Saint Petersburg State University
Department of mathematical game theory and statistical decisions

Mayorov Alexey

Master`s thesis

Research of indicative analysis and supervised machine learning methods for predicting stock quotes

Specialization 01.04.02

Applied Mathematics and Informatics

Master`s Program Game Theory and Operation Research

Research advisor,
Pankratova Iaroslavna Borisovna

Saint Petersburg
2023

ASSIGNMENT FOR THE FINAL QUALIFYING WORK

TO THE STUDENT OF GROUP 21.M09-ПУ

Mayorov Alexey

The topic of the work: **Research of indicative analysis and supervised machine learning methods for predicting stock quotes**

Input data: *Financial metrics, K-Nearest Neighbors, Bayes Classifier, Decision Tree, Ridge regression, LSTM model, stock quotes of Intel and HP.*

Structural parts of the work**:**

- *reviewing technical analysis metrics;*
- *research of regression algorithms;*
- *research of classification algorithms;*
- *research of machine learning methods;*
- *preparing a dataset for the study;*
- *conducting experiments using classification algorithms;*
- *conducting experiments using regression algorithms;*
- *conducting experiments using the LSTM algorithm using thinning and without thinning;*
- *analysis of the results.*

SYNOPSIS

STOCK TRADING, FINANCIAL METRICS, K-NEAREST NEIGHBORS, BAYES CLASSIFIER, DECISION TREE, RIDGE REGRESSION, NEURAL NETWORK, LSTM MODEL.

The object of research is algorithms for the technical analysis of financial instruments on the stock exchange, as well as machine learning algorithms that allow predicting the price of a financial instrument. In this paper, a financial instrument refers to an ordinary share.

The objective of the work is to research and compare the methods of technical (indicative) analysis and supervised machine learning for predicting stock quotes.

The most popular technical analysis metrics were studied. Regression, classification and machine learning methods for predicting stock quotes were explored. Data were prepared for each of the methods. Machine learning and traditional technical analysis algorithms for predicting the price of stock quotes were implemented, and their performance was compared.

CONTENTS

Introduction 6
1 The impact of various trends on the stock market 7
2 Basic financial metrics used in technical analysis 11
2.1 Open Price metric 11
2.2 Close price metric 13
2.3 Low price metric 14
2.4 High Price metric 15
2.5 Exponential Moving Average Based Indicator 16
2.6 Average Directional Movement Index 18
2.7 RSI – Relative Strength Index 24
2.8 Stochastic Oscillator 27
2.9 MACD – convergence and divergence of moving averages 29
3 Classification of machine learning models 33
3.1 Deep learning. Neural networks 33
3.2 LSTM network 34
4 Classification problem 35
4.1 kNN – k–nearest neighbors 35
4.2 Decision tree 36
4.3 Naive Bayes 37
5 Regression problem 40
6 Data preparation 45
6.1 Receiving and processing data 45
7 Training models using the scikit–learn library 50
7.1 Regression prediction models for Intel stock prices 50
7.2 Classification prediction models for Intel stock prices 55

7.3 LSTM network 57

Conclusion 61

References 62

INTRODUCTION

Now we live in an amazing era. Decades ago, it was inconceivable that every household would have a personal computer with computing power surpassing that of early rovers. Current laptops can even exceed the processing power of the PERSEVERANCE rover. The innovations that have led to increased processing power are truly unique.

This increase in power has allowed researchers to conduct larger studies in medicine, predict weather forecasts, and calculate the performance of sports teams. New technologies have also transformed previously conservative economic sectors. Computerization and automation have made international bank transfers and trading stocks from home possible. Analysts can now analyze economic events using various tools like Bloomberg.

In recent years, neural networks and machine learning have replaced classic financial algorithms. Machine learning has enabled researchers to solve complex economic problems that were previously unsolvable or difficult. Previously, people used fundamental or technical analysis to predict stock prices, but these methods were mostly manual.

Now, machine learning has taken a big step towards predicting stock prices. However, there are various problems with data collection, reactive influences of news agencies and fabricated incidents that can alter the algorithm's predictions. The main challenge in economic predictions is that they are influenced by a vast amount of loosely related information, making it almost impossible to process these events.

Therefore, an important question arises: is classical technical analysis better or are neural networks better? Would investors be better off relying on oscillators, robots, and charts or on neural networks that base their predictions on exchange data?

1 The impact of various trends on the stock market

The modern world is full of various information. Much of what seemed unattainable a couple of years ago is now the norm. This trend could not but affect the rapidly developing stock market. Until the 80s of the last century, he remained conservative and poorly accepted innovations. However, already in the 21st century, this segment began to actively update its software, introduce new technologies based on artificial intelligence, allow people to trade securities remotely and make the trading process itself the most transparent and understandable to the average person.

However, facilitating securities trading brings with it a new problem: insufficient investor qualifications, resulting in loss of investments. The abundance of financial data that can be collected and accessed through various means, including reports from dealing centers, news releases, brokerage terminals, etc., presents two further problems. Firstly, the data is highly interlinked in its structure. News releases and world events can have significant impacts on stock exchanges, and it is difficult to predict which event will affect the final price of a particular stock. For instance, a power outage caused by a hacker attack on a semiconductor manufacturing plant could lead to a deficit in semiconductor products and upset end consumers who were eagerly awaiting the release of the latest model of a smartphone. Customer dissatisfaction often correlates with investor sentiment, and hence the stocks of the affected companies could decrease in value.

One real-life example of the psychological influence of a crowd on unqualified investors and speculators happened in the case of CD Projekt Red (CDPR), following the release of their highly anticipated game, “Cyberpunk 2077”. The company claimed the game would revolutionize the world of video games, and the numerous delays only added to the hype surrounding it, causing the share price to soar. However, it was later revealed that the developers had been under immense pressure from management to complete the game, and that the final product failed to live up to the expectations set by the company's PR and marketing team [1]. The share price rapidly declined after key access was granted to journalists and investors, causing panic selling among those who had invested in the company. Since the

game's release in December 2020, the share price has slowly started to recover. A graph displaying the changes in share value of the CDPR company from January 4th, 2011 to March 24th, 2021 can be seen in Figure 1.1 [2].

Figure 1.1 – Graph of the stock price of the company “CD Project Red” [2]

Although the game was visually stunning, only users with powerful computers were able to fully appreciate it. CDPR had already released the system requirements for the game, so buyers knew they would need high-end hardware. Once the initial hype wore off, however, players began to notice significant flaws, graphical imperfections, poor optimization, and other problems with the game. As a result, CDPR's stock price began to plummet. The situation was worsened by the fact that the game was almost unplayable on previous generation consoles such as Xbox One S and PlayStation 4. It was a strange decision for CDPR to release such a demanding and poorly optimized game on older consoles. Eventually, gaming marketplaces and CDPR itself launched a refund campaign for purchased and pre-ordered games. This negative news caused investor sentiment to mirror the downward trend in CDPR's shares as indicated in Figure 1.1.

Figure 1.1 also shows that Bulls fueled by the expectations and promises made by CDPR's marketers dominated the company’s stock. Bulls are a common exchange term for market participants who thinks the market, a specific security, or an industry is poised to rise [3]. When “Bulls” prevail in the order book, there are

many orders to buy, there is a shortage, and the price rises. In Figure 1.1, Bulls are marked with a green arrow

After the release of the game, the “Bulls” moved to the camp of the “Bears” and began to actively sell shares (just try to keep part of the investment) or tried to play on the price decrease (“Bears”) [3].

The result of misleading investors, on the part of the company's management and investors, was the loss of investment in this company.

The stock market itself is subject to rapid changes. It is very difficult to predict what will happen to the stock market in general and to stocks in particular in a certain period of time. Significant sums of money can be earned, saved, and lost [4].

In itself, the predictive task to the next point in the time interval is a rather difficult task, like all other data extrapolation tasks [4].

However, many employees of the banking and financial industry can make calculations, including analytical and predictive ones, using data that can be obtained in the public domain. The data can be obtained from various financial aggregators and loaded into the calculation program, or the data set can be obtained from the open source Kaggle [5].

Often, data providers provide information about the data in the form of some data sets, which are tables whose headings are indicated as Close (instrument price at the close of trading), Open (instrument price at the opening of trading), High (the highest share price that was reached on the current day), Low (the lowest share price that was reached on the current day) [3].

Of course, it is reasonable to assume that stocks, funds, options are rather complex financial instruments. Currently, several methods are used to predict the behavior of a trend in the stock market.

The first of them is fundamental analysis [3]. This type of analysis often requires the direct participation of the trader. The main task of fundamental analysis is to find the relationship between various world news, natural disasters, and other events that could potentially affect the price of shares at one time or another.

This method requires a thorough knowledge of the principles of the world economy, and the subject making the analysis must have an analytical mindset. This investment method is mainly used by conservatives and people whose goal is to save and increase money in the long term [3].

The second method is technical analysis. There are sub-items of this type of analysis, technical indicators (indicative, mathematical methods), wave and candlestick analysis, as well as the use of the artificial intelligence method.

However, ordinary people do not have access to complex software systems that would allow bypassing most news sites and popular exchange sites, collect information and then make predictions using neural networks or other technical analysis tools. However, some economists have developed quite interesting metrics that allow researchers to get converted data, which can be used to predict the trend reversal dynamics only for four components: High price, Low price, Close price, Open price. Using them, it is possible to calculate indicators, which can be placed in a recurrent neural network and, with a certain degree of error, get the result of predicting the price of a given financial instrument.

There are also many ready-made technical indicators that can signal the possible occurrence of a particular financial event.

2 Basic financial metrics used in technical analysis

To make accurate predictions regarding stocks, such as trend and price variations, one needs to understand the fundamental and widely used metrics. These metrics are crucial not only for technical analyses but also for effectively processing data that will be fed into neural networks. It is essential for researchers to have a sound understanding of these metrics, including their calculation methods and their significance in predicting favorable outcomes.

2.1 Open Price metric

The opening price is the price at which a security starts trading immediately after the opening of the exchange on a trading day [3, 6]. Taking the NYSE (New York Stock Exchange) as an example, trading on the platform commences every day at 9:30 am ET, except on holidays [6]. The price of the first transaction on the quoted share is the opening price.

The opening price is the most important marker of trading activity on that day. Especially, short-term investors and "scalpers" (speculators who make many transactions in a short period of time) should pay attention to the Open price.

NASDAQ uses an approach that is called "opening cross" [6]. This approach is aimed at determining the best opening price, taking into account the orders that were accumulated overnight. Simply put, it is needed to choose the best price from the order book. Figure 2.1 shows an exchange order book. The best opening price will correspond to the best BID price (BID - an order to buy). In this case, the Open Price will be the price of 10985.

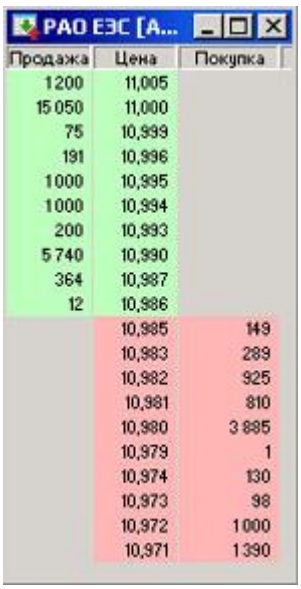

Figure 2.1 – Exchange order book

The essence of the approach used by the NASDAQ is that during non-working hours of the exchange, any economic and non-economic events may occur that may affect the trend and price of shares.

Announcements by the issuing company or news events that come out after the close of the market and stock exchange can greatly influence investor sentiment regarding a given stock. Large-scale natural disasters, wars, or hacks can have a similar, disruptive effect on stock prices.

Figure 2.2 is a chart of Tesla's stock price. The figure shows that the opening price on March 29, 2021 was 608.50 USD [7]. In this case, the chart display is Japanese candlesticks, where the opening price on a particular day can be clearly visually traced.

Figure 2.2 – Tesla company share price [6]

However, not every order that entered the market during non-working hours will be executed during the working day. Many investors are wary of the lack of liquidity for such orders. Lack of liquidity generates wide spreads [3]. When the market opens the next day, or the day after the holidays, the discrepancy between the prices of orders placed on the previous trading day and the orders of the current trading day causes a mismatch between supply and demand for these securities. This causes the price to deviate from the previous day's closing price in the direction of various market forces that act directly or indirectly on this stock [3].

2.2 Close price metric

This metric shows the price at the time of closing the stock market after the exchange day [3]. Many financiers use it as a separate metric completely indirect and separate from other metrics. However, this is a mistake [3]. Such data is highly discouraged from sending to the neural network, because for only one input, the prediction will be very inaccurate, if it happens at all. However, using this tool in combination, at least in the open price, is a much more correct solution, however, even from this data it is impossible to isolate the entire pool of necessary data. Figure

2.3 is a valid example. By extrapolating this image to the price charts for any stock, it is simple to see the open price, close price, low price, high price.

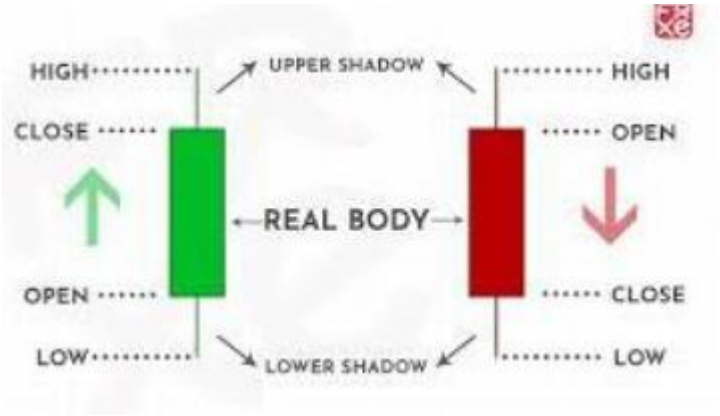

Figure 2.3 – Displaying metrics: open, close, high, low using Japanese Candlestick chart

Figure 2.4 shows the Tesla chart for March 29, 2021. On it, the daily closing price is marked with a red arrow.

Figure 2.4 – Tesla company stock price chart [6]

2.3 Low price metric

Low Price or, as this metric is also called Low Today Price, is the lowest trading price for a given security in the interval from the opening of a trading session (trading day) until the closing of a trading session [3].

As a rule, the Low Price is lower than the opening price and the closing price. This happens usually, quite rarely there is a situation when the lowest price occurs

at the end of the trading session. This metric, as well as High Price, which will be discussed below, have a strong influence on traders who operate with technical analysis. Studying Low Price helps investors and traders understand current and future price trends.

Often, traders use Low Price together with High Price in order to identify so-called “gaps”, gaps or jumps in the stock price [3]. Gaps, on the other hand, are often used in technical analysis to determine the direction of the trend, candle patterns, and so on.

2.4 High Price metric

High price or Today`s High Price is a metric that determines the maximum price for a security during the period of the trading session, that is, between the opening of the market and its closing [3]. Typically, a High Price is higher than the opening price and higher than the closing price of the market. The advantage of High Price is that this metric can be used to calculate a moving average.

Often, High Price can be found on trading or information websites. As a rule, it is placed between the opening price and the closing price. In Figure 2.5, the red arrow shows the closing price, and at the top left there is an underlined “H” value, which just means High Price.

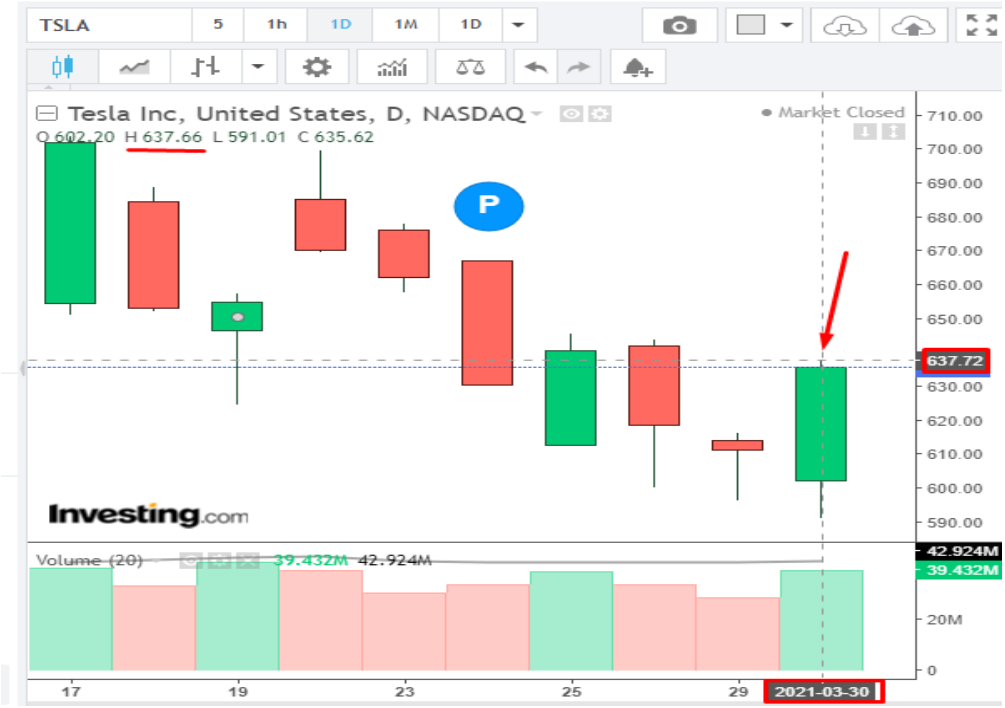

Figure 2.5 – High Price for Tesla company shares

Volatility is a statistical measure of the dispersion of returns for a given security or market index. In most cases, the higher the volatility, the riskier the security. Volatility is often measured from either the standard deviation or variance between returns from that same security or market index [8]. Figure 2.5 shows the daily Tesla security prices where Open Price was $602.20, the Close Price was $635.62, the High Price was $637.66, and the Low Price was $591.01. Judging by the big difference between these prices it can be concluded that Tesla shares are quite volatile, which means they are more attractive for scalpers and speculators than for long-term investors.

2.5 Exponential Moving Average Based Indicator

Exponential moving average is a type of moving average that gives more weight and more meaning to the data. This method is more responsive to price changes than the simple moving average (SMA), which applies equal weights to all observations over a period [3, 6].

The EMA based indicator for closing price is calculated using the following formula:

$$EMAI_{Today} = \left(ClosePrice_{Today} \times \left(\frac{Smoothing}{1 + Days} \right) \right) + EMAI_{Yesterday} \times \left(1 - \left(\frac{Smoothing}{1 + Days} \right) \right),$$

where $EMAI_{Today}$ is today`s exponential moving average based indicator value.

There are many options for smoothing, which is used in the formula above, but the most common is $Smoothing$ which is equal to 2.

Smoothing gives more weight to the last observation in the dataset. It can be seen that as the smoothing factor increases, the later observation has a greater effect on the EMA.

The calculation of the EMA based indicator requires one observation more than the SMA based indicator does. If, for example, 40 is chosen as the number of observations for the EMA based indicator then it is needed to wait up to 40 days to get an SMA based indicator. And the next day (day 41), the previous day's SMA indicator value can be used as the first EMA indicator value of yesterday ($EMAI_{Yesterday}$).

The calculation of SMA based indicator value is very simple. It is just the sum of Close prices for a certain period of time, divided by the number of observations for this period [3].

Next, a smoothing multiplier should be obtained. Smoothing multiplier is calculated using the following formula:

$$Smoothing\ multiplier\ = \frac{Smoothing}{(1 + number\ of\ observations)}.$$

For example, the $Smoothing\ multiplier$ for 40 days, calculated using this formula with $Smoothing$ taken as 2 will be 0.04878.

As a result, the formula $EMAI_{Today}$ for closing price can be written as follows:

$$EMAI_{Today} = Closing\ price \times Smoothing\ multiplier + + EMAI_{yesterday} \times (1 - Smoothing\ multiplier).$$

The main difference between the EMA based indicator and the SMA based indicator is that the weight attached to the last price (yesterday`s price) is greater for the EMA based indicator, while the weight for the SMA based indicator is the same for all values.

2.6 Average Directional Movement Index

The Average Directional Movement Index, or simply the Average Directional Index, is an oscillator or an oscillating indicator. Its popularity is due to the fact that it measures the trendiness of a market or a stock [9].

J. Wells Wilder, entitled "New Concepts in Technical Trading Systems" [9], introduced the concept that became the basis for this indicator back in 1978 in the book. To this day, many traders follow the trend very closely. Trend followers hope that by dividing into different markets and stocks of different companies, they can find a market or security that is trending enough to cancel out hypothetical failures in other securities. The problem is that trends that are suitable for investing assets for the long term (long investments) occur very rarely, although they certainly exist.

The main problem with trend following is that trades can, in a short period of time, change their trend, which leads to a loss of money in the near term. Another problem is that when following a trend, there is no possibility of holding profits. These two problems are what the ADX indicator is trying to solve. This indicator tries to determine the current market trend for the last n days in one value [9]. If the value of the ADX index is high, then the market is in a trend, and if it is low, then the market is not in a trend, but in the so-called congestion.

The theory behind this index is based on the direction of the market. And this move is either positive (bullish) or negative (bearish). Further, positive movement will be set as plus DM, and negative movement as minus DM. This indicator is calculated according to the following scheme. The difference between the lows of two adjacent bars is compared, as well as the difference between their highs. plus DM is obtained when the high of the current bar minus the high of the previous bar is greater than the low of the previous bar minus the low of the current bar. The

resulting value will then be equal to the current high minus the high of the previous bar, while this value is positive, if it is negative, then plus DM is set to zero. Minus DM makes the same comparison. In this case, the difference between the two minima is assigned minus DM as long as the value is positive. If the value is negative, it is assigned zero, as well as plus DM [9]. Figure 2.6 shows the visual patterns of the algorithm presented above.

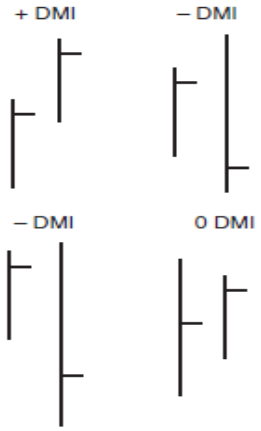

Figure 2.6 – Visual pattern of plus DM and minus DM

Formula (2.1) calculates the absolute price movements:

$$+M = High_{t-1} - High_t - 1, -M = Low_{t-1} - Low_t, \quad (2.1)$$

where $High$ – maximum bar price;

Low – minimum bar price;

t – current bar;

$t-1$ – previous bar.

Then from formula (2.1) one can obtain the following expressions:

$$+DM = \{+M, if\ +M > -M\ and\ +M > 0 \\ 0,\ if + M < -M\ and\ +M < 0\} \quad (2.2)$$

$$-DM = \{0, if\ -M < +M\ or\ -M < 0 \\ -M,\ if\ -M > +M\ and\ -M > 0\} \quad (2.3)$$

From expressions (2.2) and (2.3) the so-called positive and negative directed motions are obtained. After that, directional indicators are calculated:

$$+DI = EMAI_n\left(\frac{+DM}{TR}\right), -DI = EMAI_n\left(\frac{-DM}{TR}\right),$$

where $EMAI_n$ – exponential moving average index for the period n;

$+DM$ – positive direction of movement;

$-DM$ – negative direction of movement;

TR – true range.

TR shows the the maximum range of prices for transactions [9]. TR itself is calculated as follows:

$$TR = max(High_t, Close_{t-1}) - min(Low_t, Close_{t-1}),$$

where $High$ - maximum bar price;

Low – minimum bar price;

t – current bar;

$t-1$ – previous bar;

$Close$ – bar closing price;

max – the largest value of the considered;

min – the smallest value of the considered.

Finally, the indicator itself is calculated as follows:

$$ADX = 100 \times EMAI_{14}\left(\frac{|+DI - -DI|}{|+DI + -DI|}\right).$$

It is also worth saying that, like all oscillating indicators, its values range from 0 to 100 [3, 9]. Usually, along with the ADX indicator, the DI indicators are displayed simultaneously. Plus DI is shown in blue and minus DI is shown in orange. The ADX indicator itself is displayed in red. Also, a 14-day period is used as a time period for the moving average.

Before demonstrating the principle of this metric, it is worth noting that the ADX indicator is non-directional, that is, it will register the strength of the trend, regardless of which direction the price of a particular stock moves.

Plus DI, minus DI and ADX charts are presented in Figure 2.7.

Figure 2.7 – Indicators chart

Figure 2.7 illustrates that when the positive direction indicator (plus DI) surpasses the negative direction indicator (minus DI), it indicates a natural rise in stock prices (as can be seen in the first blue box). The ADX indicator (red) should also be taken into consideration. In the first rectangle, there is a significant spike indicating a high level of strength in the uptrend. At this stage, it was possible to invest in these shares. Conversely, in the second (right) blue box, the negative indicator prevails over the positive indicator, but the strength of the trend is relatively small and does not have a stable trend.

For this indicator, there are boundaries that can show how strong the trend is. These boundaries are described in Table 1.

Table 1 – Description of the strength of the trend, depending on the indicator values

ADX value	Trend strength
0–25	Missing or weak trend
25–50	Strong trend
59–75	Very strong trend
75–100	Extremely strong trend

Many traders do not use trend trading strategies when the strength of the trend is less than 25 points.

In figure 2.8, one can see the so-called “accumulation” phenomenon. Accumulation means that the strength of the trend is lower than 25 points, in more than 30 bars. The advantage of this pattern is that the price movement, after breaking through the 25-point mark, is easier to identify [9, 10]. In addition, at this moment it is important to follow the direction indicators. In Figure 2.8, they show inconsistent data on the interval that is in the blue rectangle, but then, at the moment when the trend strength goes beyond 25 points, the positive indicator of direction begins to prevail sharply, over the negative indicator. After that, the ADX indicator breaks through the border of 25 points and gains strength.

Figure 2.8 – The phenomenon of “Accumulation” on the chart

According to Figure 2.8, it can also be concluded that the price of a security moves (increases or decreases) if the trend strength increases, and vice versa, if the trend strength decreases, then the price enters a correction period.

Many novice investors mistakenly associate the strength of a trend with the reversal event of this very trend, which is not true. The weakening of the trend does not mean a trend reversal [10]. If the ADX indicator is above 25 points, and if the ADX indicator falls, it only means that the trend strength is decreasing [10].

A clear illustration of how the trend behaves on the ADX indicator is shown in Figure 2.9.

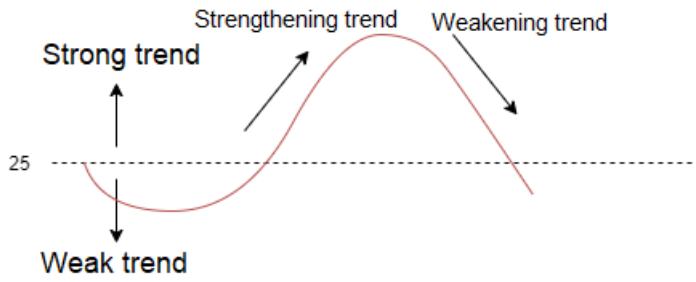

Figure 2.9 – Description of the trend movement by the ADX indicator

Another important phenomenon that can occur on this indicator is called the divergence phenomenon. This is a multidirectional price movement, and the strength of the trend between two trend tops [9, 10]. Figure 2.10 shows the phenomenon of divergence. In general, the phenomenon of divergence is not a signal for a reversal, but rather a warning that the trend is changing.

In figure 2.10 two peaks of the trend chart can be also seen. These peaks show us the momentum of the price, that is, its speed. A series of higher ADX peaks signifies an increase in trend momentum. A series of lower ADX peaks means that the momentum of the trend is decreasing.

Any ADX peak above 25 is considered strong, even if it is a lower peak.

Figure 2.10 – The phenomenon of divergence on the chart of the ADX indicator

Knowing when the momentum of a trend is increasing gives the trader confidence that profits will rise and not exit before the trend ends. However, a series of lower ADX peaks is a warning to watch the price and manage risk.

It is worth using ADX in conjunction with other indicators and strategies. As an independent trading strategy, the ADX index is not suitable. ADX gives excellent signals in combination with the price. The ADX should be used first to determine if prices are trending or not, and then the appropriate trading strategy should be chosen for the given condition. In trend conditions, entries are made on pullbacks and are taken in the direction of the trend. In a range ("sideways") trend trading strategies are not suitable.

2.7 RSI – Relative Strength Index

The Relative Strength Index (RSI) is a momentum indicator that is used in technical analysis. This indicator measures the magnitude of recent price changes to assess the conditions for an overbought or oversold event in a stock's price. RSI is an oscillating chart, which, like ADX, moves between 0 and 100 [3].

The essence of this indicator is that if the chart has a value of more than 70 points, then the security is overbought, and a trend reversal or price correction may soon begin. Conversely, the value of the RSI indicator is less than 30 points or lower, may indicate that the paper is oversold and also mean that a trend reversal may soon begin. The calculation takes place according to the following formula:

$$U = close_{today} - close_{yesterday}, D = 0. \quad (2.4)$$

It can be seen from formula (2.4) that both positive (U - up) and negative (D - down) price changes are used to calculate the RSI index [2]. Also, formula (2.4) means that the day is "up" or "positive", when the closing price today is higher than the closing price yesterday. Conversely, a day is said to be negative if today's closing price is lower than yesterday's closing price, as shown in formula (2.5).

$$\begin{aligned} U &= 0, \\ D &= close_{yesterday} - close_{today}. \end{aligned} \quad (2.5)$$

After that, the values of U and D are smoothed using an exponential moving average with period n. This formula is presented below:

$$RS = \frac{EMAI_n(U)}{EMAI_n(D)},$$

where RS – relative trend strength value.

After that, the final formula for calculating the RSI index can be derived (formula (2.6)):

$$RSI = 100 - \frac{100}{(1 + RS)}. \quad (2.6)$$

It is possible that the denominator in the RS calculation formula can become equal to 0. This happens if the simple moving average based indicator (SMAI) instead of the exponential moving average based indicator (EMAI) is used [3]. In this case, RSI = 100.

As a result, formula (2.6) can be rewritten in the form (2.7) or (2.8):

$$RSI = 100 - \frac{100}{\left(1 + \frac{EMAI_n(U)}{EMAI_n(D)}\right)} \quad (2.7)$$

$$RSI = 100 \times \frac{EMAI_n(U)}{EMAI_n(U) + EMAI_n(D)}. \quad (2.8)$$

After that, the calculated data can be plotted on the chart and separated by lines at the border of 70 points and 30 points. The RSI will rise as the number and volume of positive trade closes increase and will decrease as the number and volume of losses from trades increase.

However, one should make a remark and say that when the “ceiling” of 70 points is broken, the trend may remain upward and not change for a long time, and when the “ground” is broken at 30 points, the trend may be downward.

Also, since the chart is oscillating, it is quite possible that the true oversold value may be above 30 points, and the overbought value may be well below 70 points [3].

The concept behind using overbought or oversold levels that match the trend is to focus on trading signals and techniques that match the trend. In other words, using bullish signals when the price is in a bullish trend and bearish signals when the stock is in a bearish trend will help avoid many of the false signals that the RSI can generate.

Figure 2.11 shows a bullish divergence. It occurs when the RSI displays oversold readings (below 30 pips) followed by a higher low that corresponds to lower price lows. This indicates that the value of the security has decreased and investors have begun to show interest in it. This can serve as a signal to enter a long position.

Figure 2.11 – RSI Divergence for Intel stocks

A bearish divergence occurs when the RSI creates an overbought value (above 70 points) followed by a lower high corresponding to higher price highs [3, 10].

However, it is worth remembering that the divergence that can be tracked by the RSI index is quite rare, and therefore other strategies should be used.

There is another strategy for using RSI, this strategy is called Bullish Rejection of RSI Shift. The strategy pattern consists of several parts:

1) RSI enters the oversold zone (below 30 points);
2) RSI crosses above 30 points again;
3) RSI forms another fall without crossing the oversold line (30 points);
4) RSI then breaks its most recent high.

This pattern can be seen in Figure 2.12. It shows a chart of AMD stock prices. In the same location of the graph, there is a bullish divergence and a shift rejection which indicates a robust upward trend. This presents an opportune moment to purchase a security for a long-term position.

Figure 2.12 – AMD Bullish Rejection Pattern

Like divergences, there is a bearish version of the swing rejection signal that looks like a mirror image of the bullish version [3, 10]. The deviation of bearish swings also consists of four parts:

1) RSI rises into overbought territory;
2) RSI crosses below 70% again;
3) RSI forms another maximum without crossing back into overbought territory;
4) RSI then breaks its most recent low.

2.8 Stochastic Oscillator

A stochastic oscillator is a momentum indicator that compares a specific closing price of a security with a range of prices for that security over a given period of time. The sensitivity of the oscillator can be reduced by adjusting the time period. This oscillator is also used to generate overbought or oversold trading signals. The stochastic oscillator also uses a range of values from 0 to 100 [3].

The formula for calculating the stochastic oscillator is as follows:

$$\%K = \left(\frac{C - L_{14}}{H_{14} - L_{14}}\right) \times 100,$$

where C – latest closing price;

L_{14} – the lowest price of a stock in 14 days of the trading session;

H_{14} – the highest price of a stock in 14 days of the trading session;

$\%K$ – current value of the stochastic indicator.

There are two lines on the chart of this indicator. The first line is %K, smoothed out by the moving average. Another line that is plotted on the chart is %D. %D is the line that is obtained by smoothing the 3-period moving average from the %K line. Often these lines are called as follows: %K - fast, %D - signal.

The theory underlying this indicator is that in an upward (upward, bullish) market, prices close near the high, and in a bearish market, prices close near the low [3, 9]. Market signals are formed when the %K fast line crosses the %D signal line. For example, a buy signal may occur if %K crosses %D from the bottom-up, and a sell signal can form if %K crosses %D from the top-down. The signal becomes more stable if it occurs in overbought or oversold zones. Figure 2.13 shows these two phenomena. In the red rectangle at label 1, there was a top-down intersection of %K, the %D signal line, which led to a downtrend. In mark 2, there was a bottom-up suppression, which caused a trend reversal and an increase in the share price.

Figure 2.13 – Stochastic Oscillator Indication for Tesla Stocks

Like the RSI indicator, this indicator shows overbought levels (above 80 points) or oversold levels (below 20 points). But here the following should be taken into account. As mentioned earlier, breaking through the overbought or oversold

level does not always signal us that a trend reversal will be observed soon. It is necessary to use the stochastic oscillator together with the ADX indicator, which shows the strength of the trend.

Divergence for this oscillator will occur when the price of a security reaches a new low, and the low on the oscillator is higher than the previous one. Conversely, a bullish divergence can occur when the price reaches a new high, and the high on the stochastic oscillator is lower than the previous one [3]. This bullish divergence phenomenon may mean that the momentum of the trend is waning and that a trend reversal can be expected soon.

2.9 MACD – convergence and divergence of moving averages

The next metric, which is actively used by both novice investors and experienced financiers, is called MACD. This acronym stands for "Moving Average Convergence Divergence". This indicator, like most indicators on the stock exchange, is a trend indicator [3]. MACD shows the ratio of two moving averages to the price of an asset.

Calculated as follows:

$$MACD = EMAI_{12} - EMAI_{24},$$

where $EMAI_{12}$ – 12-period exponential moving average based indicator value;

$EMAI_{24}$ – 24-period exponential moving average based indicator value.

The result of this calculation is the MACD line, after which a 9-day EMAI line is superimposed on it - the "signal line". The essence of this line is that, under certain circumstances, it can serve as a trigger to buy or sell a security. Often the tactics of using this signal is simple. If the MACD crosses the signal line from the bottom up, then the security is worth buying, and if the MACD crosses the signal line from the top down, then the security should either be sold or shorted.

The signal line is calculated as follows by the following formula:

$$Signal = EMAI_9(EMAI_{12} - EMAI_{24}),$$

where $EMAI_9$ – 9-period exponential moving average based index value.

The chart of the MACD indicator itself is shown in Figure 2.14.

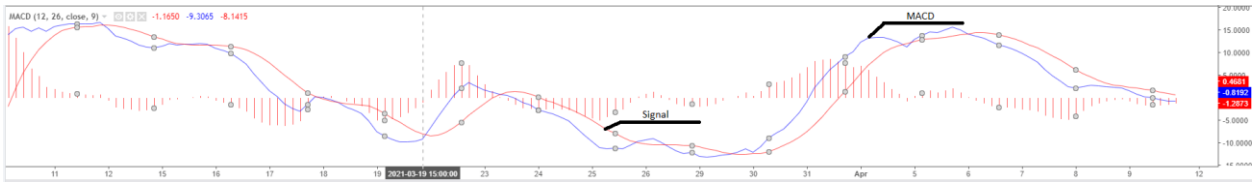

Figure 2.14 – Tesla stock MACD chart

The MACD itself is shown as a blue line, and the signal line is shown as a red one. The histogram in Figure 2.14 shows the difference between MACD and Signal.

MACD can be greater than zero or less than zero. If the MACD value is above zero, then the bullish trend prevails, if it is below zero, then the bearish trend is working. An illustration of this statement, presented in Figure 2.15.

Figure 2.15 – Using Zero MACD for Tesla Stocks

The first large rectangle displays a downward trend and MACD below zero, while the following rectangle indicates an upward trend and MACD above zero. The histogram provides valuable information about zero line crossings, with the column reflecting when the graph crosses the zero mark.

However, by right the most common metric in the MACD indicator is divergence. Its essence lies in the fact that when the price forms a new, largest maximum, and the MACD forms a smaller maximum, this can serve as a signal for a weakening of the trend and its subsequent reversal. But, important remarks, is that

divergence should be looked for only when the trend becomes weak (for this the ADX indicator should be used), during a correction or “sideways”, MACD divergence is not effective and should not be used. Figure 2.16 shows a combination of using the MACD and the ADX oscillator. It can be seen that the price has a second high, while the MACD, at this time interval, has set a maximum that is less than the previous one. But as mentioned earlier, this signal should be interpreted as “correct” only if the trend, in the area of divergence, is weak enough. On this chart, to determine the strength of the trend, the ADX indicator was used, which was used earlier. Table 1 shows that the ADX oscillator value is below 25 points in the MACD divergence section, indicating a weak trend and validating the MACD signal. After this divergence, the share price collapsed.

Figure 2.16 – Divergence of the MACD indicator with ADX reinforcement

MACD also often uses a technique called “crossover” (English crossing). In contrast to the method of simple intersection, which was described earlier, crossover differs in that to confirm a trend signal, it is necessary to increase the strength of the trend and the absence of overbought or oversold in the market [3, 10]. Figure 2.17 shows how the MACD indicator interacts with the ADX and RSI indicators to check their own signals for validity.

Figure 2.17 – Crossover using MACD, ADX, RSI indicators on Intel stocks

The analysis of Figure 2.17 reveals that on the 16th day of the month, the MACD chart intersected with the signal-chart, indicating a potential trend reversal from bearish to bullish. This was further confirmed by a weak trend, and subsequently, the price dropped on the 17th day, leading to a short-term bearish trend. On the 31st day of the month, a crossover event occurred when the signal line crossed the MACD indicator on the MACD chart, resulting in a strengthening trend as seen on the ADX chart. Additionally, the RSI chart indicated the absence of overbought or oversold conditions. Further, the strength of the trend continued to increase, and the MACD is higher than the signal line. This is a clear sign that now is a good time to enter into a “long” purchase, that is, to buy securities in the hope that their price will rise [3, 10].

When used in isolation, indicators can generate a significant amount of data that may be contradictory. Hence, it is advisable to utilize them in combination for optimal results. Upon receiving a signal from one indicator, it is crucial to cross-check with the others. If the original indicator's findings are backed by at least one more indicator and there are no discrepancies among the other indicators, it is appropriate to execute a trade.

3 Classification of machine learning models

Artificial intelligence (AI) is developing in the world with renewed vigor. AI-based technologies are now used in many areas of sociology, engineering, economics and other sciences. The ability to teach a computer to distinguish any objects by certain signs, to search for a malfunction based on the noise emanating from the machine, is amazing. However, there is no magic, and machine learning is just an algorithm or set of algorithms that can perform one monotonous task. In this paper, the emphasis is on classical machine learning, but there are several more classes of machine learning, these are:

- ensembles of algorithms;
- reinforcement learning.

3.1 Deep learning. Neural networks

What is a neural network. For many, the brain immediately comes to mind, and the connection of neurons within it. Figure 3.1 shows a mouse cerebral cortex neuron illuminated with green fluorescent dye [11].

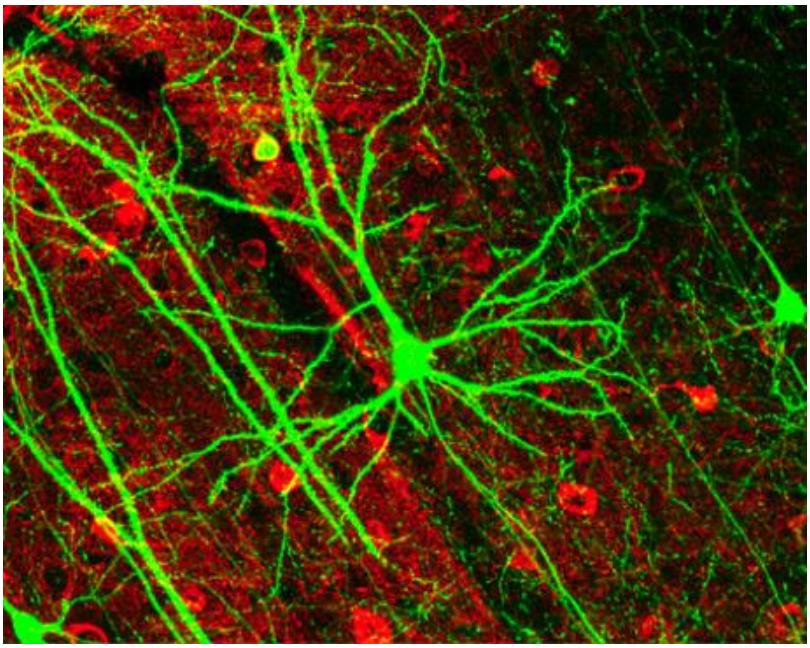

Figure 3.1 – Mouse brain neuron [11]

Although it is not currently possible to cultivate neurons capable of computation, it is feasible to replicate their natural functions in a computer by

developing a mathematical algorithm based on the mathematical model of a human brain neuron. By itself, a neuron cannot do anything useful. Its natural function is to transmit and process various signals sent to it by the body. But when combined into so-called layers, neurons can perform a larger amount of specific computational actions [11].

3.2 LSTM network

One of the most popular stock price prediction algorithms is the LSTM algorithm. This abbreviation stands for Long Short-Term Memory [12].

Long short-term memory is, by definition, a type of recurrent neural network architecture proposed in 1997 by Sepp Hochreiter and Jurgen Schmidhuber. It belongs to the class of so-called universal neural networks. Well suited for the analysis of time series, respectively, well suited for the analysis of stocks on stock exchanges [12].

Of the architectural features, one can single out such important details as the ability of the network to remember values, both for relatively short and for fairly wide periods of time. This possibility is ensured by the fact that the LSTM model does not use the activation function inside its recurrent components [13].

LSTM networks can be represented as a chain of recurring ordinary neural networks, but repetitive modules have a much more complex structure [13]. For example, instead of a single-layer neural network, they have a four-layer one organized in a specific way [12].

However, in order to be able to use LSTM, data must be properly prepared, because only three-dimensional data can be placed in an LSTM network [13].

4 Classification problem

Classification is basically the process of predicting a class for given data. That is, the purpose of classification is to predict the relation of any object to a certain class. Classes are often also referred to as targets or labels. Classification is also an approximation of the mapping function *f* from input variables *X* and discrete output variable *y* [14].

For example, the task of classification will be the recognition of animals in a photo, the detection of spam in e-mail and, if going indirectly to the field of economics, then the binary impact of news on the trading trend of the market in general or a particular security in particular [14]. In the case of trend detection, the classification will be binary. The trend is up or the trend is down. The classifier uses the data to understand how the given input variables relate to the class, how this or that news affects the trading trend [15].

There are the following classification methods: «k–nearest neighbors», «decision Tree», «Naive Bayes».

4.1 kNN – k–nearest neighbors

This algorithm is quite simple and easy to implement supervised learning algorithm. Refers to “lazy algorithms”. The “lazy algorithm” accumulates the entire amount of information about the previous values, so the problem of a possible lack of memory in “lazy algorithms” is high [16]. A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that uses labeled inputs to learn a function that produces the appropriate output when given new, unlabeled data. Applying this concept to the economy, specifically in the realm of securities trading, teaching a novice employee how to identify indicative patterns using the conditional ADX algorithm can be envisioned. Initially, the beginner is shown the accurate pattern and provided an explanation of how it operates. Subsequently, an incorrect pattern is exhibited, and statement that it is not appropriate is given. Through this training, the employee can eventually differentiate between various patterns on the indicator.

The output classification, as mentioned earlier, receives some discrete value, for example, “correct pattern”, “incorrect pattern” [16].

The block diagram of the algorithm itself is as follows:

1. Downloading data.
2. Initializing the value of K with the selected number of neighbors.
3. For each data sample, it is needed to:
    1) calculate the distance between the target object and the objects from the training sample;
    2) add the distance and index of the target object to the ordered collection.
4. Sorting an ordered collection of distances and indices from smallest to largest by distances [16].

After which an object class is obtained, an object based on the most frequently occurring among k nearest neighbors.

4.2 Decision tree

As the name implies, decision tree creates classification models using a tree structure that consists of “if-else” rules and answers based on the initial question [16].

The if-else rules are executed sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed [17]. This process continues on the training set until the termination condition is met.

The tree is built in a recursive divide-and-conquer fashion from top to bottom. All attributes must be categorical [17]. Otherwise, they must be discretized beforehand. Attributes at the top of the tree have more influence on the classification and they are identified using the concept of getting information.

A decision tree can be easily overloaded, generating too many branches, and may exhibit anomalies due to noise or outliers. The overloaded model has very poor performance on unseen data, even though it gives impressive performance on training data [17].

The advantage of a decision tree is the fact that it is possible to find out on which “leaf” (node) of the tree the algorithm made a particular decision. Figure 4.1 shows a simplified decision tree model.

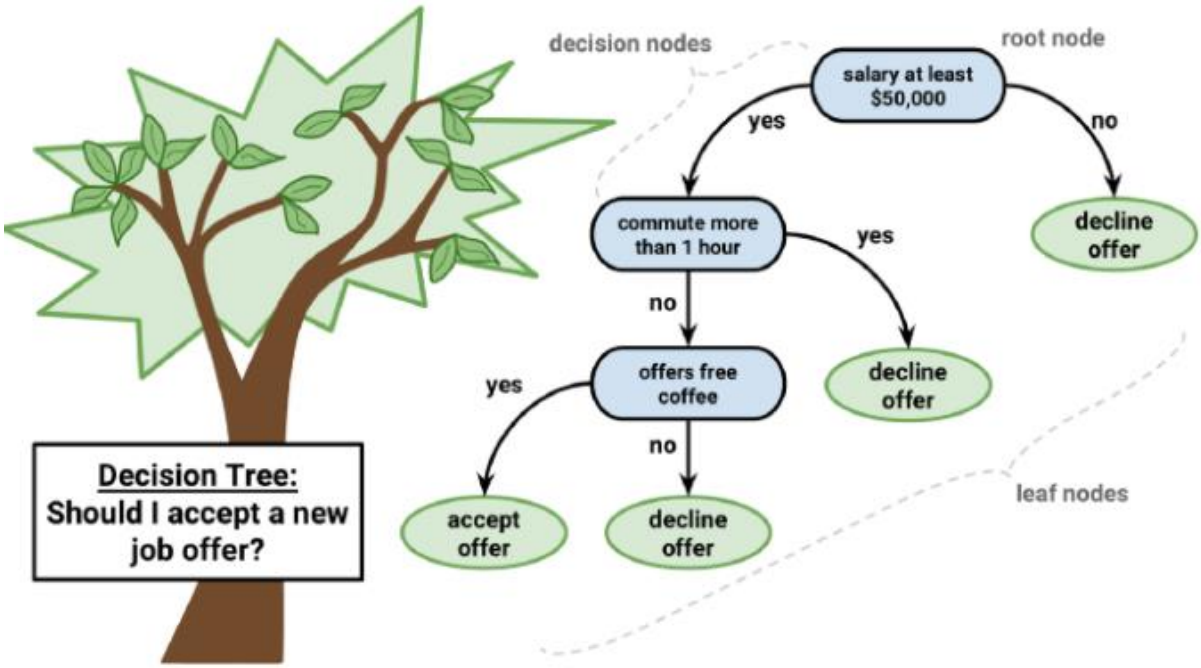

Figure 4.1 – Decision tree

4.3 Naive Bayes

Naive Bayes is a probabilistic classifier inspired by Bayes' formula under the simple assumption that the attributes are conditionally independent [18].

The Bayes formula itself is shown below:

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)},$$

$$P(X) = P(c) \times P(c) \times \cdots \times P(c) \times P(c),$$

where $P(c)$ – prior probability of the hypothesis A;

$P(c \mid x)$ – probability of hypothesis A when event B occurs;

$P(x \mid c)$ – probability of occurrence of event B if hypothesis A is true;

$P(x)$ – full probability.

The naive Bayesian classifier assumes that the presence of features in any class does not lead to the presence of any other feature [18]. An example would be

that a car is a car if it is square, has more than three wheels, and has one driver's seat. These features make an independent contribution to the fact that the item or object is a car and nothing else.

The following table can serve as an example of the work of the Bayesian classifier. Table 2 shows two columns. One of them is called “Weather”, the second “Play”. The bottom line is that in some cases the matches will take place and in others they won't. In some cases, the probability of a match depends directly on the weather, and in some it does not.

Table 2 – Training dataset for the Naive Bayes classifier

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No

In this case, there is one attribute, this is “Weather”, and whether the match will take place or not will be the target variable.

The next step is to convert this dataset into a frequency table. The frequency table is in table 3.

Table 3 – Frequency table

Frequency Table		
Weather	**No**	**Yes**
Overcast	0	2
Sunny	1	2
Rainy	2	1
Total	3	5

The next step is to compile a likelihood table, where the corresponding probabilities can be calculated. This table is displayed in table 4.

Table 4 – Likelihood table

Likelihood Table			
Weather	**No**	**Yes**	
Overcast	0	2	2/8 = 0,25
Sunny	1	2	3/8 = 0,375
Rainy	2	1	3/8 = 0,375
Total	3	5	
	3/8 = 0,375	5/8 = 0,625	

After building the likelihood table, it is possible to apply Bayes' theorem and find the probability that, given the weather, the game will be played or not played.

To simplify, the following problem can be posed in the following way. Will the match be played in rainy weather?

$$P(Rainy \mid Yes) = 1/5 = 0,2,$$

$$P(Rainy) = 3/8 = 0,375,$$

$$P(Yes) = 5/8 = 0,625.$$

The next step is to calculate $P(Yes \mid Rainy)$:

$$P(Yes \mid Rainy) = 0,2 \times 0,625 / 0,375 = 0,333(3).$$

This implies that if it is raining, the likelihood of canceling the match is higher than proceeding with it. Likewise, by considering multiple characteristics, it is possible to predict various categories. This algorithm is commonly utilized for multi-class categorization.

However, this algorithm has a problem. This problem is that if the test set contains a categorical feature that has not previously been found in the training set, then the model will give a zero probability for it [18]. This feature is called “zero frequency”. This problem is overcome by smoothing, such as Laplace smoothing. Another problem may be that predictions are not always sufficiently accurate, and completely separated and independent features, although they occur in practice, are quite rare [18].

5 Regression problem

The regression problem in the supervised learning problem means that for each row of the original data table there is one or another desired output. Approximating this definition to our world, the task of regression can be the prediction of stock prices, or the prediction of demand for a particular car model. In a more formal language, the task of regression is a forecast based on a sample of different objects with different features. Once the algorithm has completed its calculation, it will produce a numerical result that corresponds to the desired output format, such as a car's worth in millions or the interest rate on a mortgage.

The first mention of this term “regression” appeared in the work of Francis Galton, which was called “Regression to the middle in the heredity of growth” [19]. In this work, Galton investigated the dependence of the growth of children on the growth of their parents. The result of this study was that the growth of the deviation of children from the average was 2/3 of the average deviation of the growth of their parents.

One of Galton's most important works is the Galton Machine. This machine is a wooden structure in the upper part of which are iron pins that form an isosceles triangle. Under this triangle is an area with partitions. Iron balls begin to fall on top of the pin triangular structure. And most of the balls, passing through the triangular structure, fall into the tender compartment and form a normal Gaussian distribution [19]. Figure 5.1 schematically shows the Galton machine.

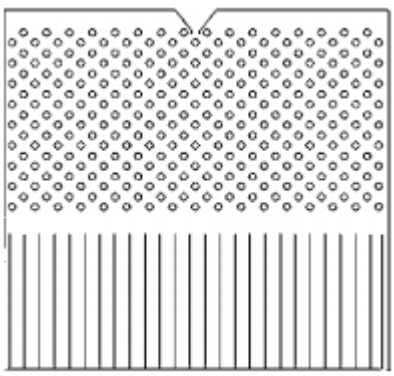

Figure 5.1 – Schematic representation of the Galton machine [19]

Returning to the definition of regression, regression is a modeling technique that utilizes independent predictors to estimate the output target value. It is commonly employed to forecast causal connections among variables [20]. An example of linear regression is shown in Figure 5.2.

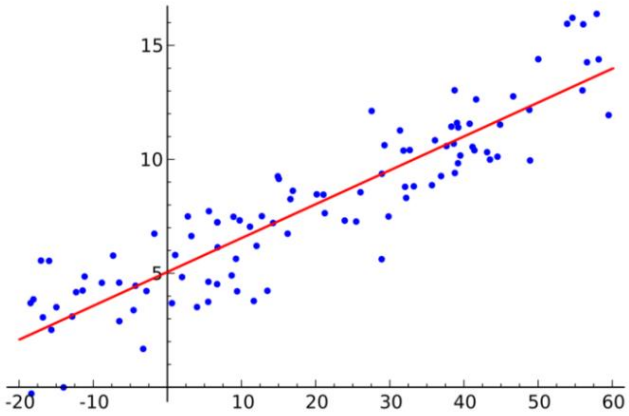

Figure 5.2 – Linear regression

Simple regression is a type of regression analysis in which the number of independent variables is equal to one and there is a linear relationship between the independent x and dependent variables y [18]. The line in figure 5.2 indicates the most appropriate way to model the points on the graph. This red line was modeled with a simple linear equation $y = mx + b$.

The essence of the linear regression method is to find the best values for b and m.

An important part of linear regression is the so-called *«cost function»* [20]. It is also called the cost function. This function helps to calculate the best possible values for a_0 and a_1, which would provide the best fit line for the data points. The cost function is shown below:

$$J = \sum_{i=1}^{n} \left(y_i - (mx_i + b)\right)^2.$$

Since the task is to get the best values for a_0 and a_1 the given problem is transformed into a minimization problem where the error between the predicted values and the actual value is going to me minimized. The root mean square error is a measure of the discrepancy between the predicted and actual values of the

dependent variable. It is calculated by squaring the difference between the predicted and actual values for each data point, summing these squared differences, and dividing by the total number of data points [20]. Now, using this function, it is needed to set the values m and b to the minimum possible values. This is done with the help of "Gradient descent" (from the English gradient descent). Gradient descent is an important concept in linear regression. Its purpose is to update m and b and decrease the value of the cost function.

The idea behind gradient descent is to start fitting from some values m and b and then iteratively change these values to reduce the value of the cost function.

The concept of gradient descent is often compared to a valley flanked by two mountains, where the objective is to reach the valley floor. The surface of the earth serves as a continuous function in this scenario, with the minimum value being the target [20]. The pit can be reached by taking a large number of small steps. These "steps" are often referred to as *learning rate*. If the steps are too small, it will take a long time to reach the bottom, while larger steps can lead to overshooting and missing the target altogether. This slip through the bottom of the valley is shown in Figure 5.3. The number of steps taken in the gradient descent algorithm is determined by the learning rate [20]. This determines how fast the algorithm converges to the minima.

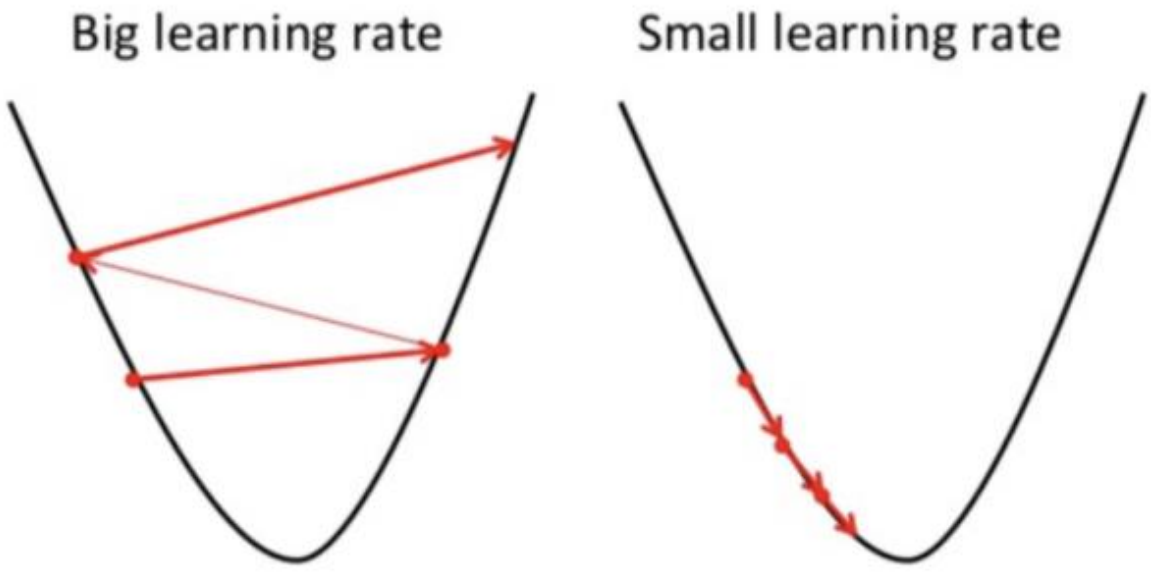

Figure 5.3 – Learning rate for large and small learning rates

Often the cost function may not be uniformly convex, in which case the problem arises that the algorithm will terminate because it realizes that the extremum of the function has been found. But it is possible that a local extremum of the function has been found. Figure 5.4 shows two functions, one of them has exactly one extremum (left), while the right one has two extrema. The first of them is local, and is located on the left, but to the right is the global extremum, the desired one.

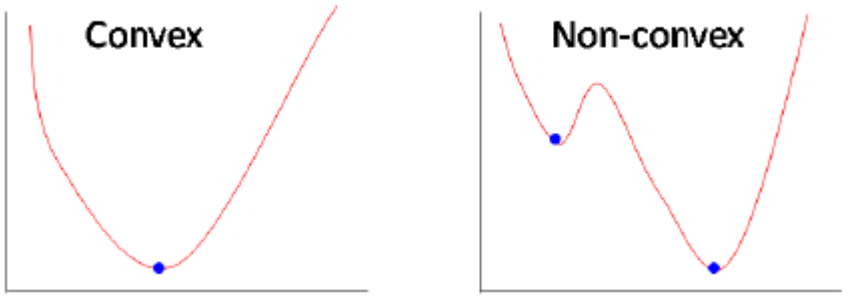

Figure 5.4 – Global and local extremes

The calculation is as follows. In the cost function, there are two parameters that can be controlled, this is the weight m and bias b. Partial derivatives are utilized to consider the influence of each of them on the final prediction [20]. To find partial derivatives, the so-called chain rule is used, or, as they are called, the nesting rules for derivatives, since indeed: $\big(y-(mx+b)\big)^2$ these are two nested functions. First: $\big(y-(mx+b)\big)$, and second: x^2 [21].

Returning to the cost function, its structure can be observed as follows:

$$f(m,b)=\sum_{i=1}^{n}\big(y_i-(mx_i+b)\big)^2.$$

The following gives:

$$\big(y_i-(mx_i+b)\big)^2 \;=\; A(B(m,b).$$

After that:

$$B(m,b)=y_i-(mx_i+b)=y_i-mx_i-b,$$

$$\frac{dx}{dm}=B'(m)=0-x_i-0=-x_i,$$

$$\frac{dx}{db} = B'(b) = 0 - 0 - 1 = -1.$$

Then the derivative chain rule is used, which says:

$$\frac{df}{dm} = \frac{df}{dx}\frac{dx}{dm},$$
$$\frac{df}{db} = \frac{df}{dx}\frac{dx}{db}.$$

And it gives the following:

$$\frac{df}{dm} = A'\big(B(m,f)\big)B'(m) = 2\big(y_i - (mx_i + b)\big) \times -x_i,$$
$$\frac{df}{db} = A'\big(B(m,f)\big)B'(b) = 2\big(y_i - (mx_i + b)\big) \times -1.$$

After that, the gradient of the cost function is calculated:

$$f'(m,b) = \left[\frac{df}{dm}\ \frac{df}{db}\right] =$$
$$= \left[\sum -x_i \times 2\big(y_i - (mx_i + b)\big) \sum -1 \times 2\big(y_i - (mx_i + b)\big)\right]$$
$$= \left[\sum -2x_i\big(y_i - (mx_i + b)\big) \sum -2\big(y_i - (mx_i + b)\big)\right].$$

6 Data preparation

Data is incredibly important, especially when it comes to neural networks and training various machine learning models. Some researchers are even sure that the stage of data preparation is one of the most difficult in building a neural network model.

Working with data before sending it to the model is necessary because many data, when they are received by the user when downloading or parsing, can often contain different data, statistical outliers, missing values, this can certainly affect the result of the study. In order to obtain accurate results from the algorithm, it is necessary to pre-process the data prior to input, as failure to do so may result in unreliable outcomes.

6.1 Receiving and processing data

Intel stocks were chosen as the main training data for this study. This choice was determined purely by the desire of the researcher, and any other company can be used as a share.

Various sources can be used to obtain data, such as websites of various exchanges, open sources for data aggregation such as Kaggle [5]. But for this work, the library from Yahoo Finance in Python and the Pandas library [22] were chosen. It is also worth noting that this study is planned to be completely performed in Python in the Jupyter Lab environment. The code for getting stock prices is shown in Figure 6.1.

```
[ ]  intel_ticker = "INTC"
     start_time = dt.datetime(2017, 3, 1)
     end_time = dt.datetime(2023, 3, 1)
     #getting Intel stocks data
     intel_data = yf.download(tickers=[intel_ticker], start=start_time, end=end_time)
     #getting rid of unused columns
     intel_data.drop(["Adj Close", "Volume"], inplace=True, axis=1)
```

Figure 6.1 – Downloading the Intel stock prices

It is worth considering that it is necessary to conduct training on as much information as possible, so it is necessary to add a few more representative

columns [21]. It is logical to assume that Intel stock prices depend not only on the behavior of the founders or shareholders of the company, but also on many other factors. Since it is almost impossible to collect information about all economic events that can affect the stock price, it is worth estimating which companies Intel can correlate with. Intel, as known, produces processors, and processors are used by many other computer manufacturing companies. Therefore, to supplement the data sample, it was decided to take the stock price of HP, which manufactures computers. The code for obtaining HP data is shown in Figure 6.2.

```
[ ]  hp_ticker = "HPQ"


     hp_data = yf.download(tickers=[hp_ticker], start=start_time, end=end_time)
     hp_data.drop(["Adj Close", "Volume"], inplace=True, axis=1)
```

Figure 6.2 – Downloading of HP's stock prices

Next, the sample tables should be combined by date, however, in order to maintain consistency in the column names, the columns must be renamed [23]. In particular, for Intel, the columns: Open, Close, High, Low should be renamed to OpenIntel, Close Intel, High Intel, Low Intel. The code for renaming columns is shown in Figure 6.3.

```
[ ]  intel_data.rename(columns={"Open": "OpenIntel", "Close": "CloseIntel", "High": "HighIntel", "Low": "LowIntel"},
                       inplace=True)


     hp_data.rename(columns={"Open": "OpenHPQ", "Close": "CloseHPQ", "High": "HighHPQ", "Low": "LowHPQ"},
                    inplace=True)
```

Figure 6.3 – Renaming columns for Intel and HP

After that, two tables can be joined into one. Next, it is needed to fill in the empty cells. In the pandas library, this is done using the fillna method, after which, for convenience, the index was reset. The code is shown in Figure 6.4.

```
intel_data['Date'] = intel_data.index
hp_data['Date'] = hp_data.index

intel_and_hp = intel_data.set_index('Date').join(hp_data.set_index('Date'))
intel_and_hp.fillna(method='ffill', inplace=True)
intel_and_hp.reset_index(inplace=True)
```

Figure 6.4 – Joining tables

The price chart based on the closing price is shown in Figure 6.5

Figure 6.5 – Intel and HP price chart

Of course, the frame should be significantly increased [23]. Adding columns will increase the number of variables by which the algorithm will be trained, which means that there will be a clear increase in the accuracy of the algorithm. Brokerage firms use various date splits (day, month, year) in order to increase the clarity of charts and allow clients and brokers themselves to use tools and metrics that depend on the splits [21]. Therefore, in this study, partitioning in order to increase the size of the data sample was also added. In addition, it would be interesting to see the dependence of the Intel stock price depending on the day of the week or over the past 9 days. 9 days were chosen because different metrics based on a moving average often use a 9-day split for calculations. The result of manipulating the data table is shown in Figure 6.6.

```
[ ]  intel_and_hp["year"] = intel_and_hp["Date"].dt.year
     intel_and_hp["month"] = intel_and_hp["Date"].dt.month
     intel_and_hp["weekday"] = intel_and_hp["Date"].dt.weekday

     rolling_days = 9

     for day in range(rolling_days):
         intel_and_hp[f"Intel Lag {day + 1}"] = intel_and_hp["CloseIntel"].shift(day + 1)
         intel_and_hp[f"HPQ Lag {day + 1}"] = intel_and_hp["CloseHPQ"].shift(day + 1)

     intel_and_hp["Intel Week"] = intel_and_hp["CloseIntel"].shift(1)
     intel_and_hp["Intel Week"].rolling(window=rolling_days).median()

     intel_and_hp["HPQ Week"] = intel_and_hp["CloseHPQ"].shift(1)
     intel_and_hp["HPQ Week"].rolling(window=rolling_days).median()
```

Figure 6.6 – Data table expansion due to inside tabular data

After that, the data table is almost ready for training. Next, it needs to be divided into training and test samples. This can be done in different proportions. The “openIsGreater” column has been added, which is designed specifically for the classifier. This parameter takes the value 1 or 0, depending on whether the closing price of the stock is greater than the opening price or not.

These transformations can be seen in Figure 6.7.

```
[ ]  df_for_ML = pd.get_dummies(intel_and_hp, columns=["year", "month", "weekday"])
     df_for_ML.drop(["Date"], axis=1, inplace=True)
     df_for_ML.dropna(inplace=True)
```

```
[ ]  df_final = df_for_ML[df_for_ML["CloseIntel"].notna() & df_for_ML["CloseHPQ"].notna()]
     df_final["openIsGreater"] = (df_final["CloseIntel"] < df_final["OpenIntel"]).astype(int)
```

Figure 6.7 – Data transformation for separation into test and training samples

After that, using the train_test_split() method from the Python scikit–learn library, the data was split into training and test samples. The target variable for the regressor will be “CloseIntel” (the closing price of Intel shares), and for the classifier “openIsGreater” (whether the closing price is greater than the opening price or not). The breakdown into test and training samples is shown in Figure 6.8.

```
[ ]  x = df_final.drop("CloseIntel", axis=1)
     y = df_final["CloseIntel"]
     x_reg_train, x_reg_test, y_reg_train, y_reg_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

```
[ ]  x = df_final.drop("openIsGreater", axis=1)
     y = df_final["openIsGreater"]
     x_class_train, x_class_test, y_class_train, y_class_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

Figure 6.8 – Splitting into test and training samples

The last thing is to prepare data for predictions. For this purpose, the prices of the 60 days preceding each day were chosen as features. The code for preparing data for the predictions is presented in the Appendix A.

7 Training models using the scikit–learn and Keras libraries

As mentioned earlier, the Python programming language was used for the study, as it offers a wide selection of libraries for working with data, as well as for machine learning. This particular library was used for this study, since it simplifies the processes of building architectures for machine learning as much as possible.

7.1 Regression prediction models for Intel stock prices

The KRR (Kernel Ridge Regression) model was chosen for the experiment. The peculiarity of this model is that before launching it, it is possible to select the kernel function. On the library's official website, this choice is called the “kernel trick” [24]. This allows to experimentally select a core that shows the best quality metric.

The form of the KRR model is identical to the regression of the support vector (SVR) [25]. However, different loss functions are used: KRR uses squared error loss, while support vector regression uses insensitive epsilon losses. Unlike SVR, the installation of the KRR model can be performed in a closed form and is usually faster for average datasets [25]. Initially, the KRR model was trained by using the first dataset, which was supplemented by various additional columns. The code is presented in Figure 7.1.

```
[ ]  kernels = ['laplacian', 'linear', 'polynomial', 'sigmoid', 'rbf']
     mse_dict_KR = {}
     mae_dict_KR = {}
     for kernel in kernels:
         model_KR = KernelRidge(kernel=kernel)
         model_KR.fit(x_reg_train, y_reg_train)
         pred_KR = model_KR.predict(x_reg_test)
         mae_dict_KR[kernel] = mean_absolute_error(y_reg_test, pred_KR)
         mse_dict_KR[kernel] = mean_squared_error(y_reg_test, pred_KR)

     print(f"Best average deviation (mse): "
           f"{min(mse_dict_KR, key=lambda unit: mse_dict_KR[unit])}:"
           f"{mse_dict_KR[min(mse_dict_KR, key=lambda unit: mse_dict_KR[unit])]} ")

     print(f"Best average deviation (mae): "
           f"{min(mae_dict_KR, key=lambda unit: mae_dict_KR[unit])}:"
           f"{mae_dict_KR[min(mae_dict_KR, key=lambda unit: mae_dict_KR[unit])]} ")
```

Figure 7.1 – Identifying the best kernel parameter

Experimentally, it was found that the best kernel parameter is linear, its average deviation was 0,21.

The following study was conducted using the Keras library. Several models were built for the experiment:

without hidden layers;

1 hidden layer;

4 hidden layers.

It is also worth saying that during the study, only the number of output layers changed. All models were trained on the same supplemented dataset, where the training sample was 70%, and the test sample was 30% For the regressor without hidden layers and for subsequent regression models, the “Adam” optimizer was used, and loss was calculated as “Mean Absolute Error”, the “accuracy” metric was set, and “Relu” was used as the activation function. The number of training epochs is 200.

The results of the study for the regressors are recorded in Table 5. KRR represents the result of the regressor's work from the scikit–learn library, the rest of the results were obtained using the Keras library. The code for a regressor with a different number of hidden layers is presented in figures 7.2, 7.3 and 7.4.

Table 5 – Regressors` results on the first dataset

	KRR	Without hidden layers	One hidden layer	Four hidden layers
Mean Absolute Error	0,21	1,17	0,63	0,61

As can be seen, KRR built using the scikit–learn library has a smaller error compared to other models built using Keras.

```
[ ]  cpt_filename = "0layers_best.hdf5"
     cpt_path = str(cpt_filename)
     checkpoint = tf.keras.callbacks.ModelCheckpoint(cpt_path,
                                                     monitor='loss',
                                                     verbose=1,
                                                     save_best_only=True,
                                                     mode='min')
     model = tf.keras.Sequential()
     model.add(tf.keras.layers.Dense(1, input_shape=(input_dim,)))
     model.compile(optimizer="adam", loss="mean_absolute_error", metrics=["accuracy"])
     model.fit(x=x_reg_train, y=y_reg_train, validation_data=(x_reg_train, y_reg_train), epochs=200, verbose=1,
               callbacks=[checkpoint])

     predicted = model.predict(x_reg_test)
```

Figure 7.2 – Constructing model with no hidden layers

```
[ ]  cpt_filename = "1layer_best.hdf5"
     cpt_path = str(cpt_filename)
     checkpoint = tf.keras.callbacks.ModelCheckpoint(cpt_path,
                                                     monitor='loss',
                                                     verbose=1,
                                                     save_best_only=True,
                                                     mode='min')
     model = tf.keras.Sequential()
     model.add(tf.keras.layers.Dense(512, input_shape=(input_dim,), activation='relu'))
     model.add(tf.keras.layers.Dense(1))
     model.compile(optimizer="adam", loss="mean_absolute_error")
     model.fit(x=x_reg_train, y=y_reg_train, validation_data=(x_reg_train, y_reg_train), epochs=200, verbose=1,
           callbacks=[checkpoint])

     predicted = model.predict(x_reg_test)
```

Figure 7.3 – Constructing model with one hidden layer

```
[ ]  cpt_filename = "4layers_best.hdf5"
     cpt_path = str(cpt_filename)
     checkpoint = tf.keras.callbacks.ModelCheckpoint(cpt_path,
                                                     monitor='loss',
                                                     verbose=1,
                                                     save_best_only=True,
                                                     mode='min')
     model = tf.keras.Sequential()
     model.add(tf.keras.layers.Dense(256, input_shape=(input_dim,), activation='relu'))
     model.add(tf.keras.layers.Dense(512, activation='relu'))
     model.add(tf.keras.layers.Dense(1024, activation='relu'))
     model.add(tf.keras.layers.Dense(2048, activation='relu'))
     optimizer = tf.keras.optimizers.Adam(learning_rate=0.00001)
     model.add(tf.keras.layers.Dense(1))
     model.compile(optimizer=optimizer, loss="mean_absolute_error")
     model.fit(x=x_reg_train, y=y_reg_train, epochs=200, verbose=1, callbacks=[checkpoint])

     predicted = model.predict(x_reg_test)
```

Figure 7.4 – Constructing model with four hidden layers

The next step is to test the same models using the second dataset, which was constructed by taking 60 preceding days` prices. For this study, sequential models built by using Keras with “Mean Squared Error” serving as the loss function.

Regressors` perfomance comparison is presented in table 6.

Table 6 – Regressors` results on the second dataset

	KRR	Without hidden layers	One hidden layer	Four hidden layers
Mean Squared Error	0,92	2,03	3,19	1,07

Again, KRR model built using the scikit–learn library has smaller error compared to other models built using Keras.

The graphs of predictions of each model are presented in figures 7.5-7.8.

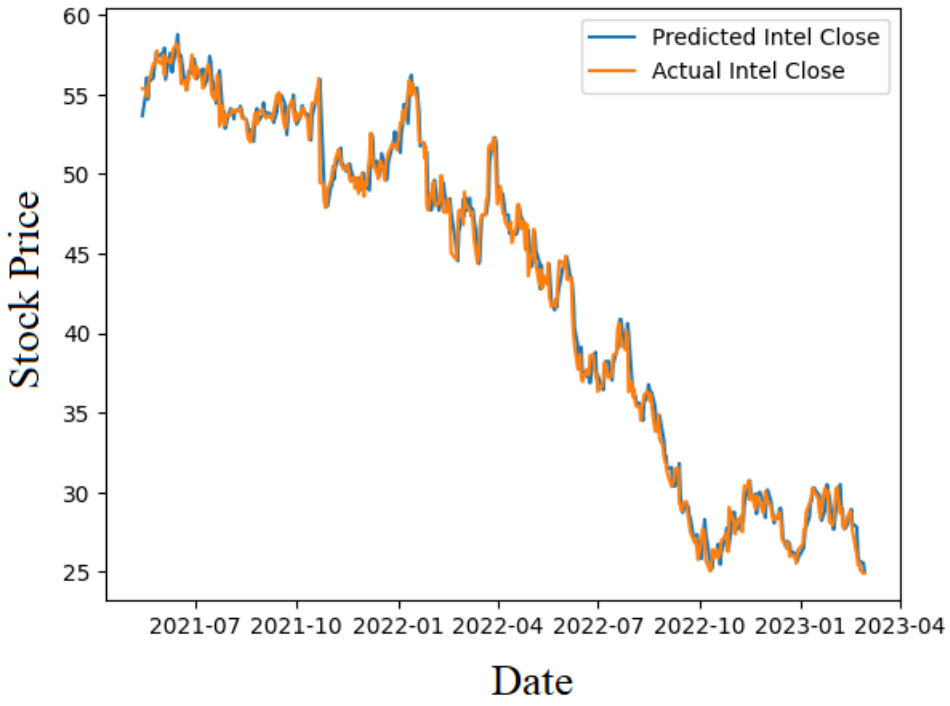

Figure 7.5 – Prediction of KRR model with linear kernel

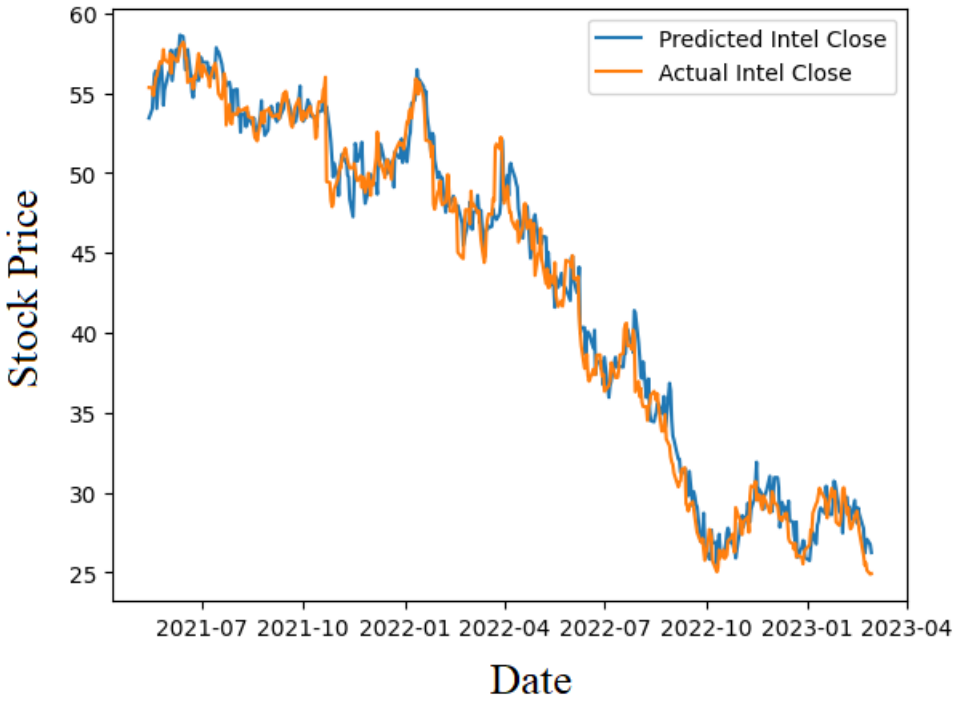

Figure 7.6 – Prediction of sequential model with no hidden layers

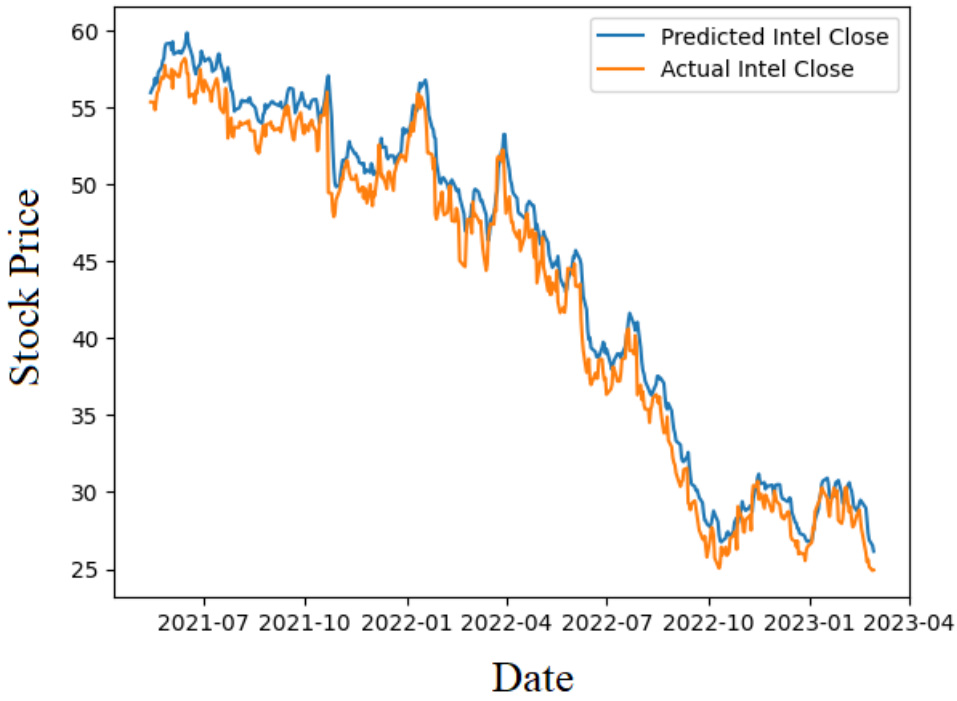

Figure 7.7 – Prediction of sequential model with one hidden layer

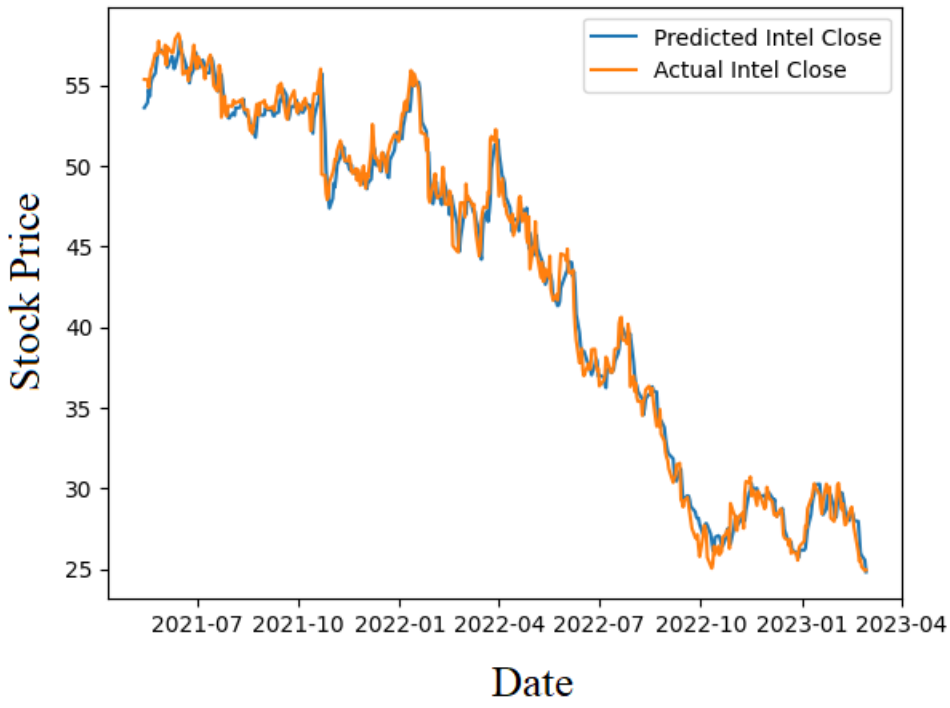

Figure 7.8 – Prediction of sequential model with four hidden layers

7.2 Classification prediction model for Intel stock price

In addition to regression algorithms, it is also worth considering classification algorithms.

The scikit-learn library's standard tools were used to implement classification algorithms, including decision trees, nearest neighbor algorithm, and naive Bayesian algorithm, for predicting Intel stock prices. A dataset was prepared specifically for this task, as described above. Unlike the regressor, binary labels were added to the classifier's data preparation, with 1 added if the closing price is greater than the opening price and 0 added otherwise.

The first trained model was the Bayesian classifier, or as it is also called the naive Bayesian classifier. This algorithm is quite simple and can be implemented in a few lines using the scikit–learn library [26].

The code for a Naive Bayes Classifier is presented in figure 7.9.

```
[ ]  naive_bayes = GaussianNB()
     model = naive_bayes.fit(x_class_train, y_class_train)
     accuracy = naive_bayes.score(x_class_test, y_class_test)
     accuracy
```

Figure 7.9 – Constructing Naive Bayes Classifier

The next algorithm used was Decision Tree. The data used to train the model remained the same as in the previous dataset.

Additionally, different parameter values were used for the study, including criterion and splitter. The first parameter determines the stopping criterion for the algorithm, while the second parameter determines the function and threshold used to split the data at each node [26].

The code is presented in figure 7.10.

```
[ ]  criteria = ["gini", "entropy"]
     splitters = ["best", "random"]

     for cri in criteria:
         for spl in splitters:
             decTreeCla = DecisionTreeClassifier(criterion=cri, splitter=spl)
             model = decTreeCla.fit(x_class_train, y_class_train)
             acc = decTreeCla.score(x_class_test, y_class_test)
```

Figure 7.10 – Constructing Decision Tree Classifiers with different parameters

The next algorithm to be implemented using the scikit-learn library is the k-nearest neighbor algorithm (kNN). For the study, various numbers of nearest neighbors were set and passed to kNN, which is responsible for searching for that number of nearest neighbors for each sample object [27].

The code is presented in figure 7.11.

```
[ ]  accuracy_list = []

     for neighbors_num in range(1, 200):
         neig = KNeighborsClassifier(n_neighbors=neighbors_num)
         model = neig.fit(x_class_train, y_class_train)
         acc = neig.score(x_class_test, y_class_test)
         accuracy_list.append(acc)
```

Figure 7.11 – Constructing KNN Classifier with different number of neighbors

The graph with the accuracy value depending on the number of clusters is shown in Figure 7.12

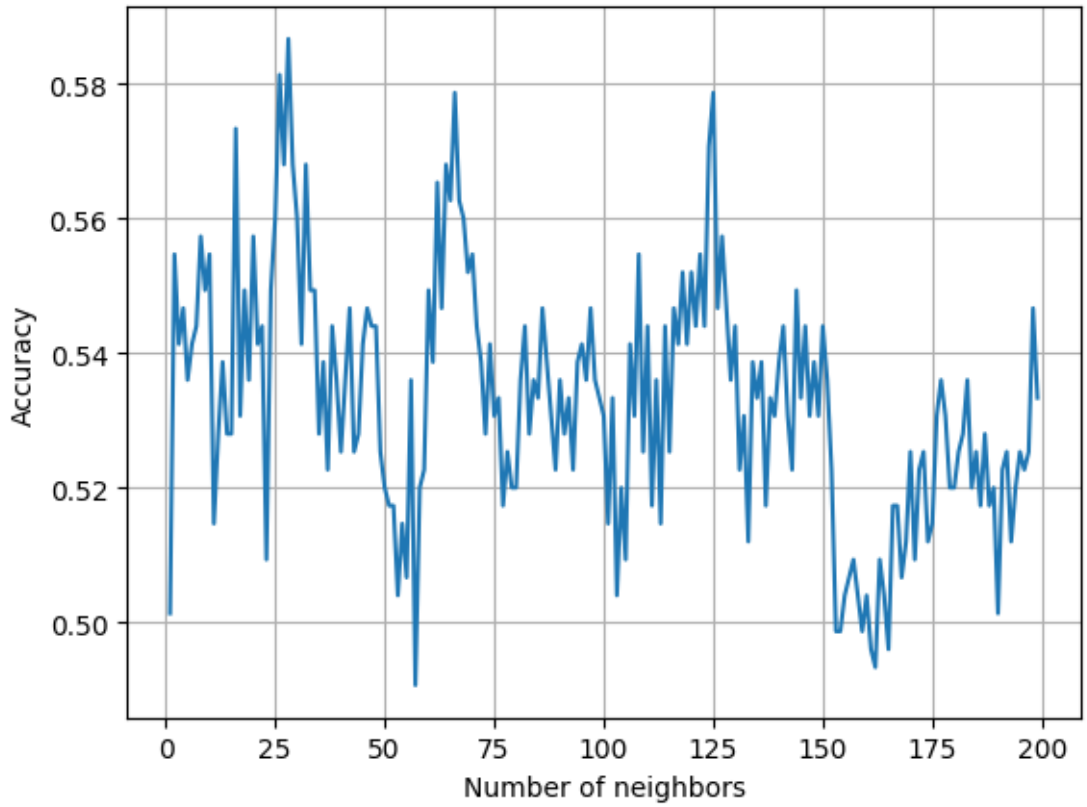

Figure 7.12 – Dependency between accuracy and number of neighbors

The scoring results of each model is presented in Table 7.

Table 7 – Accuracy of various models from the scikit–learn library

GaussianNB	kNN	Decision Tree			
		GINI/Best	GINI/Random	Entropy/Best	Entropy/Random
0,495	0,586	0,645	0,493	0,704	0,576

7.3 LSTM network

One of the most popular algorithms for predicting stock prices is the LSTM algorithm. This abbreviation stands for Long Short–Term Memory (from the English long-term short-term memory) [12].

By definition, long–term short-term memory is a type of recurrent neural network architecture proposed in 1997 by Sepp Hochreiter and Jurgen Schmidhuber [13]. It belongs to the class of so-called universal neural networks. It is well suited

for the analysis of time series, respectively, it is well suited for the analysis of stocks on stock exchanges [12].

From the architectural features, it is possible to distinguish such important details as the ability of the network to remember values, both for relatively small and for fairly wide periods of time. This possibility is provided by the fact that the LSTM model does not use the activation function inside its recurrent components [13].

LSTM networks can be represented in the form of a chain of repeating ordinary neural networks, but repeating modules have a much more complex structure [13]. For example, instead of a single-layer neural network, they have a four-layer one organized in a specific way [12].

However, in order to be able to use LSTM, it is necessary to prepare the data correctly, because only three-dimensional data can be placed in the LSTM network [12]. The code for preparing data for the LSTM algorithm included min-max scaling to scale features values to the range from 0 to 1. It is also worth saying that the same task was solved here as in previous networks. It was necessary to predict the closing price for Intel shares.

The code for the LSTM algorithm is presented in Appendix B. There are two versions available - one with thinning and one without. Thinning is used to prevent overtraining of the neural network by reducing or preventing complex coadaptations between individual neurons during training. If the thinning function is not used, there is a high probability that the network will overtrain and show high accuracy results on the training data, but perform poorly on the test data. This is because the neural network simply memorizes the values from the training data rather than actually learning how to properly train. The Dropout value ranges from 0 to 1 and indicates what percentage of the training data will be disregarded during training.

The model consisted of five layers, each with an input space dimension of 60. Following the fifth layer, a convolution output layer was included. During network training, the “Adam” optimizer was utilized, and the loss function was “Mean Squared Error”. The model underwent 200 training epochs.

The LSTM model trained without thinning showed disappointing prediction results at 2.73 for Mean Squared Error metric.

Figure 7.13 shows a graph of the predicted price (indicated in blue) and a real chart of the closing price (indicated in orange).

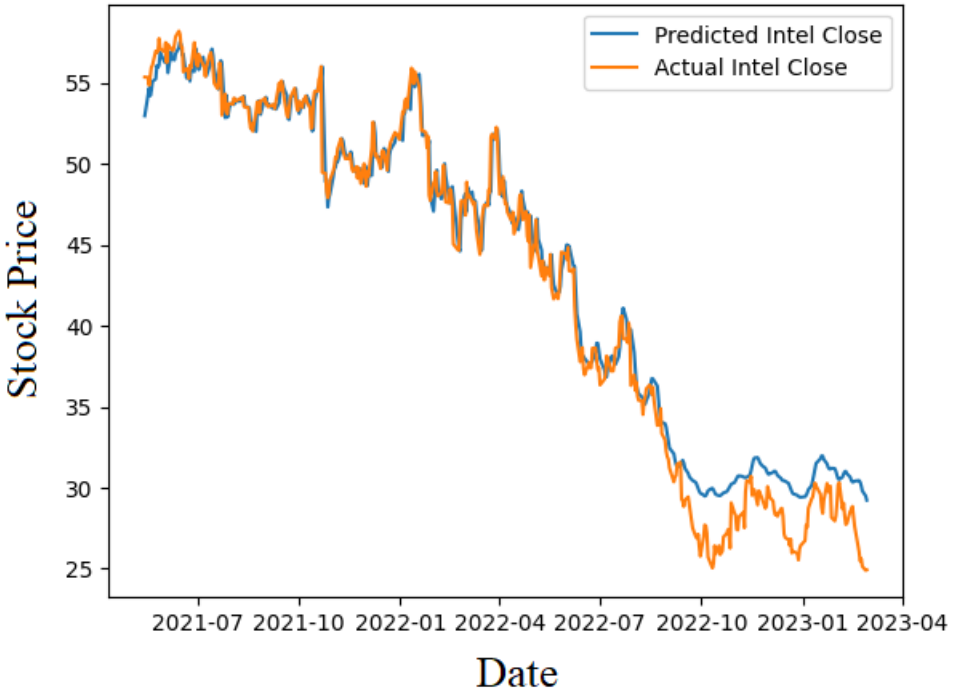

Figure 7.13 – Prediction of the LSTM model without thinning

Further, thinning was activated without altering the network architecture. The LSTM model trained with thinning produced significantly improved prediction outcomes, with a Mean Squared Error metric of 1.01.

Figure 7.14 shows a graph with predicted prices and actual closing prices for Intel shares.

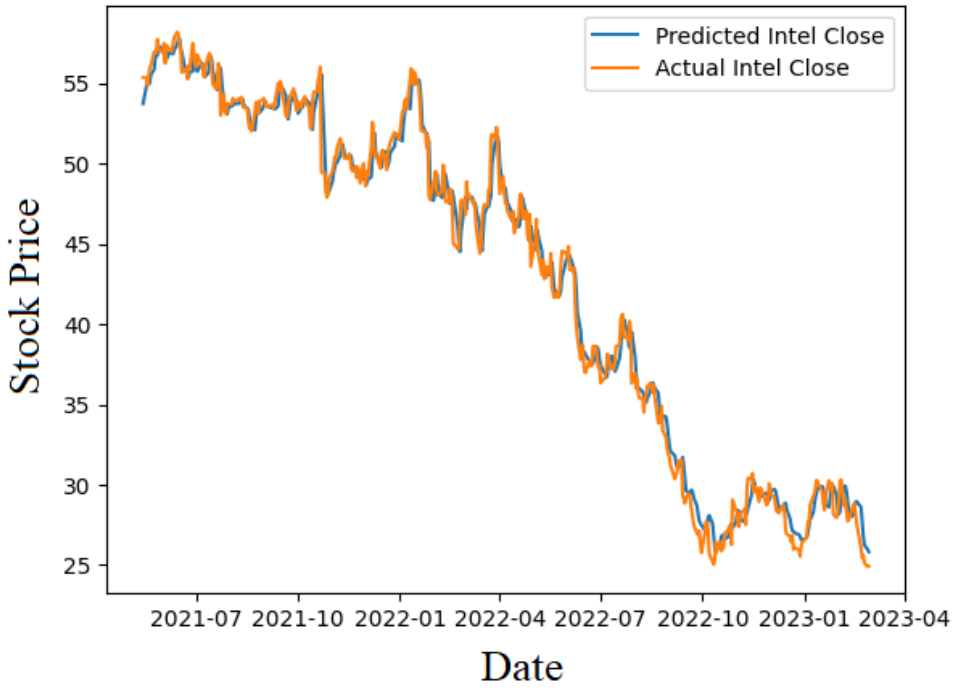

Figure 7.14 – Prediction of the LSTM model with thinning

The graphs in Figures 7.13 and 7.14 are visually different, especially at their ends, from which it can be conclude that thinning prevented our model from degradation and overtraining.

CONCLUSION

The paper showcased the utilization of technical analysis techniques in professional stock trading and presented multiple machine learning algorithms. These included KRR regression algorithms implemented with the scikit-learn library, a Keras library-based regressor with different hidden layer variations, a naive Bayesian classifier, a kNN classifier, and a decision tree algorithm. The implementation of these algorithms was carried out with the scikit-learn library. Lastly, an LSTM model was implemented using the Keras library.

KRR method showed the smallest error result compared to all sequential models built by Keras and amounted to 0.21 of Mean Absolute Error metric and 0.92 of Mean Squared Error metric for estimation supplemented dataset and prediction dataset respectively. As for classifiers, the decision tree algorithm performed the best with an accuracy of 0.70, which is not ideal but still 1.42 times better than the Bayesian classifier algorithm and 1.20 times better than the kNN algorithm.

The LSTM model created using the Keras library and Tensorflow yielded Mean Squared Error metrics of 1.01 and 2.73 for the prediction dataset trained with and without thinning, respectively.

Based on the findings of the study, it can be inferred that traditional technical analysis methods are not less and might be even more effective than contemporary machine learning. In some cases, machine learning approaches might be a redundant overhead.

REFERENCES

1 Cyberbug 2077: How CDPR failed astronomically at Cyberpunk launch but is set for massive success / dailysabah.com. – 2023. – URL: https://www.dailysabah.com/life/cyberbug-2077-how-cdpr-failed-astronomically-at-cyberpunk-launch-but-is-set-for-massive-success/news.

2 CDR Stock CD PROJECT / investing.com. – 2023. – URL: https://investing.com/equities/cdproject.

3 Murphy, J. Technical Analysis of the Financial Markets / J. Murphy. – Penguin Publishing Group, 1999. – 576 p.

4 Petrusheva, N. Comparative analysis between the fundamental and technical analysis of stocks / N. Petrusheva, I. Jordanski. // Journal of Process Management. New Technologies. – 2016. – Vol. 4(2). – P. 26-31.

5 Kaggle: an online community of data scientists and machine learning practitioners / kaggle.com . – 2023. – URL: https://www.kaggle.com/

6 Sharp, W. Investments / W. Sharp, G. Alexander, J. Bailey. – Prentice Hall International, Inc, 1995. – 1028 p.

7 Tesla Stock Price (TSLA) / investing.com. – 2023. – URL: https://www.investing.com/equities/tesla-motors.

8 Investopedia: world's leading source of financial content / investopedia.com . – 2023. – URL: https://www.investopedia.com/terms/v/volatility.asp

9 Wilder, J. New Concepts in Technical Trading Systems / J. Wilder. – Trend Research, 1978. – 141 p.

10 Bulkowski, T. Encyclopedia of Chart Patterns / T. Bulkowski. – Wiley, 2021. – 1312 p.

11 Dynamic remodeling of dendritic arbors in GABAergic interneurons of adult visual cortex / W.C.A. Lee // PLoS Biol. – 2005. – Vol. 4(2). – e29 p.

12 Pawar, K. Stock market price prediction using LSTM RNN / K. Pawar, R. Jalem, V. Tiwari // In Emerging Trends in Expert Applications and Security. – 2019. – P. 493-503.

13 Neural networks in financial trading / G. Sermpinis // Annals of Operations Research. – 2019. – P. 293-308.

14 Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities / G. Cheng // IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. – 2020 – Vol. 13. – P. 3735-3756.

15 Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis / M. Nabipour // IEEE Access. – 2020. – Vol. 8. – P. 150199-150212.

16 Kohli, S. Sales Prediction Using Linear and KNN Regression / S. Kohli, G.T. Godwin, S. Urolagin // Advances in Machine Learning and Computational Intelligence. – 2021. – P.321-329.

17 Decision trees and random forests / R. Casarin // The Essentials of Machine Learning in Finance and Accounting. – 2021. – P. 7-36.

18 Setiani, I. Prediction of Banking Stock Prices Using Naive Bayes Method / I. Setiani, M. N. Tentua, S. Oyama // Journal of Physics: Conference Series. – 2021. – Vol. 1830(3). – P. 012059.

19 Kozlov, V.V. Galton board / V.V. Kozlov, M.Y. Mitrofanova // ArXiv. – 2005. – URL: https://arxiv.org/pdf/nlin/0503024.pdf.

20 Emioma, C.C. Stock price prediction using machine learning on least-squares linear regression basis // C.C. Emioma, S.O. Edeki // Journal of Physics: Conference Series. – 2021. – Vol. 1734(1). – P. 012058.

21 Кудрявцев, Л.Д. Курс Математического Анализа [Текст]: учеб. пособие / Л.Д. Кудрявцев. – М.: Дрофа, 2003. – 703 с.

22 Pandas: fast, powerful, flexible and easy to use open source data analysis and manipulation tool / PyData.org. – 2023. – URL: https://pandas.pydata.org/.

23 Zhu, Y. Stock price prediction using the RNN model / Y. Zhu // Journal of Physics: Conference Series. – 2020. – Vol. 1650(3). – P. 032103.

24 SkLearn / scikit-learn.org. – 2023. – URL: https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html.

25 Burnaev, E. Conformalized kernel ridge regression / E. Burnaev, I. Nazarov // 15th IEEE international conference on machine learning and applications (ICMLA). – 2016. – P. 45-52.

26 Yang, K.C. Applying K-Means Technique and Decision Tree Analysis to Predict Taiwan ETF Performance / K.C. Yang, W.P. Chao // IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). – 2020. – P. 635-639.

27 Machine Learning Methods for Stock Market Analysis / S. Kalyoncu // In 3rd International Conference on Data Science and Applications. – 2020. – 5 p.

APPENDIX A

```
rows_number, _ = df_final.shape
train_portion_ratio = rows_number // 10 * 7
test_portion_ration = rows_number - train_portion_ratio

training_set = df_final.iloc[:train_portion_ratio, 3:4].values
test_set = df_final.iloc[test_portion_ration:, 3:4].values

xtrain = []
ytrain = []

previous_days = 60

for i in range(previous_days, train_portion_ratio):
    xtrain.append(training_set_scaled[i - previous_days:i, 0])
    ytrain.append(training_set_scaled[i, 0])
xtrain, ytrain = np.array(xtrain), np.array(ytrain)

xtrain = np.reshape(xtrain, (xtrain.shape[0], xtrain.shape[1], 1))

dataset_train = df_final.iloc[:train_portion_ratio, 3:4]
dataset_test = df_final.iloc[train_portion_ratio:, 3:4]
dataset_total = pd.concat((dataset_train, dataset_test), axis=0)

inputs = dataset_total[len(dataset_total) - len(dataset_test) - previous_days
:].values
inputs = inputs.reshape(-1, 1)

xtest = []

for i in range(previous_days, len(inputs)):
  xtest.append(inputs[i - previous_days:i, 0])

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

APPENDIX B

```
cpt_filename = "best.hdf5"
cpt_path = cpt_filename
checkpoint = tf.keras.callbacks.ModelCheckpoint(cpt_path,
                                    monitor="loss", verbose=1,
                                    save_best_only=True, mode='min')

regression= Sequential()
regression.add(LSTM(units=60,return_sequences=True,kernel_initializer='glo
rot_uniform',input_shape=(xtrain.shape[1], 1)))
regression.add(Dropout(0.2))

regression.add(LSTM(units=60,kernel_initializer='glorot_uniform',return_se
quences=True))
regression.add(Dropout(0.2))

regression.add(LSTM(units=60,kernel_initializer='glorot_uniform',return_se
quences=True))
regression.add(Dropout(0.2))


regression.add(LSTM(units=60,kernel_initializer='glorot_uniform', return_s
equences=True))
regression.add(Dropout(0.3))

regression.add(LSTM(units=60,kernel_initializer='glorot_uniform'))
regression.add(Dropout(0.3))

regression.add(Dense(units=1))

regression.compile(optimizer='adam',loss='mean_squared_error')

regression.fit(xtrain,ytrain,batch_size=30,epochs=200, callbacks=[checkpoi
nt])
```