Ministry of Education and Science of Ukraine

Kharkiv National University of Radioelectronics

Faculty: Computer Science

Department: Artificial Intelligence

Bachelor Thesis

Development of an Intelligent Assistant for Trading Systems Based on a Recurrent Neural Network

Author:
Maksym Alieksieiev
4th-year student
Group: ITSHI-21-1
Specialty: 122 Computer Science
Program Type: Educational and Professional
Educational Program: Artificial Intelligence

Supervisor:
Oleg Zolotukhin
Associate Professor, PhD

2025

ABSTRACT

Bachelor’s thesis contains: 75 pages, 24 figures, 3 tables, 2 appendices, 22 references.

ECONOMIC FORECASTING, INTELLIGENT SYSTEM, NEURAL NETWORKS, STOCK MARKET, JAVASCRIPT APPLICATION, REACT, TENSORFLOW.

The object of research is stock markets and the rules by which they operate.

The aim of this work is to develop an intelligent virtual assistant for financial forecasting of stock market data, using the JavaScript programming language.

The research methods include analysis of the subject area of financial markets, examination of neural network concepts, and review of literature and electronic resources.

The result of the bachelor thesis is a developed demo version of an intelligent virtual assistant for forecasting stock market data of selected companies.

TABLE OF CONTENTS

INTRODUCTION

The stock market, also known as the securities market, is a public platform for trading shares of various companies. Stock market forecasting refers to the attempt to determine the future price of a company's stock. Accurate price forecasting can yield significant profits for traders. One of the main hypotheses suggests that stock prices are formed based on all available information in the world about a given company. The desire to gain insight into the future state of the stock market has driven the development of several forecasting methodologies. There are three main categories of methodologies used for predicting stock prices: fundamental analysis, technical analysis, and technological methods. Each of these approaches aims to maximize the accuracy of predictions. They typically focus either on forecasting the exact value of a future price or predicting the direction of its movement within a specified time interval.

The challenge of accurately predicting stock prices remains relevant to all market speculators. This thesis proposes a solution based on data mining and machine learning technologies, specifically neural networks. Neural networks are not a novel approach to this problem — on the contrary, they are widely used. The most commonly applied architecture is the feedforward neural network trained using the backpropagation algorithm, which is the core method for training multilayer perceptrons.

Neural networks are used in financial analysis as tools that simulate agents trading on stock or other markets, allowing for the potential to achieve significant returns.

One of the most critical challenges in working with neural networks is the ambiguity surrounding input data preprocessing, as no universally accepted methodology exists for evaluating the effectiveness of a given approach. This can be attributed to the vast variety of potential input data, which makes it nearly impossible to design a universal classifier and methodology for each data type.

Therefore, this thesis dedicates a separate section to the process of input data preprocessing and the selection of dependent variables, which together aim to improve the effectiveness of stock market forecasting. In the practical part of the work, software will be developed that uses a neural network as the computational engine for forecasting. The output will consist of specific economic indicators, predicted from the numerical input data. The accuracy of the developed application will be assessed by comparing the neural network's predictions with real values from a test dataset.

The results are presented in the form of an explanatory note, which includes the following sections: description of the automation object; review of the current state of the problem; requirements for the developed system; requirements for the system's functional structure; requirements for supporting components; problem definition; development of informational support elements; development, justification, and evaluation of the problem-solving algorithm; development of software components; selection and justification of the hardware platform; information security; and methodological recommendations for using the system.

1 SUBJECT DOMAIN ANALYSIS AND PROBLEM STATEMENT

1.1 Analysis of the Stock Market

The development of the market and the complication of commodity-money relations have objectively led to the formation and improvement of its infrastructure. Market infrastructure is a system of specialized organizations intended to support the functioning of specific markets. In particular, wholesale and retail trade systems, as well as commodity exchanges, operate in the markets for goods and services. Trading enterprises act as the primary elements of market infrastructure, ensuring the organization of commodity flows and the collection of necessary information on the state of individual markets for goods and services.

A modern market is understood as any system that enables buyers and sellers to freely conduct the purchase and sale of goods. A highly developed modern market performs six basic interrelated functions:

– intermediary function;
– price formation function;
– information function;
– regulatory function;
– incentive function;
– stabilizing function.

Like any objectively existing system, the market has its own infrastructure. The term “infrastructure” was first used in economic analysis at the beginning of the last century to refer to facilities and structures that ensured the normal functioning of the armed forces. In the 1940s, in the West, infrastructure came to be understood as the aggregate of sectors that contribute to the normal operation of the production of material goods and services. In relation to the market (market economy), infrastructure represents a set of organizational, legal, and economic relationships that unite these connections, despite their diversity, into a coherent whole. A market economy requires infrastructure—a system of interrelated specialized organizations that serve the flow of goods, services, money, securities, and labor. The classification of market infrastructure and its specific content are determined by the characteristics of the markets in which these elements operate. The stock market is a place for trading ownership of securities, where prices for these securities are determined by supply and demand, and the buying and selling process is regulated by rules and norms.

1.1.1 The Stock Market and Its Functions

An exchange is a non-profit organization, i.e., a non-commercial entity exempt from paying income tax. To cover the costs of organizing exchange trading, various fees are charged to participants (a fee for a transaction concluded on the trading floor; companies' payments for including their securities in the exchange list; annual contributions from exchange members, etc.). All these fees make up the main sources of the exchange's income, which are used for maintaining the exchange institution and paying its employees.

The role of the stock exchange in a country's economy is primarily determined by the degree of privatization, more precisely, by the share of joint-stock ownership in the production of gross national product. Additionally, the role of the exchange depends on the level of development of the stock market as a whole.

Characteristics of a classical stock exchange:

– it is a centralized market with a fixed trading location, i.e., a physical trading floor;

– this market has a procedure for selecting the best products (securities) that meet certain criteria (financial stability and large size of the issuer, wide distribution of the security as a uniform and standardized product, broad demand, etc.);

– existence of a trading time schedule for securities and standard trading procedures;

– centralization of deal registration and settlements;

– establishment of official (exchange) quotations;

– supervision of exchange members (in terms of financial soundness, safe business conduct, and adherence to stock market ethics).

Based on this, we can conclude that the stock exchange ensures the concentration of supply and demand for securities, their balance through exchange-based price formation, and accurately reflects the efficiency level of joint-stock capital functioning.

The items sold on the stock exchange are securities, and the prices for these goods are the market quotations of those securities. The method of price determination at the exchange is auction-based — the price is called out. Electronic exchanges also exist, but classical exchanges remain dominant.

Not all products can be sold on the exchange — only those that are admitted. Moreover, the method of buying and selling is much more varied and strictly regulated. Securities that are traded on the exchange must go through a listing procedure (being added to the exchange quotation list), after which such securities are called listed. To be included in this list, a company's securities must meet certain requirements set by the exchange. In particular, these requirements may relate to the revenue level, amount of working capital, market price of the securities issue, volume of publicly placed securities, etc. These requirements may differ across exchanges. After a listing application is submitted, a special exchange committee reviews it, and if both parties' requirements are met, the company signs an official listing agreement with the exchange.

The main functions of an exchange are: mobilizing temporarily free funds through the sale of securities; determining their market value; directing capital flows among sectors, companies, and industries; enabling circulation of securities on the secondary market; creating a continuously functioning market for high-quality securities; forming prices for securities; disseminating information about listed securities (principle of "information transparency"); maintaining professionalism among participants and, to some extent, supervising their activities; and connecting buyers and sellers for deal conclusion. The exchange serves as a barometer of business activity, an indicator of market conditions, and reflects the cyclical nature of reproduction.

Most company shares are traded on virtual stock websites. All actual trading takes place in the form of a specific type of auction, where a potential buyer announces the price they are willing to pay for the shares, and the potential seller, in turn, announces their asking price for the intellectual property they possess.

For a long time, stock market players were investors with long-term relationships with a particular company listed on the exchange. At present, participant types vary from small investors owning shares in a few companies to entire investment funds.

1.1.2 Securities as an Exchange-Traded Commodity

A security is a document that, when issued in a prescribed form and containing the required legal attributes, certifies proprietary rights. The exercise and transfer of these rights are only possible upon presentation of the document. In other words, securities certify the property rights of the holder, which can only be realized and transferred when the security is presented.

Securities must meet the following criteria:

- negotiability;
- availability for civil turnovers;
- standardization and series issuance;
- document form;
- state regulation and legal recognition;
- marketability;
- liquidity;
- risk exposure.

Securities, as objects of civil rights, are freely transferable from one person to another through universal succession and are not restricted in circulation.

The functions of securities are highly significant for society. First, they help attract financial resources into various sectors of the economy and national industries. Second, they establish specific rights for their holders, such as: the right to receive a portion of the issuer’s profit in the form of dividends; the right to priority dividend payments; the right to recover the principal with interest; the right to participate in management; and others.

Securities are mass-issued, standardized, and interchangeable. They are bought and sold at market-determined prices, which makes them suitable to be considered as exchange-traded commodities.

On stock exchanges, various types of securities are traded. These can be classified according to the type of property rights they confer: either rights to the underlying asset of the security or rights to changes in its price. This classification is traditional, though not the only possible one:

– basic securities – securities based on property rights to a specific asset (typically goods, money, capital, property, or various resources);

– primary securities – securities based on assets that do not include other securities themselves (e.g., shares, bonds, promissory notes, etc.).

– secondary securities – issued based on primary securities. These are securities backed by other securities (e.g., depositary receipts);

– derivative securities – emerged as the stock market developed and operations with securities became more complex, requiring formalization of trade agreements. They represent financial contracts for making transactions with securities at predetermined terms;

– production securities – non-documentary forms representing property rights (or obligations) that arise due to changes in the price of the underlying asset. Underlying assets can include commodities (grain, meat, oil, gold, etc.), basic securities (stocks and bonds), and others. Futures contracts and options fall under this category;

– futures contract – a standardized agreement to buy a specific number of securities during a defined period at a base price set when the contract is concluded. These contracts are strictly standardized and reflect specific expectations of buyers and sellers. A futures contract is an agreement in which one party sells a certain quantity of securities to another at a fixed price, with the obligation to execute the transaction by a specified future date rather than immediately. The buyer must accept the securities and pay the agreed amount regardless of the market value of the securities at that time. Thus, the moment of fulfillment by both parties does not coincide with the date of the agreement. At the time of selling a futures contract, the seller may not actually possess the securities offered, expecting to purchase them before the contract's execution date at a price lower than the contracted price.

The essence of an option lies in establishing a contract that grants the right to buy or sell a certain quantity of securities. The option buyer pays the seller a fee (premium), and may choose to exercise or not exercise the purchased right. The variety of market conditions and trading strategies involving options, as well as their combinations with futures, make these financial instruments quite attractive to investors.

Stock exchange trading of securities carries significant risk, so it is important to accurately assess securities from the perspective of the risks inherent to each type. There is particularly high risk when dealing with commercial securities issued by private companies. Firstly, there is the risk of capital loss invested in the securities due to the possibility of issuer bankruptcy; secondly, the risk of losing liquidity, meaning the purchased security cannot be sold on the market without incurring substantial price losses; and thirdly, market risk, i.e., the risk of price decline caused by a deterioration in overall market conditions.

When evaluating the risk levels of various securities, derivative securities (futures, options) are considered high-risk but potentially more profitable, while government securities and corporate bonds are seen as low-risk and less profitable.

Classification of securities refers to dividing securities into types based on certain characteristics. Securities can be classified according to the following criteria:

- term of existence – fixed-term, indefinite;
- origin – primary, secondary;
- form of existence – paper, dematerialized (paperless);
- national affiliation – domestic, foreign;
- type of use – investment, non-investment;
- ownership order – bearer, registered, order;
- form of issuance – issued (emission), non-issued (non-emission);
- form of ownership – state-owned, non-state-owned;
- nature of circulation – marketable, non-marketable;
- risk level – high, low;
- income presence – income-bearing, non-income-bearing;
- form of capital contribution – debt, equity, participatory;
- economic essence (type of rights) – shares, bonds, bills of exchange, etc.;
- degree of protection – high-grade, low-grade;
- scope of rights granted – with ownership rights, with management rights, with credit rights;
- territory of circulation – municipal, state, foreign, nationwide;
- form of income receipt – with fixed income, with lump-sum income;
- exchangeability – convertible, non-convertible.

In economically developed countries, the classification of securities is primarily determined by their intended purpose, which in turn defines the conditions of issuance, quotation, and profitability. According to this criterion, securities are divided into:

- shares of private, mixed, and state-owned companies;
- bonds of private companies and corporations;

− government bonds issued by the central government and local authorities;

− other types of securities.

There are certain standards in the securities market, which represent a combination of economic, legal, and technical requirements for securities.

A share (stock) is a security without a fixed term of circulation that certifies the contribution of funds and grants its holder the right to receive a portion of the company's profits in the form of dividends.

Shares serve three main purposes. First, their issuance is necessary when organizing a joint-stock company to provide the new enterprise with a certain initial capital to commence its business activities.

Second, they allow for the attraction of additional financial resources during ongoing business operations.

Third, the issuance of shares is used for exchange purposes, such as mergers with another company.

The profitability of shares is determined exclusively by the payment of dividends on them.

Shares can be regarded as units measuring the ownership interests of members of a joint-stock company, or shareholders. A share, as an object of ownership rights, represents a category of rights depending on its type: the right to vote; the right to participate in the company’s profits (receive dividends); the preemptive right to purchase new shares; rights upon the liquidation (dissolution) of the corporation; and the right to inspection (audit).

Currently, shares are becoming increasingly diverse, as joint-stock companies cannot limit themselves to a single type of security. There is a need to issue securities with various characteristics to balance the cost of capital and risk. Shares are classified into common and preferred shares. Preferred shares have distinct features that differentiate them from common shares.

As high-risk securities, shares generally attract investors with the possibility of higher returns, which may consist of dividend payments and capital gains from an increase in share price. Due to their higher profitability, shares usually provide better protection against inflation. Therefore, the main motive driving investors to invest in shares is the desire to achieve capital growth through price appreciation and to receive substantial dividends.

In addition to equity securities, such as shares, debt securities—bonds—are also traded in the securities market. Their issuers include government and local authorities, as well as joint-stock companies.

A bond is a security that certifies the holder's contribution of funds and confirms the issuer's obligation to repay the bond's nominal value at the specified maturity date, with payment of a fixed interest rate (unless otherwise stipulated by the terms of issuance).

Main differences between a bond and a share:

– bonds generate income only during the specified term;

– unlike the non-guaranteed dividend on common shares, a bond usually provides its holder with income in the form of a predetermined interest rate based on its nominal value;

– a bond issued by a joint-stock company does not grant its holder the rights of a shareholder in that company, meaning it does not confer voting rights at the general shareholders' meetings.

Other main types of securities include bills of exchange, depositary receipts, checks, savings certificates, and mortgages, which have a secondary, derivative nature in relation to the existing shares and bonds on the market. The objects of exchange trading include options on securities and, to a limited extent, depositary receipts.

A bill of exchange is a type of security representing a monetary obligation — an unconditional and incontestable debt instrument. In international trade, as well as in the domestic circulation of capitalist countries, the bill of exchange is one of the primary means of formalizing credit and settlement relations.

Bills of exchange can be:

− simple (paid by the borrower and contain an obligation to pay the creditor);

− transferable (contain a written order from the bill holder to the payer to pay the specified sum of money to a third party);

− term (with a specified payment date);

− payable on demand (a bill without a specified payment date);

− treasury;

− bank;

− commercial.

Warrant (subscription certificate) – a certificate that gives the holder the right to purchase securities (shares, bonds) at a certain price and within a specified period.

Depositary receipt – a certificate confirming the deposit of shares. They have a number of restrictions: they are not allowed for circulation on the US stock market, and in Europe they can be traded only on the over-the-counter market. Demand for depositary receipts in the Ukrainian market is characterized by significant growth.

Cheque – a document of a prescribed form containing a written order of the drawer to the bank to pay the holder of the cheque the amount indicated therein. Thus, a cheque is, in essence, a type of bill of exchange, but with certain peculiarities.

Savings certificate – a written certificate issued by a credit institution confirming the deposit of funds, which certifies the holder’s right to receive the deposit (savings) amount and interest on it after the expiry of the established term. Savings certificates are intended for individuals. They are issued by banks at an interest rate determined by the agreement, for a specified term or on demand. If the holder of a certificate demands the return of the deposited funds under a term certificate before the agreed-upon term, interest is paid in an amount determined on a contractual basis at the time of deposit.

Mortgage bond – a registered security that certifies the right to receive property secured by a mortgage after the fulfilment of a monetary obligation, as well as the right of lien to the property specified in the mortgage agreement. Mortgage bonds are subject to mandatory state registration. They are widely used in the issuance of pawn loans.

1.1.3 Characteristics of the Stock Market from the Perspective of Economic Predictability

The functioning of the stock market in a market economy plays a crucial role in the self-regulation of the economy and in performing a range of functions, such as:

- mobilization of temporarily idle funds through the sale of securities;
- establishing their market value;
- reallocating capital between industries, companies, and sectors;
- trading securities on the secondary market;
- determining prices for securities (contributing to their formation);
- maintaining the professionalism of market participants and, to some extent, monitoring their activities;
- shaping a flexible institutional structure of the economy.

Stock market data is among the most extensively studied, as it serves as a primary source of income for numerous individuals and legal entities. However, the behavior of the stock market is often quite non-trivial and sometimes unpredictable for humans, as it is influenced by a wide range of factors.

In periods of market stability within a market economy, it is possible to construct forecasts, yet it is important to consider the impact of various factors that may introduce distortions or deviate the prediction from actual outcomes.

Various methods have been employed to obtain information about future price values, but this study proposes to examine an approach based on the application of neural networks.

1.2 Analysis of Neural Networks

1.2.1 General Information about Artificial Neural Networks

Artificial neural networks (ANNs) are commonly understood as computational systems capable of self-learning and gradually improving their performance. The main structural components of neural networks include:

– artificial neurons, which are elementary, interconnected units;

– synapse – a connection used to send and receive information between neurons;

– signal – the actual information being transmitted.

The transmission of a signal from one neuron to another through synapses is a complex chemical process, during which specific transmitter substances are released by the neuron sending the signal through the connection. The effect results in either an increase or decrease in the electrical potential within the body of the receiving cell. If this potential exceeds a certain threshold, the neuron initiates a corresponding response. Artificial neural networks aim to replicate this very characteristic. The neural model shown in Figure 1.1 is the most common and frequently used model.

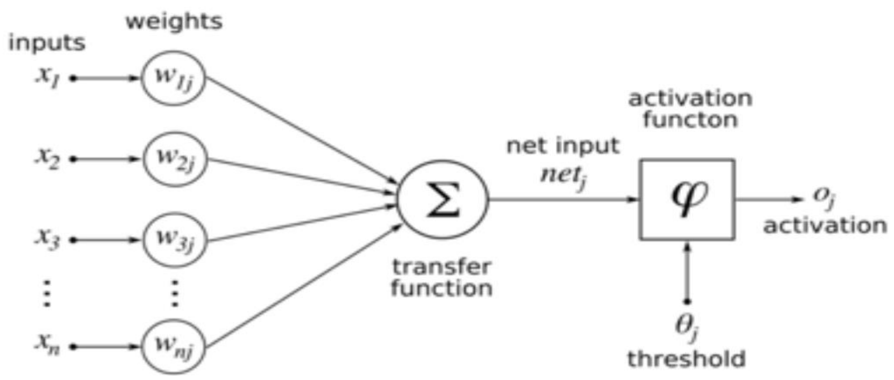


Figure 1.1 – Structure of an artificial neuron

Each set of connections (x) is described by corresponding weights (w). The weight of a neuron can take positive or negative values depending on the effect of the input signal. The input signals are summed together by a summator, which performs a linear combination operation. Finally, an activation function transforms the output signal into a normalized value within a specified range.

It is assumed that in a neural network with such a structure, information passes sequentially through layers, one after another, in a single direction, ultimately transforming into a final output value. Such a network is called a feedforward network.

Every neural network must contain at least two components: input data and an output layer. The number of hidden layers and the number of neurons within them are unlimited and depend on the task specification. Figure 1.2 depicts a typical two-layer feedforward neural network.

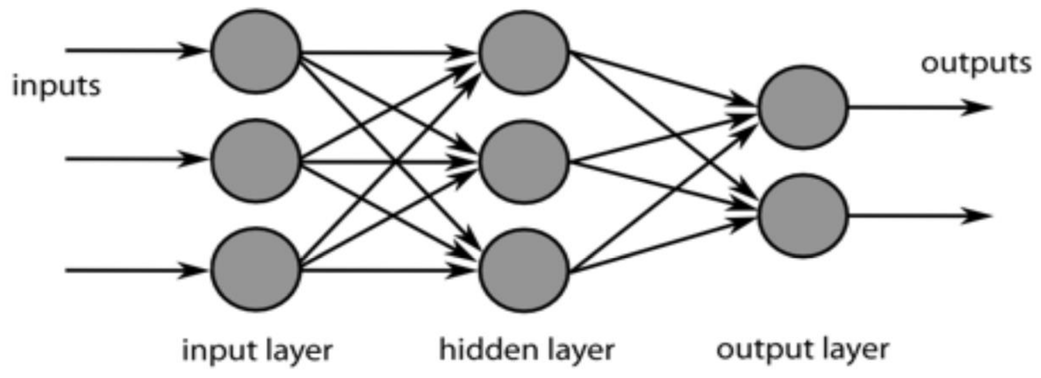


Figure 1.2 – Diagram of a two-layer (input layer not included) feedforward neural network

The input value of each neuron is weighted similarly to the synaptic connections in biological neurons. The activation potential formula can be expressed as follows:

$$S = \sum_{j=1}^{N} w_i x_j + \theta, \tag{1.1}$$

where S – neuron activation;

X – input value of the neuron;

w – neuron weight;

n – number of neurons in the previous layer;

θ – activation threshold.

The weighted sum S of a hidden layer neuron, calculated by the formula above, serves as the argument for the nonlinear function f (S), which produces the output value of the hidden layer Y (see Figure 1.3):

$$Y = f(S). \tag{1.2}$$

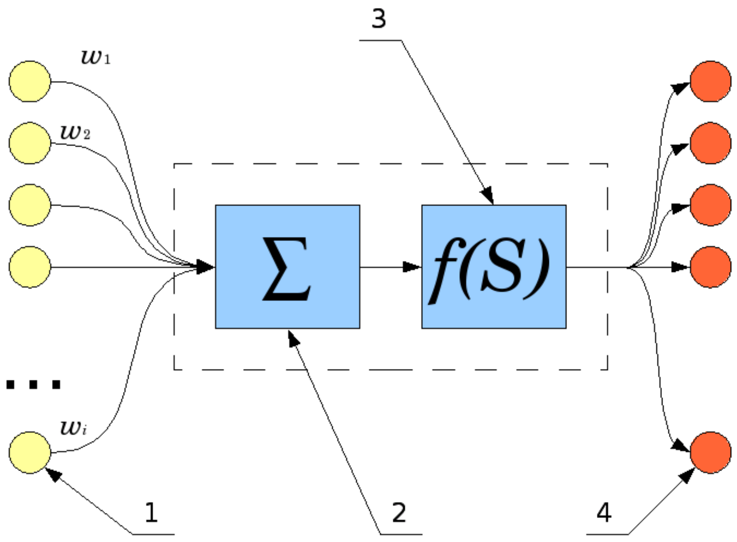


Figure 1.3 – Diagram of summing neuron values in the hidden layer

It is the nonlinear function that enables solving complex tasks. This type of function was also inspired by biological neural networks. It is believed that biological neurons in the human brain decide how to respond to incoming information. This decision-making process is formally called the activation function (see Figure 1.4).

There are numerous activation functions used in designing the architecture of neural networks. For example, when constructing artificial neural networks, the following types of activation functions may be employed.

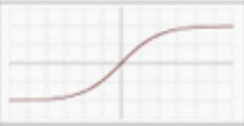
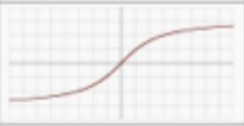
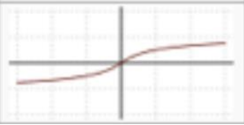
Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \dfrac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \dfrac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign [7]		$f(x) = \dfrac{x}{1 + \|x\|}$

Figure 1.4 – Activation functions

In practice, only a few activation functions are most commonly used. For example, the logistic activation function, which produces values in the range from 0 to 1, is very popular.

The field of application of artificial neural networks is expanding every year, and today they are used for the following purposes:

– pattern recognition and classification. Patterns can be objects of various nature: text symbols, images, sound samples, etc. During training, the network is presented with different pattern samples along with their corresponding class labels. When the network is given a certain pattern, one of its outputs should indicate that the pattern belongs to that class. At the same time, other outputs should indicate that the pattern does not belong to those classes;

– decision making and control. This task is closely related to classification. Situations whose characteristics are fed into the neural network are subject to classification. The network's output should indicate the decision it has made;

– clustering. Clustering refers to dividing a set of input signals into classes without prior knowledge of the number or characteristics of these classes. After training, such a network can determine to which class an input signal belongs;

– forecasting. The forecasting ability of a neural network directly stems from its capacity to generalize and identify hidden dependencies between input and output data. After training, the network can predict the future value of a sequence based on several previous values or other available factors;

– data compression and associative memory. Neural networks' ability to detect relationships between different parameters allows data of high dimensionality to be represented more compactly, especially if the data are closely interrelated. The reverse process — restoring the original data set from partial information — is called associative memory.

1.2.2 Training Methods of Artificial Neural Networks

Artificial neural networks are not programmed in the conventional sense of the word; they learn. The ability to learn is one of the main advantages of neural networks over traditional algorithms. Technically, learning involves finding the coefficients of the connections between neurons. During the training process, a neural network can detect complex dependencies between input and output data, as well as perform generalization.

For the training process, it is necessary to have a model of the external environment in which the neural network operates — the information needed to solve the task. Secondly, it is essential to determine how to modify the network's weight parameters.

1.2.2.1 Supervised Learning

Supervised learning assumes that for each input vector X, there is a target vector Y^T, which represents the desired output. Together, they form a training pair. Typically, the network is trained on a set of such training pairs (training dataset). During training, the input vector X is fed into the network, the network's output Y is computed and compared to the corresponding target vector Y^T. The difference $D \sim Y^T - Y$ is then fed back into the network via a feedback mechanism, and the weights W are adjusted according to an algorithm that aims to minimize the error ε. The process of reading input vectors from the training set and adjusting weights continues until the total error over the entire training set reaches a predefined low level.

1.2.2.2 Unsupervised Learning

Despite numerous practical achievements, supervised learning has been criticized for its biological implausibility. It is difficult to imagine a learning mechanism in the brain that compares desired and actual output values and performs corrections through feedback. If such a mechanism were assumed in the brain, then where would the desired outputs originate from?

Unsupervised learning is a much more plausible model of learning in biological systems. Developed by Kohonen and many others, it does not require a target output vector and therefore does not rely on comparisons with ideal responses. The training dataset consists solely of input vectors. The learning algorithm adjusts the network's weights so that consistent output vectors are produced; that is, sufficiently similar input vectors produce the same outputs.

1.2.2.3 Genetic Algorithm

A genetic algorithm is a method that models the natural evolution of problem-solving methods, primarily for optimization tasks. Genetic algorithms are search procedures based on the mechanisms of natural selection and inheritance. They employ the evolutionary principle of survival of the fittest individuals. Genetic algorithms differ from traditional optimization methods by several fundamental elements. In particular, genetic algorithms:

– process not the parameter values of the problem itself, but their encoded form;

– perform the search for a solution starting not from a single point, but from a population of points;

– use only the objective function, without relying on its derivatives or other additional information;

– apply probabilistic rather than deterministic selection rules.

The four listed properties, which can also be formulated as parameter encoding, population-based operations, minimal use of problem-specific information, and operation randomization, ultimately result in the robustness of genetic algorithms and their advantage over other widely used technologies.

1.2.3 Analysis of Neural Networks Suitable for Forecasting

Classical neural networks, whose working model is described in the first section, are ideally suited for solving classification problems. These are problems where for each individual input signal X there is a unique output Y. In other words, feedforward neural networks are well suited for solving functional mapping tasks. This statement can be expressed mathematically as follows:

$$Y(t) = W * X(t). \tag{1.3}$$

Time series prediction is not a typical classification problem. Problems of this type must take into account the events that occurred prior to the input signal. The absolute value of stock market indicators at a particular moment in time is not sufficient information for accurate forecasting of future values. It is important to consider the events that happened before the immediate event.

This statement can be expressed mathematically as follows:

$$Y(t) = W * X(t) + Y(t-1). \tag{1.4}$$

Thus, a more suitable neural network architecture for solving the forecasting task is a recurrent neural network. For recurrent networks, the input data is always a sequence of values, which makes them ideal for problems based on time series. Therefore, it is proposed to consider both feedforward neural networks and recurrent networks, compare them, and evaluate their advantages and disadvantages.

1.2.3.1 Feedforward Neural Networks

Feedforward Neural Networks are also called multilayer perceptrons. In such networks, the input signal propagates forward, layer by layer. A multilayer perceptron generally consists of the following components:

– a set of input nodes forming the input layer;

– one or more hidden layers of computational neurons;

– a single output layer of neurons.

The multilayer perceptron represents a generalization of Rosenblatt's single-layer perceptron. An example of a multilayer perceptron model is shown in Figure 1.5.

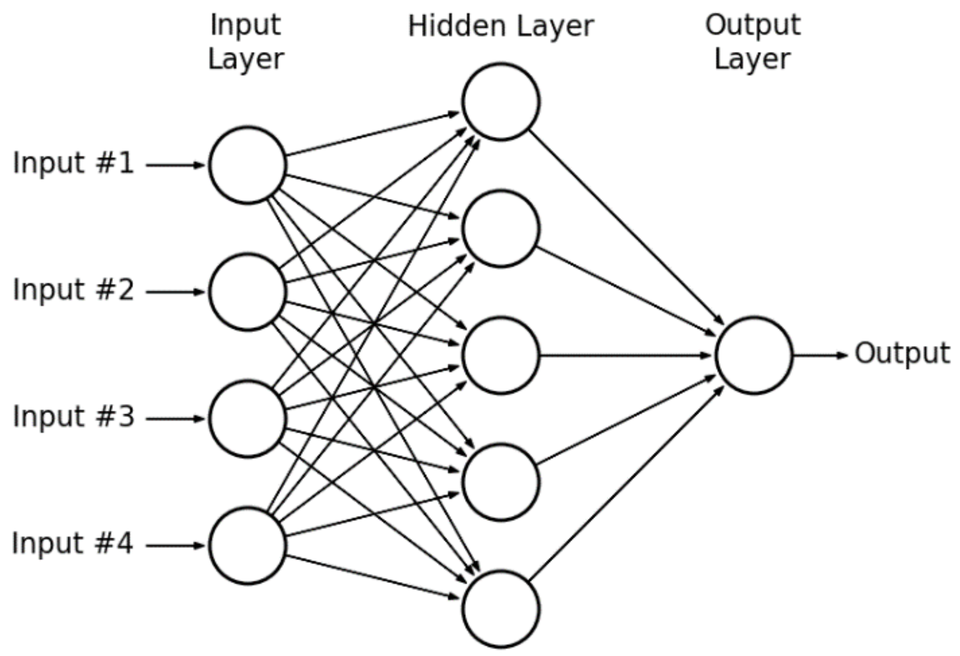


Figure 1.5 – Multilayer perceptron model

The number of input and output units in a multilayer perceptron is determined by the requirements of the task. Questions may arise regarding which input values to use and which to exclude. The issue of how many hidden layers to use and how many units should be in them is still quite unclear. As an initial approximation, one hidden layer can be used, with the number of units in it set equal to the average of the number of input and output units.

Multilayer perceptrons are successfully applied to solve various complex problems and have the following three distinguishing features.

The first of these indicates that each neuron in the network has a nonlinear activation function.

It is important to emphasize that such a nonlinear function must be smooth (i.e., differentiable everywhere), unlike the hard threshold function used in Rosenblatt’s perceptron. The most popular form of a function satisfying this requirement is the sigmoid function. An example of a sigmoid function is the logistic function, which is given by the following formula 1.5:

$$OUT = \frac{1}{1+\exp(-\alpha Y)}, \qquad (1.5)$$

where α – slope parameter of the sigmoid function.

By adjusting this parameter, we can construct functions with different steepness (see Figure 1.6).

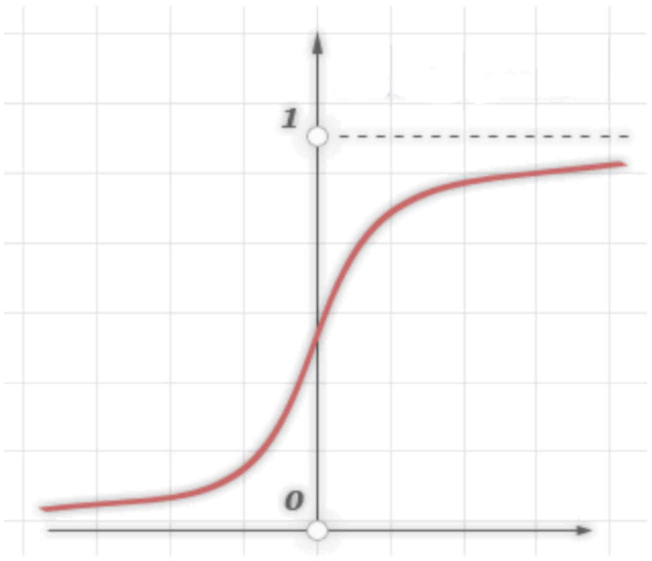


Figure 1.6 – Graph of the sigmoid activation function

The presence of nonlinearity plays a very important role, because otherwise the network's input-output mapping could be reduced to that of a simple single-layer perceptron.

The second feature refers to multiple hidden layers.

A multilayer perceptron contains one or more layers of hidden neurons, which are neither part of the input nor the output of the network. These neurons allow the network to learn to solve complex tasks by progressively extracting the most important features from the input pattern.

The third feature refers to high connectivity.

A multilayer perceptron has a high degree of connectivity, realized through synaptic connections. Changing the connectivity level of the network requires modifying many synaptic connections or their weight coefficients.

The combination of all these properties, along with the ability to learn from experience, provides the computational power of the multilayer perceptron. However, these same qualities also contribute to the incomplete understanding of the behavior of such networks: the distributed form of nonlinearity and the high connectivity of the network significantly complicate the theoretical analysis of multilayer perceptrons.

1.2.3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a class of artificial neural networks in which connections between nodes form a directed graph along a temporal sequence. This creates an internal state of the network that allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, recurrent neural networks can use their internal memory to process arbitrary sequences of inputs.

These neural networks have the following characteristics:

– all biological neural networks are recurrent;

– several training algorithms are known, but none is recognized as superior;

– theoretical and practical challenges hinder their straightforward implementation in real-world applications.

The presence of feedback in recurrent neural networks contributes to their more effective performance. One of the main drawbacks of such networks is that recurrent neural networks are difficult to train.

Traditional techniques used for training classical feedforward neural networks do not lead to the desired results. However, some subtypes of recurrent neural networks are more trainable, allowing them to form an adequate solution for tasks based on time series data.

Examples of such recurrent neural networks include the Elman network and the Jordan network, which extend the multilayer perceptron by introducing feedback connections.

The most popular training algorithm is backpropagation of error, which consists of interconnected processes. In the forward process, the input signal passes through the network, generating a certain output. In the backward process, the error (the difference between the desired and obtained output) is propagated from the output layers back to the input layers, simultaneously modifying the neuron connections so that during the next pass of information through the network, the error at the output layer is reduced. This constitutes the learning of the network.

The success of neural network predictions depends on the type of input information and which characteristics of the output data are significant for the forecasting task. Among different neural network configurations, a recurrent network with feedback from the hidden layer to the input layer was chosen (see Figure 1.7).

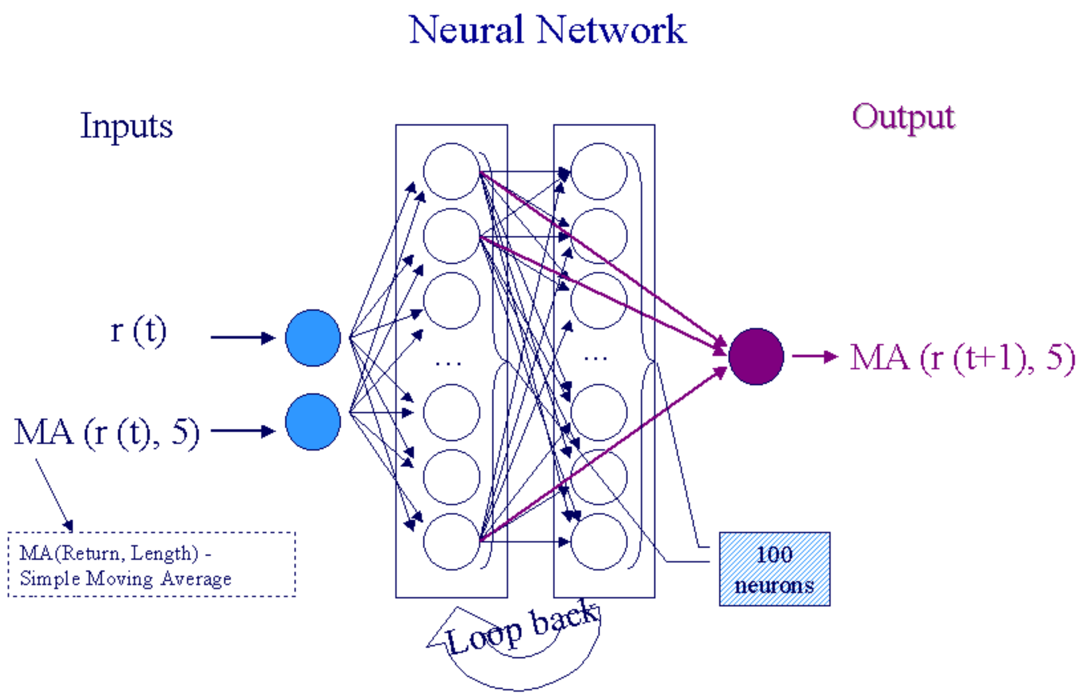


Figure 1.7– The Elman Neural Network Archirecture

The Elman network, like the Jordan network, is derived from the multilayer perceptron by introducing feedback connections. However, in the Elman network, signals to the input layer come not from the network’s outputs but from the outputs of the hidden layer neurons. Since the goal in this work is to obtain a single output signal predicting the stock price, the Elman network is considered because it generally provides higher accuracy in this case.

This type of network, trained by the backpropagation algorithm, has been successfully used for financial market forecasting because recurrent networks learn patterns in sequences of values, which is essential for working with time series data. A drawback of such networks is their long training time. Feedforward networks trained by this method produce the same output pattern for each identical input pattern. Recurrent networks, however, can produce different outputs for the same input pattern depending on the preceding pattern. Therefore, the order of training examples is critical. In other words, recurrent networks can be trained similarly to standard backpropagation networks, but training examples must be ordered and cannot be presented to the network randomly.

An important difference from feedforward networks is the presence, in the Elman-Jordan network, of an additional block that stores information about previous inputs. This block can be interpreted as the network’s long-term memory.

The hidden layer neurons of the Elman-Jordan network use the logistic activation function, while the output layer neurons have a linear activation function. This combination allows the approximation of any function with a finite number of discontinuities to a given accuracy. A symmetric logistic activation function can also be used. This does not affect the predictive capability of the network but leads to faster convergence of the training algorithm for this type of time series.

The most challenging aspect of using this type of neural network is selecting the stopping point for training. If the network is trained for too short a time, it will not learn the training sample adequately. Conversely, if the network is trained for too long, it will memorize the training examples, including noise, with arbitrarily high accuracy but will fail to generalize to new data (i.e., it will perform poorly on data outside the training set).

To overcome these difficulties, a calibration procedure can be employed to optimize the network by applying it to an independent test set during training. Calibration helps to find the network optimum on the test set, which indicates the network's ability to generalize — that is, to achieve good results on new data.

This is accomplished by calculating the mean squared error (MSE) between the actual and predicted outputs. The MSE is commonly used as a standard statistical measure of filtering quality.

1.3 Research Problem Statement

After conducting a theoretical assessment of the potential capabilities of neural networks, the goal was set to examine and compare several of the most suitable types of neural networks in the context of financial forecasting and to draw conclusions about the most effective among them. Based on these conclusions, a customized neural network specification should be developed in practice to calculate stock profitability on the stock exchange.

The work process will involve collecting data in the form of time series on stock prices of various companies, preprocessing input data, designing the most efficient neural network architecture, forming input variables for the neural networks, performing the forecasting process, and simulating trading.

Specifically, the task can be broken down into the following steps:

– to familiarize oneself with existing types of neural network structures;

– to study different neural network training methods;

– to select a research object in the form of a technology company whose stocks are traded on the stock exchange;

– to analyze available data, perform preprocessing, and introduce necessary additional custom variables;

– to determine the required neural network structure to achieve the set goal;

– to train the neural network;

– to validate the quality of the obtained results and measure the accuracy of the artificial neural network.

As a specific example of a stock exchange, it was decided to consider one of the leading American stock exchanges, NASDAQ. Thus, this exchange will serve as the object of the study. The subject of the research is forecasting stock profitability in financial markets using artificial neural network models.

2 ANALYSIS OF EXISTING SYSTEMS

Time Series Analysis is a set of mathematical and statistical techniques designed to identify the structure of time series and to make forecasts. These include, in particular, regression analysis methods. Detecting the structure of a time series is necessary in order to build a mathematical model of the phenomenon that generates the analyzed time series. Forecasting future values of the time series is applied to enable effective decision-making.

The forecasting procedure represents a sequence of steps through which a forecasting model for a specific time series is determined.

At the first stage of classification, methods can be conditionally divided into two groups: intuitive and formalized (see Figure 2.1).

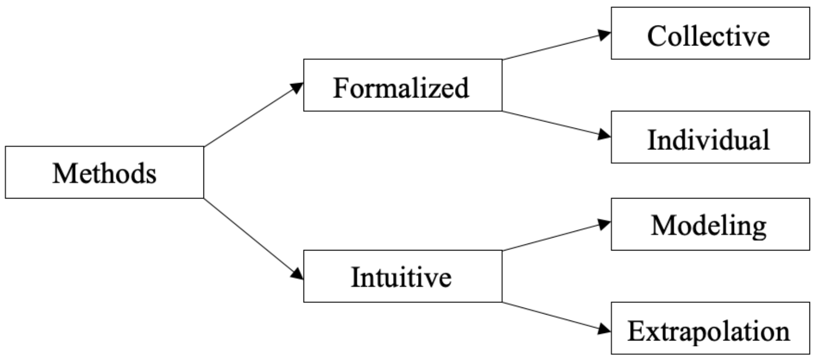


Figure 2.1 – Classification of forecasting methods

Intuitive forecasting methods are based on expert judgments and assessments. They are frequently used in marketing, economics, and politics, as the system whose behavior is to be predicted is either too complex to be described mathematically or too simple to require such a description.

Formalized methods, widely covered in scientific literature, are forecasting tools that allow the construction of mathematical models to predict the behavior of processes. The essence of these methods lies in determining a mathematical relationship that enables the calculation of future values of a process, thereby ensuring the implementation of the forecast.

This section of the work focuses on the analysis of formalized methods. The foundation of any forecasting method involves not only building the model but also evaluating the accuracy and quality of the resulting predictions. A general iterative approach to creating mathematical forecasting models typically involves the following main steps:

At the first stage, based on one's own or external prior experience, a general class of models is identified that will be used to forecast the time series over a specified horizon.

This general model class is usually quite broad. To adapt it to a specific time series, simplified identification methods are applied to select appropriate subclasses of models. These methods rely on qualitative evaluation of the time series characteristics.

Once a model subclass is chosen, its parameters are estimated (in the case of a parametric model), or its structure is refined (if it is a structural model). This stage typically involves iterative procedures, evaluating part or all of the time series under varying parameter values. This process is often the most resource-intensive, as it frequently requires consideration of all available historical time series data.

Next, a diagnostic check of the selected forecasting model is conducted. This involves selecting one or more segments of the time series—sufficient in duration—for test forecasting and accuracy assessment. These segments are often referred to as validation periods.

If the forecast accuracy at the diagnostic stage meets the requirements of the task, the model is considered ready for deployment. If the accuracy is insufficient, the entire process may be repeated from the beginning, starting with the selection of the general model class.

2.1 Forecasting Models for Financial Time Series

Based on formalized methods, two main groups of models can be distinguished: domain-specific models and time series models.

Domain-specific models are mathematical forecasting models developed on the basis of the laws and principles specific to a given subject area. For instance, weather forecasting relies on models that include equations of fluid dynamics and thermodynamics. Similarly, forecasting population dynamics involves models formulated using differential equations. These models take into account regularities and dependencies characteristic of a specific field of knowledge and therefore require a customized approach to construction.

Time series models, on the other hand, are mathematical tools aimed at identifying dependencies between future and past values within the same process. These dependencies can be used to compute forecasts. Time series models are considered universal, as their general structure remains consistent regardless of the nature of the data. Thus, they are often treated as generalized models.

When attempting to compile a general classification of models across various subject domains, one faces significant challenges due to the heterogeneity of the data. However, in the context of time series, models can be broadly classified into statistical and structural.

In statistical models, the relationship between future and past values is described by a specific equation. This approach encompasses such models as:

- regression models (linear regression, nonlinear regression);
- autoregressive models (ARIMAX, GARCH, ARDLM);
- exponential smoothing model;
- model based on maximum similarity sampling.

In structural models, the future value is determined based on the dependence on previous values, which is modeled using a defined structure and transition rules between its elements. Such models include:

- neural network models;

– models based on Markov chains;

– models based on classification and regression trees.

We will focus in more detail on the standard economic models mentioned above. The neural network model is not considered in this section, as its analysis was already carried out in the previous chapter.

2.1.1 Regression Forecasting Models

These models belong to the earliest methods in statistical analysis; however, their modern application is somewhat limited. At the same time, there are many tasks that require analyzing relationships between two or more variables. Regression analysis is actively used to solve such problems. Today, regression plays an important role in various fields, particularly in forecasting and management. The main goal of regression analysis is to identify the relationship between a dependent variable and a set of external factors known as regressors. The calculation of regression coefficients can be performed using the least squares method or the maximum likelihood method. Below, we will consider the main types of regression models.

2.1.1.1 Linear Regression Model

One of the simplest forms of regression modeling is linear regression. It is based on the assumption of the existence of a discrete external factor X(t) that influences the studied process Z(t), with a linear relationship between this process and the external factor. Forecasting within the framework of linear regression is modeled using the following equation:

$$Z(t) = \alpha_0 + \alpha_1 X(t) + \varepsilon_t, \quad (2.1)$$

where α_0 and α_1 – regression coefficients; ε_t – model error.

To calculate the predicted values Z(t) at a given time t, it is necessary to have the values of X(t) for that same moment. However, in practice, ensuring this condition is quite challenging and rarely achievable.

2.1.1.2 Multiple Regression Model

In practive, the process Z (t) is influenced by a number of discrete external factors $X_1(T)$, ..., $X_s(T)$. In this case, the forecasting model takes the form:

$$Z(t) = \alpha_0 + \alpha_1 X_1(T) + \alpha_2 X_2(T) + \cdots + \alpha_s X_s(T) + \varepsilon_t, \qquad (2.2)$$

where α_0 and α_1 – regression coefficients; ε_t – model error.

A drawback of this model is that to calculate the future value of the process Z (t), it is necessary to know the future values of all factors $X_1(T)$, ..., X_s (T), which is practically almost impossible.

2.1.1.3 Nonlinear Regression Model

The basis of the nonlinear regression model is the assumption of the existence of a given function that describes the relationship between the dependent variable and the independent variables:

$$Z(t) = F(X(t), A), \qquad (2.3)$$

where Z (t) – output process;

X (t) – external factor on which the process Z (t) depends;

A – function's parameter set that needs to be determined during the model building.

For example, one might assume that $Z(t) = \alpha_1\cos(X(t)) + \alpha_0$. In this case, to build a model it is sufficient to determine the parameters $A = \{\alpha_0, \alpha_1\}$, where α_0

and α_1 are regression coefficients. However, in real-world conditions, processes for which the exact functional form of the dependence betweenZ(t) and the external factor X(t) is known in advance are extremely rare. Therefore, nonlinear regression models are generally used infrequently in practical applications.

2.1.2 Exponential Smoothing Models

Developed in the mid-20th century, exponential smoothing models remain widely used today due to their simplicity and transparency.

The Exponential Smoothing (ES) method is primarily applied for the analysis and forecasting of financial and economic processes. The core of this method lies in the continuous updating of forecasts in response to newly received actual data. A key characteristic of the exponential smoothing method is the use of weighting coefficients that decrease exponentially as the data become older. As a result, more recent observations have a significantly greater influence on the formation of forecasted values compared to observations from the past.

The mathematical expression describing the function of the exponential smoothing model takes the following form:

$$Z(t) = S(t) + \varepsilon_t, \tag{2.4}$$

$$S(t) = \alpha * Z(t-1) + (1-\alpha) * S(t-1) + \varepsilon_t, \tag{2.5}$$

where α – smoothing coefficient, $0 < \alpha < 1$; the initial condition is defined as $S(1) = Z(0)$.

In the presented model, each subsequent smoothed value S(t) is calculated as a weighted average between the current value of the time series Z(t) and the previous smoothed value S(t – 1).

2.1.2.1 Holt's Model or Double Exponential Smoothing

This model is used for modeling processes that exhibit a trend. In such cases, the model structure should take into account two key components: level and trend.

Smoothing of the level and trend is performed separately.

$$S(t) = \alpha * Z(t-1) + (1-\alpha) * (S(t-1) + B(t-1)), \tag{2.6}$$

where α – smoothing coefficient for the level; $B(t)$ – linear trend function.

$$B(t) = \gamma * (S(t-1) + S(t-2)) + (1+\gamma) * B(t-1), \tag{2.7}$$

where γ – smoothing coefficient or the trend.

2.1.2.2 Holt-Wilters Model or Triple Exponential Smoothing

This method is similar to Holt's method and is applied to processes that exhibit a seasonal component in their trend:

$$Z(t) = (R(t) + G(t)) * S(t), \tag{2.8}$$

where $R(t)$ – smoothed level excluding the seasonal component;

$G(t)$ – smoothed trend;

$S(t)$ – seasonal component.

2.2 Comparison of Forecasting Models

The main advantages and disadvantages of different models are summarized in Table 2.1.

Table 2.1 – Comparison of Forecasting Models

Model and Method	Advantages	Disadvantages
Regression models and methods	Verification, flexibility, and transparency of modeling; ensuring consistency of approaches to analysis and design.	Defining the functional dependency is a complex task, caused by several factors, including the significant labor intensity of calculating the coefficients of this dependency. There are also certain limitations in the ability to model nonlinear processes, especially in the context of applying nonlinear regression.
Autoregressive models and methods	Simplicity and transparency of modeling; uniformity of analysis and design; numerous application examples.	High labor intensity and significant resource consumption for model identification; limited capability to model nonlinear processes; insufficient level of adaptability.
Exponential smoothing models and methods	Simplicity of modeling; uniformity of analysis and design.	Insufficient flexibility; narrow applicability of models.
Neural network models and methods	Nonlinearity of models; high adaptability; scalability; uniformity of analysis and design; numerous application examples.	Lack of transparency, difficulties in selecting an appropriate architecture, strict requirements for the training dataset, complexity in choosing the optimal training algorithm, as well as high resource consumption during the training process.

The nonlinearity of neural networks allows capturing nonlinear relationships between future and actual values of processes. Other important advantages include scalability — the parallel structure of artificial neural networks accelerates computations, which is especially relevant at an industrial scale when processing terabytes of data is required.

2.3 Choice of Neural Network Type

The first step in working with artificial neural networks is to determine the type of network architecture (the design of nodes and connections between them). Simon Haykin identifies three fundamental classes of neural network architectures: single-layer and multi-layer feedforward networks, as well as recurrent networks [5].

For the given task, which involves analyzing and working with stock market data as time series, it is proposed to consider recurrent neural networks, which are derived from the multi-layer perceptron by introducing feedback connections. Unlike classical neural networks, their distinctive feature is the presence of cycles—feedback loops that allow information to cyclically flow from the output back to the input layer.

In the context of forecasting tasks, it is important to take into account previous market data (such as stock volumes and prices), and the presence of feedback connections precisely addresses this need. Therefore, it is proposed to examine recurrent neural networks in detail as the primary forecasting tool.

3 SYSTEM DESIGN

Based on the analysis of the networks described in the previous chapter, this section will focus on developing a hybrid model that accumulates the positive findings of previous works and integrates them into a unified system for stock price forecasting. The workflow involves collecting data in the form of time series on the stock prices of various companies, processing the input data, designing the most effective neural network architecture, forming input variables for the neural networks, conducting the forecasting process itself, and simulating trading.

3.1 Definition of Input Dataa

Forecasting future stock prices is based on their past values. The main explanatory variable is the stock price itself in the past, and the closer the price value is to the current time period, the greater its influence on the current price formation. Therefore, an essential component of the input data is a sequence of stock price values of the company.

In addition, based on the work of Anish and Babita Majhi, there are grounds to include trading volume values in the analysis. The studies described in that work indicate the presence of a correlation between changes in trading volume and stock price values. Also, as input data, it is proposed to consider a number of other parameters that will further characterize and describe the stocks from a financial perspective. The defined set of input data is as follows:

- date;
- trading volume;
- opening price;
- daily minimum price;
- daily maximum price;
- closing price.

The second main parameter, volume, like the price parameter, is formed based on the sequence of volume values from previous periods.

Forecasting the variable's future values based on past values becomes more accurate when explanatory dependent variables are added. As additional variables, 20-period moving averages will be used, formed separately for forecasting trading volume and stock price.

3.2 Data Preprocessing

Before feeding the input data into the neural network for training, it should be pre-normalized. For more accurate performance of the artificial neural network, it is recommended to normalize the data using Principal Component Analysis (PCA) [7], as a result of which all input data values will be scaled between 0 and 1.

3.3 Forecasting Algorithm

The forecasting algorithm will be as follows: for period t, the stock closing price will be predicted based on data from period t–1. Thus, each predicted value will serve as input for forecasting the subsequent value.

Following this logic, it makes sense to minimize the number of independent variables, considering that each independent variable must be generated separately. Among the presented input data, it is evident that the opening price, minimum, and maximum values are less relevant for predicting the closing price.

As a result, the closing price at period t will be predicted based on the values of two independent variables from period t–1: trading volume and closing price.

3.4 Additional Explanatory Variables

Leaving the two independent variables as the fundamental features, it makes sense to introduce additional dependent variables that link the closing price values over several previous periods. This function is well performed by technical analysis indicators, such as moving averages. According to the work of Nikfarjam, Emadzadeh, & Muthaiyah (2010) [15], technical indicators are more informative than raw price data.

The list of technical indicators added to the analysis includes:

- EMA (Exponential moving average);
- ZLEMA (Zero lag exponential moving average);
- SMA (Simple moving average);
- EVWMA (Elastic Volume Weighted Moving Average);

A window length of 20 was chosen for each moving average, meaning that each will consider 20 previous price values.

In addition to the moving average indicators, the closing prices of the five previous periods will also be included in the analysis, as they have the greatest influence on the current price. A similar process will be applied to the volume variable.

3.5 Accuracy of the Results

As the accuracy evaluation metric, the Root Mean Square Error (RMSE) will be used:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})}{n}}. \quad (3.1)$$

At this stage, it is necessary to determine the magnitude of the prediction error on the test dataset. After setting a quality threshold as a fixed value, exceeding or falling short of this threshold will indicate the success or failure of the experiment. The specific value of the threshold is conditional and should be adjusted according to the requirements of each individual experiment. For the experiment conducted in this work, the quality threshold is set close to zero.

If the prediction error exceeds the established threshold, indicating insufficient accuracy of the neural network, it will be necessary to return to the previous stages of neural network architecture design to reconsider alternative options.

Conversely, if the error value is minimal (does not exceed the set threshold), there are grounds to proceed with the direct implementation of trading on the stock market.

3.6 Forecast Modeling

3.6.1 Volume Forecasting

First of all, it is necessary to build a neural network to predict trading volume. The decision was made to predict volume first because the dependency of price on volume is more significant than vice versa. The subsequent results of this neural network will be used as input data for the neural network predicting the stock price.

For volume prediction, the first three moving average indicators will be considered (since the fourth indicator directly depends on the volume value), as well as the previous volume values for the last five periods.

The formula for the neural network predicting volume is as follows:

$$\text{volume} \sim \text{ema.20} + \text{sma.20} + \text{zlema.20} + x_1 + x_2 + x_3 + x_4 + x_5, \qquad (3.2)$$

where the first three variables are moving averages, and the next five are volume values from previous periods (x_1 – at period t–1, x_2 – at period t–2, and so on).

The dataset will be divided into training and testing subsets, and by using the training data to train the neural network — the structure of which is described in the previous section — the following neural network will be obtained (see Figure 3.1).

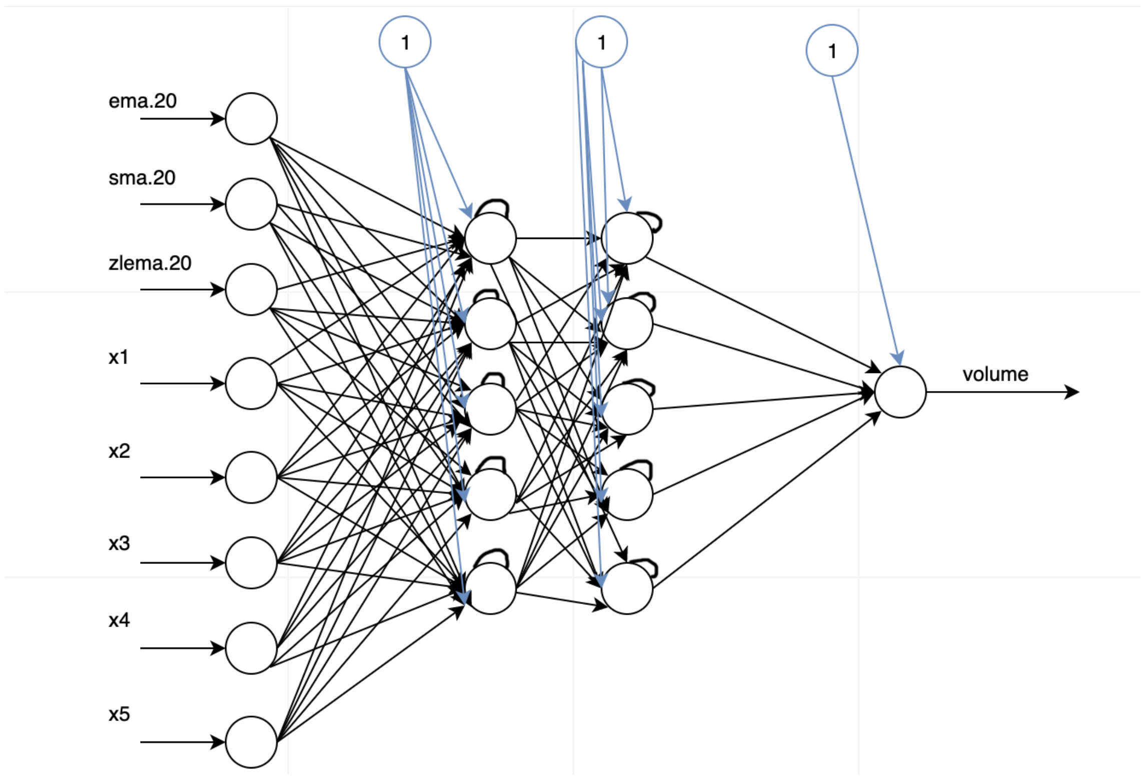


Figure 3.1 – Scheme of the RNN for volume forecasting

In Figure 3.1, it can be seen that the resulting neural network has three layers, with the most important being the hidden second layer, which contains 5 neurons.

3.6.2 Price Forecasting

Having predicted the trading volume in the previous step, we now have all the data necessary to form a similar neural network for forecasting the stock closing price. The formula for stock price forecasting in this case is as follows:

$$\text{close} \sim \text{volume} + \text{ema.20} + \text{sma.20} + \text{evwma.20} + \text{zlema.20} + x_1 + x_2 + x_3 + x_4 + x_5, \quad (3.2)$$

where volume is the trading volume, the other 4 variables are the moving averages calculated over the previous 20 price values, and separately the closing prices of the stock for the previous five periods.

Using a similar algorithm as in the case of building the network for volume forecasting, we will construct the feedforward neural network scheme for price forecasting (Figure 3.2).

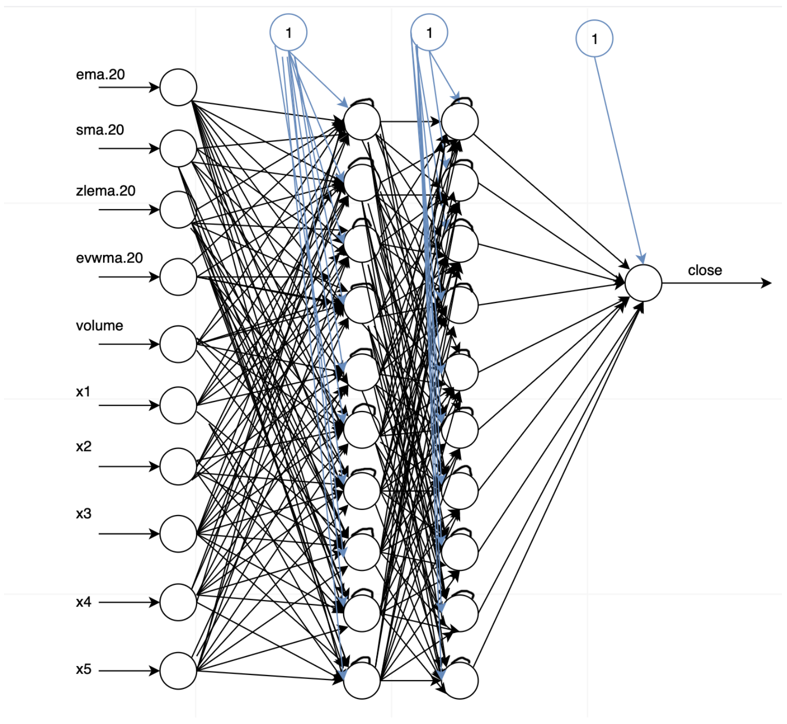


Figure 3.2 – Scheme of the RNN for price forecasting

In this case, considering the larger amount of input data, to achieve more accurate results it is necessary to introduce a greater number of layers and a higher number of neurons in each of them.

3.7 Trading Simulation

Having obtained the forecasted stock prices of the company for several future periods from the neural network output, it is necessary to analyze the results and generate signals that will ensure the most profitable trading with minimal risks.

In practice, traders use numerous technical indicators as a means to form correct market signals. It is proposed to create a similar algorithm capable of indicating the optimal signal at the intersection points of the used indicators. The most basic trading signals are generated using moving average indicators.

As an evaluation of the usefulness of the information obtained from the constructed neural network, it is proposed to conduct a test trading on the stock market with various options to generalize the trading results. An initial capital of $100,000 can be accepted and used for trading on the market with the help of the developed tool.

4 SOFTWARE IMPLEMENTATION OF THE SYSTEM

This section will discuss an example implementation of a neural network trained on a time series dataset from the stock market. As a practical example, the NASDAQ stock market will be considered.

Within this practical implementation, the operation of neural networks will be analyzed using the example of one company — 2U (https://2u.com/) — an educational technology company offering online learning.

The raw data, downloaded from nasdaq.com, mainly contains information about daily closing stock prices and trading volumes. Since these two variables are mutually independent, each will be forecasted separately — a separate neural network will be built for each. The first will predict trading volume several dozen days ahead based on several dozen recent volume values. The predicted volume values obtained will be used as input data for the second neural network. The second neural network, based on the predicted volume data and the available recent price data, will provide a forecast of stock prices for the coming days.

4.1 Choice of Programming Language, Development Environment, Frameworks, and Libraries

The programming language chosen for implementation was JavaScript, which is very popular today. Initially, JavaScript was used to add interactivity to web pages, but over the years it has gained many new features and frameworks that allow it to work with all types of applications — browser-based applications and websites, servers, mobile applications, and desktop applications.

For development, the IntelliJ WebStorm environment was selected.

WebStorm is an integrated development environment (IDE) for working with JavaScript, CSS, and HTML, created by JetBrains based on the IntelliJ IDEA platform. This IDE offers features such as code autocomplete, real-time code analysis and checking, navigation between code elements, efficient refactoring, program debugging, and integration with version control systems.

One of WebStorm's key advantages is its ability to handle projects of varying complexity. This includes refactoring JavaScript code located in multiple files and folders. The tool also ensures correctness of refactoring in cases of nested code. For example, when an HTML document contains a JavaScript script with embedded HTML code, which in turn contains another block of JavaScript. In such complex structures, WebStorm guarantees correct code handling and optimization.

Using the capabilities of HTML/XHTML and XML, WebStorm provides tools for automatic code completion, including styles, links, attributes, and other markup elements. When working with CSS, it offers autocomplete for classes, IDs, keywords, and numerical HTML values. The program also automatically solves tasks related to selecting formats, properties, classes, file references, and other CSS attributes. Thanks to integration with the Zen Coding tool, WebStorm significantly simplifies the process of creating HTML code by instantly reflecting tag changes on the web page.

Additionally, the editor provides intelligent autocomplete for JavaScript code, including keywords, labels, variables, parameters, and DOM functions, taking into account the specifics of popular browsers. Implemented refactoring features allow effective restructuring of JavaScript code and related files. WebStorm offers powerful JavaScript debugging tools, including breakpoint management in HTML and JavaScript, their configuration, and real-time syntax checking.

The product supports popular development platforms such as JQuery, YUI, Prototype, DoJo, MooTools, Qooxdoo, and Bindows. To ensure high code quality, it includes integrated tools for checking tags, code sequence, and spelling errors. Moreover, the editor allows editing files and automatically synchronizing them on demand during remote work or upon saving.

Version control functionality support allows tracking all changes and saving previous file versions. Thanks to the change history in WebStorm, users can restore code expressions, individual blocks, or even entire files, significantly easing project management.

As JavaScript frameworks, React.js and TensorFlow.js were chosen.

4.1.1 Working with React.js

React is an open-source JavaScript library designed for building user interfaces. It is developed and maintained by companies such as Facebook and Instagram, as well as communities of independent programmers and corporations.

The primary use case of React is the development of single-page web applications and mobile apps. The key principles on which the library is built focus on ensuring high performance, ease of use, and scalability. When creating user interfaces, React is often integrated with additional libraries like Redux, which extend its functionality and simplify application state management (see Figure 4.1).

The core principle of the framework is hierarchical component-based architecture, where components have their own behavior, lifecycle, and space for storing data.

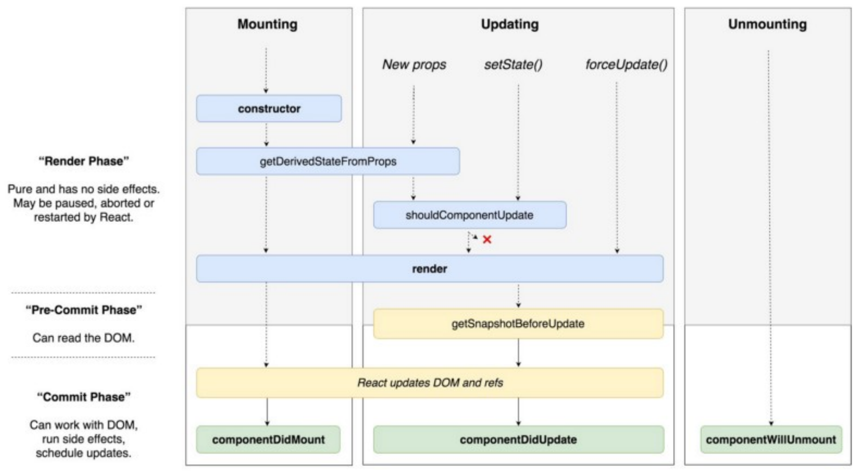


Figure 4.1 – React component lifecycle methods

4.1.2 Working with Tensorflow.js

TensorFlow.js enables running machine learning programs entirely on the client side, a concept known as In-Browser ML.

Using machine learning directly in the browser provides simplicity and convenience for users: there is no need to install additional libraries or drivers. Simply open the web page, and your application is ready to work.

This technology supports hardware acceleration via GPU by leveraging TensorFlow.js's functional use of WebGL. This allows automatic optimization of your code execution if a GPU is available, improving performance seamlessly.

Moreover, mobile users can have a unique experience because the web page can interact with their device sensors, such as gyroscopes or accelerometers.

Importantly, all data processing happens exclusively on the client side. This approach not only reduces latency during operations but also ensures a high level of privacy, making TensorFlow.js an ideal choice for tasks that require fast output or data confidentiality.

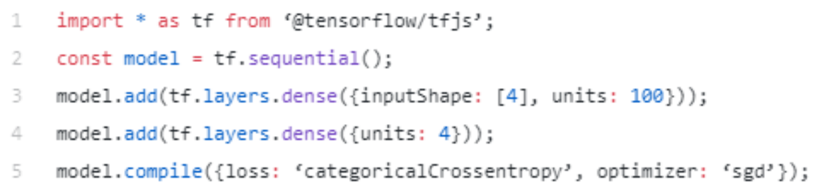
There are several approaches to working with machine learning models in TensorFlow.js:

– import a pre-trained model. If you have a model trained offline using TensorFlow or Keras, it can be converted into TensorFlow.js format and loaded for further use directly in the browser;

– fine-tune the imported model. For example, as demonstrated in the Pac-Man game demo, transfer learning can be applied to adapt an existing offline-trained model. This allows using a small amount of data collected right in the browser via methods like Image Retraining. Such an approach enables precise model tuning within a relatively short time and with minimal resources;

– create a model directly in the browser. Using TensorFlow.js, you can create, train, and run models directly in the web environment with JavaScript and a high-level API. Those familiar with Keras will quickly adapt to this tool since the API has a similar structure and logic.

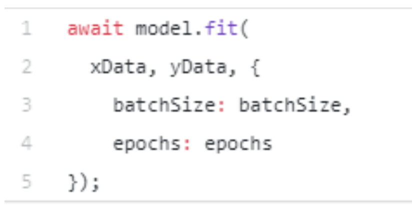
As a quick example, below is a code snippet defining a neural network for flower classification. The model is built using a stack of layers (see Figure 4.2).



```
import * as tf from ‘@tensorflow/tfjs’;
const model = tf.sequential();
model.add(tf.layers.dense({inputShape: [4], units: 100}));
model.add(tf.layers.dense({units: 4}));
model.compile({loss: ‘categoricalCrossentropy’, optimizer: ‘sgd’});
```

Figure 4.2 – Model definition

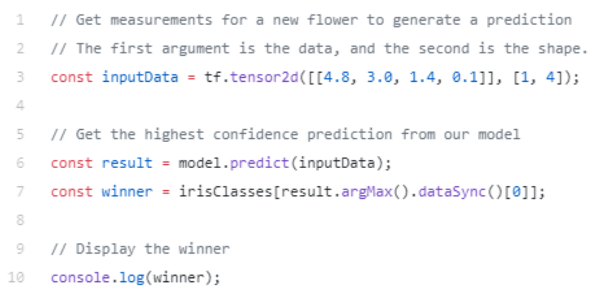
The API used in this application supports all Keras layers (including Dense, CNN, LSTM, and so on). Then, the model can be trained using the same API, which is compatible with Keras, by calling the corresponding method (see Figure 4.3).



```
await model.fit(
  xData, yData, {
    batchSize: batchSize,
    epochs: epochs
});
```

Figure 4.3 – Model training

After this, the model will be ready for use (Figure 4.4).



```
// Get measurements for a new flower to generate a prediction
// The first argument is the data, and the second is the shape.
const inputData = tf.tensor2d([[4.8, 3.0, 1.4, 0.1]], [1, 4]);

// Get the highest confidence prediction from our model
const result = model.predict(inputData);
const winner = irisClasses[result.argMax().dataSync()[0]];

// Display the winner
console.log(winner);
```

Figure 4.4 – Model ready for forecasting

TensorFlow.js also includes a low-level API (formerly deeplearn.js) and supports Eager execution (Figure 4.5).

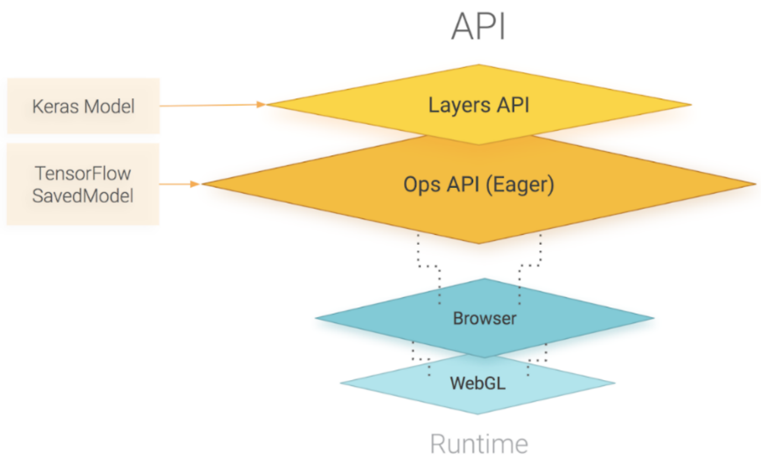


Figure 4.5 – Tensorflow API

4.2 Application Development

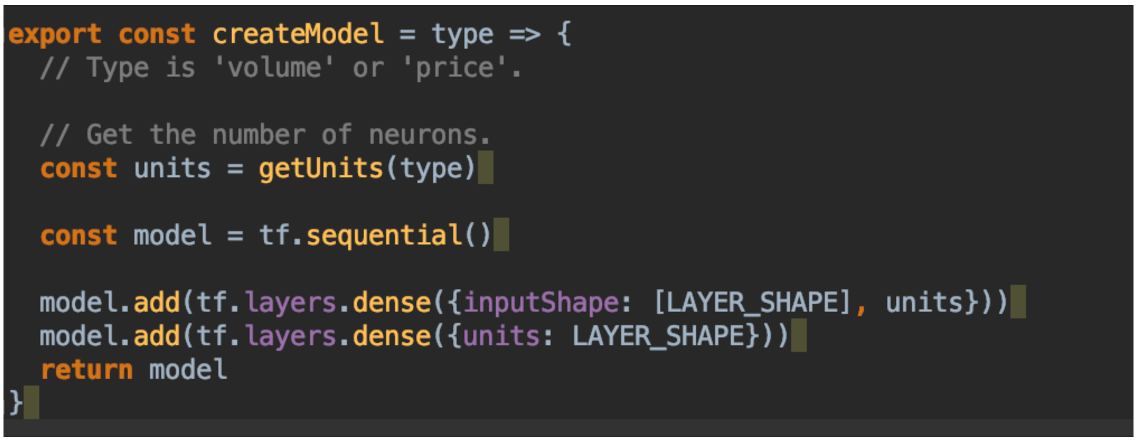
4.2.1 Model Creation

Within the concepts of TensorFlow.js, deploying a neural network typically begins with defining the training model and creating the corresponding model object. A model is a set of layers that can be either simple or complex. It can be used for training or making predictions. TensorFlow.js supports two main types of models: regular and sequential.

The regular model is based on a graph structure and allows creating various configurations where the layers of the neural network can be connected arbitrarily. This provides greater control over the customization of the training process and the computation of results. Such models are usually applied in situations where specific algorithms or prediction mechanisms need to be implemented.

The sequential model has a more linear structure, where each new layer is added on top of the previous one. The input of each layer is automatically linked to the output of the preceding one. For training and predicting with sequential models in TensorFlow.js, model object methods such as model.fit(…) for training and model.predict(…) for prediction are used. The sequential model is especially convenient for tasks requiring forecasting, such as time series prediction like financial market fluctuations. In such cases, its structure and principles are easier to implement while still providing highly accurate predictions, which can be more efficient compared to using complex custom training algorithms.

Figure 4.6 shows an empty sequential model, which serves as the foundation for further expanding its architecture. In this process, the method model.add(…) is used, allowing new layers to be added step-by-step to the model according to the defined structure and configuration.



```
export const createModel = type => {
  // Type is 'volume' or 'price'.

  // Get the number of neurons.
  const units = getUnits(type)

  const model = tf.sequential()

  model.add(tf.layers.dense({inputShape: [LAYER_SHAPE], units}))
  model.add(tf.layers.dense({units: LAYER_SHAPE}))
  return model
}
```

Figure 4.6 – Creating a neural network model

4.2.2 Working with Input Data

As mentioned before, the initial input data consists of:

– date;

– trading volume;

– opening price;

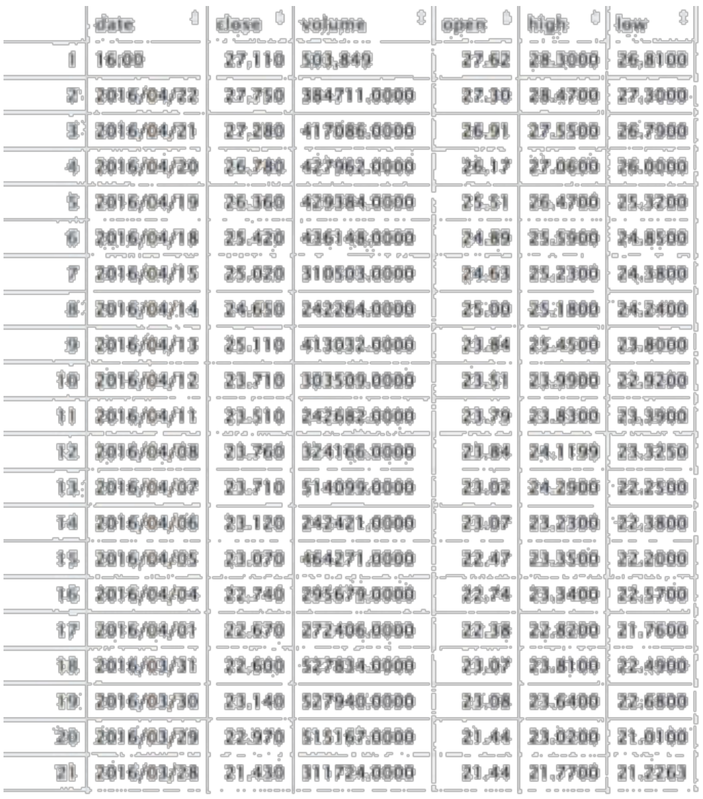
– daily minimum price;

– daily maximum price;

– closing price.

To demonstrate the complete practical forecasting process, at each logical step the software provided tables of input data and graphs showing changes in forecasted output values alongside changes in actual data, taking into account the time period of operation.

First, let's present the full range of initial input data in a single table (Figure 4.7).



	date	close	volume	open	high	low
1	16:00	27,110	503,849	27.62	28.3000	26,8100
2	2016/04/22	27,750	384711.0000	27.30	28.4700	27.3000
3	2016/04/21	27,280	417086.0000	26.91	27.5500	26.7900
4	2016/04/20	26.780	427962.0000	26.17	27.0600	26.0000
5	2016/04/19	26.360	429384.0000	25.51	26.4700	25.3200
6	2016/04/18	25.420	436148.0000	24.89	25.5900	24.8500
7	2016/04/15	25.020	310503.0000	24.63	25.2300	24.3800
8	2016/04/14	24.650	242264.0000	25.00	25.1800	24.2400
9	2016/04/13	25.110	413032.0000	23.84	25.4500	23.8000
10	2016/04/12	23.710	303509.0000	23.51	23.9900	22.9200
11	2016/04/11	23.510	242682.0000	23.79	23.8300	23.3900
12	2016/04/08	23.760	324166.0000	23.84	24.1199	23.3250
13	2016/04/07	23.710	514099.0000	23.02	24.2900	22.2500
14	2016/04/06	23.120	242421.0000	23.07	23.2300	22.3800
15	2016/04/05	23.070	464271.0000	22.47	23.3500	22.2000
16	2016/04/04	22.740	295679.0000	22.74	23.3400	22.5700
17	2016/04/01	22.670	272406.0000	22.38	22.8200	21.7600
18	2016/03/31	22.600	527834.0000	23.07	23.8100	22.4900
19	2016/03/30	23.140	527940.0000	23.08	23.6400	22.6800
20	2016/03/29	22.970	515167.0000	21.44	23.0200	21.0100
21	2016/03/28	21.430	311724.0000	21.44	21.7700	21.2263

Figure 4.7 – Initial form of input data

After obtaining the input data, the next stage involves selecting the most relevant data for forecasting and performing normalization. Additionally, four indicators of moving averages for volume and stock prices will be introduced and normalized. As discussed in the previous section, it was decided to pre-normalize the data using the Principal Component Analysis (PCA) method.

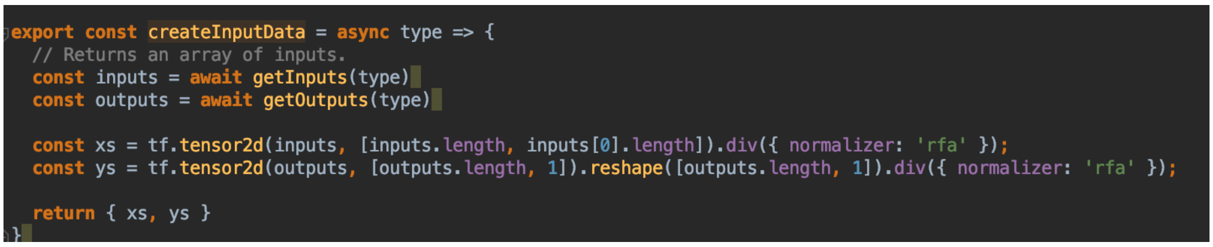
The data import in JavaScript was carried out by reading from an XLS file (a table containing data from NASDAQ).

TensorFlow represents input and output data using a special object called a tensor. Tensors are abstract objects used to store datasets passed to the input or output signals of a training model. In TensorFlow.js, tensors can store one-, two, three-, or four-dimensional arrays of data.

Moreover, tensors provide functionality for reshaping different data arrays by increasing or decreasing the number of dimensions. For example, data stored in a two-dimensional tensor can be transformed into one-dimensional data using tensor object methods, which will be discussed in the following section.

Normalization takes place simultaneously with tensor creation. TensorFlow supports the Principal Component Analysis method as one of the normalization techniques provided by the library, so there is no need to implement it manually.

The process of creating tensors and normalizing the input data are shown in Figure 4.8.



```
export const createInputData = async type => {
  // Returns an array of inputs.
  const inputs = await getInputs(type)
  const outputs = await getOutputs(type)

  const xs = tf.tensor2d(inputs, [inputs.length, inputs[0].length]).div({ normalizer: 'rfa' });
  const ys = tf.tensor2d(outputs, [outputs.length, 1]).reshape([outputs.length, 1]).div({ normalizer: 'rfa' });

  return { xs, ys }
}
```

Figure 4.8 – Creating tensors and normalizing the input data

4.2.3 Working with Recurrent Connections

The Recurrent Neural Network (RNN) is one of the key layers of the model being developed. To improve forecasting accuracy, an RNN that includes several Long Short-Term Memory (LSTM) units is used. According to the RNN architecture, the input data for this layer is presented as a three-dimensional tensor with the following structure: [number of samples, number of time steps, number of features].

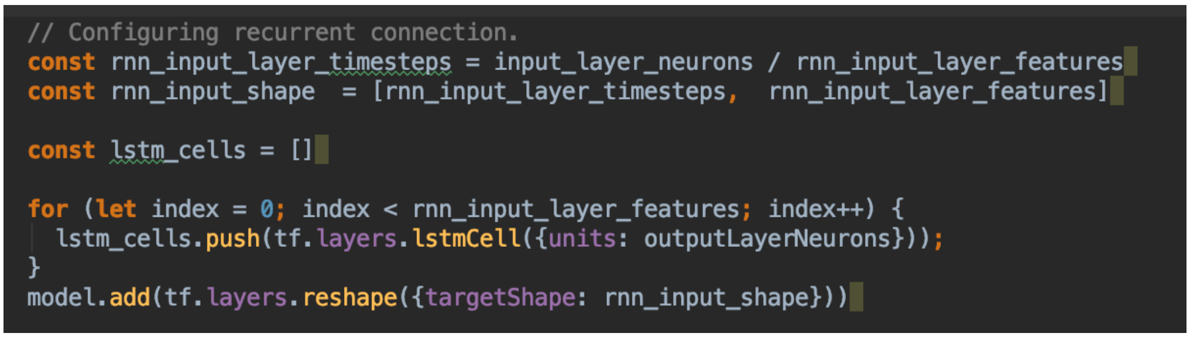
The first parameter of this structure defines the number of samples (data sets) passed from the output of the input dense layer to the respective inputs of the RNN in the next model layer.

The second parameter corresponds to the number of RNN time steps, which reflects the number of iterations the network uses during training.

The third parameter indicates the number of features (or values) present in each sample.

As mentioned earlier, the output of the input dense layer is a one-dimensional tensor of numerical values. To pass this data in a format suitable for the RNN input, it must be transformed into a three-dimensional tensor as described above.

To achieve this, a reshape layer is typically used. This layer does not perform any calculations by itself, but it changes the shape of the output data. This transformation can be implemented using the appropriate code (see Figure 4.9).



```
// Configuring recurrent connection.
const rnn_input_layer_timesteps = input_layer_neurons / rnn_input_layer_features
const rnn_input_shape  = [rnn_input_layer_timesteps,  rnn_input_layer_features]

const lstm_cells = []

for (let index = 0; index < rnn_input_layer_features; index++) {
  lstm_cells.push(tf.layers.lstmCell({units: outputLayerNeurons}));
}
model.add(tf.layers.reshape({targetShape: rnn_input_shape}))
```

Figure 4.9 – Creating recurrent connections

4.2.4 Training and Testing the RNN

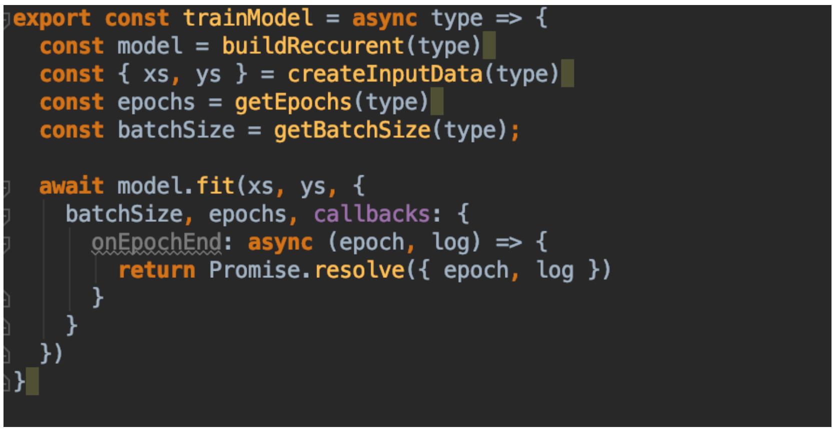
To train the model using a sample dataset, specific tensors must be passed as arguments to the model.fit(...) method. As shown in the sample code, this process is performed asynchronously.

The first argument is the batch size. The batchSize value determines the number of features (input data points) the model processes at one time. In experimental conditions, the batch size is usually equal to the number of features in each individual data example (i.e., the window size).

The second argument of the method specifies the number of epochs, which indicates how many full passes over the dataset the model should perform. An epoch is one complete cycle through the entire training dataset. Thus, this parameter defines how many times the model will iteratively process the same data to improve its prediction accuracy.

The third argument, onEpochEnd, allows defining a callback function that will be executed after the completion of each training epoch.

An example of using this function is shown in Figure 4.10.

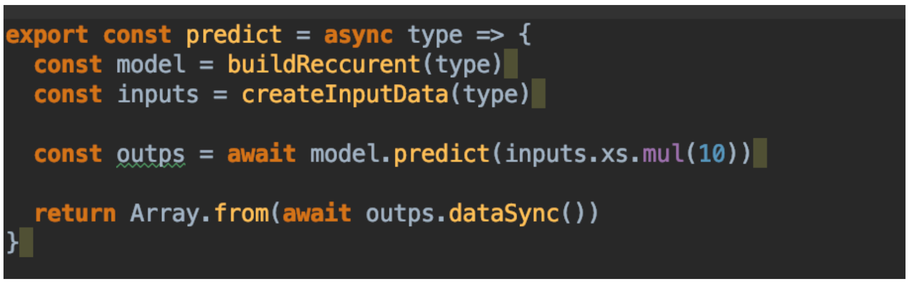


```
export const trainModel = async type => {
  const model = buildReccurent(type)
  const { xs, ys } = createInputData(type)
  const epochs = getEpochs(type)
  const batchSize = getBatchSize(type);

  await model.fit(xs, ys, {
    batchSize, epochs, callbacks: {
      onEpochEnd: async (epoch, log) => {
        return Promise.resolve({ epoch, log })
      }
    }
  })
}
```

Figure 4.10 – Code example for RNN training

Once the model has been trained on the generated dataset, it is ready to be used for prediction purposes. In particular, to implement forecasting, the following code should be used (see Figure 4.11).



```
export const predict = async type => {
  const model = buildReccurent(type)
  const inputs = createInputData(type)

  const outps = await model.predict(inputs.xs.mul(10))

  return Array.from(await outps.dataSync())
}
```

Figure 4.11 – Code example for RNN prediction

4.3 Program Results

A simple web application was created, which had the following interface (see Figure 4.12).

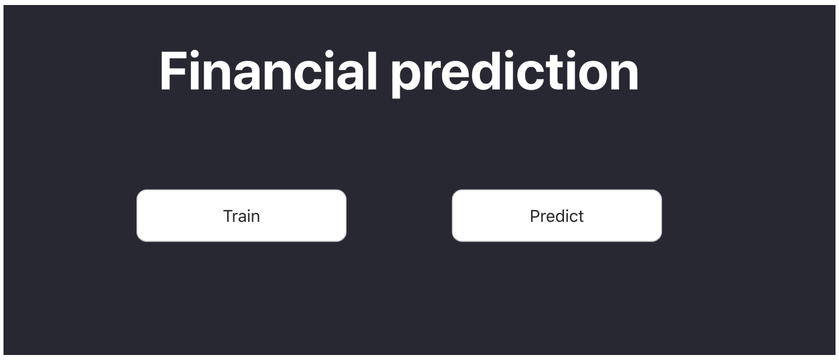


Figure 4.12 – Application menu

According to the button labels, each button triggered either the training or the forecasting process of the neural network.

To conveniently display the results, a React-oriented library called Recharts was chosen. It is user-friendly and capable of building graphs of varying complexity.

4.3.1 Volume Prediction

The forecasting results were visualized using charts and appeared as shown in Figures 4.13–4.14.

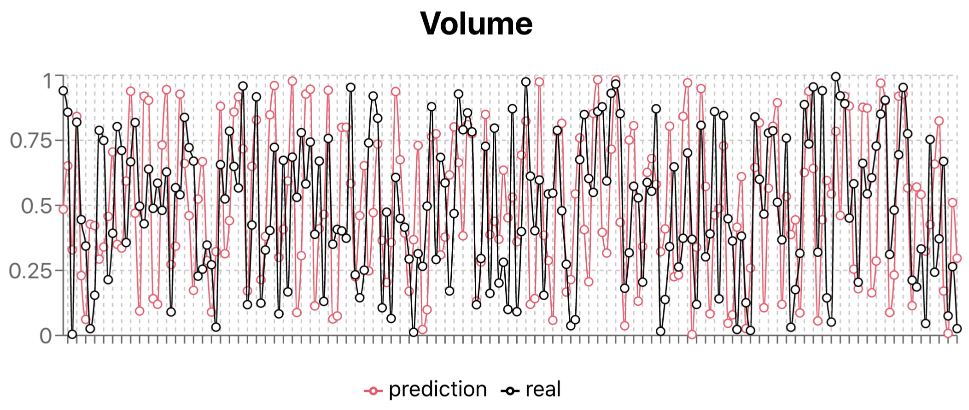


Figure 4.13 – Training the RNN for volume

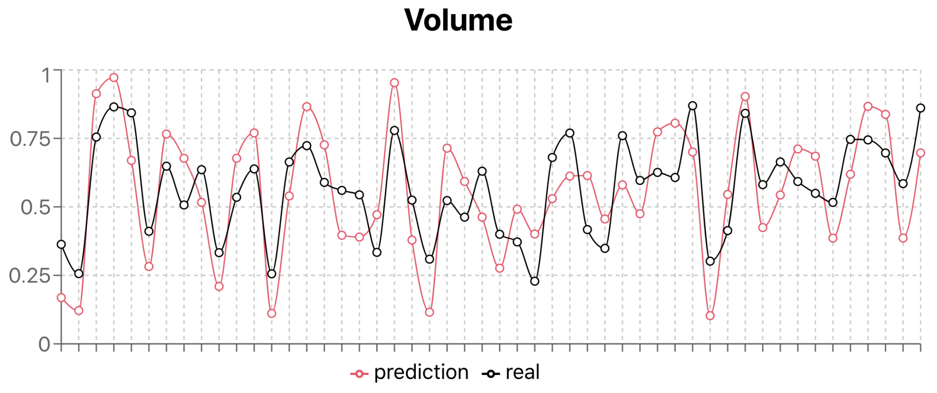


Figure 4.14 – Testing the RNN for volume

From this chart, one can clearly observe both the strengths and weaknesses of the resulting recurrent neural network. The constructed Elman network successfully captures the trend of trading volume but shows inaccuracies in predicting the exact absolute values.

4.3.2 Price Prediction

The predicted trading volume values obtained are fed as input data into the second Elman neural network, whose task is to generate the company's stock prices for future periods. The quality of the final neural network's training can be represented by the following graph (Figure 4.15).

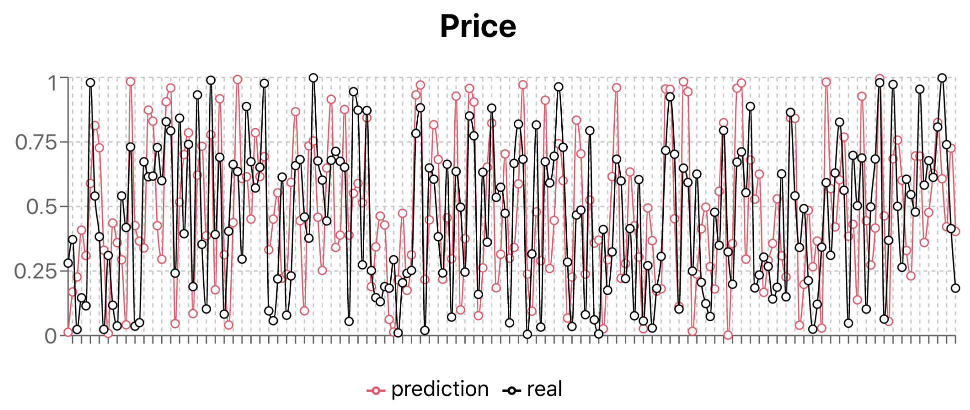


Figure 4.15 – Training the RNN for volume

After achieving minimal errors during the generation of the neural network formula (which can be found in the appendix), we obtained the forecast on the test dataset (Figure 4.16).

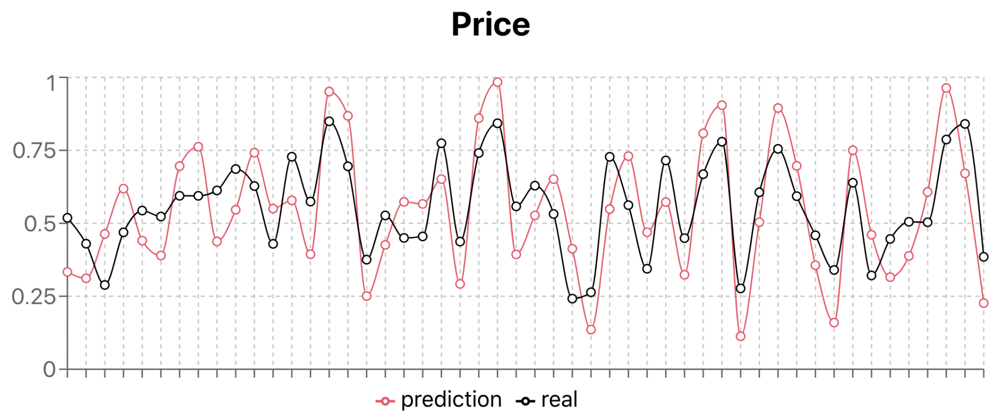


Figure 4.16 – Testing the RNN for price

As in the case of volume prediction, the modified Elman neural network demonstrates fairly accurate trend forecasting of stock price movements. However, the accuracy of the absolute predicted stock prices continues to be an area for improvement.

4.4 Error Estimates

The list of companies whose stock data were tested during the development of the neural networks (Table 4.1):

- 2U;
- Apple;
- Elite Data Services;
- TECO Energy;
- Ruby Tuesday;
- Follow;
- PowerShares International Corporate Bond Portfolio;
- St. Joe Company;
- Coca-Cola Company;
- QAD;
- KLA-Tencor Corporation;
- JunkieDog;
- Mid-America Apartment Communities;
- Quantum Energy;
- MEDIFAST.

Table 4.1 – Error metrics for each company

Company number	RMSE error
1	0.13533211
2	0.12323412
3	0.09712342
4	0.06452013
5	0.08569012
6	0.12345921
7	0.07512489
8	0.05934012

9	0.04634224
10	0.08523412
11	0.09823491
12	0.12342392
13	0.07823491
14	0.07123324
15	0.08932412

4.5 Technical Summary

Based on numerous stock forecasting tests conducted within this work, the developed recurrent neural networks are not able to accurately reproduce the absolute stock price values.

However, it was found that they are particularly successful in predicting the overall trend of price movement. Trading based on trends makes sense only in long-term perspectives.

The developed input data specifications and neural networks do not allow for effective trading on short time intervals. Technical analysis appears to be a more reliable signal for trading on short windows.

Nevertheless, neural networks, including the specifications developed in this work, can be used as one of the signals alongside many technical indicators.

Neural methods can be successfully applied to tasks such as crisis phenomena in capital markets, tax revenues, dynamics of derivative financial instruments prices, including stock index forecasting, portfolio diversification efficiency, credit risk assessment, or bankruptcy prediction of corporations and banks. This fact suggests reconsidering the structure of the developed neural networks for more accurate index forecasting.

CONCLUSIONS

During this qualification work, both theoretical and practical aspects of artificial neural networks were studied. Various ways of organizing such networks were examined, and the advantages and disadvantages of each were considered. Several technical analysis modifiers were also studied and applied to ensure data consistency within a window of 20 observations.

As a practical result, a group of interconnected neural networks was built, which are capable of demonstrating forecasting with high accuracy during their operation. This result can be considered positive, as it indicates the feasibility of applying this approach in real trading with relative success.

The work revealed that recurrent neural networks are much more effective when working with time series than feedforward networks. In particular, the effectiveness of the Elman network was demonstrated. As a continuation of research in this area, it is recommended to test the theory that such neural networks predict future stock price trends with higher accuracy, which is sufficient for planning profitability strategies for speculators' funds.

Moreover, as a direction for further development of the neural network model created in this work, it is possible to build a stock market trading simulation system that, based on real-time data, will perform operations for opening and closing market positions.

REFERENCES

1. Deep J. Machine learning for beginners: an introductory guide to learn and understand artificial intelligence, neural networks and machine learning. Independently Published, 2019.

2. Aggarwal C. C. Neural networks and deep learning: a textbook. Springer, 2019. 520 p.

3. Eknath, Prof. Upasani Dhananjay, 1st, Shete, Prof. Virendra Virbhadra, 1st. Machine learning with python: machine learning with python. *INSC International Publisher (IIP)*, 2021.

4. Zolotukhin, O., Filatov, V., Yerokhin, A., Lanovyy, O., Kudryavtseva, M., Semenets, V.: An approach to the selection of behavior patterns autonomous intelligent mobile systems. *In: Proceedings of the IEEE International Conference on Problems of Infocommunications Science and Technology (PIC S&T),* P. 349–352. Kyiv (2021).

5. Zolotukhin, O., Filatov, V., Yerokhin, A., Kudryavtseva, M.: The methods for the prediction of climate control indicators in the Internet of Things systems. *CEUR Workshop Proc*. (2021).

6. Human-level control through deep reinforcement learning / V. Mnih et al. *Nature*. 2015. Vol. 518, no. 7540. P. 529–533. URL: https://doi.org/10.1038/nature14236 (date of access: 28.04.2025).

7. Understanding deep learning (still) requires rethinking generalization / C. Zhang et al. *Communications of the ACM*. 2021. Vol. 64, no. 3. P. 107–115. URL: https://doi.org/10.1145/3446776 (date of access: 28.04.2025).

8. Training asymptotically stable recurrent neural networks / N. J. Dimopoulos et al. *Intelligent automation & soft computing*. 1996. Vol. 2, no. 4. P. 375–388. URL: https://doi.org/10.1080/10798587.1996.10750681 (date of access: 28.04.2025).

9. A derivative-free optimization method with application to functions with exploding and vanishing gradients / S. Al-Abri et al. *IEEE control systems*

letters. 2021. Vol. 5, no. 2. P. 587–592. URL: https://doi.org/10.1109/lcsys.2020.3004747 (date of access: 28.04.2025).

10. Matthes E. Python crash course, 3rd edition. No Starch Press, Incorporated, 2022.

11. Pilgrim M. Dive into python 3. *Berkeley, CA : Apress*, 2009. URL: https://doi.org/10.1007/978-1-4302-2416-7 (date of access: 28.04.2025).

12. Chollet F. Deep learning with python, second edition. Manning Publications Co. LLC, 2021. P. 400 .

13. Filatov, V., Yerokhin, A., Zolotukhin, O., Kudryavtseva, M.: Methods of intellectual analysis of processes in medical information systems. *Inf. Extr. Process.* 48(124), P. 92–98 (2020). https://doi.org/10.15407/vidbir2020.48.092 (date of access: 28.04.2025).

14. Filatov, V., Semenets, V., Zolotukhin, O.: Synthesis of semantic model of subject area at integration of relational databases. *In: Proceedings of the IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL)*, P. 598–601. (2019). https://doi.org/10.1109/CAOL46282.2019.9019532 (date of access: 28.04.2025).

15. Lane H., Hapke H., Howard C. Natural language processing in action: understanding, analyzing, and generating text with python. Manning Publications, 2019. 544 p.

16. Chen Z., Wu M., Li X. Generalization with deep learning. *WORLD SCIENTIFIC,* 2021. URL: https://doi.org/10.1142/11784 Natural language processing in action: understanding, analyzing, and generating text with python. Manning Publications, 2019. 544 p.

17. Recurrent neural networks / ed. by L. Medsker, L. C. Jain. CRC Press, 1999. URL: https://doi.org/10.1201/9781420049176 (date of access: 28.04.2025).

18. Nakamoto P. Neural networks and deep learning: neural networks & deep learning, deep learning, blockchain blueprint. Createspace Independent

Publishing Platform, 2018. 152 p.

19. Bengio Y., Courville A., Goodfellow I. Deep learning. MIT Press, 2016. 800 p.

20. Rungta K. TensorFlow in1 day: make your own neural network. Independently Published, 2018. 364 p.

21. Tingiris S., Kinsella B. Exploring GPT-3: an unofficial first look at the general-purpose language processing API from openai. Packt Publishing, Limited, 2021. 296 p.

22. Zeng Z., Xiong D. Unsupervised and few-shot parsing from pretrained language models. Artificial intelligence. 2022. Vol. 305. P. 103665. URL: https://doi.org/10.1016/j.artint.2022.103665 (date of access: 28.04.2025).

APPENDIX A

Source Code of Components

1. App.js

```
import React from 'react'
import './App.css'
import {PREDICT_TYPES} from './lib/constants'
import ChartList from './components/ChartList'
class App extends React.Component {
  state = { predictType: null }
  onPredict = predictType => {
    this.setState({ predictType })  }
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <h1>Financial prediction</h1>
        </header>
        <div className="App-content">
          <div className="prediction-pane">
            <button              onClick={()              =>
this.onPredict(PREDICT_TYPES.TRAIN)}>Train</button>
            <button              onClick={()              =>
this.onPredict(PREDICT_TYPES.PREDICT)}>Predict</button>
          </div>
        </div>
        <ChartList predictType={this.state.predictType} />
      </div>
    )
  }
}
export default App;
```

2. Chartlist.js

```
import React, {Component} from 'react'
import {TYPES} from '../lib/constants'
```

```
      import Chart from './Chart'
      import {predictData} from '../lib/model'
      class ChartList extends Component {
        state = { data: { [TYPES.VOLUME]: [], [TYPES.PRICE]: [] },
isLoading: false }
        async componentDidUpdate(prevProps) {
          if ((prevProps.predictType === this.props.predictType) &&
this.state.isLoading) {
            return
          }
          const { predictType } = this.props
          this.setState({ isLoading: true })
          const data = {
            [TYPES.VOLUME]:     await     predictData(TYPES.VOLUME,
predictType),
            [TYPES.PRICE]:      await      predictData(TYPES.PRICE,
predictType)
          }
          this.setState({ data, isLoading: false })
        }
        render() {
          if (!this.props.predictType) {
            return null
          }
          const { data, isLoading } = this.state
          if (isLoading) {
            return <h1>Loading...</h1>
          }
          return (
            <div className='chart-list'>
              <Chart    name='Volume'    data={data[TYPES.VOLUME]}
predictType={this.props.predictType} />
              <Chart     name='Price'     data={data[TYPES.PRICE]}
predictType={this.props.predictType} />
            </div>
          );
        }
      }
      ChartList.propTypes = {}
      export default ChartList
```

3. Chart.js

```
import React, {Component} from 'react';
import {CartesianGrid, Legend, Line, LineChart, Tooltip,
XAxis, YAxis} from 'recharts'
const TYPES = {
  PREDICTION: 'prediction',
  REAL: 'real'
}
class Chart extends Component {
  render() {
    const { data, name } = this.props
    return (
      <div className="chart">
        <h2>{name}</h2>
        <LineChart width={730} height={250} data={data}>
          <CartesianGrid strokeDasharray="3 3" />
          <XAxis dataKey={name} />
          <YAxis />
          <Tooltip />
          <Legend />
          <Line type="monotone" dataKey={TYPES.PREDICTION}
stroke="#E65B6D" />
          <Line    type="monotone"    dataKey={TYPES.REAL}
stroke="#000" />
        </LineChart>
      </div>
    );
  }
}
Chart.propTypes = {}
export default Chart
```

APPENDIX B

Source Code of Algorithms

1. Model.js

```
import {buildReccurent, getBatchSize, getEpochs} from
'../lib/model'
import {createInputData} from '../lib/input'

class Model {
  constructor(type) {
    this.type = type
    this.epochs = getEpochs(type)
    this.model = buildReccurent(type)
  }
  async train() {
    const { type, epochs, model } = this
    const { xs, ys } = await createInputData(type)
    const batchSize = getBatchSize(epochs, type)
   await model.fit(xs, ys, {
      batchSize, epochs, callbacks: {
        onEpochEnd: async (epoch, log) => {
          return Promise.resolve({ epoch, log })
        }
      }
    })
  }
  async predict() {
    const { model, type } = this
    const inputs = await createInputData(type)
    const outputs = await model.predict(inputs.xs.mul(10))
    return Array.from(await outputs.dataSync())
  }
}

export default Model
```

2. model.js

```
import tf from '@tensorflow/tfjs'
import Model from '../models/Model'
import {LAYER_SHAPE, RNN_INPUT_LAYER_FEATURES} from
'./constants'
export const getEpochs = () => {
  return tf.resolveEpochs(RNN_INPUT_LAYER_FEATURES)
}
export const getBatchSize = type => {
  return tf.resolveBatchSize(getEpochs() / LAYER_SHAPE)
}
const getUnits = type => {
  return type ? 8 : 10
}
const createModel = type => {
  // Type is 'volume' or 'price'.
  // Get the number of neurons.
  const units = getUnits(type)
  const model = tf.sequential()
  model.add(tf.layers.dense({ inputShape: [LAYER_SHAPE],
units }))
  model.add(tf.layers.dense({ units: LAYER_SHAPE }))
  return model
}
export const buildReccurent = (type) => {
  const rnn_input_layer_features =
RNN_INPUT_LAYER_FEATURES
  const input_layer_neurons = getUnits(type, 'input')
  const outputLayerNeurons = getUnits(type, 'output')
  const model = createModel(type)
  // Configuring recurrent connection.
  const rnn_input_layer_timesteps = input_layer_neurons /
rnn_input_layer_features
```

```
    const   rnn_input_shape   =   [rnn_input_layer_timesteps,
rnn_input_layer_features]
    const lstm_cells = []
    for  (let  index  =  0;  index  <  rnn_input_layer_features;
index++) {
      lstm_cells.push(tf.layers.lstmCell({           units:
outputLayerNeurons }));
    }
    model.add(tf.layers.reshape({               targetShape:
rnn_input_shape }))
    return model
  }
  export const predictData = (dataType, predictType) => {
    const Predictor = new Model(dataType)
    return Predictor[predictType]()
  }
```

3. input.js

```
  import tf from '@tensorflow/tfjs'
  import {readXlsxFile} from 'read-excel-file'
  const INPUTS_PATH = '../input-data.xls'
  const OUTPUTS_PATH = '../output-data.xls'
  const getXlsData = path => {
    readXlsxFile(path).then(rows => {
      // `Rows` is an array of rows
      // Each row being an array of cells.
      return Promise.resolve(rows)
    })
  }
  const getInputs = () => {
    return getXlsData(INPUTS_PATH)
  }

  const getOutputs = () => {
    return getXlsData(OUTPUTS_PATH)
```

```
}
export const createInputData = async type => {
  // Returns an array of inputs.
  const inputs = await getInputs(type)
  const outputs = await getOutputs(type)
  const   xs   =   tf.tensor2d(inputs,   [inputs.length,
inputs[0].length]).div({ normalizer: 'rfa' });
  const   ys   =   tf.tensor2d(outputs,   [outputs.length,
1]).reshape([outputs.length, 1]).div({ normalizer: 'rfa' });
  return { xs, ys }
}
```

4. constants.js

```
export const TYPES = {
  VOLUME: 'volume',
  PRICE: 'price'
}
export const PREDICT_TYPES = {
  TRAIN: 'train',
  PREDICT: 'predict'
}
export const LAYER_SHAPE = 2
export const RNN_INPUT_LAYER_FEATURES = 10
```