Real-Time Stock Prediction Using Hadoop, PySpark, and Kafka

Matt Gregorat
Florida Polytechnic University
Lakeland, USA
mgregorat0995@floridapoly.edu

Prateek Mukherjee
Florida Polytechnic University
Lakeland, USA
pmukherjee9928@floridapoly.edu

Alexander Burkhart
Florida Polytechnic University
Lakeland, USA
aburkhart6899@floridapoly.edu

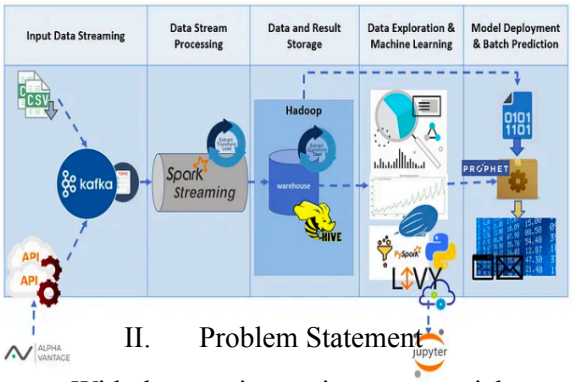
Abstract - We aim to highlight the significance of Developing a Scalable Solution for Forecasting Stock Prices Using Hadoop, Kafka, and PySpark. Using ticker data, we created an Extract-Transform-Load (ETL) - pipeline to transfer data in real time and conduct technical analysis to identify trends.
Keywords – Bullish, Bearish, MA, RSI, Bollinger Bands, shorting

I. Introduction

In the landscape of financial markets, accurately forecasting stock prices accurately has profound implications upon investors, analysts, and businesses. This project involves developing a scalable solution for processing and analyzing real-time stock data, leveraging the combined power of Hadoop, Kafka, and PySpark. Our focus will be on showcasing the solution's capabilities on Nvidia and Tesla. The solution can be deployed and utilized on any stock with available ticker data.

We will emulate the role of a data consulting firm that specializes in stock market forecasts as our main product. In order to do this, we will utilize Hadoop, a distributed computing framework, which will handle the vast amounts of historical data that will be handled for comprehensive analysis. Using Hadoop, we will lay the foundation for our technical analysis, enabling us to find trends and patterns within the data.

In addition to Hadoop, we will take advantage of Kafka for real-time data streaming, allowing for our market data to be updated continuously. PySpark will be another tool that we will utilize that will serve as our analytical engine that will aid us in predicting the prices of stocks based on different indicators. Our indicators will include Moving Average (MA), RSI (Relative Strength Index) and Bollinger Bands that will assist us in making informed decisions. Below, you will see the entire workflow pipeline.



II. Problem Statement

With the ever-increasing exponential growth of financial data and demand for real-time insights, most modern methods fall short in terms of scalability and speed. These limitations in scalability and speed that other methods possess would naturally hinder the process of analyzing large amounts of data. Due to these limitations, we aim to forecast stock prices by creating a *scalable* solution with the

combined capabilities of Hadoop, Kafka and PySpark, allowing for real-time parallel processing. By utilizing these existing frameworks, we will be able to efficiently process large volumes of financial data, allowing for the most accurate predictions.

III. Justification

Prediction using Hadoop, Kafka, and Pyspark is not just a technical endeavor, but it is a strategic choice which can be used for investors, financial analysts, and even businesses that are attempting to undergo the navigation of modern-day financial markets. Typical forecasting methods are not able to streamline data as effortlessly as Kafka allows, which is extremely important in an accurate price predictor. Using Hadoop, Kafka, and PySpark, we will tackle the disadvantages that average predictors bring, and use all of the tools listed to make informed decisions on the future of different stocks. By using our approach, we provide a very practical application that can be used within the financial industry.

IV. Related Work

Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction:

- "In this study, an attempt has been made for making financial decisions such as stock market prediction, to predict the potential prices of a company's stock and to serve the need of this, Twitter data has been considering for scoring the impression that is carried for a particular firm"
- Published by Sushree Das , Ranjan Kumar Behera, Mukesh kumar , Santanu Kumar Rath

Stock market prediction: A big data approach

- This article highlights two different types of analyses: technical and fundamental. Within their analysis, they take into account the historical data of the stock along with the current sentiment, which is monitored using different social media platforms. This project involves using text mining and utilizing machine learning algorithms to predict future stock prices.
- Published by IEEE

Stock Price Prediction based on Stock Big Data and Pattern Graph Analysis:

- This paper introduces another approach to daily stock price prediction while acknowledging the challenge of the irregularity of stock prices. Within their approach, they identify similar patterns in historical stock data by clustering and then using feature selection techniques to determine significant determinants. In addition, they use RMSE in order to validate their prediction accuracy.
- Published by Seungwoo Jeon, Bonghee Hong, Hyun-jik Lee, Juhyeong Kim

V. Innovation

Our project introduces several innovative techniques to address the challenges of forecasting stock prices:

Integration of Hadoop, Kafka, and PySpark:

- Our project integrates Hadoop, Kafka, and Pyspark. Utilizing these frameworks, we then combine the power of distributed computing, real-time data streaming, and advanced analytics for up-to-date, fast, scalable, and accurate models

VI. Core Design

The data comprises historical CSV files imported using the Yahoo Finance API, supplemented with real-time data captured at one-day intervals using Apache Kafka, Zookeeper, and Python. A real-time data streaming architecture using Apache Kafka, Zookeeper, and Python is implemented. To begin, a Zookeeper server is initiated, which is essential for maintaining configuration information and providing distributed synchronization within the Kafka cluster. For reception, storage, and delivery of messages from producers to consumers a Kafka server, or broker is started. To manage the flow of data and ensure its readiness for processing, a Python module is created which interacts with a JSON file to store and manipulate a boolean 'flag' to indicate the availability of new data. The Kafka producer is configured to fetch data, parse the JSON responses, and publish the data to specified Kafka topics. This process is repeated for each stock. After this process the system enters a sleep state for one day before restarting the data fetching. Kafka consumers are set up to consume messages from the Kafka topics and write the data to CSV files. The synchronization between data production and processing is managed through the shared flag, which is set to 'True' after all stock-data are fetched and stored, showing the availability of new data. The files were imported using the API into dataframes using Pandas. A spark session was started and the minimum time frame for all data frames was found. The data was scaled to a uniform time period between '2014-09-17'and '2024-04-11' to obtain accurate analysis.

A file called methods.py was created to serve as the skeleton for the three technical analysis indicators.

1. plot_moving_average(df, stock_name)

The plot_moving_average function is a vital technical indicator. It visually represents the moving average of a stock's closing price alongside the actual closing prices over a specified period. Using the DataFrame containing historical price data (df) and the name of the stock being analyzed (stock_name), this function first ensures that the 'Date' column is in the appropriate datetime format. It then filters the data to extract information for the last year. It then calculates the moving average using a rolling window of 30 days and adds this data to the plot, providing analysts with insights into the stock's trend over a specified time.

pseudo code:

```
FUNCTION plot_moving_average(df, stock_name):

    # Ensure that the 'Date' column is in datetime format

    df['Date'] = convert_to_datetime(df['Date'])

    # Filter the dataframe to get the data for the last year

    last_year_df = filter_dataframe(df, 'Date', '>=', '2023-03-21')

    # Plot the true data

    plot(last_year_df['Date'], last_year_df['Close'], label='True Data')

    # Calculate the moving average
```

```
    last_year_df['Moving Average'] =
calculate_moving_average(last_year_df['Cl
ose'], window_size=30)

    # Plot the moving average

    plot(last_year_df['Date'],
last_year_df['Moving Average'],
label='Moving Average')

    # Set the x-axis label and title

    set_xlabel('Date')

    set_title('Moving Average vs True Data
for ' + stock_name)

    # Rotate the x-axis labels for better
readability

    rotate_x_axis_labels(45)

    # Adjust the x-axis to display dates every
30 days

    set_x_axis_locator(interval=30)

    set_x_axis_formatter('%Y-%m-%d')

    # Display the legend

    display_legend()

    # Show the plot

    show_plot()
```

2. calculate_rsi(spark_df, period=14)

The calculate_rsi function calculates an indicator that determines the Relative Strength Index (RSI) of a given stock based on its historical price. This function computes the daily price changes, distinguishing between gains and losses. By utilizing a 14 day period, it calculates the average gains and losses. Thus, deriving the RS (Relative Strength) and RSI values. These indicators provide analysts with insights into the stock's momentum and potential overbought or oversold conditions, aiding analysts in making informed trading decisions. A value over 70 would indicate that the stock is overbought while under 30 indicates its oversold.

pseudo code:

```
FUNCTION calculate_rsi(spark_df,
period=14):

    # Calculate daily price change

    window =
DefineWindow().orderBy("Date")

    spark_df =
spark_df.withColumn("Delta",
spark_df["Close"] -
Lag(spark_df["Close"], 1).over(window))

    # Separate gains and losses

    spark_df =
spark_df.withColumn("Gain",
If(col("Delta") > 0, col("Delta"),
0).Otherwise(0))

    spark_df =
spark_df.withColumn("Loss",
```

```
If(col("Delta") < 0, -col("Delta"),
0).Otherwise(0))

   # Calculate the average gain and loss

   avg_gain =
DefineWindow().orderBy("Date").rowsBet
ween(-period + 1, 0)

   avg_loss =
DefineWindow().orderBy("Date").rowsBet
ween(-period + 1, 0)

   spark_df =
spark_df.withColumn("AvgGain",
Avg(col("Gain")).over(avg_gain))

   spark_df =
spark_df.withColumn("AvgLoss",
Avg(col("Loss")).over(avg_loss))

   # Calculate RS and RSI

   spark_df = spark_df.withColumn("RS",
col("AvgGain") / col("AvgLoss"))

   spark_df = spark_df.withColumn("RSI",
100 - (100 / (1 + col("RS"))))

   return spark_df
```

3. plot_rsi(df, stock_name)

The plot_rsi function visually represents the Relative Strength Index (RSI) of a stock over a defined period. This function first converts the Spark DataFrame to a Pandas DataFrame for ease of manipulation and filters the data to extract RSI values for the last year, ensuring relevance to recent market conditions. Subsequently, it plots the RSI values against time, providing analysts with insights into the stock's strength and potential reversal points. Furthermore, it adds horizontal lines at 70 and 30 on the RSI scale to indicate overbought and oversold levels, facilitating interpretation and decision-making.

Pseudo code:

```
FUNCTION plot_rsi(df, stock_name):

   # Convert Spark DataFrame to Pandas
DataFrame

   df_pd = convert_to_pandas(df)

   # Ensure that the 'Date' column is in
datetime format

   df_pd['Date'] =
convert_to_datetime(df_pd['Date'])

   # Filter the dataframe to get the data for
the last year

   last_year_df = filter_dataframe(df_pd,
'Date', '>=', current_date() -
date_offset(years=1))

   # Plot the RSI

   create_figure(size=(12, 6))
```

```
    plot(last_year_df['Date'],
last_year_df['RSI'], label='RSI')

    # Set the x-axis label, y-axis label, and
title

    set_xlabel('Date')

    set_ylabel('RSI')

    set_title('RSI for ' + stock_name + ' (Last
Year)')

    # Set y-axis range to [0, 100] for RSI

    set_ylim(0, 100)

    # Add a horizontal line at 70 and 30 to
indicate overbought and oversold levels

    add_horizontal_line(70, color='r',
linestyle='--', label='Overbought (70)')

    add_horizontal_line(30, color='g',
linestyle='--', label='Oversold (30)')

    # Rotate the x-axis labels for better
readability

    rotate_x_axis_labels(45)

    # Display the legend

    display_legend()

    # Show the plot

    show_plot()
```

4. plot_bollinger_bands_last_year(df, stock_name, window=20, num_std=2)

The plot_bollinger_bands_last_year function is pivotal in analyzing stock price volatility and identifies potential trading opportunities. Given a Spark DataFrame (df) containing historical price data, the name of the stock (stock_name), and optional parameters such as the window size and the number of standard deviations. This function computes and plots Bollinger Bands for the last year. By calculating the moving average and standard deviation over the specified window, it delineates upper and lower bands around the moving average, representing potential price ceilings and floors. This visualization aids analysts in identifying periods of high volatility and potential price reversals, thereby assisting in devising effective trading strategies.

Pseudo code:

```
FUNCTION
plot_bollinger_bands_last_year(df,
stock_name, window=20, num_std=2):
    # Convert the PySpark DataFrame to a
Pandas DataFrame
    df_pd = convert_to_pandas(df)

    # Ensure that the 'Date' column is in
datetime format
    df_pd['Date'] =
convert_to_datetime(df_pd['Date'])

    # Filter the dataframe to get the data for
the last year
    last_year_df = filter_dataframe(df_pd,
'Date', '>=', current_date() -
date_offset(years=1))

    # Calculate the moving average and
standard deviation
    last_year_df['Moving_Average'] =
```

```
calculate_moving_average(last_year_df['Close'], window=window)
    last_year_df['Standard_Deviation'] = calculate_standard_deviation(last_year_df['Close'], window=window)

    # Calculate the upper and lower Bollinger Bands
    last_year_df['Upper_Band'] = last_year_df['Moving_Average'] + (last_year_df['Standard_Deviation'] * num_std)
    last_year_df['Lower_Band'] = last_year_df['Moving_Average'] - (last_year_df['Standard_Deviation'] * num_std)

    # Plot the closing price
    plot(last_year_df['Date'], last_year_df['Close'], label='Close', color='blue')

    # Plot the moving average
    plot(last_year_df['Date'], last_year_df['Moving_Average'], label='Moving Average', color='green')

    # Plot the upper and lower Bollinger Bands
    plot(last_year_df['Date'], last_year_df['Upper_Band'], label='Upper Band', color='red', linestyle='--')
    plot(last_year_df['Date'], last_year_df['Lower_Band'], label='Lower Band', color='red', linestyle='--')

    # Set the title and labels
    set_title('Bollinger Bands for ' + stock_name + ' (Last Year)')
    set_xlabel('Date')
    set_ylabel('Price')

    # Rotate the x-axis labels for better readability
    rotate_x_axis_labels(45)

    # Display the legend
    display_legend()

    # Show the plot
    show_plot()
```

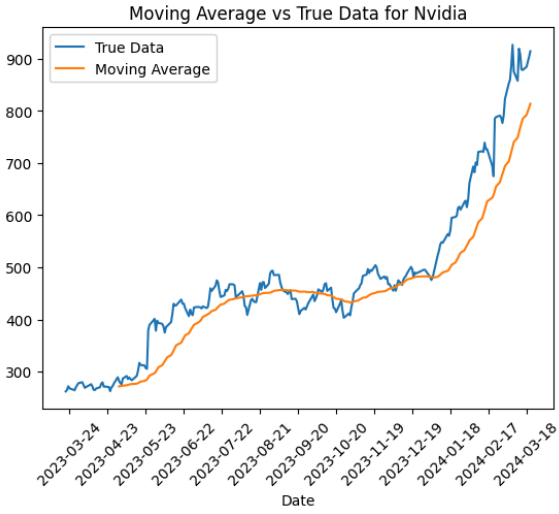
A file called 3_Project.ipynb was created to deploy the analysis tool. The script is designed to fetch, process, and visualize stock data for any given ticker in real-time. The script iterates over a dictionary containing stock names and their corresponding historic data. For each ticker, it reads the historical data CSV file and focuses on the 'Close' and 'Date' columns. Using a flag mechanism it continuously checks for new data and updates the data frame accordingly. Data Processing and Analysis: Upon receiving new data, it appends it to the existing dataframe. It then calls the requested technical indicators. Waiting: The script iterates over the stocks continuously with a wait time of 20 seconds between iterations, checking for new data. After processing data for all stocks, it sets the flag to indicate that new data has been processed. This approach allows for the handling of multiple stocks simultaneously and demonstrates the real-time fetching, processing, and visualization of financial data.

VII. Evaluation

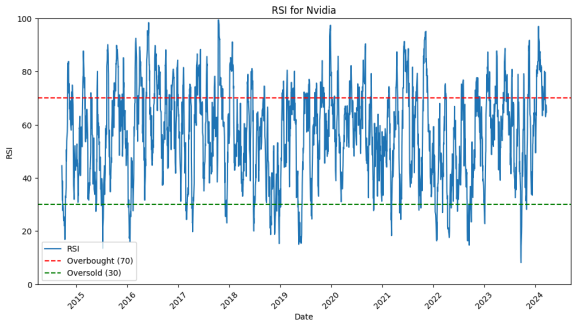
Using our described methods, we were able to make an informed prediction on the direction the stocks we chose will be moving. In the charts below, we will present the results for each of our indicators (RSI, Bollinger Bands, MA) for Nvidia and give an informed prediction on the short-term price movement.

Nvidia

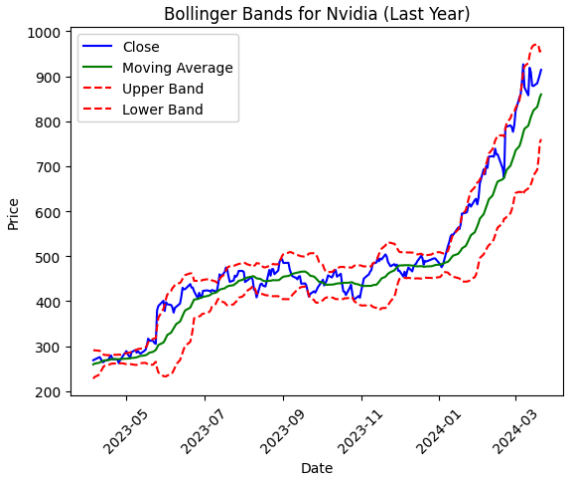
Based on evaluating the MA, Nvidia has shown strong signs of performance. Since March of 2023, Nvidia has traded above its MA for the most part, which suggests a bullish uptrend in the stock's price movement. In addition, the company's sustained ability to trade above MA indicates that the company itself has strong financial performance. Simply based on this indicator alone, Nvidia poses a compelling investment opportunity due to its bullish sentiment and strong fundamentals.



Below, you can see the RSI bands from 2015 to 2024. RSI is a vital tool when it comes to determining trends in Nvidia's (and others) stock price. The RSI gives valuable insights regarding future movements, current sentiment, and market dynamics. By demonstrating the magnitude of recent changes in the stock, you can determine periods that the stock is overbought, and when there is likely to be a downward retracement. In 2024, you can see that RSI is relatively high, and almost signals an overbought condition. With this information in mind, it is possible that nvidia may be facing a retracement of some sort soon, and that is necessary information to keep in mind.



The bollinger bands for Nvidia are composed of an upper band (upper resistance) and lower band (lower resistance) as well as a moving average, which is the green middle band. In 2024, Nvidia is nearing the upper resistance, which can be seen as a potential ceiling for the price movement and further supports the RSI's notion that Nvidia is likely to see some sort of retracement in the near future. Although some sort of retracement is likely to happen, it will more than likely bounce around the MA and will continue to move up, as the MA shows a bullish uptrend.

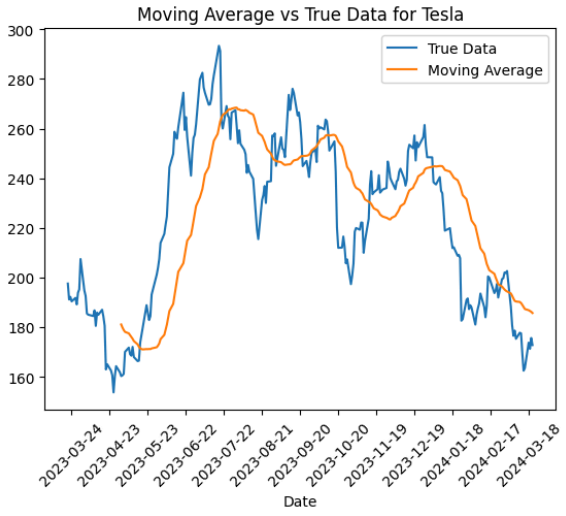


Based on the analysis from the Bollinger Bands, Relative Strength Index (RSI), and MA, Nvidia is likely to experience a retracement in the short-term from its current levels. However,

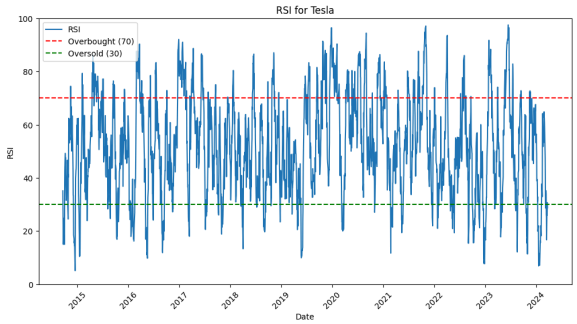
the retracement will more than likely be temporary and Nvidia will continue its upward trend based off the bullish MA indicator. Therefore, it may be wise to short the stock in the short term, but Nvidia is a great candidate for a long-term hold.

Tesla

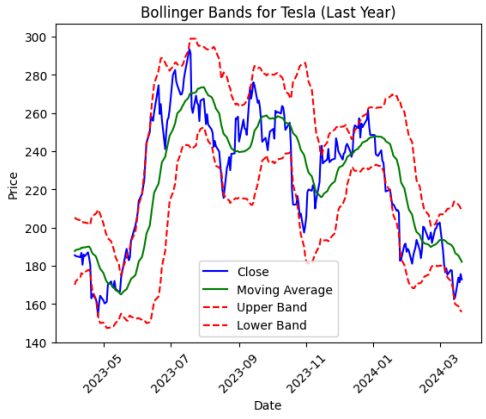
At the beginning of March 2023, up until June 2023, Tesla's MA suggested a bullish uptrend, as it traded significantly above the MA for a long period of time. The tables turned, though, as from July 2023 to March 2024, Tesla has been consistently trading below MA. With the performance of this indicator, it can be inferred that Tesla is currently in a downtrend based on price history. Future price cannot be predicted on just MA alone, though, so we need to analyze both RSI for oversold/overbought conditions and use Bollinger Bands to determine resistance.



The RSI history for Tesla from 2015 to 2024 can be seen, and it is another vital indicator that will assist us in our prediction. Since 2015, Tesla has traded within a wide range of RSI values, which demonstrate how volatile price movements for Tesla can be. It is quite noticeable that in March 2024, Tesla is trading in the 'oversold zone', which suggests that the stock may be undervalued compared to its past performance. This could signal a buying opportunity for investors as it points to a potential reversal. With this in mind, it may be a good time to buy.



By examining Tesla's Bollinger Bands based on price history from May 2023, it is evident that Tesla has only traded within the bands, which shows that there is strong resistance whenever Tesla approaches the upper band, and strong support when it nears the lower band. This shows that Tesla's movements have traded within a stable range, and that fluctuations in price can be captured within the width of the bands. In March of 2024, Tesla's price is nearing support, and it is more likely to bounce off the band into an uptrend until it nears upper resistance. However, it is important to note that Bollinger Bands are dynamic and can respond to changes in volatility, so it is extremely important to note that bands should continually be used for identifying whether a breakout/breakdown in price is soon to occur.



Taking into account all of our indicators (MA, RSI, and Bollinger Bands), you can come to the conclusion that in the short term (since March 2023), Tesla has been on a downtrend as it has consistently been trading below MA. However, when looking at the RSI on the larger scale, it can be concluded that Tesla is severely undervalued in comparison to their past price history. Based strictly on the analysis of these indicators, it is a good conclusion to infer that Tesla may continue to keep dropping in the short term while staying above the bottom Bollinger band, but in the longer term Tesla is quite likely to make a rebound.

VIII. Conclusions

In conclusion, this paper has explained our approach in utilizing Hadoop, Kafka, and Pyspark for our stock predictor. By analyzing the past performance of our chosen stocks and index funds our approach has proven effective in providing accurate predictions for investors.

In our integration of Hadoop, Kafka, and PySpark, we have addressed the challenges of scalability, speed, and real-time data processing, which enables us to be as efficient as possible while handling large volumes of data. In addition, our use of Bollinger Bands, Relative Strength Index (RSI), AND Moving Average (MA) indicators further enhance our prediction accuracy. These predictor determine market trends, reversals, resistance, and provide insights for investors

In summary, we have presented an approach to stock price forecasting that leverages advanced techniques and methods. Through our innovation, we have presented a methodology that represents advancements in stock-market analytics, and offers valuable benefits for investors to make informed financial decisions in the stock market.

IX. References

1. Sushree Das, Ranjan Kumar Behera, Mukesh kumar, Santanu Kumar Rath, Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction, Procedia Computer Science, Volume 132, 2018, Pages 956-964, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.111.
2. G. V. Attigeri, Manohara Pai M M, R. M. Pai and A. Nayak, "Stock market prediction: A big data approach," TENCON 2015 - 2015 IEEE Region 10 Conference, Macao, China, 2015, pp. 1-5, doi: 10.1109/TENCON.2015.7373006. keywords: {Companies;Market research;Prediction algorithms;Algorithm design and analysis;Annealing;Support vector machines;Big data;prediction;social media analytics;machine learning},
3. Jeon S., Hong B., Lee H. and Kim J. (2016). **Stock Price Prediction based on Stock Big Data and Pattern Graph Analysis** . In *Proceedings of the International Conference on Internet of Things and Big Data - Volume 1: IoTBD,* ISBN 978-989-758-183-0, pages 223-231. DOI: 10.5220/0005876102230231