Automated Technical Analysis for Efficient Financial Market Analysis

Vijay Choudhary

Avantika University

Machine Learning

3 February 2024

Abstract

Detecting stock chart patterns, such as channels, triangles, and head and shoulders formations, within high-dimensional stock data poses a formidable challenge. Experts in technical analysis often rely on their experience to visually identify these patterns in candlestick charts and make predictions regarding stock price movements. However, the subjective nature of this process can lead to variations in pattern interpretation among different analysts, influenced by factors like scale and timeframe. Moreover, the inherent volatility of stock prices adds another layer of complexity to the task of pattern recognition in the stock market.

To address these challenges, various machine learning algorithms have been explored for the automated detection of stock chart patterns. Among these approaches, linear regression stands out as a particularly efficient method for handling dynamic stock data and accurately representing patterns on charts. By leveraging linear regression, we aim to minimize human errors associated with subjective pattern interpretation and streamline the process of pattern identification.

In this paper, we present our implementation of linear regression for detecting triangle patterns in stock charts. We focus on analyzing the top ten stocks from both the Information Technology and Pharmaceutical sectors listed on the Indian National Stock Ex-

change. Our methodology achieves an impressive average accuracy of 98.60% in identifying triangle patterns within these datasets.

By automating the process of pattern detection through linear regression, we not only enhance the efficiency of stock analysis but also reduce the potential for subjective biases inherent in manual interpretation. This approach holds promise for improving decision-making in stock trading and investment strategies, offering valuable insights into market trends and price movements.

Introduction

Financial markets are always changing, offering both opportunities and challenges for traders and investors. However, analyzing all the data manually to find the best investment options takes a lot of time and effort. This involves carefully looking at stock charts, using different indicators, and spotting patterns, which can be mentally taxing. Plus, documenting and reporting all this analysis adds even more work, making it hard to make quick decisions.

To tackle these issues, we're creating an Automated Technical Analysis System. It's like having a smart assistant that uses historical stock data and advanced algorithms to do all the analysis work for you. Our goal is to make this system precise, efficient, and scalable, so it can take the burden off manual analysis. We want to empower traders and investors with insights based on data, improving their investment decisions.

This project is important because it can make decision-making faster, reduce mistakes, and give users valuable insights from complex data analysis. By automating tasks like interpreting price movements and spotting important patterns in stock charts, we aim to make advanced analysis tools more accessible to everyone. We're building an easy-to-use interface using Python and popular libraries like pandas and numpy. With ma-

chine learning models, we'll analyze price actions and recognize patterns, making it easier for users to understand market trends. Ultimately, we hope to change how people navigate financial markets, making investment strategies more efficient and effective in today's competitive environment.

Methodology

Obtained historical stock data from Yahoo Finance using the `yfinance` library for our analysis. This dataset contains information about stock prices, including opening, high, low, and closing prices, along with timestamps for each data point. The size of the dataset varies depending on the specified parameters such as the day range and time interval passed to the `fetch_stock_data` function.

Data Retrieval:

Retrieved historical stock data using the `fetch_stock_data` function, specifying parameters like the ticker symbol, desired day range, and time interval.

Data Cleaning:

- Ensured data integrity by removing any missing values from the dataset using the dropna() method.
- Subsequently, we transformed the dataset into a DataFrame format to facilitate further processing.

Feature Engineering:

- Relevant features such as 'Open', 'High', 'Low', and 'Close' prices were selected from the dataset to focus on essential stock price metrics.
- To enhance clarity, we reset the index of the DataFrame.

Additional Feature Creation:

Introduced two new columns to the DataFrame:

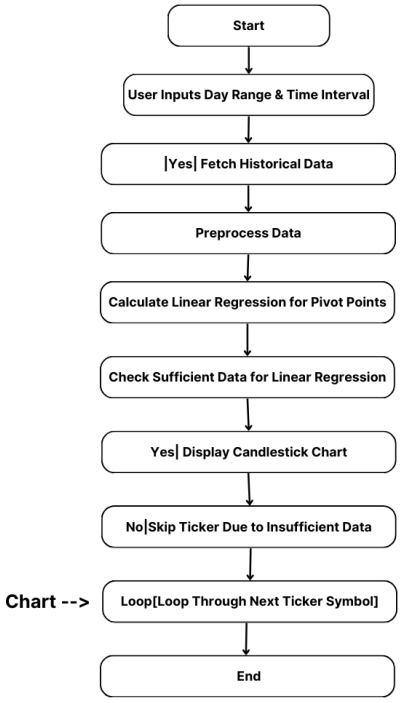
- 'pivot': Calculated using the `pivot_id` function with predefined parameters for pivot calculation.
- 'pointpos': Determined by the `point_position` function, which assesses the position of data points within the dataset.

Linear Regression Calculation Function:

Developed a function named **calculate_linear_regression** to compute linear regression models specifically tailored for low and high pivot points. This function is designed to operate on a DataFrame containing stock data, requiring parameters such as the number of previous candles to consider and an optional test size for evaluating the model's performance.

Upon invocation, the function splits the dataset into training and testing subsets. Subsequently, it proceeds to train distinct linear regression models for both low and high pivot points. These models are trained separately to capture the unique characteristics associated with each pivot type.

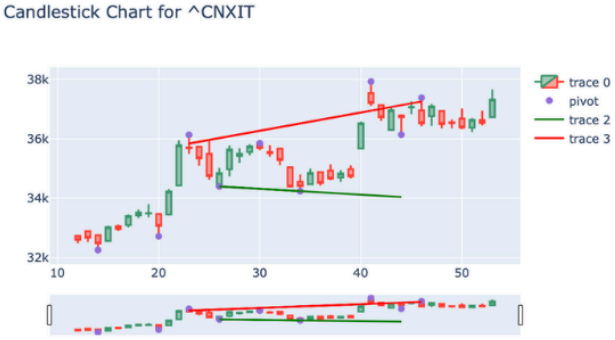
Upon completion of the training process, the function returns essential parameters such as coefficients and intercepts, which are vital for further analysis and I nterpretation of the linear regression models. These parameters provide valuable insights into the relationship between historical stock data and pivot points, aiding traders and investors in making informed decisions based on statistical analysis.



Patterns Visualisation:


```
# Print R-squared values
print(f"R-squared (Train) - Low: {r_squared_min_train:.2f}%, High: {r_squared_max_train:.2f}%")
print(f"R-squared (Test) - Low: {r_squared_min_test:.2f}%, High: {r_squared_max_test:.2f}%")
```

```
R-squared (Train) - Low: 0.97%, High: 1.00%
R-squared (Test) - Low: nan%, High: nan%
```



Conclusion

In conclusion, theAutomated Technical Analysis System marks a significant advancement in transforming stock market trading and investment practices. By automating technical analysis, it tackles challenges in the financial landscape, providing traders with a powerful decision-making tool. Through machine learning and historical data, we've created a comprehensive approach to analyze stock charts, identify patterns, and offer actionable insights. Our goal is to democratize access to advanced analysis tools, making them user-friendly through Python programming. Looking ahead, our system holds promise in driving innovation and efficiency in trading, poised to adapt to evolving markets through ongoing refinement.

References

https://www.investopedia.com/articles/technical/112601.asp

https://www.jpmorgan.com/technology/technology-blog/searching-for-patterns

https://finance.yahoo.com/quote/%5ENSEBANK/history?p=%5ENSEBANK

https://plotly.com/python/candlestick-charts/

https://ieeexplore.ieee.org/document/10146731/metrics#metrics