Stock Trading Algorithm and BI Tool Implementation

Taran Arora
Computer Science
Okanagan College
Kelowna, Canada
taransarora@gmail.com

Harsh Saw
Computer Science
Okanagan College
Kelowna, Canada
Mister.harshkumar@gmail.com

Parag Jindal
Computer Science
Okanagan College
Kelowna, Canada
Paragjindal023@gmail.com

Kartik Arora
Computer Science
Okanagan College
Kelowna, Canada
kartikarora0001@icloud.com

Allen Dior
Computer Science
Okanagan College
Kelowna, Canada
allendior.ca@gmail.com

***Abstract*— This paper presents the design, implementation, and evaluation of a stock trading algorithm utilizing the Moving Average Crossover (MAC) strategy. The primary goal of the project is to automate the process of identifying profitable trading opportunities by generating buy and sell signals based on the crossing of short-term and long-term moving averages. The system is built using historical stock price data stored in a PostgreSQL data warehouse, with the data processed and analyzed in Python. The results of the algorithm are visualized through Tableau for business intelligence reporting.**

I. Introduction

The rapid advancement of technology and the growing availability of large datasets have revolutionized the field of finance, particularly in the development of automated trading systems. These systems use algorithms to analyze market data, identify patterns, and execute trading decisions without human intervention. The focus of this project is to implement an automated stock trading strategy using the **Moving Average Crossover (MAC)** technique, a widely-used technical analysis tool in financial markets.

The **Moving Average Crossover (MAC)** strategy involves the use of two moving averages: a short-term moving average and a long-term moving average. A **buy signal** is generated when the short-term moving average crosses above the long-term moving average, indicating an upward trend. Conversely, a **sell signal** is triggered when the short-term moving average crosses below the long-term moving average, suggesting a downward trend. This strategy aims to identify key points in the market to maximize returns while minimizing risk

Furthermore, the project integrates **Business Intelligence (BI) tools**, specifically **Tableau**, to visualize the performance of the algorithm, helping stakeholders understand the algorithm's effectiveness in different market conditions.

This report covers the project's methodology, from data collection and processing to the development of the algorithm and the creation of the visualization dashboards. Additionally, we discuss the challenges encountered, the evaluation of the algorithm, and future improvements that could enhance the system's performance and robustness.

II. Existing Works

A. Algorithmic Trading and Strategies

Algorithmic trading, often referred to as **automated trading** or **black-box trading**, is the use of computer algorithms to automate the process of trading financial securities. These algorithms are designed to execute trades based on predefined criteria, without requiring manual intervention. The main objective of algorithmic trading is to leverage computational speed, pattern recognition, and quantitative analysis to improve trading decisions.

Over the past two decades, algorithmic trading has become a dominant strategy in global financial markets. Numerous algorithmic strategies have been proposed in academic literature, each with its own strengths and limitations. Broadly speaking, the main categories of algorithmic trading strategies include:

- **Trend-following strategies**: These strategies are designed to identify and follow trends in the market. Popular examples include **Moving Average Crossover** and **Momentum Strategies**.
- **Mean-reversion strategies**: These strategies assume that asset prices will tend to revert to a historical mean or average over time. One example is the **Bollinger Bands** strategy.
- **Arbitrage strategies**: These strategies aim to exploit price discrepancies between related assets in different markets or instruments.

While trend-following and mean-reversion strategies are most commonly employed, this project focuses on the **Moving Average Crossover (MAC)** strategy, which is widely recognized for its simplicity and ease of implementation in algorithmic trading.

B. Moving Average Crossover Strategy

The **Moving Average Crossover (MAC)** strategy is one of the simplest and most widely used technical analysis tools in financial markets. It operates on the principle of comparing the prices of short-term and long-term moving averages to identify market trends.

- **Short-term vs Long-term moving averages**: In the MAC strategy, a **short-term moving average** (e.g., 50-day moving average) is compared with a **long-term moving average** (e.g., 200-day moving average). The idea is that a short-term moving average reacts faster to recent price changes, while the long-term moving average captures the broader market trend.
- **Crossovers**: The key feature of the MAC strategy is the **crossover** event:
    - **Buy Signal**: When the short-term moving average crosses above the long-term moving average, it indicates a potential uptrend, signaling a buying opportunity.
    - **Sell Signal**: Conversely, when the short-term moving average crosses below the long-term moving average, it signals a potential downtrend, prompting a selling decision.
- **Advantages and Limitations**:
    - The **advantages** of the MAC strategy include its simplicity, ease of implementation, and ability to identify key trends in the market. Additionally, the MAC strategy can be used across different markets and asset classes, from stocks to commodities to cryptocurrencies.
    - However, **limitations** include its tendency to generate **lagging signals** during periods of high market volatility, as the strategy reacts to past price data. Additionally, in highly volatile or sideways markets, the MAC strategy may generate a **high number of false signals**, leading to unnecessary trades and losses.

Despite these limitations, the Moving Average Crossover remains a popular choice due to its ability to capture long-term market trends and is often used in combination with other technical indicators to filter out false signals.

C. Backtesting in Algorithmic Trading

Backtesting is a crucial component in the development of algorithmic trading strategies. It refers to the process of testing a trading strategy using historical market data to evaluate its potential performance before implementing it in live trading. Backtesting allows traders and developers to assess the effectiveness of a strategy in a risk-free environment.

Key steps in the backtesting process include:

- **Historical Data Collection**: Collecting accurate and comprehensive historical price data for the assets of interest. This includes both price data (open, close, high, low) and volume data.
- **Trade Simulation**: Simulating the execution of trades based on the signals generated by the algorithm using historical data. This includes simulating the timing of orders, order execution, and transaction costs.
- **Performance Evaluation**: Evaluating the strategy's performance based on key metrics such as **total returns**, **win rate**, **maximum drawdown**, and **Sharpe ratio**.

Backtesting helps to identify potential flaws in the algorithm, such as overfitting to historical data or issues with trade execution. It also allows the trader to assess whether the strategy aligns with their risk tolerance and investment goals.

However, backtesting is not without challenges. Some of the key issues include:

- **Overfitting**: Designing a strategy that works too well on historical data but fails in live trading due to market changes.
- **Lookahead Bias**: The use of data that would not have been available at the time of trade execution, resulting in unrealistic performance results.
- **Data Snooping**: The tendency to use historical data to design a strategy and then test it on the same data, which can lead to biased results.

D. Data Storage and Visualization

The effectiveness of any algorithmic trading system is contingent upon its ability to handle large amounts of financial data and present it in an easily interpretable format. In this project, the data is stored in a **PostgreSQL** database and visualized using **Tableau**.

- **PostgreSQL**: A powerful, open-source relational database management system, PostgreSQL is used to store historical stock data, trading signals, and backtesting results. The data schema is designed to facilitate efficient querying, aggregation, and analysis. Key tables in the database include the **Stock Dimension**, **Date Dimension**, **Sector Dimension**, and **Fact Stock Data**.
- **Data Integration**: Data is ingested from external sources, such as financial APIs (e.g., Yahoo Finance), and loaded into the PostgreSQL database. The data is then processed to calculate technical indicators such as **SMA**, **RSI**, and **Bollinger Bands**.
- **Tableau**: Tableau is used to create interactive visualizations and dashboards that display key performance metrics, trends, and trading signals. With Tableau, users can visualize historical stock price movements, moving average crossovers, and the results of backtested strategies. Dashboards can be customized to show:

Identify applicable funding agency here. If none, delete this text box.

- **Candlestick charts** to visualize price movements.
- **Moving average crossover signals** to identify trading opportunities.
- **Performance metrics** such as total returns, win rate, and drawdown.

Data visualization tools like Tableau are essential in making complex trading strategies and their performance easy to understand for stakeholders, helping inform trading decisions.

III. Methodology

This section provides a detailed description of how the project was approached, the steps followed in designing the system, the tools and technologies used, and how the various components of the project were integrated. It explains the workflow of the algorithmic trading system from data collection to backtesting and finally to the visualization of trading performance.

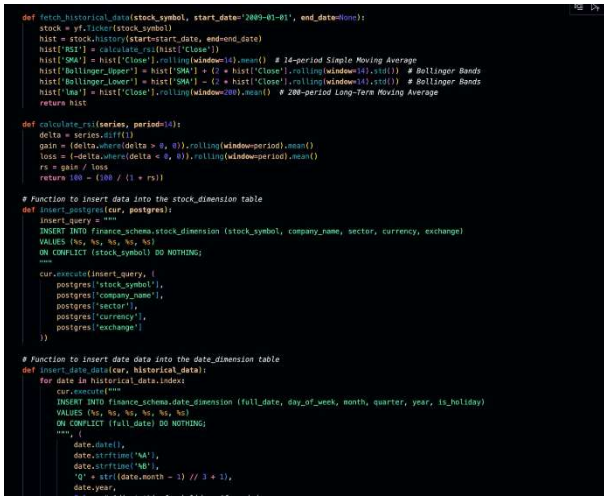
A. Data Collection and Preparation

The foundation of the trading system relies on acquiring historical stock price data and technical indicators. To obtain this data, **financial API** such as Yahoo Finance was used. This API allow us to retrieve various stock-related data points, including opening, closing, high, and low prices, as well as trading volume.

- **Data sources**:
  - **Yahoo Finance API**: Used to collect historical stock data for the assets being analyzed.

The data was then **cleaned and processed** for further use in the trading algorithm:

- **Missing values** were handled using interpolation or dropping rows, depending on the data type.
- **Data normalization** was performed to ensure consistency across various stock datasets.

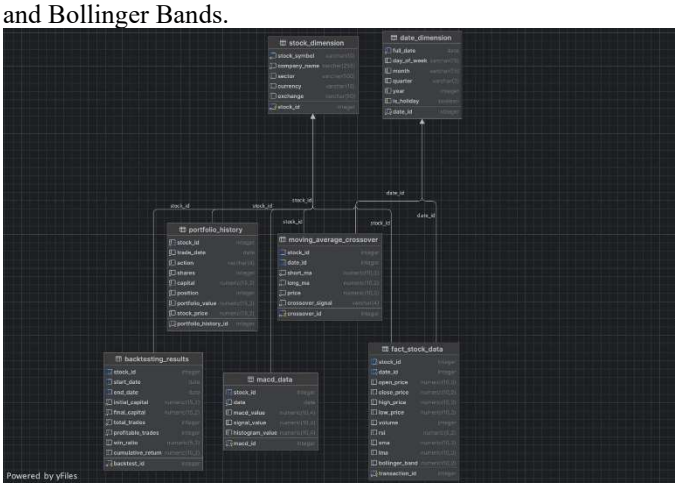


```
def fetch_historical_data(stock_symbol, start_date='2009-01-01', end_date=None):
    stock = yf.Ticker(stock_symbol)
    hist = stock.history(start=start_date, end=end_date)
    hist['RSI'] = calculate_rsi(hist['Close'])
    hist['SMA'] = hist['Close'].rolling(window=14).mean()  # 14-period Simple Moving Average
    hist['Bollinger_Upper'] = hist['SMA'] + (2 * hist['Close'].rolling(window=14).std())  # Bollinger Bands
    hist['Bollinger_Lower'] = hist['SMA'] - (2 * hist['Close'].rolling(window=14).std())  # Bollinger Bands
    hist['lma'] = hist['Close'].rolling(window=200).mean()  # 200-period Long-Term Moving Average
    return hist

def calculate_rsi(series, period=14):
    delta = series.diff(1)
    gain = (delta.where(delta > 0, 0)).rolling(window=period).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=period).mean()
    rs = gain / loss
    return 100 - (100 / (1 + rs))

# Function to insert data into the stock_dimension table
def insert_postgres(cur, postgres):
    insert_query = """
    INSERT INTO finance_schema.stock_dimension (stock_symbol, company_name, sector, currency, exchange)
    VALUES (%s, %s, %s, %s, %s)
    ON CONFLICT (stock_symbol) DO NOTHING;
    """
    cur.execute(insert_query, (
        postgres['stock_symbol'],
        postgres['company_name'],
        postgres['sector'],
        postgres['currency'],
        postgres['exchange']
    ))

# Function to insert date data into the date_dimension table
def insert_date_data(cur, historical_data):
    for date in historical_data.index:
        cur.execute("""
        INSERT INTO finance_schema.date_dimension (full_date, day_of_week, month, quarter, year, is_holiday)
        VALUES (%s, %s, %s, %s, %s, %s)
        ON CONFLICT (full_date) DO NOTHING;
        """, (
            date.date(),
            date.strftime('%A'),
            date.strftime('%B'),
            'Q' + str((date.month - 1) // 3 + 1),
            date.year,
```

B. Database Design and Structure

To efficiently store and retrieve the stock data, a **PostgreSQL database** was used. The database schema was designed to include the following key tables:

- **Stock Dimension Table**: Stores information about the stocks being analyzed, including their symbols, company names, sector, currency, and exchange.
- **Date Dimension Table**: Stores information about the dates for each stock transaction, including the full date, day of the week, month, quarter, year, and holiday status.
- **Sector Dimension Table**: Stores details about the sector to which each stock belongs.
- **Fact Stock Data Table**: Stores the actual stock price data, including open, close, high, low prices, volume, and calculated technical indicators such as SMA, RSI, and Bollinger Bands.



C. Implementation of the Moving Average Crossover Strategy

The **Moving Average Crossover** (MAC) strategy was implemented to identify buy and sell signals based on the crossover of short-term and long-term moving averages.

- **Short-term Moving Average (SMA)**: A 50-day moving average.
- **Long-term Moving Average (SMA)**: A 200-day moving average.

The algorithm works as follows:

1. **Buy Signal**: When the short-term moving average crosses above the long-term moving average, a buy signal is generated.
2. **Sell Signal**: When the short-term moving average crosses below the long-term moving average, a sell signal is generated.

The implementation involved:

- **Fetching the moving averages** for each stock over a specified window (e.g., 50 days and 200 days).
- **Signal generation** based on the crossover logic.
- Storing the signals along with the trading data in the **PostgreSQL database**.

The algorithm was designed to generate trade signals and execute simulated trades using historical data. **Position**

sizes and **portfolio allocation** were also considered to simulate realistic trading conditions.

D. Backtesting the Algorithm

Backtesting is a crucial step in validating the performance of a trading strategy. The backtesting system was designed to test the MAC strategy using historical stock data stored in the **PostgreSQL database**.

Steps involved in backtesting:

1. **Simulated Trading**: The algorithm was run on historical data to generate buy and sell signals.
2. **Portfolio Simulation**: A portfolio was simulated based on the generated signals. For simplicity, a 100% allocation was used for each trade.
3. **Metrics Calculation**: The performance was evaluated based on key metrics such as:
    - **Total Return**: The percentage return on the portfolio over the test period.
    - **Win Rate**: The percentage of profitable trades.
    - **Maximum Drawdown**: The largest loss from peak to trough in the portfolio.
    - **Sharpe Ratio**: The risk-adjusted return of the strategy.

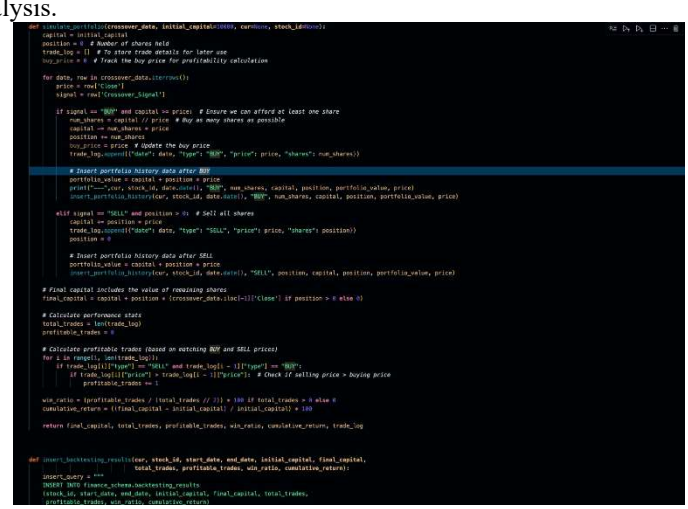
The backtesting system allowed for the testing of different strategies and optimization of parameters, such as the look-back period for moving averages and risk management techniques.

E. Data Visualization and Dashboard

Once the trading algorithm and backtesting results were generated, the next step was to visualize the results for analysis and decision-making. **Tableau** was used as the Business Intelligence (BI) tool for visualizing key metrics and performance indicators.

- **Candlestick Charts**: Used to visualize stock price movements over time.
- **Moving Average Crossovers**: Displayed on price charts to visually represent buy and sell signals.
- **Performance Metrics Dashboards**: Included visualizations such as line charts for total return, bar charts for win rate, and drawdown analysis.

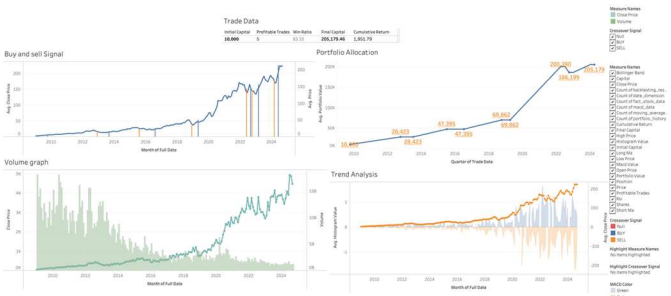
The visualizations allowed users to assess the effectiveness of the trading algorithm in an interactive and user-friendly manner. The dashboards were continuously updated with the most recent data, enabling real-time analysis.



IV. Results and Discussion

A. Backtesting Results and Visualizations

Backtesting results were visualized using **Tableau** to provide a comprehensive view of the strategy's performance over time. The following visualizations were created:



1. Buy and Sell Signal Analysis

The **Buy and Sell Signal** graph is an essential part of evaluating the effectiveness of the Moving Average Crossover strategy. This line chart represents the average stock price over time, with key points indicating when the algorithm triggered buy or sell signals. The orange and blue vertical markers correspond to the buy and sell signals, respectively, based on the Moving Average Crossover conditions.

- **Interpretation**:

- Buy signals occur when the short-term moving average crosses above the long-term moving average, suggesting a potential upward trend and an opportunity to buy.
- Sell signals appear when the short-term moving average crosses below the long-term moving average, signaling a potential downward trend and an opportunity to sell.

- **Performance**: The chart shows that buy and sell signals align with key stock price movements, confirming the viability of the strategy. A well-performing strategy should ideally show a consistent increase in stock price following buy signals and a decrease or stabilization after sell signals.

2. **Volume Graph:**
   The **Volume Graph** provides a visualization of the volume of trades executed over time. This is displayed alongside the average stock price, which gives insights into the trading activity relative to price movements.

- **Interpretation**:
  - High trading volumes often coincide with significant price movements, both upward and downward. This might indicate that market participants are reacting to news, earnings reports, or other factors.
  - The rising trend in the volume graph around the time of key buy and sell signals further supports the idea that higher volume often precedes or follows significant price movements, helping confirm the algorithm's predictions.
- **Analysis**: The volume spikes are indicative of the market's reaction to price fluctuations, which provides insight into investor sentiment. Increased volume during uptrends may indicate strong market confidence, while volume increases during downtrends might signal panic selling.

3. **Portfolio Value Over Time:**
   The **Portfolio Allocation** graph illustrates the growth of the portfolio value over time, starting with an initial investment of $10,000. The graph shows how the portfolio value increases as a result of the buy and sell decisions made by the algorithm.

- **Interpretation**:
  - The portfolio value steadily increases over time, peaking around 2024. This suggests that the Moving Average Crossover strategy is profitable in the long term, with the portfolio growing from an initial capital of $10,000 to over $200,000.
  - Periodic dips in the portfolio value coincide with the market's downturns, which may reflect less favorable market conditions, such as sell-offs triggered by the algorithm.
- **Performance**:
  - The upward trajectory in portfolio value suggests that the algorithm's decision-making is relatively accurate, although there may be periods where more volatile conditions lead to temporary losses. A more detailed risk assessment (e.g., drawdown analysis) would provide better insight into how the algorithm handles market corrections.

4. **Trend Analysis:**

The **Trend Analysis** graph overlays the stock price and some technical indicators to showcase the stock's performance relative to its underlying trend.

- **Interpretation**:
  - The trend analysis shows that the stock has experienced a significant uptrend, especially from 2022 onwards. The MACD (Moving Average Convergence Divergence) indicator, represented by shaded areas, highlights the periods of bullish (green) and bearish (red) momentum.
  - The orange and blue lines represent the stock's price fluctuations, and the histogram beneath indicates the intensity of momentum, with green indicating stronger bullish momentum and red indicating bearish momentum.
  - The chart suggests that the Moving Average Crossover strategy is effective during trending periods, particularly when the stock is in an upward trend, providing significant profits.

V. Future Works

- **Refinement of Buy/Sell Logic**: Implementing additional indicators such as the Relative Strength Index (RSI) or Bollinger Bands could refine buy and sell logic, providing more accurate signals.
- **Risk Management**: Incorporating risk management tools like stop-loss orders or position sizing strategies could help protect the portfolio from large drawdowns during market corrections.
- **Incorporating Other Data Sources**: Expanding the data set to include macroeconomic indicators or company-specific news might improve the algorithm's prediction power and adaptability to different market conditions.

VI. Conclusion

This project focused on developing a comprehensive trading algorithm using the Moving Average Crossover strategy, integrated with a data warehouse for storing historical stock data and performance metrics. The primary objective was

to create an algorithm that identifies key buy and sell signals based on short-term and long-term moving averages, with the ultimate goal of generating consistent returns.

The project utilized data sourced from Yahoo Finance, which was processed and stored in PostgreSQL. This data was visualized using Tableau, providing key insights into the performance of the trading algorithm over time. The moving average crossover strategy successfully demonstrated its ability to predict upward and downward trends, providing profitable buy and sell signals.

The backtesting process was an essential part of validating the strategy's performance. By simulating historical trades based on past data, we were able to assess the profitability of the algorithm and identify key periods where the strategy performed well or faced challenges. The results showed that the strategy is particularly effective during trending markets, with substantial returns observed during bullish trends.

Key visualizations, such as the Buy and Sell Signal chart, Portfolio Allocation graph, and Trend Analysis, highlighted the efficiency of the strategy in identifying profitable trading opportunities. Furthermore, the incorporation of the volume data provided a more detailed understanding of market sentiment and trading activity, reinforcing the reliability of the buy and sell signals.

Despite the positive results, there are areas for improvement. For instance, refining the algorithm by incorporating additional technical indicators (e.g., RSI, Bollinger Bands) and implementing risk management strategies (such as stop-loss orders) could further enhance the performance and reduce the risk of drawdowns. Additionally, expanding the data set to include macroeconomic indicators or real-time news sentiment could provide more dynamic insights, helping the algorithm adapt to different market conditions.

In conclusion, the project successfully demonstrated the potential of using algorithms, data warehousing, and business intelligence tools for stock trading analysis. The results validate the effectiveness of the Moving Average Crossover strategy in identifying market trends and generating significant returns, positioning it as a valuable tool for traders and investors. Future work will focus on refining the algorithm, incorporating additional data sources, and improving risk management strategies to make it more robust and adaptable to changing market environments.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to everyone who contributed to the successful completion of this project.

First and foremost, we would like to thank our project supervisor for their continuous support, guidance, and insightful feedback throughout the project. Their expertise and encouragement were invaluable in shaping the direction of the project and helping us navigate the various challenges we faced.

We also extend our appreciation to the team members who worked tirelessly to bring this project to life. Each member contributed their skills and knowledge, from designing the data warehouse and implementing the trading algorithm to creating visualizations and refining the backtesting process. Their dedication and teamwork were critical to the success of the project.

Special thanks to the open-source community for providing resources like Yahoo Finance for stock data, PostgreSQL for database management, and Tableau for visualizing the results. These tools played a significant role in enabling the development of this trading system.

Finally, we would like to acknowledge our families and friends for their patience and support during the course of the project. Their understanding and encouragement helped us remain motivated and focused on completing the work.

Thank you all for your contributions to this project.

REFERENCES

[1] **Wikipedia contributors**. (2024). *Stock market*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Stock_market.

[2] **PostgreSQL Global Development Group**. (2024). *PostgreSQL Documentation*. https://www.postgresql.org/docs/.

[3] **Yahoo Finance**. (2024). *Yahoo Finance API*. https://www.yahoofinanceapi.com/.

[4] **Tableau Software**. (2024). *Tableau Documentation*. https://www.tableau.com/learn.M.