In this research, we are interested in modelling and forecasting the daily prices of a primary index in the Moroccan stock market. To this aim, we propose a comparative study between the results obtained from applying two independent approaches in machine learning (ML) methods as the support vector regression (SVR), long short-term memory (LSTM), and the classical method autoregressive integrated moving average (ARIMA) models. The empirical results show that LSTM and SVR nonlinear models can yield slightly better performance accuracy than the linear model ARIMA on average for log closing price series (37 trading days).

Keywords: time series forecasting; autoregressive integrated moving average; ARIMA; support vector regression; SVR; long short-term memory; LSTM; stock price; Moroccan all shares index; MASI.

Reference to this paper should be made as follows: Oukhouya, H. and El Himdi, K. (2023) 'A comparative study of ARIMA, SVMs, and LSTM models in forecasting the Moroccan stock market', *Int. J. Simulation and Process Modelling*, Vol. 20, No. 2, pp.125–143.

Biographical notes: Hassan Oukhouya received his BSc and MSc in Applied Mathematics, Statistics and Econometrics from the Faculty of Sciences at Mohammed V University in Rabat, Morocco, in 2017 and 2020, respectively. He is currently a PhD student specializing in Time Series Forecasting. His research interests encompass a wide range of topics, including stochastic processes, statistical modeling, time series analysis and modeling, data analysis, financial mathematics, R&Python programming, regression modeling, actuarial mathematics, non-parametric statistics, statistical testing, parametric statistics, time series forecasting, causality, machine learning, deep learning, stock markets and data science.

Khalid El Himdi is a seasoned professional with a PhD in Informatics and Statistics from the University of Montreal, Canada. He is currently a Professor-researcher at the Mohammed V University of Rabat, his expertise spans statistics, time series, data mining, text mining, and probability. He is particularly interested in statistical analysis, modelling, and forecasting, showcasing extensive experience in prediction methods, stochastic processes, and machine learning. His career reflects a commitment to advancing research in these fields, contributing significantly to the academic community.

1 Introduction

After the COVID-19 crisis had an impact on different sectors such as industrial, finance, health, aviation, etc., investors boosted the stock market due to the economic recession; see, for example, Foroni et al. (2020). Modelling and forecasting the future price of a stock is important for investors as it helps them reduce the risk of making the wrong decision by successfully predicting price values or movement (Baek and Kim, 2018). Forecasting stock price trends is a challenging task because it constantly deals with a dynamic market that is influenced over time by numerous

STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **12**, January 2024, pp 200–209.
Published online in International Academic Press (www.IAPress.org)

Forecasting International Stock Market Trends: XGBoost, LSTM, LSTM-XGBoost, And Backtesting XGBoost Models

Hassan Oukhouya [1,*], Hamza Kadiri [2], Khalid El Himdi [1], Raby Guerbaz [2]

[1]*Laboratory LMSA, Department of Mathematics, Faculty of Sciences, Mohammed V University of Rabat, Morocco*
[2]*Laboratory MAEGE, Department of SMAEG, FSJES Ain Sebaa, Hassan II University of Casablanca, Morocco*

Abstract Forecasting time series is crucial for financial research and decision-making in business. The non-linearity of stock market prices has a profound impact on global economic and financial sectors. This study focuses on modeling and forecasting the daily prices of key stock indices - MASI, CAC 40, DAX, FTSE 250, NASDAQ, and HKEX, representing the Moroccan, French, German, British, US, and Hong Kong markets, respectively. We compare the performance of machine learning models, including Long Short-Term Memory (LSTM), eXtreme Gradient Boosting (XGBoost), and the hybrid LSTM-XGBoost, and utilize the skforecast library for backtesting. Results show that the hybrid LSTM-XGBoost model, optimized using Grid Search (GS), outperforms other models, achieving high accuracy in forecasting daily prices. This contribution offers financial analysts and investors valuable insights, facilitating informed decision-making through precise forecasts of international stock prices.

Keywords Time series, Modeling, Forecasting, Stock market, LSTM, XGBoost, Hybrid model, Backtesting, Grid Search

DOI: 10.19139/soic-2310-5070-1822

1. Introduction

In a laissez-faire world economy, the stock market is a central platform, enabling the exchange of funds between individuals and companies. Renowned for its accessibility and the potential for significant profits, it has long been a magnet for investors searching for lucrative returns and businesses needing capital [1]. Forecasting is crucial in this dynamic landscape, utilising historical data to anticipate forthcoming events. This practice spans diverse disciplines such as environmental science, economics, business, and finance, demonstrating its broad-ranging impact and relevance. The stock market occupies a central position within financial markets, captivating the interest of researchers from both financial and technical backgrounds. The prediction of stock market price trends has emerged as a prominent research area due to its relevance to investors [2]. A significant portion of forecasting challenges revolves around examining time. A time series is a sequential arrangement of observations related to a variable. In this context, the variable under consideration is the stock price. Using sentiment analysis for predicting stock price trends within the context of financial time series helps improve the forecasting of closing prices [3]. In exploring the accuracy of classical, Machine Learning (ML), and Deep Learning (DL) methods in time series forecasting, recent researchers delve into these challenges [4].

The stock price forecasting methods can be divided into two categories: Linear Models such as Autoregressive (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving

*Correspondence to: Hassan Oukhouya (Email: hassan.oukhouya@um5r.ac.ma). Laboratory LMSA, Department of Mathematics, Faculty of Sciences, Mohammed V University of Rabat 4 Avenue Ibn Battouta, B.P. 1014 RP, Rabat, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

computer sciences and mathematics forum

MDPI

Proceeding Paper

Comparing Machine Learning Methods—SVR, XGBoost, LSTM, and MLP—For Forecasting the Moroccan Stock Market †

Hassan Oukhouya * and Khalid El Himdi

Laboratory LMSA, Department of Mathematics, Faculty of Sciences, Mohammed V University in Rabat, Rabat 10000, Morocco; k.elhimdi@um5r.ac.ma

* Correspondence: hassan.oukhouya@um5r.ac.ma

† Presented at the 1st International Online Conference on Mathematics and Applications; Available online: https://iocma2023.sciforum.net/.

Abstract: Forecasting and modeling time series is a crucial aspect of economic research for academics and business practitioners. The ability to predict the direction of stock prices is vital for creating an investment plan or determining the optimal time to make a trade. However, market movements can be complex to predict, non-linear, and chaotic, making it difficult to forecast their evolution. In this paper, we investigate modeling and forecasting the daily prices of the new Morocco Stock Index 20 (MSI 20). To this end, we propose a comparative study between the results obtained from the application of the various Machine Learning (ML) methods: Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), and Long Short-Term Memory (LSTM) models. The results show that using the Grid Search (GS) optimization algorithm, the SVR and MLP models outperform the other models and achieve high accuracy in forecasting daily prices.

Keywords: time series; modeling; forecasting; MSI 20; stock price; SVR; XGBoost; LSTM; MLP; GS

Citation: Oukhouya, H.; El Himdi, K. Comparing Machine Learning Methods—SVR, XGBoost, LSTM, and MLP— For Forecasting the Moroccan Stock Market. *Comput. Sci. Math. Forum* **2023**, *6*, 0. https://doi.org/10.3390/IOCMA2023-14409

Academic Editor: Francisco Chiclana

Published: 28 April 2023

1. Introduction

The 2008 recession, the stock market crash of 2015, the COVID-19 Pandemic, and the Russian Invasion of Ukraine are some of the most recent crises which have had an immense impact on the financial markets and the destruction of wealth worldwide. Modeling and forecasting the stock market is a challenge that many engineers and financial researchers face. The literature review examined studies on stock market prediction using Machine Learning (ML) models. It concluded that Deep Learning (DL) was the most commonly utilized model for forecasting stock price trends [1,2]. Traditional econometric methods might require improved performance in relevant nonlinear time series and may not be appropriate for directly forecasting stock prices because of their volatility [3]. However, for complex nonlinear financial time series, methods such as Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), and Long Short-Term Memory (LSTM) can detect nonlinear relationships in the forecasting of stock prices [4] and achieve better fitting results by tuning multiple parameters [5]. Hyperparameter optimization or tuning in ML refers to selecting the most appropriate parameters for a particular learning model [6]. Some studies use Grid Search (GS) optimization [7], while others use Bayesian [8,9] or pigeon-inspired optimization algorithms [10]. In this paper, the GS algorithm is used to optimize the parameters of each model, such as SVR, XGBoost, MLP, and LSTM models. Then, we compare them using seven measures: Mean Error (ME), Mean Percentage Error (MPE), Mean Square Error (MSE), Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and R^2. In [11], it was discovered that using LSTM with Moving Averages (MA) yields superior results for predicting stock prices compared to SVR, as measured by different performance criteria such as MAE, MSE, MAPE, RMSE, and R^2 values. In their research [12], Al-Nefaie et al.

Abstract

This thesis develops classical and machine learning techniques to examine forecasting models for the Moroccan stock market. We assess the performance of several models, including ARIMA, Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), and hybrid models. Our studies reveal that nonlinear models such as LSTM and SVR slightly outperform ARIMA in forecasting the MASI index. Optimized SVR and MLP models achieve high accuracy in predicting the MSI 20 index. For international indices, the hybrid LSTM-XGBoost model demonstrates superior accuracy. Hybrid models like SVR-XGBoost and LSTM-XGBoost are particularly effective for the Moroccan All Shares Index (MASI). Comparisons of post-COVID data highlight the advantages of combining linear and nonlinear elements. A robust hybrid model integrating machine learning and deep learning methods exhibits improved predictive accuracy. This research offers valuable insights for investors, financial analysts, and policymakers, emphasizing the potential of hybrid models in financial forecasting. Future work will refine these models and explore their applications in algorithmic trading and volatility forecasting.

Keywords: Time Series; Modeling; Forecasting; Moroccan Stock Market; ML Models; ARIMA; SVR; XGBoost; MLP; LSTM; Hybrid Models; Bilinear Models; Stock Price Prediction.

Résumé

Cette thèse développe des méthodes classiques et des techniques d'apprentissage automatique afin d'examiner les modèles de prévision pour le marché boursier marocain. Nous évaluons la performance de plusieurs modèles, y compris ARIMA, Support Vector Regression (SVR), eXtreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), Long Short-Term Memory (LSTM), et des modèles hybrides. Nos études révèlent que les modèles non linéaires tels que LSTM et SVR sont légèrement plus performants que ARIMA pour la prévision de l'indice MASI. Les modèles SVR et MLP optimisés atteignent une grande précision dans la prévision de l'indice MSI 20. Pour les indices internationaux, le modèle hybride LSTM-XGBoost fait preuve d'une précision supérieure. Les modèles hybrides tels que SVR-XGBoost et LSTM-XGBoost sont particulièrement efficaces pour l'indice MASI (Moroccan All Shares Index). Les comparaisons des données post-COVID mettent en évidence les avantages de la combinaison d'éléments linéaires et non linéaires. Un modèle hybride robuste intégrant des méthodes d'apprentissage automatique et d'apprentissage profond présente une précision prédictive améliorée. Cette recherche offre des perspectives précieuses pour les investisseurs, les analystes financiers et les décideurs politiques, en soulignant le potentiel des modèles hybrides dans les prévisions financières. Les travaux futurs permettront d'affiner ces modèles et d'explorer leurs applications dans le trading algorithmique et la prévision de la volatilité.

Mots clés : Séries Chronologiques ; Modélisation ; Prévision; Marché Boursier Marocain; Modèles ML; ARIMA; SVR; XGBoost; MLP; LSTM; Modèles Hybrides; Modèles Bilinéaires; Prédiction des Cours de Bourse.

Année Universitaire: 2023-2024