Name: Dilrabo Kodirova

<u>**Efficiency and Scalability of Trading Algorithms**</u>

INTRODUCTION

High-frequency trading has become very important in modern financial markets, with speed being sought and small price differences across securities being exploited. Yet, efficiency and scalability are important properties of such algorithms, which determine their appropriateness in practical applications. This project considers the computational side of trading algorithms, specifically with regard to time and space complexity, for an understanding of how they will perform under different conditions. In this paper, an empirical analysis of such algorithms is shown for different data volumes with insights into practical performance characteristics.

DATA

The data used in this project was from Yahoo Finance by use of the Python package, Yahoo Finance. It allows one to download historical price data for a wide range of stocks, therefore providing one with a good dataset with which to test trading algorithms. The date range is spread over some years, which is important to test the robustness of the algorithms.

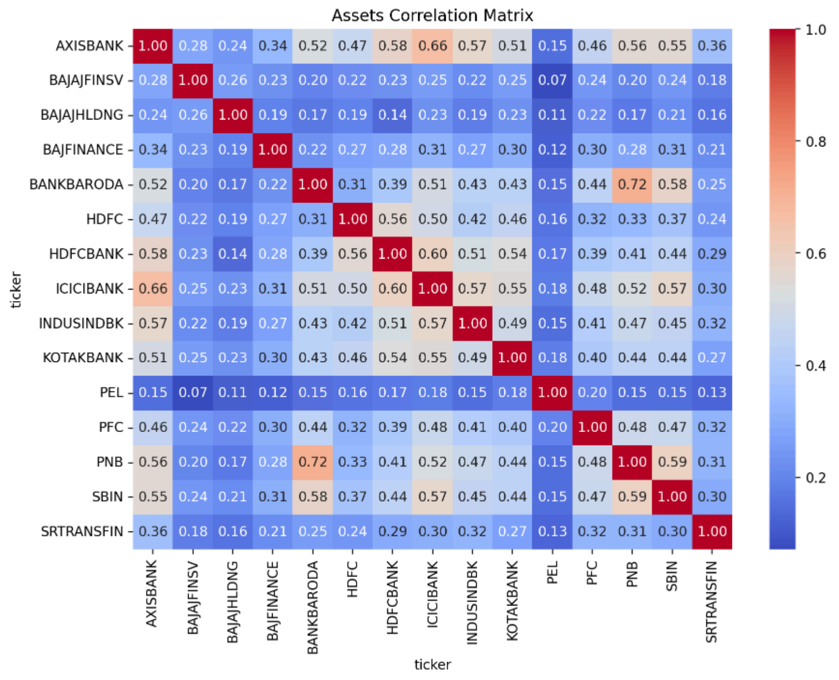
ALGORITHMS

Algorithm #1: Statistical Arbitrage and Pairs Trading with Machine Learning

Stat Arb is a trading strategy that looks to capitalize on the differences in the mean reversion of share prices or market microstructure anomalies. Pairs trading involves a market-neutral strategy for finding two stocks that are historically correlated. When the price relationship turns up, it is assumed that mean reversion is in order and one should buy into the underperforming stock and sell out of the overperforming stock. Then, prices will move back toward each other and the trade will have been profitable.

Implementation of Algorithm #1:

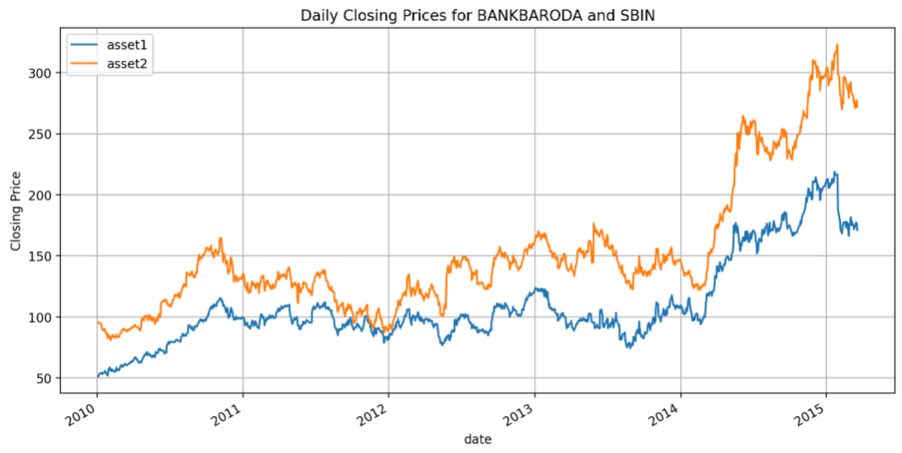
1) Stock Universe and Cointegrated Pair Identification
   The first step in pairs trading is to identify a universe of stocks that should be considered in a particular study or backtest. Next is to identify pairs that are most likely to be cointegrated. In this project, we'll use Pearson correlation to find highly correlated pairs from a preselected universe of stocks. Below is a representation of example set of stocks:



ticker	AXISBANK	BAJAJFINSV	BAJAJHLDNG	BAJFINANCE	BANKBARODA	HDFC	HDFCBANK	ICICIBANK	INDUSINDBK	KOTAKBANK	PEL	PFC	PNB	SBIN	SRTRANSFIN
AXISBANK	1.00	0.28	0.24	0.34	0.52	0.47	0.58	0.66	0.57	0.51	0.15	0.46	0.56	0.55	0.36
BAJAJFINSV	0.28	1.00	0.26	0.23	0.20	0.22	0.23	0.25	0.22	0.25	0.07	0.24	0.20	0.24	0.18
BAJAJHLDNG	0.24	0.26	1.00	0.19	0.17	0.19	0.14	0.23	0.19	0.23	0.11	0.22	0.17	0.21	0.16
BAJFINANCE	0.34	0.23	0.19	1.00	0.22	0.27	0.28	0.31	0.27	0.30	0.12	0.30	0.28	0.31	0.21
BANKBARODA	0.52	0.20	0.17	0.22	1.00	0.31	0.39	0.51	0.43	0.43	0.15	0.44	0.72	0.58	0.25
HDFC	0.47	0.22	0.19	0.27	0.31	1.00	0.56	0.50	0.42	0.46	0.16	0.32	0.33	0.37	0.24
HDFCBANK	0.58	0.23	0.14	0.28	0.39	0.56	1.00	0.60	0.51	0.54	0.17	0.39	0.41	0.44	0.29
ICICIBANK	0.66	0.25	0.23	0.31	0.51	0.50	0.60	1.00	0.57	0.55	0.18	0.48	0.52	0.57	0.30
INDUSINDBK	0.57	0.22	0.19	0.27	0.43	0.42	0.51	0.57	1.00	0.49	0.15	0.41	0.47	0.45	0.32
KOTAKBANK	0.51	0.25	0.23	0.30	0.43	0.46	0.54	0.55	0.49	1.00	0.18	0.40	0.44	0.44	0.27
PEL	0.15	0.07	0.11	0.12	0.15	0.16	0.17	0.18	0.15	0.18	1.00	0.20	0.15	0.15	0.13
PFC	0.46	0.24	0.22	0.30	0.44	0.32	0.39	0.48	0.41	0.40	0.20	1.00	0.48	0.47	0.32
PNB	0.56	0.20	0.17	0.28	0.72	0.33	0.41	0.52	0.47	0.44	0.15	0.48	1.00	0.59	0.31
SBIN	0.55	0.24	0.21	0.31	0.58	0.37	0.44	0.57	0.45	0.44	0.15	0.47	0.59	1.00	0.30
SRTRANSFIN	0.36	0.18	0.16	0.21	0.25	0.24	0.29	0.30	0.32	0.27	0.13	0.32	0.31	0.30	1.00

2) Perform Stationarity Test
Once a pair is identified, it is highly important to test for stationarity. Stationarity of spread between two stock prices is usually tested using the Augmented Dickey-Fuller (ADF) test. The major assumption in pairs trading is the fact that the spread is stationary and is expected to mean revert over time.

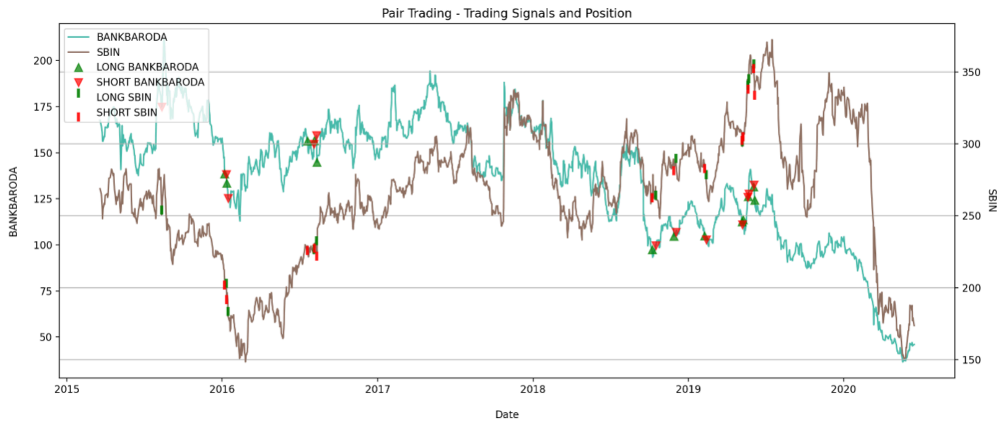
From the example set above, two stocks are selected as most correlated one, and chart below confirms the matching pattern in their performance



3) Generate Trading Signals using Z-Score
A z-score of the difference between two selected stocks is calculated to yield the trading signals. In fact, the z-score equals the number of standard deviations the spread is away

from its mean. A high z-score reflects that the difference between the two prices is wide; hence, a sell signal is generated in the overperformed stock and a buy signal in the underperformed stock.

We can apply z-scores for the above example pair of stocks to see what type of signals are there over a particular timeline:



Algorithm #2: Long Short-Term Memory (LSTM)
Long Short-Term Memory (LSTM) is one type of advanced RNN, which possesses long-term dependencies and is fit to treat sequential data. The power of LSTM lies in the power to learn and store information for long sequences, so it is proper for time series prediction. In financial markets, LSTM models yield high effectiveness at predicting the stock prices because they can capture complex temporal patterns and dependencies embedded in historical trading data.

Implementation of Algorithm #2:

1) Split the dataset into Xtrain and Ytrain:
   Separate the data into feature inputs that the model will use to make predictions (Xtrain) and the corresponding target outputs (Ytrain) that the model will predict.

2) Convert the 2D into 3D format:
   Transform the 2D data into a 3D array format, where each entry contains a sequence of time steps and features, to meet the input requirements of the LSTM model.

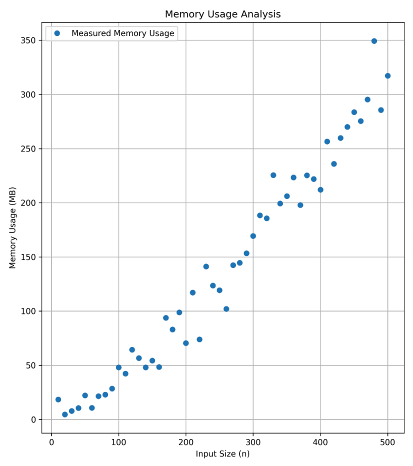
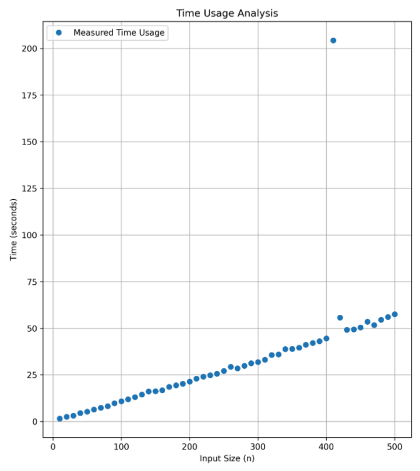
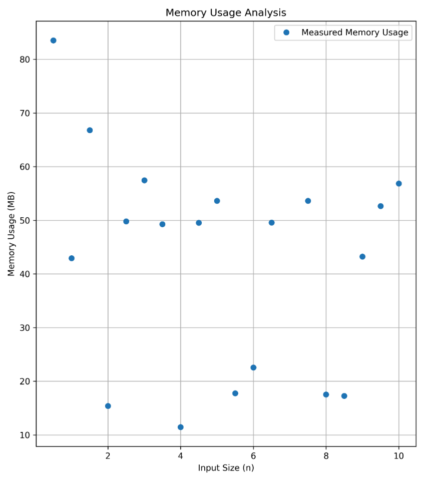
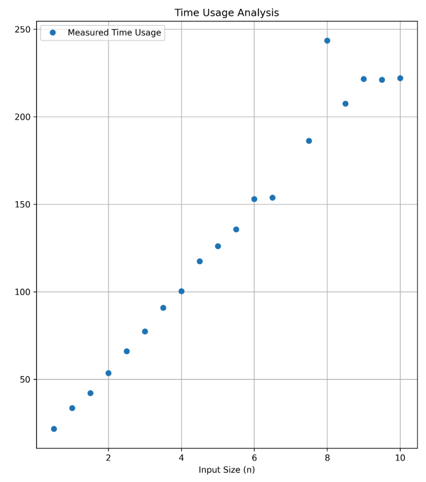
3) Build an LSTM model:
   Design a deep learning model with sequential layers, including two LSTM layers for capturing temporal patterns and two Dense layers for refining the output predictions, all constructed using Keras.

4) Configure the model:
   Compile the model to minimize errors using the Adam optimizer, and train it over multiple iterations (epochs) with a defined batch size to optimize its predictive performance.

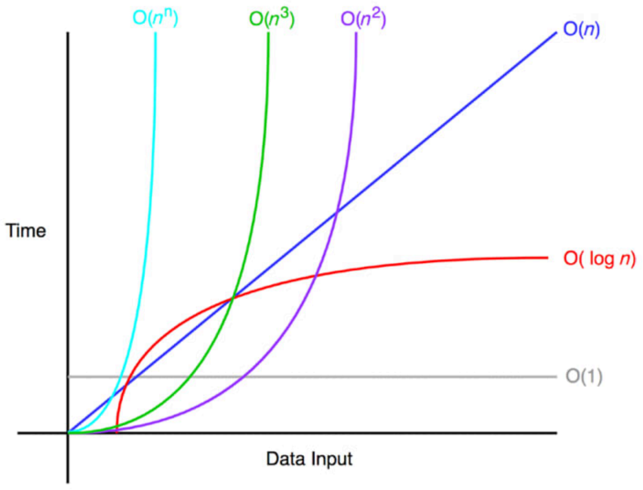
PERFORMANCE MEASURES
Since the practical performance of these trading algorithms is associated with time complexity and space complexity, they were taken as measures to evaluate. These metrics would provide insight into the scalability of the algorithms when applied to large datasets, i.e. under real-time trading conditions.

Time complexity is the total amount of computational time taken by an algorithm relative to input size. Space Complexity is defined to be the amount of memory an algorithm will require as a function of the input size.

If we plot the time and memory used for both of the algorithms under different size of dataset, we can observe the following trends:

Algorithm	**Space Complexity**	**Time Complexity**
Algorithm #1	 Space Complexity: O(n)	 Time Complexity: O(n)
Algorithm #2	 Space Complexity: N/A	 Time Complexity: O(n)

Discussion



The analysis of time and space complexity for these algorithms reveals insightful contrasts. For Algorithm #1, both the space and time complexity are linear, represented as O(n), indicating that the resource usage grows proportionally with the input size. While linear complexity is manageable, it becomes a constraint when dealing with large datasets, potentially leading to slower execution times. Thus, Algorithm #1 is more suitable for smaller datasets or scenarios where the stock universe has been pre-filtered based on criteria such as industry, competition, or financial health.

In contrast, Algorithm #2 presents an interesting case where space complexity is not easily identifiable, as the performance does not follow a predictable polynomial pattern. However, since the resource usage remains within a specific range regardless of dataset size, it suggests a constant space complexity, which would imply a more efficient use of memory. On the other hand, the time complexity of Algorithm #2 is also linear, similar to Algorithm #1, meaning that while it handles increasing dataset sizes well, it does not offer significant gains in speed over Algorithm #1. Nonetheless, the potential constant space complexity of Algorithm #2 positions it as a more efficient algorithm overall, especially when working with extensive datasets.

Conclusion

The inability to definitively identify the space complexity of Algorithm #2 makes it challenging to declare one algorithm superior to the other conclusively. However, the evidence suggests that Algorithm #2 may be more practical in real-world applications, particularly due to its potentially constant space complexity and comparable time complexity.

Code and corresponding documents used for implementation of algorithms and measurement of their performance is available on GitHub.

References:

GeeksforGeeks. “Time Complexity and Space Complexity.” *GeeksforGeeks*, 11 July 2021, www.geeksforgeeks.org/time-complexity-and-space-complexity/.

Grimste, Christy. “Equity Hedge Fund Strategies.” *Wall Street Oasis*, Wall Street Oasis, 23 Dec. 2022, www.wallstreetoasis.com/resources/skills/finance/hedge-fund-equity-strategies.

The AI Quant. “Statistical Arbitrage and Pairs Trading with Machine Learning.” *Medium*, Medium, 7 Apr. 2024, theaiquant.medium.com/statistical-arbitrage-and-pairs-trading-with-machine-learning-875a221c046c.

Thenuja Shanthacumaran. “Stock Price Prediction Using LSTM (Long Short-Term Memory).” *Medium*, Analytics Vidhya, 7 June 2020, medium.com/analytics-vidhya/stock-price-prediction-using-lstm-long-short-term-memory-e8c125a853e4.