DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

STOCK MARKET PREDICTION USING MACHINE LEARNING MODELS

by
Atakan SİTE

August, 2020
İZMİR

STOCK MARKET PREDICTION USING MACHINE LEARNING MODELS

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 Computer Engineering

by
Atakan SİTE

August, 2020
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"STOCK MARKET PREDICTION USING MACHINE LEARNING MODELS"** completed by **ATAKAN SİTE** under supervision of **ASST. PROF. DR. ZERRİN IŞIK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Zerrin IŞIK

Supervisor

Asst. Prof. Dr. Mustafa ÖZUYSAL

(Jury Member)

Assoc. Prof. Dr. Derya BİRANT

(Jury Member)

Prof. Dr. Özgür ÖZÇELİK
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

First of all, I would like to thank my academic advisor, Asst. Prof. Dr. Zerrin IŞIK, for her guidance and useful advices that always show me the right direction in my research. During my research, she contributed to a rewarding graduate school experience by giving me intellectual freedom in my study, supporting my attendance at various conferences, and demanding a high quality work in all my efforts.

I would also like to thank my parents, *Yasemin* and *Mehmet*, who supported me with love and understanding. Without you, I could never have reached this current level of success.

Atakan SİTE

STOCK MARKET PREDICTION USING MACHINE LEARNING MODELS

ABSTRACT

Forecasting on stock market data is a challenging task due to dynamic and complex nature of the data in this field. Here, we present a comprehensive analysis on effects of using different data sources and deep learning architectures for forecasting stock prices. One of the biggest factors in the performance of deep learning models is effective representation of data. In this study, sentiment analysis was conducted on news articles that may cause financial external effects on stock markets, and various technical indicators extracted from the stock price datasets. Other than data representation, the performance of deep learning algorithms is dependent on hyperparameters, thus, we optimized various hyperparameters of the developed deep learning models using automated Bayesian optimization.

We developed optimal Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Stacked Autoencoder (SAE) models for forecasting daily close prices from ten different historical stock price datasets traded on NASDAQ and NYSE target markets. Our study covers the effects of sentiment analysis on stock market forecasting and assessment of the proposed deep learning models in complex data representation. The evaluation of forecasting is analyzed with several error metrics. The obtained results revealed that the developed CNN model provides better performance than other ones. The integration of sentiment analysis and technical indicators improved the prediction performances, especially in the LSTM model. Forecasting on stock market data could be more robust when higher frequency data is coupled with automated feature extraction.

Keywords: Stock price prediction, technical indicators, sentiment analysis, stacked autoencoders, convolutional neural networks, long short-term memory

MAKİNE ÖĞRENMESİ MODELLERİNİ KULLANARAK HİSSE SENEDİ ÖNGÖRÜSÜ

ÖZ

Borsa verilerinin öngörüsü, bu alandaki verilerin dinamik ve karmaşık yapısı nedeniyle zorlu bir iştir. Burada, hisse senedi fiyatlarını öngörebilmek için farklı veri kaynaklarını ve derin öğrenme mimarileri kullanmanın etkileri üzerine kapsamlı bir analiz sunuyoruz. Derin öğrenme modellerinin başarımındaki en büyük etmenlerden birisi verilerin etkili bir şekilde temsil edilmesidir. Bu çalışmada, hisse senedi piyasaları üzerinde finansal dış etkilere neden olabilecek haber makaleleri üzerinde duygu analizi yapılmış ve hisse senedi fiyat veri setlerinden çeşitli teknik göstergeler çıkarılmıştır. Veri gösterimi dışında, derin öğrenme algoritmalarının başarımı hiperparametrelerine bağlıdır, bu nedenle otomatik Bayes optimizasyonunu kullanarak geliştirilen derin öğrenme modellerini optimize ettik.

NASDAQ ve NYSE hedef pazarlarında işlem gören on farklı hisse senedi veri setinden günlük kapanış fiyatlarınını öngörebilmek için optimum Evrişimsel Sinir Ağı (ESA), Uzun Kısa Süreli Bellek (UKB) ve Yığınlı Otomatik Kodlayıcı modellerini geliştirdik. Çalışmamız duygu analizinin hisse senedi öngörüsü üzerindeki etkilerini ve karmaşık veri sunumlarında önerilen derin öğrenme modellerinin değerlendirilmesini kapsamaktadır. Öngörünün değerlendirilmesi çeşitli hata ölçevleri ile analiz edilmiştir. Elde edilen sonuçlar, geliştirilen ESA modelinin diğer modellerden daha iyi başarım sağladığını göstermiştir. Duygu analizi ve teknik göstergelerin entegrasyonu, özellikle UKB modelinin öngörü başarımını iyileştirmiştir. Yüksek sıklığa sahip veriler otomatik öznitelik çıkarımı ile birleştirildiğinde hisse senedi öngörüsü daha gürbüz olarak gerçekleştirilebilir.

Anahtar kelimeler: Hisse senedi öngörüsü, teknik göstergeler, duygu analizi, yığınlı otomatik kodlayıcı, evrişimsel sinir ağları, uzun kısa dönemli bellek

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 Motivation

In the field of finance, stock markets and their trends have many variations and volatility due to their nature. Investors and stock market analysts examine the behavior of stocks with different approaches and determine their trading strategies accordingly. Since the stock market produces large amounts of data every day, it is very difficult to evaluate all current and past information of stock markets for forecasting future trend. Nevertheless, stock forecasting has been a popular topic by researchers. In line with different demands in the field of finance, many solutions were sought for various purposes.

Stock market analysis techniques are generally composed of two different components as fundamental analysis and technical analysis. Fundamental analysis is based on the evaluation of different political events, which are thought to have an impact on the financial markets, as well as financial factors such as general economic conditions, external events, earnings, expenses, production, unemployment, and industry conditions. Besides, technical analysis corresponds to the forecasting of the stock prices for the future through different signals extracted from the historical stock price dataset. Considering these, it is clear that many parameters should be taken into consideration for the solution of any kind of problem in the financial field. This broad scope has started to be referred to as financial computing, with researchers from different disciplines offering various methods and techniques for problems in financial domain. Stock price forecasting and trend analysis stand out among the studies in this field. This orientation has raised several controversies over the predictability of stock prices. In semi-strong form of Efficient Market Hypothesis (EMH), Fama E. stated that stock series reflects all publicly available information and therefore, emerging new trend movements are unpredictable (Fama, 1970). However, many studies have been carried out on stock market forecasting. There is an extensive literature proposed for the solution of specific financial problems. This

extensive literature has led to the emergence of methods based on different approaches.

Traditional autoregressive models such as autoregressive moving average (ARIMA) used in time series analysis include estimation of parameters for modelling data. However, traditional machine learning and autoregressive models are insufficient in modelling complex data because they include few non-linear operations. With the increase of the computation capacity in recent years, use of deep learning techniques has increased the effectiveness of solution suggestions in finance field. The consistency of intelligent trading systems achieved with more effective technical and fundamental analysis is discussed with semi-strong from of EMH. Another approach for modelling complex time series is to extract new relevant features and use different data sources that include certain relationship to the analyzed time series. Automated feature extraction is realized with deep learning models on high dimensional feature space enriched with external data sources. In recent years, deep learning models with automated feature extraction are used to solve various problems in financial domain (Long, Lu, & Cui, 2018; Hoseinzade, Haratizadeh, 2019). With the proposed deep learning models in the studies, higher performance is aimed with the relevant information extracted from stock series. This approach is a mixture of related information, fundamental analysis, and technical analysis.

Our motivation in this study is to propose an approach that aims to combine different external data sources and optimal deep learning models to forecast stock datasets traded in different financial markets. Accordingly, we also aim to evaluate the relationships between historical stock time series and polarity scores obtained by sentiment analysis on news articles. Forecasting performance of the proposed optimal deep learning models will be compared; advantageous and disadvantageous of the architectures are will be emphasized. On the forecasting results, the profit of an investor who trades with different trading strategies is examined with a simulation.

1.2 Problem Definition

Many researches have been conducted for stock price forecasting, bringing different disciplines together for many years. In addition to human analysts, there is no rigorous method developed on the creation of intelligent systems that can make consistent trading decisions in the target markets. For this reason, many models have been proposed to investors that can provide solutions to different problems in this domain.

It is not directly tied to the economic situation of the country where the stock price of any company is located. Nowadays, stock prices are affected by many different external effects like political events, news, fast data processing etc. The stock market includes many different sectors. This led to some relationships in the data of companies with similar orientations. This situation has caused people who invest in the stock market to invest in companies that are traded in similar sectors. Thus, in most studies, models developed for stock forecasting were used on a certain target market and stock dataset for a certain sector. Especially, when developing machine learning and deep learning-based models, training only on a specific target market enabled us to question the generalizability of the results obtained. Accordingly, one of the problems to be solved in this study is the generalizability of the developed models on different stock data and target markets.

To solve such problems, many statistical and computational approaches must be used. In addition to the technical indicators defined in conjunction with technical analysis in stock market forecasting, multiple different data sources can increase model performance and reliability. In parallel, the relationship between stock series and polarity scores obtained by conducting sentiment analysis on related news articles should be investigated. The resulting high dimensional attribute space requires the implementation of some data mining steps such as preprocessing, data mapping, feature selection, and validation before being given to the models to be developed.

On the other hand, since the performance of a model largely depends on the dataset used, choosing the optimal model for stock price forecasting is another challenge. Therefore, a comparative evaluation is usually required on as many different models and configurations as possible. A lot of computational resources are needed to perform experiments on different configurations and datasets. Parallel computing is of great importance, especially during the training of deep learning models. Functional improvements experienced in graphics processing units (GPU) in recent years ensures training processes have been carried out using GPUs. This also led to the questioning of the reproducibility of the results obtained. Variables such as the differences in the problems to be solved, forecast horizon, different target markets etc. make it difficult to compare the results obtained from previous researches.

As a summary, stock price forecasting is a challenging problem that requires developing some learning models that combine independent preprocessing methods and different data sources. In this study, stock price forecasting and trend analysis were performed using different data sources with proposed deep learning models. While doing this process, the relationship between polarity scores obtained from sentiment analysis on news headlines and stock series and their effects on forecasting results were examined.

1.3 Contribution

In this study we included the different features obtained as a result of technical and fundamental analysis methods for the prices of 10 different stocks traded on the NASDAQ and NYSE target markets. The most meaningful features were selected over the obtained feature space. Hyperparameters of the developed deep learning models are optimized with Bayesian Optimization. Subsequently, the forecasting results were compared with different configurations.

The first focus of the analysis applied in this study is automated feature extraction. In this study, automated feature extraction was performed with CNN and SAE

architectures to achieve a good prediction with the complexity of the feature space created with three different data sources; polarity scores obtained by sentiment analysis applied on news articles, historical stock prices, and calculated technical indicators. The main contributions of this study can be summarized in two folds. Firstly, stock close prices are predicted by applying feature extraction with the proposed deep learning architectures using different data sources. In addition to comparing the performance of the commonly used deep learning algorithms, the effects of the sentiment analysis on the performance of deep learning models has been discussed. Secondly, how fundamental and technical analysis effects to behavior of stocks traded in different target markets interpreted in line with collected external information are presented.

Some of the experiments performed during this thesis study presented in the Innovations in Intelligent Systems and Applications Conference (ASYU) 2019 (Site, Birant & Işık, 2019) and another experiment performed during the thesis study accepted as an oral presentation in the 28th Conference on Signal Processing and Communications Applications (SIU) 2020 (Site & Işık, 2020).

1.4 Organization of the Thesis

This thesis consists of five chapters organized as follows:

In Chapter 2, we provide detailed background information and literature review of the related studies and techniques. With these, how this study related to other researches in the literature is summarized. The background information of traditional machine learning models, deep learning models, feature selection methods, some numerical transformation for technical analysis, sentiment analysis and text processing techniques used in the study are explained in different subsections.

In Chapter 3, we introduce the proposed methodology and describe preliminary experiments, the workflow of the study, technical and fundamental analysis for

creating datasets, our proposed deep learning models, and their learning configurations.

In Chapter 4, we present the evaluation of the forecasting performance by explaining the obtained results in different experiments with the proposed models. The advantages and disadvantages of the proposed models on different datasets are discussed, and the improvements of sentiment analysis on the forecasting performance are evaluated. With this, the profits of a particular trading strategy are simulated with the obtained results.

In Chapter 5, we summarize our conclusions and findings.

CHAPTER TWO
LITERATURE REVIEW

In this chapter, firstly, we will present detailed literature review of relevant studies and available techniques for stock price forecasting. We think that it will be too challenging to gather the studies carried out in this domain under one topic, thus we will present the deep learning models that are described as the state of the art in recent studies. In addition, the background of the techniques and methods used in the experiments of this study will be explained in a detail. Accordingly, the methods used for exploratory data analysis, sentiment analysis, the traditional machine learning models and deep learning models used for forecasting will be explained respectively.

2.1 Related Work

Many different studies have been carried out in the field of stock forecasting. Different techniques and solutions were offered by researchers according to the problem to be solved. In previous studies for prediction of stock prices, statistical analysis of past stock price movements was widely implemented. These studies (Hamilton, 1989; Tong, 1983; Zhang 2003) used linear autoregressive models to predict returns of stock prices. The application of traditional machine learning based models in this area has increased considerably with the increasing popularity of machine learning (Sezer, Gudelek, & Ozbayoglu, 2020). Along with the technical analysis; the use of linear and non-linear machine learning techniques with various preprocessing and feature extraction methods provided various solutions for stock forecasting problems. Artificial Neural Network (ANN) and Support Vector Machine (SVM) models, which provide non-linearity and flexibility among traditional machine learning models, become prominent in stock market forecasting and analysis. Since stock data are high dimensional time series with a lot of noise, in most cases linear learning models outperformed by non-linear ones (Aras, & Karakoc, 2016; Qiu, Song, & Akagi, 2016; Mohamed, 2010). Hiransha et al. compared the forecasting performance of most common linear and non-linear machine learning models used for forecasting on five different stocks traded on the

NSE and NYSE target markets (Hiransha, Gopalakrishan, Menon, & Soman, 2018). They found non-linear machine learning models more successful than linear models. Apart from this, performance acquisition of deep learning was examined with four different deep learning models. Patel et al. forecasted the direction of movement in the Indian stock markets using ten different technical indicators (Patel, Shah, Thakkar, & Kotecha, 2015). The best performance achieved with the RF classifier compared to ANN, SVM, and naïve Bayes. Appendant features increased the performance of all models. In addition, some researchers found linear models are more successful than non-linear ones in their experiments (Thaworwong et al., 2004; Enke et al., 2004). Hegazy et al. worked on the daily prediction of closing prices of different companies in various sectors in the S&P 500 target market (Hegazy, Soliman, & Salam, 2013). They proposed the least squared SVM (LS-SVM) model for the prediction process and they optimized the model with Particle Swarm Optimization (PSO). The proposed LS-SVM model was compared with ANN, and achieved higher performance than the ANN model. Kara et al. compared the performance of a feed forward ANN and a traditional SVM on stock datasets using with ten different technical indicators to predict the price movement direction of Istanbul Stock Exchange (Kara, Boyacioglu, & Baykan, 2011). They found that ANNs prediction performance is better than SVM in their high dimensional feature space.

The focus of recent studies has been on neural network architectures based on deep learning. Studies using deep learning algorithms can be divided into two main groups. While the first group aims to increase the performance of deep learning models on datasets, the second group tries to provide a meaningful hypothesis space suitable for the learning model by extracting new features from stock datasets. Gündüz et al. created a correlation matrix with feature selection from technical indicators and historical stock prices (Gündüz, Çataltepe, & Yaslan, 2017). The forecasting of traded stocks in the BIST 100 index has been realized with a Convolutional Neural Network (CNN) architecture. Their proposed model achieved better performance than linear machine learning models used in the study. Among the deep learning-based models, the studies are generally focused on certain deep

learning model architectures. Long Short-Term Memory (LSTM)-based models, which can cope well with sequential data containing long-term dependencies, are one of the most widely used learning models in stock market forecasting. Thanks to the high representation capacity and data sensitivity of deep learning models, it is common to use different data sources for stock market forecasting and analysis (Sezer, Gudelek, & Ozbayoglu, 2020). Technical indicators are used by human financial analysts to determine stock returns for certain periods. The complex and noisy structure of stock markets can cause major errors in this case.

The behavior of the stocks is quite open to external influences, which has spread use of different data sources in studies. Sentiment analysis performed on news articles is one of the most common secondary data sources. Vargas et al. forecasted movements of the stock prices traded in S&P 500 index by performing sentiment analysis on 106494 financial news articles obtained from Reuter's website (Vargas, de Lima, & Evsukoff, 2017). The relationship between news data and historical stock data was confirmed in a study using CNN and Recurrent Neural Network (RNN) architectures. Apart from that, working on only one target market makes the generalization of the achieved results open for discussion. Nayak et al. examined the daily and monthly forecast performance of traditional machine learning models using tweets and two different news datasets (Nayak, Pai, & Pai, 2016). They observed different improvement rates for each stock, which is sampled from three different sectors as bank, oil, and mining. Apart from the news about financial markets, different solutions have been proposed with various data sources containing relevant information about stock market behavior. Dong et al. used the popularity ratio of stocks traded in the Chinese markets in the search engine Baidu (Dong, Dai, Liu, Yu, & Weng, 2019). Kim et al. proposed a hybrid CNN-LSTM model to predict stock prices on S&P 500 target market (Kim T. & Kim H. Y., 2019). While doing this, candle stick charts, a different representation of stock data, were used as additional data source. In proposed model, the candle stick charts are handled as a Computer Vision problem via CNN part of the hybrid model and its effects on forecasting performance are examined. In the hypothesis space created by using features obtained from external data sources, more meaningful feature extraction can be made

with deep learning models. Therefore, one of the most crucial approaches in this field is to perform automatic feature extraction through the learning model over the created datasets. With this motivation, automated feature extraction is the first focus of this thesis study. Reproducing results of the studies carried out stock price forecasting is quite challenging. These studies in the literature focused on different problems on the same research topic. This situation originated due to having different time periods, datasets, goals, and methods. Therefore, the preprocessing steps and methods for the problem to be solved are quite diverse. This situation requires the studies to be conducted collectively and comparatively. The summary and scope of the studies mentioned in this section are presented in Table 2.1.

Table 2.1 A summary of studies on stock market forecasting

Authors	Datasets	Range	Feature Set	Goal	Method
(Kara et al., 2011)	BIST 100	-	Technical Indicators	Index Movement	ANN, SVM
(Patel et al., 2015)	4 Indian Stocks and two different indices	2003-2010	Technical Indicators	Stock Price Movements	ANN, SVM, RF, NB
(Gündüz et al., 2017)	The most traded 3 different stocks from BIST 100	2011-2016	Technical Indicators	Stock Price Movements	Logistic Regression, Gradient Boosting Machine
(Nayak et al., 2016)	Indian Stocks from three different sectors, tweets from Twitter API, 2 different news articles	2002-2015	Polarity Scores	Stock Price Movements	Boosted Decision Tree, Logistic Regression, SVM
(Vargas et al., 2017)	Different Stocks from S&P 500 target market, different representations of textual data (word embedding and sentence embedding)	2006-2013	Technical Indicators, polarity scores	Stock Price Movements	CNN, RNN, RCNN hybrid model, ANN, SVM
(Hegazy et al., 2013)	13 different stocks	2009-2012	Technical Indicators	Stock Price Forecasting	ANN, LS-SVM with PSO optimization

Table 2.1 continues

(Hiransha et al., 2018)	3 different stock from NSE and 2 different stock for NYSE	2007-2017	-	Stock Price Forecasting	LSTM, CNN, RNN, MLP, ARIMA
(Kim et al., 2019)	Different S&P 500 stocks and their chart images	2016-2017	Features extracted by CNN	Stock Price Forecasting	CNN, LSTM, feature fusion CNN-LSTM
(Dong et al., 2019)	Historical stock data from Chinese stock market and Baidu search indexes	2011-2016	Search Engine index ratios	Stock Price Forecasting	ARIMA, ANN, Least Squares SVR, and their proposed methods (LBSI and TS-GFT)
Our Final Experiment	10 different stocks from NASDAQ and NYSE indexes, news articles from New York Times	2007-2017	Technical Indicators, polarity scores	Stock Price Forecasting	CNN, LSTM, Stacked Autoencoder

2.2 Exploratory Data Analysis

2.2.1 Autocorrelation Function

Evaluation and analysis of the stock datasets used in the study has great importance to deal with the problem to be solved. Since the stock data is a time series, it carries different characteristic patterns. One of them is the measure of the relationship between the values on the time series. It is of great importance that the time series data used in the study is non-stationary. Non-stationary time series does not have constant mean or variance throughout the series. Therefore, we analyzed the stock data used in the study with the autocorrelation function to obtain the degree of similarity between variables in weekly time lags. In addition to this, we determined our stocks on different target markets accordingly. The autocorrelation function can be used to see how much the past stock prices have an impact on the future price. It can be assumed that an investment with a positive correlation carries less risk for the future. Therefore, autocorrelation function is also a technical analysis method used by analysts. Equation 2.1 specifies the autocorrelation function value within the time interval defined by the value *T*. Autocorrelation function can be defined as follows,

$$r_k = \frac{\sum_{t=k+1}^{T} \left(y_t - \acute{y}\right)\left(y_{t-k} - \acute{y}\right)}{\sum_{t=1}^{T} \left(y_t - y\right)^2} \tag{2.1}$$

where *y(t)* represents sorted dataset by ascending order with time *t, y(t – k)* represents same dataset by lagged by *k* units and variable *y'* represents mean of original dataset.

2.2.2 Pearson Correlation Coefficients

Pearson Correlation Coefficients returns the values of the association between the different variables. Besides, Pearson Correlation Coefficients gives information about the direction of relationships and magnitude. In particular, technical indicators have revealed high correlations. By obtaining correlation values between the features, feature selection is performed and the technical indicators that are meaningful for the models in the multidimensional feature space. Pearson Correlation Coefficient can be defined as follows,

$$r = \frac{\sum \left(X - \overline{X}\right)\left(Y - \overline{Y}\right)}{\sqrt{\sum \left(X - \overline{X}\right)^2}\sqrt{\sum \left(Y - \overline{Y}\right)^2}} \tag{2.2}$$

where variable $\overline{X}$ represents mean of variable X, variable $\overline{Y}$ represents mean of variable Y. Variable r takes values between +1 and -1 to identify correlation of given features. A value of 0 indicates that there is no relationship between the existing features.

2.2.3 Z-Score Normalization

Providing a rich feature space, especially deep learning-based models, provides great flexibility for learning data samples that appear to be irrelevant. Accordingly, a

z-score normalization was applied on the features in order to obtain more consistent results on a certain sample range. Also, z-score normalization is the strategy of normalizing data that prevents outliers from datasets. Thus, z-score normalization indicates how far the scaled variable moves from the mean of dataset. Z-score normalization can be defined as follows,

$$z-score(x)=\frac{(x-\mu)}{\sigma} \tag{2.3}$$

where μ represents mean of variables and σ represents standard deviation of samples.

2.3 Sentiment Analysis

Sentiment analysis is a Natural Language Processing (NLP) field that uses various methods to identify, measure, and classify emotional states and subjective information. Sentiment analysis is useful to solve different problems in wide range of area. In general, sentiment analysis can be classified under two main titles as lexicon-based methods and machine learning methods. In this thesis, we aimed to define the patterns related to the behavior of stock series by applying sentiment analysis on headlines of news articles. Accordingly, we focused on lexicon-based models since it does not require any training data. Sentiment analysis has many difficulties in practice despite different methods and techniques. Some of these difficulties come from the fact that the written text has a contextual sparseness and uses different language representations such as shortness of the sentences while expressing emotions in the text. Thus, such type of problems requires a high-quality lexicon to be able to perform fast and accurate sentiment analysis. In literature, there are a significant number of sentiment lexicons. Among them, VADER, AFINN, SentiStrength, and SentiWordnet have been widely used in recent years (Sadia, Khan, & Bashir, 2018; Musto, Polignano, & Semeraro, 2014). These suggested methods have been used on our datasets by trial and error. Since the most suitable for the

purpose of the study and the best performance is obtained with VADER sentiment analyzer, it was used for deep learning models. As a general definition, sentiment lexicon is the list of words with labels based on their semantics. It is very time-consuming to create lexicons manually for evaluating text datasets. For this reason, it is more common to use sentiment analysis relies on existing lexicons. Besides, the weakest point of lexicon-based techniques is that most of the words on the processed text data are not found in sentiment lexicons.

2.3.1 Valence Aware Dictionary and Sentiment Reasoner (VADER)

We used VADER (Valence Aware Dictionary and Sentiment Reasoner) which is a lexicon and rule-based sentiment analysis tool (Hutto & Gilbert, 2014). VADER was built for social media data analysis, so it also contains emoji lexicon, emoticons, slang words and abbreviations. News articles are also similar in terms of meaning and emphasis to social media data or microblogs, so better results are obtained than other lexicons. The VADER lexicon contains 4 main parameters: token, mean-sentiment, standard deviation, and human-sentiment ratings. These parameters can be defined as follows:

- token: corresponds to the defined word in lexicon.
- mean-sentiment: corresponds to mean of human-sentiment ratings.
- standard deviation: It is the standard deviation of human sentiment ratings. (2.5 is taken as the limit.)
- human-sentiment ratings: includes ten different human valence ratings between +4 and -4 for corresponding token.

The current algorithm works using only the token and mean-sentiment parameters. The other two parameters provide a verification for new words to be added to lexicon. In addition, VADER sentiment automatically performs a number of preprocessing stages, such as remove stop words. Such additional features reduce processing overhead, significantly saving processing time.

2.4 Traditional Machine Learning Methods

Traditional machine learning models were used in first experiment conducted during the thesis study. Some of linear and non-linear machine learning models compared with state-of-the-art deep learning models. The traditional machine learning models that we used in our experiment will be summarized as subsections.

2.4.1 Linear Regression

Linear regression is one of the simplest machine learning algorithms. In the experiments, a linear machine learning model was used as a baseline. In general, linear regression is a system that takes data points as input vector and maps a scalar value. This output can be defined as follows,

$$\hat{y} = w^T x + b \tag{2.4}$$

where $\hat{y}$ is a predicted scaler value, w^T is a weight matrix, x is input vector, and b is bias vector.

Weight matrix (w^T) determines how features affect the prediction. The parameter b is called bias vector. When the linear regression cannot find any input vector, model biases in the direction of value b. Difference between predicted $\hat{y}$ value and value of y calculated with a loss function such as Mean Squared Error and weight values are tried to be optimized. Mean Squared Error can be defined as follows,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(f_i - y_i\right)^2 \tag{2.5}$$

where N represents the number of observations, f_i represents ith observation of actual output value, and y_i represents ith observation of predicted value.

2.4.2 Ridge Regression

Ridge regression is a linear prediction model. Ridge regression performs L2 regularization, which adds a penalty parameter equal to the square of the coefficients (Tikhonov & Arsenin, 1977). This penalty may cause some coefficients of the model to be zero, which results in simpler models being produced. In this way, produced models can be interpreted more easily. In addition, this allows Ridge regression to be used as a feature selection method. Parameter a defines a function to control the weight of the coefficients. This technique is a good alternative to avoid overfitting problems of linear models. Ridge regression can be defined as follows with L2 normalization added to the least squares cost function,

$$J(w)=\sum_{i=1}^{N}\left(f_i-y_i\right)^2+\alpha\sum_{j=1}^{m}w_j^2 \tag{2.6}$$

where N represents the number of observations, f_i represents ith observation of actual output value, and y_i represents ith observation of predicted value. a parameter defines L2 regularization strength and decreases the weights of the model as the a value increases. If a = 0, the model becomes linear regression.

2.4.3 Support Vector Regression

Support Vector Machines (SVM) are nonparametric machine learning models that can be applied to both linear and non-linear datasets (Cortes & Vapnik, 1995). After that, Support Vector Regression (SVR), which is an adaptation of SVMs for regression tasks, was proposed. (Vapnik, Drucker, Burges, Kaufman, & Smola, 1996). SVR is the same as the classification-based Support Vector Machine models except one key difference. Since SVM models operate on Hilbert space, predicting a real number during regression has infinite possibilities. For that reason, SVR cannot

produce probabilistic predictions, but prediction of data point can be made with SVR. Therefore, the epsilon parameter is defined to predict a real number. A part of error is tolerated with the epsilon parameter. Epsilon parameter provides flexibility to define how much error is acceptable in our model so that the SVR will find a suitable line (or hyperplane in higher dimensions) to fit the data. In non-linearity cases, using kernel functions, input vectors are directly substituted in the kernel function to find the value in the high dimensional feature space. Since there are different variants of SVR, in the preliminary thesis study, non-linear SVR and polynomial kernel (Equation 2.7) were used. Mathematically, the fundamentals of SVR finds the coefficients that minimize can be defined in form of Lagrange multipliers in Equation (2.8) and the function used to predict new values is defined in Equation 2.9.

$$G(x_i, x_j) = (1 + x'_i * x_j)^q \tag{2.7}$$

$$L(a) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_i - a'_i)(a_j - a'_j)G(x_i, x_j) + e\sum_{i=1}^{N}(a_i + a'_i) - y_i\sum_{i=1}^{N}(a_i - a'_i) \tag{2.8}$$

$$f(x) = \sum_{n=1}^{N}(a_n - a'_n)G(x_n, x) + b \tag{2.9}$$

where x_i and x_j represent the input vectors, q defines degree of polynomial in polynomial kernel ($G(x_i, x_j)$); a_i and a_j are non-negative multipliers, e is epsilon parameter in function $L(a)$. a_n, x_n, and b define multipliers, support vectors, and bias vector respectively in function $f(x)$. Additionally, SVR has Karush-Kuhn-Tucker (KKT) optimization conditions to achieve optimal solutions (Karush, 1939; Kuhn & Tucker, 1951).

2.5 Deep Learning

In this thesis study, a wide feature space is given to models that will realize forecasting. With calculated technical indicators and sentiment analysis, a higher dimensional feature space is composed. Generalization with new samples becomes

exponentially difficult when working with high dimensional data. Traditional machine learning methods are insufficient to learn complex functions in high-dimensional spaces. The first experiment in this thesis study is about questioning this situation. With deep learning algorithms, we can enable models to define complex functions by adding more layers and increasing the number of units in each layer. In this way, a larger hypothesis space is obtained for designed learning models. In addition to this, learning of models runs slower as the hypothesis space increases. Deep learning models include many different architectures and configurations to solve various problems. Accordingly, different deep learning models were tested in different configurations in line with this addressed problem. We briefly explain deep neural network architectures on which the proposed models are based, and used in our experiments in the following sections.

2.5.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) basically act as a short-term memory and enable the neural network to remember recent historical data. The first models proposed for RNN date back to the 1990s, and earl models are called the Simple Recurrent Neural Network (SRN). The idea of SRN was first proposed by Jeff Elman in 1990 (Elman, 1990). This network, referred as Elman networks, is one of the-first SRN structures in literature and was developed mainly from autoregressive models. Too many variants of SRN models have been proposed. The designed networks have different characteristics developed for different problems. Today, RNNs used in deep learning models can better generalize the problem. This situation is the result of significant developments in artificial neural network studies that started in 1940's. With the acceleration of the studies, some problems which could not be solved before have been solved one by one. As a result, various network topologies used differently in the literature. These include, different RNN types such as Hopfield Networks, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) etc. The structure of Basic Recurrent Neural Networks and the special RNN architectures that form state of the art models for solving today's sequential learning problems are the main ideas of many models implemented for stock market forecasting.

Basic Recurrent Neural Networks can be thought of as a neural network containing multiple copies of regular Feed Forward Neural Networks (FNN). As mentioned earlier, the main difference of RNNs from FNNs is that, they have feedback from previous layers within the network. Thus, different loops occur within the network. The structure of these loops ensures that the information is continuous throughout the network. The distributed hidden state on the network allows to gather a lot of information about the past efficiently. The architecture of Basic RNN neuron presented in Figure 2.1, in which x_t and h_t are defined as the input and the hidden state respectively, at time t. Basic RNN neuron does not contain memory unit.

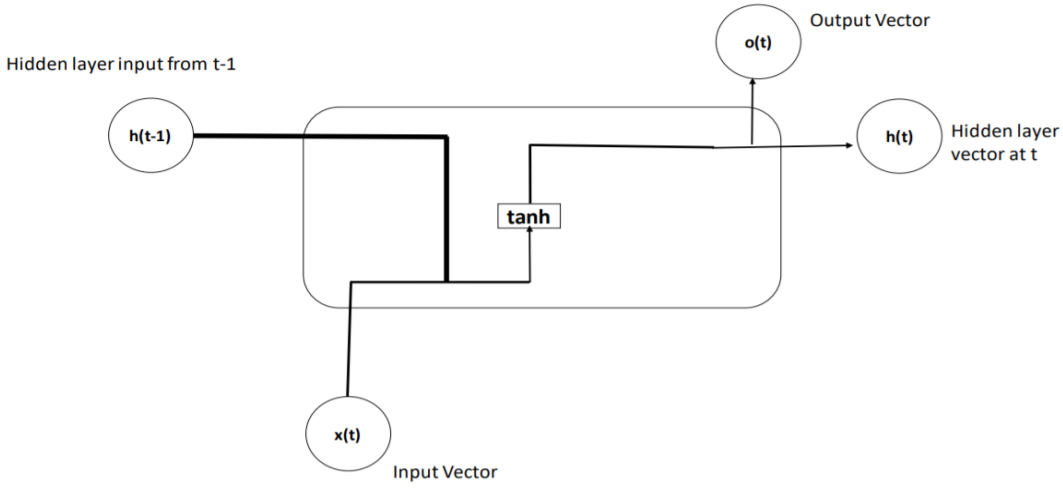

Figure 2.1 The architecture of a Basic RNN neuron

Mathematically, the fundamentals of Basic Recurrent Neural Networks can be defined as follows,

$$h_t = \tanh\left(U_h * x_t + V_h * h_{t-1} + b_h\right) \tag{2.9}$$

$$y_t = \left(W_y * h_t + b_h\right) \tag{2.10}$$

where U_h, W_y, V_h are weight matrices, and b_h is bias vector. h_t represents the hidden layer vector, y_t is an output vector.

During thesis study, Basic Recurrent Neural Networks, LSTM, and GRU architectures were compared in the preliminary experiments, and the developed LSTM model proposed in the third experiment of thesis study.

2.5.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) was proposed by Hochreiter et al. for the solution of long-term dependencies in sequential learning problems (Hochreiter & Schmidhuber, 1997). In regular RNNs, the problem of vanishing gradients occurs during the learning of long-term dependencies of sequential data, LSTMs completely overcome this problem with the uninterrupted gradient flow over them (Hochreiter, 1998). Backpropagation of error occurs in a special way during training, as in other RNN structures. During backpropagation, instant time lag is given to the network as input again. This type of error minimization is called backpropagation through time (BTT) (Rumelhart, Hinton, & Williams, 1986). LSTM structurally consists of cell blocks. Opposite to regular RNNs, LSTM cells are recurrently connected to each other instead of existing hidden units. In addition to this, LSTM can dynamically control the time scale of the network and the behavior of different units within it. These units are divided into three parts as forget gate, input gate and output gate. Input gate allows to change the state of the current memory cell. On the other hand, output gate controls output flow from the cell state. Also forget gate can choose to remove or add new information from memory cell input to memory cell output.

The information available in the stock market forecasting always replaced with new ones. Due to the nature of the markets, the frequency of rare events are quite variable. For this reason, a model that can capture rare events and count distances between events is a good alternative for such problems. Hence, Peephole LSTM structures were presented by Schmidhuber et al (Gers, Schraudolph, & Schmidhuber, 2003). Unlike LSTMs, Peephole LSTMs can interfere with the contents of the forget gate and output gate with thie cell states. This structural difference allows the model to understand the available information on the cell state.

The architecture of LSTM cell presented in Figure 2.2, in which x_t, h_t, and c_t are defined as the input, the hidden state, and the cell state, respectively, at time t.

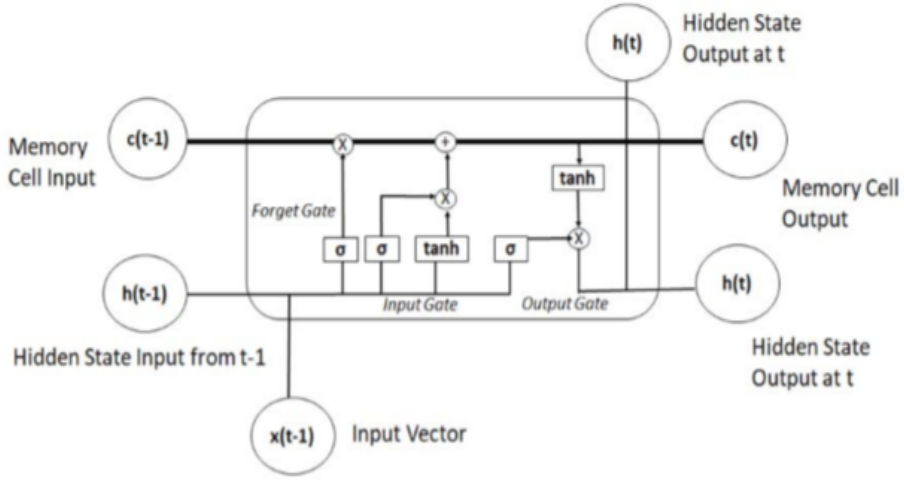

Figure 2.2 The architecture of an LSTM cell

In Equation 2.11-2.15, forget gate, input gate, output gate, the cell state (c_t) and hidden state (h_t) are defined as follows,

$$forgetgate = \sigma\left(W_f * x_t + U_f * h_{t-1} + b_f\right) \tag{2.11}$$

$$inputgate = \sigma\left(W_i * x_t + U_i * h_{t-1} + b_i\right) \tag{2.12}$$

$$outputgate = \sigma\left(W_o * x_t + U_o * h_{t-1} + b_c\right) \tag{2.13}$$

$$c_t = f_t * c_{t-1} + i_t * \tanh\left(W_c * x_t + U_c * h_{t-1} + b_c\right) \tag{2.14}$$

$$h_t = \left(o_t * \tanh\left(c_t\right)\right) \tag{2.15}$$

where W_f, W_i, W_c, W_f, U_f, U_i, U_o and U_c are weight matrices, and b_f, b_i, b_o and b_c are bias vectors.

2.5.3 Gated Recurrent Unit

Gated Recurrent Units (GRU) were first proposed by Cho et al. in 2014 (Cho, Chung, Gulcehre, & Bengio, *2014*). GRUs have the characteristics of LSTMs but are a simpler variant of LSTMs. GRUs can cope with the vanishing gradients problem

similar to LSTMs. The main difference between two network structures is that the GRUs do not contain any memory and they include only two gates. GRUs combine forget and input gate as a single update gate. This main difference allows GRUs to consume less resources and work much more efficient than LSTMs. According to LSTM, deeper GRU networks can be created with similar resource consumption and running time complexity. Generally, LSTM model has a more sophisticated memory than GRU models, so it has the ability to predict more temporal related data. GRUs will perform better than LSTMs when data is low or the risk of overfitting is high (high noise levels). LSTM and GRU based models provide state-of-the-art solutions for some learning problems such as character level text modelling, handwriting recognition, image captioning, and machine translation. The architecture of GRU cell presented in Figure 2.3, in which x_t and h_t are defined as the input vector and the hidden state, respectively, at time t.

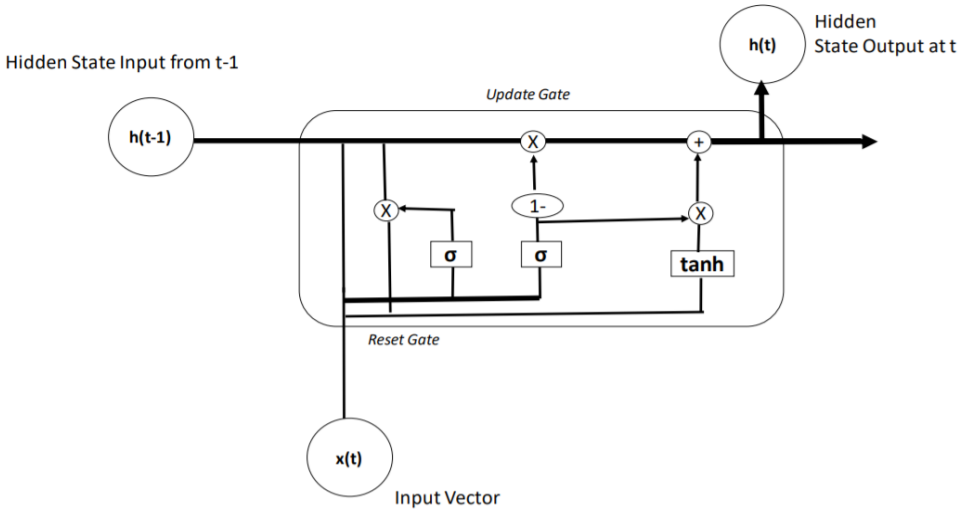

Figure 2.3 The architecture of an GRU cell

In Equation 2.16-2.19, reset gate (r), update gate (z), and hidden state or output vector (h_t) are defined as follows,

$$z = \sigma\left(W_z * x_t + U_z * h_{t-1} + b_z\right) \tag{2.16}$$

$$r = \sigma\left(W_r * x_t + U_r * h_{t-1} + b_r\right) \tag{2.17}$$

$$h' = \tanh\left(W_h * x_t + r * U_h * h_{t-1} + b_z \right) \tag{2.18}$$

$$h = z * h_{t-1} + (1 - z) * h' \tag{2.19}$$

where W_z, W_r, W_h, U_z, U_r, and U_h are weight matrics, and b_z, b_r, b_z are bias vectors. z and r represent the update gate vector and reset gate vector respectively. While h' and h represent the intermediate memory vector and output vector respectively.

2.5.4 Convolutional Neural Networks

Convolutional Neural Network (CNN) is a type of neural network used to learn convolutional filter weights. The convolution process on the neural network is performed linearly instead of the matrix multiplication. Thanks to the convolution process, every output unit in the network does not interact with every input unit. This structure ensures that CNNs have sparse connectivity structure. Sparse connectivity reduces the networks total time complexity and decreases memory consumption. Another important concept for CNNs is parameter sharing. Parameter sharing refers to the use of the parameters so that a learning model can define more than one complex function. Input values given to the network depends on the value of a different parameter weights on the network. This means learning a set of weights instead of learning each weight individually. Generally, a CNN consists of three main parts. After the convolution process carried out, a non-linear activation function is used in the second part of network. Following this, pooling functions are used in the third part of network. Pooling functions change the network's output in a specific statistical distribution. This ensures that the network is more robust against changes in input values. CNNs have different forms and architectures for different problem domains. 1-Dimensional convolution (Conv1D) is applied in CNN models performed on sequential data such as text, audio and stocks. Conv1D refers sliding the filter on the dataset for only one dimension. The optimal CNN architecture developed in the third part of our experiments, also moves in one dimension since the problem we want to solve is sequential time series.

2.5.5 Autoencoders

Autoencoders (AE) are neural network types that copy the input layer weights to the output layer. Hidden layer on the network is often called as code and it is responsible for representation of network input (Goodfellow, Begio, & Courville, 2016). The code basically consists of two main parts called encoder and decoder. Encoder maps the input values to the code, and decoder part perform reconstruction of original input values. In other words, the hidden layer of networks tries to learn an approximate distribution of the input, so the output in the network and the input are not identical. AEs have a wide range of usage and have different variants for various problems. The most important feature we will use for this thesis study is the ability to perform dimensionality reduction for high-dimensional spaces. Extracting meaningful representation from multidimensional and non-linear data samples in our dataset, has a great importance for more efficient learning. The main reason for this is that the observations are scattered around each axis in high dimensional space. Meaningful representation of data corresponds to learn simpler manifold from this high dimensional feature space. Depth can be added to the AEs with additional layers, as a result, encoder and decoder parts can achieve significant gains. Mapping of input to the code (Equation 2.20) and reconstruction of original input values from the code (Equation 2.21) are defined as follows,

$$M_t[\upsilon, \nu\,] = (M_{t-1}[\upsilon, \nu\,])^p \tag{2.20}$$

$$M_t[\,\upsilon,\, \nu] = (M_{t-1}[\,\upsilon,\, \nu])^p \tag{2.21}$$

As AEs behaves like regular feed forward neural networks, they are trained with backpropagation algorithm to minimize total reconstruction errors. In Equations 2.22-2.23 squared errors, and optimization of neural network defined as follows,

$$L(x, x') = \left| x - \left(W' \left((Wx + b) \right) + b' \right)^2 \right| \tag{2.22}$$

$$argmin[E] = argmin \frac{1}{N} \sum_{i=1}^{N} L\left(x_i, x_i'\right) \tag{2.23}$$

where x and x' are input and reconstructed input, $a(x)$ is the code, W and W' are weight matrices, b and b' are bias vectors, is activation function, $L(x, x')$ represents squared loss function, and in Equation (2.23) with x_i and $x_i^{'}$ identify i^{th} element of input and reconstructed input by decoder, and N defines the number of training samples.

2.6 Hyperparameter Optimization

Hyperparameters are all parameters of a model that are not updated during training and they are used for configuration of structure of learning models. Hence, the hyperparameter tuning is an optimization cycle to find the hyperparameter group that will lead the lowest error in validation dataset. Therefore, adjusting the hyperparameters of models does not always promise the best performance. The challenge in tuning these hyperparameters makes difficult to reproduce and extend the obtained results. Recent studies (Bergstra, Yamins, & Cox, 2013; Ilievski, Akhtar, Feng, & Shoemaker, 2017; Snoek, Larochelle, & Adams, 2012) have revealed that hyperparameter optimization is one of the biggest bottlenecks in performance improvement of learning models. One of the important requirements in hyperparameter tuning is that the applied tuning method should have a lower computational complexity. Tree-Structured Parzen Estimators (TPEs) is, a Bayesian optimization technique that we obtained more efficient results compared to well-known hyperparameter tuning methods such as Grid Search or Random Search. Hyperparameter optimization is basically optimizing loss functions of learning models on a graph-structured parameter configuration space. Using TPE, we limit our models to a tree-structured configuration space. So, this process offers a more optimistic time complexity for hyperparameter tuning. The best values obtained as a result of hyperparameter tuning by TPE were set as the latest configurations of proposed deep learning models in our third experiment.

CHAPTER THREE
METHODOLOGY

In this chapter, we will present the methods and techniques used in the performed experiments in detail. Accordingly, configurations created for experiments, datasets, preprocessing steps, enhancements for sentiment analysis, and proposed models for forecasting will be explained in next subsections respectively.

3.1 System Overview

We present a general overview of the proposed system in for the third experiment in Figure 3.1. In addition, three independent experiments were carried out in this thesis study. Datasets and methods used in the experiments will be explained in the next sections. Generally, some of the machine learning and deep learning models described in Chapter 2 were compared with different configurations on different datasets in the preliminary (first and second) experiments. Besides, deep learning models that we developed and proposed as a result of thesis work will be presented in the third experiment.

In the first experiment, the forecasting performance of different RNN architectures and traditional linear and non-linear machine learning models were compared weekly and monthly intervals on different datasets. Besides, it has been observed to what extend deep RNN architectures, with limited number of layers can model long-term dependencies. No external data sources were used throughout this experiment.

In the second experiment, close values of stocks we are tried to predict with selected technical indicators. Accordingly, deep learning-based models have been developed to reach constant results. Multi-layered CNN and Peephole LSTM architectures were designed and their forecasting performance on datasets were evaluated under different metrics. The results obtained constituted the technical infrastructure for the third experiment, the final part of the thesis study.

In the third experiment, we analyzed stock price forecasting performance by performing sentiment analysis on various news articles as well as technical analysis on stock data of different qualities. Accordingly, reliable results are aimed to obtain with different deep learning models that are most suitable for the problem, with different features selected for hypothesis space. In this context, this experiment is an investigation for the relationship between news articles and stock prices. The workflow of the study is divided into two pipelines accordingly. The final optimal models to be proposed in the thesis study are presented in the third experiment.

Summary and scope of the experiments carried out in thesis study are presented in Table 3.1.

Table 3.1 A summary of the experiments

	Datasets	**Range**	**Feature Set**	**Horizon**	**Method**
First Experiment	2 different stocks from DJIA and S&P 500	2006-2017 and 2013-2018	-	Weekly and Monthly	Basic RNN, LSTM, GRU, Linear Regression, Ridge Regression, SVR
Second Experiment	3 different stocks from NASDAQ	2007-2017	Technical Indicators	Weekly	CNN, Peephole LSTM
Third Experiment	10 different stocks from NASDAQ and NYSE indexes, news articles from New York Times	2007-2017	Technical Indicators, Polarity Scores from related news articles	Daily	CNN, LSTM, Stacked Autoencoder

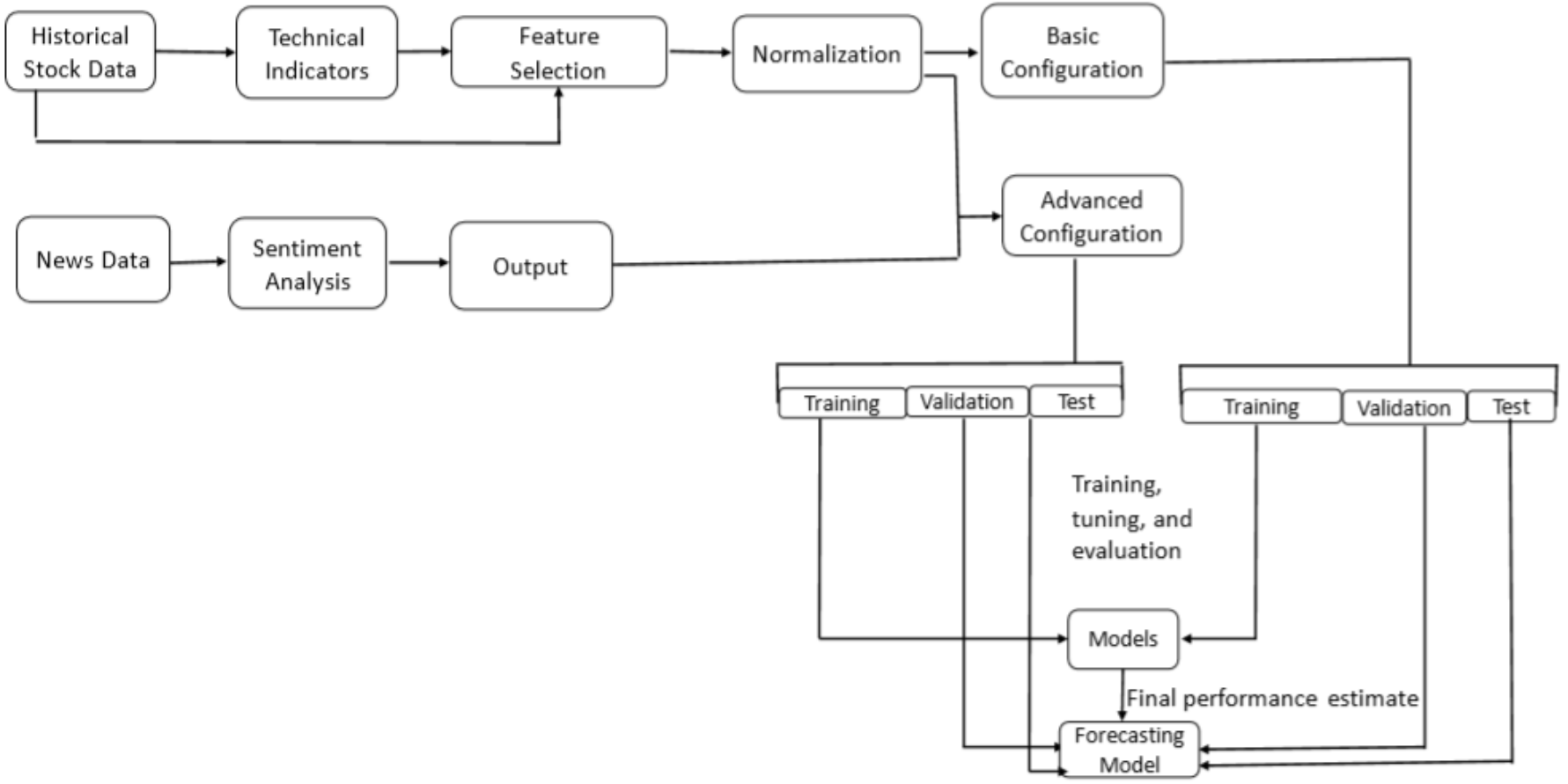

Figure 3.1 The system overview of the proposed models in the third experiment

3.2 Data Sources

3.2.1 Datasets of Preliminary Experiments

Datasets of the first two experiments will be presented under this section. In the thesis study, we conducted two independent studies as a preliminary in order to understand the methods based on the solutions of the different problems we mentioned in the previous section and the behavior of the stock series.

In our first experiment, we used two different stock series obtained from Yahoo Finance (*https://www.finance.yahoo.com/*). In order to evaluate the robustness of our results, we used datasets from S&P 500 and DJIA target markets and different number of observations. Datasets include daily open, high, low and close (OHLC) values for a given day. Among the selected stocks, GOOGL dataset covers a 10- year period, while AMZN dataset covers a 5-year period. With this choice, we tried to understand how models interpret long-term dependencies for forecasting of trend. For evaluation of observations between the specified range, we divided them into 80% - 20% to use them as (train and test samples, respectively). All datasets used in the first experiment are presented in Table 3.2.

Table 3.2 Selected stocks and their indices in the first experiment (Site, Birant, & Işık, 2019)

Indices	Symbols	Entity	Number of Samples	Start Date	End Date
DJIA	GOOGL	Alphabet Inc. Cl A	3019	2006-01-03	2017-12-29
S&P 500	AMZN	Amazon.com Inc.	1259	2013-02-08	2018-02-07

In our second experiment, we used three different stock series on the NASDAQ index obtained from Yahoo Finance (*https://www.finance.yahoo.com/*). The daily open, high, low and close (OHLC) values of the stocks cover the 10- year period between (2007-01-03) to (2017-12-29). A total of 2768 observations were obtained within the specified range. For evaluation of observations between the specified range, we divided them into 80% - 20% to use them as train and test samples, respectively. All datasets used in the second experiment are presented in Table 3.3. In addition, technical indicators are used in the second experiment. The selected

technical indicators and preprocessing stages provide the basis for the third experiment. Therefore, the used technical indicators will be described in the description of the third experiment. Datasets to be used for selecting the final models are limited to single index and stock series of companies belonging to a particular domain.

Table 3.3 Selected stocks and their indices in the second experiment (Site & Işık, 2020)

Indices	Symbols	Entity	Number of Samples	Start Date	End Date
NASDAQ	AAPL	Apple Inc.	2768	2007-01-03	2017-12-29
NASDAQ	GOOG	Alphabet Inc.	2768	2007-01-03	2017-12-29
NASDAQ	NVDA	NVDIA Corporation	2768	2007-01-03	2017-12-29

3.2.2 Data Sources For Third Experiment

3.2.2.1 Stock Datasets

In order to evaluate of observations, we used two different target market (NASDAQ and NYSE) and five different stocks covering ten-year time period. Different target markets bring various fundamental differences for stock exchange. One of these is NASDAQ, which is a dealer's market, while NYSE is and auction market. As this directly affects the functioning of the markets, it is a factor that affects the trend and other components of stock data over time. That's why we used two different markets to make more consistent evaluation in the final experiment. In addition, the fact that the datasets consist of stock series of companies belonging to various sectors constitute a criterion for the generalizability of the obtained results. Datasets obtained using Yahoo! Finance (*https://www.finance.yahoo.com/*) and include daily open, high, low and close (OHLC) values for a given day between (2007-01-03) to (2017-12-29). The total number of samples of selected stocks is 2768. For the evaluation of stock datasets, the first 2216 days were used for training and cross validation, and 552 days were used for testing. All stocks and their indices presented in Table 3.4.

Table 3.4 Selected stocks and their indices in the third experiment

Indices	Symbols	Entity	Number of Samples	Start Date	End Date
NASDAQ	AAPL	Apple Inc.	2768	2007-01-03	2017-12-29
NASDAQ	AMD	Advanced Micro Devices Inc.	2768	2007-01-03	2017-12-29
NASDAQ	CSCO	Cisco Systems Inc.	2768	2007-01-03	2017-12-29
NASDAQ	INTC	Intel Corporation	2768	2007-01-03	2017-12-29
NASDAQ	KO	The Coca-Cola Company	2768	2007-01-03	2017-12-29
NYSE	AXP	American Express Company	2768	2007-01-03	2017-12-29
NYSE	BA	The Boeing Company	2768	2007-01-03	2017-12-29
NYSE	GS	The Goldman Sachs Group Inc.	2768	2007-01-03	2017-12-29
NYSE	IBM	International Business Machines Corporation	2768	2007-01-03	2017-12-29
NYSE	JPM	JPMorgan Chase&Co.	2768	2007-01-03	2017-12-29

3.2.2.2 News Data

In the third experiment, New York Times Archive API (*https://developer.nytimes.com/*) is used as a textual data source. News articles published between 01-2007 to 01-2018 were parsed and filtered to match the active stock trading days. As a result, 1324274 articles with independent content were obtained as a json file. The content of obtained articles filtered according to the determined headlines and their polarity scores calculated. The workflow of this process will be explained in step by step in the preprocessing section.

3.2.3 Technical Indicators

The use of technical indicators for forecasting of stock prices is quite common in the literature (Patel, Shah, Thakkar, & Kotecha, 2015; Hegazy, Soliman, & Salam, 2013; Kara, Boyacioglu, & Baykan, 2011; Gündüz, Çataltepe, & Yaslan, 2017). Technical indicators are forecasting methods to monitor future price movements based on past price movements. This is based on assumption that stock prices follow

certain pattern. For this reason, technical indicators are the parameters frequently used by human analysts in their analysis. Time series generally decomposed into four basic components: trend, residual variations, seasonal and cyclic variations. While trend defines the long-term movement of the time series, seasonal and cyclic variations correspond to short-term and periodic fluctuations in time series. In addition, unpredictable, random and irregular variations in the time series are defined as residual variations. Many of the technical indicators provide different expressions of these components. In this way, short-term and long-term price movements are analyzed and time dependent entry and exit points of stocks are identified.

In this study, commonly used technical indicators with different aspects were collected to compose a rich feature space for the proposed learning models. Technical indicators were used in the second experiment as well as in the third experiment. Even if the used stock datasets are different, the selection of technical indicator group is the same for both of them. Since the calculation of some technical indicators depends on the time intervals, more features can be obtained by a single technical indicator. Accordingly, 20 technical indicators with different features and time lags were calculated (Table 3.5). This type of high-dimensional space exemplifies the curse of dimensionality problem presented by Bellman (Bellman & Schmidhuber, 1957). Technical indicators have created high correlations as they are mathematical transformations that are related to each other. Therefore, increasing the number of features is not meaningful for the model after a certain point. Pearson Correlation Coefficient was used to sparse the multidimensional feature space obtained by technical indicators. Besides that, various methods have been applied for the representation and analysis of the available data sources. These methods are explained in exploratory data analysis section in Chapter 2.

Table 3.5 Selected technical indicators

Indicator	Description
MACD	Corresponds to the difference between 26-day exponential moving average and 12-day exponential moving average.
Percentage Change	Corresponds to the daily percentage change of stocks.

Table 3.5 continues

ATR	The average of the difference between the highest and lowest value throughout the time series.
EWMA12D	Corresponds to the difference of the close price of the last share corresponding to the 12-day exponentially weighted moving average.
EWMA26D	Corresponds to the difference of the close price of the last share corresponding to the 26-day exponentially weighted moving average.
Momentum	It defines the weekly movement direction of increase and decrease of stocks.
Bollinger Bands	Specifies the measure of how much stock values deviate from the standard deviation and mean.
RSI	Relative Strength Index defines the percentage of the days in which there an increase and decrease in 14-day periods.
Target Signal	Corresponds to the trend signal being defined as 0 and 1 according to the closing price value of MA15's rising and falling patterns for weekly.

3.3 Preprocessing

Preprocessing of raw data before feeding to models often improves the performance of learning models. There are various techniques available for data preprocessing. Preprocessing steps performed on the financial datasets in this study are similar in each of three experiments. In addition, there are preprocessing steps for textual data sources for the third experiment. Data preprocessing steps for experiments will be discussed in this section for news and financial data, separately.

3.3.1 Data Preparation

Preprocessing steps for the stock data in the experiments are similar to each other. Since the stock data are not processed during the public holidays, the sliding windowing method is used. The size of the used lagged window varies according to the forecast horizon. Since the weekly and monthly forecasts were made in the first experiment, the lagged window size was determined as 10 in order to keep its performance at the top level in two cases. Although the window size is kept as 5-day time lags in the second and third experiments, the forecast horizon is different. In the second experiment, forecasting is performed weekly, in the third experiment, it is

performed daily. Since a forecasting with a short-term time lags will be a more realistic assumption for any investor or company, the mainly focus is on models that can achieve more constant results in the shorter term. With this method, in order to obtain the degree of similarity between observations in the defined time interval of stock data, which is also a time series, an analysis was performed with autocorrelation function mentioned in Chapter 2. With this approach, the relationship between past values in the same time interval was measured. The selection of stock series on the different target markets used in the experiments was made in this way. With the obtained features, a z-score normalization was applied to the features to obtain more consistent results in a given range. When the forecasting is realized with trained models, all datasets are de-normalized again to make forecasting consistency on the test datasets.

3.3.2 Text Processing

News data has been processed for its intended use. Since articles in the obtained text data have independent content, a content-based filtering was carried out based on the factors that might be related to financial transactions. For this purpose, the data were filtered by production date and headlines on the tree structured json file. Firstly, articles were subjected to a material-based filtering. Here, material lists with influences such as blog, news, editorial, and analysis were selected. In the json hierarchy, the name of the relevant section is found under the parameter "news_desk". Accordingly, headlines with "news_desk" labels such as business, national, world, etc. were parsed to reference for financial transactions. As a result of the preprocessing, 473166 news headlines were obtained for sentiment analysis. Preprocessing steps such as removing the stop words, removing the punctuation marks performed in the text processing were not applied additionally, because they were performed within the VADER sentiment analysis. In addition to this, sentence tokenization has been applied on the text data, since the VADER lexicon to be used has achieved more successful results at the sentence level.

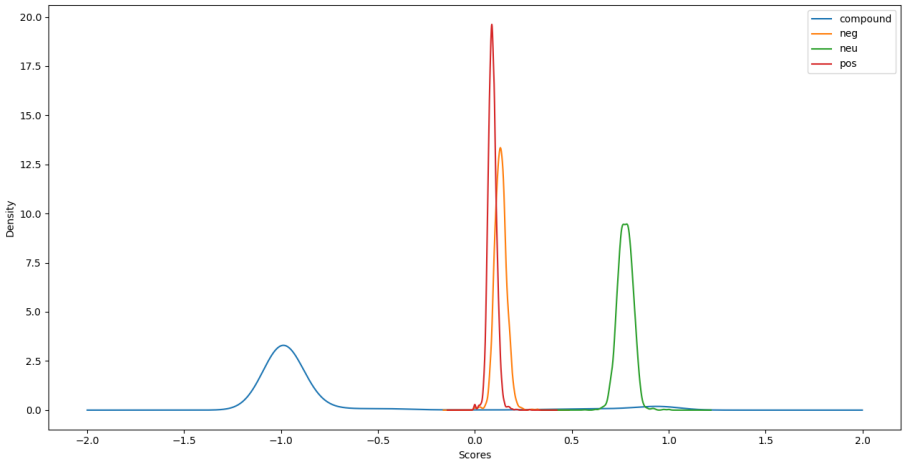

Figure 3.2 The distribution of sentiment polarity

3.3.2.1 Sentiment Analysis

After preprocessing steps for textual datasets, sentiment analysis was performed with the VADER lexicon described in Chapter 2. In addition, 5190 sentence-level snippets from 500 New York Times opinion news articles were used with the VADER lexicon as an improvement for sentiment analysis.

As a result of sentiment analysis with VADER, four different features (compound, positive, neutral, negative) are obtained. Compound score is computed by summing up the valence scores of each word in the lexicon and normalized it in between -1 and 1. The compound score synthesizes the sentiment a given sentence as a vector, which a unidimensional representation. Multidimensional evaluation of sentences in our experiments provided higher forecasting performance on our datasets. Therefore, positive, neutral and negative features were used in the study. The positive, neutral and negative scores were obtained by categorizing the compound value between -1 and 1. Thus, multidimensional sensitivity measurements were made for sentences in processed news data. The distribution of sentiment polarity is presented in Figure 3.2.

Along with the features obtained with sentiment analysis, the datasets were adapted to input requirements of deep learning models. For training of learning models, the sliding window method was used together with 5-day time lags for irregular effects, such as the fact that the stock data were not processed on public holidays, and investors had different trading attitudes. Accordingly, datasets were packaged in three-dimensional tensors including window size, number of instances, and features. For testing of models, we used our trained models and performed daily forecasting on the test datasets in the third experiment.

3.4 Proposed Models

In the experiments carried out, different solutions were sought independently for various problems. With each experiment, the final models to be proposed in the thesis study have been reached. In the first experiment, various RNN architectures suitable for traditional machine learning models were compared. Performance of the models on datasets with different time intervals is observed and the optimal hyperparameters of the models were presented. In addition to the use of technical indicators as an external data source in the second experiment, the LSTM architecture with the most successful results in the first experiment was developed and compared with the CNN model based on automated feature extraction. Both models are presented with optimal hyperparameters in following sections. In the third experiment, the final models of the thesis study that can make forecasting with external data sources are presented.

3.4.1 Proposed Models in the First Experiment

In the first experiment, we present a comparison of the forecasting performance of traditional machine learning models and deep learning models. Deep learning models developed in the study are RNN variants and comparison of the performance of the RNN model is presented. Hyperparameters of each model used were adjusted in accordance with the problem addressed. Accordingly, Tree Structured Parzen Estimators (TPE) are used to adjust the hyperparameters of deep learning models,

while Grid Search is used for traditional machine learning models. While the hyperparameter tuning with Grid Search is aimed at certainty, the tuning with TPE is aimed lower computational time complexity. Table 3.6 summarizes selected hyperparameters of deep learning-based models.

Table 3.6 Hyperparameters of deep learning models (Site, Birant, & Işık, 2019)

Model	Hyperparameters
LSTM	Hidden Layer (1): 32 Cells, activation function: linear Dropout Layer (0.2 ratio) Hidden Layer (2): 16 Cells, activation function: linear Dropout Layer (0.2 ratio) Dense Layer: 1 Neuron, activation function: linear Adam optimizer for Stochastic Gradient Descent (SGD) with 0.002 learning rate
GRU	Hidden Layer (1): 32 Cells, activation function: tanh Dropout Layer (0.2 ratio) Hidden Layer (2): 16 Neurons, activation function: tanh Dropout Layer (0.2 ratio) Dense Layer: 1 Neuron, activation function: linear Stochastic Gradient Descent (SGD) with momentum = 0.9
Basic RNN	Hidden Layer (1): 32 Neurons, activation function: tanh Dropout Layer (0.2 ratio) Hidden Layer (2): 16 Neurons, activation function: tanh Dropout Layer (0.2 ratio) Dense Layer: 1 Neuron, activation function: linear RMSProp optimizer for Stochastic Gradient Descent (SGD) with 0.002 learning rate

Deep learning-based models were evaluated with their own hyperparameters. Layer and hyperparameter specifications of deep learning models will be explained in the proposed models of the second and third experiments. Although all deep learning models have different number of parameters with each other, the same number of layers and cells are used to create similar capacities. Even if each model has similar network depth, they have different learning rate, gradient optimizer, and various hyperparameters. Especially since the Basic RNN structure suffers from vanishing gradients problem mentioned in Chapter 2, the learnig rate has been left as low as possible. The performance of three different RNN architectures were compared with hyperparameters in Table 3.6. In addition to the developed models, deeper network structures have been tested in subsequent experiments with external data sources and different feature sets. Richer datasets and external data sources are more suitable for deeper network architectures. Additionally, some linear and non-linear traditional machine learning algorithms were used and their hyperparameters tuned with Grid Search algorithm. The forecasting performances of these models were compared with different number of observations and different time intervals. It

has been questioned to what extent linear machine learning models such as Linear Regression and Ridge Regression can model the trend especially. Also, all of our datasets contain positive trend and linearity. So, as the data size of linear machine learning algorithms (Linear Regression (LR), Ridge Regression (RR)) increases, there is a chance to observe performance changes. Additionally, Support Vector Regression (SVR), a non-linear machine learning algorithm, has been compared with both linear machine learning algorithms and other RNN architectures in the experiment. Table 3.7 summarizes hyperparameters of traditional machine learning models. Mean Squared Error (MSE), Mean Average Error (MAE), and Mean Absolute Percent Error (MAPE) metrics were used to evaluate the forecasting accuracy of all developed models.

Table 3.7 Hyperparameters of traditional machine learning models (Site, Birant, & Işık, 2019)

Model	Hyperparameters
SVR	C': 1.5, 'epsilon': 0.1,' gamma': 1e-07, 'kernel': polynomial, 'degree': 3
Linear Regression	With default parameters. (used for baseline model)
Ridge Regression	'alpha': 1.0, 'copy_X: True, 'fit_intercept': True, 'max_iter': None, 'solver':auto, 'tol': 0.001

3.4.2 Proposed Models in the Second Experiment

The models created in the second experiment are designed to achieve the best generalization performance on the datasets used. After the models were created, the hyper-parameters of models were tried to be optimized by using the TPE to improve the forecasting performance. The models developed in this experiment constitute the primitive form of the final models to be proposed at the end of thesis study.

3D tensors obtained after preprocessing steps are given as input to the designed CNN model. The received input is passed through the 1D convolution layer. Depending on the number of features in the datasets, it is not enough to perform convolution in a single layer to obtain reliable results. Increasing the number of

convolution layers here causes complicated computation and mentioned vanishing gradients problem. As a result of the observations made, the most reliable results were obtained with two convolution layers. In addition, since the problem addressed is not linear; a leaky rectified linear unit (Leaky ReLU) activation function (Andrew, Awni, & Andrew, 2013) is used to break the linearity and prevention for vanishing gradients problem. This activation function creates an additional partial solution to the problem of faster convergence. In the last stage of the convolution process, the maximum pooling layer was used. As the maximum polling layer performs subsampling, it contributes to the model being more robust to external effects. The feature map obtained as the output is given to a separate layer of fully connected neural network for interpretation. Nadam optimizer was used together with the backpropagation algorithm for training of the model.

The Peephole LSTM model works with 3D tensors like the CNN model. The input is processed as a sequence of data with the time in the model. Through the peephole in the LSTM, the model takes into account the content of the information in the memory throughout the model. For faster convergence, batch normalization is used. The output from Peephole LSTM was flattened and supported by a separate layer of fully connected neural network. The series obtained as output with the Peephole LSTM training was finally interpreted through this layer. Hyperparameters of developed models are presented in Table 3.8.

Table 3.8 Hyperparameters of developed deep learning models (Site & Işık, 2020)

Model	Hyperparameters
CNN	Convolution Layer (1): 128 units, 7 grid LeakyReLU activation function Convolution Layer (2): 256 units, 7 grid LeakyReLU activation function Max Pooling Layer (1) Dropout Layer (0.2 ratio) Fully Connected Layer: 256 units Fully Connected Layer: 1 unit, linear activation function Optimizer: Nadam

Table 3.8 continues

Peephole LSTM	LSTM Layer: 64 units, recurrent normalizer: L2 (0.01) Batch Normalization Dropout Layer (0.3 ratio) Fully Connected Layer: 128 units Batch Normalization Dropout Layer (0.3 ratio) Fully Connected Layer: 1 unit, linear activation function Optimizer: Nadam with 0.002 learning rate ve 0.2 gradient clipping

3.4.3 Proposed Models in the Third Experiment

3.4.3.1 Proposed LSTM

We have developed an optimal LSTM model to forecast daily stock prices using with sequential features over our expanding feature space with the new features added with sentiment analysis and technical indicators. The structure of the model consists of two stacked LSTM layers and two fully-connected layers (FCL). FCL part of the network is responsible for interpreting of the sequential feature sets with long-term dependencies extracted by LSTM layers non-linearly. LSTM layers have 128 and 256 hidden nodes in the first layer and second layer, respectively. The weight matrices from the LSTM layer are vectorized with a flatten layer before passing through the FCL part. The FCL part of network passes weights to the output layer with 128 neurons. The Leaky RELU activation function is used for non-linearity between layers (Andrew, Awni, & Andrew, 2013). In addition, since the number of observations in the stock series are limited, deep learning models face with the overfitting problem. Dropout layers added between the layers inject certain amounts of noise into the weights and help to overcome the overfitting problem. Dropout probability is set to 0.2 in dropout layers. LSTM model was trained with Nadam optimizer which is Adam optimizer with nesterov momentum and 0.001 learning rate. Before training, the parameters of the model were initialized with the Xavier initializer. The selection of hyperparameters such as optimizer, learning rate, batch size, dropout probability and number of hidden neurons was performed with the Tree-Structured Parzen Estimators discussed in Chapter 2. Hyperparameters to form the structure of the model are illustrated in Figure 3.3.

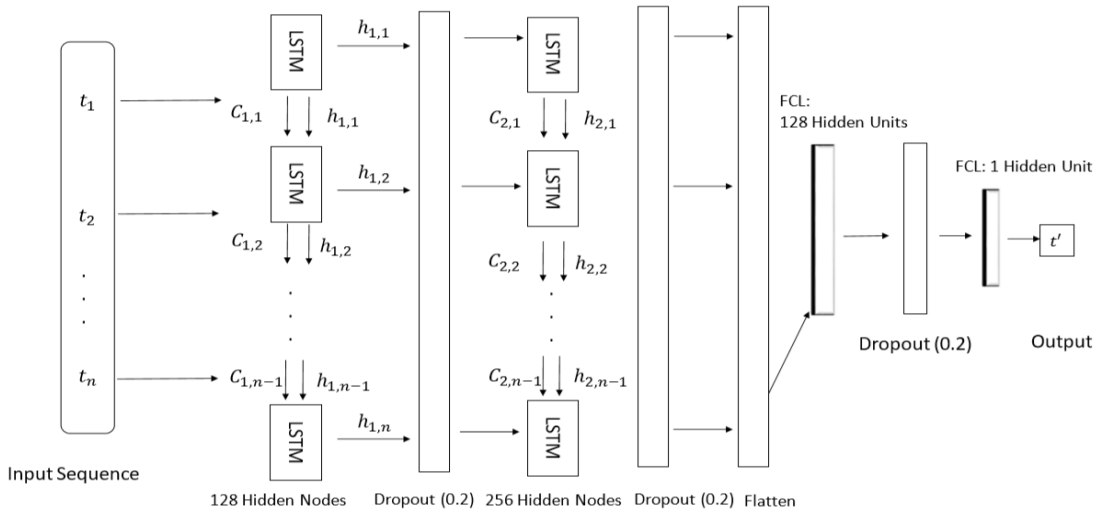

Figure 3.3 The architecture of our LSTM Model

3.4.3.2 Proposed CNN

We have developed an optimal CNN model for automated feature extraction from feature space. CNNs contain many hyperparameters in their structure. One of the biggest expectations in CNN models is performing Conv1D operation to identify long term dependencies on sequential data. Unlike LSTM model, CNN architecture only focuses on the given input sequence, it does not contain information about past. This approach also helps us to provide a different perspective for understanding dynamics of different stock markets. Final structure of the model is shaped by the hyperparameter selection. The proposed CNN model includes 2 convolutional layers, 2 max pooling layers and 1 FCL. The first convolutional layer has 128 filters with kernel size 2, the second convolutional layer has 256 filters with kernel size 2 and Leaky ReLU activation function is used in convolutional layers. The size of the max pooling layers used after each convolutional layer is equal to 2. Unlike LSTM models, CNN models may encounter the problem of vanishing gradients in solving sequential learning problems. To cope with this problem, the Leaky ReLU activation function has been specifically chosen. In addition, a batch normalization is used on the convolutional layers. Batch normalization ensures a solution to a problem called covariate shift. Covariate shift represents the different distributions of input features between train and test datasets. Since the stock datasets used in the study have

different market dynamics and external changes, this is an example for this situation. For neural network perspective, as weights are updated on convolutional layers in the CNN model, different distributions occur. As a result, the next layer should adapt to the change on the weights. This situation causes slowdown in training of neural network. With the batch normalization layer, the mean and standard deviation of batches are taken and the input of the network is standardized with them. Batch normalization makes the network more robust, because it makes neural network to less affected by the change of structural key points such as, weight initialization, learning rate, and optimizer. Batch normalization also reduces overfitting of model by using empirical means and variances calculated from mini-batches of convolutional layers. After the second pooling layer, weights are vectorized with Flatten layer and created feature vector given as input to FCL containing 256 hidden neurons. FCL is balanced with the dropout layer for regularization. Dropout probability is set to 0.3 in the dropout layer. The selection of hyperparameters such as number of layers, filter sizes, dropout probability and size of fully connected layer was performed with the Tree-Structured Parzen Estimators discussed in Section Chapter 2. Hyperparameters to form the structure of the model are illustrated in Figure 3.4.

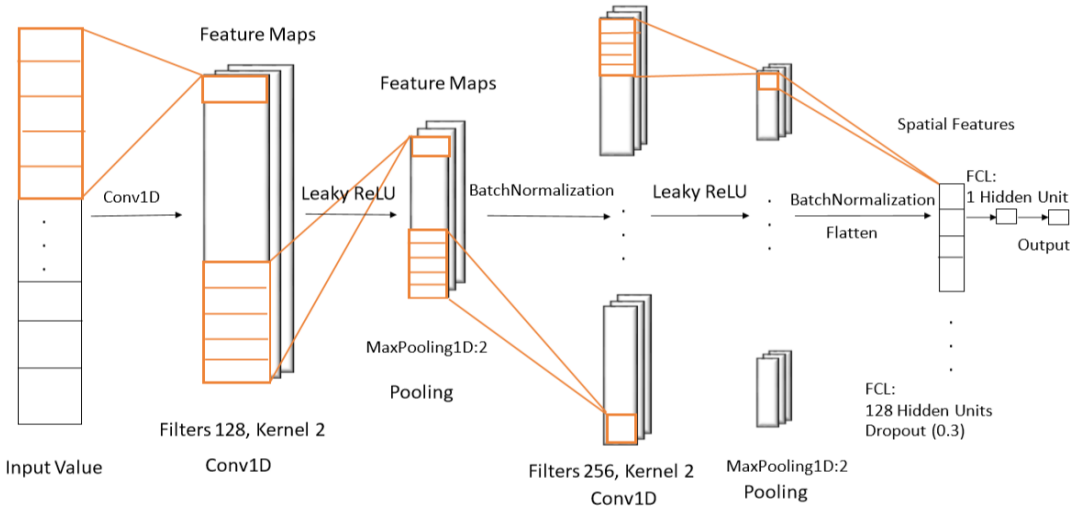

Figure 3.4 The architecture of our CNN Model

3.4.3.3 Proposed Autoencoder

A simple stacked autoencoder (SAE) was developed with stacking of fully connected layers. Increasing the number of layers provides better density estimation for datasets. If the high dimensional feature space obtained with external data sources is qualified as a complex function, the output is desired to be a simple function. Therefore, A 5-layer AE architecture consisting of 4 simple AEs has been developed, where the best performance is achieved. The operating philosophy of the system is no different from a classic FNN as mentioned earlier. First AE maps the features and constructs the first AE input layer to the hidden vector in the code layer. After the training of the first layer is finished, the code layer of the first AE behaves like the input layer of the second AEs. This process continues to the end of the network, covering all stacked layers developed. With this way, the entire network is trained with the stochastic gradient descent (SGD) algorithm till the last layer. In this way, the density estimation between the reconstructed data and the actual dataset is realized with minimal error. In this study, the input dimension of the features varies between 18 and 21 according to the presence of sentiment analysis results in the given dataset. Dimension of the code layers on the network are set to 12. More consistent results were obtained with this structure. Depth plays an important role in the SAE architecture. Increasing the number of layers in the model by a certain level has brought higher performance for feature extraction process. As a result, the depth of the network is determined as 5 layers. After the AE part of the network, feature extraction was made and the features reduced to 12 dimensions were interpreted by the fully-connected layer with 64 hidden units and regression is performed with the last part of network. In addition, the dropout layer containing 0.4 dropout probability was used to avoid overfitting problem. The selection of hyperparameters such as number of layers, dropout probability and depth of the AE was performed with the Tree-Structured Parzen Estimators discussed in Chapter 2. Hyperparameters to form the structure of the model are illustrated in Figure 3.5.

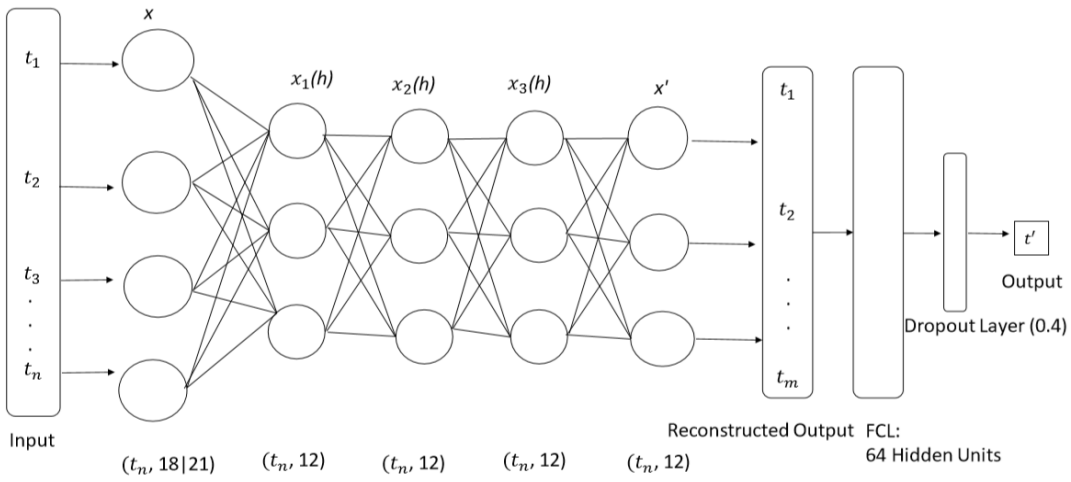

Figure 3.5 The architecture of our SAE Model

3.5 Model Performance Measures

MSE, RMSE, MAE, MAPE and Adjusted R^2 error metrics were used to evaluate the results obtained with the developed learning models during the experiments. MSE, RMSE, MAPE, and MAE can be defined as following formulas,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(f_i - y_i\right)^2 \tag{3.1}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f_i - y_i\right)^2} \tag{3.2}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|f_i - y_i\right| \tag{3.3}$$

$$MAPE = \frac{100\,\%}{N}\sum_{i=1}^{N}\left|\frac{f_i - y_i}{f_i}\right| \tag{3.4}$$

where N represents the number of observations, f_i represents i^{th} observation of actual output value, and y_i represents i^{th} observation of predicted value. Since the number of independent variables in dataset is high, the effects of the independent variables on the models analyzed with Adjusted R^2 metric as an observation tool. The standard R^2

metric can increase to the perfect value of 1 with number of added independent variables. Therefore, it was used in the second and third experiments with rich feature space. It can produce misleading results in comparison of different models. Adjusted R^2 can be defined as following,

$$Adjusted\, r^2 = 1 - \frac{\left(1 - R^2\right)(n-1)}{n-k-1} \tag{3.5}$$

where R^2 represents the standard R^2 metric , n represents the total number of observations, and k represents the number of features.

3.6 Libraries, Functions, and Running Environment

We used scikit-learn (Buitinck et al., 2013), keras (Chollet, 2015)and tensorflow (Abadi et al., 2016) python packages for developing some of the traditional machine learning and deep learning models, NVIDIA Deep Neural Network (cuDNN) (Chetlur, 2014) library as backend of tensorflow for GPU acceleration of proposed deep learning models, hyperopt python module for Bayesian Optimization for proposed deep learning models (Bergstra et al., 2015), NLTK (Loper & Bird, 2002) python library for text processing operations. All executions have been performed on a computer with Intel Core i7-6900K processor, 16 GB of RAM, Geforce GTX 1080TI graphics card and Windows 10 operating system.

CHAPTER FOUR
RESULTS AND DISCUSSIONS

We have followed an integrative approach, which is traded in different target markets and aims to combine multiple data sources and learning methods to successfully forecast stock series from different sectors. Different experiments in various configurations performed and the final proposed models were created. Each experiment carried out provides a technical basis for the next one. The results and discussions obtained from the experiments will be explained in the next subsections.

4.1 Results of the First Experiment

In the first experiment, forecasting performance of the models was measured on two different dataset from different indexes. Forecasting performances with monthly and weekly time intervals are presented in Table 4.1 and 4.2 with MSE, MAE, and MAPE error metrics.

Deep learning models were trained with 300 epochs on datasets. Since deep learning models contain a lot of adjustable parameters, they can quickly overfitting stock series with a certain time interval. Even if the developed deep neural network architecture is simple, the datasets contain limited number of observations like stock series, and especially in high number of epochs, overfitting problem is encountered. Increasing the network depth does not provide a gains to deep learning models without enriching the feature space. During the first experiment, deep learning models achieved more succesful predictions with the increase in the number of observations and time intervals. On the other hand, traditional machine learning models made more successful predictions with fewer observations and narrower time intervals. Based on this, successful forecasts were made with appropriate observations and appropriate time intervals. Since the stock market data has very dynamic behavior, the time period covered should not be very wide. A narrower time interval is a more realistic scenario for the market and investors, so datasets covering

a very wide time interval may be insufficient to capture the fluctuations of stock series.

Additionally, the first experiment includes an evaluation of the weekly and monthly forecast horizon. Deep learning models were not affected much by the change in the forecast horizon interval. Although SVR, which is a non-linear model, calculates optimal solutions globally; achieved the worst results for weekly and monthly predictions. We think that noise and fluctuations in the datasets constitute this situation. With different datasets and configurations, SVR can achieve better performance.

Among the developed deep learning models, LSTM and GRU models achieved the highest performance. Since LSTM and GRU models perform sequential learning, capturing long-term dependencies in the datasets is very important for effective forecasting. Since the problem of vanishing gradients was encountered during the modeling of long-term dependencies, the Basic RNN architecture lagged far behind other RNN variants in some cases. The performances of the weekly and monthly forecasting realized by the models in the experiment are presented in Figure 4.1 and Figure 4.2. While the LSTM model and GRU models achieved similar forecasting performance, the GRU model runs much faster than the LSTM model in training process. As can be seen in the results in the AMZN dataset, the GRU model showed lower performance than LSTM in cases where there were excess fluctuations. The occurrence of this situation is mainly due to the fact that the LSTM model contains more complex memory than the GRU model.

Table 4.1 Weekly prediction results of all models (Site, Birant, & Işık, 2019)

Dataset	**GOOGL**	**AMZN**
LSTM	MSE:0.027 MAE:0.013 MAPE:0.018	MSE:0.025 MAE:0.013 MAPE:0.019
GRU	MSE:0.026 MAE:0.014 MAPE:0.17	MSE:0.026 MAE:0.0145 MAPE:0.017

Table 4.1 continues

Basic RNN	MSE:0.023 MAE:0.012 MAPE:0.015	MSE:0.028 MAE:0.016 MAPE:0.020
SVR	MSE:0.16 MAE:0.28 MAPE:0.81	MSE:0.036 MAE:0.17 MAPE:0.24
LR	MSE:0.0033 MAE:0.0087 MAPE:0.0153	MSE:0.0016 MAE:0.0085 MAPE:0.013
RR	MSE:0.037 MAE:0.0094 MAPE:0.017	MSE:0.0037 MAE:0.014 MAPE:0.021

Table 4.2 Monthly prediction results of all models (Site, Birant, & Işık, 2019)

Dataset	**GOOGL**	**AMZN**
LSTM	MSE:0.0247 MAE:0.125 MAPE:0.169	MSE:0.023 MAE:0.011 MAPE:0.018
GRU	MSE:0.025 MAE:0.127 MAPE:0.17	MSE:0.024 MAE:0.016 MAPE:0.019
Basic RNN	MSE:0.028 MAE:0.17 MAPE:0.184	MSE:0.024 MAE:0.014 MAPE:0.0194
SVR	MSE:0.12 MAE:0.24 MAPE:0.72	MSE:0.033 MAE:0.16 MAPE:0.238
LR	MSE:0.00132 MAE:0.0078 MAPE:0.0128	MSE:0.0015 MAE:0.0082 MAPE:0.012
RR	MSE:0.0022 MAE:0.016 MAPE:0.0145	MSE:0.0044 MAE:0.014 MAPE:0.022

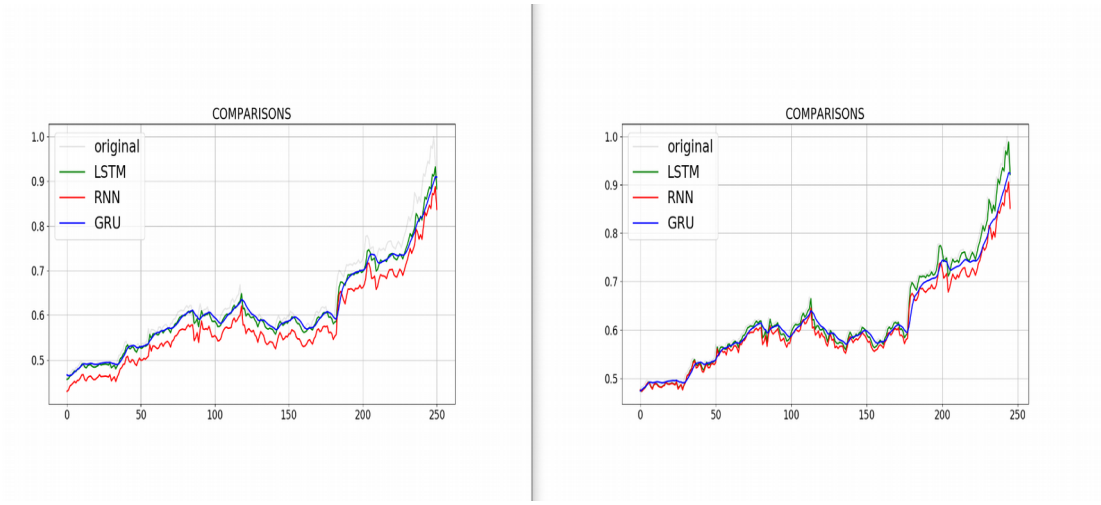

Figure 4.1 Weekly (left) and Monthly (right) forecasting results of the AMZN dataset

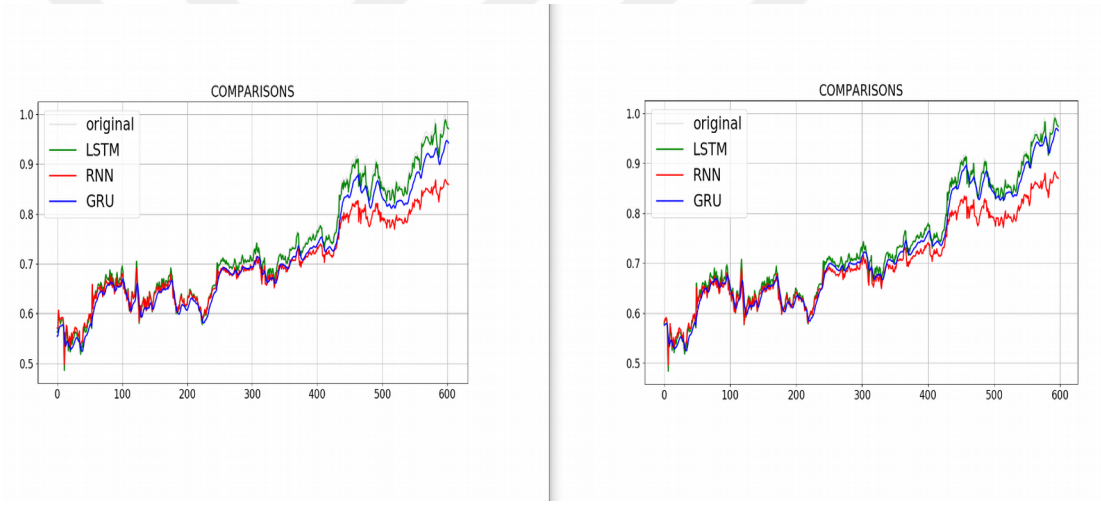

Figure 4.2 Weekly (left) and Monthly (right) forecasting results of the GOOGL dataset

4.2 Results of the Second Experiment

The models were evaluated on AAPL, NVDA, and GOOG stocks independently. Both models (Peephole LSTM and CNN) were trained with 300 epochs on the training datasets. Contrary to the first experiment, only weekly forecasts were made. While evaluating the weekly forecast performance, mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) and Adjusted R^2 error metrics were used. Values of error metrics are presented in Table 4.3.

Table 4.3 Prediction results of all models (Site & Işık, 2020)

Dataset	AAPL	NVDA	GOOG
Peephole LSTM	RMSE:0.1977 MSE:0.0391 MAE:0.1590 Adjusted R^2: 0.9624	RMSE:0.3330 MSE:0.1109 MAE:0.2620 Adjusted R^2: 0.9892	RMSE:0.1433 MSE:0.0205 MAE:0.1130 Adjusted R^2:0.9082
CNN	RMSE:0.1999 MSE:0.0399 MAE:0.1602 Adjusted R^2: 0.9247	RMSE:0.3783 MSE:0.1431 MAE:0.3011 Adjusted R^2: 0.9720	RMSE:0.1554 MSE:0.024 MAE:0.1227 Adjusted R^2: 0.9483

Since the number of independent features on the data was high, the effects of independent features on the models were corrected with the R^2 metric. Even though the independent variables added with the R^2 metric are not related, they can approach the perfect value 1 uncontrollably. With the adjusted R^2 metric, the effect of independent variables on the dependent variable can be observed more consistently. MSE, RMSE and MAE error metrics statistically measure the difference between the predicted value and the actual value. The lower these values, the more consistent the forecast is. Figure 4.3 shows plots of predictions made for all three datasets.

The developed models achieved predictions with more consistency and fewer errors on the GOOG dataset. The predictions made on the AAPL dataset deviate from actual values. Peephole LSTM's predictive performance is better than CNN in test data. It has been observed that the consistency of forecasting over the NVDA dataset is not as good as the other two stocks. Since the CNN model extracts the relevant information required for the forecasting via the dataset, a significant improvement can be seen in the forecasting performance with a richer feature space. NVDA dataset contains much more noise and fluctuations than others. This is the main factor in the decline of the performance of the models. Peephole LSTM captures rarely changes on stocks better than the regular LSTM model. However, as a result of our observations, the regular LSTM architecture captures the trend of stocks better than the Peephole LSTM. Therefore, Peephole LSTM architecture was not used in the third experiment.

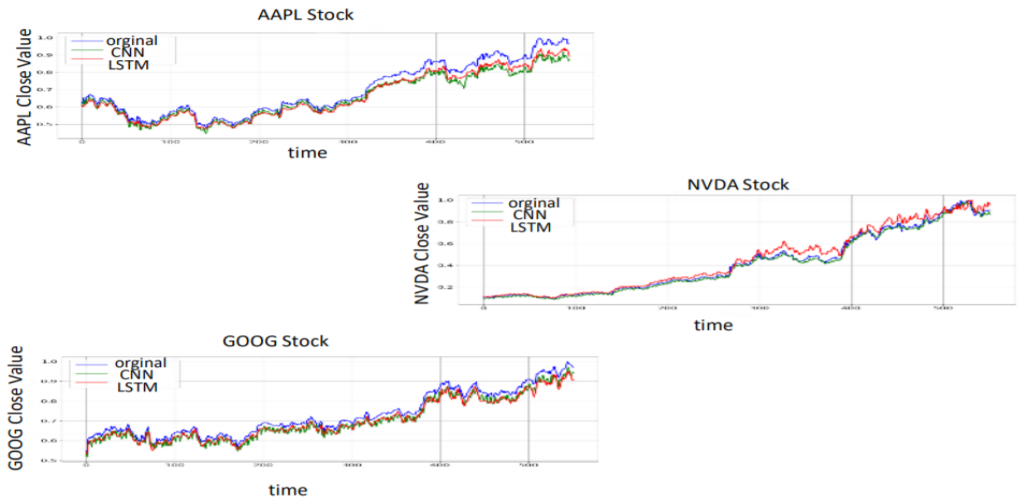

Figure 4.3 Weekly prediction results of all three datasets

4.3 Results of the Third Experiment

In this experiment, we used three different deep learning architectures for stock price forecasting on NASDAQ and NYSE stock markets. Models are trained with 300 epochs on training datasets. RMSE, MSE, MAE, and Adjusted R^2 error metrics were used to evaluate the daily forecasting performance of learning models. The values of obtained error metrics are presented in Table 4.4. Since the features on the datasets were normalized before they were fed for training, the data was also de-normalized to make a one-to-one comparison in experiments performed to measure the forecasting consistency on the test data. Figure 4.4 contains forecasting plots of stock data traded on the NASDAQ target market, while Figure 4.5 contains forecasting plots of the NYSE target market.

Table 4.4 Prediction results of all models in the final experiment.

	Only Technical Indicators			**Technical Indicators Combined with Sentiment Analysis**		
Datasets	**CNN**	**LSTM**	**AE**	**CNN**	**LSTM**	**AE**
AAPL	RMSE:0.2013 MSE:0.0405 MAE:0.1614 Adjusted R^2: 0.9723	RMSE:0.2051 MSE:0.0421 MAE:0.1650 Adjusted R^2: 0.6979	RMSE:0.1852 MSE:0.03430 MAE:0.1500 Adjusted R^2: 0.7803	RMSE:0.2004 MSE:0.0401 MAE:0.1607 Adjusted R^2: 0.9335	**RMSE:0.2062** **MSE:0.0425** **MAE:0.1660** **Adjusted R^2: 0.8367**	**RMSE:0.2207** **MSE:0.0487** **MAE:0.1764** **Adjusted R^2: 0.9522**

Table 4.4 continues

AMD	RMSE:0.3216 MSE:0.1034 MAE:0.2596 Adjusted R^2: 0.9922	RMSE:0.3247 MSE:0.1054 MAE:0.2622 Adjusted R^2: 0.9945	RMSE:0.3159 MSE:0.0997 MAE:0.2559 Adjusted R^2: 0.9848	**RMSE:0.3244** **MSE:0.1052** **MAE:0.2617** **Adjusted R^2: 0.9934**	RMSE:0.3177 MSE:0.1009 MAE:0.2571 Adjusted R^2: 0.9885	RMSE:0.3171 MSE:0.1006 MAE:0.2569 Adjusted R^2: 0.9812
CSCO	RMSE:0.1684 MSE:0.0283 MAE:0.1328 Adjusted R^2: 0.9089	RMSE:0.1660 MSE:0.0275 MAE:0.1311 Adjusted R^2: 0.9357	RMSE:0.1731 MSE:0.0299 MAE:0.1373 Adjusted R^2: 0.9513	RMSE:0.1674 MSE:0.0280 MAE:0.1322 Adjusted R^2: 0.8822	**RMSE:0.1651** **MSE:0.0272** **MAE:0.1303** **Adjusted R^2: 0.9608**	**RMSE:0.1722** **MSE:0.0296** **MAE:0.1364** **Adjusted R^2: 0.9680**
INTC	RMSE:0.1417 MSE:0.0201 MAE:0.1042 Adjusted R^2: 0.9845	RMSE:0.1405 MSE:0.0197 MAE:0.1049 Adjusted R^2: 0.8921	RMSE:0.1369 MSE:0.01874 MAE:0.1026 Adjusted R^2: 0.9064	**RMSE:0.1435** **MSE:0.0206** **MAE:0.1058** **Adjusted R^2: 0.9851**	**RMSE:0.1397** **MSE:0.0195** **MAE:0.1037** **Adjusted R^2: 0.9387**	RMSE:0.1342 MSE:0.0180 MAE:0.0997 Adjusted R^2: 0.8321
KO	RMSE:0.0825 MSE:0.0068 MAE:0.0674 Adjusted R^2: 0.9556	RMSE:0.1006 MSE:0.0101 MAE:0.0828 Adjusted R^2: 0.6590	RMSE:0.0811 MSE:0.0065 MAE:0.0661 Adjusted R^2: 0.9550	**RMSE:0.0819** **MSE:0.0067** **MAE:0.0667** **Adjusted R^2: 0.9569**	**RMSE:0.0869** **MSE:0.0075** **MAE:0.07107** **Adjusted R^2: 0.8798**	**RMSE:0.0816** **MSE:0.0066** **MAE:0.0666** **Adjusted R^2: 0.9601**
AXP	RMSE:0.1841 MSE:0.0339 MAE:0.1486 Adjusted R^2: 0.9727	RMSE:0.1759 MSE:0.0309 MAE:0.1416 Adjusted R^2: 0.9686	RMSE:0.1790 MSE:0.0320 MAE:0.1441 Adjusted R^2: 0.9949	**RMSE:0.1850** **MSE:0.0342** **MAE:0.1492** **Adjusted R^2: 0.9899**	**RMSE:0.1778** **MSE:0.0316** **MAE:0.1431** **Adjusted R^2: 0.9895**	**RMSE:0.1810** **MSE:0.0327** **MAE:0.1458** **Adjusted R^2: 0.9968**
BA	RMSE:0.2416 MSE:0.0583 MAE:0.1833 Adjusted R^2: 0.9626	RMSE:0.2266 MSE:0.0513 MAE:0.1698 Adjusted R^2: 0.8729	RMSE:0.2151 MSE:0.0462 MAE:0.1594 Adjusted R^2: 0.7569	**RMSE:0.2361** **MSE:0.0557** **MAE:0.1800** **Adjusted R^2: 0.9825**	**RMSE:0.2290** **MSE:0.0524** **MAE:0.1737** **Adjusted R^2: 0.8981**	**RMSE:0.2154** **MSE:0.0464** **MAE:0.1607** **Adjusted R^2: 0.8164**
GS	RMSE:0.2346 MSE:0.0550 MAE:0.1892 Adjusted R^2: 0.9653	RMSE:0.2341 MSE:0.0548 MAE:0.1883 Adjusted R^2: 0.9827	RMSE:0.2278 MSE:0.05190 MAE:0.1849 Adjusted R^2: 0.9270	RMSE:0.2350 MSE:0.0552 MAE:0.1898 Adjusted R^2: 0.9495	RMSE:0.2324 MSE:0.0540 MAE:0.1873 Adjusted R^2: 0.9810	**RMSE:0.2248** **MSE:0.0505** **MAE:0.1835** **Adjusted R^2: 0.9446**
IBM	RMSE:0.1248 MSE:0.0155 MAE:0.0983 Adjusted R^2: 0.9935	RMSE:0.1217 MSE:0.0148 MAE:0.0960 Adjusted R^2: 0.9820	RMSE:0.1291 MSE:0.0166 MAE:0.1017 Adjusted R^2: 0.9913	**RMSE:0.1238** **MSE:0.0153** **MAE:0.0976** **Adjusted R^2: 0.9946**	**RMSE:0.1212** **MSE:0.0146** **MAE:0.0955** **Adjusted R^2: 0.9837**	**RMSE:0.1270** **MSE:0.0161** **MAE:0.1001** **Adjusted R^2: 0.9954**
JPM	RMSE:0.2092 MSE:0.0438 MAE:0.1689 Adjusted R^2: 0.8833	RMSE:0.2118 MSE:0.0448 MAE:0.1732 Adjusted R^2: 0.8661	RMSE:0.2096 MSE:0.0440 MAE:0.1677 Adjusted R^2: 0.9062	**RMSE:0.2122** **MSE:0.0450** **MAE:0.1703** **Adjusted R^2: 0.9885**	RMSE:0.2107 MSE:0.0444 MAE:0.1702 Adjusted R^2: 0.8358	**RMSE:0.2110** **MSE:0.0445** **MAE:0.1688** **Adjusted R^2: 0.9296**

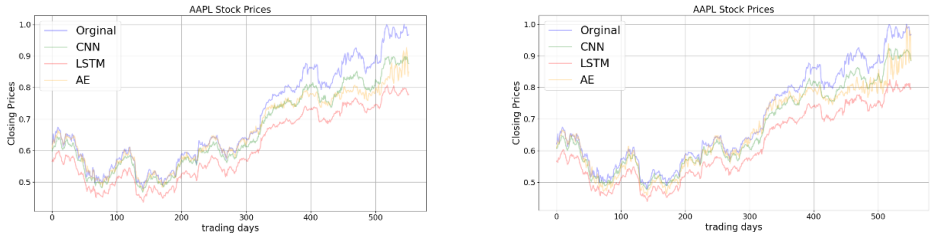

Figure 4.4 Actual and forecasted values of five stocks from NASDAQ target market by using proposed models. For each row, the models on the left plots are using only technical indicators, the ones on the right plots are using both technical indicators combined with sentiment analysis

Figure 4.4 continues

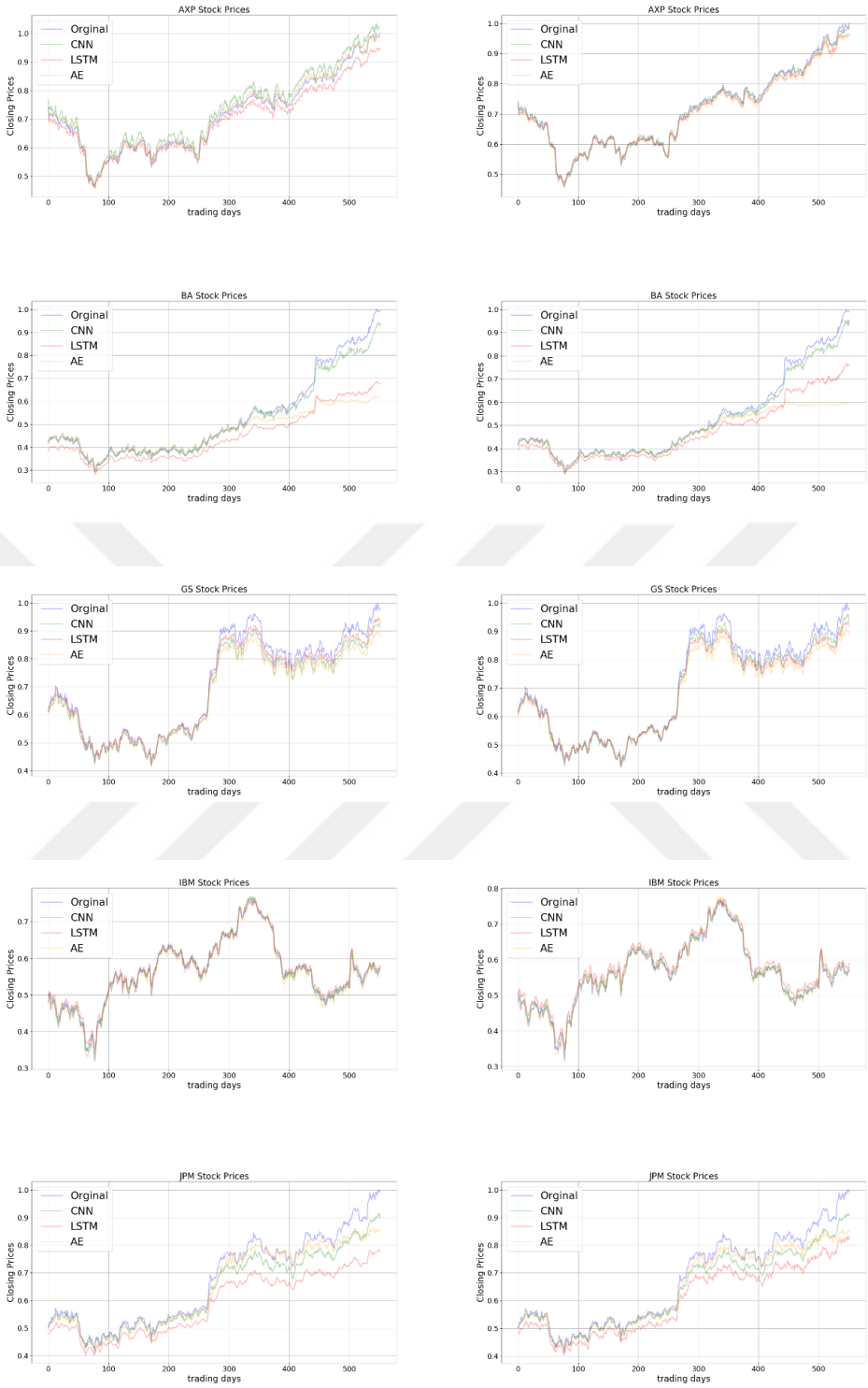

Figure 4.5 Actual and forecasted values of five stocks from NYSE target market by using proposed models. For each row, the models on the left plots are using only technical indicators, the ones on the right plots are using both technical indicators combined with sentiment analysis

As can be seen with the error metrics obtained, the CNN model has generally achieved the lowest error values on both target-markets. When sentiment analysis is not performed, the obtained results are less different than others, since the CNN model does not use historical information for forecasting. The data sources enriched with the added polarity scores provided a slight improvement in overall model performance. In addition, the variance of the results obtained with the CNN model is lower than the other models in this study. This observation can be interpreted as the model is more robust to external effects. In AXP and BA datasets, this situation can be observed more than other datasets in the study.

On the other hand, while making forecasting about stock data with the LSTM model, the long-term dependencies on the datasets, which uses information about the past, have been able to model effectively. Polarity scores obtained as a result of sentiment analysis have a similar trend with stock data. Sentiment analysis has significantly increased forecasting performance of the LSTM model on KO and AAPL stocks traded in the NASDAQ target market. Apart from that, the LSTM model has generally achieved higher forecasting performance gains with the extended features. Since the LSTM model contains more parameters, factors such as the use of historical information about the past observations and high dimensional feature space, which directly affect forecasting performance of the model. As clearly seen in the AAPL dataset, this led a better forecasting of the stock price trend. Conversely, the LSTM model becomes more fragile against fluctuations and residual effects on the datasets compared to the CNN model. This is clearly observed on the BA dataset.

Additionally, with the SAE model, a serious performance improvement was also obtained by sentiment analysis on the AAPL dataset. This is a good example of the synthesis of external data sources, such as different economic factors that are often used for long-term forecasting in the field of finance. The model successfully extracted hidden spatial features from the input resources. With the extracted polarity scores, an increase was achieved in both forecasting performance of different target

markets and independent stock prices. However, considering the automated feature extraction part, the CNN model shows much more consistent results. The performance of the SAE model on datasets such as AAPL, INTC, and BA is not suitable for generalization of forecasting problem.

Another point that needs to be questioned is how different markets have financial dynamics. After applying the sentiment analysis, the forecasting performance of almost all stock data traded in the NYSE target market with different deep learning models has increased. Models, which can extract relevant information from dataset instantly through the given time window, seem advantageous for forecasting. This ensures better modelling of data points with fluctuations and residual effects. This observation can be clearly identified on INTC and BA datasets. The CNN model has fewer shifts in trend than other models in the study. Instant changes in the stock data of different companies were captured more successfully with the CNN model. Therefore, the results obtained with the CNN model are more generalizable due to lower variances of the experiments on different markets and datasets.

CHAPTER FIVE
CONCLUSION AND FUTURE WORK

In this thesis study, stock price close values were tried to be predicted for a certain period of time. Along with this, different models and techniques developed during the workflow of the study, and the behavior of stock series and external factors were tried to be captured. Three different experiments performed in this direction complementary to each other. Different models and analysis techniques have been interpreted under different titles and shaped the forecasting performance of the final models. In this context, the final experiment covers the presentation of the technical and fundamental analysis carried out throughout the study along with the final models to be proposed for the solution of the problem. Each achievement after the experiments was put forward independently and helped shape the final workflow. Thus, it was ensured that the study objectives were achieved.

The first experiment provides an assessment of the forecasting performance of different machine learning models for two different stock datasets. This is an important step in the selection of learning models to be used in the next experiments of the study. Obtained results revealed that better forecast performance can be obtained in deeper RNN models. Also, this experiment showed comparison between RNN variants and presentation of specific properties of models. LSTM networks provided better forecasting performance compared to other models with similar network depth. Apart from high computational workload, LSTM was the most successful model especially in capturing the trend both weekly and monthly basis. Weekly forecasting problem is quite insufficient for traditional machine learning models due to effects on holidays. Different external effects are observed in stock prices after public holidays. Besides, monthly forecasting has achieved much better performance compared to weekly ones with traditional machine learning models. However, these results are extremely data-dependent. LSTM and GRU models are less affected by changes in the forecast horizon and the fluctuations in the dataset than traditional machine learning algorithms.

The second experiment, the close values of three different stocks traded on NASDAQ index are forecasted weekly. The performance of presented two different deep learning models (Peephole LSTM, CNN) examined. Various technical indicators and independent variables were calculated for more consistent results obtained on high dimensional feature space with developed deep learning models. Based on these results, the final experiment was carried out with the use of additional external data sources and more advanced deep learning architectures and various methods.

In the final experiment, we used three different deep learning architectures with different approaches for stock price forecasting on NASDAQ and NYSE stock markets. In order to analyze the financial time series, the relevant features for forecasting of stock prices have been extracted with deep learning architectures on external data sources. Accordingly, the results of sentiment analysis on news articles, technical indicators, and open, high, low, close (OHLC) values of 10 different stocks traded in NASDAQ and NYSE stock markets, are used as inputs for deep learning models. In this way, we aimed to experiment the generalizability of the results for the data from different sectors and stock markets. The developed models have been tuned to offer the best forecasting performance on datasets with the Bayesian optimization. A comparison of common architectures used for stock forecasting is presented. As a result, the proposed models have advantageous and disadvantageous points for forecasting task. Long-term dependencies on datasets have been successfully captured by the LSTM model. In almost all datasets, trend of stock data was modeled with the proposed LSTM model. The greatest increase of forecasting performance by including sentiment analysis to the stock datasets was achieved by the LSTM model. In a way, this observation reveals the relationship between sentiment analysis and stock market. Since, polarity results obtained as a result of sentiment analysis have a similar trend with stock data.

The LSTM model, which uses information from past, has significant performance increases in forecasting on some datasets. In general, automated feature extraction based on CNN and SAE architectures can forecast stock prices with less error. With

SAE model, high-dimensional feature space is reduced with stacked AE layers, and relevant temporal information extracted by the model. Since the CNN architecture focuses only given input sequence, it has achieved the highest forecasting performance among all models thanks to the rich feature space created. It has been observed that it is more successful than SAE model in automated feature extraction. As mentioned earlier, the LSTM model catches the stock price trend compared to other models, but it is more sensitive to fluctuation and residual effects. The CNN model clearly better coped with such situations. An important reason of this is the use of the sliding windowing approach. This allows the CNN model to understand the dynamical changes and patterns that have occurred in the current window with feature extraction.

As a summary, this study revealed that the hidden dynamics of stocks are successfully captured with the proposed deep learning models. Changes in the stock market may not always be on a regular basis or not follow the certain cycle. In the study, this situation was emphasized by analyzing stock data from different target markets and sectors. In addition, there are key points that will provide performance increases in forecasting studies. Especially for models with feature extraction, using higher frequency data may increase the forecasting performance. Additionally, large financial lexicon can be created and applied for sentiment analysis to provide semantic analysis of factors that have an impact on financial markets. With these factors, we think that a better forecasting performance related to financial problems will be experienced.

REFERENCES

Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D. G., Steiner B., Tucker P., Vasudevan V., Warden P., Wicke M., Yu Y., & Zheng X. Tensorflow: a system for large-scale machine learning. *USENIX Symposium on Operating Systems Design and Implementation (OSDI 16),* 265-283.

Andrew L. M., Awni Y. H., & Andrew Y. Ng. (2013). Rectifier nonlinearities improve neural network acoustic models. *ICML Workshop on Deep Learning for Audio, Speech and Language Processing, 30* (1), 3.

Aras S., & Karakoc I. D. (2016). A new model selection strategy in time series forecasting with artificial neural networks. *Neurocomputing,* (174), 974-987.

Bellman S., & Schmidhuber J. (1957). *Dynamic programming (*1st ed.*)*. Princeton, NJ, USA: Princeton University Press

Bergstra J., Yamins D., & Cox D. D. (2013). Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. *30th International Conference on Machine Learning* (ICML 2013), (28), 115-123.

Buitinck L., Louppe G., Blondel M., Pedregosa F., Mueller A., Grisel O., Niculae V., Prettenhofer P., Gramfort A., & Grobler J. (2013). API design for machine learning software: experiences from the scikit-learn project. *European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases*, Prague, Czech Republic, 108-122.

Chetlur S., Woolley C., Vandermermersch P., Cohen J., Tran J., Catanzaro B., & Shelhamer E. (2014). cuDNN: efficient primitives for deep learning. *Computer Science, ArXiv,* 1-9.

Cho K., Chung J., Gulcehre C., & Bengio Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modelling. *NIPS 2014 Workshop on Deep Learning,* 1-9.

Chollet F. (2015). Keras. Github. Retrieved June 6, 2020 from https://github.com/fchollet/keras.

Cortes C., & Vapnik V. (1995). Support-vector networks. *Machine Learning,* 20(3), 273-297.

Dong J., Dai W., Liu Y., Yu L., & Weng J. (2019). Forecasting Chinese stock market prices using Baidu search index with a learning-based data collection method. *International Journal of Information Technology & Decision Making,* (18), 1605-1629.

Drucker H., Burges C. J. C., Kaufman L., Smola A. J., & Vapnik V. (1996). Linear support vector regression machines. *Advances in Neural Information Processing Systems,* (9), 155-161.

Elman J., L. (1990). Finding structure in time. *Cognitive Sciences,* (14), 179-211.

Fama, E. F. (1970). Efficient capital markets: a review of theory and empirical work. *The Journal of Finance,* (25), 383-417.

Gers F., Schraudolph N. N., & Schmidhuber J. (2003). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research, 3*(1), 115-143.

Goodfellow I., Begio Y., & Courville A. (2016). *Deep learning (*1st ed.*).* 499-523. Cambridge, Massachusetts: MIT Press.

Gündüz H., Çataltepe Z., & Yaslan Y. (2017). Stock daily return prediction using expanded features and feature selection. *Turkish Journal of Electrical Engineering & Computer Sciences*, *25*(6), 4829-4840.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, (57), 357-384.

Hegazy, O., Soliman O. S., & Salam M. A. (2013). A machine learning model for stock market prediction. *International Journal of Computer Science and Telecommunications*, (4), 17-23.

Hiransha, M., Gopalakrishan E. A., Menon V. K., & Soman K. P. (2018). NSE stock market prediction using deep-learning models. *International Conference on Computational Intelligence and Data Science*, (132), 1351-1362.

Hochreiter S., & Schmidhuber J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780.

Hochreiter S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, (6), 107-116.

Hoseinzade E., & Haratizadeh S. (2019). CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications,* (129), 273-285.

Hutto C. J., & Gilbert E. E. (June). VADER: A parsimonious rule-based model for sentiment analysis of social media text. *In Eight International Conference on Weblogs and Social Media* (*ICWSM-14*), *81*, 82.

Ilievski I., Akhtar T., Feng J., & Shoemaker C. A. (2017). Efficient hyperparameter optimization of deep learning algorithms using deterministic RBF surrogates. *Thirty first AAAI Conference on Artificial Intelligence*, 822-829.

Kara, Y., Boyacioglu M., & Baykan M. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul stock exchange. *Expert Systems with Applications*, (38), 5311-5319.

Karush W. (1939). *Minima of functions of several variables with inequalities as side conditions*. Master's Thesis*,* Department of Mathematics, University of Chicago.

Kim T., & Kim H. Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PLoS ONE,* (14)2.

Kuhn H. W., & Tucker A. W. (1951). Nonlinear programming. *Second Berkley Symposium on Mathematics, Statistics and Probability*, (72), 481-492.

Long W., Lu Z., & Cui L. (2018). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems,* (164), 163-173.

Loper E., & Bird S. (2002). NLTK: the natural language toolkit. *ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics,* 214-217.

Mohamed M. (2010). Forecasting stock exchange movements using neural networks:empirical evidence from Kuwait. *Expert Systems with Applications*, (37), 6302-6309.

Musto C., Polignano M., & Semeraro G. (2014). A comparison pf lexicon-based approaches for sentiment analysis of microblog. *XIII AI*IA Symposium on Artificial Intelligence* (AI*IA 2014), 59-68.

Nayak, A., Pai M. M. M., & Pai R. M. (2016). Prediction models for Indian stock market. *Twelfth International Multi-Conference on Information Processing* (IMCIP), 441-449.

New York Times Developer Network. (2020). Retrieved June 6, 2020 from https://developer.nytimes.com/

Patel J., Shah S., Thakkar P., & Kotecha K. (2015). Predicting stock and stock movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications,* (42), 259-268.

Qiu M., Song Y., & Akagi F. (2016). Application of artificial neural network for the prediction of stock market returns: the case of the Japanese stock market,. *Chaos Solutions & Fractals*, (85), 1-7.

Rumelhart D., Hinton G. E., & Williams R. J. (1986). Learning representations by back-propagating error. *Nature*, (323), 553-536.

Sadia A., Khan F., & Bashir F. (2018). An overview of lexicon based approach for sentiment analysis. *International Electrical Engineering Conference* (IEEC 2018), 154-164.

Sezer O. B., Gudelek M. U., & Ozbayoglu A. M. (2020). Financial time series forecasting with deep learning: a systematic literature review: 2005-2019. *Applied Soft Computing*, (90) 106181.

Site A., Birant D., & Işık Z. (2019). Stock market forecasting using machine learning models. *2019 Innovations in Intelligent Systems and Applications Conference* (ASYU), 1-5.

Site A., & Işık Z. (2020). The analysis and forecasting of stock price with deep learning. *The 28th IEEE Conference on Signal Processing and Communication Applications* (SIU), 1-4.

Snoek J., Larochelle H., & Adams R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *25th International Conference on Neural Information Processing Systems*, (2), 2951-2959.

Thawornwong, S., & Enke, D. (2004). The adaptive selection of financial and economic variables for use with artificial neural networks. *Neurocomputing*, (56), 205–232.

Tikhonov A. N., & Arsenin V. Y. (1977). *Solutions of III-posed problems*. Washington, DC: VH Winston and Sons.

Tong H. (1983). Threshold models. *Threshold Models in Non-linear Time Series Analysis*, (21), 59-121.

Vargas, M. R., de Lima B. S. L. P., & Evsukoff A. G. (2017). Deep learning for stock market prediction from financial news articles. *International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications* (CIVEMSA), 2377-9322.

Yahoo Finance.(2020). Retrieved June 6, 2020 from https://www.finance.yahoo.com/

Zhang G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, (50), 159-175.

APPENDICES

APPENDIX 1: LIST OF ACRONYMS

AE	Autoencoder
ARIMA	Autoregressive Moving Average
ANN	Artificial Neural Network
API	Application Programming Interface
ATR	Average True Range
BIST	Borsa Istanbul
BTT	Backpropagation Through Time
CNN	Convolutional Neural Network
cuDNN	CUDA Deep Neural Network Library
DJIA	Dow Jones Industrial Average
EMH	Efficient Market Hypothesis
EWMA	Exponentially Weighted Moving Average
FCL	Fully Connected Layer
FNN	Feed Forward Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
KKT	Karush-Kuhn-Tucker
LSTM	Long Short-Term Memory
LS-SVM	Least Squared Support Vector Machine
LR	Linear Regression
MACD	Moving Average Convergence Divergence
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentege Error
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
NASDAQ	National Association of Securities Dealers Automated Quotations

NB	Naive Bayes
NLP	Natural Language Processing
NSE	National Stock Exchange
NYSE	The New York Stock Exchange
OHLC	Open High Low Close
PSO	Particle Swarm Optimization
RF	Random Forest
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RMSProp	Root Mean Squared Propagation
RNN	Recurrent Neural Network
RR	Ridge Regression
RSI	Relative Strength Index
SAE	Stacked Autoencoder
SGD	Stochastic Gradient Descent
SRN	Simple Recurrent Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
S&P500	Standard & Poor 500
TPE	Tree Structured Parzen Estimator
VADER	Valence Aware Dictionary and Sentiment Reasoner