UNIVERSITY OF MACEDONIA
DEPARTMENT OF APPLIED INFORMATICS
MSC IN ARTIFICIAL INTELLIGENCE AND DATA ANALYTICS

MACHINE LEARNING TECHNIQUES FOR STOCK MARKET PREDICTION

A dissertation
by

Pavlidis Pavlos

July 3, 2023

MACHINE LEARNING TECHNIQUES FOR STOCK MARKET PREDICTION

Pavlidis Pavlos

Bachelor in Applied Informatics, University of Macedonia, 2019

Dissertation

Submitted in partial fulfilment of the requirements for
THE MSC IN ARTIFICIAL INTELLIGENCE AND DATA ANALYTICS

Supervisor Professor
Samaras Nikolaos

Approved by the three-member Examination Committee on ../../2023

N. Samaras	I. Refanidis	D. Hristu-Varsakelis
.....	.....	.....

Pavlidis Pavlos

.....

UNIVERSITY OF MACEDONIA

Abstract

MSc in Artificial Intelligence and Data Analytics
Department of Applied Informatics

Master's Thesis

Machine learning techniques for stock market prediction

Pavlidis Pavlos

This paper attempts to provide a comprehensive study on stock price prediction using LSTM neural networks and compare their performance. Using 10 years of data from the US stock market index S&P 500, several simple LSTM and LSTM with Attention models were trained. A novel rolling window approach was utilized for the training procedure, where each model was trained on subsequent, non overlapping subsets so that the weights of the model are updated regularly to capture the ongoing trends. The experimental results revealed that models with smaller architecture outperformed larger models and that dropout, loss function, and model type all have a little impact on performance.

Keywords: Deep learning, Stock prediction, Neural network, LSTM

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Samaras Nikolaos, Professor of the University of Macedonia, for his guidance, support and encouragement throughout my research journey. I am deeply grateful for all the time and energy he devoted to help me develop as a researcher. I would also like to thank my colleague, Apostolos Kouzoukos, for his significant contribution to my research project. His expertise, insights, and enthusiasm have been an invaluable source of support, and I am grateful for the opportunity to work with him.

Finally, I would like to thank my family and friends for their love, support and encouragement throughout this journey.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
RNN	**R**eccurent **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
SVM	**S**upport **V**ector **M**achine
LSTM	**L**ong **S**hort-**T**erm **M**emory
RL	**R**einforcement **L**earning
ANN	**A**rtificial **N**eural **N**etwork
DT	**D**ecision **T**rees
PCA	**P**rincipal **C**omponent **A**nalysis
DNN	**D**eep **N**eural **N**etworks
AE	**A**uto **E**ncoder
RBM	**R**estricted **B**oltzmann **M**achine
MSE	**M**ean **S**quared **E**rror
MAE	**M**ean **A**verage **E**rror
RBFNN	**R**adial **B**asis **F**unction **N**eural **N**etwork
RW	**R**andom **W**alk
WT	**W**avelet **T**ransformations
RR	**R**andom **F**orest
LOG	**L**ogistic **R**egression **C**lassifier
SFM	**S**tate **F**requency **M**emory
AR	**A**uto**R**egressive
ASE	**A**verage **S**quare **E**rror
RMSE	**R**oot **M**ean **S**quare **E**rror

Chapter 1

Introduction

The stock market and stock forecasting selection are among the most significant and interesting issues in the financial field. The stock market allows corporations to raise their income by selling stocks, while investors may benefit by selling assets for more than what they initially paid for and receiving dividends (the distribution of a proportion of the company's profits back to investors).

The price of a stock fluctuates and is influenced by market supply and demand, which is driven by investors willing to purchase or sell stocks. As a result, a solid approximation of a share's actual value might be an investment opportunity. Meanwhile, reliable stock price or direction forecast is challenging for both investors and academics. The complexities of the stock prices revolve around certain factors that involve quarterly earnings' reports, market news, and various changing behaviors. The traders rely on several technical indicators, which are collected on a regular basis, but forecasting daily or weekly market patterns is too difficult. The accurate prediction of stock trends is interesting and a complex task in the ever-changing industrial world. Several factors affect the behavior of stock trends, including:

- Economic variables (interest rates, exchange rates, inflation/deflation, monetary growth rates, commodity prices, general economic conditions and economic outlook)
- Political variables (changes in governments, political scandals)
- Company specific variables (changes in company policies)

- industry specific variables (growth rates of industrial production and consumer prices)
- Psychological variables of investors (investors' expectations)

Forecasting stock trends is a complex task since it is impacted by various factors, including trader expectations, financial situations, administrative events, and specific market patterns. Furthermore, the stock market is a system that is dynamic, nonlinear, nonstationary, nonparametric, noisy, and chaotic. Forecasting financial time series becomes difficult owing to complicated aspects such as volatility, noise, and shifting patterns [5].

Over the years, various forecasting techniques have been used, such as ARIMA and Regression models. However, in recent years artificial intelligence has been developing rapidly and has a practical impact on every industry. It has evolved into a huge superpower that has altered the way we interact and, in the future, may alter the way we live our lives. The two leading factors for the rapid growth of AI in this decade are:

1. the amount of data generated is growing exponentially
2. the computational power was increased and many companies have started creating hardware specifical for training Deep Learning models

Stock market data are time series and consist of various variables so it can lead to big noisy data sets that would be impossible for a human to analyze and make a prediction. This gave the researchers an opportunity to test different machine learning techniques for stock prices prediction such as Feedforward nets, RNN, CNN and SVM.

The performance of AI models is highly correlated to the data used. For example, if the input we give to a deep learning model is noisy or has highly correlated features, the model will not be able to achieve high accuracy, thus suggesting that the data preprocessing phase is as important as the model architecture. The goals of this thesis is to provide a comprehensive study on stock prices prediction using LSTM neural networks and compare their performance. We aim to create a framework that will be able to extract the useful information from the given data set and successfully predict stock prices returns.

This thesis is structured as follows: The following chapter 2 will present the fundamentals and theoretical background needed to understand this work. Chapter 3 presents the related literature and the methods on which the proposed models and methods are based in. Chapter 4 describes the feature engineering and feature scaling that applied to the data set. The training and testing methodology and the method for the model optimization are explained in Chapter 4 too. Chapter 5 demonstrates the results of the experimentation on both LSTM models. The conclusions of this work and a discussion on the overall work and future research directions and/or improvements are discussed in Chapter 6.

Chapter 2

Theoretical background

2.1 Stocks

A stock is a type of certificate that represents ownership of part of the assets of any company [22]. Stocks are an important part of the global economy because they allow corporations to obtain funds for company operations by selling shares (or pieces of ownership) to the public. Stocks can be purchased or sold on an exchange, or in some situations, privately [35] and they can be categorized into three types:

- **Common:** Shareholders who own common stock have theoretically unlimited profit potential and they are entitled to a proportionate part of the value of the company's remaining assets if the company is dissolved. Common stocks, unlike preferred stocks, allow investors to attend yearly shareholder meetings and vote on business topics such as board member elections and general corporate strategy [11].
- **Preferred:** Preferred stock is typically chosen by investors who do not need to vote on corporate matters and are interested in getting a steady dividend check. Owners of preferred stocks are legally entitled to receive dividend payments before any dividends can be issued to other shareholders.
- **Convertible preferred stocks:** A convertible preferred stock is basically a preferred stock with an option of converting into a fixed number of common shares, usually any time after a predetermined date [15].

2.2 Types of stock analysis

Stock markets have been researched for years in order to extract useful patterns and predict their movements. Despite the extensive scientific attempts, no definitive strategy for correctly forecasting stock price movement has been developed. Stock market analysis assists investors in determining if a stock is appropriately priced in the market in order to make better purchasing and selling decisions. The evaluation and examination of an individual stock or an investment sector is part of the analysis. There are three basic types of stock analysis that are mostly applied today: fundamental, technical, and quantitative.

Fundamental analysis is the study of a company's stock prices in relation to the macroeconomic and microeconomic factors affecting the organization such as its financial records, revenue sources, profitability, expenses, company assets, market share, company's management etc. Essentially it utilizes anything connected to the economic well-being of a company to determine the 'fair value' of a company and then examines whether the stock prices are undervalued or not. If an analyst estimates that the stock's value should be higher than the stock's current market price, the investors may consider to invest. On the contrary, if the analyst calculates a lower intrinsic value than the current market price, the stock is considered overvalued and investor should consider to sell.

Technical analysis is the study of past and present price movements in order to predict the likelihood of future price movements. It is based on the assumption that a security's past trading activity and price changes can be valuable indicators of the security's future price movements when combined with appropriate investing or trading rules. However, technical stock analysis is only useful when supply and demand influence the price trend being studied and no other outside factors are present.

The candlestick chart (Fig. 2.1) is the most commonly used chart pattern in technical analysis because it displays the four major prices for the day: opening, closing, highest, and lowest. The rectangle section of the candlestick is known as the "real body," and it represents the link between the day's opening and closing prices. When the real body is filled with red, the close price is lower than the open price (bearish candle), otherwise, if the real body is green, the close price was higher than the open price (bullish candle).

The thin vertical lines outside the real body are known as the wicks or shadows, and they represent the day's highest and lowest prices.

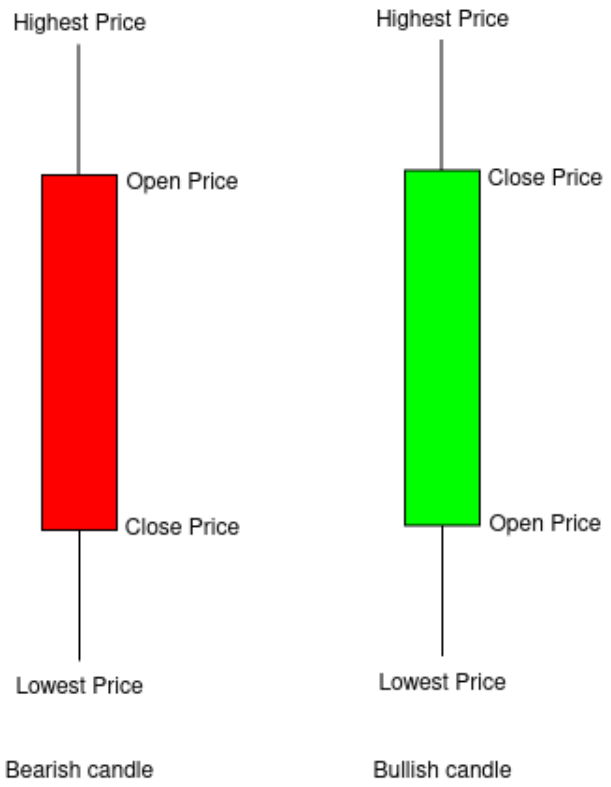

FIGURE 2.1: Candlestick chart: Bearish and Bullish candles

Quantitative analysis develops models for price forecasting using a variety of variables, including historical investments and stock market data. It uses both the same indicators as technical analysis does but it also uses any kind of state of the art statistical tools (eg. machine learning).

2.3 Machine Learning

Human's main goal from ancient times until today is to create the conditions that will ensure the highest quality of life and the fullest and most effective satisfaction of his needs. Machines, from the simple tool to the most complex modern devices, are part of human history. For many years researchers have been trying to make computers to learn since the impact this may have on humans is enormous. From robots to rocket science, machine learning can be used everywhere. The last decade, computers have gotten a lot more capable and are faster than ever before, allowing us to apply machine

learning to more complex tasks. However, it is still not as good as people in learning because humans' ability to build complex patterns from given data is difficult to replicate synthetically.

Arthur Samuel defined machine learning in 1959 as "a field of study that gives computers the ability to learn without being explicitly programmed" [32]. In other words, machine learning is a method used to devise complex models and algorithms that can be used in prediction. Such algorithms work by building models based on features extracted from experimental data, without being explicitly programmed, in order to make predictions based on the data or to derive decisions expressed as the result.

A feature is a measurable property of a data-set entity. In a data set there is a large percentage of observations from which a machine learning algorithm is trained. The goal is to select an appropriate set of features for the algorithm to utilize so that it learns to recognize what describes each observation in the data set.

Based on the nature of the task that needs to be solved and the type and the amount of the data that are available, machine learning is split into four categories: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

2.3.1 Supervised learning

Supervised learning algorithms are given labeled input-output pairs and the model is trained to predict the output (the label) of new, unseen instances [39]. For example for financial applications, input parameters can be stock related variables (e.g close and open price, highest and lowest price and volume) and the corresponding output parameter can be the close price values of stock for the next day or the prediction on whether the close price will go up or down. The goal is to find a function that minimizes the prediction error that is expressed as the difference between the real and the computed value.

The supervised learning process begins with training, in which the model is fed input-output pairs in vector form. After the algorithm has been trained and is successful in extracting data patterns, it is used to forecast previously unknown examples in order to evaluate its performance on them. This procedure is continued until the test error has been decreased to an acceptable level 2.2.

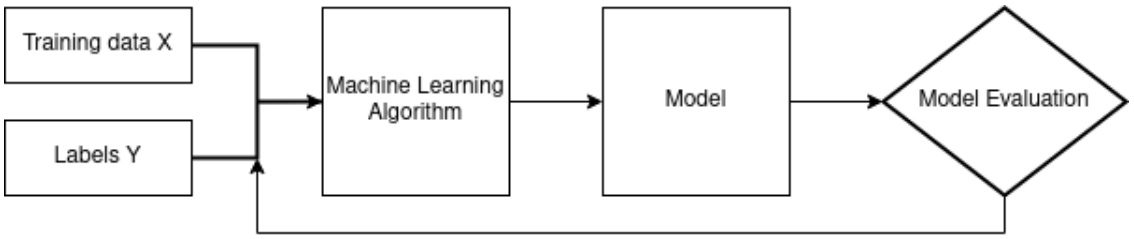

FIGURE 2.2: The Supervised learning process

Each model is built using a different algorithm. There are many factors that affect the algorithm selection but most significant of them are the data set's size, the type of the data and their special characteristics, resources usage and speed of training [23]. If the target class has discrete or continuous values, supervised learning is classified as classification or regression respectively.

Classification algorithms are used when the outputs are discrete, for example if stock's close price will go up or down the next day, and regression algorithms are used when the outputs are continuous, for example predicting the stock close price value. Figures 2.3 and 2.4 represent an example for classification and regression respectively. In figure 2.3 the blue line defines the boundary that separates the two classes which shows that the point (8,5), belongs to the red class. On the contrary, in figure 2.4 the blue line shows the prediction itself meaning that for the value 8, the predicted value is a point in the blue line - more specifically it is -8.

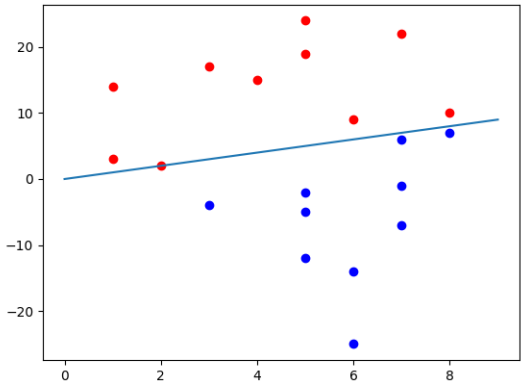

FIGURE 2.3: Classification example

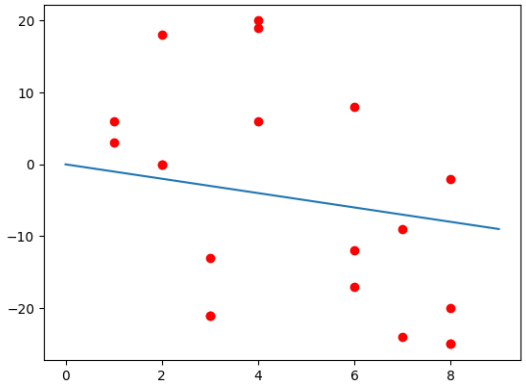

FIGURE 2.4: Regression example

Depending on the type of the task, the type error is calculated differently. In classification the metric accuracy 2.1 is most common and it is defined as the number of wrong

predicted targets of the test set divided by the number of all the instances of the test set [9]:

$$accuracy = \frac{FP + FN}{TP + TN + FP + FN} \tag{2.1}$$

- FP-False Positive: the number of instances that wrongly predicted that belong to the class
- FN-False Negative: the number of instances that wrongly predicted that do not belong to the class
- TP-True Positive: the number of instances that correctly predicted that belong to the class
- TN-True Negative: the number of instances that correctly predicted that do not belong to the class

For regression tasks, the most commonly used error metrics are the Mean Squared Error (MSE) 2.2 and Mean Average Square (MAS) 2.3.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y_i}\right)^2 \tag{2.2}$$

and

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid Y_i - \hat{Y_i} \mid \tag{2.3}$$

where:

- Y_i is the true value of the i^{th} instance
- $\hat{Y_i}$ is the predicted value of the i^{th} instance
- n is the size of the test set

2.3.2 Unsupervised learning

Unsupervised learning methods use unlabeled data to find hidden patterns or intrinsic structures in input data. A good example of unsupervised learning usage is the task in

which the training set contains uncharacterized texts from a newspaper and the aim of the training is to separate them into a given number of groups according to their content. In other words, at the end of the training, the algorithm output should classify the texts with sports or political content to the corresponding group. Clustering and association are two popular families of methods for unsupervised learning problems:

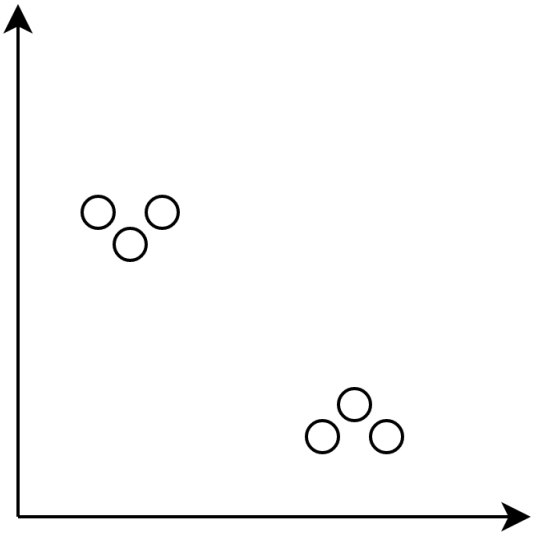

FIGURE 2.5: Data points before clustering

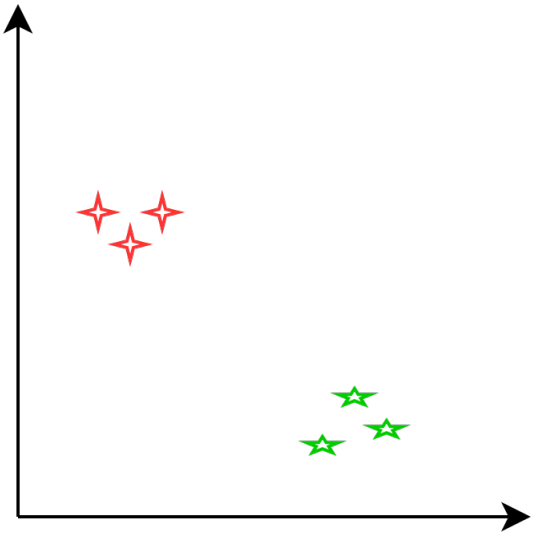

FIGURE 2.6: Data points after clustering

- **Clustering**: Clustering is one of the most common methods for unsupervised learning and it is used for tasks that need to organize objects into groups whose members are similar in some way. For example in Figure 2.5 there are 6 data points and in Figure 2.6 these data points are clustered in two clusters, red and green. There are two types of clustering: hard and soft. Each data point in hard clustering belongs to only one cluster (for example, whether a tweet is positive or negative), whereas each data point in soft clustering might belong to many clusters. In the latter case, each data point is assigned a probability for each class, indicating how probable it is that this specific point belongs to that class.

- **Association**: Association method focuses on finding relationships between variables in a given data set [18]. A typical example of association rules is shopping basket analysis which allows companies to find relationships between different products and understand consumers' habits. If for example a supermarket knows that

when a customer buys bread, he tends to buy butter too, it can use this information to improve the performance of their strategies (e.g where the advertisements or the products should be placed in the shop).

2.3.3 Semi-supervised learning

The basic disadvantage of supervised learning is that it requires the data to be labeled. In other words, it needs a person to label the data by hand which is a particularly expensive process and time-consuming. On the other hand supervised learning techniques tend to be more accurate than unsupervised. Someone may say that semi-supervised learning lies between supervised and unsupervised learning since it is used when a large number of data records are unlabeled (usually in greater proportions than labeled records). It tries to predict a label for the unlabeled data (a process known as pseudo-labeling) in order to use them to increase the prediction's accuracy. More specifically the following process is usually used:

1. Using supervising learning it trains the model with a small set of labeled data.
2. In the next step, the model is used to label the unlabeled data and set the prediction as the pseudo-label.
3. Using the new data set (the one that contains the pseudo-labels for the unlabeled data) the model is getting trained again as it did in the first step.

2.3.4 Reinforcement learning

Reinforcement Learning (RL) is a feedback-based machine learning technique in which an agent learns to behave in an environment by performing actions and receiving rewards depending on their outcomes, with the goal of maximizing the sum of those rewards [31]. It is used when the model needs to learn to do something such as play a game.

The word agent refers to the component of the algorithm that makes the decisions and the rest are defined as part of the environment. The agent receives as input the current state of the environment and chooses an action. The action alters the state of the environment, and the agent receives a reward as a result from this action.

For example, if the goal is to teach a model to play a game (e.g minesweeper), supervised learning requires a person to play the game numerous times and record each action he does in order to feed them into the model. This will result in a model that can play the game, but it does it just as the human did, making all of the same mistakes. In RL, the agent only takes as an input the current state of the environment (i.e. the current grid of squares in minesweeper) and chooses on its own through trial and error what is the best strategy to follow utilizing the rewards that it receives at each step as guidance 2.7. The process ends when the agent has found the actions that it needs to take in order to achieve the largest total sum of rewards.

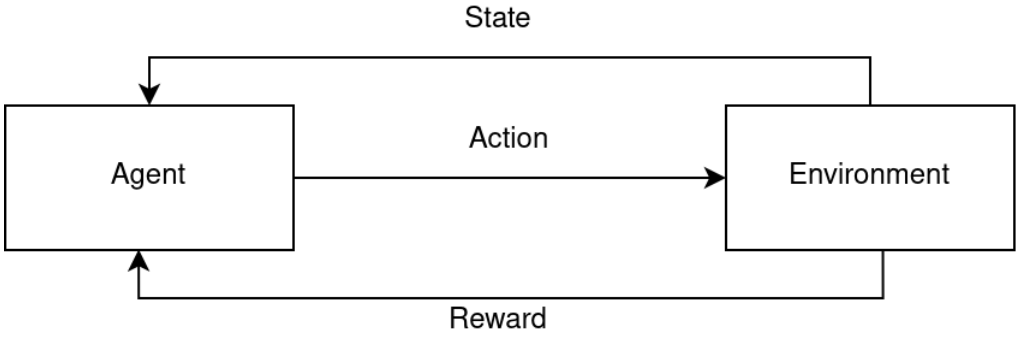

FIGURE 2.7: The Reinforcement learning process

Chapter 3

Literature Review

Stock market indices prediction is one of the most important and extremely challenging financial time series forecasting problems for both investors and researchers. This is mainly caused by the fact that the stock market is essentially an unstable, nonlinear and complex system of dynamic change, and is affected by many factors, such as major economic policies, government decrees, the change of political situation, investor's psychology, the future economy, and so on. The benefits involved in accurate prediction have motivated researchers to develop newer and more advanced tools and methods some of which are represented in this chapter.

3.1 Feedforward nets (Artificial Neural Networks - ANN)

The feed forward neural network was the first and the plainest type of artificial neural network to be developed. In this type of network the information pass from the input layer to the output only in one direction and without forming a cycle 3.1. Thus, the information is forwarded to the hidden layer, where the complex calculations are performed, and then the prediction is made without getting the results backwards to the input layer. The simplest type of feed forward networks are the single layer perceptron, which consists of a single layer of output nodes - the inputs are fed directly to the outputs via a series of weights. However, more complex networks exist, the multilayer percepron technique, in which each neuron in one layer has directed connections to the neurons

of the subsequent layer. Many of these networks' units use a sigmoid function as an activation function.

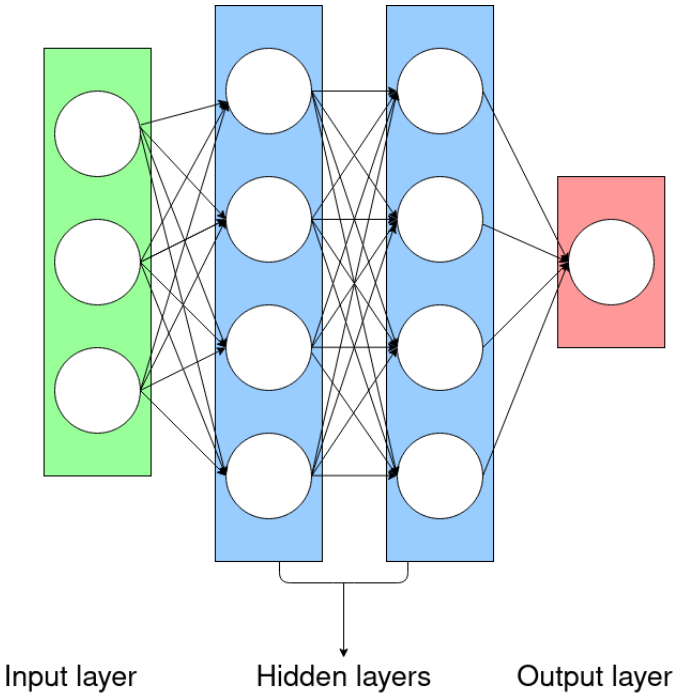

FIGURE 3.1: Typical Feedforward architecture

In the stock prediction analysis there are many applications that uses the feed-forward neural networks, in order to predict the slopes and fluctuations of economical indexes. Tsai and Wang [36] inspired from previous studies that combined multiple techniques, suggested a model which combines ANN and Decision Trees. Feed-forward models have high prediction accuracy in stock price forecasting whereas DTs have excellent ability to describe cause and effect relation of information. Therefore, this paper combines the commonly used ANN techniques and the great explanation ability of decision tree for a better decision support system in order to help investors to make more correct decision in stock investment. In their experiments they used fundamental, technical and macroeconomic indexes as the indicators for ANN and decision trees to forecast the stock price in electron industry in Taiwan. In their methodology, they first used the PCA method to filter out unrepresentative variables, and then trained four different models: ANN, DT, ANN-DT, and DT-DT for different number of epochs and neurons in the hidden layer, using average prediction accuracy and Type I/II errors for evaluation. The data that they used were retrieved from the TEJ database for the period 02Q1-07Q2. The results showed that the ANN-DT variant presents a higher average prediction accuracy with Error Type II equal to 1.95% which makes it very useful for predicting stock price

increases. On the contrary, none of the models appeared to be capable of predicting the stock market's downward trend. However, the proposed models are considered out of date, and no comparisons in various stock indices are made.

The ability of DNNs to extract features from a large set of raw data without relying on prior knowledge of predictors is one of their main advantages, but their performance is highly dependent on input variables and network parameters such as the number of hidden layers and the number of nodes in each layer. Deep learning algorithms extract features from data and require minimal human effort during feature selection. Therefore, Chong, Han and Park [14] investigated the effects of three unsupervised feature extraction methods on the network's overall ability to forecast future market behavior using high-frequency intraday stock returns as input data: Principal Component Analysis (PCA), Auto Encoder (AE), and Restricted Boltzmann Machine (RBM). As input they used 380 features concerning the prices of 38 stocks for the period 04/01/2010 to 30/12/2014 from the Korean stock exchange KOPSI and their experiments showed that the combination of deep learning and autoregressive models achieved good performance. They fund that the DNNs perform better than a linear autoregressive model in the training set but the advantage disapperars in the test set. Applying the DNN to the residuals of the autoregressive model improves the prediction accuracy, while the reverse is not true, demonstrating the advantages of DNNs over the autoregressive model.

Zhong and Enke [41] also studied data dimensionality reduction techniques combined with neural networks to successfully forecast the daily direction of the S&P 500 Index ETF (SPY) return based on 60 financial and economic features for the period 01/06/2003 to 31/05/2013. The three dimensional reduction methods used are: Principal Component Analysis (PCA), Fuzzy Robust PCA (FRPCA) and Kernel-based PCA (KPCA). From the three techniques, 36 new data sets are created with different levels of dimensionality reduction. Each of the 36 data sets (along with the original data set) are given as input to the feedforward neural network ANN with three layers. The results showed that the PCA method combined with the neural network yields the best results (mean of daily return = $8.40E-04$, std. of daily return = 0.0079 and sharpe ratio = 0.1011). Finally, the computational experiments showed the importance of dimensionality reduction in financial time series, as the predictions are much more accurate when PCA is combined with the neural network compared to the neural network using the original data or the methods FRPCA and KPCA. However, only one neural network and only three

dimensionality reduction methods were used, while data reduction techniques could be combined with other machine learning methods.

Singh and Srivastava [34] aimed to develop a price forecasting model for a stock that would outperform existing models. Based on prior work utilizing $(2D)^2PCA$ (2-Directional 2-Dimensional PCA) in RBFNN (Radial Basis Function Neural Network) [21], they hypothesized that using $(2D)^2PCA$, DNN will yield better results. More specifically, the data they used was collected from NASDAQ for the period 01/06/2011 to 10/12/2015 and it consists of the daily opening and closing price of the stock, the highest and lowest price of the stock and the quantity of the stock bought or sold. After they removed repeated entries (risking to lose useful information), they created more variables for forecasting using various formulas. Finally, (2D)PCA was applied to the data and the result was given as input back to (2D)PCA in order to generate $(2D)^2PCA$ whose results will be given as input to DNN. They used hyperbolic tangent as activation function, MSE (Mean Square Error) as loss function and 100 input nodes, 200 nodes in the hidden layer and 1 node in the output. ADADELTA was used to calculate weights and bias, and regularization was used to avoid overfitting. To determine the dimensionality reduction to be performed on the data and the number of days (window size) to use in order to predict the stock price for the following day, they first ran experiments with the DNN. Based on these values, they trained the RBFNN and RNN models and discovered that the RNN performed the worst of the three. RBFNN performed well, but DNN outperformed it by 4.8% in terms of hit rate and it had higher correlation coefficient between actual and predicted value.

As mentioned previously, the stock market is a system that is dynamic, nonlinear, nonstationary, nonparametric, noisy, and chaotic. A data set with financial time-series can have both linear and nonlinear relationships between its features. Random Walk (RW) models are well-suited to recognizing linear patterns, while Artificial Neural Networks (ANNs) are better at recognizing nonlinear correlations. Therefore, Adhikari and Agrawal [2] attempted to combine these methodologies to produce a hybrid model capable of forecasting financial data. They integrated the forecasts from FANN (Feedforward ANN) and EANN (Elma ANN), and then they combined the new model with the RW to get the final model. The empirical results from the experiments they conducted with four real-world financial time series 3.1 showed that the hybrid method outperformed each

Data set	Period	Total size	Testing size
USD–INR exchange rate	01/07/2009 - 11/09/2011	681	116
GBP–USD exchange rate	02/01/1980 - 31/12/1993	731	52
S&P 500	02/01/2004 - 31/12/2007	1006	106
IBM stock price	02/01/1965 - 31/12/2011	564	64

TABLE 3.1: Information of the four time series that Adhikari and Agrawal [2] used in their research

of the individual component models and significantly improved the overall forecasting accuracy.

3.2 Recurrent Neural Networks (RNN) - Long Short Term Memory (LSTM)

Recurrent neural networks (RNN) is a type of artificial neural networks where connections constitute an undirected or directed graph along a temporal sequence. This ability of RNN's facilitate the procedure of identifying patterns and temporal dynamic behavior in various fields like NLP, time-series forecasting, pattern/speech recognition, intermittent demand etc. This architecture saves the output of a state and feeds it back to the input layers which relates to future predictions. In the latter case, the model takes into account both current inputs and preceding elements. RNNs can be likened to a chain-like recurrent module in a conventional neural network, followed by a layer of memory cells. For prediction purposes, this type of network processes a series of events and manages the entire context, by maintaining the memory of previous states to learn long-term dependencies.

RNNs have the form of a chain of repeating modules of neural network. In standard RNNs, the repeating module have a very simple structure, such as a single tanh layer 3.2. Since the RNN is prone to exploding and vanishing gradients (gradients tend to infinity or to zero respectively) during the training process, the gradient cannot be passed down in long sequence and as a result, the model cannot keep track of long-term dependencies. In 1997, Hochreiter and Schmidhuber [25] introduced LSTM networks which are able to remember information for long periods and as a result they can solve the vanishing gradient problem. LSTMs also have this chain like structure but the repeating module has four layers which interact with each other 3.3.

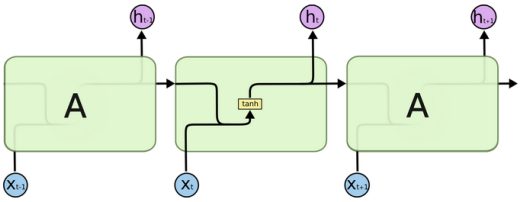

FIGURE 3.2: The repeating module in an RNN

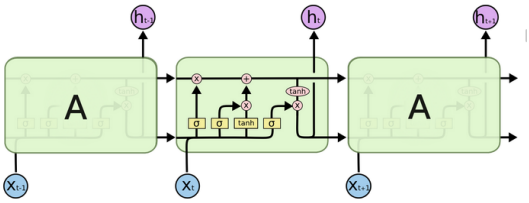

FIGURE 3.3: The repeating module in an LSTM [29]

One of the most important achievements in deep learning model preparation in the last few decades has been the attention mechanism. Bandanau et al. [7] presented it initially, and it has since been widely employed in NLP problems. The models in NLP are built on an encoder-decoder system, in which the encoder encodes the data into a context vector and generates a summary of the input data. The summary then moves on to the decoding part, where the model interprets and translates the data. If the summarization is poor, the decoder will translate and understand the data incorrectly. This is a major weakness of the basic model where the accuracy of the model is good but in the case that the input sentence is too long, the summarization does not work well and the model produces poor results. This is known as the **long-range dependency problem of RNN/LSTMs**.

Another problem is that there is no way to give more importance to some of the input words compared to others while translating the sentence. Suppose, for example, that the last word of a sentence ***"Although the fact that she studied pharmacy, she became a hairdresser because she always liked hairdressing"*** is asked to be predicted. The words ***became***, ***hairdresser*** and ***liked*** should be given higher weight.

Bahdanau et al. [7] proposed that not only should all of the input words be considered in the context vector, but that relative importance should also be assigned to each one of them. More precisely, when constructing the context vector, they focused on embeddings of all the words in the input (represented by hidden states). They accomplished this by

simply adding the weighted total of the hidden states. The weights are learned by a feed-forward neural network and using the following equations

[3]:

$$c_i = \sum_{j=1}^{T} a_{ij} h_j \tag{3.1}$$

where c_i is the context vector for the output word y_i. The weights a_{ij} computed by a softmax function given by the equations 3.2 and 3.3.

$$a_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{T} \exp e_{ik}} \tag{3.2}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.3}$$

where e_{ij} is the output score of a feedforward neural network that aims to capture the alignment between input at j and output at i as specified by the function a.

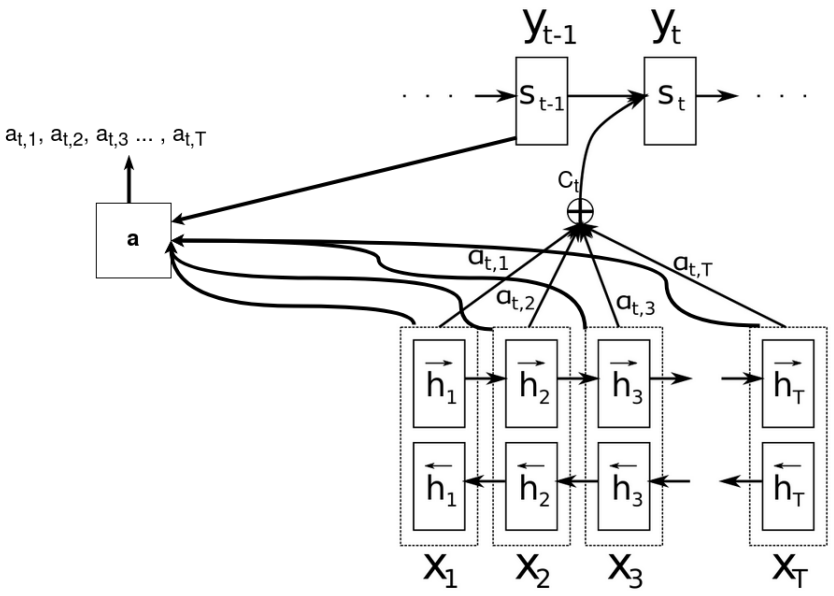

FIGURE 3.4: Attention Mechanism architecture

Figure 3.4 is based on the diagram of the Attention model presented from [7]. The Bidirectional LSTM they used generates a sequence of annotations $(h1, h2, ..., hTx)$ for each input sentence. All the vectors $h1, h2.., etc.$, used in their work are basically the concatenation of forward and backward hidden states in the encoder 3.4. The attention block receives the output of each encoder unit as well as the previous time step decoder output. The amount of attention to be paid to the hidden encoder unit h_j at each time

step t at the decoder is represented by a_j and computed as a function of both h_j and the previous state of the decoder s_{t-1}.

$$h_j = \left[\vec{h}_j^T; \overleftarrow{h}_j^T\right]^T \tag{3.4}$$

X. Pang et al [30], based on the processing of natural language and the representation of words in vector, attempted to develop a model for forecasting stock price movements that uses stock data represented in vector. However, when the data is for multiple shares, such a representation gives huge dimensions to the data and increases the hardware resources requirements. In addition, because the data may contain unnecessary information that may reduce the model's performance or delay training, they suggested two dimensionality reduction techniques and the use of Long Short-Term Memory (LSTM). More specifically, the first model is ELSTM (Long Short-Term Memory with Embedded Layer) where the embedded layer is located before the LSTM training and undertakes the filtering of unnecessary information and the representation of the data in vector before being given as input to the LSTM. The second model is AELSTM, which instead of the embedded layer, contains an automatic encoder with CRBM (Continuous Restricted Boltzmann machine). Both methods had a better predictive performance for the Shanghai A-share composite index (for the period 01/01/2006 - 19/10/2016). Finally, they observed that due to the volatility of stock prices, changes in the learning rate did not have much effect, confirming that the effect of the forecast when the stock is influenced by external factors may differ from the real one.

Bao, Yue and Rao [8], proposed an innovating novel deep learning framework for stock price forecasting that combines wavelet transformations (WT), stacked auto-encoders (SAEs), and LSTM. More specifically, at preprocessing stage, they applied wavelet transformation to eliminate the noise from the data. For the deep daily features extraction, they applied SAEs which has a deep architecture trained in an unsupervised manner. Finally, the output of the stacked auto-encoders was fed into the lstm as input for stock prediction. By comparing the WSAE-LSTM (Weighted SAE) method with the WLSTM (a combination of WT and LSTM), LSTM and RNN methods, they observed that the proposed method achieves better predictive ability and therefore better profit. The proposed method was applied to 6 years of data (2010-2016) while the results were interpreted using both predictive accuracy performance (MAPE, R2 and Theil U) and

profitability performance. The primary contribution of this work to the community is that it is the first attempt to use the SAEs method to extract deep invariant daily features from financial time series.

For forecasting Shanghai and Shenzhen stock indexes, Chen, Zhou, and Dai [12] used a variety of LSTM-based models. In terms of complexity, the model is straightforward, with only four levels. The sequences to be sorted are in the first input layer, the LSTM levels are in the second, the dense layer is in the third, and the simple output layer is in the fourth. The sequence is divided into seven categories based on the earning rate. The findings are based on six distinct models that varied in the number of attributes and whether or not they contribute to stock indexes. The six models are: the random prediction model, the M1 which is an LSTM model with closing price and trading volume as a learning feature, the M2 which is the M1 with normalization, the M3 which is the M2 with three additional features (high price, low price and opening price), the M4 which is the M2 with five additional features (high price, low price, opening and closing price and the price of Shanghai Securities Composite Index) and the M5 which has only SSE shares ETF180 Index used for training and validation. All models had poor accuracy (14,3% - 27,2%), however, they discovered from their findings that different stock sets affect forecast accuracy, and it is required to conduct predictions for different types of stocks independently. Furthermore, they observed that excluding sequences with exceptionally low or high earning rates improved model accuracy.

Because deep neural networks are larger and deeper, their performance is correlated with the size of the data set and if the data is insufficient it can lead to overfitting. ModAugNet is a method proposed by Baek and Kim [6] and it is based on LSTM. The authors presented a novel augmentation strategy for financial time-series forecasting to make the model resistant to overfitting without the requirement to produce artificial time-series to supplement available training data. ModAugNet-c was implemented in three steps. Firstly they built a modular lstm network that could accept the direct information (fed into the Prediction Module) and a lstm network that got as input the indirect information (fed into the Prevention Module). Secondly they picked 10 relevant features as input candidates, and finally they chose one of the combinations of 10 companies, choosing 5 at a time, every 200 epochs and feeding them into the Prevention Module for the duration of the succeeding 200 epochs, while the stock market index remained constant. The output of this two lstm networks is given as input to the final lstm

network. The aforesaid methodology was applied to two indices, the S&P500 and the KOSPI2000, and the results showed an improvement in accuracy when compared to using only the prediction network. Furthermore, by feeding noisy data to each module, they discovered that the prediction is solely made through the Prediction Module and that the Prevention Module is no longer used once the training is done. During training, the Prevention Module's input data behaved as injected noise, causing the network to focus on learning important patterns from the Prediction Module. Finally, the biggest point from this paper is that the suggested methodology may be used for a variety of deep learning-based financial problems (financial market movement classification, volatility prediction, portfolio optimization), as well as other domains such as medicine.

Fischer and Krauss [19] provided an in-depth guide on data preprocessing, as well as development, training, and deployment of LSTM networks for financial time series prediction tasks. More specifically, they employ the lstm to solve a classification problem in which the Class is equal to 0 or 1 depending on whether the return outperforms the cross-section median or not. Using data for the index S&P 500 for the period 12/1989 - 09/2015 from the Thomson Reuters database, they created and compare 3 memory-free classification methods: Random Forest (RR), Deep Neural Net (DNN), and Logistic Regression Classifier (LOG). From the results they found that LSTM outperformed the memory-free methods and led to the creation of more profitable portfolios. In addition the authors analyze the performance of the portfolio using a trading strategy in individual periods: 1993-2000, 2001-2009, 2010-2015. It is noteworthy that in the most recent period, when the global financial crisis started, the performance of the portfolio was negative. However, the method was tested on a single stock market index. Also, the use of a single variable, the standarized returns, on the one hand speeds up the training of the models, however on the other, it removes useful information that the model could use to improve its performance.

Due to the trading activity carried out at various places and cycles, the stock price shows multi-frequency patterns. As a result, identifying meaningful frequency patterns may provide useful hints about the future price movement. Based on this theory, Zhang, Aggarwal and Qi [40] tried to predict stock prices using a State Frequency Memory model (SFM) [26]. The SFM topology shares many similarities with the LSTM architecture, but it also has a crucial distinction, the input data is deconstructed and stored in memory. In other words, the data is discretely separated into frequencies, similar to the Fourier

transform. This topology makes it possible to separate the high frequencies from the low ones and therefore to make both short-term and long-term forecasts. The data set they used was retrieved from Yahoo Finance and it contained the daily opening prices of 50 stocks from 10 different sectors for the years 2007-2016. In their experiments, the authors compared SFM to the AR (Autoregressive) and LSTM models and discovered that the proposed topology had a lower ASE (Average Square Error) than the other two topologies and it can lead to more accurate prediction.

Chen and Ge [13] compared the performance of a LSTM model and a LSTM model with an Attention layer on Hong Kong stock market data trying to predict the direction of daily close price movement. The two models have 2 LSTM layers with the difference that one has an attention layer inserted before the LSTM layers. The dropout is used in the LSTM layer, and there is a dense layer at the end of the two LSTM layers to obtain the two-valued outputs, which are the prediction classes (increasing movement and decreasing movement). Finally, following the dense layer, a BatchNormalization layer is applied, and the output of the BatchNormalization layer is passed into a Softmax classification layer. From the experiments they condacted, they showed that attention layer increases the accuracy of the LSTM model which was also verified from the experiments they did with the "Long-only" trading strategy.

3.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) is a type of artificial neural network, most commonly used in computer vision or image classification. The hidden layers of a convolutional neural network apply convolutions to the raw data. The convolution layer is typically a layer that performs the dot product on the input matrix (Frobenius inner product), with ReLU as its activation function. The convolution operation generates a feature map as the convolution kernel slides along the input matrix for the layer, which then contributes to the input of the next layer 3.5. Other layers such as pooling layers, fully connected layers, and normalization layers follow. The term convolution is derived from the central concept of CNNs, which is the concept of sliding or convolving a predetermined window of data. Nonetheless, it has proven to be one of the most popular methods for capturing the distinctiveness of a particular financial index in low-frequency

data. CNNs are able to classify the slopes of time series from noise data, therefore they could be a good solution for the portfolio optimization problem [37].

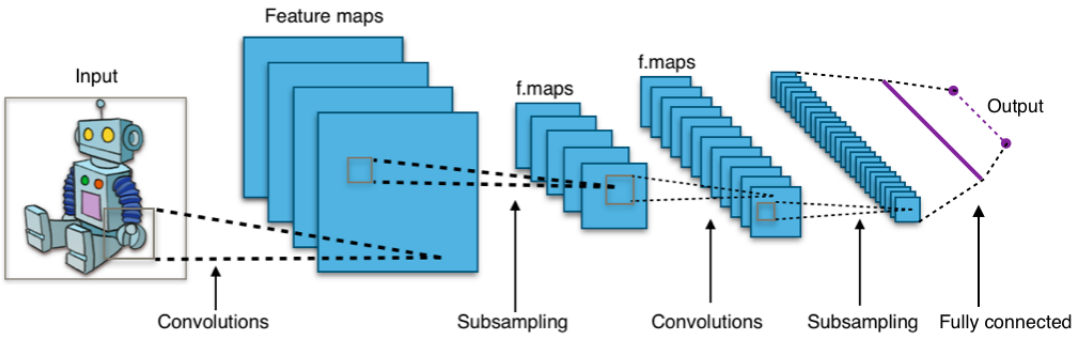

FIGURE 3.5: Typical CNN architecture [4]

Over the years many researchers attempted to solve the problem of stock prediction using CNNs. For example, Gunduz, Yaslan and Cataltepe [20] proposed the use of a CNN network to predict the intraday direction of 100 stocks in the Turkish stock market. Logistic regression and CNN were used to predict the direction of closing prices of stocks. In logistic regression, the relationship between labels and technical indicators is estimated where if the probability obtained by the model was more than 0.5, then it is assumed that the price is predicted to increase, otherwise, it is predicted to decrease. In order to reduce the number of features to be used as input, the Chi-Square method was chosen. The suggested CNN classifier contains seven layers: one input layer, two parallel convolution layers, one merge layer, one convolution layer, and one fully-connected layer. Adadelta was used for the optimization, ReLU was used as the activation function with 200 training epochs, the batch size was set to 32 and the dropout at 0.3. This is the first study to use CNN in stock prices to estimate their direction, as well as the first to use data from the Turkish stock market. In the experiments they conducted, they gave the indices in CNN first in random order and then in a selected order and the results showed that CNN performed best with the selected order of features, with a Macro-Averaged F-Measure of 0.563. However, due to a lack of data (from January 2011 to December 2015), the prediction was made for one year.

Long, Lu and Cui [28] proposed a model that by applying filters through convolutional and recurrent neurons they extract variables from time-series data. Through this technique (named multi-filters neural network - MFNN), they aimed to build a model with which they will be able to predict the trend of the stock price. The difference

between this research from other similar ones lies in the fact that the extraction of further information from the data set takes place within the model and not during the pre-processing of the data (e.g. calculation of technical indicators). In addition, the combination of convolutional and recurrent neurons allows both the reduction of data dimensions and the inclusion of information and relationships that may exist in historical data. In the process of pre-processing the data, they first scaled the data and generate 30 data sets, aiming to explore the relationship between the predictability of samples, ϑ and $t_{forward}$ (ϑ is the percentage by which they gave class value to the data and $t_{forward}$ is the time period we want to make a forecast (e.g. the stock price after 5 minutes). In the training model, the width of the convolutional filters at each level is equal to six, while the number of recurrent neurons is equal to the number of convolutional neurons. Essentially, the output of the neurons in the first level is aggregated and then given as input to the next level of filtering. The new result is stored in a vector, which is then fed into a fully connected layer to apply the softmax function and create the prediction. The data they used for both training and testing was from CSI 300 market for the period 02/12/2013 - 07/12/2016. The results showed that the accuracy of the model increases as ϑ decreases while changes to $t_{forward}$ do not appear to have an effect. Furthermore, the evaluation of training results from RNN, CNN, LSTM, Logistic Regression, Random Forest, and Linear Regression models revealed that the MFNN model performed better. In addition, to determine how beneficial it would be for an investor to use the MFNN and compare the performance with other models, they created an investing strategy in which the models, without taking into account the transaction fees, purchased and sold shares for around 8 months. Initially, all models outperformed randomly selecting and purchasing stocks, demonstrating that they can all be useful. Except for the RNN and the metrics Sharpe ratio and Daily winning rate, the MFNN outperformed all other models. However, eliminating transaction fees in order to simplify the investment approach can lead to incorrect results and the difference between the performance of MFNN and RNN is not significant. A change in their investing strategy, as well as the addition of transaction fees, could result in different outcomes and conclusions.

M et al. [24] attempted to predict the values of five stocks from the NSE (3 stocks - Maruti, Axis bank, and Hcltech for the period 01/01/1996 - 30/06/2015) and NYSE (2 stocks - Bank of America and Chesapeak Energy for the period 03/01/2011 - 30/12/2016) indexes. Firstly they normalized their data and then, through tests, they set the rolling

window equal to 10 since the MAPE was lowest for this size. At next step, they fed their data into the LSTM, MLP, RNN, CNN, and ARIMA models to see which one achieved the best prediction. Based on the results, it is obvious that the algorithms are capable of spotting trends in both stock markets. This demonstrates that there is an underlying dynamic that is shared by both stock markets. Linear models, such as ARIMA, forecast univariate time series and so are incapable of discovering underlying dynamics within multiple time series. CNN outperformed the other three networks in the suggested study because it is capable of catching abrupt changes in the system since a specific window is used to forecast the following moment.

Selvin et al. [33] used three deep learning topologies (CNN, RNN and LSTM) to predict three stocks from the NSE index. Specifically, they applied the rolling window method for short-term forecasting which has a size of 100 minutes with an overlap of 90 minutes' information for 10 minutes prediction in future. The ideal window size was determined after experimenting to minimize error. The three methods were compared to a classic linear time series forecasting method, ARIMA, and from the experiments they concluded that CNN shows better predictability compared to other models (minimum RMSE equal to 2.26 versus 3.83). This is because the unpredictability of stock market indexes causes issues for recursive topologies like RNN and LSTM. CNN, on the other hand, does not rely on prior knowledge to make predictions. It predicts only the present window, allowing the model to understand the dynamic changes and patterns that occur in this window.

Ding et al. [17] attempted to predict the trend of the S&P500 index and 15 stock prices by converting financial news from Reuters and Bloomberg (over the period 10/2006 - 11/2013) into event embeddings. In contrast to prior research that used simple features from news documents, such as bags-of-words, noun phrases, and named entities, which do not capture structured relations in events, Open IE is suggested here to extract information. To deal with the increased sparsity of structured representations, they developed dense vector event embeddings. Each event is represented as a tuple $E = (O1, P, O2, T)$ (Ding et al. [16]), where P represents the action, O1 represents the person doing the action, O2 represents the item on which the action is performed, and T represents the timestamp. In an economic text sentence, ReVerb is used first to extract possible event tuples, and then the sentence is parsed with ZPar to extract the subject, object, and predicate. Because the events are very sparse, a Neural Tensor Network was

used, which takes word embeddings as input and outputs event embeddings. The events of the previous month are shown as long-term for the model that makes the forecasts, the previous day as short-term, and the prior week as medium-term, and the model learns how much the news affects prices depending on the different time periods. It accepts event embeddings in chronological order as input and the output is a binary class that indicates whether the stock price is expected to go up or down. The remaining architecture consists of a CNN network with a hidden convolutional layer followed by max pooling. MCC (Matthews Correlation Coefficient) and accuracy are the measures employed. For the experiments, 5 different networks were tested: a standard neural network with word embeddings, a CNN network with word embeddings, a CNN with structured events, a standard neural network with event embeddings and finally the proposed CNN network with event embeddings. The experimental results reveal that the suggested network obtains an accuracy of 4.21% and an MCC of 0.40 on the S&P500 index, while it achieves an accuracy of 65.48% and an MCC of 0.41 on average for individual stocks. They also evaluated their model using Lavrenko's technique [27], in which the model invests $10,000 in the opening price if it predicts a stock will rise in price the next day. After the purchase, it holds the stock for 1 day. If during that time it thinks it can make a profit of 2% or more then it sells immediately, otherwise it sells the stock at the closing price. With this technique they achieved a profit of $16,774 on average.

Chapter 4

Data preprocessing and training methodology

The problem of predicting the price of a stock, at some future time horizon, is a time series input matching problem, since at its input the prediction model accepts a time series of data, at a price, which is the price of the stock at some future time. However, it can turn into a time-series-to-time-series matching problem if more than one stock price, including non-future prices (i.e., already known), are requested in the output.

One of the most critical requirements for a machine learning algorithm to perform successfully is data. Choosing the appropriate data is critical, particularly in the world of financial markets. The proper data is difficult to obtain since it is not quite clear what the right data is. The data used in this study is freely available and may be obtained from Yahoo Finance using the given API. The study focuses on the daily performance of one major US stock market index, the S&P 500, utilizing data from 28/06/1991 through 10/08/2001. The initial data set contained the typical Open, High, Low, Close, and Volume values, and it was filtered by eliminating stocks that were no longer available in stock market and stocks with insufficient data to calculate the technical indicators, yielding a final data set of 338 stocks. Before the data is ready to be utilized as input into the model, it must go through numerous processes and transformations, which are detailed in the following sections.

4.1 Feature engineering

The fundamental characteristics of stock prices include open, high, low, close, and volume data, which are publicly available on the internet. These are used to construct more complex and informative features. These features, often known as "technical indicators", will be calculated using the mathematical formulas outlined below. Technical indicators are mathematical formulas that investors use to forecast the future movement of a stock or stock market index based on previous price and volume data. The technical indicators used are listed below:

- **ADX:** Average Directional Index is an indicator of trend dynamics and is a combination of two other technical indicators, the Positive Directional Indicator ($+DI$) and the Negative Directional Indicator ($-DI$). Let $+DM$ and $-DM$ be the positive and negative directional movement respectively and ATR be the Average True Range. The following indices are defined:

$$+DI = 100 \cdot \frac{SMMA_{14}(+DM)}{ATR} \tag{4.1}$$

$$-DI = 100 \cdot \frac{SMMA_{14}(-DM)}{ATR} \tag{4.2}$$

  Finally, the Average Directional Index is defined as:

$$ADX = 100 \cdot \frac{SMMA_{14}(|(+DI) - (-DI)|)}{(+DI) + (-DI)} \tag{4.3}$$

  where $SMMA_{14}$ is the Smoothed Moving Average for 14 days.

- **Bollinger Bands:** Bollinger Bands are a set of volatility indicators placed above and below a moving average.

$$HBand = SMA_{20}(C) + 2 \cdot \sigma_{20}(C) \tag{4.4}$$

$$LBand = SMA_{20}(C) - 2 \cdot \sigma_{20}(C) \tag{4.5}$$

  where SMA_{20} is the 20 day simple moving average, σ_{20} is the 20 day standard deviation and C is the closing price. $MBand$ is defined as the mean of $HBand$ and $LBand$:

$$MBand = \frac{HBand + LBand}{2} \tag{4.6}$$

and is essentially the 20-period moving average.

- **MACD:** Moving Average Convergence Divergence is a trend-following momentum indicator that shows the relationship between two moving averages of close prices.

$$MACD = EMA_{12}(C) - EMA_{26}(C) \tag{4.7}$$

$$MACD_{Histogram} = ACD - EMA_9(MACD) \tag{4.8}$$

where EMA_n is the exponential moving average of n days. In the above equations the choice of the number of days n is indicative and based on empirical estimates.

- **Parabolic SAR:** Parabolic Stop and Reverse is a trend following indicator that seeks to find support and resistance levels at the close price on an asset.

$$SAR_{i+1} = SAR_i + \alpha \cdot (EP - SAR_i) \tag{4.9}$$

where EP is the highest price in an uptrend or the lowest price in a downtrend and α is the accelerator factor.

- **RSI:** Relative Strength Index is a momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the trading of an asset.

$$RSI = 100 - \frac{100}{1 + \frac{AvgGain}{AvgLoss}} \tag{4.10}$$

The standard is to use 14 periods to calculate the initial $AvgGain$ and $AvgLoss$.

- **ROC:** Rate-of-Change indicator, which is also referred to simply as Momentum, is a pure momentum oscillator that measures the percent change in price from one period to the next.

$$ROC_i = \frac{C_i}{C_{i-1}} - 1 \tag{4.11}$$

Where C_i is the closing price on day i.

- **Stochastic Oscillator:** Stochastic Oscilliator is a momentum indicator that shows the location of the close price relative to the high-low range over a period.

$$\%K = 100 \cdot \frac{C - L14}{H14 - L14} \tag{4.12}$$

where C is the close price, L14 is the lowest low over the past 14 periods and H14 is the highest high over the past 14 periods.

- **Williams %R:** Williams %R is a momentum indicator and measures the overbought and oversold levels.

$$\%R = -100 \cdot \frac{H14 - C}{H14 - L14} \tag{4.13}$$

where C is the close price, L14 is the lowest low over the past 14 periods and H14 is the highest high over the past 14 periods.

Although the input features frequently interact in unexpected and nonlinear ways, a learning algorithm can identify and describe these relationships. A popular method is to create new features that reveal these interactions and then test if they improve model performance. Despite the higher expense of adding more features, these additions can occasionally improve the model's performance. Furthermore, transformations such as increasing input variables to a power can aid in revealing crucial correlations between input variables and the target variable.

Features that are formed by raising existing features to an exponent are known as polynomial features [10]. For example, if a data collection has only one input feature, X, the third-degree polynomial will add two new features (columns), X^2 and X^3. The polynomial's "degree" is used to regulate the amount of features added, e.g a degree of 4 will add three additional variables. A small degree, such as 3 or 4, is usually utilized. A degree of 4 was employed in this thesis to create the polynomial features for the close value.

4.2 Feature scaling

One of the most important phases in the pre-processing of data before developing a machine learning model is feature scaling as it can be the difference between a poor and a good machine learning model. A large disparity in feature range might lead to an incorrect prioritizing of some features over others. If one input has a huge variance and another has a small variance, the model will assume that the latter has little or no effect on the results. Scaling typically improves model performance and allows optimization

approaches to converge faster. Standardization 4.14, also known as z-score normalization, is a well-known scaling procedure in which each feature of the data set is transformed so that its distribution will have a mean value of zero and a standard deviation equal to one.

$$\begin{aligned} \text{Standardization: } z &= \frac{x-\mu}{\sigma} \\ \text{with mean: } \mu &= \frac{1}{N}\sum_{i=1}^{N}(x_i) \\ \text{standard deviation: } \sigma &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2} \end{aligned} \tag{4.14}$$

4.3 Training & testing methodology

The problem of stock prediction is treated as a regression problem where a unified model predicts the future prices of the assets. More precisely, the model predicts the stock prices for 1 month ahead. To simplify the backtesting procedure, holidays and days where the stock market was closed, have been forward filled with the most recent valid value. This means that each week will have exactly 5 working days.

In order to enable the model to perform better and adapt in the ever-changing nature of financial time series, a rolling window training approach is adopted where the model is trained several times on subsequent, non-overlapping subsets in order to update the optimal parameters. To this end, the retrain dates are first determined based on a predefined interval. This interval is 40 working days (or 8 work weeks). Before moving forward, several constants are defined:

- N is the number of stocks in each retrain subset.
- S is the number of look-back working days.
- T is the number of **daily** timesteps.
- F is the number of features.

The model is trained on a X_{train}, y_{train} pair where X_{train} is a 3-dimensional tensor with the dimensions being $(N \times S) \times T \times F$. Essentially, a $T \times F$ matrix is generated for

every stock and for every S_i. The last timestep of the $T \times F$ matrices match the date of each S_i, so there will be N 2-dimensional matrices for each S_i. y_{train} is simply a vector with the response values (stock close prices for 1 month ahead) for every $T \times F$ matrix, so there will be $N \times S$ elements in said vector.

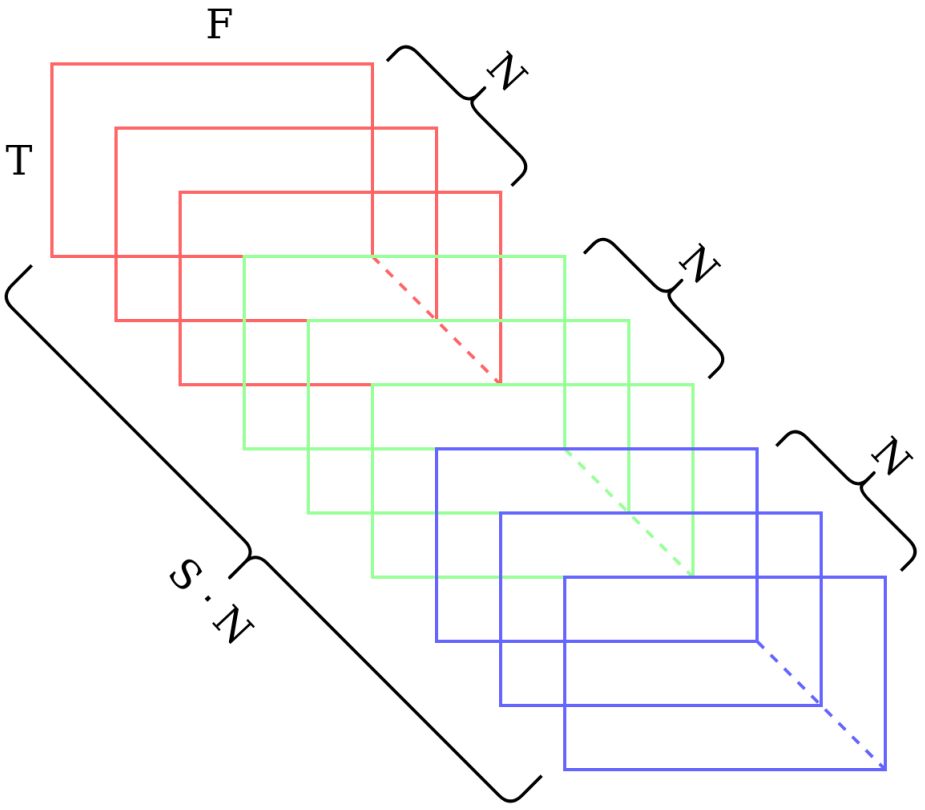

FIGURE 4.1: Structure of X_{train}.

Subsequently, the model is tested on X_{test}, y_{test} pairs for the following 40 working days. The shape of X_{test} is similar to X_{train} with the only difference being that $S = 1$ (i.e. only 1 look-back working day). Of course, y_{test} is a vector with N elements.

4.4 Models Grid Search Method

In this thesis two models were used and compared in terms of their predictability. Both are LSTM but the second one after the LSTM layers, has an Attention layer. In order to find the appropriate subset of parameters of the networks that will make the predictions, an exhaustive search using the Grid Search method is necessary. Specifically, the parameters that lead to the best model error metrics are searched for among a set of possible parameters. Table 4.1 presents the parameter sets and ranges.

Parameter	From	To	Step	Cases
Layers	1	3	1	3
Neurons	2	256	4,8, 16, 32, 64...	8
Dropout	0,0	0,4	0,2	3
Loss function	{mse, mae, huber}			3

TABLE 4.1: Grid Search Parameters

Each type of model is trained for each of the combinations ($3 * 8 * 3 * 3 = 216$ combinations in total), until the one that optimizes the error metrics is found. The decision on the best combination was made primarily based on MAPE (Mean Absolute Percentage Error).

Chapter 5

Results

In this section are presented the results obtained from the grid search method (Table 4.1) using various metrics which assess the performance of the models:

- **MSE:** is the Mean Square Difference between the actual and the predicted value 2.2
- **MAE** is the Mean Absolute Difference between the actual and the predicted value 2.3
- **MAPE** is the Mean Absolute Percentage Difference between the actual and the predicted value. The main difference from MAE is that MAPE is returned as a percentage.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} | \frac{Y_i - \hat{Y_i}}{Y_i} | \tag{5.1}$$

  where:

  - Y_i is the true value of the i^{th} instance
  - $\hat{Y_i}$ is the predicted value of the i^{th} instance
  - n is the size of the test set

- **Huber** [38] is less sensitive to outliers in data than the squared error loss. This function is quadratic for small values of α and linear for large values where $\alpha = Y_i -$

$\hat{Y}_i$. δ is a constant whose value varies according to the needs of the measurement.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases} \tag{5.2}$$

The calculated results are presented grouped each time by a parameter from Table 4.1 and sorted by **MAPE**. MAPE is more understandable to end users than MAE since it is expressed as a percentage, making model comparison easier. However, it should be noted that in this thesis, for the calculation of the metrics, the scikit-learn library at version 1.0 is used which, according to their documentation, does not return a percentage but a value [1].

neurons	dropout	Loss Function	Model type	MSE	MAE	MAPE	Huber
2	0.2	mae	lstm_plain	78.92	2.59	0.13	2.20
2	0.2	huber	lstm_plain	79.20	2.59	0.13	2.20
2	0.4	mae	lstm_plain	79.22	2.60	0.13	2.20
2	0.4	mse	lstm_plain	80.31	2.60	0.13	2.21
2	0	mae	lstm_plain	80.90	2.61	0.13	2.22
2	0	mse	lstm_plain	79.66	2.61	0.13	2.21
4	0.2	huber	lstm_plain	82.72	2.63	0.14	2.24
4	0	mae	lstm_plain	87.00	2.65	0.14	2.25
4	0	mse	lstm_plain	93.11	2.67	0.14	2.27
4	0.4	mse	lstm_plain	80.18	2.64	0.14	2.24
8	0.4	mse	lstm_plain	91.69	2.68	0.14	2.28
32	0.4	mae	lstm_plain	99.53	2.70	0.14	2.31
8	0.2	huber	lstm_plain	91.60	2.69	0.14	2.30
8	0.2	mae	lstm_plain	119.52	2.73	0.14	2.33
4	0.4	mae	lstm_plain	99.69	2.74	0.14	2.34
4	0	huber	lstm_plain	93.19	2.74	0.14	2.35
4	0.4	huber	lstm_plain	88.62	2.74	0.14	2.35
4	0.2	mae	lstm_plain	88.68	2.74	0.14	2.34
2	0.2	mse	lstm_plain	82.23	2.71	0.14	2.31
2	0.4	huber	lstm_plain	91.98	2.74	0.14	2.34

TABLE 5.1: The top 20 models based on mape with 1 layer

neurons	dropout	Loss Function	Model type	MSE	MAE	MAPE	Huber
2	0	mse	lstm_simple_attn	80.05	2.50	0.13	2.12
2	0.2	huber	lstm_simple_attn	80.88	2.59	0.13	2.19
2	0	mse	lstm_plain	91.30	2.64	0.14	2.25
8	0.2	mse	lstm_plain	80.70	2.63	0.14	2.24
8	0.4	mse	lstm_plain	83.25	2.64	0.14	2.25
4	0.2	huber	lstm_plain	82.75	2.63	0.14	2.24
2	0.2	huber	lstm_plain	82.68	2.66	0.14	2.26
16	0.4	mse	lstm_plain	89.08	2.63	0.14	2.24
16	0.4	mae	lstm_plain	81.21	2.64	0.14	2.24
8	0.2	huber	lstm_plain	96.47	2.64	0.14	2.24
8	0.4	mae	lstm_plain	81.26	2.65	0.14	2.26
2	0	mae	lstm_plain	85.51	2.68	0.14	2.28
4	0.2	mae	lstm_plain	83.90	2.65	0.14	2.26
8	0.4	huber	lstm_plain	83.99	2.65	0.14	2.25
2	0.2	mse	lstm_simple_attn	85.53	2.66	0.14	2.27
4	0.4	huber	lstm_plain	84.96	2.68	0.14	2.28
4	0.4	mse	lstm_plain	84.21	2.68	0.14	2.28
2	0.4	mse	lstm_plain	85.34	2.68	0.14	2.28
2	0.2	mae	lstm_plain	83.48	2.66	0.14	2.26
8	0.2	mae	lstm_plain	82.08	2.68	0.14	2.28

TABLE 5.2: The top 20 models based on mape with 2 layers

neurons	dropout	Loss Function	Model type	MSE	MAE	MAPE	Huber
2	0	mae	lstm_plain	65.57	2.30	0.12	1.92
2	0.4	mse	lstm_simple_attn	70.56	2.39	0.12	2.00
2	0	huber	lstm_plain	68.00	2.37	0.12	1.98
2	0.4	huber	lstm_simple_attn	72.62	2.43	0.13	2.05
2	0	mse	lstm_simple_attn	75.65	2.44	0.13	2.05
2	0.2	mae	lstm_simple_attn	74.24	2.45	0.13	2.07
2	0.2	huber	lstm_plain	73.89	2.45	0.13	2.07
4	0.4	huber	lstm_plain	74.34	2.50	0.13	2.11
2	0	huber	lstm_simple_attn	77.79	2.57	0.13	2.18
4	0.2	mae	lstm_plain	78.44	2.55	0.13	2.16
2	0	mse	lstm_plain	77.72	2.56	0.13	2.17
8	0.4	mse	lstm_plain	78.22	2.59	0.13	2.20

Table 5.3 continued from previous page

neurons	dropout	Loss Function	Model type	MSE	MAE	MAPE	Huber
4	0.4	mae	lstm_plain	82.44	2.61	0.13	2.21
4	0	huber	lstm_plain	83.78	2.61	0.13	2.22
2	0.4	mse	lstm_plain	82.55	2.60	0.14	2.21
8	0.2	mse	lstm_plain	79.99	2.60	0.14	2.21
16	0.4	mae	lstm_plain	80.72	2.61	0.14	2.21
4	0.2	huber	lstm_plain	83.38	2.62	0.14	2.23
8	0.2	huber	lstm_plain	85.82	2.62	0.14	2.22
4	0.4	mae	lstm_simple_attn	85.00	2.64	0.14	2.24

TABLE 5.3: The top 20 models based on mape with 3 layers

Based on Tables 5.1 to 5.3 we see that the MAPE is relatively at the same values. The same is true for the other metrics - MSE, MAE, Huber - as well as for dropout, loss function and model type, as all their possible values are shown. However, it is noteworthy that for $layer = 1$ there was no LSTM with attention model that achieved better performance than any other simple LSTM.

layers	neurons	Loss Function	Model type	MSE	MAE	MAPE	Huber
3	2	mae	lstm_plain	65.57	2.30	0.12	1.92
3	2	huber	lstm_plain	68.00	2.37	0.12	1.98
3	2	mse	lstm_simple_attn	75.65	2.44	0.13	2.05
2	2	mse	lstm_simple_attn	80.05	2.50	0.13	2.12
3	2	huber	lstm_simple_attn	77.79	2.57	0.13	2.18
3	2	mse	lstm_plain	77.72	2.56	0.13	2.17
1	2	mae	lstm_plain	80.90	2.61	0.13	2.22
1	2	mse	lstm_plain	79.66	2.61	0.13	2.21
3	4	huber	lstm_plain	83.78	2.61	0.13	2.22
2	2	mse	lstm_plain	91.30	2.64	0.14	2.25
3	4	mae	lstm_plain	96.21	2.65	0.14	2.25
2	2	mae	lstm_plain	85.51	2.68	0.14	2.28
1	4	mae	lstm_plain	87.00	2.65	0.14	2.25
1	4	mse	lstm_plain	93.11	2.67	0.14	2.27
3	8	mae	lstm_plain	110.12	2.68	0.14	2.29
3	8	mse	lstm_plain	149.65	2.70	0.14	2.31
3	256	huber	lstm_plain	105.46	2.69	0.14	2.30

Table 5.4 continued from previous page

layers	neurons	Loss Function	Model type	MSE	MAE	MAPE	Huber
3	4	huber	lstm_simple_attn	89.08	2.72	0.14	2.32
3	4	mse	lstm_plain	82.83	2.64	0.14	2.25
2	4	huber	lstm_plain	96.01	2.71	0.14	2.32

TABLE 5.4: The top 20 models based on mape for $dropout = 0$

layers	neurons	Loss Function	Model type	MSE	MAE	MAPE	Huber
3	2	mae	lstm_simple_attn	74.24	2.45	0.13	2.07
3	2	huber	lstm_plain	73.89	2.45	0.13	2.07
3	4	mae	lstm_plain	78.44	2.55	0.13	2.16
2	2	huber	lstm_simple_attn	80.88	2.59	0.13	2.19
1	2	mae	lstm_plain	78.92	2.59	0.13	2.20
1	2	huber	lstm_plain	79.20	2.59	0.13	2.20
3	8	mse	lstm_plain	79.99	2.60	0.14	2.21
3	4	huber	lstm_plain	83.38	2.62	0.14	2.23
3	8	huber	lstm_plain	85.82	2.62	0.14	2.22
3	2	mse	lstm_plain	83.26	2.63	0.14	2.23
3	4	mse	lstm_plain	82.17	2.63	0.14	2.23
2	8	mse	lstm_plain	80.70	2.63	0.14	2.24
2	4	huber	lstm_plain	82.75	2.63	0.14	2.24
2	2	huber	lstm_plain	82.68	2.66	0.14	2.26
1	4	huber	lstm_plain	82.72	2.63	0.14	2.24
3	16	huber	lstm_plain	88.07	2.64	0.14	2.24
2	8	huber	lstm_plain	96.47	2.64	0.14	2.24
2	4	mae	lstm_plain	83.90	2.65	0.14	2.26
3	8	mae	lstm_plain	95.86	2.68	0.14	2.28
3	2	huber	lstm_simple_attn	97.99	2.69	0.14	2.30

TABLE 5.5: The top 20 models based on mape for $dropout = 0.2$

layers	neurons	Loss Function	Model type	MSE	MAE	MAPE	Huber
3	2	mse	lstm_simple_attn	70.56	2.39	0.12	2.00
3	2	huber	lstm_simple_attn	72.62	2.43	0.13	2.05
3	4	huber	lstm_plain	74.34	2.50	0.13	2.11
1	2	mae	lstm_plain	79.22	2.60	0.13	2.20
1	2	mse	lstm_plain	80.31	2.60	0.13	2.21

Table 5.6 continued from previous page

layers	neurons	Loss Function	Model type	MSE	MAE	MAPE	Huber
3	8	mse	lstm_plain	78.22	2.59	0.13	2.20
3	4	mae	lstm_plain	82.44	2.61	0.13	2.21
3	2	mse	lstm_plain	82.55	2.60	0.14	2.21
3	16	mae	lstm_plain	80.72	2.61	0.14	2.21
3	4	mae	lstm_simple_attn	85.00	2.64	0.14	2.24
3	16	huber	lstm_plain	77.90	2.60	0.14	2.20
3	16	mse	lstm_plain	82.33	2.62	0.14	2.22
3	32	mae	lstm_plain	80.08	2.62	0.14	2.23
3	4	mse	lstm_plain	80.49	2.63	0.14	2.23
2	8	mse	lstm_plain	83.25	2.64	0.14	2.25
3	2	mae	lstm_plain	84.11	2.64	0.14	2.24
3	8	huber	lstm_plain	83.12	2.64	0.14	2.25
2	16	mse	lstm_plain	89.08	2.63	0.14	2.24
2	16	mae	lstm_plain	81.21	2.64	0.14	2.24
2	8	mae	lstm_plain	81.26	2.65	0.14	2.26

TABLE 5.6: The top 20 models based on mape for $dropout = 0.4$

According to Tables 5.4 to 5.6, the MAPE remains relatively constant for all dropout values. The same holds true for the other metrics, namely MSE, MAE, and Huber, as well as dropout, loss function, and model type. However, it is notable that the models with layer = 3, neurons = 2, and loss function Huber or MSE outperformed the others.

layers	neurons	dropout	Model Type	MSE	MAE	MAPE	Huber
3	2	0.00	lstm_plain	68.00	2.37	0.12	1.98
3	2	0.40	lstm_simple_attn	72.62	2.43	0.13	2.05
3	2	0.20	lstm_plain	73.89	2.45	0.13	2.07
3	4	0.40	lstm_plain	74.34	2.50	0.13	2.11
3	2	0.00	lstm_simple_attn	77.79	2.57	0.13	2.18
2	2	0.20	lstm_simple_attn	80.88	2.59	0.13	2.19
1	2	0.20	lstm_plain	79.20	2.59	0.13	2.20
3	4	0.00	lstm_plain	83.78	2.61	0.13	2.22
3	4	0.20	lstm_plain	83.38	2.62	0.14	2.23
3	8	0.20	lstm_plain	85.82	2.62	0.14	2.22
3	16	0.40	lstm_plain	77.90	2.60	0.14	2.20

Table 5.7 continued from previous page

layers	neurons	dropout	Model Type	MSE	MAE	MAPE	Huber
3	8	0.40	lstm_plain	83.12	2.64	0.14	2.25
2	4	0.20	lstm_plain	82.75	2.63	0.14	2.24
2	2	0.20	lstm_plain	82.68	2.66	0.14	2.26
1	4	0.20	lstm_plain	82.72	2.63	0.14	2.24
3	16	0.20	lstm_plain	88.07	2.64	0.14	2.24
2	8	0.20	lstm_plain	96.47	2.64	0.14	2.24
3	32	0.40	lstm_plain	89.52	2.66	0.14	2.27
3	2	0.20	lstm_simple_attn	97.99	2.69	0.14	2.30
2	8	0.40	lstm_plain	83.99	2.65	0.14	2.25

TABLE 5.7: The top 20 models based on mape with huber loss function.

layers	neurons	dropout	Model Type	MSE	MAE	MAPE	Huber
3	2	0.00	lstm_plain	65.57	2.30	0.12	1.92
3	2	0.20	lstm_simple_attn	74.24	2.45	0.13	2.07
3	4	0.20	lstm_plain	78.44	2.55	0.13	2.16
1	2	0.20	lstm_plain	78.92	2.59	0.13	2.20
1	2	0.40	lstm_plain	79.22	2.60	0.13	2.20
1	2	0.00	lstm_plain	80.90	2.61	0.13	2.22
3	4	0.40	lstm_plain	82.44	2.61	0.13	2.21
3	16	0.40	lstm_plain	80.72	2.61	0.14	2.21
3	4	0.40	lstm_simple_attn	85.00	2.64	0.14	2.24
3	4	0.00	lstm_plain	96.21	2.65	0.14	2.25
3	32	0.40	lstm_plain	80.08	2.62	0.14	2.23
3	2	0.40	lstm_plain	84.11	2.64	0.14	2.24
2	16	0.40	lstm_plain	81.21	2.64	0.14	2.24
2	8	0.40	lstm_plain	81.26	2.65	0.14	2.26
2	2	0.00	lstm_plain	85.51	2.68	0.14	2.28
1	4	0.00	lstm_plain	87.00	2.65	0.14	2.25
2	4	0.20	lstm_plain	83.90	2.65	0.14	2.26
3	8	0.20	lstm_plain	95.86	2.68	0.14	2.28
3	8	0.00	lstm_plain	110.12	2.68	0.14	2.29
3	16	0.20	lstm_plain	85.56	2.66	0.14	2.27

TABLE 5.8: The top 20 models based on mape with mae loss function.

layers	neurons	dropout	Model Type	MSE	MAE	MAPE	Huber
3	2	0.40	lstm_simple_attn	70.56	2.39	0.12	2.00
3	2	0.00	lstm_simple_attn	75.65	2.44	0.13	2.05
2	2	0.00	lstm_simple_attn	80.05	2.50	0.13	2.12
3	2	0.00	lstm_plain	77.72	2.56	0.13	2.17
1	2	0.40	lstm_plain	80.31	2.60	0.13	2.21
3	8	0.40	lstm_plain	78.22	2.59	0.13	2.20
1	2	0.00	lstm_plain	79.66	2.61	0.13	2.21
3	2	0.40	lstm_plain	82.55	2.60	0.14	2.21
3	8	0.20	lstm_plain	79.99	2.60	0.14	2.21
2	2	0.00	lstm_plain	91.30	2.64	0.14	2.25
3	2	0.20	lstm_plain	83.26	2.63	0.14	2.23
3	16	0.40	lstm_plain	82.33	2.62	0.14	2.22
3	4	0.20	lstm_plain	82.17	2.63	0.14	2.23
2	8	0.20	lstm_plain	80.70	2.63	0.14	2.24
3	4	0.40	lstm_plain	80.49	2.63	0.14	2.23
2	8	0.40	lstm_plain	83.25	2.64	0.14	2.25
2	16	0.40	lstm_plain	89.08	2.63	0.14	2.24
1	4	0.00	lstm_plain	93.11	2.67	0.14	2.27
2	2	0.20	lstm_simple_attn	85.53	2.66	0.14	2.27
3	4	0.40	lstm_simple_attn	86.43	2.68	0.14	2.28

TABLE 5.9: The top 20 models based on mape with mse loss function.

Similarly, from tables 5.7 to 5.9 appears that the metrics have similar values regardless of the loss function. The models with 3 Layers and 2 neurons achieved better performance while there seems to be a superiority of the simple LSTM compared to the LSTM with attention.

layers	neurons	dropout	Loss Function	MSE	MAE	MAPE	Huber
3	2	0.00	mae	65.57	2.30	0.12	1.92
3	2	0.00	huber	68.00	2.37	0.12	1.98
3	2	0.20	huber	73.89	2.45	0.13	2.07
3	4	0.40	huber	74.34	2.50	0.13	2.11
3	4	0.20	mae	78.44	2.55	0.13	2.16
3	2	0.00	mse	77.72	2.56	0.13	2.17
1	2	0.20	mae	78.92	2.59	0.13	2.20

Table 5.10 continued from previous page

layers	neurons	dropout	Loss Function	MSE	MAE	MAPE	Huber
1	2	0.20	huber	79.20	2.59	0.13	2.20
1	2	0.40	mae	79.22	2.60	0.13	2.20
1	2	0.40	mse	80.31	2.60	0.13	2.21
1	2	0.00	mae	80.90	2.61	0.13	2.22
3	8	0.40	mse	78.22	2.59	0.13	2.20
1	2	0.00	mse	79.66	2.61	0.13	2.21
3	4	0.40	mae	82.44	2.61	0.13	2.21
3	4	0.00	huber	83.78	2.61	0.13	2.22
3	2	0.40	mse	82.55	2.60	0.14	2.21
3	8	0.20	mse	79.99	2.60	0.14	2.21
2	2	0.00	mse	91.30	2.64	0.14	2.25
3	16	0.40	mae	80.72	2.61	0.14	2.21
3	4	0.20	huber	83.38	2.62	0.14	2.23

TABLE 5.10: The top 20 LSTM models based on mape.

layers	neurons	dropout	Loss Function	MSE	MAE	MAPE	Huber
3	2	0.40	mse	70.56	2.39	0.12	2.00
3	2	0.40	huber	72.62	2.43	0.13	2.05
3	2	0.00	mse	75.65	2.44	0.13	2.05
3	2	0.20	mae	74.24	2.45	0.13	2.07
2	2	0.00	mse	80.05	2.50	0.13	2.12
3	2	0.00	huber	77.79	2.57	0.13	2.18
2	2	0.20	huber	80.88	2.59	0.13	2.19
3	4	0.40	mae	85.00	2.64	0.14	2.24
3	2	0.20	huber	97.99	2.69	0.14	2.30
2	2	0.20	mse	85.53	2.66	0.14	2.27
3	4	0.40	mse	86.43	2.68	0.14	2.28
3	4	0.20	huber	84.48	2.70	0.14	2.30
3	4	0.20	mse	132.05	2.76	0.14	2.36
3	4	0.00	huber	89.08	2.72	0.14	2.32
3	8	0.20	huber	84.12	2.70	0.14	2.30
3	4	0.20	mae	86.77	2.71	0.14	2.32
3	2	0.00	mae	96.04	2.78	0.14	2.38
3	16	0.40	huber	82.51	2.70	0.14	2.30

Table 5.11 continued from previous page

layers	neurons	dropout	Loss Function	MSE	MAE	MAPE	Huber
3	8	0.20	mae	89.01	2.75	0.14	2.35
2	2	0.00	mae	90.85	2.76	0.14	2.36

TABLE 5.11: The top 20 LSTM models with attention based on mape.

As expected, the error metrics had similar values in the case of comparing the performance of LSTM with LSTM with attention. Comparing tables 5.10 and 5.11, however, it can be shown that there was no LSTM model with attention and only 1 layer that achieved good performance compared to the simple LSTM. It may also be stated that models with fewer neurons outperformed bigger ones.

layers	neurons	dropout	Loss Func.	Model Type	MSE	MAE	MAPE	Huber
3	2	0.00	mae	lstm_plain	65.57	2.30	0.12	1.92
3	2	0.40	mse	lstm_simple_attn	70.56	2.39	0.12	2.00
3	2	0.00	huber	lstm_plain	68.00	2.37	0.12	1.98
3	2	0.40	huber	lstm_simple_attn	72.62	2.43	0.13	2.05
3	2	0.00	mse	lstm_simple_attn	75.65	2.44	0.13	2.05
3	2	0.20	mae	lstm_simple_attn	74.24	2.45	0.13	2.07
3	2	0.20	huber	lstm_plain	73.89	2.45	0.13	2.07
2	2	0.00	mse	lstm_simple_attn	80.05	2.50	0.13	2.12
3	4	0.40	huber	lstm_plain	74.34	2.50	0.13	2.11
3	2	0.00	huber	lstm_simple_attn	77.79	2.57	0.13	2.18
3	4	0.20	mae	lstm_plain	78.44	2.55	0.13	2.16
3	2	0.00	mse	lstm_plain	77.72	2.56	0.13	2.17
2	2	0.20	huber	lstm_simple_attn	80.88	2.59	0.13	2.19
1	2	0.20	mae	lstm_plain	78.92	2.59	0.13	2.20
1	2	0.20	huber	lstm_plain	79.20	2.59	0.13	2.20
1	2	0.40	mae	lstm_plain	79.22	2.60	0.13	2.20
1	2	0.40	mse	lstm_plain	80.31	2.60	0.13	2.21
1	2	0.00	mae	lstm_plain	80.90	2.61	0.13	2.22
3	8	0.40	mse	lstm_plain	78.22	2.59	0.13	2.20
1	2	0.00	mse	lstm_plain	79.66	2.61	0.13	2.21

TABLE 5.12: The top 20 models based on mape

Table 5.12 summarizes the top 20 models. The highest performing model is a basic LSTM with three layers, two neurons, no dropout, and mae as a loss function. Tables 5.1 to 5.12 show that models with smaller architectures perform better and that dropout, loss function, and model type, while affecting performance, have a minor influence.

Chapter 6

Conclusions and future work

In the previous chapters, the application of our methodology and the results of the experimentation process were described. This chapter discusses the overall results that have emerged from the current research and summarizes the overall research conducted for this thesis.

This work presented a complete framework for stock market prediction using LSTM models. The stock market index S&P 500 was studied in a 10-year period, from 28/06/1991 to 10/08/2001. A novel rolling window approach was utilized for the training procedure, where each model was trained on subsequent, non-overlapping subsets so that the weights of the model are updated regularly to capture the ongoing trends. The predictive models used in this work are simple LSTM models and LSTM models with the Attention mechanism, with structurally simple architectures to enable quick training. The work successfully combined aspects and best practices found in works in the field of stock market prediction, such as a long out-of-sample period, the use of technical indicators as input features and a rolling window approach for the training procedure. Also, a grid search method was used to determine the most suitable architecture, which involves a detailed search within a "grid" of hyperparameters whose boundaries have been defined by the relevant literature.

The experimental results showed that models with small architecture outperformed bigger models and that dropout, loss function, and model type, while affecting performance, they have a minor influence.

Although this study has been quite extensive, there is still room for further experimentation. Future work may include the investigation of the behavior of other machine learning models such as SVM, Ridge regression or Lasso (Least Absolute Shrinkage and Selection Operator). These models have been proven to perform well on regression tasks although it is not quite clear how they perform on time series.

Another major improvement in this framework would be to utilize the predictions that have been made from the models in order to construct an optimal portfolio based on the Mean-Variance optimization framework (Modern Portfolio Theory). The predictions may enable the optimization algorithm to select the stocks that are expected to yield the highest return.

Bibliography

[1] sklearn.metrics.mean_absolute_percentage_error. https://scikit-learn.org/1.0/modules/generated/sklearn.metrics.mean_absolute_percentage_error.html#sklearn.metrics.mean_absolute_percentage_error. Accessed: November 21, 2022.

[2] Ratnadip Adhikari and RK Agrawal. A combination of artificial neural network and random walk models for financial time series forecasting. *Neural Computing and Applications*, 24(6):1441–1449, 2014.

[3] american_express. A Comprehensive Guide to Attention Mechanism in Deep Learning for Everyone. https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/, 2020. Accessed: September 14, 2022.

[4] Aphex34. Typical CNN architecture. https://commons.wikimedia.org/wiki/File:Typical_cnn.png, 2015. Accessed: August 7, 2022.

[5] Ricardo de A Araújo and Tiago AE Ferreira. A morphological-rank-linear evolutionary method for stock market prediction. *Information Sciences*, 237:3–17, 2013.

[6] Yujin Baek and Ha Young Kim. Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module. *Expert Systems with Applications*, 113:457–480, 2018.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

[9] Fabjola Filio Sofia Braho. *Stock Price Prediction with LSTM Ensembles*. PhD thesis, Aristotle University of Thessaloniki, 2020.

[10] Jason Brownlee. How to Use Polynomial Feature Transforms for Machine Learning. `https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/`, 2020. Accessed: August 21, 2022.

[11] Dan Caplinger. Different Types of Stocks to Invest In: What Are They? `https://www.fool.com/investing/stock-market/types-of-stocks/`, 2021. Accessed: July 14, 2022.

[12] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE international conference on big data (big data)*, pages 2823–2824. IEEE, 2015.

[13] Shun Chen and Lei Ge. Exploring the attention mechanism in lstm-based hong kong stock price movement prediction. *Quantitative Finance*, 19(9):1507–1515, 2019.

[14] Eunsuk Chong, Chulwoo Han, and Frank C Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83:187–205, 2017.

[15] What is 'Stocks'. `https://economictimes.indiatimes.com/definition/stocks`. Accessed: July 15, 2022.

[16] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1415–1425, 2014.

[17] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[18] IBM Cloud Education. Unsupervised Learning. `https://www.ibm.com/cloud/learn/unsupervised-learning`, 2020. Accessed: July 31, 2022.

[19] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, 270(2):654–669, 2018.

[20] Hakan Gunduz, Yusuf Yaslan, and Zehra Cataltepe. Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, 137:138–148, 2017.

[21] Zhiqiang Guo, Huaiqing Wang, Jie Yang, and David J Miller. A stock market forecasting model combining two-directional two-dimensional principal component analysis and radial basis function neural network. *PloS one*, 10(4):e0122385, 2015.

[22] Adam Hayes. Stocks. `https://www.investopedia.com/terms/s/stock.asp`, 2022. Accessed: July 14, 2022.

[23] Heli Helskyaho, Jean Yu, and Kai Yu. Machine learning for oracle database professionals.

[24] Ma Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Nse stock market prediction using deep-learning models. *Procedia computer science*, 132:1351–1362, 2018.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Hao Hu and Guo-Jun Qi. State-frequency memory recurrent neural networks. In *International Conference on Machine Learning*, pages 1568–1577. PMLR, 2017.

[27] Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. Mining of concurrent text and time series. In *KDD-2000 Workshop on text mining*, volume 2000, pages 37–44. Citeseer University Park, PA, USA, 2000.

[28] Wen Long, Zhichen Lu, and Lingxiao Cui. Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, 164:163–173, 2019.

[29] Manu. LSTM chain of 3 nodes and internal structure depiction. `https://medium.com/swlh/`

a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b, 2021. Accessed: September 11, 2022.

[30] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, and Victor Chang. An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, 76(3):2098–2118, 2020.

[31] Reinforcement Learning Tutorial. https://www.javatpoint.com/reinforcement-learning. Accessed: August 2, 2022.

[32] Arthur L Samuel. Some studies in machine learning using the game of checkers. ii—recent progress. *IBM Journal of research and development*, 11(6):601–617, 1967.

[33] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

[34] Ritika Singh and Shashi Srivastava. Stock prediction using deep learning. *Multimedia Tools and Applications*, 76(18):18569–18584, 2017.

[35] What is a Stock? https://learn.robinhood.com/articles/6FKal8yK9kk22uk65x3Jno/what-is-a-stock/, 2021. Accessed: July 14, 2022.

[36] Chih F Tsai and Sammy P Wang. Stock price forecasting by hybrid machine learning techniques. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, page 60, 2009.

[37] Wikipedia. Convolutional neural network. https://en.wikipedia.org/wiki/Convolutional_neural_network, 2022. Accessed: August 17, 2022.

[38] Wikipedia. Huber loss. https://en.wikipedia.org/wiki/Huber_loss, 2022. Accessed: November 20, 2022.

[39] Wikipedia. Supervised learning. http://en.wikipedia.org/w/index.php?title=Supervised%20learning&oldid=1093417731, 2022. Accessed: July 29, 2022.

[40] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2141–2149, 2017.

[41] Xiao Zhong and David Enke. Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications*, 67:126–139, 2017.