DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EMPLOYING DEEP LEARNING APPROACHES FOR FINANCIAL TIME SERIES ANALYSIS

by
Fırat Melih YILMAZ

February, 2023
İZMİR

EMPLOYING DEEP LEARNING APPROACHES FOR FINANCIAL TIME SERIES ANALYSIS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University In Partial Fulfillment of the Requirements for the Master of Science in Statistics, Statistics Program

by

Fırat Melih YILMAZ

February, 2023
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"EMPLOYING DEEP LEARNING APPROACHES FOR FINANCIAL TIME SERIES ANALYSIS"** completed by **FIRAT MELIH YILMAZ** under supervision of **Asst. Prof. Dr. ENGIN YILDIZTEPE** and we certify that inour opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Engin YILDIZTEPE

Supervisor

Prof. Dr. Femin YALÇIN KÜÇÜKBAYRAK

(Jury Member)

Prof. Dr. Burcu HÜDAVERDİ

(Jury Member)

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

First and foremost, I want to express my thanks to my esteemed adviser, Dr. Engin YILDIZTEPE, for all of the help and support he has given me during my academic career. As a result of his careful direction and tireless work, I was able to finish my thesis successfully. I want to express my appreciation to Prof. Dr. Özer ARABACI for his thoughtful feedback and suggestions that helped me deepen my knowledge and gain a better grasp of the topic at hand. I want to express my gratitude to everyone at Dokuz Eylül University's Department of Statistics for providing a warm and welcoming environment that helped me develop as a scholar.

My mother, Suyagul YILMAZ, deserves the greatest amount of gratitude for raising me with such confidence and unwavering support.

Fırat Melih YILMAZ

EMPLOYING DEEP LEARNING APPROACHES FOR FINANCIAL TIME SERIES ANALYSIS

ABSTRACT

Artificial intelligence (AI) is used in the financial industry to manage various operations, including algorithmic trading, portfolio allocation, and stock return predictions. Predicting stock returns, in particular, has been a hot topic recently. Picking a reliable model to predict stock returns is crucial, and there are several tools, including machine learning, pattern recognition, and artificial intelligence. Although a number of studies have reported on the accuracy of predictions made by models based on recurrent neural networks (RNNs), relatively few have investigated in detail the accuracy of predictions made using temporal convolution networks (TCNs) in the context of stock return forecasting settings. Since TCNs can better capture long-term correlations between previous returns and present stock price changes, it provides superior stock return forecasting over RNNs. Through this thesis, we assess how well state-of-the-art machine learning and deep learning models predict stock returns. To this end, we employ numerous predictive models, such as those based on linear regression, tree regression, and recurrent neural networks. We analyze the relative performance of the predictive models across various market scenarios by comparing and contrasting their respective findings. Using the model confidence set, we also show which model (or models) is statistically superior at projecting stock returns. Empirical results on four financial datasets show that the TCN model outperforms competing models in terms of out-of-sample forecast accuracy.

Keywords: Temporal convolutional networks, recurrent neural networks, model confidence set, time series forecasting, Bayesian optimization

DERİN ÖĞRENME YAKLAŞIMLARININ FİNANSAL ZAMAN SERİLERİ ANALİZİNDE KULLANILMASI

ÖZ

Yapay zeka (YZ), finans sektöründe algoritmik ticaret, portföy tahsisi ve hisse senedi getiri tahminleri de dahil olmak üzere çok çeşitli işlemleri yönetmek için kullanılmaktadır. Özellikle hisse senedi getirilerinin tahmin edilmesi son zamanlarda gündemde olan bir konudur. Hisse senedi getirisini tahmin etmek için güvenilir bir model seçmek çok önemlidir ve makine öğrenimi, örüntü tanıma ve yapay zeka dahil olmak üzere çeşitli araçlar kullanılmaktadır. Birçok çalışma, tekrarlayan sinir ağı (TSA) modellerinin tahmin gücünü göstermiş olsa da, çok azı hisse senedi getirisi tahmininde zamansal evrişimsel ağlarının (ZEA) tahmin performansını sistematik olarak araştırmıştır. ZEA'lar önceki getiriler ile mevcut hisse senedi fiyat değişiklikleri arasındaki uzun vadeli korelasyonları daha iyi yakalayabildiğinden, TSA'lara göre daha üstün hisse senedi getirisi tahmini sağlar. Bu tez çalışmasında, gelişmiş makine öğrenimi ve derin öğrenme modellerinin gelecekteki hisse senedi getirilerini ne kadar iyi tahmin ettiği araştırılmıştır. Bu amaçla, doğrusal regresyon, ağaç regresyonu ve tekrarlayan sinir ağlarına dayalı olan çok sayıda tahmin modeli kullanılmıştır. Tahmin modellerinin göreceli performansı, farklı piyasa senaryolarında analiz edilmiştir. Model güven kümesi testi kullanılarak, hangi modelin (veya modellerin) hisse senedi getirilerini öngörmede istatistiksel olarak daha üstün olduğu analiz edilmiştir. Dört farklı finansal zaman serisi üzerinde yapılan deneysel çalışma ZEA modelinin örneklem dışı tahmin doğruluğu açısından rakip modellerden daha iyi performans gösterdiğini ortaya koymuştur.

Anahtar kelimeler: Zamansal evrişimsel ağlar, tekrarlayan sinir ağları, model güven kümesi, zaman serisi tahmini, Bayes optimizasyonu

CONTENTS

Page

M.Sc THESIS EXAMINATION RESULT FORM ii

ACKNOWLEDGMENT iii

ABSTRACT v

ÖZ vi

LIST OF FIGURES viii

LIST OF TABLES ix

CHAPTER 1 - INTRODUCTION 1

CHAPTER 2 - BACKGROUNDS 4

2.1 Literature Review 4

2.2 Introduction to Time Series Forecasting Settings 7

2.3 Time Series Analysis 11

2.4 Classical Statistical Models 13

2.4.1 Multiple Linear Regression Models 13

2.4.2 Ridge Regression 14

2.4.3 LASSO Regression 15

2.4.4 Elastic-Net Regression 16

2.4.5 K-Nearest Neighbor regression 16

2.5 Tree-Based Machine Learning Models 17

2.5.1 Decision Tree 17

2.5.2 Random Forest Regression Tree 18

2.5.3 Gradient Boosting Regression Tree 19

2.5.4 LightGBM 20

2.5.5 eXtreme Gradient Boosting 21

2.6 Deep Learning Models 22

2.6.1 Multi-layer Perceptron 22
2.6.2 Recurrent Neural Networks 24
2.6.3 Long-short-term Memory 25
2.6.4 Gated Recurrent Unit 26
2.6.5 Temporal Convolutional Network 27

CHAPTER 3 - PROPOSED FRAMEWORK AND EXPERIMENTAL SETUP 30

3.1 Experimental Framework 30
3.2 Dataset 31
3.3 Hyperparameter Tuning 32
3.4 Model Evaluation 34
3.4.1 Performance Measures 35
3.4.2 Model Confidence Set 35

CHAPTER 4 - EMPIRICAL RESULT AND CONCLUSION 37

4.1 Empirical Results 37
4.2 Conclusion 40

REFERENCES 42

LIST OF FIGURES

Page

Figure 2.1 Multi-layer perceptron 23
Figure 2.2 Recurrent neural network 24
Figure 2.3 Long-short-term memory network 25
Figure 2.4 Gated Recurrent Unit network 27
Figure 2.5 Temporal convolutional network 28
Figure 3.1 Experimental framework 30
Figure 3.2 Original and log returns of four stock data 31
Figure 4.1 Forecasting results for the TCN model 40

LIST OF TABLES

Page

Table 3.1 Summary statistics of stock returns 32
Table 4. 1 Forecasting accuracy measurements for all models..... 38
Table 4. 2 Model confidence set result of the models..... 39

CHAPTER 1
INTRODUCTION

Advances in the application of artificial intelligence in the financial industry have garnered considerable attention among investors and researchers. During the past few years, there has been a notable increase in interest, especially in predicting stock prices. Forecasting stock prices is a challenging endeavor due to the inherently chaotic structure of the stock market as well as the high degree of volatility of markets. These aspects, which affect stock price behavior, range from macroeconomic conditions to the expectations of investors (Committee & Prize, 2013; Fama, 1965; Oh & Kim, 2002). Fama (1965) asserts, in accordance with the generally believed efficient market hypothesis, that stock prices already represent all historical and current information. This theory indicates that new information has no impact on the feature stock price (Fama, 1965, 1991; Schwartz, 1970). As a result, the stock price must follow a random walk pattern. According to Timmermann and Granger (2004), this widely accepted view in the stock price forecasting literature directly consequences the martingale difference sequence hypothesis. Based on the hypothesis, the stock price moves in accordance with a martingale, which indicates that it is impossible to estimate the value of the stock in the future based on the information that is currently available. This theory has been examined in a wide range of stock markets, from developing to well-established ones. All of the findings show that the mentioned hypothesis has some inconsistencies. Malkiel (2003), for instance, criticizes the efficient market hypothesis, claiming that it has become far less prevalent since the dawn of the twenty-first century. Recent researches also show that the price of a stock can be predicted to some extent based on how it has behaved in the past. These patterns allow investors to get returns that are higher than average when risk is taken into account.

Four main but unique areas of study in the literature can be used to study stock prices within that sense. In order to understand the underlying stock price changes, fundamental and technical analyses both look at current economic conditions and historical stock prices. Secondly, some studies have concentrated on modeling asset volatility, which is critical in various financial applications that use the autoregressive

conditionally heteroscedastic (ARCH) model family (Andersen, Bollerslev, & Meddahi, 2011; Brandt & Jones, 2006; Jayawardena, Todorova, Li, & Su, 2020). To round out this group of models, there are nonlinear alternatives—for instance, autoregressive integrated moving average, autoregressive fractionally integrated moving average, and smooth transition autoregressive (STAR) models, which are used in both univariate and multivariate time series analysis (Adebiyi, Adewumi, & Ayo, 2014; Panas, 2001; Sarantis, 2001). The fourth body of studies heavily relies on data-driven approaches as well as machine learning models (Abdullah & Ganapathy, 2000; Başoğlu Kabran & Demirberk Ünlü, 2020; Kim, Ku, Chang, & Song, 2020; Kumar, Singh, & Jain, 2021; Parray, Khurana, Kumar, & Altalbe, 2020; Yilmaz & Yildiztepe, 2022; Yu & Yan, 2020). Moosa and Vaz (2015) assert that even with all of these advancements, models that assert to be superior to the RW model frequently have no sophisticated statistical evaluation testing.

Existing studies have extensively focused on the inability of linear forecasting models to predict stock prices due to the high degree of volatility in the stock market. However, there is much evidence suggesting stock prices would be better described by a model that includes nonlinearities, even though most research studies the prediction of stock prices using linear models (Abhyankar, Copeland, & Wong, 1997). In light of such results, nonlinear time series models have become increasingly common for predicting stock prices. Concurrently, ANNs are gaining popularity because it is data-driven and adaptable and can estimate any unknown function without requiring a priori knowledge of the functional structure of data. ANNs, on the other hand, are incapable of learning the long-term dependence of time series in the financial domain. Deep learning models have been developed specifically for the purpose of overcoming this challenge. This development has led to the rise in the popularity of models such as convolutional neural networks (CNNs), gated recurrent units (GRUs), recurrent neural networks (RNNs), and short-term memory (LSTM) networks for predicting financial time series.

A growing amount of recent research has asserted the ability of deep learning models to perform financial time series analysis. While various evaluation measures

such as the MAE or RMSE have been used to assess model performance in previous research on forecasting stock markets, this thesis study endeavors to *plenish a lacuna* in the literature by applying fifteen different models, including classical machine learning models and state-of-the-art deep learning models, to forecast four major stock markets: the standard & poor's 500 index (S&P 500), the bombay stock index (BSE), the taiwan delayed price (TSEC), and the dow jones industrial average (DJIA). The accuracy of forecasting results of these models is assessed with the help of three distinct performance metrics: mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE). To determine whether a model (or models) outperformed forecasting methods by a significant amount, the model confidence set was applied (MCS)—a technique introduced by Hansen et al. (2011) as an advanced evaluation test. Also, the best hyperparameters for the models were found using a method called Bayesian optimization, which is a cutting-edge way to tune models.

This thesis is organized as follows: The study's models (described in Chapter 2) and experimental methodology (discussed in Chapter 3) are presented in subsequent chapters. Findings and interpretations from the empirical study are presented in Chapter 4.

CHAPTER 2
BACKGROUNDS

2.1 Literature Review

Researchers in the field of machine learning are focusing on building a model which can provide real-time functioning applications for the finance system. Stock price forecasting is one such area of research. As a result, there are a vast amount of studies and applications in the literature. Deep learning, on the other hand, is a new area of study in the science of machine learning that is gaining more and more attention every year. This has resulted in an increase in the number of deep learning models for finance that are being published in academic journals. In this section, listed below are some of them in chronological order, along with a discussion of our motivations and distinctive perspectives.

It is believed that the first-time neural networks were used for stock price forecasting was in 1991 by Yoon and Swales (1991), who demonstrated that it outperforms other models. This accomplishment allowed Chen, Leung, Daouk (2003) to prove their claim that ANNs are superior to GARCH and ARIMA models when it comes to forecasting the Taiwan Stock Exchange stock index. In order to determine whether or not the Spanish Ibex-35 stock index exhibits nonlinear qualities, Pérez-Pérez-Rodrguez, Torra, & Andrada- Félix (2005) demonstrated that the ANN model outperformed the STAR and RW models in both one-step-ahead and multi-step-ahead forecasting approaches.

Fully connected networks have the drawback of not being able to detect or foresee long-term correlations between financial time series. Fortunately, deep learning methods have been developed to fulfill this need. As a result, many neural network models have been studied for their potential in financial time series forecasting. These models include recurrent neural networks (RNNs), long short-term memory (LSTM), gated recurrent units (GRUs), and convolutional neural networks (CNNs).

RNNs are seen as being successful in financial modeling in a number of recent papers. This success is evidenced by the work of Dang Khoa, Sakakibara, & Nishikawa (2006), who utilized a feed-forward neural network (FFNN) as well as RNN models trained with time-based back-propagation and early stopping, and Jiang, Tang, Chen, Wang, & Huang (2019), who evaluated the success of RNNs and LSTMs for forecasting the Shanghai Composite Index and the Dow Jones Index and discovered that the latter outperformed the former. Shen, Tan, Zhang, Zeng, & Xu (2018) also reported that the RMSE and MAE scores of the hybrid model featuring the support vector machine and GRU and the GRU model alone were smaller than the scores of the ANN and SVM models, respectively. In order to predict stock prices for algorithmic stock trading, Karaoglu, Arpaci, & Ayvaz (2017) utilized market microstructure-based swap information as input for an RNN and LSTM. Zhang, Aggarwal, & Qi (2017) introduced the State Frequency Memory (SFM) network, which was found to be more accurate in both prediction and trading for forecasting stock prices with various frequency trading patterns. To foretell price changes in a high-frequency trading system (HTF), Tran, Magris, Kanniainen, Gabbouj, & Iosifidis (2018) proposed a deep learning model that uses tensor representation and high-frequency limit order book data. Many studies have found that LSTM is the preferred method for financial analysis, including those by Fischer & Krauss (2018), Sezer, Gudelek, & Ozbayoglu (2020), and Ozbayoglu, Gudelek, & Sezer (2020).

CNNs, also known as the ConvNet model, are more regulated variants of ANNs that are more efficient. In the beginning, it was offered as a solution to problems involving computer vision. Sequential modeling tasks are another area where it can be applied. Despite RNNs typically being considered superior for time series forecasting, an increasing number of researchers are using CNNs in this field.

Researchers including Tsantekidis et al. (2017), Selvin, Vinayakumar, Gopalakrishnan, Menon, & Soman (2017), Hiransha, Gopalakrishnan, Menon, & Soman (2018), and Hosaka (2019) have found that CNNs can perform as well as or better than RNNs for time series modeling. Because CNNs are able to learn larger dependencies in the time series due to their larger maximum path sizes $\left(O(\log_k (n))\right)$,

and they are able to use parallelized computation because there is no dependency between the current and previous time steps in their sequential operations ($O(1)$). In contrast, RNNs require sequential computation because the current time step depends on the previous time step and have smaller maximum path sizes ($O(n)$), which limits their ability to learn larger dependencies in the time series. These differences in power and performance have led researchers to investigate the use of CNNs for sequential modeling tasks such as time series analysis and natural language processing, among other applications.

Temporal convolutional networks (TCNs), a type of CNN designed explicitly for time series modeling, have gained popularity due to their performance. Bai, Kolter, & Koltun (2018) proposed the use of causal dilated convolution to learn long-term dependence on sequential data without information loss, and researchers have found that TCN-based neural network designs can produce results that are comparable to or better than those obtained with RNNs (Bai et al., 2018; Dauphin, Fan, Auli, & Grangier, 2016; Hanson & Raginsky, 2019; Yin, Fan, Auli, & Grangier, 2017). Borovykh, Bohte, & Oosterlee (2018) also demonstrated that stacks of dilated convolution networks outperform LSTM without requiring a long historical time series for multiple financial time series. Chen et al. (2020) used a representation learning-based probabilistic TCN model to forecast a sales domain dataset and outperform all other benchmark models, and Wan, Mei, Wang, Liu, & Yang (2019) and Lara-Bentez, Carranza-García, Luna-Romera, & Riquelme (2020) found that the TCN model consistently produced the best forecast performance.

As a result of the inspiration discussed earlier and the study that was conducted within the research area, this thesis will *confer two novae* to the corpus of literature:

- In the first step of this process, we will contribute to the existing body of study on TCN networks as a means of prediction by evaluating the stability of four major stock markets. These markets are the TSEC, DJIA, the S&P 500, and the BSE. Therefore, it will be fascinating to determine whether there are differences in performance and differences in the durability of performance. In addition, we will add

to the work that has already been done by utilizing state-of-the-art methods for model tuning, namely Bayesian optimization, in an effort to determine whether or not this can increase performance.

- Second, rather than using essential performance evaluation metrics, we will use an advanced evaluation test known as the model confidence set (MCS) to assess the accuracy of forecasts produced by competing models. This examination will put us in a position to go further than any previous research in this area. The vast majority of academic research does not make use of any kind of formal statistical evaluation testing process. Instead, studies of this nature typically rely on fundamental forecasting evaluation measures. As a result, we will work to make the thesis more applicable to actual practices and situations in the real world.

2.2 Introduction to Time Series Forecasting Settings

Forecasting methods are composed of historical data, referred to as "time series," and a series of observations that are ordered according to time in a given period. A time series is a set of values $\{Y_t: t = 1, \dots, n\}$ that is finite in length and comprises the numbers that were measured for a variable Y from one point in time to the next. The purpose of time series analysis is to investigate the development of a phenomenon throughout the course of history; the prediction is reached by presuming that behaviors exhibiting comparable regularities will continue to be observed in the foreseeable future. To put it another way, it is essential to present the concept of stochastic processes to provide a precise mathematical definition.

Consider the space of probabilities denoted by (Ω, F, P), in which Ω denotes a set of instances, F defines σ-algebra, which is the collection of all possible outcomes, and the P specifies the probability measure $(P: \Omega \rightarrow [0,1])$. In the presence of a probability space (Ω, F, P), one can conceive of a time series as a realization of a stochastic process.

Def 1. *A group of random variables* $(Y_t)_{t\in T}$ *that are described on* (Ω, F, P)*, and their values are found in* $(\mathbb{R}, B(\mathbb{R}))$ *is an example of a stochastic process. The variable* T *represents the collection of index values that uniquely identify each element of the stochastic process.*

For this reason, a stochastic process is defined as a collection of random variables ordered for a certain parameter. One way to understand a time series is to see it as an instantiation of a stochastic process since it is a finite collection of observations relating to a specific occurrence arranged in time. Between these two definitions, it is possible to distinguish one from the other. A discrete parameter stochastic process is defined by a set of distributions or by having all feasible probability distribution functions, while a time series is a set of numbers sorted by time variable.

In the case where T is a continuous space of $\mathbb{R}$, the stochastic process can be defined as continuous and is denoted by $\{Y(t): t \in T\}$; if $T \subseteq N(\mathbb{Z})$ is a discrete interval of the integers, the stochastic process is said to be discrete and is indicated by $\{Y(t): t \in T\}$. Here, $\mathbb{Z}$ represents the set of all integers, and $N(\mathbb{Z})$ denotes a subset of $\mathbb{Z}$ that is either finite or countably infinite. In order to keep things simple, we shall solely discuss discrete-time stochastic processes. At time t, fixed $t \in T, Y_t(\cdot)$ denotes a random variable that shows the level of the process; fixed $\in \Omega$, the application $t \in T \to Y_t(\omega)$, represents a realization of the process.

Def 2. *Consider the set T, which contains all the vectors* $\{\boldsymbol{t} = (t_1, \dots, t_n)' \in T^n: t_1 < \cdots < t_n, n = 1,2, \dots\}$*. Suppose that the finite probability distribution functions of the stochastic process* $\{Y(t): t \in T\}$ *are denoted by the function* $\{F(t), t \in T\}$ *specified for* $t = (t_1, \dots, t_n)'$ *as*

$$F_t(y) = P\left(Y_{t_1} \leq Y_1, \dots, Y_{t_n} \leq y_n\right), \ \mathrm{Y} = (Y_1, \dots, Y_n)' \in \mathbb{R}^n \tag{2.1}$$

If the distributions of $\left(Y_{t_1}, \dots, Y_{t_n}\right), \forall (t_1, \dots, t_n) \in T, \forall n \in \mathbb{N}, \forall Y = (Y_1, \dots, \mathrm{y}) \in \mathbb{R}^n$ are highly compatible described as well as the distribution functions $F_t, \forall t \in T$ are known, then the process $\{Y_\mathrm{t}\}_{t\in T}$ can be determined probabilistically from the

distribution functions F. Here, $\mathbb{N}$ denotes the set of natural numbers, which encompasses all positive integers.

When conducting an analysis of time series, the principal objective is to identify a suitable stochastic process. This process should have trajectories that agree with the data and be able to be utilized for forecasting the future. Our first order of business should be to narrow our attention to a particular group of stochastic processes. This will assist us in unambiguously identifying the process that serves as our model. The concept of *stationary processes* is introduced in this section, and it is a very significant class of stochastic processes. A time series is considered stationary if its mean and variance do not vary greatly. Stationary processes can be further divided into two categories:

Def 3. *A sequence of events in time donated by* $\{Y_t\}_{t\in T}$ *is considered to be strongly stationary If* $\forall k \in \mathbb{R}, \forall t_1, \dots, t_n \in T, \forall n \in \mathbb{N}, \forall y = (Y_1, \dots, Y_n) \in \mathbb{R}^n$:

$$F_{Y_{t_1},\dots,Y_{t_n}}(Y_1, \dots, Y_n) = F_{Y_{t_1+k},\dots,Y_{t_n+k}}(y_1, \dots, y_n) \tag{2.2}$$

As a result, a temporal slip k has no effect on the joint distribution of a series that fits the criteria of the definition above. The verification of this requirement in practice, on the other hand, is highly challenging.

Def 4. *When* $E[Y_t] = \mu$ *is constant and does not depend on t and* $E\left[Y_{t_1}Y_{t_2}\right] = E\left[Y_{t_1+k}Y_{t_2+k}\right], \forall t_1, t_2 \in T, \forall k \in \mathbb{R}$*, then the process* $(Y_t)_{t\in T}$ *can be called as weakly stationary in* k.

As a consequence of this, the statistical moments of the process are purely controlled by time differences and not by the times at which the data that was used to predict the statistical moments occurred. As a result, this type of stationarity requires invariance of the first two moments. This is a matter of concern at this point in time.

Generally speaking, in the context of financial time series forecasting, the time series is supposed to be weakly stationary, and for the purposes of this discussion, we will assume that this is the case currently. When the empirical conditions fail to meet this requirement, it is sufficient to revert to the stationary situation, in which $Z_t = Y_t - Y_{t-1}$ is taken to be the series in the first place. It is possible to satisfy the criteria for stationarity by continuously repeating this procedure. Time series analysis and forecasting can be classified into two categories: analysis and forecasting. It is essential to do modeling to acquire an awareness of the key characteristics of an event that has occurred, which can aid in the forecasting process. The models that are most typically utilized to predict future behavior are generated from the form of the sequence in the past, minus any unintentional changes. These changes will be distinguished from the pattern in order to avoid variability having an impact on the future tendency in the series.

Let us describe the concept of lag: the y_{t-1} represents the first lag in a time series y_t, and the y_{t-k} represents the last lag in a k^{th} series. The mean, variance, and covariance of density functions can be defined as $E(Y_t) = \mu_t$, $\mathrm{Var}\,(Y_t) = \sigma_t^2$, and $\mathrm{Cov}\,(Y_t, Y_{t-k}) = \gamma(k,t)$, respectively, if the density functions have moments.

In the context of stochastic processes, a process is said to be stationary if its mean and variance are constant and finite. The mean and variance are denoted by μ and σ^2, respectively. Furthermore, the covariances at lag k are denoted by

$$\gamma_k = \mathrm{E}[(Y_t - \mu)(Y_{t-k} - \mu)] \tag{2.3}$$

Because they are not dependent on time i but rather on k, they are also known as autocovariance. This is due to the fact that they vary independently of k, —for example, such as $\gamma(k) = \gamma_k$.

Def 5. *In time series analysis, autocorrelation is used to determine how closely a time series is related to itself in the past. When k is a lag, the autocorrelation of y_t is*

$$\rho_k = \frac{\text{Cov}\,(Y_t, Y_{t-k})}{\sqrt{\text{Var}\,(Y_t)\text{Var}\,(Y_{t-k})}} = \frac{\gamma_k}{\gamma_0} \tag{2.4}$$

here γ_0 denotes the variance term, beginning with $\rho_0 = 1$. The ρ_k can be estimated using the set $\{Y_t\}_{t=1}^{n}$, which is as follows

$$\hat{\rho}_k = \frac{\text{Cov}\,(Y_t, Y_{t-k})}{\text{Var}\,(Y_t)} = \frac{\sum_{t=1}^{n-k}(Y_t - \bar{Y}_0)(Y_{t+k} - \bar{Y}_k)}{\sqrt{\sum_{t=1}^{n-k}(Y_t - \bar{Y}_0)^2 \sum_{t=1}^{n-k}(Y_{t+k} - \bar{Y}_k)^2}},\ 0 \le k < n-1 \tag{2.5}$$

here $\bar{Y}_0 = \frac{1}{n-k}\sum_{t=1}^{n-k} Y_t, \bar{Y}_k = \frac{1}{n-k}\sum_{t=k+1}^{n} Y_t$ denotes the mean of the sample and mean of sample with a lag of k, respectively. It is possible to detect underlying similarities in a time series using the autocorrelation function. If a sequence has a strong regularity but no discernible trend, then $\hat{\rho}_k$ will be greater at each period; if, on the other hand, there is a discernible trend, the autocorrelation estimation becomes very significant all over the place. As a result, for the function $\hat{\rho}_k$ to be useful, it is necessary that the trend is be moderate.

2.3 Time Series Analysis

One way to look at the composition of a historical series is as the combination of numerous elements, including the followings:

- The *trend*, which is an indication of long monotonic movement, and cyclical variations, which are combinations of stretching of the market situation where the phenomenon being considered is placed, are the two parts that make up the trend cycle. The trend indicates long monotonic movement. Cyclical variations are combinations of expansion and contraction. This feature undergoes several subtle shifts as the game progresses and is essential in establishing where a level fits in the overall series.

- *Seasonality* refers to the existence of one or more periodic elements that occur at or close to a predetermined point in time. This is typical for interim time series, such as monthly or quarterly measurements.

- The *irregular* factor refers to the perturbation part that cannot be described by the preceding parts, whose cycles are characteristic of a limited timeframe.

The analysis of time series can be approached from a number of different perspectives, the most common of which are the classical and modern methodologies. The classical one assumes that the process described by the series has both a deterministic component, which enables one to estimate the virtual components that have been defined and a random disruption component. The modern methodology, on the other hand, begins with the presumption that the series was produced by a stochastic process with associated components that are capable of being characterized by suitable probabilistic models. In order to create accurate statistical predictions, it is crucial to have a solid understanding of which of the numerous methods to use. Because of this, it is essential to be familiar with the assumptions that serve as the foundation for the various approaches.

The followings are the stages that go into developing a forecast for a time series:

- Gathering data followed by checking to ensure that the data are high quality and accurate. It is essential to conduct an accurate analysis of the definitions and sources of the data that is now at one's disposal, as well as the ability of data to be compared across different periods.

- Exploratory investigation on the structure of time series is a study of the series from a graphical perspective as well as using descriptive statistics to emphasize the likely presence of time series components.

- Selection and estimation of the model that is thought to be the most suited for use in calculating the values of the parts of the historical data.

- Evaluating the performance of the models and exploring potential applications for improving forecasting.

The presentation of graphs that illustrate the trend and the potential existence of variations is the first step that is helpful for comprehending data and analyzing a time series. In most cases, one will start by building and analyzing a graph covering the whole series. The values of the observed phenomena *Y* are published inordinately along the x-axis at each time *t* in this form of a graph, which is frequently referred to as a lengthy period of time plot. Imagine that the level of the series remains relatively consistent for the entirety of the time period that is being observed. In this scenario, if the series level stays almost the same over the observation time, then this suggests that the series is stationary on average. As a result, the spline that represents the time plot should fluctuate around a constant value that is equivalent to the series average. If, on the other hand, the series is transformative, which denotes that it is not stationary, then the time plot will show the increasing but rather decreasing linear or nonlinear trend, depending on the case.

2.4 Classical Statistical Models

This section presents models that are referred to as "competitive models," based on a classical statistical approach. These models include the MLR, Ridge, Lasso, Elastic-Net, and KNN models, each of which is based on different statistical assumptions.

2.4.1 Multiple Linear Regression Models

In forecasting contexts, where the response variable is typically continuous, and the explanatory variables are typically discrete, multiple regression is a fundamental technique for time series modeling. By fitting the dependent variable to the independent variables, the linear model models the correlation between the response variable *Y* and the descriptive factors *X*. This straight line is also known as the regression line. The model of multiple regressions can be described as follows:

$$Y_i = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki} + \varepsilon_i,\ i\ = 1,2,\cdots,n \tag{2.6}$$

where Y is the response variable, $\beta_0, \beta_1, \cdots, \beta_k$ are $(k+1)$ coefficients, $X_1, \cdots, X_k$ are k-dimensional variables, ε are the linearly independent, identically and Gaussian-distributed errors, and n is the number of observations. There are various methods for estimating the coefficients, but the ordinary least squares (OLS) method is the most commonly used method for this purpose. The ordinary least squares estimator can be defined as

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T Y \tag{2.7}$$

$\hat{\beta}_{OLS}$ is the unbiased estimator of β_0 in this case. The least-squares estimator relies on several assumptions, including linearly independent, identically distributed errors with a mean of zero, constant variance, and homoscedasticity. However, it can be affected by multicollinearity, which can lead to unstable estimates, incorrect signs for the coefficients, and increased variation in the estimates, making it difficult to determine the accuracy of the model. Alternative approaches for addressing the issue of multicollinearity have been proposed to address this problem.

2.4.2 Ridge Regression

If the explanatory variables are correlated—or, to put it another way, if the data struggle from multicollinearity—the least square estimates are unbiased; however, their variance is high, which results in a significant difference between the value that was observed and the value that is being measured. As a result, the ridge regression technique, invented by Horel and Kennard (2000), is used in conjunction with biasness, which minimizes the adequate number of parameters and, as a result, leads less standard errors. More precisely, let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ is be a set of observed data, $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^\top$ be the explanatory data matrix and $Y = (y_1, y_2, \ldots, y_n)^\top$ be the response data vector. The objective function for the ridge regression is defined as

$$\hat{\beta}^{\text{ridge}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2.8}$$

here, the penalty factor λ acts as a form of regularization for the coefficients, penalizing the optimization function whenever the coefficients assume large values. Other than normality, the assumptions used in this regression are identical to those made in the least square technique, except for the assumption of normality of the error term.

2.4.3 LASSO Regression

When using ridge regression, one potential pitfall is that estimated coefficients may be shrunken toward zero, but they will never be absolutely zero. This feature is a by product of the imposition of a quadratic loss function in ridge estimation, and it has the potential to degrade forecast accuracy in some situations. In order to permit some coefficients to be precisely zero, an absolute loss function must be used. One such function is Tibshirani's (1996) least absolute shrinking and selecting agent (LASSO), which permits certain values to be precisely zero. The LASSO is intended to decrease some regression coefficients approaching zero while maintaining the value of others at zero. As a result, it combines the characteristics of model selection with the characteristics of shrinkage estimation. Assuming that the linear regression has been performed by standardizing the $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^\top$ explanatory data matrix, and centering the response data vector $Y = (y_1, y_2, \dots, y_n)^\top$, the lasso solves the penalized regression problem of determining β in order to minimize

$$\hat{\beta}^{\text{lasso}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{n=1}^{N} (y_n - \beta x_n)^2 + \lambda \sum_{i=1}^{p} |\beta_i| \tag{2.9}$$

This is identical to minimizing the sum of squares while subjected to a constraint of the type $\Sigma|\beta_j| \leqslant s$, as described above. A ridge regression with constraint $\Sigma_j \beta_j^2 \leqslant t$ is analogous to this method of regression (Tibshirani, 2011). A consequence of the l_1-penalty's shape is that the lasso does both variable selection and shrinkage while the ridge regression method shrinks.

2.4.4 Elastic-Net Regression

The elastic-net model, developed by Zou and Hastie (Zou & Hastie, 2005), integrates the strengths of LASSO and ridge regression to address their limitations. Its first part selects variables, and the second stabilizes the solution, leading to improved prediction accuracy, particularly when correlations between explanatory variables are large, through the use of a convex combination ℓ_1 and ℓ_2 penalties

$$\hat{\beta}^{\text{ElNet}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_i (y_i - \beta' x_i)^2 + \lambda_1 \sum_{k=1}^{K} |\beta_k| + \lambda_2 \sum_{k=1}^{K} \beta_k^2 \tag{2.10}$$

The elastic net, containing the LASSO and ridge regression as examples, is located within the range of $\lambda \in [0,1]$. The elastic-net problem is accurately solved using coordinate descent techniques, similar to the LASSO.

2.4.5 K-Nearest Neighbor regression

The K-nearest neighbor is a nonparametric method that makes predictions about the target outputs of the K-nearest neighbors of a given data instance by basing those predictions on the target outputs of the data instance itself (Ziegel, 2003). When a data point is provided, a calculation is made to determine the Euclidean distance between that data point and every other data point that is part of the training set. After that, the training data points that are the most relevant are chosen, and the value of the prediction is determined by taking the mean of the target output for each of these points. Let $\mathcal{g}(x)$ represent the collection of points that are closest to point x in terms of numbers. Then the prediction is given by

$$\hat{y} = \frac{1}{K} \sum_{m \in \mathcal{F}(x)} y_m \tag{2.11}$$

where y_m is the target output for a training data point x_m. This approach relies on the parameter x, which must be chosen with care because it is vital. In general, a significant

value of *K* will result in a smoother fit and consequently a lower variance, but at the expense of a higher bias and the converse for a small value of K.

2.5 Tree-Based Machine Learning Models

This section presents models include the DT, RF, GBRT, LightGBM, and XGBoost, which are all predictive models that can be used to classify, predict, and learn from data.

2.5.1 Decision Tree

The decision tree is a methodology usually employed for various applications, including classification and regression analysis. Initially, the purpose of the decision tree was to tackle classification problems, such as those posed by the Iterative Dichotomiser 3 (Quinlan, 1986) and C4.5 (Quinlan, 1996; Salzberg, 1994). As a result of its exceptional performance, its usage in resolving problems involving regression and classification (CART) (Breiman, Friedman, Olshen, & Stone, 2017) has seen rapid expansion in recent years. In a tree algorithm, each interior node refers to a specific input, and each interior node is split into child nodes depending on the values of the input parameter. These child nodes can then be used to process the data further. Each terminal node stands for a particular value of a target attribute, such as the correct category when dealing with classification tasks or the actual numerical value when dealing with regression issues. To put it another way, in order to construct the tree, at each stage, one of the variables that are input is chosen, and the samples are divided in half based on that variable. An attribute value test is performed on a split point that is located along a tentative variable. Given the results of the test, the ideal split point is selected to split the interior node into consecutive nodes. When constructing a decision tree, the objective is to split the input data as much as possible to produce estimated outputs closer to the actual outputs. This will result in fewer discrepancies between the estimated and actual outputs. Following is a formula that can be used to calculate the estimated outcomes from a terminal node. This formula works by calculating the mean of the actual outputs that were obtained from the training set that were obtained from the terminal node,

$$\hat{y}_i = \frac{\sum_{j \in t_i} y_j}{|t_i|} \tag{2.12}$$

where t_i refers to the leaf node while $|t_i|$ represents the set of samples found within the i-th node. The impurity measure using the least-squares deviation determines the splitting criterion, such as,

$$I(t_i) = \sum_{j \in t_i} \left(y_j - \hat{y}_i\right)^2 \tag{2.13}$$

where $I(t_i)$ denotes the impurity measure within the i-th node. The splitting ratio is calculated as follows, using the least-squares deviation as a basis

$$\Delta I = I\left(t_p\right) - P_l I(t_l) - P_r I(t_r) \tag{2.14}$$

In this scenario, the parent node is denoted by t_p, two child nodes of t_p are denoted by t_l and t_r, and the percentage of data samples that are allocated to each child node is denoted by P_l and P_r, respectively. In order to optimize ΔI, the split point is calculated.

2.5.2 Random Forest Regression Tree

The random forest, a form of ensemble learning introduced by Breiman (2001), can be used for classification and regression tasks. It comprises bagging with a random subspace approach and random feature subspace analysis at each classification node and regression tree node. In other words, it generates decision trees that function as regression functions and produces the mean of those outputs as its final output. Assume that $S = \{(x_i, y_i)\}_{i=1}^{N}$ is a set of observed data in which $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^\top$ and $Y = (y_1, y_2, \dots, y_n)^\top$demonstrate the explanatory data and response data, respectively. An L tree-structured base classifier $h(X, \Theta_k]$, is used in the training step (Breiman, 2001). It is an ensemble method that uses a bagging algorithm to merge all generated decision trees. Bagging may be employed with several regression

algorithms to reduce prediction variance and hence improve prediction performance. In order to obtain a bootstrap sample, one must first select π samples at random from S, with each observation having a chance of 1/n to be selected. The bagging technique starts with a set of bootstrap samples called $\left(S_{\pi}^{B_1}, \ldots, S_{n}^{\theta_4}\right)$. It then uses the prior tree decision process to these samples in order to build a set of q estimation trees called $\hat{h}\left(X, S_{n}^{\theta_1}\right), \ldots, \hat{h}\left(X, S_{1}^{\theta_1}\right)$. The ensemble generates q outputs, one for each of the $\hat{y}_1 = \hat{h}\left(X, S_{x}^{\theta_{11}}\right), \hat{y}_2 = \hat{h}\left(X, S_{x}^{\mathrm{D}_2}\right), \rightarrow \hat{y}_{\mathrm{q}} = \hat{h}\left(X, S_{n}^{\theta_4}\right)$ trees. The aggregation process is then carried out by calculating the mean of outputs. As a result, the following objective function can acquire the estimation Y of the output

$$\bar{y} = \frac{1}{q}\sum_{1=1}^{q} \bar{y}_i = \frac{1}{q}\sum_{1=1}^{p} \hat{h}\left(X, 5_{n}^{q_1}\right) \tag{2.15}$$

where $\hat{y}$ represents the final output of the random forest regression is the mean of those outputs.

2.5.3 Gradient Boosting Regression Tree

In contrast to the traditional CART algorithm, the gradient boosting regression tree (GBRT) algorithm improves upon it by using boosting, a method that combines numerous poor classifiers to create a powerful one instead of optimizing a single model (He, Kamarianakis, Jintanakul, & Wynter, 2013). GBRT sequentially generates new decision trees by reducing the residuals of the existing trees, a process called residual reduction (Elith, Leathwick, & Hastie, 2008). Adding a new tree at each trial optimizes the prediction by decreasing the loss function, making this serial modeling approach essentially a type of functional gradient descent. Given a set of observed data $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^{\top}$ and response data $Y = (y_1, y_2, \ldots, y_n)^{\top}$ and a pre-selected loss function $L(y, F(x))$ that measures the deviation between the predicted value $F(x)$ and the actual response value y, GBRT starts with the initial model $F_0(x)$ and iteratively reduces the residuals by optimizing the coefficients the coefficients ρ_m and α_m for each of the M decision trees produced as

$$(\rho_m, \alpha_m) = \arg\min_{\rho,\alpha} \sum_{i=1}^{N} L[y_i, F_{m-1} + \rho h(x_i; \alpha)] \tag{2.16}$$

and get,

$$F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m) \tag{2.17}$$

While training, overfitting is a common problem that can arise. Shrinkage is just one regularization strategy that gradient boosting frequently makes use of shrinkage, which provides a new variable *v* to control the update rate of the model according to the boosted gradients. In doing so, we can reduce the likelihood of overfitting and enhance the model's adaptability to novel data such as,

$$F_m(x) = F_{m-1}(x) + \nu_m \cdot \rho_m h(x; \alpha_m),\ 0 < \nu_m < 1 \tag{2.18}$$

In areas of the map with low values of ν_m, the model is updated at a slower pace. Research has shown that using small learning rates can improve the generalization capacity of models, but at the cost of longer computation time due to the need for additional decision trees (Ziegel, 2003).

2.5.4 LightGBM

LightGBM is a machine learning method that aims to improve model accuracy by using Exclusive Feature Bundling (EFB) and Gradient-based One-Side Sampling (GOSS) (Japkowicz & Shah, 2011). LightGBM also uses the histogram method to process feature data, which involves discretizing continuous data into *K* numbers to generate a histogram. The histogram is constructed by calculating the number of instances based on the *K* numbers at the time of data input into the classifier. The optimal feature separation point is determined based on the number of attributes. As a result, model attributes have to be calculated once, avoiding the recurring effort of standard machine learning methods for attribute calculation, reducing memory requirements, and boosting computing performance. Assume that $S = \{(x_i, y_i)\}_{i=1}^{N}$ is

a set of observed data in which $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^\top$ and $Y = (y_1, y_2, \dots, y_n)^\top$demonstrate the explanatory data and response data, respectively. Using the CART regression tree as a learner and iterating during K times, it is possible to define the final LightGBM method $\hat{y}_i$ integrated across K iterations in the following ways

$$\hat{y}_i = \sum_{t=1}^{K} f_t(x_i), f_t \in \Phi \tag{2.19}$$

where f_t is a function in the joint K function space. The objective function is defined as follows:

$$J(f_t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{K} \Omega(f_t) \tag{2.20}$$

where l is the error term, and Ω prevents the model from generalization problems.

2.5.5 eXtreme Gradient Boosting

XGBoost (T. Chen & Guestrin, 2016) is a recently developed ensemble method that has received widespread adoption in a variety of fields. As a technique that integrates the Cause Based Decision Tree (CBDT) and the Gradient Boosting Machine (GBM) into a single effective algorithm, it has gained the capacity to improve the tree-boosting approach's capability to handle practically all types of data rapidly and effectively. Assume that $S = \{(x_i, y_i)\}_{i=1}^{N}$ is a set of observed data in which $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^\top$ and $Y = (y_1, y_2, \dots, y_n)^\top$demonstrate the explanatory data and response data, respectively. The XGBoost can be defined with $\hat{y}_i$ integrated and K iterations in the following ways

$$\hat{y}_i = \sum_{t=1}^{K} f_t(x_i), f_t \in \Phi \tag{2.21}$$

where f_t is a function in the joint K function space and belongs to Φ. The objective function of XGBoost is defined as the sum of the error term, $l(y_i, \hat{y}_i)$, for all data points in the observed set, as well as the regularization term, $\Omega(f_t)$, for all trees in the model:

$$J(f_t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{t=1}^{K} \Omega(f_t) \tag{2.22}$$

where l is the error term, and Ω prevents the model from generalization problems.

2.6 Deep Learning Models

This section discusses models that are referred to as "major models." These models include MLP, RNN, LSTM, GRU, and TCN. In addition, MLP and RNN are the models for predicting stock returns employed frequently and have been subjected to the most research in the published literature.

2.6.1 Multi-layer Perceptron

The universal approximation theory (Cybenko, 1989) played a significant role in the popularity of multi-layer Perceptrons in the 1990s. Specifically, Takens (1981) proposed a powerful theorem that mathematically shows the existence of a functional connection, represented by $\mathcal{G}: \mathbb{R}^m \to \mathbb{R}$ such as

$$x_t = \mathcal{G}(x_{t-1}, x_{t-2}, \dots x_{t-m}) \tag{2.23}$$

for a given time series with a delay when the embedding dimension *m* is considered. While the time delay embedding theory does not specify the shape of $\mathcal{G}(\cdot)$, Hornik (1991) showed ANNs could accurately and efficiently approximate any objective function $\mathcal{G}(\cdot)$.

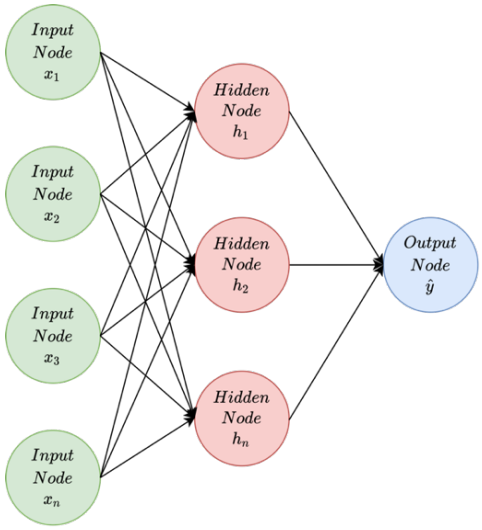


Figure 2.1 Multi-layer perceptron

The feed-forward neural network (FFNN) is a type of artificial neural network (ANN) frequently used to model time series. This type of ANN comprises three components: input nodes, hidden units, and output units (see, Figure 2.1). The input layer is constructed of p-nodes, each of which is connected to the hidden units utilizing an input vector. In time series forecasting settings, the hidden layer is typically built with a single layer of q nodes, and it serves as a link of both the input units and the output units. This is because the hidden layer contains no other information. In order to maximize the usefulness of the FFNN, it is critical to make an informed decision on the activation function used in the hidden layer. The following is an example of how the sigmoid function, which is a popular choice when it comes to activation functions:

$$f(x) = \frac{1}{1+\exp(-x)} \tag{2.24}$$

The following is a mathematical representation of the interaction between the units of input and output:

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j f\left(\beta_0 + \sum_{i=1}^{p} \beta_{ij} y_{t-i}\right) + \epsilon_t \tag{2.25}$$

The number of input nodes is represented by p, and the number of hidden nodes is represented by q. The weights of the network are denoted as α and $\beta \in \mathbb{R}^n$. There is no standard method for determining the optimal ratio of input-to-hidden nodes (p-to-q) in an artificial neural network (ANN). However, various search techniques can be used to identify the optimal hyperparameters.

2.6.2 Recurrent Neural Networks

Recurrent neural network (RNNs) is a form of neural network that was developed specifically for the purpose of analyzing serial data, like time series (Bandara et al., 2020). In most cases, RNNs are designed with a closed-circuit architecture that functions as short-term memory. As new information is obtained, people think about the temporal values of the past.

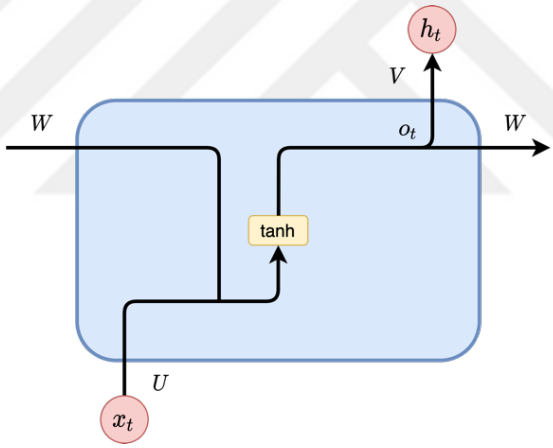


Figure 2.2 Recurrent neural network

The RNN architecture can be "unrolled" to form a series of ANNs over time, as shown in Figure 2.2. The input vector $x_t \in \mathbb{R}^m$ at time step t, the weights of the RNN represented by matrices $U \in \mathbb{R}^{m \times n}, W \in \mathbb{R}^{n \times n}$, and $V \in \mathbb{R}^{n \times k}$, and the hidden state $h_t \in \mathbb{R}^n$ and output $o_t \in \mathbb{R}^k$ can be described by the following equation:

$$\begin{aligned} h_t &= \sigma(Ux_t + Wh_{t-1}) \text{ and} \\ o_t &= \sigma(Vh_t) \end{aligned} \tag{2.26}$$

where σ is a nonlinear function, such as a sigmoid or tanh, and h_{t-1} can be expressed as h_{t-1}, can be set to 0 at the start. The error function is minimized during training using the back-propagation over time (Zipser & Williams, 1995) method and the real-time recurrent learning algorithm (Robinson & Fallside, 1987) is often used for training.

2.6.3 Long-short-term Memory

Long Short-Term Memory (LSTM) was developed by Hochreiter and Schmidhuber (1997) as a solution to the problems that typical recurrent neural networks (RNNs) have when it comes to dealing with long-term dependencies. The design of an LSTM is identical to that of a regular RNN; however, it incorporates input, forget, and output gates into its structure. This structure allows it to distinguish between relevant and irrelevant information and, as a result, to have better memory retention. It was discovered that an LSTM has a higher memory capacity than a normal RNN, although having the same number of parameters, and it can deal with prolonged data streams without requiring further adjustments. In addition, LSTM may be trained to require fewer epochs than traditional RNNs and is more resistant to noisy training data.

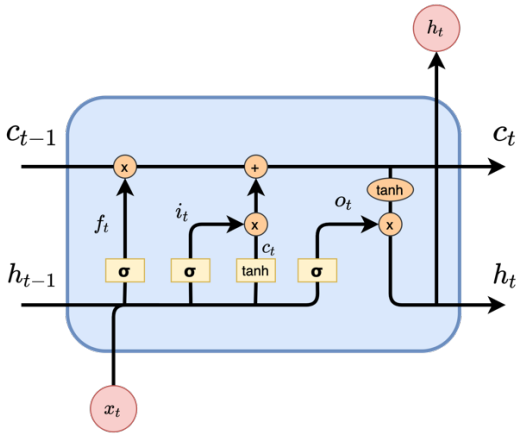


Figure 2.3 Long-short-term memory network

Figure 2.3 shows the structure of the LSTM. The input vector at step *t* is represented by $x_t \in \mathbb{R}^m$. Similarly, $C_t \in \mathbb{R}^n$ and $C_{t-1} \in \mathbb{R}^n$ and $h_t \in \mathbb{R}^n$ and $h_{t-1} \in \mathbb{R}^n$ represent the state cell, and the output vector at step *t*, respectively. According to this definition, m represents the dimension of the feature vector, and n represents both the size of the memory and the size of the output vector. The following equation can characterize the LSTM,

$$\begin{aligned} i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i), \\ o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o), \\ f_t &= \sigma\big(W_f h_{t-1} + U_f x_t + b_f\big), \\ C_t &= f_t \odot C_{t-1} + i_t \odot \phi(W_c h_{t-1} + U_c x_t + b_c), \text{ and} \\ h_t &= o_t \odot \phi(C_t), \end{aligned} \tag{2.27}$$

where $\big(W_i, W_o, W_f, W_c\big) \in \mathbb{R}^{n \times n}$ represents memory cell weights and the $\big(U_i, U_o, U_f, U_c\big) \in \mathbb{R}^{m \times n}$ shows input weights correspondingly. The biases of the gates mentioned earlier are also contained inside the vector $\big(b_i, b_o, b_f, b_c\big) \in \mathbb{R}^n$. At the end of the equation, the operator $\odot$ symbolizes the elementwise product, whereas σ and ϕ are the activation functions.

2.6.4 Gated Recurrent Unit

Chung et al. (2014) proposed first the GRU, which is the simpler variant of LSTM. Between LSTM and GRU, the most significant changes are the number of gates and the weights assigned to each gate. Since GRU has two gates, it cannot be used to regulate a memory unit in the network since it lacks an output gate. Because of the more straightforward structure of GRU, it is regarded to be more efficient than LSTM in computing. Due to the fact that it has fewer parameters than LSTM, it is also much easier to train. Because of its simplicity, it is also significantly faster than LSTM when learning from large sequences, which is advantageous when speed is critical in some situations.

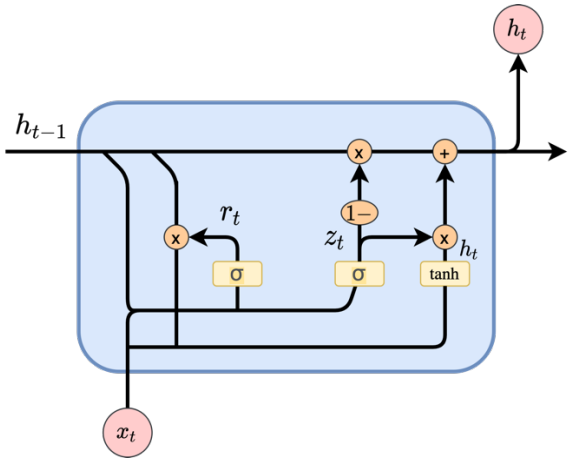


Figure 2.4 Gated Recurrent Unit network

Similarly to these other RNNs, Figure 2.4 depicts the GRU network, in which $x_t \in \mathbb{R}^m$ denotes data vector $h_t \in \mathbb{R}^n$ signifies the state vector, and $h_{t-1} \in \mathbb{R}^n$ represents the previous state vector. The model has the following organizational structure:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \text{ and} \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \end{aligned} \tag{2.28}$$

where the weights for the update, reset, and the matrices define state gates $(W_z, W_r, W_h) \in \mathbb{R}^{n \times n}$, respectively. The weights for x_t are represented by the matrices $(U_z, U_r, U_h) \in \mathbb{R}^{m \times n}$, respectively.

2.6.5 Temporal Convolutional Network

The temporal convolutional network (TCN) is a type of Convolutional Neural Network (CNN) specifically designed to handle temporal dependencies in sequence data. Like other CNNs, TCN can perform parallel computations, which allows it to address the challenging training problem of RNNs. Additionally, TCN follows two fundamental principles: (a) it uses causal convolution to prevent information leakage from the future to the past (Bai et al., 2018), and (b) it takes a variable-length sequence

and produces a fixed-size sequence using a 1-D fully convolutional network (Moosa & Vaz, 2015). TCN also uses a one-dimensional dilated convolution process to learn long-term dependencies in sequential data, allowing it to build deep networks without sacrificing resolution (F. Yu & Koltun, 2016). The TCN is composed of four main components: a 1-D fully convolutional network, dilated convolutions, causal convolution, and residual blocks (as shown in Figure 2.5).

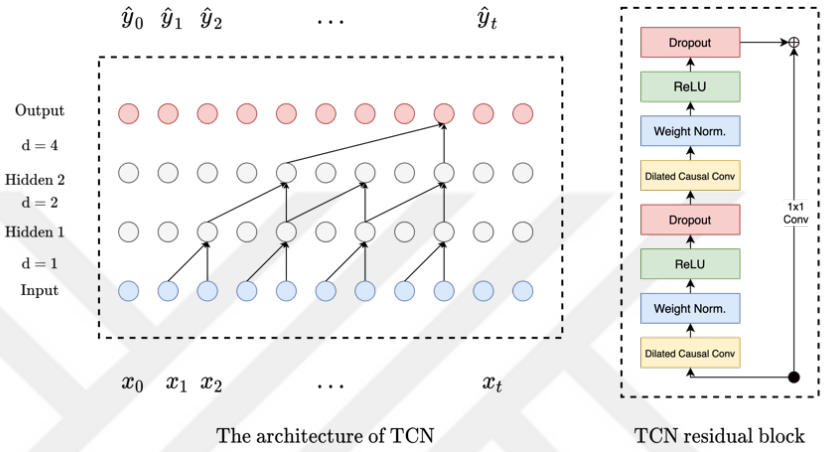


Figure 2.5 Temporal convolutional network

The 1-D fully TCN uses zero-padding to produce an output sequence of the same length as the input sequence, similar to the growing LSTM. At time step t, the input vectors are denoted by $x \in \mathbb{R}^n$, with x_{t-1} representing the input to the ℓ^{th} layer and x_t representing the output of the ℓ^{th} layer. The 1-D filters for the ℓ^{th} layer can be expressed as:

$$x_\ell^t = f\left(\sum_{i=1} x_\ell^{t-1} \times w_{i,\ell}^t + b_\ell^t\right) \tag{2.29}$$

where the weights of the ℓ^{th} layer at time step t are represented by $w_{i,\ell}^t$, the bias term is represented by b^t, and the rectified linear units (ReLU) activation function is represented by $f(.)$. The dilated convolution operator is critical for learning long-term dependencies in sequence data. It addresses the problem of receptive field size in

standard convolution without increasing the kernel size. Dilated convolution can be expressed as:

$$\mathrm{F}(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{2.30}$$

where $x \in \mathbb{R}^n$ represents the input vector, d is the dilation factor, k indicates the kernel size, $f: \{0,1,2, \dots, k-1\} \rightarrow \mathbb{N}$ represents the convolution kernel, $s - d.$ stands for the number of previous data points. A filter of size two is shown in Figure 2.5 as an illustration of dilations 1, 2, and 4.

Causal convolution ensures that the output of a filter $(y_o, \dots, y_t)$ at a given time, step t is only dependent on the previous time step input $(x_o, \dots, x_t)$. This structure prevents the information from leaking from the future into the past by minimizing the objective function $\mathcal{L}$, which estimates the discrepancies between the observed and predicted output. The residual temporal block is the final component of the TCN (K. He et al., 2016). It allows the TCN to be deepened without risking instability, which can occur when the depth of the TCN increases the receptive field size. The proposed residual block is divided into two sections, as shown in Figure 2.5. The input sequence $(x_o, \dots, x_t)$ is first modified using a series of networks $\mathcal{F}$ consisting of dilated causal convolution and ReLU to produce the output sequence $(y_o, \dots, y_t)$. Weight normalization and dropout are also applied after each layer at this point. The input sequence $(x_o, \dots, x_t)$ is then projected directly through the network in the second section. The output of the residual block can therefore be defined as:

$$o = \phi(x \oplus \mathcal{F}(x)) \tag{2.31}$$

where ϕ is the activation function of the network.

CHAPTER 3
PROPOSED FRAMEWORK AND EXPERIMENTAL SETUP

3.1 Experimental Framework

The experimental procedure outlined in Figure 3.1 consists of three steps. In the first phase, the first logarithmic difference in stock prices and log returns are modeled, and the data is pre-processed. The entire series is then split into three subsets: a training sample (80% of the total), a validation sample (10%), and a test sample (10%).

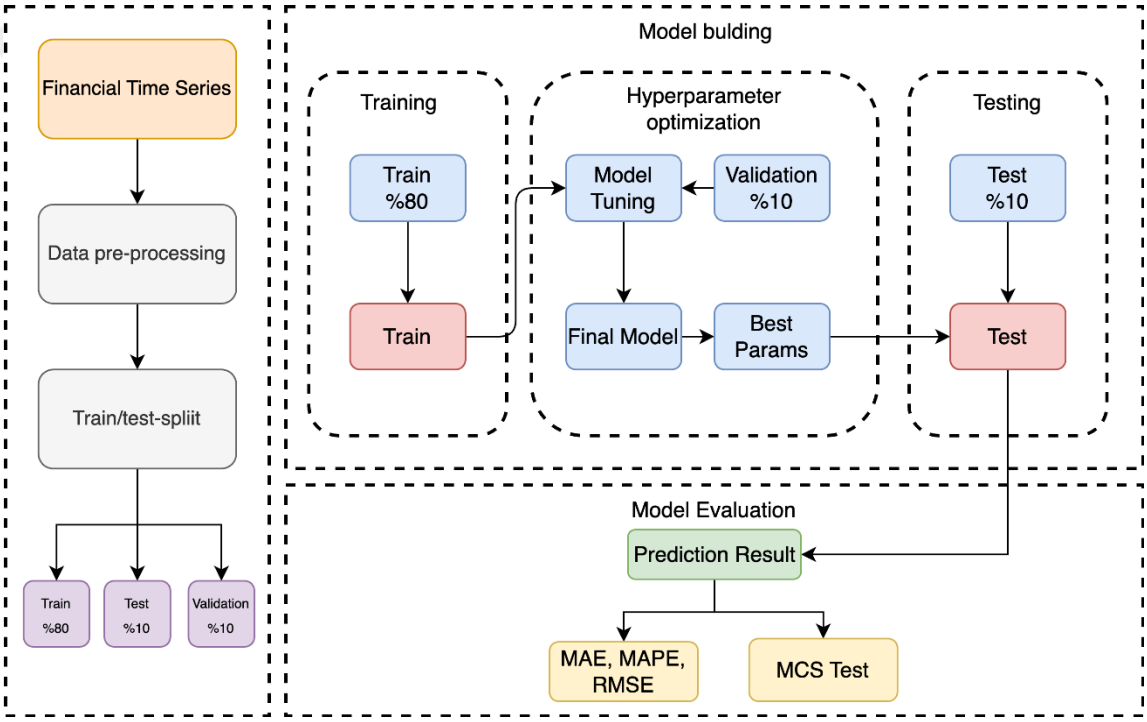


Figure 3. 1 Experimental framework

In the second phase, Bayesian optimization is used to find the optimal hyperparameters for the models in the hyperparameter space and to refine the models. After the optimal parameters are identified, the models are trained to make predictions. In order to determine the appropriate lag order for each model, Bayesian optimization is also used based on the MSE criteria. In the third step, the performance of the models is evaluated in terms of forecasting accuracy using three statistical evaluation

techniques. The performance indicators are MAE, MAPE, and RMSE, as well as the MCS test to identify which model(s) perform best with statistical significance compared to the others in the database. All calculations were completed using R 4.0 and Python using TensorFlow (Abadi et al., 2016), Keras (Chollet & others, 2015), Keras Tuner (O'Malley et al., 2017) libraries, and MCS (Bernardi & Catania, 2018) package.

3.2 Dataset

The data on stock prices used in this thesis were selected because of their great liquidity and volatility. The data was gathered from Yahoo Finance and included the Bombay Stock Exchange (BSE), the S&P 500, the Dow Jones Industrial Average (DJI), and the Taiwan Delayed Price (TWII). The daily stock price data covered the period from March 3, 2015, to November 1, 2022 (see Figure 3.2).

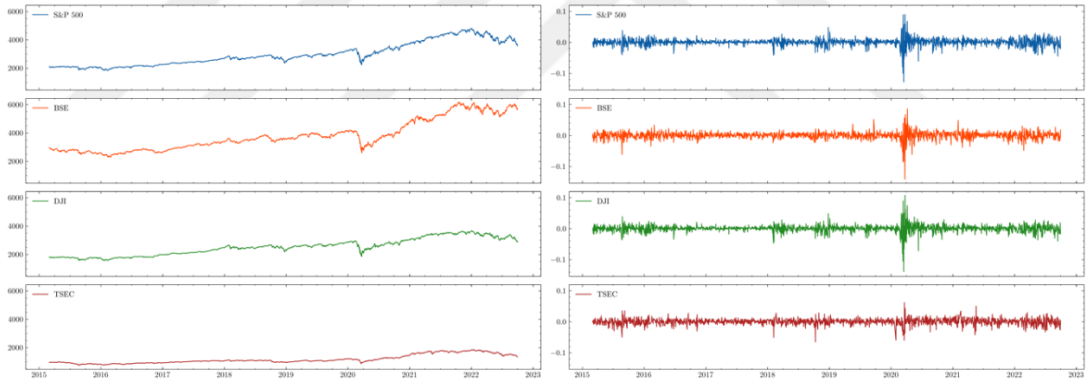


Figure 3.2 Original and log returns of four stock data

Both the first logarithmic difference in stock prices and the log returns were modeled and analyzed. A training sample of 1508 observations, covering the period from March 3, 2015, to April 5, 2021, was set aside for use in training the models, representing approximately 80% of the total sample. The remaining data, representing approximately 20% of the whole sample, was used to evaluate the performance of the model using more advanced statistical models.

Table 3.1 Summary statistics of stock returns

	Mean	Median	Std. Dev.	Skewness	Kurtosis	JB	ADF
BSE	0.000369	0.000519	0.0115	-1.45	21.32	32313.94	-3.43
S&P 500	0.000308	0.000680	0.0118	-1.14	18.09	23193.60	-3.43
DJI	0.000269	0.000568	0.0119	-1.26	23.69	39606.71	-3.43
TSEC	0.000187	0.000741	0.0100	-0.66	5.65	2349.51	-3.43

The results of the descriptive statistics, Jarque-Bera (JB) normality test, and augmented Dickey-Fuller (ADF) unit root test on the experimental data are reported in Table 3.1. At the 1% level of significance, the JB normality test rejected the null hypothesis for each of the four stock prices, and the Shapiro-Wilk test also produced similar results. The results of the ADF unit root tests indicated that none of the stock returns are moving forward or backward in time, suggesting that they can be represented using the proposed approach without requiring further transformations.

3.3 Hyperparameter Tuning

Hyperparameter tuning is the process of finding the ideal model architecture by adjusting the hyperparameters, which define the model architecture but cannot be learned directly from data. Two primary methods for hyperparameter tuning are manual searching and automatic searching. Manual searching is inefficient and time-consuming due to many parameters, while automatic searching methods such as grid search and random search can help overcome these limitations. Bayesian optimization is a technique that uses the Bayesian formula to optimize the performance of an unknown function by finding the optimal values of its prior and posterior distributions (Snoek et al., 2012). The optimization task is formulated as finding the maximum value of the objective function f, with the hyperparameter space Θ being searched for the optimal values θ^*. The process of finding the best parameters θ^* through Bayesian optimization consists of two parts: first, the prior distribution of the objective function f is used to determine a new observation point based on a Gaussian process model, and then the posterior distribution of $f(\theta)$ is calculated by updating the covariance matrix of the Gaussian process based on the function f. The acquisition function U,

which uses upper confidence bounds, is used to identify the maximum of the function f and decide whether to use the current optimal value or search for an alternate low-confidence zone. The constant β_{t+1} can be used to balance these two options. More precisely, the optimization task that seeks to maximize the objective function f can be defined as:

$$\theta^* = \arg\max_{\theta \in \Theta} f(\theta) \tag{3.1}$$

The process of finding optimal parameters θ^* in hyperparameter space, Θ through Bayesian optimization consists of two stages (Wang et al., 2019). The first phase involves using a prior distribution of the objective function f to determine the new observation point $D_{t+1} \in D = \{x_n y_n\}_{n=1}^{t}$. A Gaussian process, often used to model the prior distribution of f, can be used to describe the objective function. In the case of the objective function f, the Gaussian process can be described as

$$f(\theta) \sim GP\big(\mu(\theta), K(\theta, \theta')\big) \tag{3.2}$$

In the second phase, the posterior distribution of $f(\theta)$ is calculated using the mean function $\mu: \theta \to \mathbb{R}$ and covariance function $K: \theta \times \theta \to \mathbb{R}$ of $f(\theta)$. This is done by assuming that the values of f are drawn from a multivariate normal distribution $f \sim N(0, K)$, in which case

$$K = \begin{bmatrix} k(\theta_1\theta_1) & \cdots & k(\theta_1\theta_t) \\ \vdots & \ddots & \vdots \\ k(\theta_t\theta_1) & \cdots & k(\theta_t\theta_t) \end{bmatrix} \tag{3.3}$$

Based on the function f, the covariance matrix of the Gaussian process is updated to

$$K' = \begin{pmatrix} K & k^T \\ k & k(\theta_{t+1}\theta_{t+1}) \end{pmatrix} \tag{3.4}$$

where $k = [k(\theta_{t+1}\theta_1), \ldots, k(\theta_{t+1}\theta_t)]$. In order to build the posterior probability distribution, it is necessary to first acquire the updated covariance matrix of f, as follows

$$P\ (f_{t+1} \mid D_{t+1}, \theta_{t+1}) \sim N(\mu_{t+1}(\theta), \sigma^2_{t+1}(\theta)) \tag{3.5}$$

In this equation, $\mu_{t+1}(\theta)$ and $\sigma^2_{t+1}(\theta)$ represent the mean and variance of the function $f(\theta)$. Furthermore, Bayesian optimization employs a heuristic process known as the acquisition function U to identify the maximum of a function, which is denoted by f (Chu & Ghahramani, 2005). To optimize the following acquisition function, we will employ upper confidence bounds in the current investigation

$$\theta_{t+1} = \arg\max_{\theta \in \Theta} U(\theta \mid D_t) = \arg\max_{\theta \in \Theta} \mu(\theta) + \beta^{1/2}_{t+1}\sigma_t(\theta) \tag{3.6}$$

in which case the function U defines whether the next sample point should use the current optimum value or seek an alternate low-confidence zone. Using the constant β_{t+1}, we may try to trade between the two options.

3.4 Model Evaluation

In the related literature, various performance measures are often used to evaluate and compare the forecasting capabilities of different models. The MAE, MAPE, RMSE, and MCS tests were all employed in this study to assess the accuracy of the predictions made by the models being compared. These metrics provide a quantifiable way to determine how well the models were able to forecast the outputs, allowing for a more objective comparison of their performance. By using a combination of these measures, we got a more comprehensive understanding of the strengths and weaknesses of each model and ultimately determined which one was the most effective at predicting the outcomes we were interested in.

3.4.1 Performance Measures

In the literature, various performance measures are used to compare the forecasting abilities to compete models. The following evaluation criteria were used: MAE, MAPE, and RMSE

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.7}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3.8}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3.9}$$

in which n denotes the number of observations, as well as y_i and $\hat{y}_i$ denote the observed and predicted values.

3.4.2 Model Confidence Set

Utilizing performance metrics to assess the prediction power of rival models is a very arbitrary process that frequently lacks proof that variations in forecast accuracy are statistically significant. However, reliable evaluation methods are available to objectively quantify one predictive model's superiority over another. The model confidence set, developed by Hansen et al. (2011), is an example of a reliable technique that uses statistical methods to measure the differences in predictive accuracy among rival models. It allows for many "best models" to be generated without the need for a "benchmark" model, and it provides information on the level of relevance that other approaches do not. Given a set $\mathcal{M} = \{m_i\}_{i=1}^{n}$ of finite competing models and a loss function $\mathcal{L}$ used to evaluate these models over test sample data $\mathcal{S} = \{x_n, y_n\}_{n=1}^{t}$, the loss function $\mathcal{L}$ can be applied to a model $m \in \mathcal{M}$ at time step t as $\mathcal{L}_{m,t}$. Consider the case of a point forecast $\hat{y}_{m,t}$ of y_t that is evaluated using a loss

function $\mathcal{L}$. After that, we can define $\mathcal{L}_{m,t} = \mathcal{L}(y_t, \hat{y}_{m,t})$. The relative performance of models is $d_{m,k,t} \overset{\text{def}}{=} \mathcal{L}_{m,t} - \mathcal{L}_{k,t}, \forall m, k \in \mathcal{M}$.

The set of superior rival models is described by

$$\mathcal{M}^* \overset{\text{def}}{=} \{m \in \mathcal{M}: \mathbb{E}[d_{m,k,t}] \leq 0, \forall k \in \mathcal{M}\} \tag{3.10}$$

where $\mathbb{E}[d_{m,k,t}]$ represents the expectation $d_{m,k,t}$ when subjected to an objective function $\mathcal{L}$. The MCS test technique will perform numerous tests to determine whether or not the null hypothesis, which states that

$$H_{0,M}: \mathbb{E}[d_{m,k,t}] = 0 \ \forall m, k \in \mathcal{M} \tag{3.11}$$

Sequential significance tests are included in the MCS test; these include the equivalency test ($\delta_{\mathcal{M}}$) and the elimination rule ($e_{\mathcal{M}}$). The, $\delta_{\mathcal{M}}$ checks the, H_0 hypothesis for any $m \in \mathcal{M}$, and the, $e_{\mathcal{M}}$ specifies the $m \in \mathcal{M}$ to be eliminated from the trials in order to have the, H_0 hypothesis rejected. When these procedures are complete, we get a set, $\mathcal{M}^*_{1-\alpha}$, called "survivors," which contains all of the competing models that have "survived" without being eliminated. According to this definition, the 'surviving model' is the p-value with a smaller than the significance level, denoted by α. In order to put the primary null hypothesis to the test, we define two statistics as follows:

$$T_R = \underset{m,k \in \mathcal{M}}{\operatorname{argmin}} \frac{|\bar{d}_{mk}|}{\sqrt{\hat{\mathbb{V}}(d_{mk})}}, \ T_{\max} = \underset{m,k \in \mathcal{M}}{\operatorname{argmin}} \frac{\bar{d}_{mk}}{\sqrt{\hat{\mathbb{V}}(d_{mk})}} \tag{3.12}$$

It is not possible to reject the null hypothesis H_0 when the *p*-values of both of the statistics are greater than the significance level of α. The MCS test implements a block bootstrap technique with the default parameters of 1000 samples and utilizes MSE as the loss function $\mathcal{L}$. Additionally, it uses $\alpha = 0.10$ as the significance level for the loss function $\mathcal{L}$.

CHAPTER 4
EMPIRICAL RESULT AND CONCLUSION

4.1 Empirical Results

To evaluate the effectiveness of our models in predicting stock returns, we used two different approaches: performance accuracy measures and the MCS test. The performance accuracy measures, such as MAE, MAPE, and RMSE, provided a quantitative assessment of how accurately the models predicted the outputs. On the other hand, the MCS test offered an estimation of the ability of the models to capture important patterns in their predictions. After evaluating our models on four active trading markets, the results in Table 4.1 showed that the TCN model seemed to be the best predictor in all the scenarios we analyzed. In terms of MAE and MAPE performance measures, it appears that the GRU model slightly outperforms other neural network models in DJI and TSEC stock returns when forecasting the future. such as linear, lasso, and ridge regression, performed significantly better than the other machine learning models when compared to models based on deep neural networks in terms of prediction accuracy.

Based on the results of our model comparisons, we can conclude that the TCN model has superior out-of-sample prediction performance when attempting to predict stock returns. This result can be attributed to its ability to capture time-varying features, as this type of model is more adept at identifying patterns across multiple time periods and distinguishing them from one-time occurrences. This result is true even when compared to the models used to evaluate the other models. In that order, it is followed by the LSTM, GRU, RNN, MLR, Ridge, and Lasso, all of which have relatively good accuracy to a certain extent. It is reasonable to say that the aforementioned benefits demonstrate that TCN has the potential to evolve into a new quantitative tool for financial forecasting. However, this outcome should not be taken as conclusive evidence that the model in question is the superior option. We used the MCS test to determine which neural network architecture offers the highest statistical significance level.

Table 4. 1 Forecasting accuracy measurements for all models

	BSE			S&P 500			DJI			TSEC		
Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Linear	0.037	0.067	1.477	0.030	0.053	1.518	0.029	0.057	1.547	0.029	0.042	1.645
Ridge	0.037	0.067	1.477	0.030	0.053	1.519	0.029	0.057	1.546	0.029	0.042	1.645
Lasso	0.057	0.092	0.997	0.069	0.115	1.251	0.064	0.109	1.178	0.065	0.091	1.135
Elastic-Net	0.043	0.075	1.112	0.069	0.115	1.251	0.040	0.074	1.085	0.065	0.091	1.133
KNN	0.056	0.093	1.536	0.050	0.084	1.833	0.047	0.084	1.921	0.047	0.067	1.775
DT	0.066	0.108	2.316	0.063	0.103	3.559	0.060	0.104	3.914	0.057	0.078	3.208
RF	0.053	0.089	1.604	0.048	0.081	1.777	0.046	0.085	1.718	0.045	0.064	1.767
GB	0.068	0.104	1.412	0.059	0.097	1.729	0.058	0.099	1.855	0.055	0.077	1.425
LightGBM	0.054	0.089	2.206	0.050	0.082	2.339	0.048	0.085	2.801	0.045	0.063	2.047
XGBoost	0.081	0.119	2.472	0.078	0.117	2.990	0.084	0.127	3.916	0.065	0.091	1.578
MLP	0.078	0.118	1.051	0.069	0.115	1.251	0.071	0.120	1.313	0.065	0.091	1.134
RNN	0.078	0.118	1.051	0.069	0.115	1.251	0.071	0.120	1.313	0.065	0.091	1.134
LSTM	0.037	0.067	1.477	0.030	0.053	1.537	0.029	0.057	1.547	0.029	0.042	1.645
GRU	0.037	0.067	1.477	0.030	0.053	1.518	0.029	0.057	1.541	0.029	0.042	1.645
TCN	0.036	0.066	1.405	0.029	0.052	1.411	0.027	0.054	1.546	0.028	0.042	1.683

To ensure that our conclusion was reliable, we employed the MCS, a more advanced evaluation tool. We determined that a model with a *p*-value greater than 0.10 in the MCS test and its corresponding model had a higher ability to predict outcomes than a model with a *p*-value less than 0.10. This setup led us to set a threshold of 0.10. In this thesis study, we defined "survival models" as those with *p*-values greater than 0.10 and "inferior models" as those with p-values less than 0.10. The performance of survival models was significantly superior to that of inferior models. The MCS test ultimately resulted in the removal of models that were deemed lower quality than the others. The MCS was designed in such a way that the determination of what constitutes the "best" model is based on the conventional understanding of "significance." Although the MCS typically only includes the best model or models, it is possible for it to include multiple inferior models in finite samples. A vital principle of the MCS method is that a model is not eliminated from consideration unless it is significantly inferior to another model. Not all models in the MCS may be considered high quality, as they will remain in the MCS unless proven to be lower quality than other models in

the MCS (Hansen et al., 2011). The *p*-values for the MCS test, based on the MSE criterion and two different test statistics, are shown in Table 4.2. Table 4.2 shows that the TCN model was the best predictor in every case we examined, using the criterion that a model is only discarded if it is significantly inferior to another model.

Additionally, we can conclude from the TSEC stock market that the linear, ridge, lasso, MLP, RNN, and LSTM models are survival models because they had a *p*-value greater than 0.10 in terms of T_{max} statistics. This is because the TSEC stock market is less volatile than other stock markets, which improves the models' ability to generalize. The stability of the stock market allows for better training and test set partitioning compared to more volatile markets, resulting in improved generalization performance. However, TCN models outperformed the other models in terms of prediction, even in highly volatile markets, suggesting that the TCN model has strong generalization capabilities and can adapt and perform well in markets with high levels of volatility. This result is further supported by Figure 4.2, which displays actual and predicted values of the TCN model for each stock return for various periods.

Table 4. 2 Model confidence set result of the models

	BSE		S&P 500		DJI		TSEC	
Models	T_R	T_{MAX}	T_R	T_{MAX}	T_R	T_{MAX}	T_R	T_{MAX}
Linear	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2432
Ridge	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2198
Lasso	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2992
Elastic-Net	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
KNN	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
DT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
RF	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
GB	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
LightGBM	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
XGBoost	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
MLP	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3980
RNN	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2482
LSTM	0.0722	0.0059	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
GRU	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000
TCN	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Overall, our research has demonstrated that the TCN model consistently outperforms other models in predicting stock returns for all four sample markets. This was confirmed through statistical evaluation techniques like MAE, MAPE, and RMSE, and the MCS test, which showed that the TCN model was the only one with a p-value high enough to be considered a "survival model" for high-volatility markets. These findings align with previous studies on the topic, further solidifying the TCN model's position as the most reliable deep-learning model for forecasting stock returns in volatile markets. It is clear that the TCN model's ability to capture time-varying features and recognize patterns across multiple time periods gives it a significant advantage in these scenarios. These results have important implications for financial forecasting and suggest that the TCN model could be a valuable tool for professionals in the field.

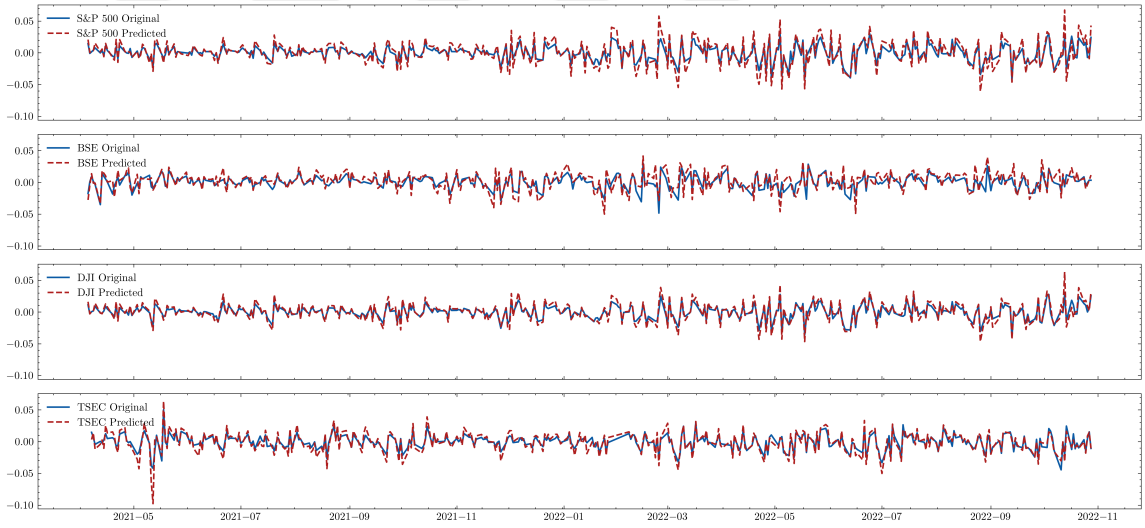


Figure 4.1 Forecasting results for the TCN model

4.2 Conclusion

Throughout this thesis study, the primary focus was evaluating the forecasting performance of advanced machine learning and deep learning models using statistical methods. To predict the prices of four major stock indices, we used various competing models and employed the Bayesian optimization technique for an extensive hyperparameter search to find the most accurate hyperparameters for the models. Our

findings from the preliminary experiments showed that when comparing the TCN model to linear-based, tree-based, and recurrent-based models, the TCN model consistently outperformed the others in terms of effectiveness and accuracy across all four datasets. The TCN structure, which includes a causal dilated convolution cell, allows the model to capture long-term dependencies present in recurrent-based networks accurately. Additionally, the dilated convolution structure enables a deeper analysis of the time series by increasing the size of the receptive field and allowing for a more efficient parameterization. These characteristics make the TCN model a promising tool for financial time-series forecasting tasks.

The TCN model's exceptional performance in predicting financial time series was confirmed through the use of the MCS test. The test examined the MSE criteria and the T_{max} and T_R statistics, all of which indicated that the TCN model was the most effective. In the future, researchers should focus on improving the predictive abilities of TCN models by studying their block ensembles. Additionally, the TCN model's performance can be further analyzed through the use of various forecasting methods, such as multi-horizon forecasting, and by employing cross-validation techniques. Likely, that the TCN model will also be utilized in future research to address time-series classification issues. Overall, the TCN model has demonstrated its effectiveness as a powerful tool in the realm of financial time-series forecasting.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J. et.al. (2016). Tensorflow: a system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 16, 265–283.

Abdullah, M. H. L., & Ganapathy, V. (2000). Neural network ensemble for financial trend prediction. *IEEE Region 10 Annual International Conference*, *3,* 157-161. https://doi.org/10.1109/tencon.2000.892242

Abhyankar, A., Copeland, L. S., & Wong, W. (1997). Uncovering nonlinear structure in real-time stock-market indexes: The S & P 500, the DAX, the Nikkei 225, and the FTSE-100. *Journal of Business and Economic Statistics*, *15*(1), 1–14. https://doi.org/10.1080/07350015.1997.10524681

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. *Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation,* 106–112. https://doi.org/10.1109/UKSim.2014.67

Andersen, T. G., Bollerslev, T., & Meddahi, N. (2011). Realized volatility forecasting and market microstructure noise. *Journal of Econometrics*, *160*(1), 220–234. https://doi.org/10.1016/j.jeconom.2010.03.032

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv*, *abs/1803.0*.

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, *140*.

https://doi.org/10.1016/j.eswa.2019.112896

Başoğlu Kabran, F., & Demirberk Ünlü, K. (2020). A two-step machine learning approach to predict S&P 500 bubbles. *Journal of Applied Statistics*, 48, 13-15. https://doi.org/10.1080/02664763.2020.1823947

Bernardi, M., & Catania, L. (2018). The model confidence set package for R. *International Journal of Computational Economics and Econometrics*, *8*(2), 144–158.

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2018). Dilated convolutional neural networks for time series forecasting. *Journal of Computational Finance*. https://doi.org/10.21314/jcf.2019.358

Brandt, M. W., & Jones, C. S. (2006). Volatility forecasting with range-based EGARCH models. *Journal of Business and Economic Statistics*, *24*(4), 470–486. https://doi.org/10.1198/073500106000000206

Breiman, L. (2001). Random Forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). Classification and regression trees. In *Classification and Regression Trees*. https://doi.org/10.1201/9781315139470

Chen, A. S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index. *Computers and Operations Research*, *30*(6), 901–923. https://doi.org/10.1016/S0305-0548(02)00037-0

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, Y., Kang, Y., Chen, Y., & Wang, Z. (2020). Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, *399*, 491–501. https://doi.org/https://doi.org/10.1016/j.neucom.2020.03.011

Chollet, F., & others. (2015). *Keras*. GitHub. https://github.com/fchollet/keras

Chu, W., & Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, *6(7)*, 1019–1041. https://doi.org/10.1145/1102351.1102369

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *ArXiv*. http://arxiv.org/abs/1412.3555

Committee, & Prize, N. (2013). Understanding Asset Prices. *Nobel Prize in Economics Documents*. https://ideas.repec.org/p/ris/nobelp/2013_001.html

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Dang Khoa, N. L., Sakakibara, K., & Nishikawa, I. (2006). Stock price forecasting using back propagation neural networks with time and profit based adjusted weight factors. *2006 SICE-ICASE International Joint Conference*, 5484–5488. https://doi.org/10.1109/SICE.2006.315683

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2016). Language Modeling with Gated Convolutional Networks. *CoRR*, *abs/1612.0*. http://arxiv.org/abs/1612.08083

Elith, J., Leathwick, J. R., & Hastie, T. (2008). A working guide to boosted regression

trees. In *Journal of Animal Ecology*, 77(4), 802–813. https://doi.org/10.1111/j.1365-2656.2008.01390.x

Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, *38*(1), 34–105.

Fama, E. F. (1991). Efficient capital markets: ii. *The Journal of Finance*, *46*(5), 1575. https://doi.org/10.2307/2328565

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Hansen, P. R., Lunde, A., & Nason, J. M. (2011). The Model Confidence Set. *Econometrica*, *79*(2), 453–497. https://doi.org/10.3982/ecta5771

Hanson, J., & Raginsky, M. (2019). Universal Approximation of Input-Output Maps by Temporal Convolutional Nets. *ArXiv*, *abs/1906.0*. http://arxiv.org/abs/1906.09211

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 770–778. https://doi.org/10.1109/CVPR.2016.90

He, Q., Kamarianakis, Y., Jintanakul, K., & Wynter, L. (2013). *Incident duration prediction with hybrid tree-based quantile regression* (pp. 287–305). Springer. https://doi.org/10.1007/978-1-4614-6243-9_12

Hiransha, M., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2018). NSE stock market prediction using deep-learning models. *Procedia Computer Science*, *132*(Iccids), 1351–1362. https://doi.org/10.1016/j.procs.2018.05.050

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoerl, A. E., & Kennard, R. W. (2000). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *42*(1), 80–86. https://doi.org/10.1080/00401706.2000.10485983

Hornik, K. (1991). Approximation capabilities of multilayer networks. *Neural Networks*, *4*(2), 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

Hosaka, T. (2019). Bankruptcy prediction using imaged financial ratios and convolutional neural networks. *Expert Systems with Applications*, *117*, 287–299. https://doi.org/10.1016/j.eswa.2018.09.039

Japkowicz, N., & Shah, M. (2011). Evaluating learning algorithms: A classification perspective. In *Evaluating Learning Algorithms: A Classification Perspective* (Vol. 9780521196). Cambridge University Press. https://doi.org/10.1017/CBO9780511921803

Jayawardena, N. I., Todorova, N., Li, B., & Su, J. J. (2020). Volatility forecasting using related markets' information for the Tokyo stock exchange. *Economic Modelling*, *90*, 143–158. https://doi.org/10.1016/j.econmod.2020.05.008

Jiang, Q., Tang, C., Chen, C., Wang, X., & Huang, Q. (2019). Stock Price Forecast Based on LSTM Neural Network. In J. Xu, F. L. Cooke, M. Gen, & S. E. Ahmed (Eds.), *Proceedings of the Twelfth International Conference on Management Science and Engineering Management* (pp. 393–408). Springer International Publishing.

Karaoglu, S., Arpaci, U., & Ayvaz, S. (2017). A deep learning approach for optimization of systematic signal detection in financial trading systems with big data. *International Journal of Intelligent Systems and Applications in*

Engineering, *Special Issue*, 31–36. https://doi.org/10.18201/ijisae.2017SpecialIssue31421

Kim, S., Ku, S., Chang, W., Chang, W., Chang, W., & Song, J. W. (2020). Predicting the direction of us stock prices using effective transfer entropy and machine learning techniques. *IEEE Access*, *8*, 111660–111682. https://doi.org/10.1109/ACCESS.2020.3002174

Kumar, G., Singh, U. P., & Jain, S. (2021). Swarm Intelligence Based Hybrid Neural Network Approach for Stock Price Forecasting. *Computational Economics*. https://doi.org/10.1007/s10614-021-10176-9

Lara-Benítez, P., Carranza-García, M., Luna-Romera, J. M., & Riquelme, J. C. (2020). Temporal convolutional networks applied to energy-related time series forecasting. *Applied Sciences*, *10*(7), 1–17. https://doi.org/10.3390/app10072322

Malkiel, B. G. (2003). The efficient market hypothesis and its. *Journal of Economic Perspectives*, *17*(1), 59–82. https://doi.org/10.1257/089533003321164958

Moosa, I., & Vaz, J. (2015). Why is it so difficult to outperform the random walk? An application of the Meese–Rogoff puzzle to stock prices. *Applied Economics*, *47*(4), 398–407. https://doi.org/10.1080/00036846.2014.972545

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., & Invernizzi, L. (2017). *Keras Tuner*. GitHub.

Oh, K. J., & Kim, K. J. (2002). Analyzing stock market tick data using piecewise nonlinear model. *Expert Systems with Applications*, *22*(3), 249–255. https://doi.org/10.1016/S0957-4174(01)00058-6

Ozbayoglu, A. M., Gudelek, M. U., & Sezer, O. B. (2020). Deep learning for financial applications : A survey. *Applied Soft Computing*, *93*, 106384.

https://doi.org/10.1016/j.asoc.2020.106384

Panas, E. (2001). Estimating fractal dimension using stable distributions and exploring long memory through ARFIMA models in Athens Stock Exchange. *Applied Financial Economics*, *11*(4), 395–402. https://doi.org/10.1080/096031001300313956

Parray, I. R., Khurana, S. S., Kumar, M., & Altalbe, A. A. (2020). Time series data analysis of stock price movement using machine learning techniques. *Soft Computing*, *0123456789*. https://doi.org/10.1007/s00500-020-04957-x

Pérez-Rodríguez, J. V., Torra, S., & Andrada-Félix, J. (2005). STAR and ANN models: Forecasting performance on the Spanish "Ibex-35" stock index. *Journal of Empirical Finance*, *12*(3), 490–509. https://doi.org/10.1016/j.jempfin.2004.03.001

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106. https://doi.org/10.1023/A:1022643204877

Quinlan, J. R. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, *4*, 77–90. https://doi.org/10.1613/jair.279

Robinson, A. J., & Fallside, F. (1987). The utility driven dynamic error propagation network. *IEEE*. https://books.google.com.tr/books/about/The_Utility_Driven_Dynamic_Error_Propaga.html?id=6JYYMwEACAAJ&redir_esc=y

Salzberg, S. L. (1994). C4.5: Programs for machine learning. *Machine Learning*, *16*(3), 235–240. https://doi.org/10.1007/bf00993309

Sarantis, N. (2001). Nonlinearities, cyclical behaviour and predictability in stock markets: International evidence. *International Journal of Forecasting*, *17*(3),

459–482. https://doi.org/10.1016/S0169-2070(01)00093-0

Schwartz, R. A. (1970). Efficient capital markets: A review of theory and empirical work: discussion. *The Journal of Finance*, *25*(2), 421. https://doi.org/10.2307/2325488

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics,* 1643–1647. https://doi.org/10.1109/ICACCI.2017.8126078

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 Journal*, *90*, 106181. https://doi.org/10.1016/j.asoc.2020.106181

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep Learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science*, *131*, 895–903. https://doi.org/https://doi.org/10.1016/j.procs.2018.04.298

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, *4*, 2951–2959.

Takens, F. (1981). *Detecting strange attractors in turbulence* (pp. 366–381). Springer, Berlin, Heidelberg. https://doi.org/10.1007/bfb0091924

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective. In Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(3), 273-282.

Tibshiranit, R. (1996). Regression Shrinkage and Selection via the Lasso. I In Journal of the Royal Statistical Society: Series B (Statistical Methodology), 58(1), 267-288.

Timmermann, A., & Granger, C. W. J. (2004). Efficient market hypothesis and forecasting. *International Journal of Forecasting*, *20*(1), 15–27. https://doi.org/10.1016/S0169-2070(03)00012-8

Tran, D. T., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2018). Tensor representation in high-frequency financial data for price change prediction. *2017 IEEE Symposium Series on Computational Intelligence,* 1–7. https://doi.org/10.1109/SSCI.2017.8280812

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. *19th Conference on Business Informatics, 01*, 7–12. https://doi.org/10.1109/CBI.2017.23

Wan, R., Mei, S., Wang, J., Liu, M., & Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, *8*(8). https://doi.org/10.3390/electronics8080876

Wang, Y., Liu, Z., Hu, D., & Zhang, M. (2019). Multivariate time series prediction based on optimized temporal convolutional networks with stacked auto-encoders. *Proceedings of Machine Learning Research*, *101*, 157–172.

Yilmaz, F. M., & Yildiztepe, E. (2022). Statistical Evaluation of Deep Learning Models for Stock Return Forecasting. *Computational Economics*, 1-24. https://doi.org/10.1007/s10614-022-10338-3

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for Natural Language Processing. *ArXiv*, *abs/1702.0*.

http://arxiv.org/abs/1702.01923

Yoon, Y., & Swales, G. (1991). Predicting stock price performance: A neural network approach. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *4*, 156–162. https://doi.org/10.1109/HICSS.1991.184055

Yu, F., & Koltun, V. (2016). Multi-Scale Context Aggregation by Dilated Convolutions. *ArXiv*, *abs/1511.0*.

Yu, P., & Yan, X. (2020). Stock price prediction based on deep neural networks. *Neural Computing and Applications*, *32*(6), 1609–1628. https://doi.org/10.1007/s00521-019-04212-x

Zhang, L., Aggarwal, C., & Qi, G. J. (2017). Stock price prediction via discovering multi-frequency trading patterns. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F129685*, 2141–2149. https://doi.org/10.1145/3097983.3098117

Ziegel, E. R. (2003). The elements of statistical learning. *Technometrics*, *45*(3), 267–268. https://doi.org/10.1198/tech.2003.s770

Zipser, D., & Williams, R. J. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. *Back-Propagation: Theory, Architectures and Applications*, 433–486. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.5418