MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
federal state autonomous educational institution of higher education
"National Research Nizhny Novgorod State University named after N.I. Lobachevsky. N.I. Lobachevsky National Research University of Nizhny Novgorod
(NNSU)

Institute of Economics and Entrepreneurship

Department of Information Technologies and Instrumental Methods in Economics
The focus (profile) of the bachelor's program:
"Applied Computer Science in Economics"

BACHELOR'S THESIS

CREATING AN APPLICATION TO ANALYZE THE VALUE OF STOCKS ON THE STOCK MARKET USING A NEURAL NETWORK AND THE PYTHON PROGRAMMING LANGUAGE.

Admitted to the defense:
Head of
"ITEME"
Doctor of Economics, Professor
_____ Yu.V. Trifonov
Signed

Performed:
Department Student Group 35191-PI-1u
Direction 09.03.03.
"Applied Informatics
_____ E.E. Sherstnev
Signed
Supervisor: Candidate of Phys&Math Sciences, Associate Professor
_____ P.A. Ruzanov
Signed

Nizhny Novgorod
2022

Table of contents

Introduction

The financial sector of the economy is a set of institutions that provide financial services to the population and businesses. As a rule, it includes tax, budget and monetary systems.

This sector has a direct impact on the development of the real economy through its funding, because it acts as a cash center and provides, through financial instruments, the possibility of access to all financial resources for borrowers. In addition, this sector also provides insurance, investment and advisory services to all economic agents. The investment sector is one of the most important parts of the financial world, which has a direct impact on the economy of the country and the world as a whole.

Over time, the instruments by which the various financial assets were handled have undergone major changes. Nowadays, advanced information and computer technologies, together with all kinds of technical tools, are actively integrated into the daily processes on stock exchanges and foreign exchange markets.

The purpose of this work is to develop an application for analyzing the value of stocks using the language "Python" and evaluate the effectiveness of modern machine learning technologies, namely neural networks with short-term memory to predict the value of financial assets.

1. Value Analysis shares

1.1 Description of the subject area

Financial market is a market where free monetary capitals and savings are redistributed between various economic entities by means of transactions with financial assets. The object of redistribution and, accordingly, of financial relations is an asset, which can be represented as national or foreign currencies, precious materials, securities and derivative financial instruments. The main pricing factor is changing supply and demand balance, which is influenced by a wide range of economic laws and information events [6].

The classical approach defines the purpose of the financial market as ensuring the functioning of the mechanism of mobilization and redistribution of financial resources between different economic entities in different sectors. This goal is achieved through structured functional segmentation [4]. Structural divisions, which the modern financial market consists of, can be seen in figure 1.

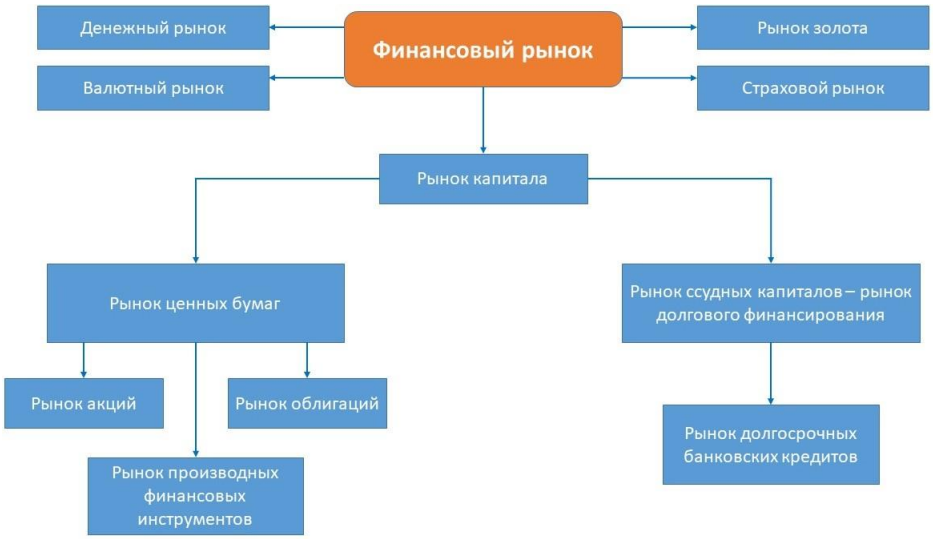


Figure 1: The structure of the financial market

Securities market is a set of economic relations on the issue and circulation of securities and derivative financial instruments between its participants in order to identify an equilibrium current (fair) price and obtain investment income [5]. Within the framework of eco-system of financial relations its main task is to form a mechanism to attract investments of different volumes into the economy by establishing relations between those who need money and those who want to invest.

The objects of the market are documents that certify property rights to a particular asset, the exercise and transfer of which is possible only by presenting them. There are several types of securities, the main of which are:

– Shares - issuance securities, fixing the rights of the owner (shareholder) to receive part of the profit of a joint stock company in the form of dividends, to participate in the management of the joint stock company and the property that remains after its liquidation [5].

– Bonds are fixed-income securities in which the issuer is obliged to pay the owner of the bond a certain amount of interest according to a certain scheme and, in addition, on the day of maturity, the face value of the bond [5]. Securities market participants are represented by trading and accounting infrastructure, financial intermediaries, dealers, information and analytical agencies, investors and issuers. Each of these subgroups appeared in the financial system as a result of gradual structural and functional development of the world market system. Similar to the formation of the first analytical firms at the end of the last century, the private equity sector is now going through a period of rapid The rise and expansion of the world.

Specialized organizations - brokers, thanks to advanced technologies in the field of communication and tele-communication, allow anyone who wishes to participate in the auction with the help of computers and smartphones.

In addition to the basic exchange functions, many modern brokers provide their clients with additional functionality related to the analysis of the current situation on the markets, forecasting the value of certain assets and even act as tax agents in order to simplify the interaction of end users with financial institutions [4].

1.2 Economic content of the task of analyzing the value of shares

The modern financial market is an extremely vast and dynamic system of relations, and to keep track of it is often an extremely difficult task. A beginning broker, or at times even an experienced professional without due experience and theoretical preparation are not able to calculate specific regularities in the information field or to make a statistical analysis of one or another asset's performance in the context of current events. That is why it was not difficult to foresee that after the possibility to trade on the stock exchange will follow the system of services, online-services and companies providing their clients with the possibility not only to get a reliable answer to the question "Why this share is worth so much", but also to understand how this price was formed and what can expect it in the future.

As it was mentioned before - the concept of "Share Value Analysis" includes several components. Since the middle of the last century the tendency of opening the next trading day was determined by the headlines and morning news reports. Today only the ways of getting information and how fast the economic and financial situation can change have changed. However, in spite of increased market volatility in recent years, classical methods of mathematical statistics and analysis are reliable and widely used tools, used by both major players and private investors.

In modern trading practice, the task of stock price analysis is a set of different mathematical and software

methods of scanning retrospective data on the price of the selected asset in order to find indicators and additional indicators that can give a relatively accurate forecast of further behavior of the stock on the exchange. The key criteria for successful design of the application that solves this problem are reliability of displayed data, speed of operation and possibility of flexible adjustment of the system for individual scenarios of use by investors of different levels.

1.3. The role and place of the task of value analysis shares

The role of stock price analysis in the economic information system of an online broker or an international exchange portal is hard to overestimate. Accurate and conveniently provided information is an essential function of any financial system, because it is based on the results of such applications that decisions on transactions are made.

Modern trading systems develop subsystems of asset analysis, which consist of different methods of statistical analysis and mathematical functions. Most of such subsystems provide the possibility to customize methods of primary data transformation and results display [7]. The goals of developing systems for analyzing the value of stocks are:

- Providing detailed retrospective data on the dynamics of asset values
- Calculation of indicators reflecting reliability and profitability of the selected asset
- Calculation of forecast data for the selected asset

2. Feasibility study for stock price analysis application

2.1. Description of the object

"Yandex is an advanced Russian transnational IT company, founded in 2000 by entrepreneur and programmer Arkady Volozh. The main activities of the corporation are the development, maintenance and development of its own search engine, Yandex.ru, as well as other online services and applications [20].

The company is registered in the Russian Federation as a limited liability company, with 100% of its authorized capital held by Yandex N.V., a joint-stock company under Dutch jurisdiction. At the moment, Yandex has over ten thousand employees, and its head office is located on Lev Tolstoy Street in Moscow.

In recent years, Yandex, keeping up with its Russian and foreign competitors, has taken the course of creating its own virtual ecosystem. Even now, such projects as Yandex.GO, Yandex.Maps, and Yandex.Food are among the leaders in the lists of the most visited services in our country and neighboring countries [20]. Most of the company's products use neural networks and other types of machine learning. "Yandex offers machine translation from dozens of languages, and branded recommendation systems help choose products and a variety of content. The corporation willingly participates in scientific research and shares technologies. Yandex has cooperated with the European Center for Nuclear Research (CERN) for many years; its proprietary CatBoost machine learning technology and ClickHouse DBMS are open to developers and researchers from all over the world. In addition to its own developments, Yandex spends a lot of time and resources to support promising technology startups and private companies to accelerate the development of the IT industry.

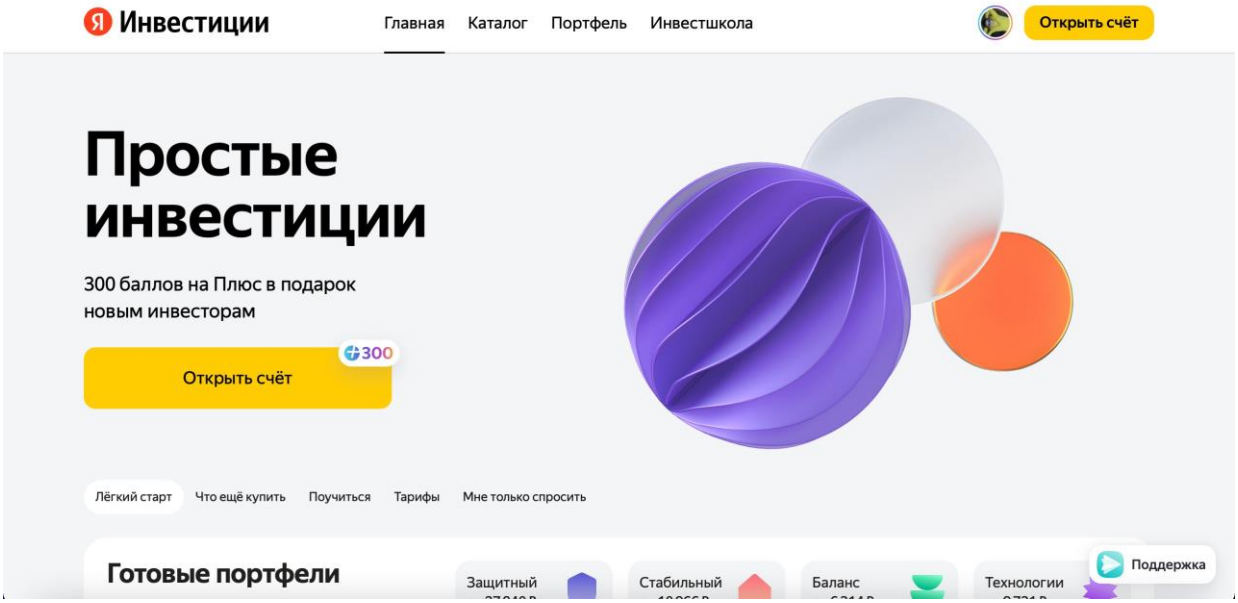


Figure 2: Yandex.Investments portal

Yandex was chosen as the practice's object of study due to the presence in its ecosystem of one of the company's fastest-growing services, the Yandex.Investments portal. This is a joint project of the IT giant and one of the largest national banks

"VTB. With the help of this portal, every user has the opportunity to invest money in securities, trade them and exchange currency on the Moscow and St. Petersburg Stock Exchanges.

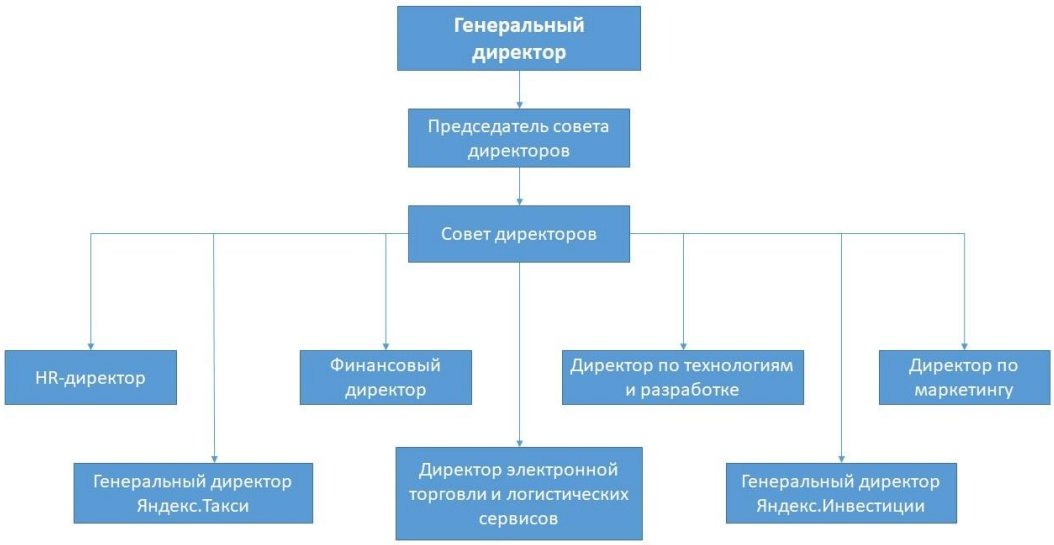


Figure 3: Organizational management structure

Similar to all divisions, subsidiaries and projects of Yandex, the Investments service is managed by the company's Board of Directors and CEO.

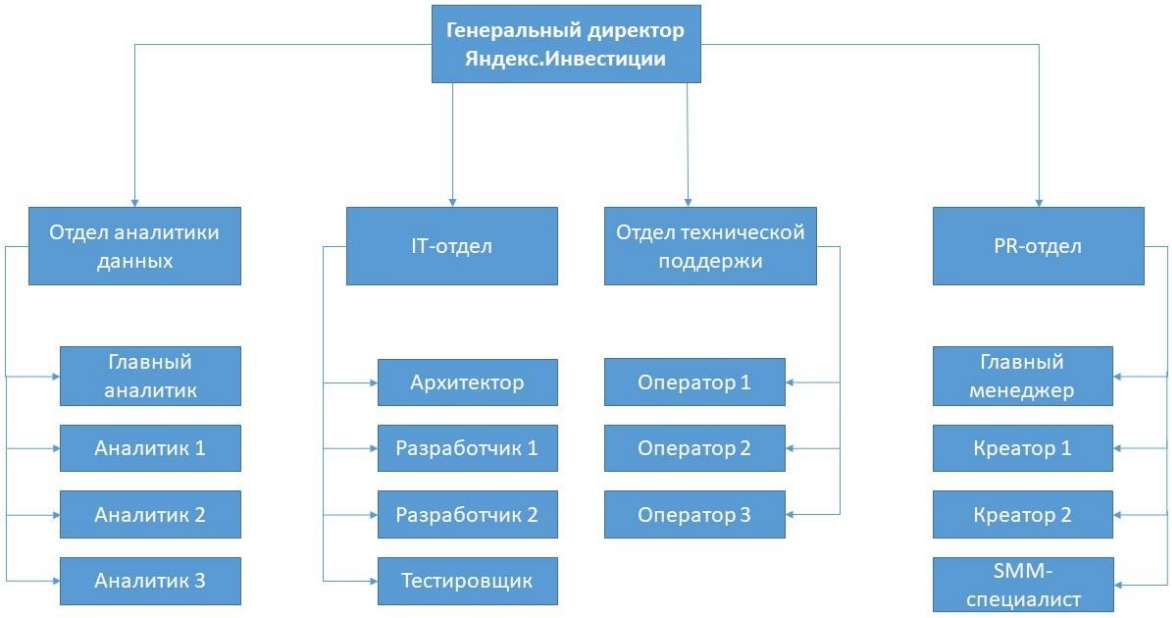


Figure 4: Functional management structure

The functional management structure reflects the list of the company's main departments and their areas of responsibility. Since Yandex.Investments is a subsidiary of Yandex, the management of the parent company has decided not to create a wide range of subdivisions within the new project, but to focus on a set of main departments and, if necessary, to bring in additional specialists from the main company.

2.2. Characteristics

The task of creating an application for analyzing the value of shares is a rational strategic stage in the development of the Yandex.Investments portal. If you look at the chronology of changes made to the service's functionality, you can see that, first of all, the service developers provided the possibility to perform basic actions on buying, selling, and a brief overview of asset prices on financial markets. A bit later there was added a possibility to add financial instruments to the favorites, and also was created

section with educational materials and reference information. A complete diagram of the relationship between the functional calculations can be found in Appendices 1 and 2.

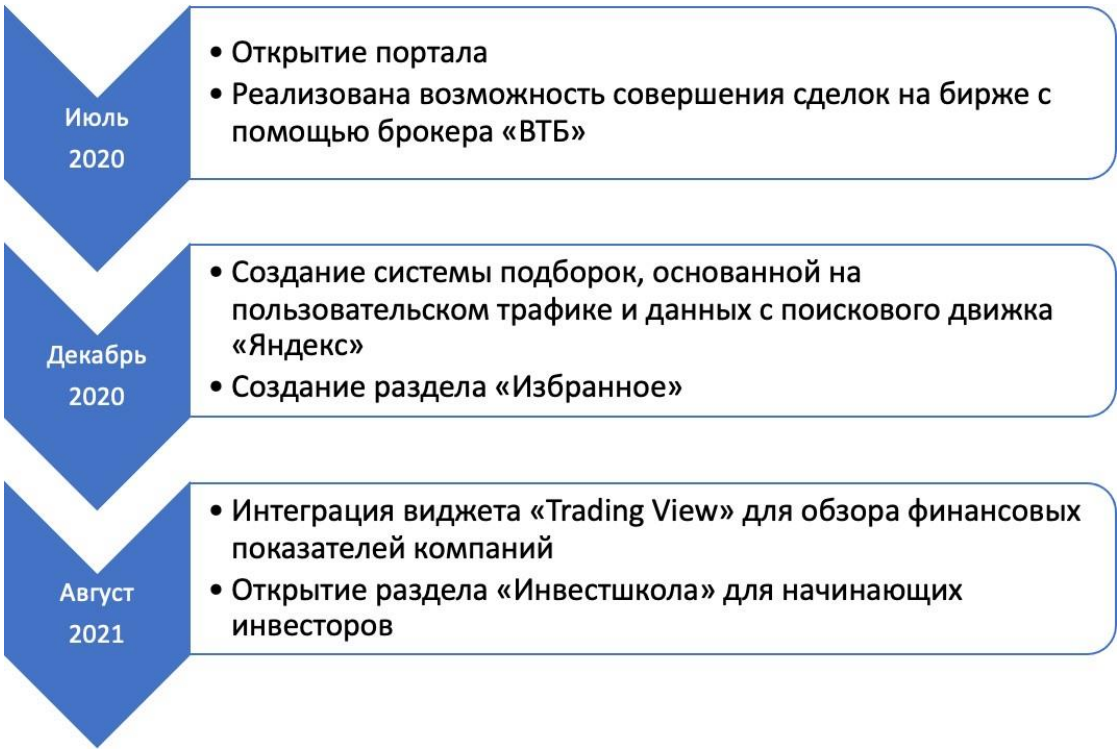


Figure 5. Chronology of Yandex.Investments service development

Taking into account that in the first months of operation the portal's target audience consisted of investors who did not have much experience in financial activities, we can conclude that the initial steps in the development were correctly defined, but for professional investors the availability of tools for deep analysis of assets is one of the priority criteria in choosing a broker or a monitoring system.

Let's take a look at the main functions and features that should be implemented in a share price analysis application:

– Ability to integrate the application into the current Yandex.Investments portal system

– Availability of information on all available shares listed on world stock exchanges

– Possibility to choose the period of the information used

– Implementation of the moving average method as one of the most extensive methods for smoothing and predicting time series

– Implementation of methods of short-term and long-term prediction of the value of the selected share

– Convenient and clear display of algorithm results

2.3. Characteristics and analysis of the existing organization of information conversion

As part of fulfilling the main goal of the Yandex.Investments service, namely to enable clients to operate with securities on the stock exchange, each of the project's divisions previously uses a set of different software tools for working with information.

The project team maintains the portal around the clock, advises and educates clients on the basic principles of investing and creates software solutions and tools that facilitate the process of interaction with the platform for end users. The list of software products required to perform the full range of tasks of this division is constantly changing due to the constantly developing technologies, but we can highlight several fundamental tools used regularly:

Yandex.Cloud is its own public cloud platform that aims to provide scalable data storage infrastructure, machine learning tools and development tools. This platform significantly reduces the effort required to handle large volumes of data and significant information flows, which is a very important plus, given the current principles of financial markets[19].

Yandex.Direct - a system of contextual advertising on the pages of "Yandex" and the sites of partners of the Yandex Advertising System. The service allows you to create

effective advertising campaigns and track the dynamics of visits to the target portal from different sources[19].

GitHub is the largest service for hosting IT projects and their collaborative development. This tool is necessary for structured and organized work of the team with the portal source code, as well as fast and convenient control over the versions of the developed solutions.

Visual Studio Code is a popular code editor from Microsoft. Thanks to a large number of users and constant support from the developer, this application can be adjusted to the constantly changing technologies and programming languages, which makes the work of the team much easier.

Jira is a comprehensive bug tracking and project management system. This service allows you to record the occurrence of errors in the work of the portal, as well as to track the process of fixing them.

Oracle SQL is an integrated database management system. Within the Yandex.Investments service, it serves as the main means of storing all types of data. It is used in conjunction with the Yandex.Cloud service.

Matlab is a package of applied programs for performing time-consuming calculations and building mathematical models. It is used for analysis of the securities market and creation of informational articles and training materials for service users.

Microsoft Office is a software package for working with different types of documents: text documents, tables, presentations. An important part of this package is the Microsoft Teams program, which implements a convenient way for employees to communicate within the project.

The Adobe software package is a software solution for working with various formats of media files: images, video, audio, animations. It is used primarily to create the visual image of the portal.

Zendesk - a software package for user support service. Allows you to quickly and quickly respond to the appeals of users.

2.4. Review and analysis of design developments and software solutions

Taking into account the software that is already used at the object and the necessary functionality of the developed system, we can conclude that the most convenient variant of solving the set task will be the creation of a web-application.

A web application is an auxiliary software tool designed to automate the execution of actions on web servers. At the same time, they use web-browsers as user interfaces. Usually web-applications are created in different variants of client-server architecture [2]. Lately this way of creating applications is very popular. This is caused by a number of factors, among which are the relative ease of creation, the ease of integration into third-party services, and the possibility of using any operating system.

There are several open-source tools for web application development today. Each of them differs in programming language, usage patterns, and speed of operation. Here are the most common options:

– Angular is a framework for dynamic web applications that allows you to use HTML as a template language and then extend the HTML syntax to express application components. With data binding and dependency embedding, you can eliminate most of the code that would have to be written [10].

– Django is a high-level Python web framework that allows you to quickly build secure and maintainable websites. Built by experienced developers, Django takes most of the work out of web development, so you can focus on writing your web application without having to delve into the intricacies of architecture. It's free and open-source, and has a growing and active

community, excellent documentation, and many options for both free and paid support [11].

– ASP.NET (Active Server Pages for .NET) is a web application development platform, which includes: web services, software infrastructure, programming model, from Microsoft. ASP.NET is part of the ".NET Framework" platform and is a development of the older "Microsoft ASP" technology. [3].

2.5. Selection of design solutions

Despite their wide popularity, none of the previously presented frameworks for developing web applications is the best implementation method in the context of our task. To substantiate the reasons for this conclusion, let's turn to the current version of the Yandex.Investments portal. On Figure 6 you can see information about SBER shares as of May 11, 2022.

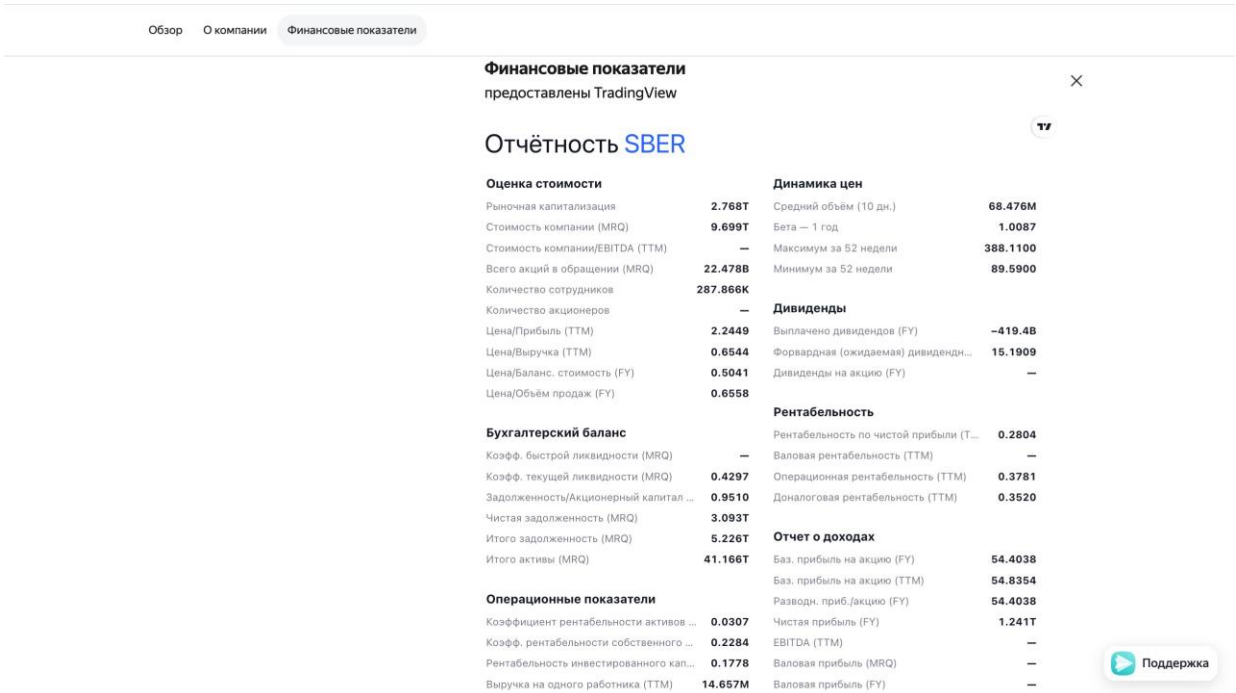


Figure 6. TradingView widget

It is easy to see that one of the sections of any asset is the "Financial Indicators" tab. This data gives clients of the service

primary view of the selected company's activity, its capitalization and profitability [9]. This information is displayed using the third-party financial analytics service "Trading View".

Trading View was founded in 2011 and since then it has managed to earn a reputation as the most famous and popular investment website in the world. The monthly activity of the service counts 30 millions of visits and the average user rating is 4.9 out of 5.

Trading view" service provides its private and corporate clients with an extremely wide range of up-to-date information about brokers and all the assets available on the market. As a method of integrating its own data into web-services and clients' websites "Trading view" uses a system of widgets - ready-made elements of graphical interface. This method is the second most popular method of connecting third-party applications to web-services, being second only to the "REST API" architecture.

Returning to our task, we can make a conclusion about the expediency of developing a system for analyzing the value of stocks, which can be integrated into the existing portal structure as a widget, like the product of TradingView. This is why the Streamlit platform was used for this task.

Streamlit is a web application platform that helps us create and develop Python-based web applications that can be used to share analytics results, create sophisticated interactive interfaces, and illustrate new machine learning models [16].

Among the main advantages of this system are:

- Ability to integrate as a widget on any existing Internet portal

– Easy-to-use out-of-the-box solutions for creating custom interface

– The availability of integrated libraries for processing and displaying of large amounts of data

– Convenient possibility of debugging via local server

– High performance

– Comprehensive documentation for developers

3. Information

3.1. Output information

The output information for the share price analysis system will be data on the retrospective value of the selected asset, as well as the results of calculations of the predicted price for the selected period.

Table 1

Output information

№	Identifier	Value type	Length	Title
1	data	number	10	Retrospective data on the value of the stock tiva
2	Exp1	number	10	The value of the moving average with timeframe 20
3	Exp2	number	10	The value of the sliding average with a timeframe of 50
4	fig	graph	-	Consolidated schedule of re tronic data
5	forecast	number	10	Data calculated with the help of "Prophet."
6	fig1	graph	-	The graph with the results of the- Prophet.
7	Pred_price	number	10	Forecast, calculated- using a neural network

3.2. Input information

The input information within our task will be all user parameters that are necessary for the correct functioning of the main calculation and prediction algorithms. The ways to enter this information will be implemented using the corresponding interface elements (see item 4 for details).

Table No. 2

Input information

№	Identifier	Value type	Length	Title
1	Selected_stock	string	4	Name of financial- of the asset
2	Start_date	date	6	The date of the beginning of the examination of the the period in question
3	End_date	date	6	The date of the end of the period under consideration ode to
4	n_years	number	2	Number of years for the system of long- term forecast

4. Software and hardware

4.1. Description of technical and software

The Python programming language is one of the most widespread tools in the IT-industry. The range of tasks solved by this development tool includes both performing various mathematical and statistical calculations, and creating interfaces to end-products and maintaining the supporting infrastructure of various applications [13].

Python has become particularly popular in the field of data analysis and cloud computing. Python has gained popularity in the field of data analysis and cloud computing, and its interpretive structure and instruction execution speed allow it to optimize the use of workstations' computing power and to quickly correct errors and inaccuracies that may occur. Another advantage is that Python is a high-level, open-source programming language. These features explain the well-known ease of learning and the ease of reading the code. The open source model also allows any developer to create plug-in libraries and modules to solve even the most specific problems.

"Python for finance has many advantages and the competitive edge to take the finance industry to a new stage of development. One reason is the strong ecosystem of millions of users, modules and lessons. Due to the increasing amount of financial data, people are no longer able to analyze and evaluate it professionally. That is why machines have come to replace them and analyze financial data at an incredibly low cost and high speed [12]. There is a very close connection between artificial intelligence (AI) and finance. Therefore, it is not at all surprising that "Python" has become the main language for AI-based data analysis.

The software modules used to solve the task can be divided into two groups: auxiliary libraries and main libraries. The first subgroup is a set of ready-made tools for convenient work with data and methods of its display. The second is a set of frameworks that contain the methods that make up the algorithms for the basic functions of the application being designed. Let us consider the modules of each group.

"Pandas is a Python language library for data processing and analysis. Pandas" works with data on the basis of the library
"NumPy", which is a lower-level tool. It provides special data structures and operations for manipulating number tables and time series. Its main application is to provide a Python environment not only for data collection and cleaning, but also for data analysis and modeling tasks, without switching to more statistical-specific languages (such as R and
"Octave).

The basic function of this library for the task at hand is the use of "Dataframe" type containers. These data arrays are multidimensional tables designed to perform the operations provided by the Pandas module. A separate subdivision of the framework is the Pandas Datareader library, the main function of which is to import into the application the data on the financial indicators of the assets placed on the information portals of the international exchange aggregators (in our case, Yahoo finance was used). The key point for our task is availability of built-in functions "ewm" and "rolling", which allow calculating values of moving averages according to formulas described in item 4.1.

A description of all the basic software methods of the "Pandas" library used in the solution is given in the table below.

Table #3

Methods of the "Pandas" module

The method	Description	Input Parameters
pandas.data-reader(...)	allows you to create data from various Internet sources	– Name of the asset you are looking for – Data source name – Start of temporary cutting – End of temporary cutting
pandas.roll-ing(...).mean()	Calculates the value simple moving average	Длина smoothing
pan-das.ewm(...).mean()	Calculates the value of exponential moving average	– Smoothing length – Exponential weighted function formula

"Plotly" is a Python programming language library for visualizing data in two- and three-dimensional graphics. The module is a flexible, easily configurable package that, together with "NumPy", "SciPy" and

"Pandas provides features similar to the programming environment

"MATLAB. The package supports many kinds of graphs and charts:

- Graphs
- Scatter Plots
- Bar charts
- histograms
- Pie charts
- Barrel chart list

– Contour charts
– Gradient fields
– Spectral diagrams

The user can specify coordinate axes, grids, add labels and explanations, use a logarithmic scale or polar coordinates.

For our task, we will use the "pyplot" section to plot the changes in the value of the asset and display the nature of the moving averages.

Table No. 4

Methods of the "Plotly" module

The method	Description	Input Parameters
plotly.figure()	Creates a fi gurus	-
plotly.add_trace(...)	Adds a new chart on the figure	□График to add
plotly.scatter(...)	The type of graph in which each element of the input data is displayed as a point on the coordinate plane	– Input data on the abscissa axis – Input data on the ordinate axis – Chart name
plotly.layout.update(...)	Additional figure display parameters	– Name of the figure – Scroll bar and mass stacking display status

Among the list of tasks facing the projected application is the ability to calculate forecast data for a selected financial asset.

Classical time series forecasting methods are based on statistical models, which requires a lot of effort to adjust the models. A person has to choose the most appropriate parameter values depending on each case. Tuning these methods requires a deep understanding of how basic time series models work and some organizations find it difficult to handle such forecasts without specially created units. That is why this functionality will be implemented using machine learning technologies. Let us consider two frameworks that are among the most convenient for working with time series forecasting at the moment.

"Prophet is a library created by Meta (formerly Face- book) for time series forecasting based on additive model. It works best with time series that have pronounced seasonal effects and large amount of initial retrospective data, which shows the validity of using this technology in financial markets. The main features of this forecasting tool are stability to non-standard changes in the trend and good performance when dealing with abnormal values [18].

Table No. 5

Methods of the Prophet module

The method	Description	Input Parameters
prophet.fit(...)	Allows To add retrospective data modeled	□Массив data
prophet.make_fu- ture_dataframe(...)	Allows you to set the duration of the forecast	□Длительность pro- gnosis
prophet.predict(...)	Based on the data entered data makes Forecast	□Длительность pro- gnosis

"Keras is an open source library written in Python that provides interaction with artificial neural networks. It is aimed at operational work with deep learning

networks and is designed to be compact, modular and extensible. This library contains numerous implementations of commonly used building blocks of neural networks, such as layers, target and transfer functions, optimizers, and many tools to simplify work with images and text [15].

Table No. 6

Methods of the "Keras" module

The method	Description	Input Parameters
Keras.add(...)	Adds a new layer into a neural network	Тип layer
Keras.compile(...)	Builds the model with the specified layers	– Optimizer – Error calculation method
Keras.fit(...)	Model training	– Array of retrospective data – Array of expected results – Number of iterations – Number of eras
Keras.predict(...)	Forecast calculation	Массив retrospective data

To implement the above-mentioned methods and stages of solving the task, "Python" version 3.10 of June 17, 2021 is used. Editing the program code, loading the necessary libraries, as well as interpreting the program and obtaining the results is performed with the help of the multifunctional development environment "Anaconda" and the built-in text editor "Spyder". The source code of the application as well as the results of the work can be seen in appendix 3-6.

4.2. Algorithm

4.2.1. Moving Average Method

Moving averages are price-based lagging indicators that display the average price of an instrument over a set period of time. This tool is a good way to assess momentum as well as confirm trends and identify areas of support and resistance [8]. Essentially, moving averages smooth out the "noise" when trying to interpret charts. The noise consists of fluctuations in both price and volume. Because moving averages are lagging indicators and react to events that have already occurred, they are not used as a forecasting tool, but rather as an interpretive method used for confirmation and analysis.

The calculation intervals or periods used may vary considerably, depending on the type of technical analysis performed. One fact that should always be kept in mind is that moving averages have a built-in lag. The longer the timeframe (time frame) used, the greater the lag. Likewise, the shorter the timeframe, the smaller the lag will be. Moving averages with shorter timeframes stay close to prices and will move as soon as prices move. Longer timeframes have much more cumbersome data, and their movements will lag significantly behind market movements. As for what timeframes to use, these are

really depends on the discretion of the user. The timeframes are tacitly divided into the following subgroups:

- Short-term (up to 20 days)
- Medium term (20 to 60 days)
- Long-term (more than 60 days)

There are several different types of moving averages, which have the same basic principle but different modifications. The most notable are the simple moving average (SMA), the exponential moving average (EMA) and the weighted moving average (WMA).

Simple moving average (SMA) - belongs to the class of indicators, which can be used to determine the beginning and end of a new trend, its strength and speed of movement. As a rule, the simple moving average is calculated as a result of addition of prices of closing of the instrument for the set number of periods and the subsequent division of the sum by number of periods [8].

$$SMA = \frac{1}{n} \sum_{i=1}^{n} P_i$$

- Pi are market prices
- N - order of the moving average - length of smoothing (the number of prices that are included in the calculation of the moving average or the number of bars for which the indicator is calculated)

As you can see from the formula, a simple moving average is an indicator of the equilibrium price, that is, a simple arithmetic average of prices over a given time period. Consequently, the shorter the moving average, the less time it takes for equilibrium to emerge. The SMA always follows the main market trend with a certain lag, while filtering out minor fluctuations. It follows that the lower the SMA parameter or the shorter it is, the faster the SMA sets a new trend but makes more false oscillations. Conversely, the longer

The slower the new trend is set, but there are fewer false fluctuations.

The exponential moving average (EMA) gives higher weight to the latter than to previous values. This fact makes it possible to react to current market price changes faster than with simple moving averages. The weight of the last price depends on the period of the moving average, and the shorter the period, the more weight the last price has.

$$_{!}\, EMA = EMA_{!\%\$} + (K \times [P_{!} - EMA_{!\%\$}])$$

- i - current time moments
- i - 1 previous moments of time
- Pi - market prices

$$K = \frac{2}{n+1}$$

- n - number of periods

It is worth noting that all prices for the entire period of its construction are used in calculating this indicator. With time the influence of the old prices decreases, but does not disappear at all. This effect disappears faster for short EMAs than for long ones. The difference between simple moving average and exponential moving average is not very noticeable on the real chart, but some differences are nevertheless present.

4.2.2. Forecasting with "Prophet"

According to an article on the developer's website, Prophet was created to predict a large number of different business metrics. In addition, the library makes it possible, by modifying easy-to-understand

parameters, improve predictions, and do not require analysts to have in-depth knowledge of the predictive model structure [17].

Let's take a look at how the Prophet library works. In essence, it is an additive regression model consisting of the following components:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_{\&}$$

The seasonal components s(t) are responsible for modeling periodic changes associated with weekly and annual seasonality. Weekly seasonality is modeled with dummy variables. Six additional attributes are added, such as [Monday, Tuesday, Wednesday, Thursday, Friday, Saturday], which take values 0 and 1 depending on the date. The sign "Sunday" corresponding to the seventh day of the week is not added, because it would depend linearly on the other days of the week and that would affect the model. The annual seasonality is modeled by Fourier series.

The trend g(t) is a piecewise linear or logistic function.

$$g(t) = \frac{C}{1 + \exp\left(-k(t - b)\right)}$$

The logistic function allows you to simulate growth with saturation, when the rate of growth decreases as the indicator increases. A typical example
- is the growth of the app's or site's audience.

Among other things, the library is able to select optimal points of trend changes based on historical data. But they can also be set manually (for example, if you know the dates of new functionality releases that strongly influenced the key indicators).

The h(t) component is responsible for user-defined abnormal days, including irregular days such as Black Fridays.

The error contains information that is not considered by the model.

4.2.3. Forecasting with "Keras"

As it was mentioned before - "Keras" is a convenient tool for working with neural networks in the "Python" environment. This module contains

a fairly large set of off-the-shelf architectures for solving problems using machine learning [13].

A neural network is a sequence of neurons connected by synapses. The structure of a neural network came to the world of programming directly from biology. Thanks to this structure, a machine acquires the ability to analyze and even remember various information. Neural networks are also capable of not only analyzing incoming information, but also reproducing it from its memory. Neural networks are used to solve complex problems that require analytical calculations similar to those performed by the human brain [1].

A neuron is a computational unit that receives information, performs simple calculations on it, and passes it on. They are divided into three main types: input, hidden and output. When a neural network consists of a large number of neurons, the term layer is introduced. So, there is an input layer that receives information, a certain number of hidden layers (usually not more than 3) that process it, and an output layer that outputs the result.

For each network element there is a certain rule according to which its output value is calculated from the combined input value of the element. This rule is called an activation function. The activation function can be any mathematical function, but the most popular ones are the sigmoid and the hyperbolic tangent.

A synapse is a connection between two neurons. Synapses have 1 parameter - weight. Thanks to it, the input information changes when it is transferred from one neuron to another.

The key principle of any neural network is to pass input information from layer to layer, taking into account weights and applying activation functions in each neuron, then, as a result of comparing the result of

networks with the expected result there is an adjustment of the synapse weights and the whole process is repeated over and over again.

The basic neural network architecture is not the best way to solve tasks of financial instruments price prediction. This is due to the fact that in case of price analysis, the so-called "pattern thinking" will have a significant advantage. This method is used by all experienced traders and its basis is the ability to find situations similar to the current situation in retrospective information and to make forecasts on the basis of past behavior.

This task is performed by a special kind of neural networks - re-current neural networks, namely, LSTM networks.

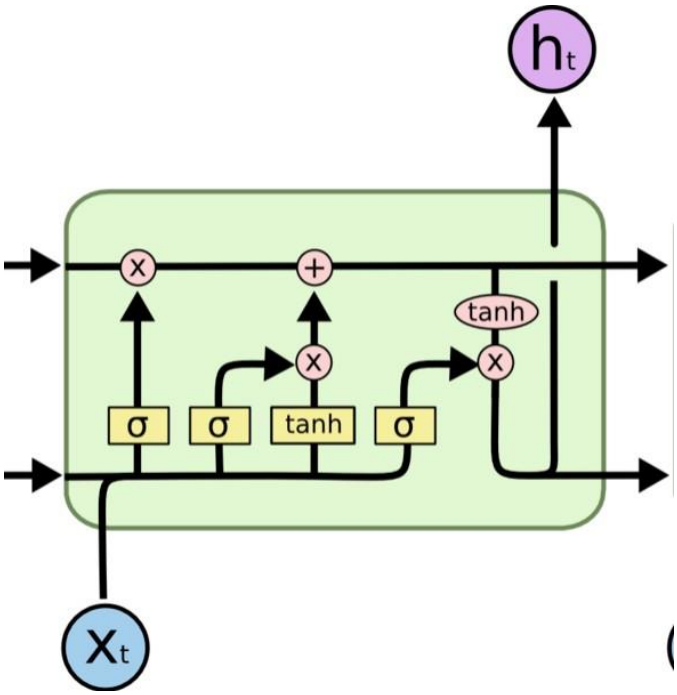


Figure 7: LSTM network

Recurrent neural networks add memory to artificial neural networks, but the realized memory is short - at each learning step the information in the memory is mixed with the new one and after several iterations it is completely overwritten.

LSTM modules are specifically designed to avoid the problem of long-term dependence by memorizing values for both short and long time intervals. This is because the LSTM module does not use an activation function within its recurrence components. Thus, the stored value is not blurred in time [14].

The LSTM network algorithm can be broken down into four steps:

First, the "forgetting filter layer" determines what information can be forgotten or left behind. The values of the previous output $h_{\&\%\$}$ and the current input $x_{\&}$ are passed through the sigmoidal layer. The resulting values are in the range [0; 1]. Values that are closer to 0 will be forgotten and those closer to 1 will be left.

$$_{\&}f = \sigma(W_{\cdot} \cdot [h_{\&\%\$}, x_{\&}])$$

Whe
re,

$\cdot$ W- matrix of weights leading to the sigmoidal layer

Next, it is decided what new information will be stored in the cell state. This step consists of two parts. First the sigmoidal layer called "input filter layer" determines which values to update. Then the tanh-layer builds a vector of new candidate values $CF_{\&}$ that can be added to the cell state.

$$i_{\&} = \sigma(W_{!} \cdot [h_{\&\%\$}, x_{\&}])$$
$$_{CF\&} = tanh(W_{(} \cdot [h_{\&\%\$}, x_{\&}])$$

To replace the old state of cell $_{C\&\%\$}$ with the new state $C_{\&}$. You need to multiply the old state by $f_{\&}$, forgetting what you decided to forget before. Then add $i_{\&} \times CF\&$. These are the new candidate values multiplied by *t* - how much to update each of the state values.

$$_{\&} C = f_{\&} * {}_{C\&\%\$} + i_{\&} * {}_{CF\&}$$

The last step determines what kind of information the output will be. The output data will be based on our cell state and some filters will be applied to it. First, the values of the previous output $_{h\&\%\$}$ and the current input $x_{\&}$ are passed through a sigmoidal layer, which decides what information from the cell state will be output. The cell state values are then passed through the tanh layer to output values in the range of -1 to 1, and multiplied with the output values of the sigmoidal layer to output only the desired information.

$$o_{\&} = \sigma(W_{)} \cdot [_{h\&\%\$}, x_{\&}$$
$$]) \; h_{\&} = o_{\&} * \tanh{(C\,)}_{\&}$$

4.3. Development and implementation of software

The first step to create an application is to import the necessary libraries and install the dependencies. The necessary modules can be downloaded to the desktop manually via the terminal, or you can use an off-the-shelf development environment and tools for data analysis
"Anaconda.

The main frameworks have already been described in the previous paragraphs of this paper, but it is worth highlighting a few libraries that are auxiliary to the task at hand.

– The "datetime" module provides classes for handling the time and date in different ways. The standard way of representing time is also supported, but more emphasis is placed on easy manipulation of date, time and its parts.

– Scikit-learn is one of the most widely used Py- thon packages for Data Science and Machine Learning. This tool allows to simplify the work on preliminary data processing, as well as classification and scaling of "raw" information. Within our application we will use the "MinMaxScaler" subsection to transform the original dataset into a suitable one to be used as input neurons of the neural network being created.

– NumPy is a Python extension that adds support for large multidimensional arrays and matrices, along with a large library of high-level mathematical functions for operations on those arrays.

```
import streamlit as st
import datetime

import pandas_datareader as web
import pandas as pd

from plotly import graph_objs as go

from fbprophet import Prophet
from fbprophet.plot import plot_plotly

from sklearn.preprocessing import MinMaxScaler
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM
```

Figure 8: Importing modules

The first thing to do is to initialize the application, define the interface features, and create basic elements that allow the user to enter initial information. In the course of this task, you can make sure that designing a web application with Streamlit is extremely user-friendly.

For ease of use, the best solution is to have all the input elements on a separate panel attached to the left side of the screen. This solution will allow the user to change the settings of the analysis system and select the necessary forecasting tool at any time.

First of all, let's define the page and sidebar headers using the methods "title" and "sidebar.header". After that you should add a field for selecting a financial instrument and a time period with retrospective data on which the forecast will be built. At the bottom of the sidebar create a slider to select the forecast term (in case of using
"Prophet") and the switch of the prediction method itself.

```
#---defining the app
st.title('Stock analysing app')

#---side bar set-up
st.sidebar.header('Choose the company for analysis')

#company choose
stocks = ('GOOG', 'AAPL', 'MSFT', 'GME')
selected_stock = st.sidebar.selectbox('Select company', stocks)

#date period choose
start_date = st.sidebar.date_input("Start date", datetime.date(2012, 1, 1))
end_date = st.sidebar.date_input("End date", datetime.date(2019, 12, 17))

#period of prediction choose
n_years = st.sidebar.slider('Years of prediction:', 1, 4)
period = n_years * 365

#Forecast method select
method = st.sidebar.radio('Forecast method',
                          ('Additive regression model',
                           'Neural network'))
```

Figure 9: Initializing the application

The next section of the program will request the value of the selected asset from a publicly available aggregator

financial information. This aggregator in our case will be the portal "Yahoo Finance", which provides the possibility to obtain actual data using the library "Pandas Datareader". This functionality is displayed in the "load_data" function; the input parameter in this case will be the name of the selected financial asset. As a result, the received data, namely several last lines, are displayed in the main application window using methods "write" and "subheader" of the library "Streamlit".

```
#getting the data
def load_data(ticker):
    df = web.DataReader(ticker,
                        data_source='yahoo',
                        start = start_date.strftime("%Y-%m-%d"),
                        end = end_date.strftime("%Y-%m-%d"))
    df.reset_index(inplace=True)
    return df

#table
data = load_data(selected_stock)
st.subheader('Raw data')
st.write(data.tail())
```

Figure 10. Loading data

The first method of analyzing the value of an asset that should be implemented is to calculate and display two moving averages on a chart. They will make it possible to trace trends in stock prices over a selected period of time. To implement this, it is necessary to sift out all unnecessary data obtained during the previous step and leave only the final instrument price as of the close of the next trading day. After that, having rearranged indexes we can call function "ewm" which will return "smoothed" data arrays to appropriate variables "exp1" and "exp2". In the framework of our task, the frames of 20 and 50 days length, respectively, were chosen for smoothing.

As a result, using the methods of the "Plotly" module, it is necessary to display all the information obtained to a given moment in the main window of the program. For the convenience of viewing the obtained data, let's add the possibility of scaling through the slider, and the graphs themselves will be displayed in different colors by default.

```
#Figuring the  ema
df = data[['Close']]
df.reset_index(level=0, inplace=True)
df.columns=['ds','y']

exp1 = df.y.ewm(span=20, adjust=False).mean()
exp2 = df.y.ewm(span=50, adjust=False).mean()

#draw a plot
fig = go.Figure()
#fig.add_trace(go.Scatter(x=data['Date'], y=data['Open'], name="stock_open"))
fig.add_trace(go.Scatter(x=data['Date'], y=data['Close'], name="stock_close"))
fig.add_trace(go.Scatter(x=data['Date'], y=exp1, name='EMA 20 Day'))
fig.add_trace(go.Scatter(x=data['Date'], y=exp2, name='EMA 50 Day'))
fig.layout.update(title_text='Time Series data with Rangeslider', xaxis_rangeslider_visible=True)
st.plotly_chart(fig)
```

Figure 11. Moving averages

Then, depending on the method selected by the user, the forecast information will be displayed. First of all, let's consider a long-term forecast using the Prophet library. If the condition described in the side menu is met, a spinner load indicator is displayed while the forecast data is being calculated.

Similar to calculation of moving averages, the first step is to prepare the data for analysis and present it in the form required by the module. Having obtained a set of prices for the specified period as a result of sifting, the next step is to create an object of "prophet" class, which will store all model parameters and source data sent to the application. Next, the variable "Future" is created, which stores retrospective information, as well as empty memory sections for prediction, which are filled at the next step. As a result, after the calculation we get a ready set of prices together with the forecast totals in the "forecast" variable.

The last step is to visualize the forecast with the integrated plotting tool based on the Plotly framework.

```
# Predict forecast with Prophet.
st.subheader('Forecast data')
if method == 'Additive regression model':
    with st.spinner('Please wait...'):
        #data opt
        df_train = data[['Date','Close']]
        df_train = df_train.rename(columns={"Date": "ds", "Close": "y"})

        #make forecast
        m = Prophet()
        m.fit(df_train)
        future = m.make_future_dataframe(periods=period)
        forecast = m.predict(future)

        # Show and plot forecast
        st.write(forecast.tail())

        # Show forecast graph
        st.write(f'Forecast plot for {n_years} years')
        fig1 = plot_plotly(m, forecast)
        st.plotly_chart(fig1)
```

Figure 12. Forecast with Prophet

The next part of the application's source code is the implementation of a short-term stock price prediction function using a neural network. This fragment is executed by selecting the corresponding item in the side menu.

After filtering the input retrospective information about the stock prices, the set of input prices is scaled in the range from zero to one. This is conditioned by the area of determination of the activation function in the neuron, which in our case is a sigmoid. Then the modified data set is split into several training sets, which are two arrays. The first one stores input prices divided into segments of sixty days (x_train), and the second one stores all prices starting from day six (y_train).

```
#Predic forecast with LSTM
elif method == 'Neural network':
    with st.spinner('Please waitl...'):
        #filter data
        datas = data.filter(['Close'])
        dataset = datas.values
        training_data_len = len(dataset)

        #scale data
        scaler = MinMaxScaler(feature_range=(0,1))
        scaled_data = scaler.fit_transform(dataset)

        #create training data set
        train_data = scaled_data[0:training_data_len, :]
        x_train = []
        y_train = []

        for i in range(60, len(train_data)):
            x_train.append(train_data[i-60:i, 0])
            y_train.append(train_data[i,0])

        x_train, y_train = np.array(x_train),np.array(y_train)
        x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],1))
```

Figure 13. Formation of a training set

The next part of this block describes how to create a neural network architecture. "Sequential()" is a neural network architecture in which each sequential layer has one input and one output vector. Next, we add two LSTM layers to the model. The main difference between them is the fact that the last state of the "path" returns to the output of the layer. This is followed by one regular layer and an output layer.

Then it is necessary to assemble the model, this is done using the method "complile". The parameters of this method indicate that the accuracy is calculated by the root-mean-square error method, and the training method is the gradient descent method.

The number of training iterations and epochs in this implementation will be equal to one. This is due to the need for fast operation of the program.

After training, the script loads the data for the last sixty days into the model and makes a forecast, with the result displayed in the main application window.

```
#build model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences= False))
model.add(Dense(25))
model.add(Dense(1))

#compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

#Train the model
model.fit(x_train,y_train, batch_size=1, epochs=1)

new_df = data.filter(['Close'])
last_60_days = new_df[-60:].values
last_60_days_scaled = scaler.transform(last_60_days)
X_test = []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
pred_price = model.predict(X_test)
pred_price = scaler.inverse_transform(pred_price)
st.metric(label="Forecast for the next day", value=pred_price)
```

Figure 14: Model training and prediction

5. Justification of the effectiveness of the functioning of project

5.1. Selection and description of the calculation methodology

The list of key performance indicators directly depends on the set of goals set for the application being developed, as well as the list of criteria defined before starting the work on designing the application. Referring to the first point of this report, we can formulate a general list of indicators that will reflect the degree of success of the task.

The first criterion is the degree of relevance, reliability and completeness of information about financial assets and the state of affairs on global exchanges. Rapid development of the Internet and communication technologies has considerably strengthened the interdependence of both individual instruments represented on the market and the whole sectors of economy. Any participant on the exchange should be able to get a quick access to any indicators and metrics for making prompt and correct decision. The methodology of application effectiveness assessment within the framework of this criterion can be defined as a manual check of the data acquisition subsystem for completeness, reliability and expandability. It is necessary to check the system not only for the correctness and failure of queries to the general database of financial assets, but also for the speed of response of the used portal to the dynamics of changes in indicators.

The second evaluation criterion is the application speed. In modern trading it is extremely important to receive up-to-date information, because instrument prices may change significantly in a few minutes, and for some securities with high volatility even seconds will play a major role. When measuring the running time of a given application, it is worth dividing the total running time of the program into several separate indicators. This can be due to the fact that one of the main tasks we face in the design of this system is the subsequent integration of the solution into the portal infrastructure

"Yandex.Investments. Due to the fact that in this scenario the module is loaded as a whole only once (when opening the page), and individual processes of forecasting and data acquisition can be launched separately, the estimation of the program operation speed will be performed according to the following metrics:

- Retrospective data loading time
- Forecasting time with "Prophet"
- Forecasting time with a neural network
- Application full load time

The third indicator will be accuracy of share value forecast. There are several well-known methodologies for calculating this indicator, each of them is used depending on the area of calculations.

In our problem we will use the Root Mean Square Error (RMSE) method. This method is widely used in economics, especially in sales and finance [8].

$$RMSE = \sqrt{\frac{\sum (Y_{\&} - YS\&)^{*}}{n}}$$

Whe
re,

Yt - Forecast value of the asset price

YS& - Real value of the asset price

n - number of experiments performed

The performance check algorithm will perform a long- and short-term forecast of the value of three different assets over different time periods and calculate accuracy using the method described above.

5.2. Calculation of indicators

To correctly evaluate the created application according to the first criterion described in the previous paragraph, it is necessary to refer to the service with

The tool is used to obtain data on the value of assets from world exchanges. "Yahoo finance has proved itself as a reliable tool for traders who want to adjust their workspace to their own needs. Here are the main results of efficiency analysis:

– The aggregator provides access to all the largest world financial markets, including the Moscow and St. Petersburg stock exchanges. Taking into account the target audience of the Yandex.Investments portal, this is more than enough, but the weak point of this solution is not the fastest rate of updating information on prices.

– For Russian exchanges the delay is about 10 minutes, which is not the best result for active traders, whose goal is to make a profit through speculative trading.

– The API implemented on the basis of this service has a delay of less than 0.5 seconds (provided that the user's internet speed is at least 10Mbit per second). This figure is an average value for an API of this kind, implemented by "REST" methodology.

When assessing the speed of the application, it is worth noting the direct dependence of the time spent on performing calculations and displaying the interface on the computing power of the user's computers. The measurements were performed on the system "Macbook Pro (late 2014) with "Intel Core i5 2.6 GHz" processor. The results of the measurements are given in the table

Table No. 7

Program running time

The name of the indicator tel	**Lead time, sec.**
Retro-load time speculative data	*2.1*
Forecast calculation time With Prophet	*8.11*
Forecast calculation time with the help of a neural network	*47.16*
Full load time applications	*3.32*

For the third indicator, let's calculate the error in percent, according to the method described in the previous paragraph. For clarity, it is also worth tracing the change in accuracy for all methods of forecasting at different volumes of input data. This will allow you to determine not only the degree of importance of sufficient information for training of models, but also the perspectives of the system development and ways of optimization of the system as a whole.

Table No. 8

Error values for short-term forecasts

Active	Data volume for training			Share price on the last day of the study of
	1 year	4 years	7 years	
Google	$64.74	$20.04	$17.5	$1361.17
Apple	$3.11	$0.88	$0.54	$71.06
Tesla	$9.6	$2.16	$1.45	$85.05

In the table you can see the value of the root-mean-square error in relation to the stock price on the last day of the training data, namely December 31, 2019. You can see the extent to which the amount of data to train the network affects the final result of the prediction calculation. We can conclude that, on average, a well-trained neural network has a short-term prediction error of about one percent.

Specific values may vary depending on the volatility of the selected asset and the general state of the market.

5.3. Directions improvement

Possible areas of improvement of the developed application can be analyzed in two directions: user functionality and efficiency of analytical tools.

In terms of user capabilities, the main focus of this application is the following list of features:

– Ability to download retrospective and analytical data from the portal

– Adding a list of favorite financial assets for easier and faster customer navigation

– Possibility to view several financial instruments in one window in parallel

– Self-updating the list of available assets

As to effectiveness of forecasting and analytical means - despite a relatively low percentage of error, the use of neural networks as a full-fledged exchange assistant is only gaining popularity. This forecasting method has its disadvantages, the main of which are incorrect work under stressful market situations and rather low speed of learning and, as a consequence, low work of the script. The most actual vector of neural networks development in financial industry is their integration with other reliable metrics into common consulting information system.

Conclusion

As a part of this graduate qualification work we have developed an application for analyzing stock prices using neural network and Python programming language. The key features of web-application development were studied, several variants of solving the task were considered and the required functionality was implemented using the most suitable and up-to-date programming and data analysis tools.

A neural network based on LSTM-architecture was used as the main analytical tool for stock price forecasting, the efficiency of using this technology in the field of trading was calculated, and conclusions were made for improvement and further development of this kind of machine learning.

The list of literature used

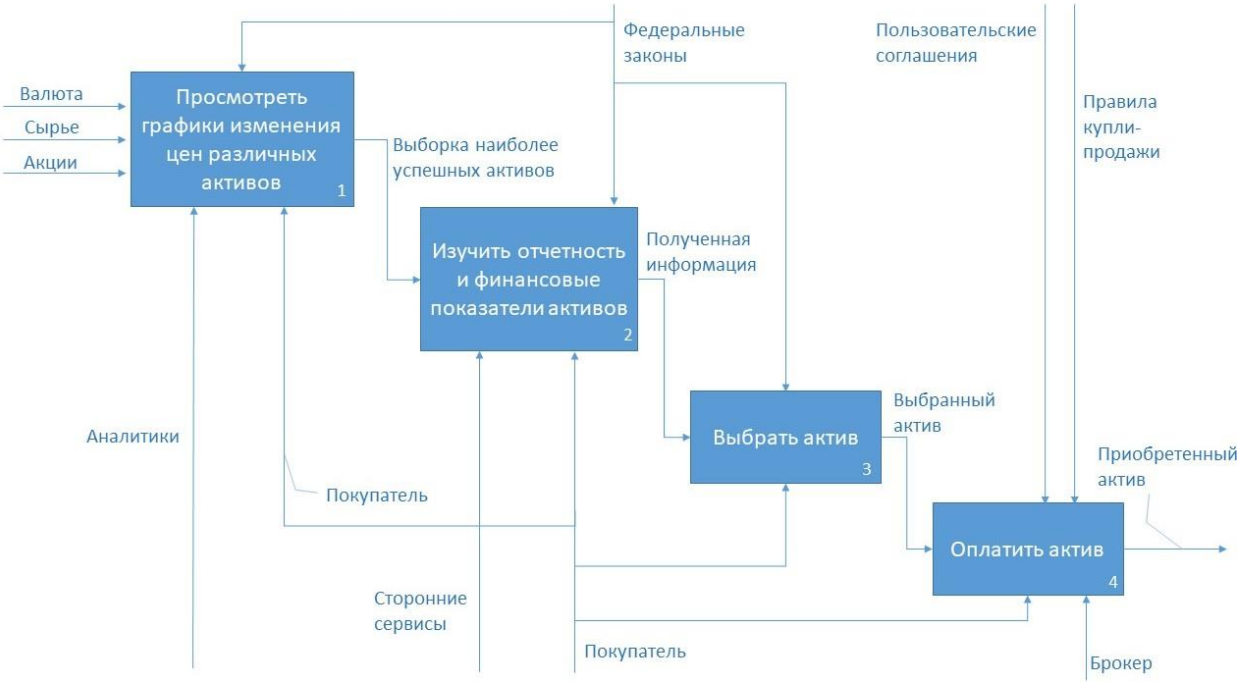
1. Artificial Neural Networks and Applications. Tutorial / Gafarov F. M., Galimyanov A. F. - Kazan, 2018.
2. VSExpress Course / P.N. Rudakov - Moscow 2020.
3. Programming in ASP.NET Ajax / Christian Wentz - 2008
4. Professional activity on the securities market. Textbook / Makarova V. A. - St. Petersburg, 2011.
5. Securities Market / Tatiannikova V.A., Razumovskaya E.A., Reshetnikova T.V. - Ekaterinburg, 2019.
6. Securities Market / Asaul A.N., Sevek V.K., - Kyzyl, 2013.
7. Modern Financial Markets / Edited by K. V. Krinichanskiy, A. A. Rubtsov, A. A. Tsyganov - Knorus 2021.
8. Statistical methods of forecasting / S.V. Arzhenovsky, I.N. Molchanov, Rostov-on-Don - 2001.
9. Financial Instruments of the Securities Market. Practical Guide for Investors / V.A. Zverev, F.A. Gudkov, S.G. Evsyukov. G., Zvereva A. V., Makeev A. V. - Intercrimpress 2007.
10. Angular for Professionals / Adam Freeman - 2019
11. Django. Developing Web Applications in Python / Paul Bissex, Jeff Forsier, Wesley Chan - 2009.
12. Python for Financial Calculations / Yves Hilpisch - 2021
13. Python Programming: An Introduction to Computer Science / John M. Zelle.
14. Long short-term memory / Horichreiter S., Schmidhuber J. - 1997.
15. About Keras / https://keras.io/about/.
16. Streamlit. Documentation / https://docs.streamlit.io/
17. Trend Changepoints / https://facebook.github.io/prophet/docs/trend_changepoints.html#automatic-changepoint-detection-in-prophet.

18. Library Facebook Prophet / https://habr.com/ru/company/ods/blog/323730/.
19. Yandex. For developers / https://yandex.ru/company/dev
20. Yandex. About the company / https://yandex.ru/company/main

Applications

Appendix 1. Interrelationships of functional calculations

Appendix 2. Interconnections of functional calculations (decomposition)



Appendix 3: Source Code

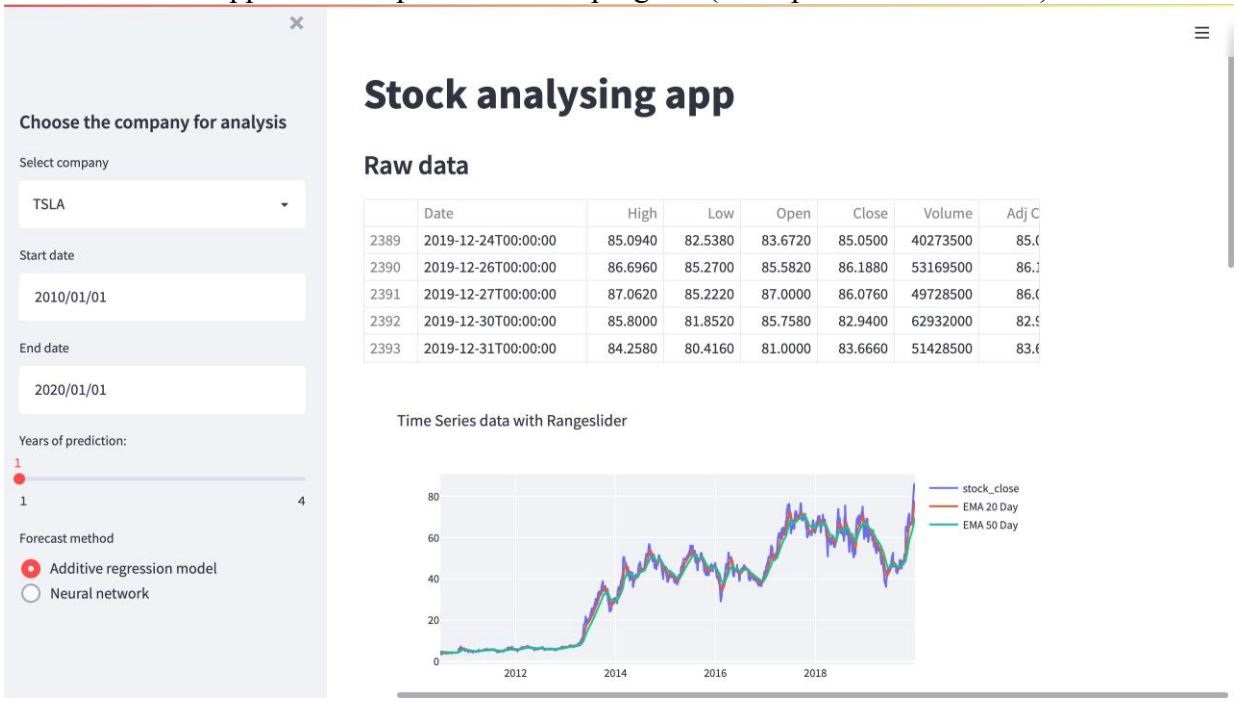
```
import streamlit as st
import datetime

import pandas_datareader as web
import pandas as pd
```

```
from plotly import graph_objs as go

from fbprophet import Prophet
from fbprophet.plot import plot_plotly

from sklearn.preprocessing import MinMaxScaler
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM

#---defining the app
st.title('Stock analysing app')

#---side bar set-up
st.sidebar.header('Choose the company for analysis')

#company choose
stocks = ('GOOG', 'AAPL', 'MSFT', 'GME')
selected_stock = st.sidebar.selectbox('Select company', stocks)

#date period choose
start_date = st.sidebar.date_input("Start date", datetime.date(2012, 1, 1))
end_date = st.sidebar.date_input("End date", datetime.date(2019, 12, 17))

#period of prediction choose
n_years = st.sidebar.slider('Years of prediction:', 1, 4)
period = n_years * 365

#Forecast method select
method = st.sidebar.radio('Forecast method',
                          ('Additive regression model',
                           'Neural network')


#getting the data
def load_data(ticker):
    df = web.DataReader(ticker,
                        data_source='yahoo',
                        start = start_date.strftime("%Y-%m-%d"),
                        end = end_date.strftime("%Y-%m-%d"))
    df.reset_index(inplace=True)
    return df

#table
data = load_data(selected_stock)
st.subheader('Raw data')
st.write(data.tail())

#Figuring the ema
df = data['Close']]
df.reset_index(level=0, inplace=True)
df.columns=['ds','y']

exp1 = df.y.ewm(span=20, adjust=False).mean()
```

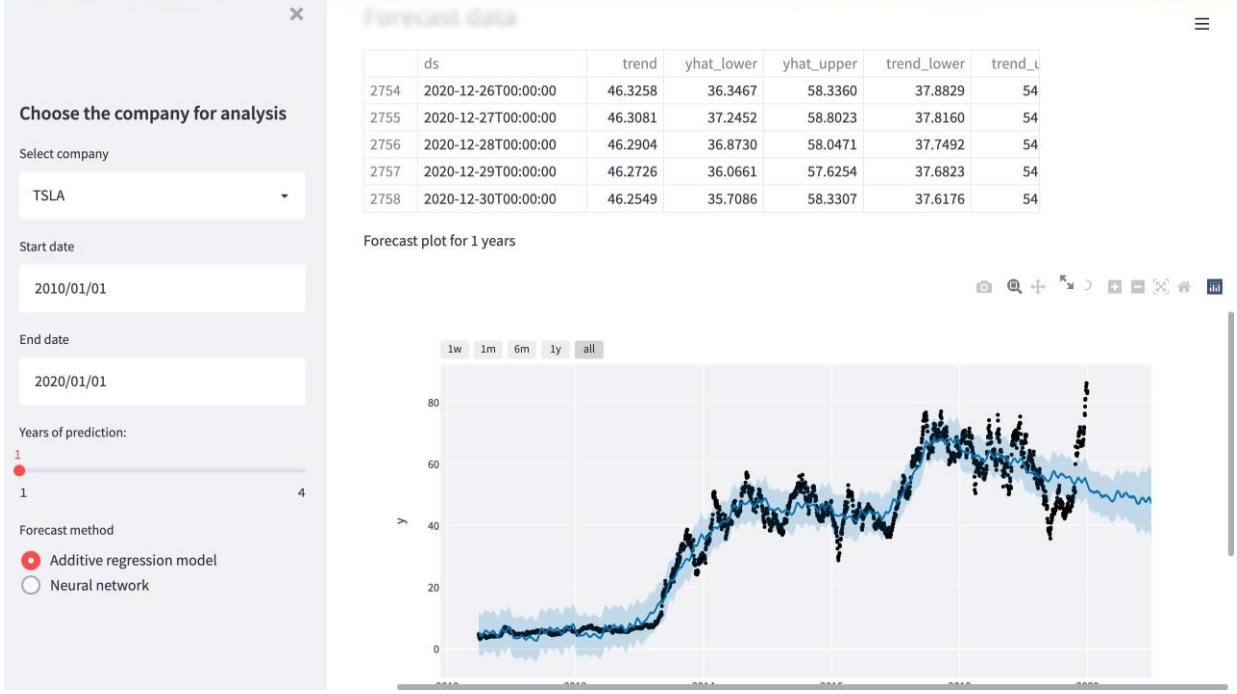
```
exp2 = df.y.ewm(span=50, adjust=False).mean()

#draw a plot
fig = go.Figure()
#fig.add_trace(go.Scatter(x=data['Date'], y=data['Open'], name="stock_open"))
fig.add_trace(go.Scatter(x=data['Date'], y=data['Close'], name="stock_close"))
fig.add_trace(go.Scatter(x=data['Date'], y=exp1, name='EMA 20 Day'))
fig.add_trace(go.Scatter(x=data['Date'], y=exp2, name='EMA 50 Day'))
fig.layout.update(title_text='Time Series data with Rangeslider', xaxis_rang-
eslider_visible=True)
st.plotly_chart(fig)

# Predict forecast with Prophet.
st.subheader('Forecast data')
if method == 'Additive regression model':
    with st.spinner('Please wait...'):
        #data opt
        df_train = data[['Date','Close']]
        df_train = df_train.rename(columns={"Date": "ds", "Close": "y"})

        #make forecast
        m = Prophet()
        m.fit(df_train)
        future = m.make_future_dataframe(periods=period)
        forecast = m.predict(future)

        # Show and plot forecast
        st.write(forecast.tail())

        # Show forecast graph
        st.write(f'Forecast plot for {n_years} years')
        fig1 = plot_plotly(m, forecast)
        st.plotly_chart(fig1)


#Predic forecast with LSTM
elif method == 'Neural network':
    with st.spinner('Please waitl...'):
        #filter data
        datas = data.filter(['Close'])
        dataset = datas.values
        training_data_len = len(dataset)

        #scale data
        scaler = MinMaxScaler(feature_range=(0,1))
        scaled_data = scaler.fit_transform(dataset)

        #create training data set
        train_data = scaled_data[0:training_data_len, :]
        x_train = []
        y_train = []

        for i in range(60, len(train_data)):
```

```
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i,0])

x_train, y_train = np.array(x_train),np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],1))

#build model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

#compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

#Train the model
model.fit(x_train,y_train, batch_size=1, epochs=1)

new_df = data.filter(['Close'])
last_60_days = new_df[-60:].values
last_60_days_scaled = scaler.transform(last_60_days)
X_test = []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
pred_price = model.predict(X_test)
pred_price = scaler.inverse_transform(pred_price)
st.metric(label="Forecast for the next day", value=pred_price)
```

Appendix 4: Operation of the program (retrospective information)



Appendix 5. Operation of the Prophet program



Appendix 6. Work of the program (Neural Network)