

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF
KAZAKHSTAN

INTERNATIONAL INFORMATION TECHNOLOGY
UNIVERSITY

FACULTY OF COMPUTER TECHNOLOGY AND CYBER
SECURITY

Almashev N.Zh.
Amanzholov Sh.N.
Kubenov T.A.

**Applied text analysis for historic data-based market incentive
forecasting**

DIPLOMA PROJECT

Major 5B070400 – Computer Science and Software Engineering

Almaty 2021

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF
KAZAKHSTAN

INTERNATIONAL INFORMATION TECHNOLOGY UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING AND INFORMATION
SECURITY

Approved

Head of Department,
cand. of tech. sc., assoc. professor
M.T. Ipalakova

«_____»_____2021

DIPLOMA PAPER

Applied text analysis for historic data-based market incentive forecasting

Major 5B070400 – Computer Science and Software Engineering

Done by:

Almashev N.Zh.

«30» May 2021



(signature)

Amanzholov Sh.N.

«30» May 2021



(signature)

Kubenov T.A.

«30» May 2021



(signature)

Research advisor:

Dauletbek Y.T.

«30» May 2021



(signature)

Reviewer:

Yunussov R.

«31» May 2021



(signature)

Almaty 2021

International Information Technology University
Faculty of Computer Technology and Cybersecurity
Department of Computer Engineering and Information Security
Major 5B070400 – Computer Science and Software Engineering

Diploma Work or Project Assignment

Students

Almashev N.Zh., Amanzholov Sh.N., Kubenov T.A.

Diploma work (project) topic

Applied text analysis for historic data-based market incentive forecasting

Approved by IITU order № 76-C dated «15» December 2020

Diploma work (project) submission date «01» June 2021

Diploma work (project) initial data

Stock market forecasting system based on text analysis using machine learning algorithms

Details of computations and explanations (list of issues due to be addressed)

- domain analysis;
- selection of machine learning model;
- website development;
- economic rational;
- system testing and conclusion.

CD containing the digital version of diploma paper and attachments Provided

- diploma project documentation ;
- diploma project presentation ;
- web application source code .

Consultations on diploma work (project) (with related project chapters named)

Consultant	Name	Signature, date	
		Assignment given	Assignment received
Consultant on Economic effectiveness of the project	Associate professor Berdykulova G.M.	27.04.2021	13.05.2021
English language consultant	Associate professor Kabdrgalinova S.	04.05.2021	21.05.2021
Compliance monitor	Shyntore G.A.	30.05.2021	30.05.2021

Date «30» May 2021

Research advisor



(signature)

Assignment received by




(signature)



Diploma project writing schedule

Almashev N.Zh., Amanzholov Sh.N., Kubenov T.A.

Title: Applied text analysis for historic data-based market incentive forecasting

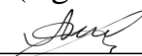
№	Assignment	Submission date
1.	Creation of the graduation paper writing schedule	November, 30
2.	Collection, study, processing, analyzing and generalizing data	November–December
3.	Drafting and submission to the Research advisor (Introduction, Chapter 1, Chapter 2, Chapter 3, Conclusion)	January– February
4.	Submission of the chapter «Economic effectiveness of the project» to the consultant	February–March
5.	Revision of the graduation paper with due consideration of the advisor's comments	March – April
6.	Submission of completed graduation paper to the Research advisor	April 15
7.	Pre-defence	April
8.	Submission of the completed diploma paper to the English language consultant	April 20 – May 3
9.	Submission of the diploma paper to the compliance monitor	April 30 – May 15
10.	Submission of the diploma paper for the plagiarism check-up	May 3 – May 24
11.	Submission to the reviewer for approval	May 3 – May 24
12.	Diploma work (project) defense	May 24 – June 19

Student: Almashev N.Zh.




(signature)

Student: Amanzholov Sh.N.



(signature)

Student: Kubenov T.A.



(signature)

Research advisor: Dauletbek Y.T.



(signature)

Date «30» May 2021

АНДАТПА

Қор нарығындағы өзгерістерді болжау - қызықты және зерттеуді қажет ететін мәселелердің бірі. Нарыққа үлкен әсер ететін факторлардың бірі - жаңалықтар мазмұны болып табылады. Қор нарығына байланысты барлық мәлімет инвесторлар үшін білім қазынасы болып саналады. Бірақ бұл ақпараттардың көп бөлігі құрылымдалмаған, сондықтан оны өңдеу көп уақыт пен адами ресурстар қажет етеді. Жаңалықтардың қор нарығындағы өзгерістерге әсерін талдай отырып, компаниялар дәлірек болжамдар жасай алады, нәтижесінде тиімді инвестицияларға мүмкіндік ашылады. Бұл зерттеулік жұмыс қор нарығындағы болашақ тенденцияларды болжаудың тиімді моделін құруға және болжамдардың дәлдігін арттыруға бағытталған. Бүгінгі күні жаңалықтардың қор нарығын болжауға әсерін есепке алу үшін түрлі прототиптер әзірленді. Бұл жұмыста машиналық оқыту арқылы мәтіндік анализ жасаудың бірнеше әдістері енгізілді, әртүрлі алгоритмдер сыналды, қаржылық жаңалықтар мен қор нарығының тарихи бағаларының өзгерістерін анализ жасауға негізделген болжау модельдері сипатталды, мәліметтер жиынтығы нақты нәтижелер алу үшін терең оқытудың әртүрлі әдістерімен оңтайландырылды.

Түйін сөздер: ҚОР НАРЫҒЫ, БОЛЖАМДАР, ЖАҢАЛЫҚТАР, ТАЛДАУ, ТЕРЕҢ БІЛІМ.

АННОТАЦИЯ

Прогноз изменений в фондовом рынке - интересная исследовательская задача, которую необходимо изучить. Одним из наиболее влиятельных факторов, оказывающих большое влияние на рынок, является содержание новостей. Большой объем данных, генерируемых фондовым рынком, считается сокровищницей знаний для инвесторов. Но большая часть этой информации не структурирована, поэтому для ее обработки требуется много времени и человеческих ресурсов. Анализируя влияние новостей на поведение фондового рынка, компании могут делать более точные прогнозы и, как следствие, более прибыльные инвестиции. Это исследование направлено на построение эффективной модели для прогнозирования будущих тенденции на фондовом рынке с минимальными отклонениями и повышение точности прогнозов. На сегодняшний день разработаны различные прототипы для учета влияния новостей на прогнозирование фондового рынка. В данной работе было введено несколько методов машинного обучения текстового анализа, протестированы различные алгоритмы, описаны прогнозные модели, основанные на анализе поведения финансовых новостей и исторических цен фондового рынка, набор данных оптимизирован различными методами глубокого обучения для получения более точных результатов.

Ключевые слова: ФОНДОВЫЙ РЫНОК, ПРОГНОЗЫ, НОВОСТИ, АНАЛИЗ, ГЛУБОКОЕ ОБУЧЕНИЕ

ABSTRACT

Stock market forecasting is an alluring research problem to be studied. One of the most influential factors that have a big impact on the market is the news content. The large amount of data generated by the stock market is considered the treasure trove of knowledge for investors. But most of this information is unstructured and, therefore, requires a lot of time and human resources to process it. Analyzing the influence of news to the stock market behavior, companies can make more accurate predictions and, as a result, more profitable investments. This study aims to build an effective model for predicting future trends in the stock market with low error rate and improve accuracy of forecast. To date, different models have been created to account for the effect of news on stock showcase estimating. In this paper, several machine learning techniques of the text analysis have been introduced and different algorithms are tested, forecast models are described based on the sentiment analysis of financial news and historical stock market prices; the data set is optimized by various deep learning techniques to get more precise results.

Keywords: STOCK MARKET, FORECASTING, NEWS, ANALYSIS, DEEP LEARNING

CONTENT

	INTRODUCTION	11
1	THEORETICAL PART	13
1.1	Description of subject area	13
1.2	Analysis of existing projects	14
1.3	Text analysis tools	15
1.4	Time series forecasting	16
1.4.1	Determination of financial time series	18
1.4.2	Components of financial time series	19
1.5	Collecting data	22
1.5.1	Data corpus	22
1.5.2	Twitter and RSS news feeds. Their impact on stock market prediction	23
1.6	Feature processing	24
2	PRACTICAL PART. ARTIFICIAL INTELLIGENCE BASED FORECASTING MODELS	28
2.1	The architecture of neural networks	30
2.2	Feedforward neural networks	30
2.3	Recurrent neural networks	36
2.4	Decision trees	37
2.4.1	The decision-making forest	37
2.4.2	The decision-making jungle	38
2.5	The ARIMA model	39
2.6	The SVM model	41
2.7	Review of the LSTM model	45
2.7.1	Benefits of LSTM	46
2.8	Methods for evaluating algorithms	46
3	PRACTICAL IMPLEMENTATION	51

3.1	Using news sentiments	52
3.2	The dataset description	53
3.3	Twitter news data	54
3.4	Data gathering	55
3.5	Data cleaning and preprocessing	57
3.6	Data modification	57
3.7	News sentiments analysis	58
3.8	Modeling	59
3.8.1	Support Vector Machine (SVM)	59
3.8.2	Random Forest	61
3.8.3	ARIMA	62
3.8.4	SARIMAX	63
3.8.5	Multilayer perceptron (MLP)	64
3.8.6	Long-Short Term Memory (LSTM)	65
3.8.7	LSTM with news sentiments	66
4	ECONOMIC RATIONALE OF THE PROJECT	69
4.1	Business idea	69
4.2	Product description	70
4.3	Market research	70
4.4	Market strategy	71
4.5	Financial analysis	71
4.6	Economic effectiveness	74
	CONCLUSION	75
	REFERENCES	76
	APPENDIX	77

INTRODUCTION

Companies receive text data continuously (emails, chats, product reviews, and etc.), and all of this unstructured data contains valuable information that people can use to make decisions about your products or services. The problem, however, is that manually analyzing text data takes a very long time. Fortunately, there are advanced tools, such as text analysis with Python, that can help you transform your data into meaningful ideas quickly and at scale.

Nevertheless, today in the practice of stock market analysis, there are many works that use high-quality data, such as text and news information that can be extracted from news articles, blogs, analytical reports, and social surveys. Usually, there is no access to internal corporate information, so to analyze the specifics of the company's activities, activities; we consider information from regularly published financial reports, analytical forecasts, and news articles that reflect some of the features of the company's activities. The news analysis is able to identify specific company news that the traditional financial analysis cannot identify. The reason for this is that decision-makers are emotional, and, therefore, subjective. Moreover, their reaction to the same events can be quite different (behavioral economics). Analyzing textual information allows us to form a more complete form of the more complete judgment about the financial state of the market participant, so the construction of the effective forecast is possible only with the combined use of both technical and fundamental analysis, and the primarily emotional analysis. Manual tracking of the new news stream and its subsequent analysis is very difficult for analytics, therefore, the use of intellectual analysis methods and the creation of tools to support them is a popular task.

The paper is devoted to the review of the current state of the research in the field of forecasting securities market quotations based on methods and tools of the intellectual analysis of the text and news articles. Our paper approaches to analyze respected sentiments in the global news field and track patterns and utilize them for predicting the market behavior. Groups of interest in our work are stock market investors and finance technology analytics.

There is an unpredictable market. People usually rely on analysts to analyze the market for them. For obtaining broader inside into the market we propose to automate some work performed by analysts and perform the advanced technical analysis of markets for everyone. Our forecasts are supported by the text analysis validated mainly on historic data.

Studying the existing works on this topic, we can conclude that artificial intelligence algorithms are effective tools for solving the problem of forecasting, but today there is no clear answer to the question of whether there is the universal

algorithm for predicting the price dynamics of stocks. In order to partially solve this problem, the authors of the study set out to conduct the comparative analysis of various algorithms based on machine learning and identify the best one in the conditions of the stock market.

To achieve this goal, the following tasks were formulated:

- study the theoretical and methodological foundations of time series analysis;
- describe the problems that may arise when forecasting financial time series;
- study the specifics of the data used;
- learn how forecasting algorithms work;
- build and train predictive models based on the studied algorithms;
- compare the performance of models;
- develop recommendations for modifying the model for a better algorithm, according to estimated indicators, in order to increase its effectiveness in the context of the specifics of the stock market.

1 THEORETICAL PART

1.1 The description of the subject area

When analysts, investors, and institutional traders evaluate current stock prices, news plays an important role in the valuation process. In fact, the news contain information about the main principles of the company and high-quality information that affects the expectations of market participants. From the theoretical point of view, the effective valuation of the firm should reflect the present value of the firm's expected future cash flows. Expectations for the firm's development largely depend on the set of information available to investors. The information set consists of news that contains both qualitative and quantitative information from a variety of sources, such as corporate disclosures, third-party news articles, and analytics reports. If the financial news contains new information leading to the adjustment of expectations regarding the firm's cash flows or the investor's discount rates, this affects the stock's return. In news, not only do financial indicators have a significant impact on the stock price, but also quality text components affect stock prices when they contain new information.

Thanks to the improved information mediation, the amount of information available has increased dramatically in recent decades. As it becomes increasingly difficult for investors to track and account for all available information, automated classification of the most important information is becoming increasingly relevant.

All investors usually have the inescapable need to find the best way to predict the future behavior of stock prices, this will help determine the best time to buy or sell shares in order to maximize the return on their investments. The stock market can be traded physically or electronically. When an investor buys the company's shares it means that this investor becomes the owner of the company according to the ownership share. This gives shareholders rights to the company's dividends.

The financial data of the stock market is complex, which makes it difficult to predict the behavior of the stock market. Data mining can be used to analyze a huge and complex volume of financial data, leading to better results in predicting stock market behavior. Using data mining techniques to analyze the stock market is an extensive area of research, as it is important for the economy, as higher prices lead to the increase in the country's income.

However, research in the field of automated classification of text-based financial news is in its infancy. Despite numerous attempts and applications, the accuracy of forecasts of the direction of stock prices after the release of corporate financial news rarely exceeded 58% - a level of accuracy slightly higher than the probability of random guessing (50%), leaving room for significant improvements.

Automatic classification of text news includes intelligent text analysis that translates unstructured information into a machine-readable format and mainly uses machine learning methods for classification. Well-known, the development of suitable textual representations is still part of ongoing research. In essence, text representation methods refer to the way text is processed. An example is the word set model, which treats text as a set of unordered individual words. In this case, the 'single words' object type is a text representation. More complex types of functions are related to word combinations. It is clear that not all words are needed to reflect a given text; text mining searches for the most appropriate functions to represent the text.

The existing literature on financial text mining usually relies on very simple textual representations, such as the aforementioned word set models. Next, a list of words used for the text representation is created, either based on dictionaries, or extracted from the message body based on the actual occurrences of words. Despite well-studied approaches to selecting the most relevant words or phrases based on exogenous feedback; existing works often rely on frequency statistics of the message body, such as the TF-IDF information retrieval tool, or even more simply, the minimum number of phrases[1][2].

1.2 The analysis of existing projects

During the search for similar projects many foreign analogues were found, and there is no analogue in the CIS. And among many foreign analogues we have collected the best analogues, which are presented in Table 1.1.

The first 4 foreign analogues are Motley Fool, Bloomberg, Morningstar, Seekingalpha. These analogues are irreplaceable stock exchange websites that are sure to provide you with reliable and factual data. These sites are also known for serving both individual investors and large hedge funds, thanks to expert comments and advice from professional analysts with real experience. The next 2 analogues already do without expert advice, and they themselves put forward theories thanks to artificial intelligence. In the AISTockFinder you will receive electronic alerts about the impending forecast.

There is no analogues in the region of Kazakhstan, so this project will be unique for our country.

Table 1.1 - Existing projects

Name	The Motley Fool	Bloomberg	Morningstar	Seeking Alpha	UMPINDEX	AISTockFinder
Head Quarters	Alexandria, Virginia;, the U.S.	the USA	Chicago, Illinois, the U.S.	New York, NY, the US	the USA	the USA

Table 1.1 continued

Price	19\$/month	2,200\$/month	199\$/annually	29,99\$/month	60\$/year	49\$/month
Pros	1. Provides access to expert advice from professionals that have real experience. 2. Allows you to construct real and hypothetical equity portfolios.	1. Provides useful insights for professional traders and analysts. 2. Bloomberg Terminal maximizes speed and data connectivity. 3. Their free TV keeps you updated with the latest market developments	1. Find new investment ideas using the most popular screeners for research. 2. Evaluate investment ideas by unveiling buying opportunities across sectors thanks to the unlocked data and ratings.	1. Ideal for those that are not versed with technical aspects of stock investing. 2. Delivers top-shelf content for free	1. UMPI's algorithm can track stock market trends that cannot be seen by humans, providing better awareness when analyzing the stock market.	1. Extracting data from the internet using Data mining. 2. Artificial intelligence algorithm analyses millions of data points to extract patterns.
Web Version	Yes	Yes	Yes	Yes	Yes	Yes

1.3 Text analysis tools

Since text analysis strategies are utilized basically in machine learning, a programming language with a wealthy set of logical and computational libraries is required. The Python language is the most excellent suited for this part, which incorporates a set of effective libraries such as Scikit-Learn, NLTK, Gensim, spaCy, NetworkX and Yellowbrick[3].

Scikit-Learn is an expansion for the SciPy (Logical Python) library that gives an application interface (API) for generalized machine learning. Python-based expansion with back for high-performance C libraries such as LAPACK, LibSVM, Boost, and others Scikit-Learn combines tall execution with simple-to-use strategies for analyzing little and medium-sized information sets. This expansion, which is conveyed in open source and permits commercial use, gives a single interface for numerous relapse, classification, clustering, and dimensionality diminishment models, as well as utilities for cross-checking and designing hyperparameters.

NLTK (Natural Language Tool-Kit - a package of tools for natural language processing) is a "batteries included" asset for common language preparation composed in Python by specialists in scholarly circles. Initially made as a device for instructing normal language handling, this bundle contains corpora, lexical assets, language structures, language handling calculations, and pre-trained models, permitting Python programmers to rapidly begin preparing text information in several natural languages.

Gensim is a robust, effective and easy-to-use library with the most objective of unsupervised semantic content modeling. Initially made to discover likenesses in archives, this library presently gives theme modeling for Inactive Semantic Examination methods and incorporates other unsupervised machine learning libraries such as word2vec.

spaCy actualizes high-quality common language handling, wrapping cutting edge scholarly calculations in a basic and user-friendly API. In specific, space permits you to perform pre-processing of content in arrangement for profound learning and can be utilized to form frameworks for extricating data or analyzing normal dialect on huge volumes of content.

NetworkX could be a comprehensive chart investigation bundle that makes a difference you make, organize, analyze, and control complex network structures. In spite of the reality that it isn't a library of machine learning or content examination, the utilization of chart information structures permits you to encode complex connections that graph algorithms are able to analyze and discover semantic highlights, and so, is a critical device for content examination.

Yellowbrick is a set of visual diagnostics apparatuses for analyzing and deciphering machine learning. Complementing the Scikit-Learn API, the Yellow brick bundle gives basic and natural visual apparatuses for selecting features, modeling and designing hyperparameters, and overseeing the method of selecting models that most successfully portray content information.

1.4 Time series forecasting

The task of time series forecasting is in favour of analysts, economists and brokers, since hidden information in the stock market can bring high profits to interested participants, as well as provide the answer to a large number of controversial questions about the completeness and accessibility of available information.

There are a large number of analysis tools that allow making forecasts. Today, financial bots and machine learning models are among the most popular tools for identifying short-term patterns. Analysts are also interested in finding long-term dependencies in order to provide scientific justification for various economic phenomena. The difficulty of forecasting the stock market is that in order to implement a high-quality forecast, it is necessary to have complete and reliable information, as well as to be able to correctly interpret it and apply it to forecasting methods.

There are two main approaches to stock market analysis: the fundamental and technical analysis. The fundamental analysis is effective for the implementation of

long-term strategies, but it also requires high time costs, since the analysis requires an individual approach for the selected companies (the assessment of the macro environment, calculation of revenue indicators, profit, and etc.). The effectiveness of the technical analysis can be demonstrated in the implementation of short-term trading. Often, the reaction and price movement occurs earlier than the fundamental reaction, which gives technical analysts a time advantage by focusing on prices. The technical analysis also helps to reduce risks in making medium-term strategic decisions, since this approach does not require an individual study of the specifics of individual companies. To increase the probability of obtaining the high accuracy of the forecast, it is recommended to predict the logical returns of the asset, since stock prices are inherently non-stationary processes[4][5].

The task of forecasting the financial time series is defined as:

- *Classification.* The prediction of the qualitative response for a certain observation can be attributed to the certain category or class.
- *Regression.* Investigation of the influence of one group of continuous random variables on another.

In mathematical terms, the time series prediction problem is the prediction of the real scalar series, i.e. $y_t \in R$.

The essence of the task is to find such a function f_T , which will depend on all known information at the time of forecasting. At the input, this function takes all the values of the series (y_1, y_T) , and this function also has parameter h , which denotes the number of predicted steps ($h \in [1, H]$):

$$y_{T+h} \approx f_T(y_T, \dots, y_1, h) \equiv \hat{y}_{+h} | T, \quad (1.1)$$

where $h \in \{1, 2, \dots, H\}$, H – forecast horizon.

1.4.1 The determination of financial time series

Time series: $y_1, \dots, y_T, y_t \in R$ – is a certain feature, the value of which is tracked at constant time intervals. The measurements of the feature occur over time, and the same amount of time passes between different measurements, which is a key feature of time series, since in the case when the intervals between reports are different, this process is random, therefore, it is disordered over time and the target variable does not depend on historical data. The methods of analyzing random processes and time series differ significantly, so it is important to pay attention to this feature[6].

As shown in Figure 1.1 within the framework of the stock market, time series are usually based on prices or their dynamics, such time series are called financial.



Figure 1.1 - Time series of KASE index

1.4.2 Components of financial time series

Time series are usually represented as models of the sum/product of certain time components:

- Additive model: $y_t = T_t + S_t + C_t + \varepsilon_t$;
- Multiplicative model: $y_t = T_t * S_t * C_t * \varepsilon_t$;

$T_t, S_t, C_t, \varepsilon_t$ - components of time series, where $(t = 1, 2, \dots, n)$.

A trend (T_t) is a smooth long-term change in the level of series, or an average movement, near which the process fluctuates. A trend can also be interpreted as the series of consecutive highs and lows. It has three conditional phases:

- origin;
- formation;
- termination.

According to the theory of the technical analysis, there are three types of trends:

- primary;
- intermediate;
- short term.

In an uptrend, the market movement is a sequence of increasing highs and lows of the asset price over time.

Figure 1.2 shows that in the period from mid-March to mid-April 2021, the KASE index grew, therefore, it can be argued that during this period, the market observed the primary trend.



Figure 1.2 - Dynamics of KASE index movement

In the intermediate or secondary trend, the market movement is a sequence of decreasing highs and lows over time. An example of the intermediate trend can be demonstrated on the shares of “JSC Kazakhtelecom”. Figure 1.3 shows that from mid-July 2020 to the start of September 2020, the price declined, which indicates that the market was experiencing the secondary trend during this period.



Figure 1.3 - Dynamics of the share price movement of KZTK

In the situation where the market does not have a clear trend to increase or decrease, there is a short term trend (the price moves between the upper and lower boundaries). Within such price dynamics the market is called flat. Figure 1.4 shows that the share price of the KAZ Minerals PLC has been moving in the same price range for a long period, which indicates a prolonged consolidation in the market, and therefore, the market is flat[7].

Financial time series are subject to seasonal effects. This is easily demonstrated by the example of stocks traded on the US stock market. Share prices largely depend on the activity of market participants in certain periods of the financial year. As a rule, the increase in the activity of traders is associated with the end of the financial year. This is a clear example of the seasonal component.

Financial time series are subject to seasonal effects. This is easily demonstrated by the example of stocks traded on the US stock market. Share prices largely depend on the activity of market participants in certain periods of the financial year.



Figure 1.4 - Dynamics of the share price movement of GB_KZMS

As a rule, the increase in the activity of traders is associated with the end of the financial year. This is the clear example of the seasonal component.

Seasonality (St) is a recurring phenomenon in the data that is characterized by a change in the level of a series with a constant period. These can be certain ups or downs. In the US stock market, the most common reasons for the appearance of a seasonal component are:

January jump. The market, as a rule, grows at the beginning of the year, due to the appearance of investment funds from market participants.

Long weekends and national holidays. The stock price increases before the three-day weekend, for example, before Thanksgiving and Independence Day in the United States.

Monday's blues. It is also believed that the value of shares varies in a certain way depending on the day of the week or month.

Cyclicity (Ct) – changes in the level of the series near the average value with a variable period.

Examples of cyclic fluctuations:

The economic cycle – the rise, decline, curtailment and revival of economic activity. The duration of the economic cycle is from one to twelve years. The average cycle duration is considered to be four years.

The construction cycle has a period of fifteen to twenty years.

The business cycle – circulation of business papers. Its duration is usually from one year to four years.

Random perturbations (ε_t) – irregular changes in the time series.

When predicting financial time series, random perturbations are usually divided into removable and non-removable ones. \hat{y}_t is usually not the ideal estimate of the y_t , and this inaccuracy leads to some error. Such an error is avoidable, since we can potentially improve the accuracy of \hat{y}_t by using a more appropriate statistical method for estimating y_t . But even if it were possible to achieve such an ideal estimate of \hat{y}_t that $y_{t+h} = \hat{y}_{t+h}$, the predicted value would still contain some error. Such an error is called fatal. No matter how well we estimate \hat{y}_t we will not be able to reduce the error introduced at the expense of ε_t . The value of ε_t may include unrecorded factors that affect the target variable y_t .

1.5 Collecting data

1.5.1 The data corpus

A corpus is a collection of interrelated documents (texts) in a natural language. The enclosure can be large or small, but usually consists of tens or even hundreds of gigabytes of data in thousands of documents. Corpora can be annotated, that is, the text or documents can be labeled specifically for supervised learning algorithms (for example, for spam filters), or unannotated, making them candidates for the topic modeling and document clustering (for example, to study changes in topics hidden in messages over time)[8][9].

The corpus can be broken down into categories of documents or into individual documents. Corpus documents can vary in size, from individual tweets to entire books, but they all contain the text (and sometimes metadata) and represent a set of related topics. Paragraphs can also be split into sentences - syntactic units; a complete sentence structurally sounds like a concrete expression. Sentences are composed of words and punctuation marks - lexical units that define the general meaning, but are much more useful in combination. Finally, the words themselves are made up of syllables, phonemes, affixes and symbols, that is, units that make sense only when they are combined into words.

Very often, testing of the natural language model begins with a generic corpus. For example, there are many examples and scholarly articles that use off-the-shelf datasets such as the Brown corpus, the Wikipedia corpus, or the Cornell corpus of movie dialogues. However, the most successful language models are often highly specialized for a particular application.

Why do language models trained on a limited domain often perform better than similar models trained on a generalized corpus? Different subject areas use a different language (vocabulary, abbreviations, typical phrases, and etc.), therefore, the corpus specialized for a specific field is analyzed and modeled more accurately than the corpus with documents from different fields.

The presence of the subject corpus is of great importance for creating a successful data application based on natural language analysis. Naturally, the next question arises: how to get the dataset to create the model language? Regardless of the method of collecting data - by scraping, extracting from RSS, or through some API - constructing a corpus with source text in a form suitable for creating a data application, is not a difficult task.

1.5.2 Twitter and RSS news feeds. Their impact on the stock market prediction

Twitter may be a real-time smaller scale blogging stage that permits individuals to communicate with brief messages. A Twitter message can contain up to 140 characters in a single so-called twit message. In general clients can right away post anything to precise their interest, how they feel, or fair to provide status overhauls to friends and family individuals. For the reason of foreseeing stock showcase patterns, there's a fabulous information source which comprises numerous characteristics that are accumulated from twitter for estimation examination[10].

Twitter is open for open utilization and any tweet can be recuperated without any security impediments. Twitter joins a clean and well-documented API that enables architects to request for a specific collection of tweets utilizing certain watchwords or

based over a period of time. In this proposed work, suppositions of the stock related tweets are analyzed and based on the limit of the feelings with the estimation score expanding from +1 to -1. In the event that the whole conclusion score regard is 0.0 to 1.0 at that point the sentence might be a positive sentence. In case the complete opinion score regard is -1.0 to 0.0 at that point the sentence might be a negative sentence. Within the occasion that the whole suspicion score regard is 0.0 at that point the sentence can be an unbiased sentence. The suspicion score is related with the changes inside the sensex fetched of stock for a certain period[10].

Truly Basic Syndication (RSS) may be organized for conveying frequently changing Web substance. Numerous news-related destinations, Weblogs and other online distributors syndicate their substance as an RSS Bolster to whoever needs it. It's an XML record that encourages substance syndication. A RSS could be a solid way to have the net substance conveyed to the Web since the information is little and fast-loading, it can be utilized with administrations like cell phones or PDA's, voice sends, and e-mail. Not at all like e-mail an RSS nourish is zero upkeep, the messages will never get boycotted or sifted. With RSS, clients can partition needed data from undesirable data. RSS reports utilize self-describing and straightforward language structure. By and large, RSS news nourish contains the creator, title, and date data in addition to the connect and portrayal. On the off chance that dynamic and good news on current undertakings is discharged at that point this contains a positive effect on the stock showcase values. This is often captured by examining the RSS news feeds.

1.6 Feature processing

Feature processing is the most important part of text mining. It can be described by three preparatory stages: feature extraction, feature selection, and feature representation.

At the feature extraction stage, the features that best reflect the content of the text are defined and extracted. And one of the simplest techniques for that is bag-of-words, which describes the frequency of words appearing in the text and simply represents text in numbers. A bag-of-words is a model of natural language texts in which each document or text looks like an unordered set of words with no information about the link between them. It includes the dictionary of known words and a metric for determining the presence of well-known terms. It can be represented as a matrix, where each row corresponds to a separate document or text, and each column corresponds to a specific word. The cell at the intersection of the row and column contains the number of occurrences of the word in the corresponding document[11].

A useful generalization of the "bag-of-words" model can be the "bag-of-terms" model. The main object of the "bag-of-terms" model is a term with a single attribute with the frequency of occurrence in the text.

A term is a symbolic expression of an object of a formal model (language, system). Term: symbolic expression: $t(X_1, X_2, \dots, X_n)$, where t is the name of a term called a functor or "function letter", and X_1, X_2, \dots, X_n are terms, structured or simple.

Terms can contain free variables (parameters), the fixation of the values of which uniquely determines, in accordance with the semantic rules of the language, a certain object-value of the term with the given values of its free variables. In logical and mathematical calculus, a term is an analog of the subject or complement of a natural language. In the bag-of-terms model, any symbolic expressions of the text, including punctuation marks, can be considered as terms.

For each word from the set of the "bag of words" model, some "weight" can be specified. Thus, the text model is a set of "word-weight" pairs. In this case, weights can be assigned to words or word bases. Here are methods for determining the weight of words:

- The binary method (common notation-BI, from binary). Only the presence or absence of certain terms in the document is determined. It is used for the logical information search and automatic text categorization using the methods of neural network classifiers ART and SOM.
- The number of occurrences (words in the document). It assumes that the score is disproportionate for texts of different lengths — larger texts will receive more weight, since they contain more words;
- The frequency of occurrence of a word in the document (TF — term frequency). Frequency is calculated as the ratio of the number of occurrences of a word to the total number of words in the text. With relative simplicity, this characteristic provides an acceptable result for information search and classification methods (it assumes that the evaluation is disproportionate for texts of different lengths — long documents are underestimated, since they contain more words and the average frequency of words in the text is lower).
- The logarithm of the frequency of occurrence of the word (LOGTF). The weight of the incoming document is defined as $1 + \log(TF)$, where TF is the frequency of the term. The use of a logarithmic scale makes the model more resistant to the reassessment of texts of different volumes.[13]
- The inverse document frequency (IDF-inverse document frequency). The parameter is the inversion of the frequency with which the term occurs in the documents.

Also there are several feature extraction techniques, each of which is effective

for specific tasks:

- Dictionary-approach - to define a list of features; no objects are extracted from the corpus. Instead, individual words from the list of positive and negative words in the dictionary are used.
- N-Gram is just a sequence of n elements (sounds, syllables, letters, words or symbols) that appear in a row in some text. In practice, they often mean a series of words. A sequence of two elements is called a bigram, of three elements - a trigram.
- 2-word combinations - this feature type represents a 2-gram-based word combination extension, allowing the distance between two words to be greater than zero.
- Noun phrases - is a word or phrase that starts with a noun followed by adjectives or other determiners.

Next stage is feature selection, and this paper has described three common feature selection approaches:

- Dictionary-based: this method makes use of a pre-existing dictionary that has been manually curated by domain experts to find the most applicable terms.
- Feature selection without exogenous market feedback: rather than using a dictionary, features are extracted entirely from the message corpus. Literature uses more complex approaches in addition to very specific steps for related terms requiring a minimum occurrence. For example, selects features based on the TF IDF, where the occurrences of one word in the processed document are linked to their occurrences across the entire data set. This method, on the other hand, relies solely on endogenous information in the corpus and does not take into account exogenous input on how messages with specific features were received by the stock market.
- Feature selection employing exogenous market feedback: The feature selection should use the customer's input as an exogenous factor to pick the most appropriate features to differentiate between positive and negative messages, in addition to the endogenous data in the corpus.

There are several steps of classification shown in Figure 1.5: data set, feature processing and machine learning[12].

Following the determination of the most important features, the next stage of feature processing is feature representation, where relevant information needs to be translated into a machine-readable format. Based on this format the machine learning algorithm classifies the information, which is used to predict stock market reaction.

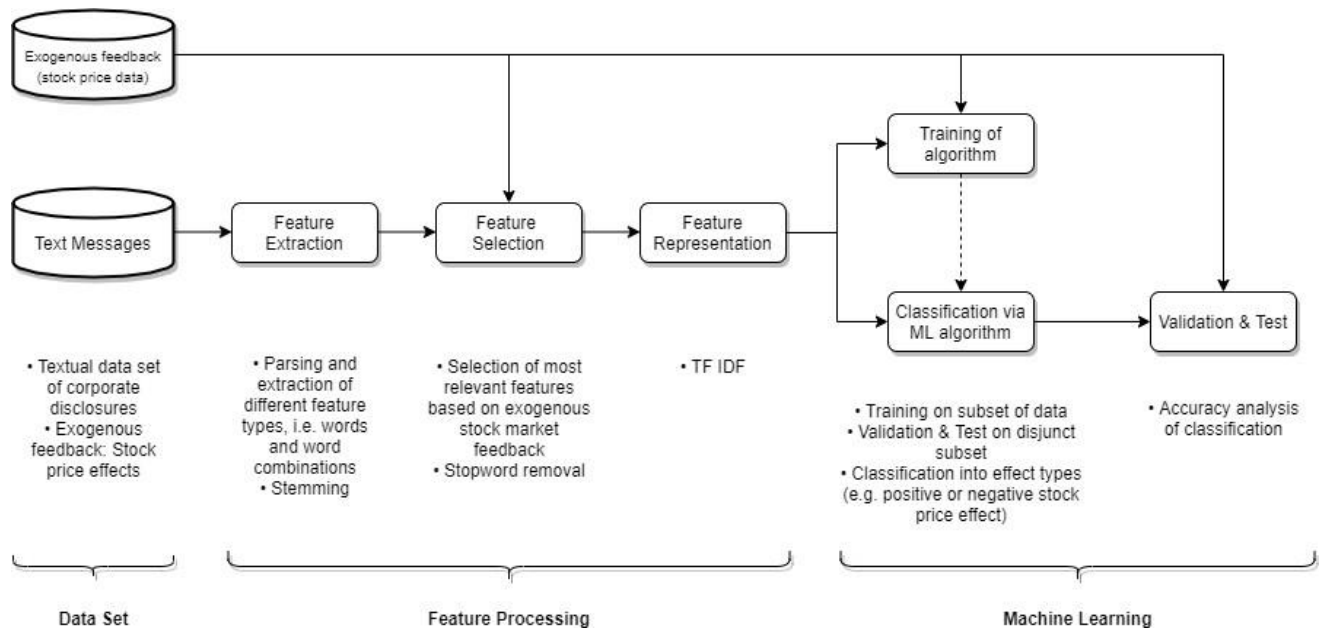


Figure 1.5 - Overview to the steps of financial news classification

2 PRACTICAL PART. ARTIFICIAL INTELLIGENCE BASED FORECASTING MODELS

Most of the methods used today to study financial markets are far from ideal and have a number of shortcomings that cannot yet be eliminated, due to the instability of the economy, as well as the imperfection of human knowledge in the field of forecasting. However, even today, with the help of artificial intelligence algorithms, it is possible to achieve high indicators of forecast accuracy.

Artificial intelligence is understood as the property of machines, computer programs and systems to perform the intellectual and creative functions of a person, independently find ways to solve problems, draw conclusions and make decisions.

Artificial intelligence is a broad field that includes knowledge about data, mathematical statistics, machine learning, including deep learning, and neural networks. Figure 2.1 shows that machine learning is a subspecies of artificial intelligence; and algorithms are divided into three main classes: deep learning, neural networks, and decision trees, which in turn are subspecies of machine learning.

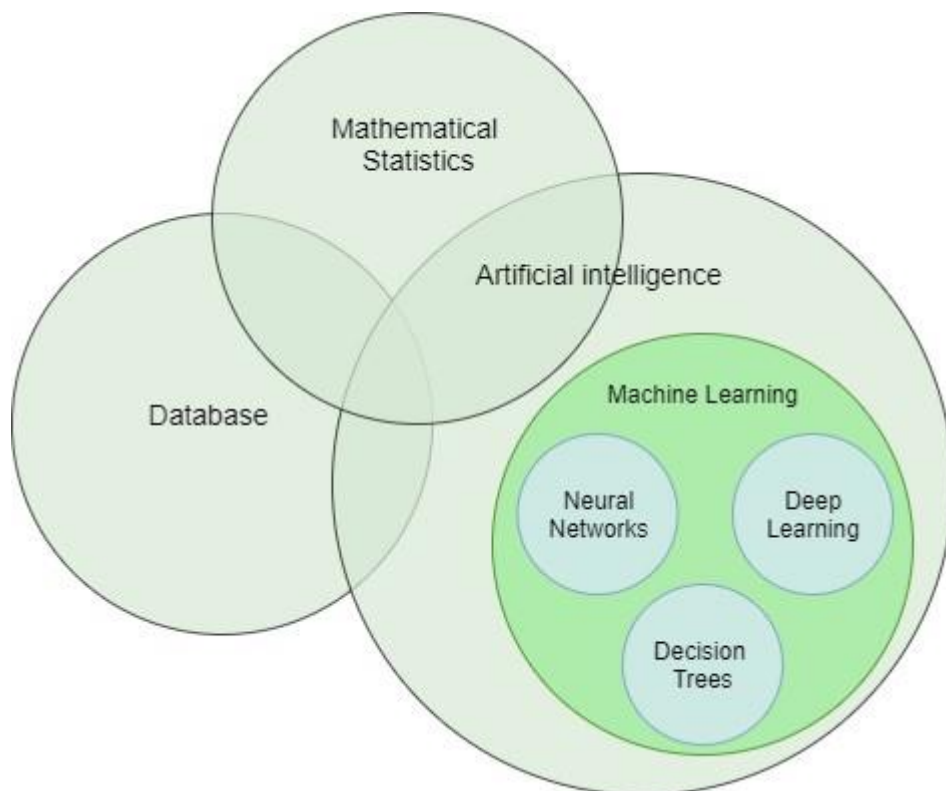


Figure 2.1 - Areas of artificial intelligence

Within the framework of the forecasting problem, it is considered that machine

learning methods allow solving problems of this type most effectively, due to scalability to accurate data (big data), high performance of algorithms and resistance to noise. However, this area has a number of disadvantages. One drawback is the complexity of interpreting the output variables.

The process of integrating machine learning models with data can be divided into five main stages (Figure 2.2):

- 1) data preparation;
- 2) building a model;
- 3) evaluation;
- 4) optimization;
- 5) forecast based on new data.

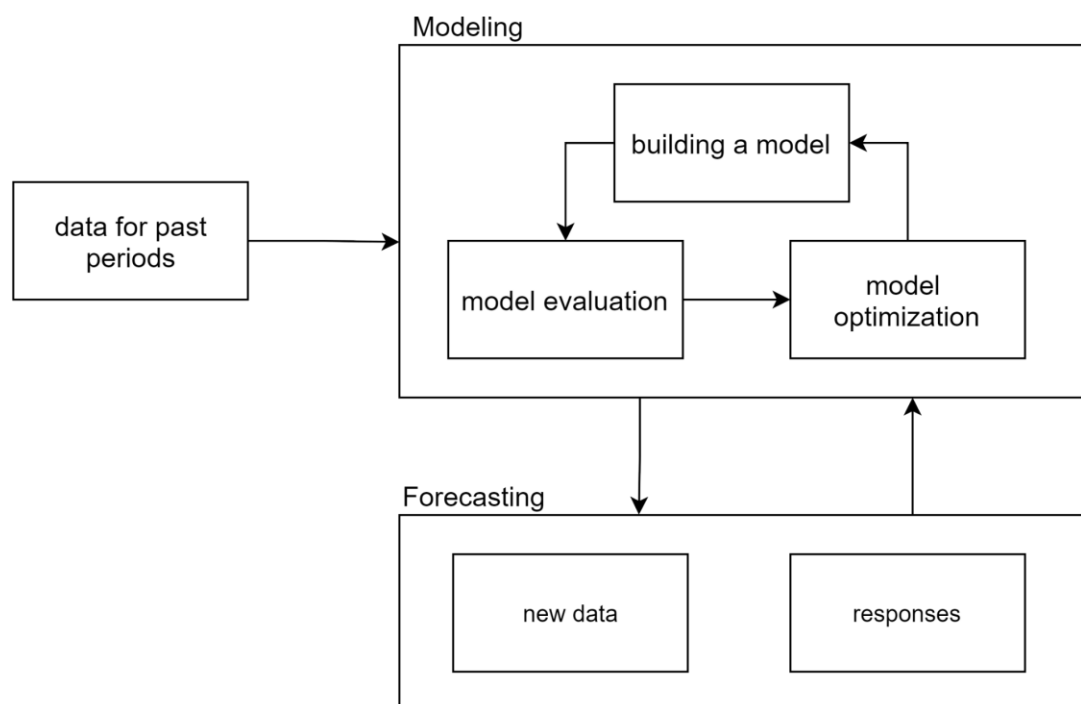


Figure 2.2 - Workflow of machine learning systems

Let's take a closer look at the second stage of the integration process. In order to build a model, you need to choose the algorithm that will be used for forecasting. Based on the analysis of the literature, it turned out that today the most effective algorithms are those with neural network and tree architectures, as well as individual configurations of neural network architectures from the deep learning class. It also turned out that the classical ARIMA model is highly effective within the stock market, therefore, there is a need to consider it and take it into account in the comparative analysis.

2.1. The architecture of neural networks

According to V.V. Nosov: “Neural networks are a new tool for data analysis, which is advisable to use where the formalization of the computational process for the studied phenomenon is impossible or extremely inefficient. It allows you to solve problems based on incomplete, noisy or distorted information, which traditional methods cannot cope with”.

The first paper suggesting the mathematical architecture of an artificial neuron and the construction of an artificial neural network was the paper by Warren McCulloch and Walter Pitts, published in 1943. The authors noted that due to the binary nature of neural activity – “the neuron is either “on” or “off”, with virtually no intermediate states”, neurons are conveniently described in terms of propositional logic, and for neural networks, they develop a whole logical apparatus that formalizes acyclic graphs. The very design of the artificial neuron, which McCulloch and Pitts call the Threshold Logic Unit (TLU), or Linear Threshold Unit, turned out to be very modern: a linear combination of inputs, which then enters the input of the nonlinearity in the form of a “step” that compares the result with a certain threshold.

In the mid-2000s, there was a revolution in the development of artificial intelligence: people learned how to train neural networks. A series of tasks that seemed impossible for artificial intelligence, such as image recognition, is now a trend. Today, there are a large number of software products based on neural network algorithms, which are also an integral part among users. A striking example is the virtual voice assistant “Alice” from the company “Yandex” (the task of speech recognition). Neural networks are divided into direct signal propagation networks, which do not have cycles, and recurrent neural networks, which use cycles for repetitive learning.

2.2 Feedforward neural networks

One of the most common types of artificial neural networks is feedforward neural networks. Such a network consists of several layers, each of which has only one-way connections to the next layer. There are no cycles in such a network as shown in Figure 2.3 .

One of the gradient learning methods can be used to train a neural network of direct propagation. The gradient is calculated on the basis of the conjugate graph of the network using the method of back propagation of the error.

Despite the fact that training an artificial neural network is a resource-intensive process, the calculations performed for a single neuron of the network are described

quite simply and do not require large resources. At the same time, in neural networks of direct propagation, neurons of the same layer do not have interneuron connections among themselves. These two factors make it possible to develop an effective algorithm for parallelizing the calculations of a separate layer of a neural network, including on a GPU.

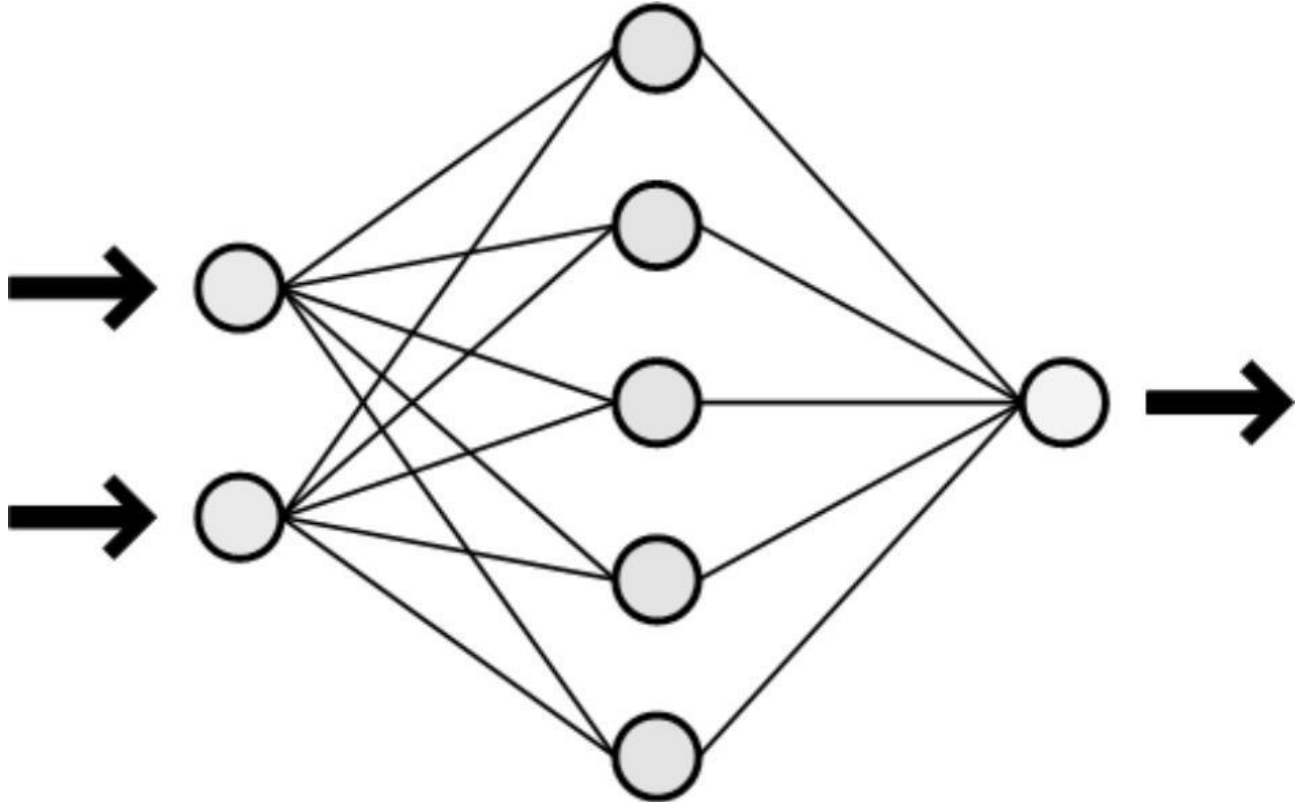


Figure 2.3 – The diagram of feedforward neural network

A neural network layer is characterized by a set of input signals that are fed to the network input, a set of weights, the activation function of each neuron, and a set of output signals of the layer. The output signal of the i -th neuron in the layer is described by the formula:

$$y_i = f(u_i), \quad (2.1)$$

where f - neuron activation function, $u_i = \sum_{j=0}^N \omega_{ij}x_j$ - the total signal that came to the i -th neuron, ω_{ij} - weight of the j -th interneuronal connection for the i -th neuron, x_j - the signal sent to the j -th input of the neuron, N - total number of source signals (including the polarization signal $x_0 = 1$), y_i - the final output signal of the neuron.

The symbols entered in the formula are schematically shown in Figure 2.4.

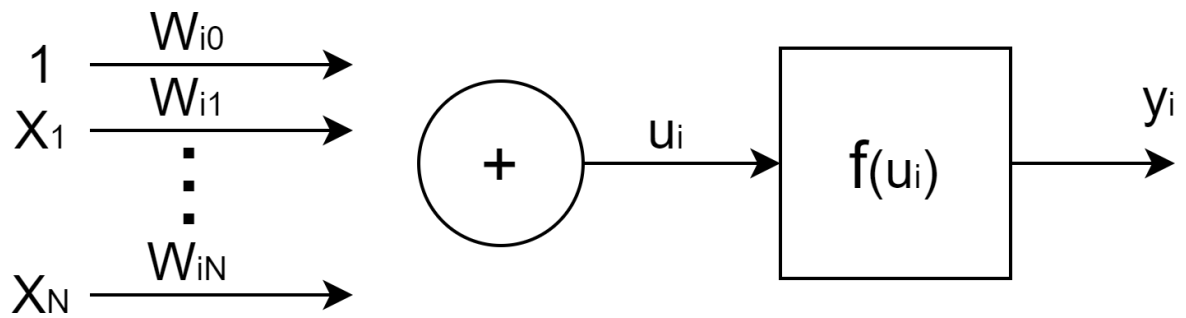


Figure 2.4 – The neuron model

One of the first neural architectures is the Multilayer Perceptron model (MLP). It consists of a set of neurons, where the neuron is an entity in the form of a computational unit. The set of such units is called the neural layer ($X_1 \dots X_n$).

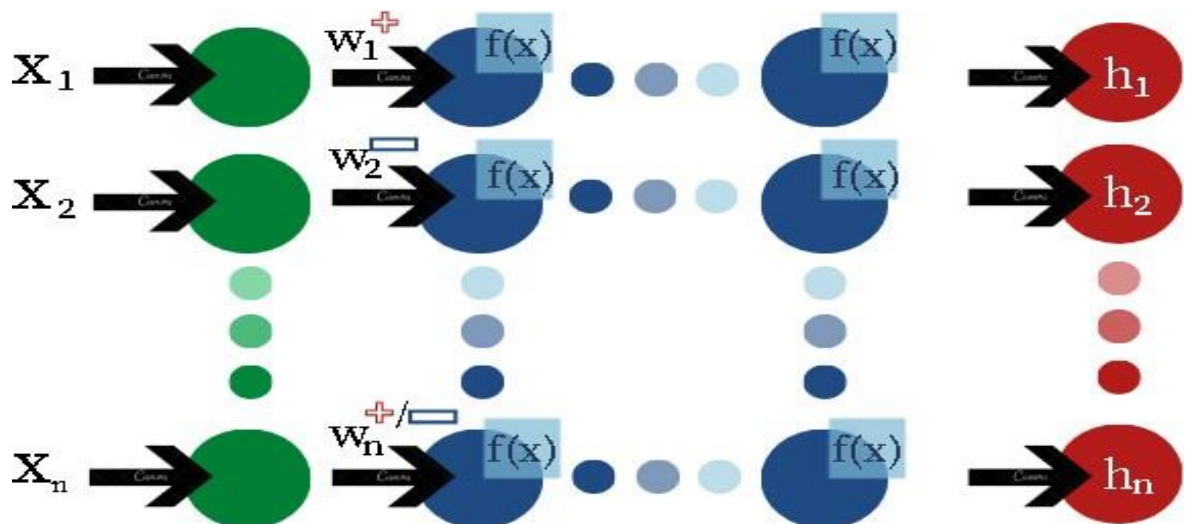


Figure 2.5 – The architecture of neural networks

Neural layers are divided into three types:

- The input layer (highlighted in green in Figure 2.5) includes information about the data intended for training the model.
- The hidden layer (highlighted in blue in Figure 2.5) includes neurons, each of which contains a specific activation function (see Table 2.1), which at the training stage classifies data according to a certain criterion, thereby creating templates. Then all the values are added together to activate the next hidden layer.
- The output layer (highlighted in red in Figure 2.5) includes information about the predicted patterns.

Table 2.1 - Activation functions

Function name	Formula
Sigmoid function	$\frac{1}{1 + e^{-x}}$
Hyperbolic function (tanh)	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$
Heaviside step function	$0, x < 0$ $1, x \geq 0$
SoftPlus	$\log(1 + e^x)$
ReLU	$0, x < 0$ $x, x \geq 0$
Leaky ReLU, Parameterized ReLU	$ax, x < 0$ $x, x \geq 0$
ELU	$a(e^x - 1), x < 0$ $x, x \geq 0$

The principle of the neural network architecture is that activation functions in one layer determine activation functions in the next layer. Each neuron is activated by a certain synaptic weight - a weighting factor (w_1, \dots, w_n). The structure of the links reflects the connections of the network elements. At the output, the template contained in the neuron with the highest value of the activation parameter (h_1, \dots, h_n) is activated.

The theoretical justification of neural network modeling is based on the theorem of A. N. Kolmogorov, who proved that any continuous multidimensional function on a unit segment $[0;1]$ can be represented as a finite number of one-dimensional functions [Golovko, 1999]:

$$f(x_1, x_2, \dots, x_n) = \sum_{p=1}^{2n+1} g(\sum_{i=1}^n \lambda_i \phi_p(x_i)), \quad (2.2)$$

where g and ϕ are continuous and one-dimensional functions, $\lambda_i = \text{const}$. From here, using any multi-layer ANN with only two processing layers, you can approximate any multidimensional function on a unit segment with any accuracy.

The possibilities of ANN training can be described by the example of a multi-layer perceptron with direct information dissemination.

As shown in Figure 2.6, neurons are regularly organized into layers, and the elements of a certain layer are connected only to the neurons of the previous layer, and

information is distributed from the previous layers to the subsequent ones. The input layer, consisting of sensitive (sensory) S -elements, to which the input signals x_i are received, does not perform any information processing and performs only distribution functions. Each S -element is connected to a set of associative elements (A -elements) of the first intermediate layer, and the A -elements of the last layer are connected to the reacting elements (R -elements).

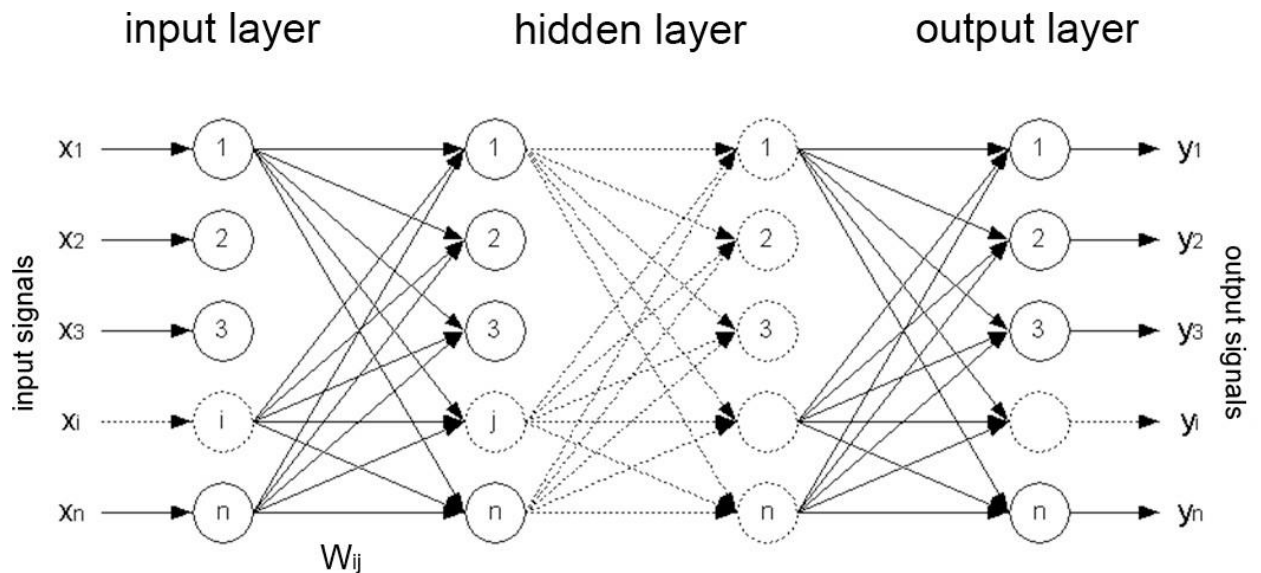


Figure 2.6 - Structure of multilayer perceptron

Weighted combinations of the outputs of the R -elements make up the reaction of the system, which indicates that the recognized object belongs to a certain image. If only two images are recognized, then a single R -element is installed in the perceptron, which has two reactions – positive and negative. If there are more than two images, then each image is assigned its own R -element, and the output of each such element is a linear combination of the outputs of the A -elements.

The choice of the number of layers and the type of activation function affects the ability of the network to solve certain tasks. A single-layer network is capable of forming linear separating surfaces and is easily reduced to the generalized portrait method. The linear model is a good reference point for comparing the efficiency of various multi-layer networks with nonlinear activation functions, allowing, as shown above, to approximate any convex multidimensional functional dependencies.

Let us clarify the mechanism of modeling separating surfaces with a multilayer perceptron. The activation level of each neuron is a simple linear function of the X s, i.e., the weighted sum of the input signals is taken, with a threshold value added to it. This activation is then transformed using the given $y = F(S)$ function. If an S-shaped sigmoid curve is used as the activation function, then the combination of a linear

function of several variables and a scalar logistic function $y = \frac{1}{1+e^{-CS}}$ results in a characteristic "sigmoid slope" profile, which produces an element of the first intermediate layer.

The corresponding surface is represented as a function of two input variables. When the weights w_1, \dots, w_n and the thresholds T change, both the orientation of the entire response surface and the slope steepness can change (higher values of the weights correspond to a steeper slope). An element with a large number of inputs produces a multidimensional analog of the represented surface.

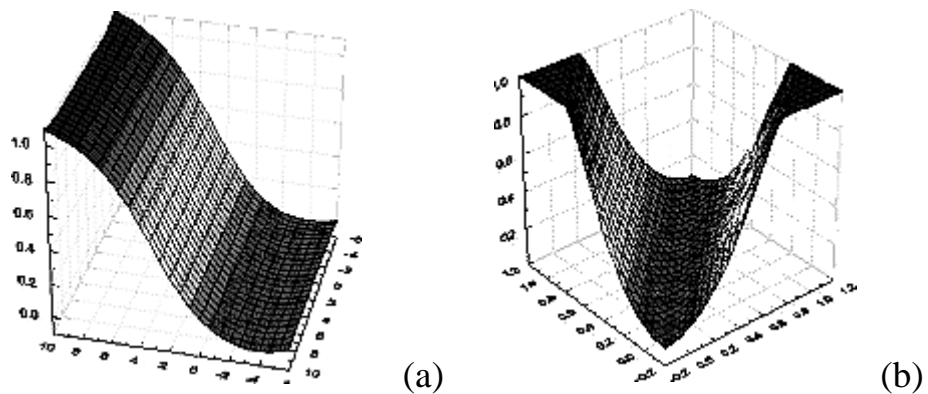


Figure 2.7 - A view of the simplest separating surface of the sigmoid type for a single neuron (a) and a perceptron with one intermediate layer (b)

In a multi-layer network, such response functions are combined with each other by sequentially taking their linear combinations and applying nonlinear activation functions. Figure 2.7 shows a typical response surface for a network with one intermediate layer consisting of two elements and one output element. Two different sigmoid surfaces are combined into a single surface that has the shape of the letter "U".

2.3 The Recurrent Neural Network

The first deep networks appeared in the mid-1960s: they were described in the works of the Soviet scientist A.G. Ivakhnenko. He developed a method for group accounting of arguments, the essence of which is as follows:

- First, a general view is chosen, a parametric family of models that will be trained.
- Using the quality metric, several best models are selected; in the case, if the best quality has already been achieved, then the cycle ends.

- If the required quality is not achieved, it is necessary to build models of the next level, using the outputs of the fitted parameters of the models at the previous step as inputs for subsequent ones.
- This process should be repeated recursively until the quality of the model either reaches the required level or stops improving.

The main features of RNN are the ability to use the initial information for the current task, as well as input values recursively, due to which the parameters are optimized and the quality of the result is improved.

For example, to predict the price of a stock today, we do not need all the data from decades of trades. Recurrent neural networks build dependencies on short-term data, updating them at each training cycle.

The recurrent neural network is shown in Figure 2.8.

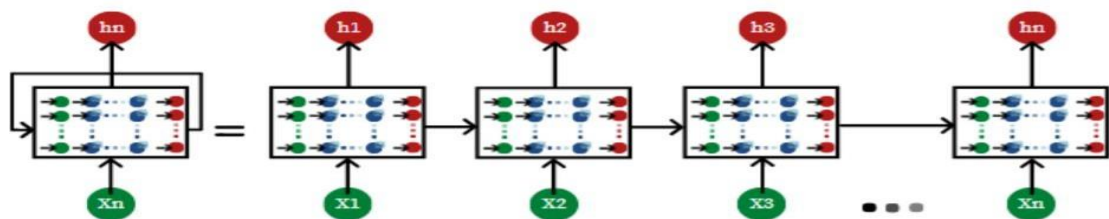


Figure 2.8 – The recurrent neural network

There are situations when more information is needed to analyze patterns, and recurrent neural networks lose the ability to analyze additions if the distance of the information flow grows. The solution for this problem was proposed by Jürgen Schmidhuber and Sepp Hochreiter. They proposed a special configuration of the recurrent neural network with the long-term short-term memory. The idea is to simulate "Memory" by adding a special gate cell, which eliminates the problem of growth of information flow. The LSTM architecture is shown in Figure 2.9.

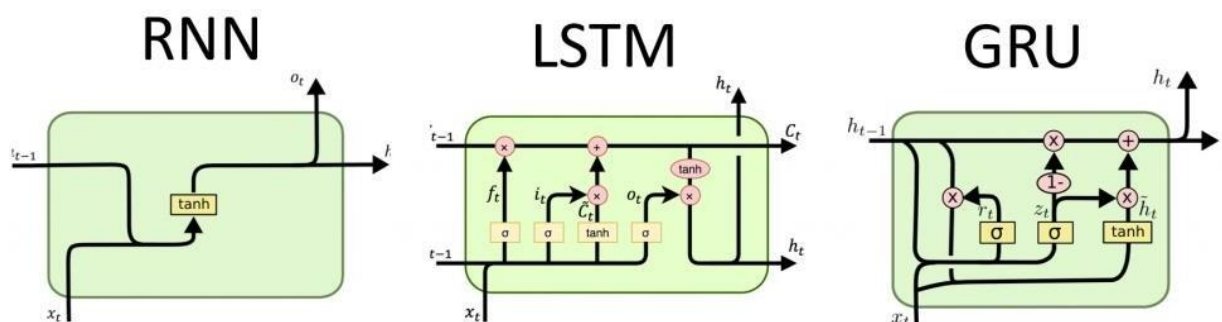


Figure 2.9 – The LSTM architecture

2.4 Decision trees

Decision trees, today, are considered to be one of the most efficient classes for solving forecasting problems.

A separate tree is a computational entity that consists of leaves and branches, where leaves are cells for storing the values of the objective function, branches are cells storing the values of the attributes, on which the objective function depends. The model building process is based on the classification of decision rules by recursively dividing the set of rules into subsets until the condition of stopping the algorithm is reached, for example, reaching the maximum tree depth.

2.4.1 The decision-making forest

The decision forest ("Multiclass Decision Forest") is an extension of the decision tree algorithm and is an ensemble learning method. As a rule, ensemble models provide higher accuracy than individual decision trees. The algorithm works by building multiple decision trees and then "voting" for the most popular class containing the output. Voting is a form of aggregation in which each classification tree from the decision forest outputs a non-score-enameled frequency histogram of labels. During the aggregation process, the histograms are summed and the results are normalized. As shown in Figure 2.11 trees with high prediction confidence have more weight in the final decision of the ensemble.

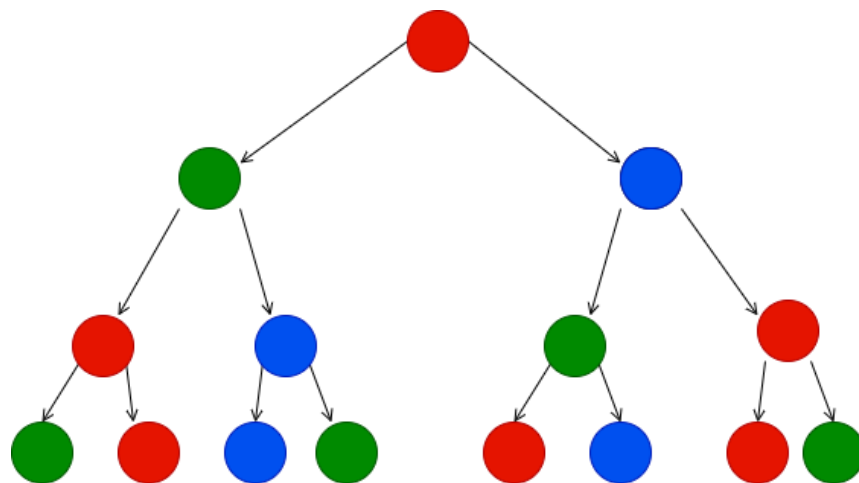


Figure 2.11 – The decision tree, based on the decision forest algorithm

Since decision forest-based models are nonparametric, they can support data with different distributions. In each tree, a sequence of simple tests is performed for

each class, increasing the levels of the tree structure until the final node (solution) is reached. The advantages of this algorithm include:

- nonlinear boundaries of data approximation with the objective function;
- high computational efficiency and resistance to noisy data;
- comprehensive selection and classification of objects.

2.4.2 The decision-making jungle

The Decision Jungle - is an extended modification of the decision forest algorithm. The decision jungle consists of acyclic graphs aimed at making decisions. Unlike classical decision trees, "jungle" requires less memory to build deep trees, while the algorithm has high performance and the ability to generalize even on noisy data. However, the algorithm requires high training time. Unlike traditional decision trees, which assume only one path to each node, the decision jungle allows multiple paths from the root to each leaf (see Figure 2.12).

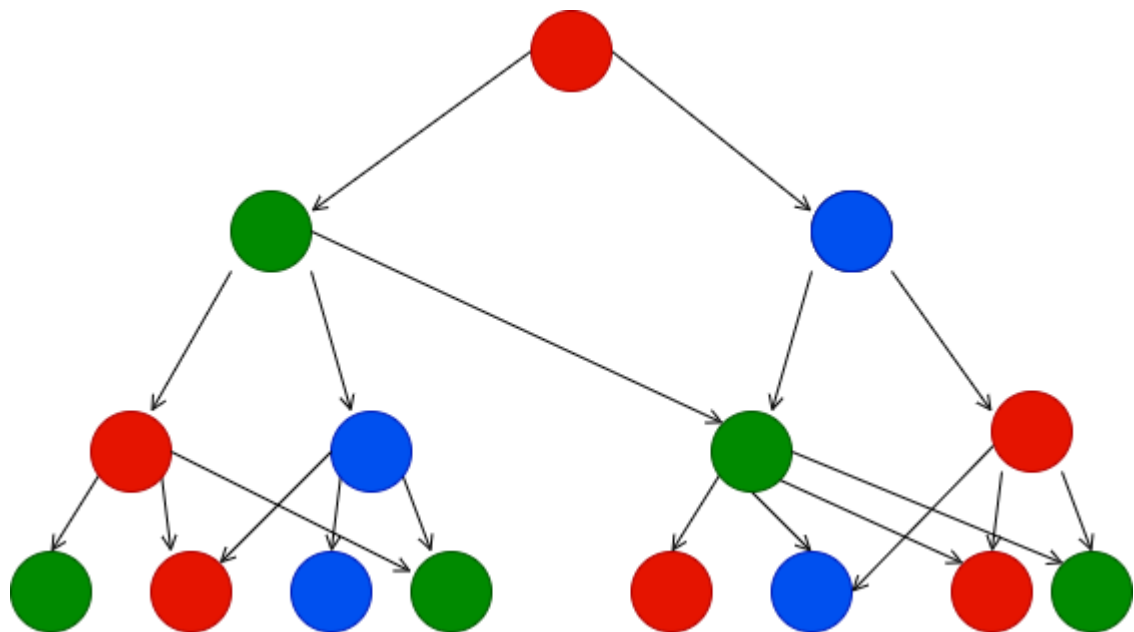


Figure 2.12 – The decision tree based on the decisive jungle algorithm

2.5 The ARIMA model

A classic and effective approach to time series forecasting is the Box-Jenkins methodology, which is a family of algorithms and one of them is the ARIMA model (Autoregressive integrated moving average).

The idea of this algorithm is to combine autoregression processes, integration processes, and moving average processes. The obvious disadvantage of this model is the limitation, which is that ARIMA does not involve the inclusion of additional factors that can affect the time series. The model is based only on the information contained in the background of the predicted time series.

The autoregression model of order p is described as

$$AR(p): y_t = c + \phi_1 y_{(t-1)} + \phi_2 y_{(t-2)} + \dots + \phi_p y_{(t-p)} + \varepsilon_t \quad (2.3)$$

and shows the dependence of the value of the current period on the past values of the periods.

The moving average order model q is described as

$$MA(q): y_t = c + \varepsilon_t + \theta_1 \varepsilon_{(t-1)} + \theta_2 \varepsilon_{(t-2)} + \dots + \theta_q \varepsilon_{(t-q)} \quad (2.4)$$

and shows the dependence of the value of the current period on the prediction errors of the previous q periods.

Autoregressive model with an integrated and moving average orders (p, d, q) is the sum $AR(p)$ and $MA(q)$ models and can be represented in the form

$$ARIMA(p, d, q): (1 - \phi_1 L - \dots - \phi_p L^p)((1 - L)^d y_t - \mu) = (1 + \theta_1 L + \dots + \theta_q L^q) \varepsilon_t, \quad (2.5)$$

where d - the number of differentiations of the original time series before reaching its stationarity, L - lag value.

The ARIMA (p, d, q) model for a non-stationary time series X_t has the form:

$$\Delta^d X_t = c + \sum_{i=1}^p a_i \Delta^d X_{t-i} + \sum_{j=1}^q b_j \varepsilon_{t-j}, \quad (2.6)$$

where c, a_j, b_j - model parameters, ε_t - stationary time series, Δ^d - the time series' difference operator of order d .

One of the main advantages of ARIMA is the ability to model integrated or difference-stationary time series.

A time series X_t is called an integrated one of order k if the differences of the series of order k , i.e. $\Delta^k X_t$, are stationary series, while the differences of a smaller order are not stationary processes with respect to some trend.

To highlight the seasonality you can use a modification of the base model $AR(p, d, q)$ - $SARIMA(P, D, Q)m$, where m is the number of measurements per year; $P, D,$

Q are the orders of seasonal autoregression, seasonal integration, and seasonal moving average, respectively.

The *auto.arima()* function accepts a time series and independently selects the necessary parameter values. To do this, the function finds the optimal number of differentiations that give a stationary series. Subsequently, the optimal model is selected among the models *ARIMA* with the parameters (2, 2), (0,0), (1,0), (0,1). After that, the parameters of the selected model are changed to ± 1 , and constants are added/removed. Such iterations occur until a change in the parameters leads to a decrease in the adjusted Akaike criterion. The selection of seasonal parameters follows the same pattern. As with most forecasting interval calculations, *ARIMA* based intervals tend to be too narrow.

2.6 The SVM model

The support vector machine (SVM) is a machine learning algorithm that solves classification and regression problems by constructing a nonlinear plane separating the solutions. Due to the nature of the feature space, in which the solution boundaries are constructed, the support vector machine has a high degree of flexibility in solving regression and classification problems of various levels of complexity. There are different types of SVM models: linear, polynomial, RBF (radial basis functions), and sigmoid.

It was introduced by Vapnik in 1995 and later was promoted to SVR (Support Vector Regression) by using ϵ -insensitive loss function. SVR is a kernel-based non-linear regression method, which provides the best regression hyperplane with small structural risk in multidimensional feature space.

The Support Vector Machines (SVM) method is shown in Figure 2.13, which based on the concept of hyperplanes that define the boundaries of hypersurfaces. A separating hyperplane is a hyperplane that separates a group of objects that have different classes. The approximate scheme is shown in the figure below.

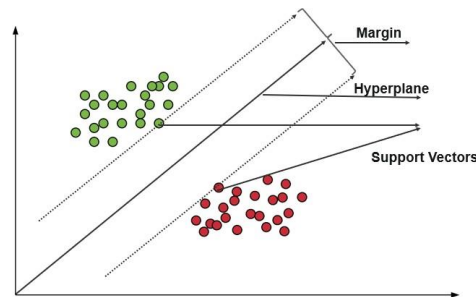


Figure 2.13 - The SVM model

There are a couple of scenarios that are worth looking into to determine the best hyperplane.

Scenario 1 (Figure 2.14): The picture below shows three hyperplanes (A, B, and C). It is easy to see that only the hyperplane B correctly separates one class of records from the others.

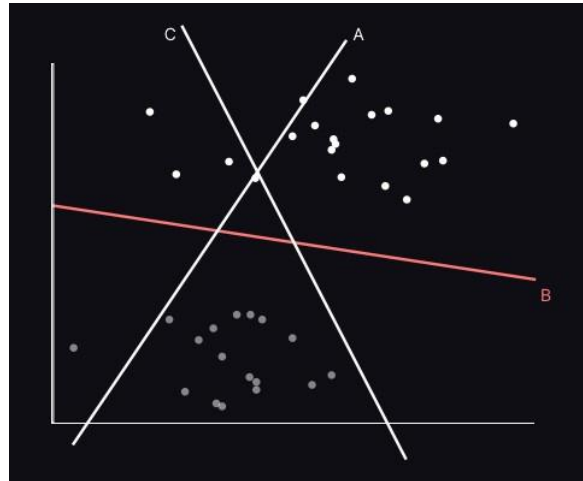


Figure 2.14 - Scenario #1

Scenario 2 (Figure 2.15): There are now three hyperplanes in the image, and they all share classes well. How to determine the correct one in this case? The plane with the greatest distance from the extreme observations of both classes is the best. This distance is called the “Margin”.

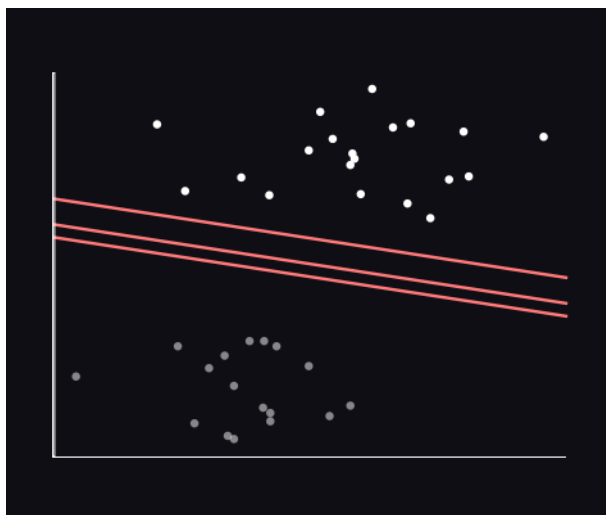


Figure 2.15 - Scenario #2

Scenario 3 (Figure 2.16) : If we project the extreme points on each of the separating hyperplanes, the median one will be the best. An important reason for choosing a hyperplane with a higher margin is reliability. If we choose a hyperplane with a low margin, then there is a high probability of an error in the classification.

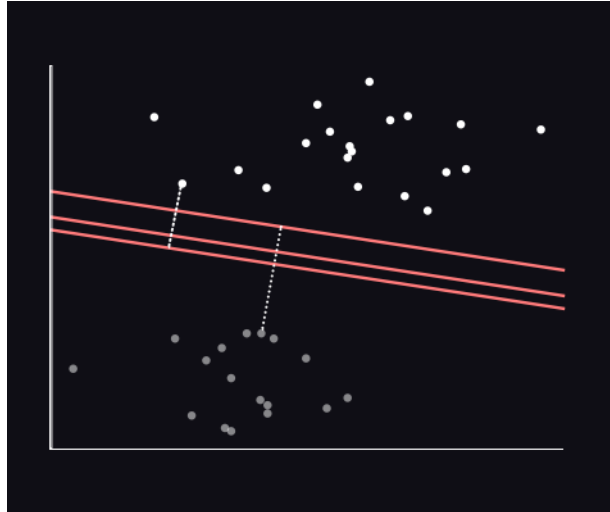


Figure 2.16 - Scenario #3

Scenario 4 (Figure 2.17) : Despite the large fields of the hyperplane B, the best boundary is A, since the latter does not make classification errors.

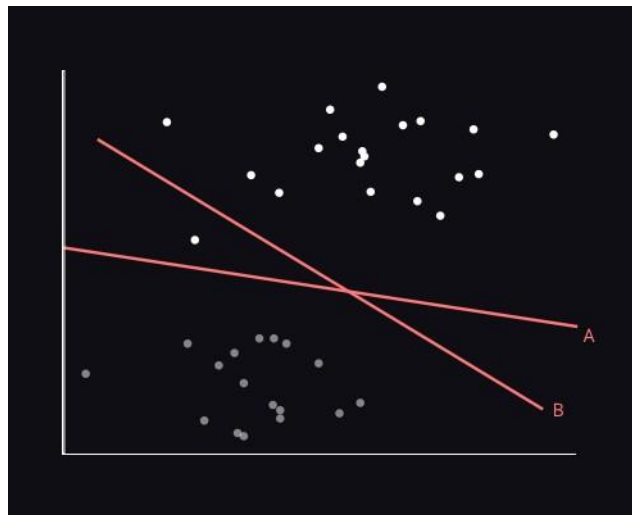


Figure 2.17 - Scenario #4

Scenario 5 (Figure 2.18) : it is not possible to divide the observations into groups using a linear hyperplane, because one of the grey records is located in the territory of

a cluster of another class, and this is an Outlier. But, the SVM algorithm is resistant to outliers and allows you to ignore outliers and find a hyperplane with a maximum field.

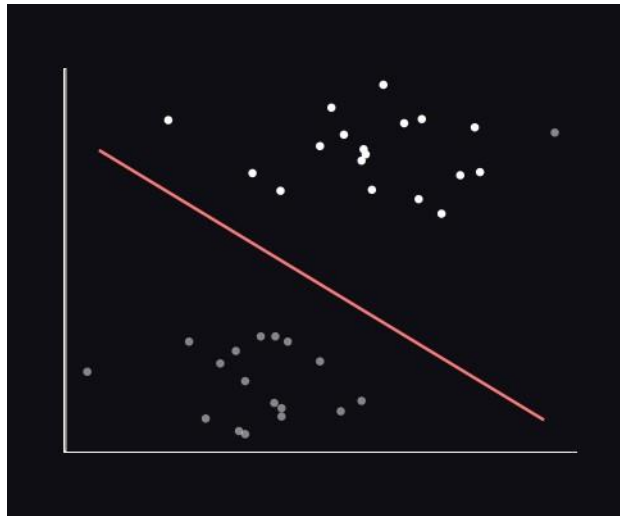


Figure 2.18 - Scenario #5

Scenario 6 (Figure 2.19) : white and grey points are scattered, and it looks like there is no solution in this situation, but it can be easily solved by SVM using the additional function $z = x^2 + y^2$. Thanks to the introduction of the z-axis, the clusters look different in the x-and z-axis pair.



Figure 2.19 - Scenario #6

In the graph above, should be considered the following:

- All z values will always be positive, because it is the sum of the squares of x and y

- In the original graph, white dots appear near the origin of the x and y axes, which reduce the z value. At the same time, the gray points are more distant from the origin, and this increases the z.

2.7 Review of the LSTM model

A special type of RNN that can be trained for long-term addition is called the Long-Short Term Memory (STM). It contains information in memory, which is similar to the computer's memory. It can read, write and delete information in his memory. This memory can be considered as a closed cell; with the closed description the cell decides to keep or delete Information. The LSTM has three gates shown in Figure 2.20: entrance, oblivion, and exit. These gates determine whether there will be a new entrance (entrance gateway) should the data be allowed to be deleted because it is not important (forget the gate), or allow it to affect the output on the current timeline (the output gate):

1. The Forget Gateway determines when certain parts of the cell will be inserted with information that is more recent.
2. The Input gate. Based on the input (for example, the previous output $O(T-1)$, input $X(T)$, and the previous state of cell $C(T1)$), this is when the network category reads the conditions, under which any information should be stored (or updated) in the status cell.
3. Output gate. Depending on the input mode and the cell, this component determines what information is sent to the next location in the network.

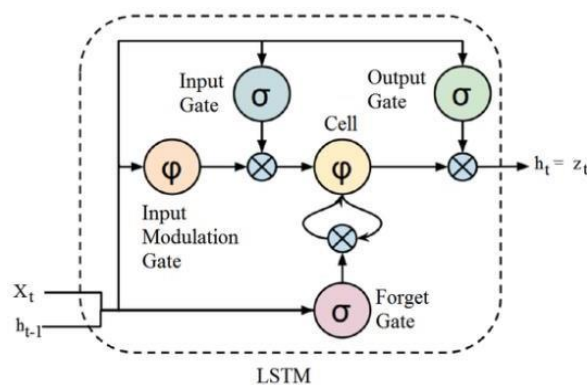


Figure 2.20 – The LSTM model

2.7.1 Benefits of LSTM

The main advantage of LSTM is shown in Figure 2.21, its ability to read the intermediate context. Each unit remembers details for a long or short period without explicitly using the activation function in repeated components. The important fact is that any cell state is repeated only with the release of the forgetting gateway, which varies from 0 to 1. That is, the forgetting gateway in the LSTM cell is responsible for both the hardware and the cell state activation function. Thus, the data from the previous cell can pass through an immutable cell instead of explicitly increasing or decreasing at each step or level, and tools can convert to their respective values within the limited time. This allows LSTM to solve the perishable gradient problem, because the amount stored in the memory cell is not converted in the repetitive way, the gradient does not end when learning the distribution in the opposite direction.

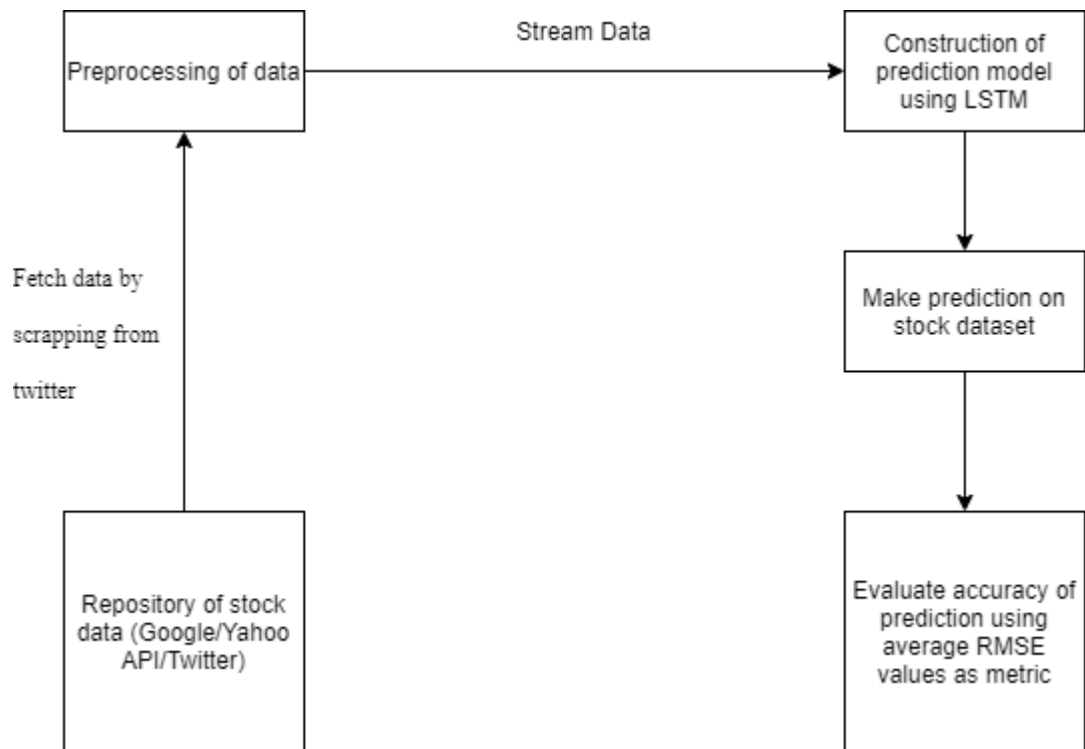


Figure 2.21 – The system architecture

2.8 Methods for evaluating algorithms

In order to numerically determine the effectiveness of algorithms, quality metrics are needed. And one of them is the Mean Squared Error or MSE. This is probably the simplest and most common metric for estimating regression, but probably the least useful. It can be defined by the equation:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2.7)$$

where y_i is the actual expected result and \hat{y} is the model's forecast.

MSE basically measures the root-mean-square error of our forecasts. For each point, the square difference between the forecasts and the target is calculated, and then these values are averaged.

We know that the higher this value, the worse the model. It is never negative, since we square the individual prediction errors before summing them up, but for an ideal model it will be zero.

Advantages: Useful if we have unexpected values that we need to take care of. We should pay attention to the value whether it is very high or low.

Disadvantages: If we make one very bad prediction, squaring will make the error even worse, and this can skew the metric towards overestimating the model's badness. This is particularly problematic behavior if we have noisy data (i.e. data that are not entirely reliable for some reason) - even in an "ideal" model, there may be the high MSE in this situation, so it becomes difficult to judge how well the model performs. On the other hand, if all the errors are small, or rather less than 1, then the opposite effect is felt: we may underestimate the shortcomings of the model.

The second method is the Root Mean Square Error (RMSE), it is just the square root of the MSE. The square root is entered so that the error scale is the same as the target scale.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2} \quad (2.8)$$

It is very important to understand in what sense the RMSE is similar to the MSE, and what is the difference. First, they are similar in terms of their minimizers, each MSE minimizer is also a minimizer for the RMSE and vice versa, since the square root is a non-decreasing function. For example, if we have two sets of predictions, A and B, and say that the MSE for A is greater than the MSE for B, then we can be sure that the RMSE for A is greater than the RMSE for B. And it also works in the opposite direction:

$$MSE(a) > MSE(b) \Leftrightarrow RMSE(a) > RMSE(b) \quad (2.9)$$

This means that if the target is the RMSE, we can still compare our models using the MSE, since the MSE will order the models in the same way as the RMSE. So we can optimize the MSE instead of the RMSE.

In fact, the MSE is a bit easier to work with, so everyone uses it instead of the RMSE. There is also a slight difference between the two gradient-based models.

$$\frac{\delta RMSE}{\delta \mathbf{y}} = \frac{1}{2\sqrt{MSE}} \frac{\delta MSE}{\delta \mathbf{y}} \quad (2.10)$$

The gradient of the coex system is relative to the i-th forecast. This means that traveling on the MSE gradient is equivalent to traveling on the RMSE gradient, but with a different flow rate, and the flow rate depends on the MSE score itself.

Thus, while the RMSE and the MSE are indeed similar in terms of the model evaluation, they cannot be immediately interchangeable for gradient-based methods. We may need to adjust some parameters, such as the learning rate.

Next evaluation method is the MAE (the Mean Absolute Error). Here, the error is calculated as the average of the absolute differences between the target values and the forecasts. The MAE is a linear estimate, which means that all individual differences are weighted equally on average. For example, the difference between 10 and 0 will be twice the difference between 5 and 0. However, the same is not true for the RMSE. Mathematically, it is calculated using the following formula:

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i| \quad (2.11)$$

What's important about this metric is that it punishes huge errors that aren't as bad as the MSE. Thus, it is not as sensitive to outliers as the root-mean-square error.

The MAE is widely used in finance, where \$10 error is usually twice as bad as \$5 error. On the other hand, the MSE metric finds that \$10 error is four times worse than \$5 error. The MAE is easier to justify than the RMSE.

Another important thing about the MAE is its gradients relative to forecasts. The gradient is the step-by-step function that takes -1 when \hat{Y} is smaller than the target, and +1 when it is larger.

Now the gradient is undefined when the prediction is perfect, because when \hat{Y} is Y , we can't estimate the gradient. This is undefined.

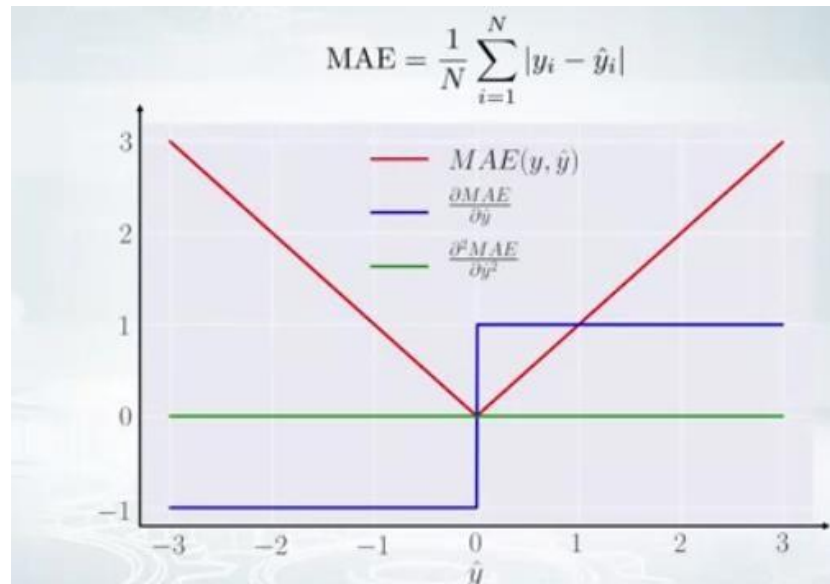


Figure 2.22 – The MAE: derivatives

So, formally, the MAE is not differentiable, but in fact, how often your forecasts accurately measure the target. Even if they do, we can write a simple IF condition and return zero if it is, and via a gradient otherwise. It is also known that the second derivative is zero everywhere and is not defined at the zero point, shown in Figure 2.22 .

The coefficient of determination, or R-squared, is another metric we can use to evaluate the model, and it is closely related to the MSE, but has the advantage of being scale-free - it doesn't matter if the output values are very large or very small, R-squared will always be between -∞ and 1.

$$R^2 = 1 - \frac{MSE(model)}{MSE(baseline)} \quad (2.12)$$

The MSE of the model is calculated as above, while the MSE of the baseline is defined as:

$$MSE(baseline) = \frac{1}{N} \sum_{i=1}^N (\bar{y} - y_i)^2 \quad (2.13)$$

where \bar{y} means the average of the observed y_i .

To make it more clear, this basic MSE can be thought of as the MSE that the simplest model will get. The simplest possible model would always be to predict the average across all samples. A value close to 1 indicates a model with an error close to zero, and a value close to zero indicates a model very close to the baseline.

3. PRACTICAL IMPLEMENTATION

Recall that the purpose of the study is to select the most appropriate algorithm for predicting price changes in the stock market based on the text analysis. We'll be using the data that would describe the state of Kazakhstan's stock market.

To achieve this goal, a number of theoretical aspects (stages) were identified in Figure 3.1 .

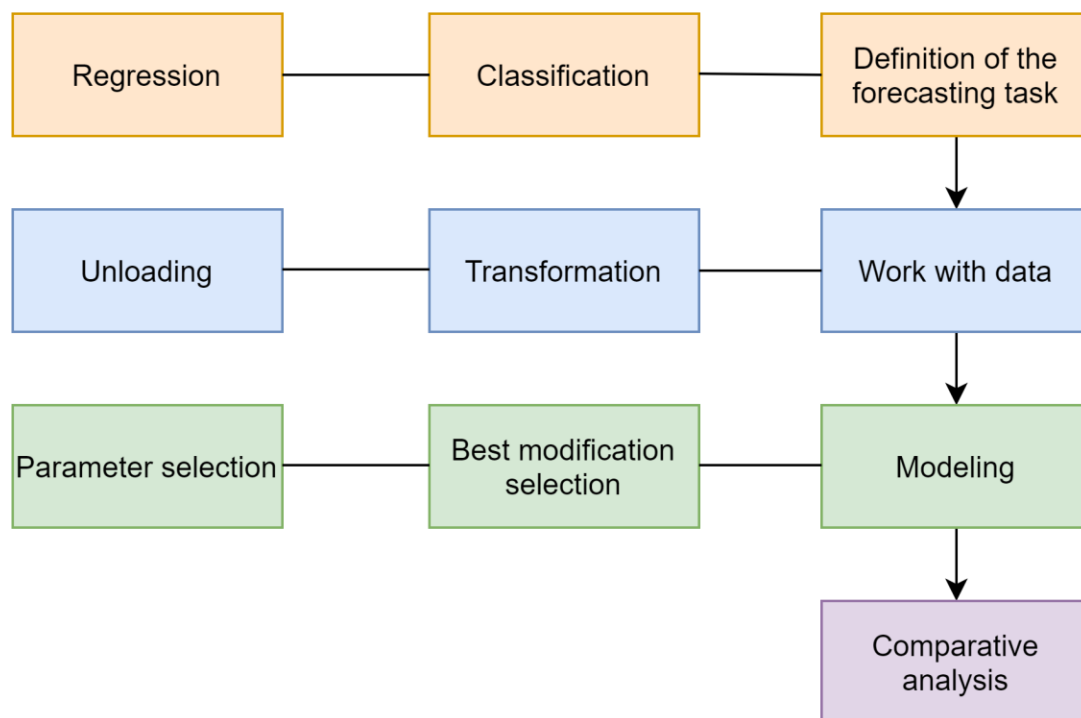


Figure 3.1 - Research stages

At the first stage, it was necessary to determine the type of the problem to be solved, and the regression approach was the most effective for predicting price changes over a certain period of time.

At the second stage, it was necessary to prepare the data. We used the Twint Scraper Library to gather data from the Twitter page @Kapitalkz, where daily updated and posted news related to the financial market. As a dataset we used KASE index data from kase.kz.

The following tools and built-in libraries were used during the work:

- Twint Scraper - an improved scraping tool which allows you to scrape Tweets from Twitter.
- Jupyter notebook - an open source web application that allows us to form and share codes and documents. It provides an environment where you'll

report your code, run it, see the result, visualize the information, and see the comes about without taking off the environment. This makes it a helpful device for performing end-to-end information handling workflows - information cleaning, factual modeling, building and preparing machine learning models, information visualization, and numerous other applications. Jupyter notebooks really sparkle when you're still within the prototyping stage. This is often since your code is composed in free cells that are executed separately. This permits the client to test a particular piece of code to an extent without having to execute the code from the exceptionally starting of the script. Many other IDEs (like RStudio) moreover do this in a few ways, but Jupyter's structure is the leading of all.

- NLTK may be a bundle of libraries and programs for creating Python programs that work with characteristic dialect. It is accompanied by broad documentation, as well as a book clarifying the fundamental concepts of the issues that this bundle is outlined to illuminate. This bundle is reasonable for zones such as computational etymology, experimental phonetics, cognitive science, fake insights, data recovery, and machine learning. NLTK is utilized basically as a preparing instrument, person preparing, or prototyping and creating systems centered on investigating activities. NLTK is a free computer program that's accessible without charge.
- Dostoevsky[14][15] is a sentiment analysis library for Python trained on the RuSentiment dataset. It takes a single word or text as input and returns its sentiment classification: the classes could be positive, negative or neutral.

3.1 Using News Sentiment

The market sentiment is the qualitative measure of the attitude and mood of investors to financial markets in general and to specific sectors or assets in particular. Positive and negative moods influence price behavior and create trading and investment opportunities for active traders and long-term investors.

Existing research on sentiment analysis has shown that there is a strong correlation between the movement of stock prices and the publication of news articles. Several attempts have been made to analyze sentiment at different levels using algorithms such as support vector machines, naive Bayesian regression, and deep

learning. The accuracy of deep learning algorithms depends on the amount of training data provided.

Our models will use machine learning models listed above. A news sentiment analysis algorithm will be applied on the most efficient model. The model efficiency is measured by the methods mentioned earlier: RMSE, MSE, ROC curve, and etc.

3.2 The dataset description

We are using the KASE Index for the stock price data and Stock News from a Kazakhstani financial news twitter handle - @Kapitalkz.

Historical KASE Stock Price data was obtained from kase.kz, a stock exchange in Almaty, Kazakhstan. The Kazakhstan Stock Exchange (KASE) is a stock exchange headquartered in Almaty, Kazakhstan, and ranks second among the CIS stock exchanges in terms of stock market capitalization. It was founded in 1993. The initiators of its creation were 23 leading Kazakh commercial banks, headed by the National Bank of the Republic of Kazakhstan. The main tasks assigned to the Exchange were the organization and development of the national currency market in connection with the introduction of the national currency of Kazakhstan – tenge.

Over the past two decades, the Exchange has become a universal platform for conducting transactions in foreign exchange, government securities, corporate securities and derivative financial instruments. Therefore, the speed and efficiency of internal operating activities are particularly important for the Exchange.

Previously, the entire document flow of the Exchange was implemented in the paper form, which significantly slowed down and lowered the overall productivity of employees. In order to automate the main processes of the Exchange's office work, it was decided to introduce the electronic document management system (hereinafter referred to as the EDS).

Being a 100% domestic development, the Documentolog EDMS allowed you to get the maximum effect from the transition from the paper to electronic document management with minimal investment.

Documentolog has automated work with such documents as incoming / outgoing correspondence, internal documents, appeals of persons, orders, orders for operational activities, contracts for the purchase of goods, works and services (TRU) and civil contracts (GPH). This made it possible to quickly track the stages of business processes, which made all the activities of the Exchange more transparent and controlled.

Especially for the Exchange, the "Resolution" document type was finalized, which became more convenient and versatile thanks to the Customer's proposals.

Each embedded document type was configured for the internal processes of the Exchange, and synchronization of the EDMS with the internal domain structure of Active Directory was also configured.

The implementation of the Documentolog EDMS lasted for a total of 3 months, during which individual routes of the Exchange's internal documents were set up and the EDMS was finalized for the specific processes of the Customer.

As part of the project, our specialists trained 140 users, after which the project was put into commercial operation.

3.3 Twitter News Data

@Kapitalkz is shown in Figure 3.2 and 3.3, that contains historical news of Kazakhstan's finances in Russian language. It includes articles from different spheres of Kazakhstan's business life: economics, politics, commerce, and etc. We found the data found on the twitter handle to be suitable for our project, because the data is often expressed in a concise and formal manner, and is easy to process.



Figure 3.2 - @Kapitalkz's Twitter page info



Figure 3.3 - Tweet examples from @Kapitalkz Twitter account

3.4 Data gathering

Twitter messages were collected using Twint Scraper Library in Figure 3.4 and 3.5. Historical data contain 3008 tweets published since 10/12/2007 and until 06/04/2021, which is approximately 1.5 months. The process and code of data collection using Twint Scraper Library, along with the resulting data is described below.

```
import twint
import nest_asyncio
nest_asyncio.apply()

#configuration

config = twint.Config()
config.Username = "Kapitalkz"
config.Lang = "ru"
config.Since = "2016-01-01"
config.Until = "2021-04-07"
config.Store_csv = True
config.Output = "Kapitalkz.csv"

#running search
twint.run.Search(config)

1379393119276400642 2021-04-06 17:18:42 +0600 <Kapitalkz> Доллар закрыл торги на KASE на отметке 429 тенге https://t.co/3fZL
Uw8Xo5
1379364070877450243 2021-04-06 15:23:16 +0600 <Kapitalkz> В Казахстане разработали 15 стандартов «Халал» https://t.co/DEHKQ
R1wjw
1379363931769159681 2021-04-06 15:22:43 +0600 <Kapitalkz> Запись на вакцинацию запустили в Нур-Султане https://t.co/bWgK12tB
YL
1379363852366839809 2021-04-06 15:22:24 +0600 <Kapitalkz> Индекс деловой активности перешел в положительную зону https://t.c
o/0ezqS1szwJ
1379306119944929283 2021-04-06 11:32:59 +0600 <Kapitalkz> Поставку более 6 млн доз вакцин ожидает Минздрав https://t.co/VLkw
Seoi9Y
1379306076294803456 2021-04-06 11:32:49 +0600 <Kapitalkz> Избран заместитель председателя правления KASE https://t.co/eB2hN
FdIwR
```

Figure 3.4 - The process of data collection using Twint Scraper Library

	id	conversation_id	created_at	date	time	timezone	user_id	username	name	place	...	geo	source	user_rt
0	1343478471666892802	1343478471666892802	2020-12-28 14:46:42 Центральная Азия (зима)	2020-12-28	14:46:42	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
1	1343438630497505281	1343438630497505281	2020-12-28 12:08:24 Центральная Азия (зима)	2020-12-28	12:08:24	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
2	1343438584125288448	1343438584125288448	2020-12-28 12:08:12 Центральная Азия (зима)	2020-12-28	12:08:12	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
3	1343438536469602304	1343438536469602304	2020-12-28 12:08:01 Центральная Азия (зима)	2020-12-28	12:08:01	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
4	1343438495075991552	1343438495075991552	2020-12-28 12:07:51 Центральная Азия (зима)	2020-12-28	12:07:51	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
...
3803	1204646353583906816	1204646353583906816	2019-12-11 12:17:26 Центральная Азия (зима)	2019-12-11	12:17:26	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
3804	1204637841097187328	1204637841097187328	2019-12-11 11:43:37 Центральная Азия (зима)	2019-12-11	11:43:37	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
3805	1204637595516489728	1204637595516489728	2019-12-11 11:42:38 Центральная Азия (зима)	2019-12-11	11:42:38	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
3806	1204366233103163392	1204366233103163392	2019-12-10 17:44:20 Центральная Азия (зима)	2019-12-10	17:44:20	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN
3807	1204360945453146112	1204360945453146112	2019-12-10 17:23:19 Центральная Азия (зима)	2019-12-10	17:23:19	600	286073949	kapitalkz	Капитал.kz	NaN	...	NaN	NaN	NaN

3808 rows × 36 columns

Figure 3.5 - The data collected using Twint Scraper Library

The data contained in our KASE Index dataset includes the following information: date of listing, high and low, open and close share prices and the volume of stocks sold during that day: in USD and in KZT.

	Date	Open	High	Low	Close	Volume, KZT m	Volume, USD th
0	01.10.07	2550.32	2571.26	2525.93	2560.72	32.07	265.13
1	02.10.07	2540.33	2694.51	2518.86	2681.17	705.08	5828.05
2	03.10.07	2685.97	2691.65	2612.81	2612.81	306.81	2536.04
3	04.10.07	2611.91	2625.22	2209.53	2422.47	495.16	4092.92
4	05.10.07	2476.84	2489.49	2378.87	2443.75	685.44	5667.62

Figure 3.6 – The KASE Index dataset

3.5 Data cleaning and preprocessing

The data collected using Twint Scraper Library shown in Figure 3.7, is extensive, yet we only need to use two columns only: the ‘date’ column and the ‘tweet’ column, shown below.

	date	tweet
0	2020-12-28	Аналитики о курсе тенге: ждите неожиданного ...
1	2020-12-28	Мораторий на создание квазигосударственных ком...
2	2020-12-28	Ставка на квартиры экономкласса https://t.co...
3	2020-12-28	Какие возможности для инвесторов открыл 2020 г...
4	2020-12-28	Доллар начал торги возле отметки 420 тенге ht...

Figure 3.7 - Tweets dataset after removing excess columns

For the KASE Index dataset, we have also decided to keep only the ‘date’ and ‘high’ columns to simplify the procedure of data processing as shown in Figure 3.8 . The data won’t require any further modifications.

	Date	High
0	2007-10-01	2571.26
1	2007-10-02	2694.51
2	2007-10-03	2691.65
3	2007-10-04	2625.22
4	2007-10-05	2489.49

Figure 3.8 – The KASE Index dataset after removing excess columns

3.6 - Data modification

After preprocessing the data in the Kapitalkz tweets dataset we also need to modify it in order to be able to use it with any language processing algorithms. The modifications include removing excess symbols, numbers and links, followed by grouping the data by the publishing date of the tweets. Here is how the data looks after the modification process:

	date	tweet
0	2019-12-10	доллар закрыл торги на уровне тенге перезагруз...
1	2019-12-11	максимальный курс доллара в обменниках тенге в...
2	2019-12-12	нацбанк казахстана планирует открыть представи...
3	2019-12-13	лифтовой бизнес на пороге перемен
4	2019-12-18	нурлан ногаев назначен министром энергетики см...
...
303	2021-03-31	условия получения лицензии на алкоголь предлож...
304	2021-04-01	три сценария развития эпидситуации в казахстан...
305	2021-04-02	в каких трц будут вакцинировать в алматы казат...
306	2021-04-05	почему трц алматы разрешили работать берик шол...
307	2021-04-06	доллар закрыл торги на кассе на отметке тенге в...

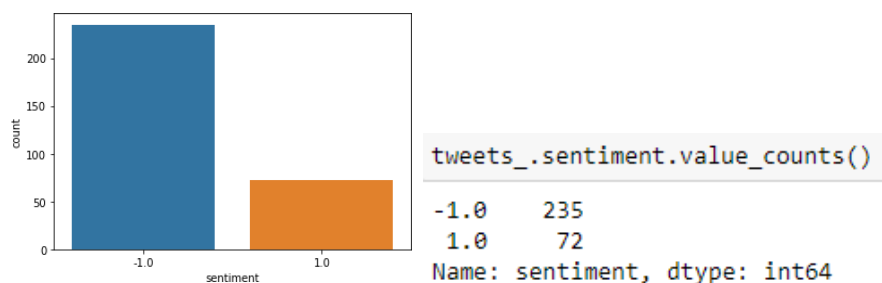
Figure 3.9 – The modified tweets dataset

3.7 - News sentiments analysis

With the news data cleaned, pre-processed and modified to our needs, it can be used as an input for the sentiment analysis model.

The procedure requires us to initialize the sentiment analysis model and process the data through the said model row by row by using a loop. After feeding the data, we can use the output to append it to our tweets dataset.

The output contains the text sentiment score - a float value that denotes the intensity of either sentiment, ranging from -1 to +1. For example, if the news is positive, it should have a sentiment score greater than zero, if the news is negative, the resulting score should be less than zero, and exactly zero, if the news is neutral. Below is the chart with the amount of the scores with either polarity within our dataset.



Figures 3.10 and 3.11 - The distribution of the news sentiments. Here the share of the positive news is equal to 23, 4%

Judging by the chart, the amount of negative news heavily outweighs the amount of positive news in Figure 3.12 .

	tweet	score	sentiment
date			
2019-12-10	доллар закрыл торги на уровне тенге перезагруз...	0.600198	1.0
2019-12-11	максимальный курс доллара в обменниках тенге в...	-0.766304	-1.0
2019-12-12	нацбанк казахстана планирует открыть представи...	-0.592677	-1.0
2019-12-13	лифтовой бизнес на пороге перемен	0.901931	1.0
2019-12-18	нурлан ногаев назначен министром энергетики см...	-0.724880	-1.0
...
2021-03-31	условия получения лицензии на алкоголь предлож...	-0.699264	-1.0
2021-04-01	три сценария развития эпидситуации в казахстан...	-0.692652	-1.0
2021-04-02	в каких трц будут вакцинировать в алматы казат...	0.839744	1.0
2021-04-05	почему трц алматы разрешили работать берик шол...	-0.637041	-1.0
2021-04-06	доллар закрыл торги на kase на отметке тенге в...	-0.743178	-1.0

Figure 3.12 - The tweets dataset after being processed through the sentiment analysis model

3.8 - Modeling

For this part we will use all of the models described earlier, i.e.: SVM, Random Forest, ARIMA and SARIMAX. The data will be split into two parts: train and test sets that will be used to train the machine learning models and evaluate them. The test set will make up the latter quarter of the dataset. The data is not shuffled, since the data includes date stamps, and that makes our project time series-related.

3.8.1 Support Vector Machine (SVM)

In order to use SVM we need to convert our data into ordinal format first which is shown in Figure 3.13.

	date	High
0	737403	2313.41
1	737404	2315.58
2	737405	2321.16
3	737406	2334.04
4	737411	2354.44

Figure 3.13 – The KASE Index data converted to ordinal format

This will allow us to use the model to fit the data and make predictions without any issues. After we implement the model on our dataset, we can see how it performs on the following graph:

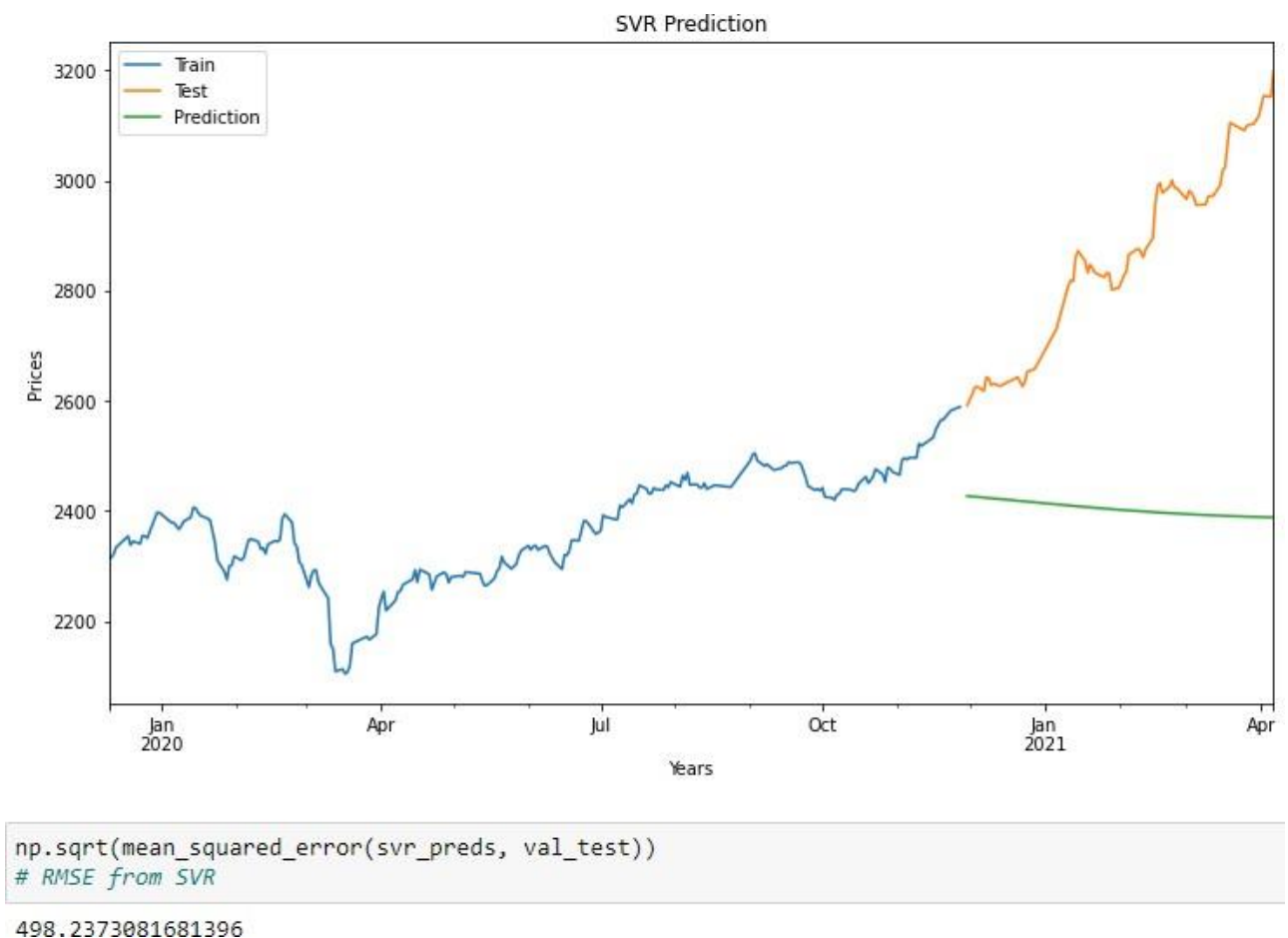
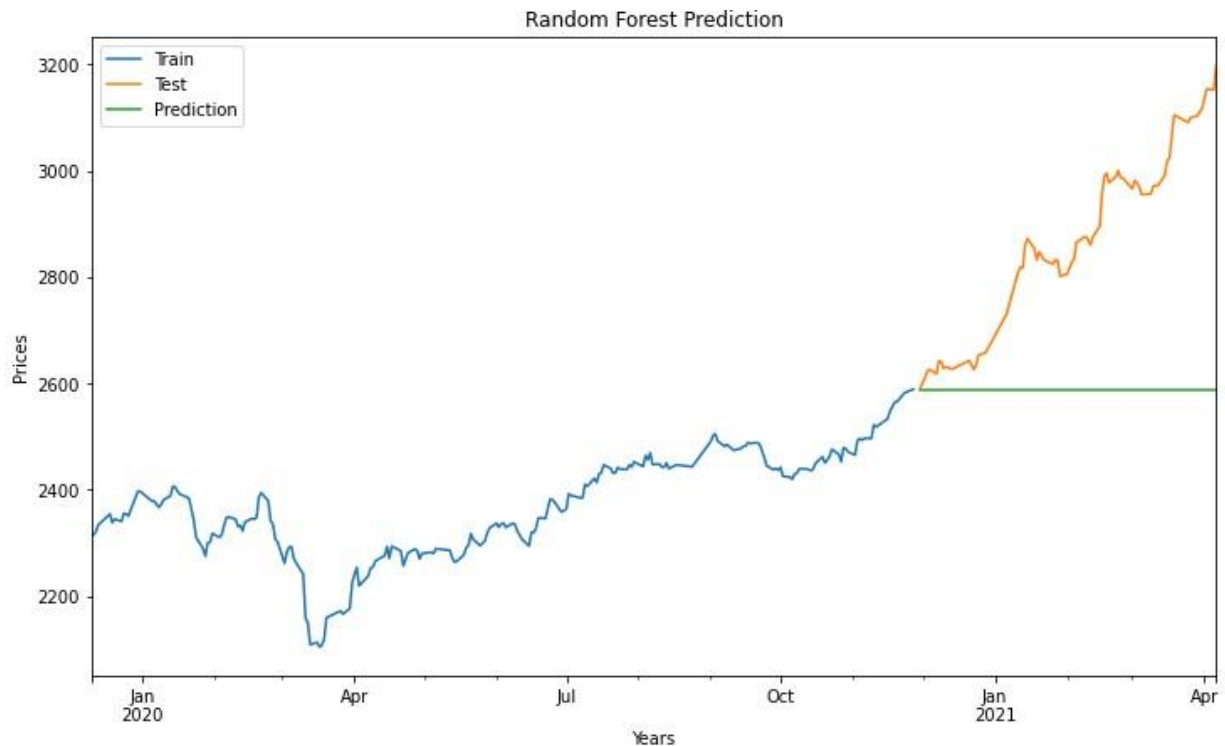


Figure 3.14 – The SVM prediction chart with the RMSE evaluation result

As we can see from the Figure 3.14, the SVM is rather inaccurate. That could be the consequence of the conversion of the data to ordinal format. This result should improve once we move on to time series analysis models.

3.8.2 Random Forest

Although this model is not a time series model, we expect it to be more accurate than SVM for its extensive decision-making capabilities. We will use data that we had converted to ordinal format this time as well. Below is the performance chart of the Random Forest model:



```
np.sqrt(mean_squared_error(rf_preds, val_test))  
# RMSE from RF
```

```
326.43446814102015
```

Figure 3.15 - Random Forest prediction chart with the RMSE evaluation result

This is a fairly more accurate result in comparison to SVM with RMSE score lowered by more than a third, but there are still ways to improve this result.

3.8.3 ARIMA

ARIMA is a time series analysis model. To make it work as effectively as possible, we need to collect additional information: autocorrelation and partial autocorrelation charts, as well as running the model multiple times in order to figure out the most fitting p, d and q parameters of the model.

Here are the autocorrelation and partial autocorrelation charts that would help us find the correct values to feed to our model:

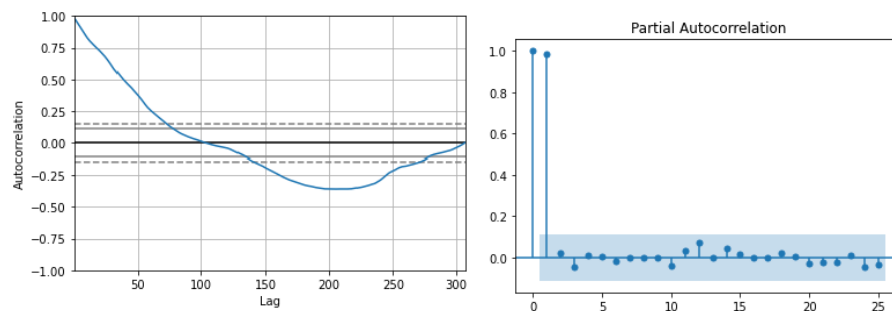


Figure 3.16 - KASE Index dataset correlation plot

Figure 3.17 - KASE Index dataset partial correlation plot

Based on the collected information, we can make conclusions on the optimal parameters for our model: $p=2$, $d=1$, $q=0$.

We have decided to make the model work iteratively, that is, to make predictions based on the train set and on the predictions of the past results. In this case we will have to initialize the model every time it makes a prediction for every row in the test set. It will increase the amount of data for our model to train on, potentially increasing the accuracy of the model.

```

arima_preds = pd.DataFrame().reindex_like(val_test)
for i, d in enumerate(arima_preds.index):
    arima = ARIMA(val_train.append(arima_preds[:i]), order=(2, 1, 0))
    arima_fit = arima.fit()
    arima_preds.iloc[i].High = arima_fit.predict(d)
    print('predicted=%f, expected=%f' % (arima_preds.iloc[i].High, val_test.iloc[i].High))

```

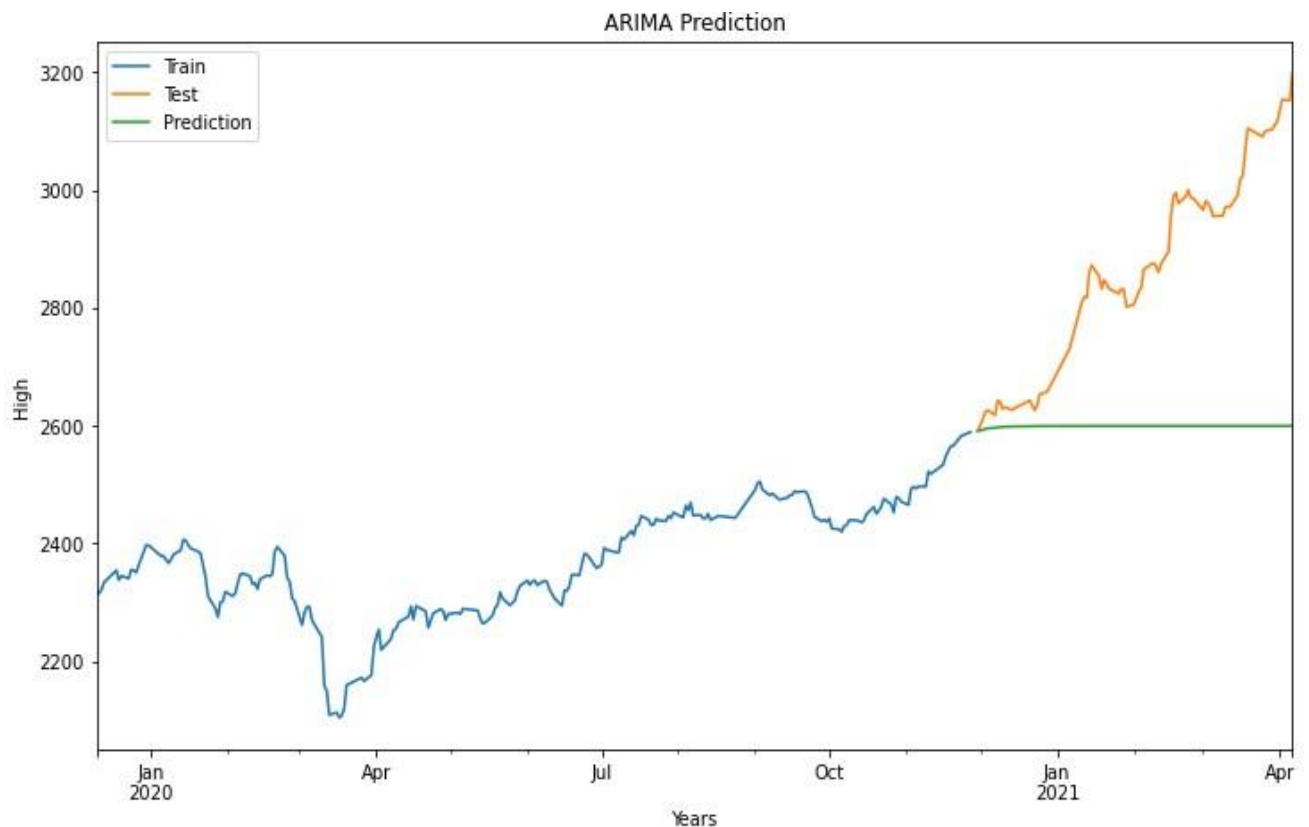
```

predicted=2591.118514, expected=2591.820000
predicted=2592.984729, expected=2610.570000
predicted=2594.365947, expected=2623.010000
predicted=2595.484880, expected=2626.070000
predicted=2596.357003, expected=2618.170000
predicted=2597.048048, expected=2642.850000
predicted=2597.591839, expected=2640.810000
predicted=2598.021047, expected=2628.690000
predicted=2598.359404, expected=2631.250000

```

Figure 3.18 – The iterative ARIMA model code

Here is the chart describing the performance of the ARIMA model implemented iteratively:



```
np.sqrt(mean_squared_error(arma_preds, val_test))
# RMSE from ARIMA
```

315.89031802263924

Figure 3.19 – The ARIMA prediction chart with the RMSE evaluation result

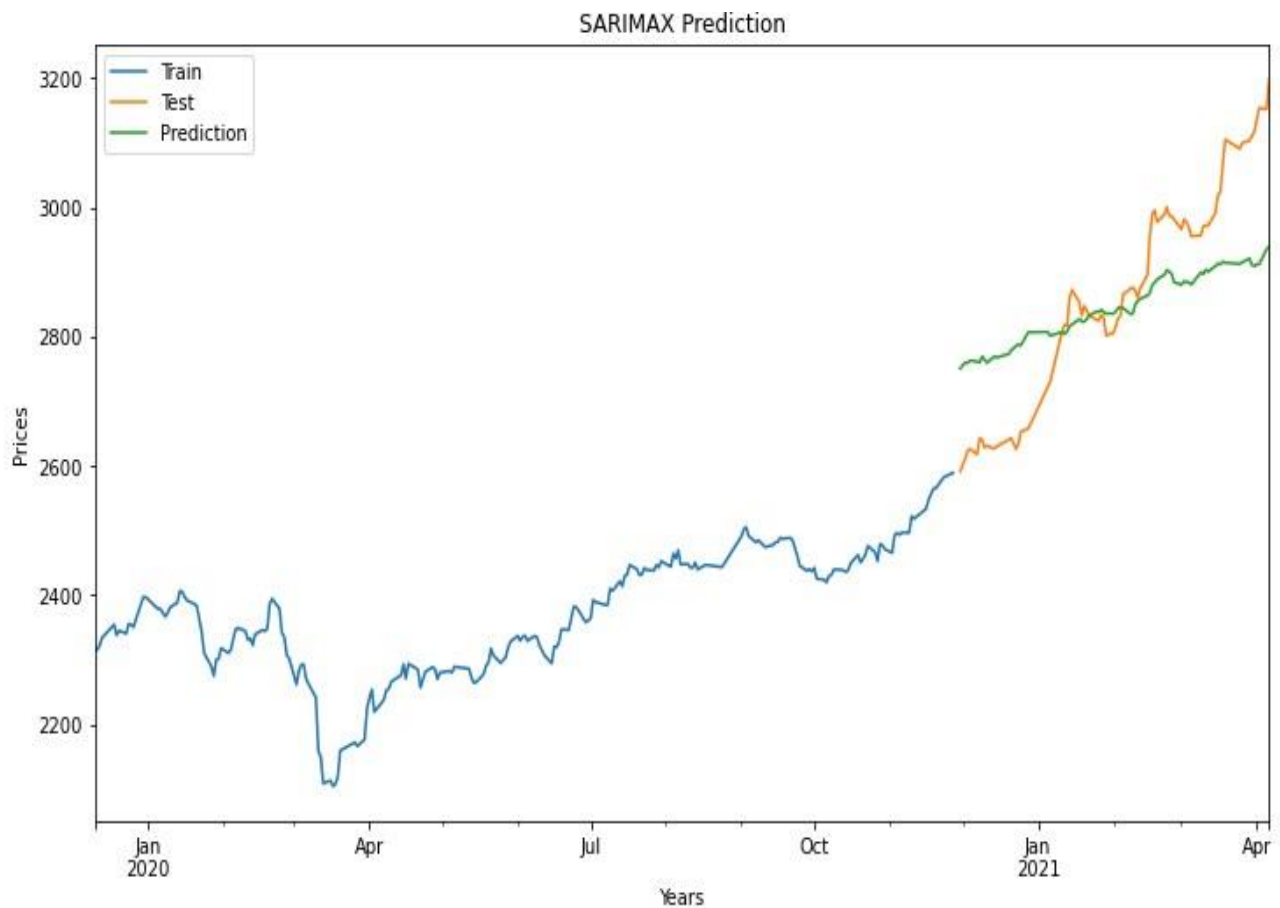
The result is an insignificant improvement over the Random Forest model.

3.8.4 SARIMAX

SARIMAX is an improvement over ARIMA in terms of its functionality in the form of its capability of seasonal analysis.

Like ARIMA, SARIMAX also requires the aforementioned parameters - p , d and q , along with new parameters that will define the seasonality of the data - P , D , Q and m .

We will use the following values for these parameters: $p=2$, $d=1$, $q=0$, $P=1$, $D=1$, $Q=1$, $m=52$. This setup will allow us to capture the weekly trend in our data. Here we can see the performance of the SARIMAX model with the values shown above:



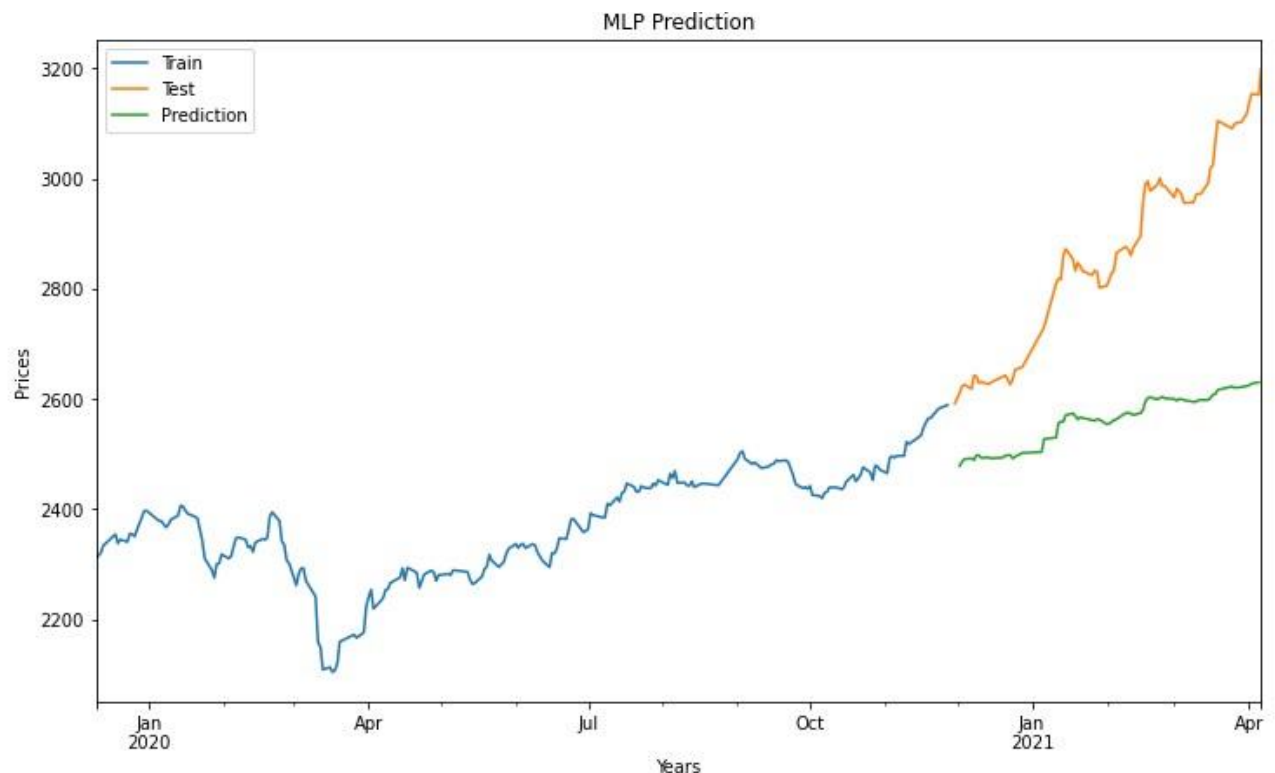
```
np.sqrt(mean_squared_error(sarimax_preds, val_test))
# RMSE from SARIMAX
113.80291787572682
```

Figure 3.20 – The SARIMAX prediction chart with the RMSE evaluation result

As we can see, this is a significant improvement over the earlier models. The seasonality function has helped us to attain even more accurate results.

3.8.5 Multilayer perceptron

The multilayer perceptron is a type of a feedforward neural network. We will be using the version introduced in the Scikit-learn Python library with default parameters, i.e: 100 neurons in each hidden layer, ReLU activation function, Adam weight optimization solver and 200 iterations maximum. Here is the graph that describes the performance of the model:



```
np.sqrt(mean_squared_error(mlp_preds, val_test.iloc[1:-1]))
# RMSE from MLP
```

325.4089134570175

Figure 3.21 – The Multilayer perceptron prediction chart with the RMSE evaluation result

Judging by the RMSE grade, the predictions of the Multilayer perceptron model are the same as the predictions of the model based on the Random Forest algorithm. However, if we look at the graph, we can see that this algorithm captures the trend better than the Random Forest algorithm.

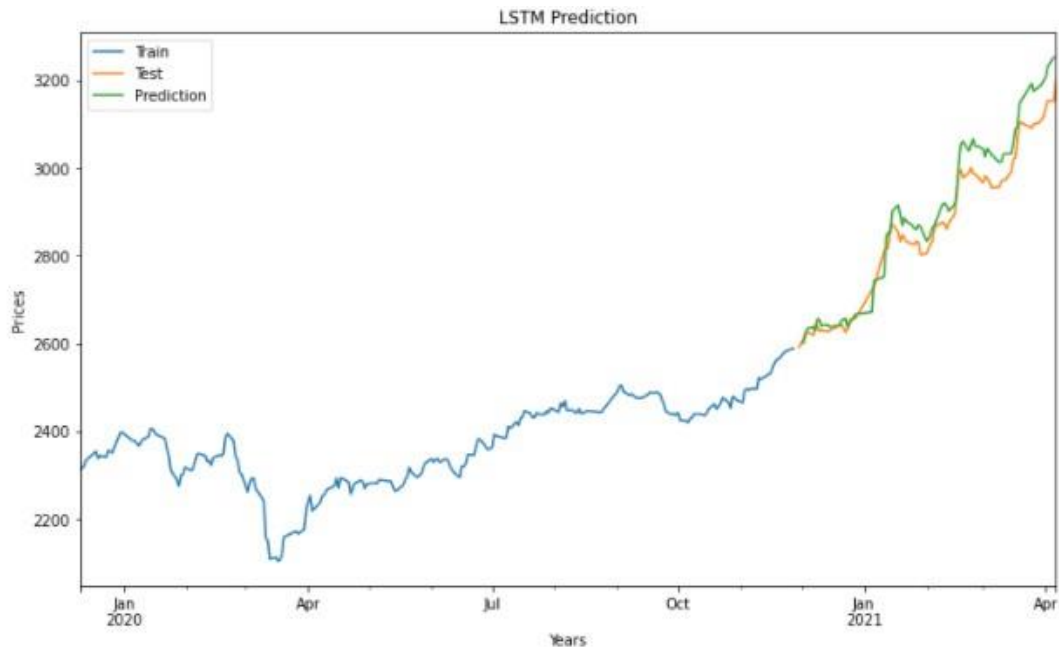
3.8.6 Long-Short Term Memory (LSTM)

For the implementation and construction of the LSTM Neural Network model, we used Keras and Tensorflow libraries.

Processing the data through the LSTM neural network requires us to prepare the data in a different manner. The data has to be split into two in such a way so that the model will try to predict tomorrow's stock value based on the value that we have today. The data also has to be scaled in order to improve the accuracy of the model.

Our LSTM Model will be sequential, having two layers: the input layer with 48 nodes and one output layer with only one node for the prediction. The network will

be optimized with the Adam optimizer with the learning rate of 0.001, using RMSE as the loss function. We also set the amount of epochs and the batch size to 50 and 16 respectively. Below is the result of the LSTM model that was built with the described hyperparameters:



```
np.sqrt(mean_squared_error(__, testY_.T))  
# RMSE from MLP  
48.71520692577153
```

Figure 3.22 – The LSTM prediction chart with the RMSE evaluation result

As we can see, the model yielded very accurate results in comparison to the models described earlier, while also capturing the trend in a precise manner.

Now we will try to further improve the result of the LSTM Neural Network by using the news sentiments that we have calculated earlier.

3.8.7 LSTM with news sentiments

For this model we will be using the dataset with the news sentiments calculated with our news sentiment analysis algorithm - Dostoevsky. The previous setup of the LSTM Neural Network will be the same, with the exception of the batch size reduced to 8, in order to improve the model's generalization capabilities[16]. Here is how the LSTM model performs with the sentiment analysis data:

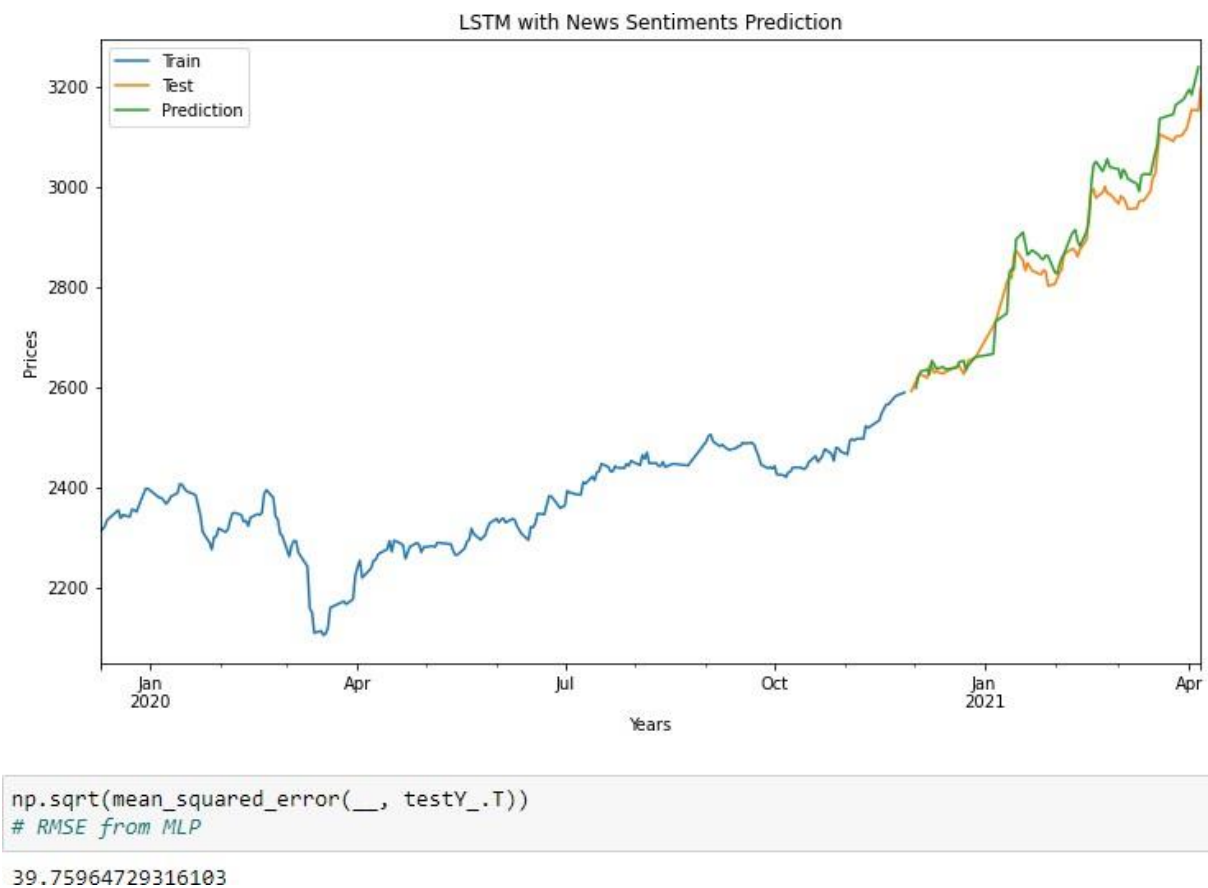


Figure 3.23 – The LSTM with News Sentiments model prediction chart with the RMSE evaluation result

According to the graph and the RMSE score, the model yielded results that are very close to the LSTM model. Also we can notice that the predictions of the model are almost the same except for the moments before the mid-January peak and the steady rise during March. There, we can notice that the predictions are closer to actual values (Table 3.1).

Table 3.1 - The RMSE scores of the models used in the work

Model	RMSE value
SVM	498.237
Random Forest	326.434
ARIMA	315.89
SARIMAX	113.802
MLP	325.408

Table 3.1 continued

LSTM without news sentiments	48.715
LSTM with news sentiments	39.759

Appendix A contains the code used to generate the results shown above.

4. ECONOMIC RATIONALE OF THE PROJECT

4.1 Business idea

The main task of our project is to predict changes in stock prices on the stock exchange market based on the latest news using a machine learning model.

During the search for similar projects many foreign analogues were found, and there is no analogue in the CIS. And among many foreign analogues we have collected the best analogues, which are presented in Table 4.1.

Table 4.1 - Existing projects

Name	The Motley Fool	Bloomberg	Morningstar	Seeking Alpha	UMPINDEX	AIStockFinder
Head Quarters	Alexandria, Virginia, the U.S.	the USA	Chicago, Illinois, the U.S.	New York, NY, the US	the USA	the USA
Price	19\$/month	2,200\$/month	199\$/annually	29,99\$/month	60\$/year	49\$/month
Pros	<ul style="list-style-type: none"> 1. Provides access to expert advice from professionals that have real experience. 2. Allows you to construct real and hypothetical equity portfolios. 	<ul style="list-style-type: none"> 1. Provides useful insights for professional traders and analysts. 2. Bloomberg Terminal maximizes speed and data connectivity. 3. Their free TV keeps you updated with the latest market developments 	<ul style="list-style-type: none"> 1. Find new investment ideas using the most popular screeners for research. 2. Evaluate investment ideas by unveiling buying opportunities across sectors thanks to the unlocked data and ratings. 	<ul style="list-style-type: none"> 1. Ideal for those that are not versed with technical aspects of stock investing. 2. Delivers top-shelf content for free 	<ul style="list-style-type: none"> 1. UMPI's algorithm can track stock market trends that cannot be seen by humans, providing better awareness when analyzing the stock market. 	<ul style="list-style-type: none"> 1. Extracting data from the internet using Data mining. 2. Artificial intelligence algorithm analyses millions of data points to extract patterns.
Web Version	Yes	Yes	Yes	Yes	Yes	Yes

Compared to other analogues on the market, our web application will have the function of sending alerts as the forecast is processed, by email or to the customer's smartphone about the behavior of the stocks. Also the biggest feature is that there will

not be any need for analysts to predict changes in the stock market, because of accurate forecasting of artificial intelligence. The advantage of the web application is also a modern forecasting model that is worth a large investment. We paid special attention to forecasting models and web application design. In the future, it is planned to create an application on iOS and Android and bring it to a competitive level with equally accessible applications on the platform.

An important reason for the development of the project is the lack of direct competitors. The preliminary name of the product is SAP (Stock analyze predictor).

The target audience will be brokers, shareholders, etc. Competitors in the market will be analysts who predict the growth/fall of stock prices; compared to them our project will be able to occupy a niche of forecasts at a lower price policy.

The conditions for the success of our project are:

- More accurate forecasts compared to competing services
- Low price policy due to lower personnel / service costs
- Transparency of business policy.

4.2 Product Description

This web application will allow users to manage and independently analyze the behavior of stocks. A registered user can open a website and request data about any stock or any company, and the system itself will determine and analyze that company, and give results of forecasting which allow making successful investments, sell shares or buy them. If the user has purchased a basic subscription, then he can choose up to 5 different companies, and every morning notifications about the current position of that company and what to do with its shares (if he owns them) will be sent to his mail or phone.

4.3 Market research

As mentioned above, our project will occupy a niche of forecasts with a lower price policy in comparison with traditional companies that provide access to data from professional analysts. Our service will provide a secondary investment recommendation service, occupying a separate role in the arena of stock market analytics.

At the initial stage of project development, it became clear what priority the web-application has in comparison with competitors. A thorough analysis of

competitors was carried out; it was possible to find out what the overall advantages and disadvantages are.

This task is more difficult, since all the conditions for developing the application must be met, that is, in order to find out the conditions, we need to find out the competitiveness.

Our clients will be people interested in the global economy. People who are interested in the stock exchange market, from expert analysts to beginners.

There are no similar analogues in the CIS, so this stage is considered achieved (Refer to table 4.1).

4.4 Market strategy

Our business offices will be located throughout Kazakhstan. There will be several of them in each city, because we need a lot of lighting. Our web app will be free, but there will be a premium version with more features like no ads and better forecasting for monthly fee. While in the basic version, ads sometimes appear. In addition, the premium version will have several unique features. However, even without the premium, the app can be used without problems. The pricing policy will adapt to the current trends in the cost of market analysis.

Our business will be promoted mainly on Google Ads(), which will serve ads on various websites for users who might be interested in using our services.

The SAP(project can become a top product on the market, since there are no such analogues in the CIS, and new features will be developed in this development to increase the comfort of using the web application and increase revenue and rating.

4.5 Financial Analysis

If we consider it from a financial point of view, then the task is unattainable, since it requires much more financial costs. Because renting a powerful server is much more expensive for the budget. However, you can expect that the project will have a payback period.

Profit: We receive revenue from the number of ad views and subscriptions to forecasts, subscriptions will be divided into Basic and Pro.

Revenue from advertising and subscriptions in the first month amounted to 50,000 tenge.

Months:

1st 50 000 KZT
2nd 68 000 KZT
3rd 45 000 KZT
4th 70 000 KZT
5th 81 000 KZT
6th 95 000 KZT
7th 65 000 KZT
8th 45 000 KZT
9th 53 000 KZT
10th 66 000 KZT
11th 70 000 KZT
12th 95 000 KZT

Project evaluation

Time spent:

- Development web applications of SAP system (Stock Analysis Predictor) – 4 months, 10 days, 14 hours, 40 minutes;
- Final readiness of the project – 5 months;

Initial investments for marketing and advertising: 200 000KZT (only once);

Monthly cost:

- Server Rental — 40 000 KZT;
- Advertisement by Google Ads - 20 000 KZT;
- Hosting – 458 KZT;
- Domain – $3500 \text{ KZT}/12 = 291 \text{ KZT}$;

Total will be - 60 749 KZT.

Payback period

This is the time it takes to recoup the additional cash costs. There are two variants available, users can choose between using Basic version with not full features or subscribe to monthly paid Premium version. Because of poor functionality of the Basic version, most people will buy a monthly subscription. Actually the Basic version is used for highlighting our service. Premium subscription costs 12 990 KZT, which is about 30\$. At the initial stages we will invest for the promotion and marketing of our service, first month 200 000 KZT, then 20 000 KZT every month. We are predicting that the number of subscribers for the Premium version will increase progressively, starting from 10 for the first month, then 20 for the second,

30 and so on. Our goal is to increase the number of active users of our service to more than 100 by the 6th month, and we will have enough of an audience of people interested in finance, which makes us able to publish ads on our website. The cost of posting advertisements of one company will be determined by the numbers of active users, 100 000 KZT monthly for 100 active subscribers.

So, there are two sources of income, first is from subscriptions and second is from advertisements.

Calculation of overall profit from ads and subscriptions for months:

Renting equipment costs 60 749 KZT every month;

Salary of administrator - 120 000 KZT every month;

$1^{st} : (+15 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 = 14\,101 \text{ KZT};$

$2^{nd} : (+25 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 = 144\,001 \text{ KZT};$

$3^{rd} : (+20 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 = 79\,051 \text{ KZT};$

In the 3rd month investments for initial promotion will be already covered.

$4^{th} : (+15 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 = 14\,101 \text{ KZT};$

$5^{th} : (+15 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 = 14\,101 \text{ KZT};$

$6^{th} : (+20 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 = 79\,051 \text{ KZT};$

For that period net income will be 79 051 KZT, this is not a lot, but after 6th month will be added profit from advertisements on the website.

$7^{th} : (+30 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 + 100\,000 = 308\,951 \text{ KZT};$

$8^{th} : (+10 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 + 100\,000 = 49\,151 \text{ KZT};$

$9^{th} : (+25 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 + 100\,000 = 244\,001 \text{ KZT};$

$10^{th} : (+15 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 + 100\,000 = 114\,101 \text{ KZT};$

$11^{th} : (+35 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 + 100\,000 = 373\,901 \text{ KZT};$

Starting from the 12th month, the cost of posting advertising will increase to 200 000 KZT per month, because the number of users will be more than 200.

$12^{th} : (+20 \text{ subscriptions} * 12\,990) - 60\,749 - 120\,000 + 200\,000 = 279\,051$
KZT;

In the last 6 months 135 subscribers will be added and net income will be 1 648 207 KZT. Totally for 1st year profit is 1 727 258 KZT.

4.6 Economic effectiveness

Calculation of economic efficiency: $(1\,727\,258 / 380\,749) * 100\% = 453\%$.

CONCLUSION

The aim of the work was to develop an applied text analysis system for predicting market incentives based on historical data with text data in the form of text news. At the end of the work, we can deduct that of the sets of classes of artificial intelligence models, the configuration of the LSTM neural network has shown itself to be the most optimal option when building models for predicting market incentives, and that its main advantage is its adaptability to different general and seasonal trends.

We can come to the conclusion that the LSTM model is capable of determining, with comparatively impeccable accuracy, changes in the value of the share on the KASE Index market, which, in turn, allows potential investors to reduce risks when making trade decisions. However, it cannot be argued that the LSTM is the best among all possible options, since only six predictive models were considered in the study.

It can also be argued that the SARIMAX model showed the best results after the models built using neural networks, with the exception of the Multilayer Perceptron. The SARIMAX model successfully captured the general upward trend in the value of a stock.

As a result of the study, it can be concluded that forecasting through the use of tools based on artificial intelligence is a very effective approach for predicting changes in the value of shares in the KASE Index market.

REFERENCES

- [1] Benjamin B., Rebecca B., Tony O. Applied Text Analysis with Python. — 2018
- [2] L. Stein, “Random patterns,” in Computers and You, J. S. Brake, Ed. New York, NY, USA: Wiley, 1994, pp. 55-70.
- [3] D. C. Montgomery, Design and Analysis of Experiments, 5th ed. New York: Wiley, 2000.
- [4] Stock Price Prediction Using News Sentiment Analysis: <http://davidanastasiu.net/pdf/papers/2019-MohanMSVA-BDS-stock.pdf>
- [5] Sentiment Analysis of Twitter Data for Predicting Stock Market Movements: <http://arxiv.org/pdf/1610.09225v1.pdf>
- [6] B. O. Wyss, —Fundamentals of the stock market, 2000.
- [7] Choudhry R., Garg K. «A hybrid machine learning system for stock market forecasting»
- [8] Contreras J. et al. ARIMA models to predict next-day electricity prices //IEEE transactions on power systems. – 2003
- [9] Hochreiter S., Schmidhuber J. Long Short-Term Memory//Neural Computation, 1997
- [10] Rakhi B., Sher D. Integrating StockTwits with sentiment analysis for better prediction of stock price movement, 2018
- [11] Gibson A., Patterson J. Deep Learning: A Practitioner’s Approach. USA: O’Reilly Media; 2016. p. 81–114.
- [12] S. Hochreiter, J. Schmidhuber. Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.
- [13] B. Johan, H. Mao and X. Zeng, “ Twitter Mood Predicts The Stock Market”, Journal of Computational Science, Vol.2, No.1, pp.1-8, 2011.
- [14] Dostoevsky - Sentiment analysis library for russian language <https://github.com/bureaucratic-labs/dostoevsky>
- [15] Anna Rogers, Alexey Romanov, Anna Rumshisky, Svitlana Volkova, Mikhail Gronas, Alex Gribov - RuSentiment: An Enriched Sentiment Analysis Dataset for Social Media in Russian, 2018
- [16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima arXiv:1609.04836 [cs.LG]

Appendix A

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
import pandas as pd
import numpy as np
import re
from matplotlib import pyplot as plt
import seaborn as sns
import datetime as dt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
# # Data Collection
# In[2]:
import twint
import nest_asyncio
nest_asyncio.apply()
# In[3]:
#configuration
config = twint.Config()
config.Username = "Kapitalkz"
config.Lang = "ru"
config.Since = "2016-01-01"
config.Until = "2021-04-07"
config.Store_csv = True
config.Output = "Kapitalkz.csv"
# In[4]:
#running search
twint.run.Search(config)
# # Data inspection
# ### KASE Index dataset
# In[5]:
kase = pd.read_excel('Index_KASE_210408.xls', header=1)
# In[6]:
kase.head()
# In[7]:
kase.dtypes
# In[8]:
```

```

kase['Date']      =      pd.to_datetime(kase.Date,      format=
'%d.%m.%y')
# In[9]:
vals = kase[['Date', 'High']].copy()
# In[10]:
vals.head()
# In[11]:
vals.shape[0]
# ### Kapital.kz tweets dataset
# In[12]:
kapital = pd.read_csv('Kapitalkz.csv')
# In[13]:
kapital
# In[14]:
kapital.shape
# In[15]:
news = kapital[['date', 'tweet']].copy()
# In[16]:
news.head()
# In[17]:
news.shape[0]
# # Data Modification
# In[18]:
def modify(text):
    text = str(text).lower()
    text = re.sub(r'http\S+', ' ', text)
    text = re.sub(r'https\S+', ' ', text)
    text = re.sub(r'pic.twitter\S+', ' ', text)
    text = re.sub(r'^a-zA-ZА-Яа-я', ' ', text)
    text = re.sub('\s+', ' ', text)
    return text
# In[19]:
modify(news['tweet'][2])
# In[20]:
news_clean      =      news.drop_duplicates(subset=['date',
'tweet'],
keep='first').groupby(['date'])['tweet'].apply(lambda x:
modify(' '.join(x))).reset_index()
# In[21]:

```

```

with pd.option_context('display.max_colwidth', None):
    display(news_clean)
# In[22]:
vals.isna().sum()
# In[23]:
vals[vals.duplicated(keep=False)]
# In[24]:
vals = vals.drop_duplicates()
# In[25]:
vals.nunique()
# In[26]:
news_clean.isna().sum()
# In[27]:
news_clean.duplicated().sum()
# In[28]:
news_clean.nunique()
# In[29]:
news_clean['date'] = pd.to_datetime(news_clean.date)
# In[30]:
news_clean
# In[31]:
dataset = pd.merge(news_clean, vals, left_on='date',
right_on='Date', how='inner')
# In[32]:
dataset
# In[33]:
dataset = dataset.drop('Date', axis=1)
# In[34]:
dataset.isna().sum()
# In[35]:
dataset.duplicated().sum()
# In[36]:
tweets = dataset[['date', 'tweet']]
tweets = tweets.set_index('date')
tweets.index = tweets.index.to_period('d')
# In[37]:
high = dataset[['date', 'High']]
high = high.set_index('date')
high.index = high.index.to_period('d')

```

```

# In[38]:
high.head()
# In[39]:
high.plot(figsize=(12, 7))
# In[40]:
text_train,      text_test,      val_train,      val_test      =
train_test_split(tweets,      high,      test_size=0.25,
shuffle=False)
# In[41]:
val_test.head()
# ## SVM (SVR) Model implementation
# In[42]:
from sklearn.svm import SVR
# In[43]:
reg_data = dataset[['date', 'High']].copy()
reg_data.date =
pd.to_datetime(reg_data.date).map(dt.datetime.toordinal)
# In[44]:
reg_data
# In[45]:
X_train,      X_test,      y_train,      y_test      =
train_test_split(reg_data.date,      reg_data.High,
test_size=0.25, shuffle=False)
# In[46]:
X_train = np.array(X_train).reshape(-1, 1)
X_test = np.array(X_test).reshape(-1, 1)
# In[47]:
model_svr = SVR().fit(X_train, y_train)
# In[48]:
svr_preds = pd.DataFrame().reindex_like(val_test)
svr_preds.High = model_svr.predict(X_test)
# In[49]:
tr = val_train.plot()
ts = val_test.plot(ax=tr)
svr_preds.plot(ax=ts, figsize=(12, 7))
plt.title('SVR Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend(['Train', 'Test', 'Prediction'])

```

```

# In[50]:
np.sqrt(mean_squared_error(svr_preds, val_test))
# RMSE from SVR
# ## Random Forest Implementation
# In[51]:
from sklearn.ensemble import RandomForestRegressor
# In[52]:
model_rf = RandomForestRegressor(n_estimators = 100,
random_state = 0).fit(X_train, y_train)
# In[53]:
rf_preds = pd.DataFrame().reindex_like(val_test)
rf_preds.High = model_rf.predict(X_test)
# In[54]:
tr = val_train.plot()
ts = val_test.plot(ax=tr)
rf_preds.plot(ax=ts, figsize=(12, 7))
plt.title('Random Forest Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend(['Train', 'Test', 'Prediction'])
# In[55]:
np.sqrt(mean_squared_error(rf_preds, val_test))
# RMSE from RF
# ## ARIMA Model Implementation
# In[56]:
from statsmodels.graphics import tsaplots
# In[57]:
pd.plotting.autocorrelation_plot(high)
tsaplots.plot_pacf(high)
plt.show()
# In[58]:
from statsmodels.tsa.arima.model import ARIMA
# In[59]:
arima_preds = pd.DataFrame().reindex_like(val_test)
for i, d in enumerate(arima_preds.index):
    arima = ARIMA(val_train.append(arima_preds[:i]),
order=(2, 1, 0))
    arima_fit = arima.fit()
    arima_preds.iloc[i].High = arima_fit.predict(d)

```

```

        print('predicted=%f,                expected=%f'                %)
(arima_preds.iloc[i].High, val_test.iloc[i].High))
# In[60]:
tr = val_train.plot(title='Train')
ts = val_test.plot(ax=tr, title='Test')
arima_preds.plot(ax=ts, figsize=(12, 7), title='Preds')
plt.title('ARIMA Prediction')
plt.xlabel('Years')
plt.ylabel('High')
plt.legend(['Train', 'Test', 'Prediction'])
# In[61]:
np.sqrt(mean_squared_error(arima_preds, val_test))
# RMSE from ARIMA
# ## SARIMAX Model Implementation
# In[62]:
from statsmodels.tsa.statespace.sarimax import SARIMAX
# In[63]:
sarimax_preds = pd.DataFrame().reindex_like(val_test)
sarimax      = SARIMAX(val_train,      order=(2,      1,      0),
seasonal_order=(1, 1, 1, 52))
sarimax_fit  = sarimax.fit()
# In[64]:
sarimax_preds                                =
sarimax_fit.predict(start=sarimax_preds.iloc[0].name,
end=sarimax_preds.iloc[-1].name)
sarimax_preds = sarimax_preds[val_test.index]
# In[65]:
tr = val_train.plot()
ts = val_test.plot(ax=tr)
sarimax_preds.plot(ax=ts, figsize=(12, 7))
plt.title('SARIMAX Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend(['Train', 'Test', 'Prediction'])
# In[66]:
np.sqrt(mean_squared_error(sarimax_preds, val_test))
# RMSE from SARIMAX
# ## LSTM Implementation
# In[67]:

```

```

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
import tensorflow as tf
import keras
from keras.callbacks import TensorBoard
from sklearn.preprocessing import MinMaxScaler
# In[82]:
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-look_back):
        a = dataset[i:(i+look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return np.array(dataX), np.array(dataY)
# In[76]:
scaler = MinMaxScaler(feature_range=(0, 1))
minmax = scaler.fit_transform(high)
# In[77]:
train_nn, test_nn = minmax[:230, :], minmax[230:, :]
# In[83]:
trainX, trainY = create_dataset(train_nn)
testX, testY = create_dataset(test_nn)
# In[84]:
# reshape input to the correct lstm format
trainX = np.reshape(trainX, (trainX.shape[0], 1,
trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1,
testX.shape[1]))
# In[85]:
def root_mean_squared_error(y_true, y_pred):
    return
keras.backend.sqrt(keras.backend.mean(keras.backend.square(y_pred - y_true)))
# In[86]:
model = Sequential()
# input layer

```



```

model.add(LSTM(units=48, activation='tanh',
kernel_initializer=tf.keras.initializers.glorot_uniform(s
eed=0), input_shape = (trainX.shape[1], 1)))
# output layer
model.add(Dense(1, name="output_layer"))
# RNN compilation
model.compile(optimizer =
keras.optimizers.Adam(learning_rate=0.001), loss =
root_mean_squared_error)
# Using Tensorboard
logdir = "logs"
tensorboard_callback = TensorBoard(log_dir=logdir,
histogram_freq=5, write_graph=True)
# Fitting the RNN to the Training set
model.fit(trainX, trainY, epochs = 50, batch_size = 16,
callbacks = [tensorboard_callback])
# In[87]:
testPredict =
scaler.inverse_transform(model.predict(testX))
testY_ = scaler.inverse_transform([testY])
# In[89]:
__ = pd.DataFrame().reindex_like(val_test.iloc[1:])
__.High = testPredict
# In[90]:
tr = val_train.plot()
ts = val_test.plot(ax=tr)
__.plot(ax=ts, figsize=(12, 7))
plt.title('LSTM Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend(['Train', 'Test', 'Prediction'])
# In[91]:
np.sqrt(mean_squared_error(__, testY_.T))
# RMSE from MLP
# ## Multilayer Perceptron (Feedforward Neural Network)
# Implementation
#
# In[76]:
from sklearn.neural_network import MLPRegressor

```

```

# In[77]:
trainX, trainY = create_dataset(train_nn)
testX, testY = create_dataset(test_nn)
# In[78]:
mlp = MLPRegressor(random_state=0, max_iter=200,
activation='relu').fit(trainX, trainY)
# In[79]:
testPredict =
scaler.inverse_transform([mlp.predict(testX)])
# In[80]:
mlp_preds = pd.DataFrame().reindex_like(val_test.iloc[1:-
1])
mlp_preds.High = testPredict.T
# In[81]:
tr = val_train.plot()
ts = val_test.plot(ax=tr)
mlp_preds.plot(ax=ts, figsize=(12, 7))
plt.title('MLP Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend(['Train', 'Test', 'Prediction'])
# In[82]:
np.sqrt(mean_squared_error(mlp_preds, val_test.iloc[1:-
1]))
# RMSE from MLP
# ## News Sentiments
# In[83]:
from dostoevsky.tokenization import RegexTokenizer
from dostoevsky.models import FastTextSocialNetworkModel
from tqdm import tqdm
# In[84]:
tweets_ = tweets.copy()
# In[85]:
tokenizer = RegexTokenizer()
analyzer =
FastTextSocialNetworkModel(tokenizer=tokenizer)
for i, j in tqdm(enumerate(tweets_.itertuples())):
    result = analyzer.predict([tweets_.tweet.iloc[i]],
k=2)[0]

```

```

        sentiment = -1 if 'negative' in list(result.keys())
    else 1
        score = max(result.values()) * sentiment
        tweets_.at[j[0], 'score'] = score
        tweets_.at[j[0], 'sentiment'] = sentiment
# In[86]:
tweets_
# In[87]:
tweets_.sentiment.value_counts()
# In[88]:
sns.countplot(x=tweets_.sentiment)
plt.show()
# ## LSTM Implementation with News Sentiments
# In[89]:
scaler = MinMaxScaler(feature_range=(0, 1))
minmax = scaler.fit_transform(high)
# In[90]:
train_nn, test_nn = minmax[:230, :], minmax[230:, :]
# In[91]:
trainX, trainY = create_dataset(train_nn)
testX, testY = create_dataset(test_nn)
# In[92]:
#             trainX             =             np.hstack((trainX,
np.atleast_2d(np.array(tweets_[['score',
'sentiment']]))[1:229])))
#             testX             =             np.hstack((testX,
np.atleast_2d(np.array(tweets_[['score',
'sentiment']]))[231:-1])))
trainX             =             np.hstack((trainX,
np.atleast_2d(np.array(tweets_['score'])[1:229]).T))
testX             =             np.hstack((testX,
np.atleast_2d(np.array(tweets_['score'])[231:-1]).T))
# In[93]:
# reshape input to the correct lstm format
trainX             =             np.reshape(trainX,             (trainX.shape[0],
trainX.shape[1], 1))
testX = np.reshape(testX, (testX.shape[0], testX.shape[1],
1))
# In[94]:

```

```

model = Sequential()
# input layer
model.add(LSTM(units=48, activation='tanh',
kernel_initializer=tf.keras.initializers.glorot_uniform(s
eed=0), input_shape = (trainX.shape[1], 1)))
# output layer
model.add(Dense(1, name="output_layer"))
# RNN compilation
model.compile(optimizer =
keras.optimizers.Adam(learning_rate=0.001), loss =
root_mean_squared_error)
# Using Tensorboard
logdir = "logs"
tensorboard_callback = TensorBoard(log_dir=logdir,
histogram_freq=5, write_graph=True)
# Fitting the RNN to the Training set
model.fit(trainX, trainY, epochs = 50, batch_size = 8,
callbacks = [tensorboard_callback])
# In[95]:
testPredict =
scaler.inverse_transform(model.predict(testX))
testY_ = scaler.inverse_transform([testY])
# In[96]:
__ = pd.DataFrame().reindex_like(val_test.iloc[1:-1])
__.High = testPredict
# In[97]:
tr = val_train.plot()
ts = val_test.plot(ax=tr)
__.plot(ax=ts, figsize=(12, 7))
plt.title('LSTM with News Sentiments Prediction')
plt.xlabel('Years')
plt.ylabel('Prices')
plt.legend(['Train', 'Test', 'Prediction'])
# In[98]:
np.sqrt(mean_squared_error(__, testY_.T))
# RMSE from MLP
# In[110]:
val_test.to_json('test_set.json', orient='index',
compression='infer')

```

```
val_train.to_json('train_set.json',          orient='index',  
compression='infer')
```