Final Report

Building a Repository for Teaching Algorithmic Trading

SUPERVISOR: DR. RUIBANG LUO
WOO CHUNG YU, ANGEL (3035473816)

April 18, 2021

Abstract

Thanks to its capacity in managing large data flows and capturing fleeting anomalies in price trends, the use of algorithmic trading has become more prevalent in the financial market. Nonetheless, the development of learning resources for algorithmic trading has failed to keep up with the progression. Majority of existing resources predominantly focus on the US market, and they only cover either end of the tech-finance spectrum. In this project, we propose to build a code and data repository that puts together the relevant financial concepts and technical skills for learning algorithmic trading. We have built pipelines for collecting stock price, company financial statements, economic indicators, property price, social media and financial news data. In addition to the database, we have implemented indicators for microeconomic, macroeconomic and sentiment analysis, which could all be evaluated with the backtester featured in the repository. These code implementations include technical & fundamental analysis strategies that trace patterns and trends in order to generate trade signals, as well as machine learning models that predict market sentiment based on economic indicators and mass media data. We have also built integrated strategies that take inputs from all microeconomic, macroeconomic and sentiment features to generate trading signals. Code snippets that demonstrate how to use the Interactive Brokers API to carry out paper trading are also featured in the repository. The database and code base will contribute to teaching and research on algo trading at the university in long term. In response to the educational aim of the project, we have documented all code implementations in a website and created tutorials that enable beginners to learn algo trading regardless of their background knowledge in finance. The repository and the documentation website are respectively available at https://github.com/awoo424/algotrading and https://algo-trading.readthedocs.io/.
Acknowledgements

I would like to extend my thanks to our supervisor, Dr. Ruibang Luo, who provided the project team guidance and support throughout this project.

I would also like to express my gratitude to the Department of Computer Science and the teaching staff, for organising all the resources and logistics of the course.

Last but not least, I want to thank all of my friends and family who directly or indirectly helped me in this journey - no matter it was a proofread, or a small constructive feedback. Without any of you, the project would not have been possible.
# CONTENTS

List of Figures  
3

List of Tables  
5

1 Introduction  
7

2 Project Background  
7

2.1 Algo Trading Courses in Undergraduate Curriculum  
8

2.2 Algo Trading Online Self-learning Resources  
10

3 Objectives  
11

3.1 Educational Objectives  
11

3.2 Research Objectives  
11

4 Design Philosophy  
11

5 Project Overview  
12

6 Methodology  
14

6.1 Part 1(a) - Database  
14

6.1.1 Microeconomic Data  
14

6.1.2 Macroeconomic Data  
22

6.1.3 Sentiment Data  
24

6.2 Part 1(b) - Backtester  
25

6.2.1 Main Backtesting Function  
25

6.2.2 Plotting Signals  
26

6.2.3 Evaluation Metrics  
26

6.3 Part 2(a) - Trading Strategies  
29

6.3.1 Technical Analysis  
29

6.3.2 Fundamental Analysis  
32

6.3.3 Macroeconomic Analysis  
35
Table with fundamentals data to be scraped ........................................... 17
Extract of stock actions data csv file ..................................................... 19
Extract of company information data ...................................................... 20
Table to be scraped on aastock.com ....................................................... 24
Table to be scraped on finviz.com .......................................................... 25
An extract of the portfolio dataframe generated by the backtester function. 26
Example figure of the stock’s closing price with buy and sell signals plotted
by the backtester. .................................................................................. 26
An extract of the signals dataframe generated by the Moving Average
Crossover function. ................................................................................ 29
A code snippet of the moving average strategy running with the Julia kernel. 32
Code snippet for univariate analysis ....................................................... 35
An example of a scatter plot with a regression line. ............................... 36
Correlation heatmap of macroeconomic indicators and HSI .................. 36
Correlation heatmap of macroeconomic indicators and the average monthly
price per saleable area in Hong Kong .................................................... 37
Correlation heatmap of the top 7 features selected in the property trans-
action data ............................................................................................ 39
The graph of actual and predicted housing price for XGBoost ............... 40
An extract of the example merged dataframe ....................................... 45
Interface of the Interactive Brokers Trader Work Station ...................... 47
The homepage of the documentation website ........................................ 49
The bullet points of learning goals listed at the beginning of each page. 50
The highlight boxes that feature important definitions and tips in the
tutorial. ................................................................................................. 50
The legal disclaimer featured at the end of each page. ........................ 50
A page of documentation website featuring equation and code example .. 51
Line plot of portfolio value against time span .................................... 52
Line plot of no. of trading signals against time span ............................ 53
30 Statistical summary of portfolio performance when RSI is applied on all
tickers in HKEx .................................................. 53
31 Heat map showing how no. of trading signals varies with strategy ...... 54
32 Heat map showing how sharpe ratio varies with strategy .................. 54
33 Heat map showing how sharpe ratio varies with both the strategy and
time span .......................................................... 55
34 Heat map showing how average no. of trading signals varies with strategy
for HK tickers (2017-2019) ............................................. 56
35 Heat map showing how average no. of trading signals varies with strategy
for HK tickers (2019-2021) ............................................. 56
36 Heat map showing how average portfolio return varies with strategy for
HK tickers (2017-2019) ................................................. 57
37 Heat map showing how average portfolio return varies with strategy for
HK tickers (2019-2021) ................................................. 57
38 Heat map showing how average sharpe ratio varies with strategy for HK
tickers (2017-2019) .................................................... 58
39 Heat map showing how average sharpe ratio varies with strategy for HK
tickers (2019-2021) .................................................... 58

**LIST OF TABLES**

1 Extract of course list for BASc(FinTech) degree at HKU .................. 9
2 Algo trading-related courses offered at other tertiary institutions in Hong
   Kong. ................................................................. 10
3 An overview fact sheet of the database in the repository. ................. 13
4 Fact sheet of ticker symbol lists collected in the database. ............... 14
5 Fact sheet of stock tick data collected in the database. ................... 15
6 Fact sheet of fundamentals data collected in the database. ................ 17
7 Fact sheet of stock actions data collected in the database. ............... 19
8 Fact sheet of company info collected in the database. ..................... 19
9 Fact sheet of property transaction data collected in the database. . . . . . 22
10 The file and format of each economic indicator collected. . . . . . . 23
11 Fact sheet of property transaction data collected in the database. . . . . . 24
12 List of technical indicators implemented in the repository. . . . . . . . . 30
13 List of common financial ratios used for fundamental analysis. . . . . . . 33
14 Accuracy scores of different machine learning models in predicting 1-year 
   and 3-year bankruptcy of a listed company. . . . . . . . . . . . . . . . . . . . . . . 34
15 RMSLE values of the housing price prediction models. . . . . . . . . . . . 40
16 The list of selected tickers for testing the baseline model and LSTM models. 60
17 Performance results of the integrated strategy models. . . . . . . . . . . 61
1. Introduction

Algorithmic trading\(^1\), is commonly defined as the use of computer programs to automatically make trading decisions, submit orders, and oversee these orders after submission (Hendershott et al., 2009). Algo trading has empowered hedge funds and a lot of financial institutions to “beat the market” over the recent decade. This data-driven approach of trading has been on the rise thanks to the introduction of electronic limit order books (Jain, 2005) and trading systems that are able to accommodate fast, dynamic information flow.

Dating back to the 1980s, people have begun to appreciate the power of algorithms in trading due to an experiment introduced by Richard Dennis and William Eckhardt, who were later characterised as “the Market Legends”. In the past, financial researchers primarily focused on the quest for sophisticated risk management techniques and superior investment opportunities (Kissell, 2013). Nevertheless, much of this initiative has shifted towards the efficient implementation of strategies upon the success of the Turtle Trading Experiment (Covel et al., 2007). In this famous experiment, Richard and William managed to train a group of novice investors to earn more than US$175 million in merely five years by teaching them how to implement a trend-following strategy\(^2\).

This story has demonstrated the power of algorithms in enabling non-traders to make great profits. Similar to the Turtle Trading Strategy, many other strategies could also be described algorithmically. A typical algorithmic trading strategy consists of the following components (in order):

- Select the ticker symbol(s) to trade
- Retrieve relevant data (e.g. daily price tick) for these companies
- Define the entry rules
- Define the exit rules
- Carry out the trades
- Evaluate the strategy performance

In essence, the key to master algo trading is to learn **how to abstract trading logic into algorithmic steps**. Under the hood, it is about translating financial knowledge into code implementations.

2. Project Background

Having understood the motivation of learning algo trading, we further investigated whether existing resources (including courses at universities and online materials) are sufficient and accessible for beginners to acquire the skill. We define a *reasonably good*

---

\(^1\) Also referred to as “algo trading” in the literature

\(^2\) To date, the exact parameters of the Turtle Trading strategy are still kept secret.
learning resource as one that could teach algorithmic trading from both finance and programming perspectives.

2.1 Algo Trading Courses in Undergraduate Curriculum

Courses offered at HKU Among the elective courses offered by HKU Department of Computer Science, there has been a 5% growth in the proportion of FinTech-related courses between the year of 2017 and 2020 (as shown in Figures 1 and 2). Currently, about 11% of the elective courses are related to FinTech. This indicates that there is an increasing demand for FinTech courses, and we anticipate that the demand would continue to surge in the coming years. However, note that all of the existing FinTech courses (within these 11%) are either about e-commerce or blockchain technology, and none of them are related to algo trading.

![Figure 1: Category of elective courses offered by HKU Department of Computer Science in the academic year 2017-2018 (Link to full course list).](image1)

![Figure 2: Category of elective courses offered by HKU Department of Computer Science in the academic year 2020-2021 (Link to full course list).](image2)

We also examined the new undergraduate programme - Bachelor of Arts and Sciences in Financial Technology (hereinafter referred to as “BASc(FinTech)”) that was introduced
at HKU in the academic year 2019-2020. It is a programme designed to cover courses in mathematics, statistics, finance, computing and law. Although some of the courses included in the programme (as shown in Table 1) are relevant to algo trading (e.g. “Investments and portfolio analysis” and “Mathematical finance”), none of them has taken algo trading as the principal focus. In addition, most of these courses are offered by the Faculty of Business and Economics, and thus their syllabi have been designed to place more emphasis on teaching the financial concept of trading strategies rather than their implementation in code.

<table>
<thead>
<tr>
<th>Core Courses (54 credits)</th>
<th>Elective Courses (30 credits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Computer programming I</td>
<td>• Machine learning / Artificial intelligence and deep learning</td>
</tr>
<tr>
<td>• Introduction to financial technologies</td>
<td>• Modern technologies on World Wide Web</td>
</tr>
<tr>
<td>• Foundations of FinTech Programming</td>
<td>• Applied deep learning</td>
</tr>
<tr>
<td>• Distributed ledger and blockchain</td>
<td>• Cyber security</td>
</tr>
<tr>
<td>• Linear algebra, probability and statistics</td>
<td>• Artificial intelligence applications</td>
</tr>
<tr>
<td>• Introduction to financial accounting</td>
<td>• Investments and portfolio analysis</td>
</tr>
<tr>
<td>• Introductory microeconomics</td>
<td>• Derivatives</td>
</tr>
<tr>
<td>• Corporate finance</td>
<td>• Mathematical finance</td>
</tr>
<tr>
<td>• Regulation of financial markets</td>
<td>• Regulatory and operational issues in finance</td>
</tr>
<tr>
<td>• Text analytics and natural language processing in finance and fintech</td>
<td></td>
</tr>
<tr>
<td>• Big data and data mining</td>
<td></td>
</tr>
<tr>
<td>• E-payment and crypto-currency</td>
<td></td>
</tr>
<tr>
<td>• Introduction of English linguistics</td>
<td></td>
</tr>
<tr>
<td>• English corpus linguistics</td>
<td></td>
</tr>
<tr>
<td>• Introduction to quantitative methods in psychology</td>
<td></td>
</tr>
<tr>
<td>• Cognitive psychology</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Extract of course list for BASc(FinTech) degree at HKU (Link to full course list).

Courses offered at other tertiary institutions  Additionally, we have looked into the course syllabi of undergraduate programs in tertiary institutions other than HKU. As shown in Table 2, The Chinese University of Hong Kong (CUHK) has incorporated algo trading as part of the Bachelor in Quantitative Finance curriculum. On the other hand, the Hong Kong University of Science and Technology (HKUST) has a related elective course in the Master of Finance programme.

To summarise, it is generally rare to find an undergraduate course that focuses on algo trading in Hong Kong. Despite the increase in the number of FinTech-related undergraduate courses at HKU, the university is still lacking an algo trading course that is offered for undergraduates.
2.2 Algo Trading Online Self-learning Resources

Moving on, we have studied available resources on the Internet that allow people to self-learn algo trading, including blog posts, online courses and websites like Investopedia. According to our research, we found that a vast majority of them suffer from the following problems:

- They are scattered, and they usually cover only a few concepts or topics about algorithmic trading.
- They have a single-sided focus, which means that they focus on either finance or programming, and assume that the user is familiar with the other.
- They do not fit the local context (e.g. trade execution in Hong Kong) and are merely focusing on the US market.

As a result, for beginners who opt for relying on online materials, the learning journey has inevitably become a scavenger hunt that requires the learner to go back and forth across various websites and blog posts. I personally find this process inefficient and daunting, as it induces the learner to spend a plenty of time in searching for the right information rather than digesting the concepts.

Based on the above findings, we come to the conclusion that available resources (inclusive of university courses and online materials) are not accessible and not comprehensive to a large extent. Thus, being a group of students who are majoring in Computer Science and Finance at the university, we advocate the proposal of developing a new algorithmic trading course for Computer Science undergraduates at HKU. We believe that the proposed course aligns with the curriculum of the BASc(FinTech) degree, and could render HKU to take the lead in nurturing FinTech talents in the industry. To realise such a proposal, this project aims to create learning resources that could be used in the course and help amateur investors to go from zero to hero in algo trading. Thus, this project kickstarts.
3. **OBJECTIVES**

This project aims to achieve several educational and research objectives in support of developing a course for teaching algorithmic trading at the university.

3.1 **Educational Objectives**

The primary educational objective of this project is to build a code and data repository that puts together the financial concepts, data science skills and algorithm design techniques that are crucial to learning algorithmic trading. Ideally, all these contents would be presented in a clear, reader-friendly manner. The repository aims to appeal to a wide audience and especially to students who have no knowledge in finance, but are intrigued to have a better understanding of how financial concepts are translated into computer code.

The following are the anticipated outcomes of the proposed course:

- To learn algorithmic trading under the financial setting of Hong Kong;
- To gain practical experience in coding trading strategies and making trading orders;
- To acquire and apply both finance and computer science knowledge in a multidisciplinary field.

3.2 **Research Objectives**

The research objective of this project is to establish a consolidated database at the university and an experimentation pipeline for testing trading strategies in the Hong Kong and United States (US) stock markets. The database consists of historical price ticks, property prices data and social media data that would be useful for future research and teaching purposes. With the pipeline as a foundation, the performance of more advanced models and strategies can also be analysed and evaluated.

4. **DESIGN PHILOSOPHY**

To assure that the objectives could be achieved, we have designed the repository to align with the following principles.

1. **Comprehensiveness** All financial concepts and mathematical formulae come along with code examples in the repository. The repository covers each component within the typical pipeline of an algorithmic trading system, starting with data collection, exploratory data analysis, to backtesting and post-trade analysis. It aims to offer students a full, complete picture of the algo trading discipline.

2. **Clarity** All concepts, terminologies and mathematical formulae in the repository are supplemented with explanation. All code examples also come along with in-line comments that explain what they are doing. Everything in the repository is designed to be easy to follow and reader-friendly. All naming conventions, file structure and coding style are consistent throughout different parts of the repository.
3. **Maintainability**  All of the specifications and usage instructions are well-documented so that the repository and database can be easily maintained in the long term. Pipelines for updating the database continuously are also created so that the downloading of data could be done efficiently.

5. **PROJECT OVERVIEW**

The repository could be divided into two parts, as shown in Figure 3.

![Flow chart showing the overall structure of the repository.](image)

**Part 1**  This part focuses on the development of the database and backtester, which serves as the backbone of the repository. The database consists of data that examine the market from three different perspectives, namely microeconomic, macroeconomic and sentimental. The backtester offers a module for testing and evaluating trading strategies with a selected date range of historical data.

**Part 2**  This part focuses on the implementation of trading strategies that make use of the data collected in the former part. The repository features conventional strategies (including technical analysis and fundamental analysis), as well as more advanced strategies (also called "integrated strategies") that generate trading signals based on more input features (e.g. stock price, macroeconomic data and sentiment labels). Additionally, the repository covers the topic of *Paper Trading*\(^3\) that allows students to learn how to execute trading orders via calling the Interactive Brokers API.

---

\(^3\)Simulated trade that allows an investor to practise buying and selling without risking real money.
Deliverables  We have the following deliverables available by now, and Table 3 provides a summary of the database in the repository:

Database (including US, Hong Kong, Japan, Shanghai and Shenzhen stock markets)

- List of stock ticker symbols, csv format
- Historical stock tick (1-minute, 1-day), csv format
- Fundamentals data, csv format
- Stock actions data (dividend, stock split), csv format
- Company info (name, sector, no. of employees, description), json format
- Tweets and news headlines for each ticker (US and HK only), csv format
- Property transaction data (HK only), csv format
- Economic indicator data (HK only), csv format

Code implementation (GitHub)

- Code used for collecting and updating the database (Python)
- Technical analysis strategies (Python & Julia)
- Fundamental analysis strategies (Python)
- Macroeconomic data analysis (Python)
- Property price prediction with machine learning (Python)
- Sentiment analysis with machine learning (Python)
- Integrated strategies (Python)
- Paper Trading with Interactive Brokers API (Python)

Documentation website (Link)

- Tutorials & code walkthrough
- Explanation of financial concepts and terminologies
- Mathematical formulae & code example for each strategy

<table>
<thead>
<tr>
<th>Microeconomic data</th>
<th>Number of rows</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ticker symbol list</td>
<td>28,003</td>
<td>1441 KB</td>
</tr>
<tr>
<td>1-day stock tick</td>
<td>48,597,879</td>
<td>4.36 GB</td>
</tr>
<tr>
<td>1-min stock tick</td>
<td>152,452,357</td>
<td>13.11 GB</td>
</tr>
<tr>
<td>Financial statements</td>
<td>254,176</td>
<td>131.1 MB</td>
</tr>
<tr>
<td>Stock actions</td>
<td>288,021</td>
<td>5.87 MB</td>
</tr>
<tr>
<td>Company info</td>
<td>4602</td>
<td>18.66 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Macroeconomic data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Centaline Property</td>
<td>251,660</td>
<td>21.7 MB</td>
</tr>
<tr>
<td>Midland Realty</td>
<td>166,481</td>
<td>26.2 MB</td>
</tr>
<tr>
<td>Economic indicator</td>
<td>212</td>
<td>25 KB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sentiment data</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HK News (aastock)</td>
<td>4864</td>
<td>4.7 MB</td>
</tr>
<tr>
<td>US News (finviz)</td>
<td>741,006</td>
<td>81.1 MB</td>
</tr>
<tr>
<td>Tweets</td>
<td>1,989,362</td>
<td>240.1 MB</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>~18 GB</strong></td>
</tr>
</tbody>
</table>

Table 3: An overview fact sheet of the database in the repository.
6. Methodology

In the following, we will elaborate on the methodology for each part.

6.1 Part 1(a) - Database

6.1.1 Microeconomic Data

Microeconomic data include ticker symbol lists, stock prices data, fundamentals data (i.e. financial statements of each ticker symbol), stock actions data and company information.

**Ticker Symbol List** The complete list of ticker symbols in csv format (example shown in Figure 4) was manually downloaded from each stock exchange’s official websites respectively. The fact sheet in Table 4 summarises the total size and average number of rows of the collected ticker symbols lists.

![Figure 4: The first few rows in the csv file storing the complete list of ticker symbols that could be traded on Hong Kong Stock Exchange (HKEx).](image)

<table>
<thead>
<tr>
<th>symbol</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Cheung Kong (Holdings) Ltd</td>
</tr>
<tr>
<td>1</td>
<td>CLP Holdings Ltd</td>
</tr>
<tr>
<td>2</td>
<td>Hong Kong and China Gas Co. Ltd</td>
</tr>
<tr>
<td>3</td>
<td>Wharf (Holdings) Ltd</td>
</tr>
<tr>
<td>4</td>
<td>HSBC Holdings plc</td>
</tr>
</tbody>
</table>

Table 4: Fact sheet of ticker symbol lists collected in the database.

**Stock Tick Data** The fact sheet in Table 5 summarises the total size and average number of rows of the collected stock tick data for the six selected stock markets. We have collected data for a total of 14,253 tickers. The total number of rows for 1-minute tick data is 152,452,357, that for 1-day tick data is 48,597,879. They are all of csv format.

To collect the historical price data (i.e. stock tick) for each listed company, we loaded the list of ticker symbols into the Jupyter Notebook (denoted as `ticker_list`) as an array. After that, we could traverse the list in order to download the data for each ticker by
<table>
<thead>
<tr>
<th></th>
<th>1-min data</th>
<th>1-day data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of tickers</td>
<td>Avg number of rows</td>
</tr>
<tr>
<td>HKEx</td>
<td>1481</td>
<td>9176</td>
</tr>
<tr>
<td>JP</td>
<td>3574</td>
<td>5796</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>3598</td>
<td>9294</td>
</tr>
<tr>
<td>NYSE</td>
<td>2498</td>
<td>16638</td>
</tr>
<tr>
<td>SH</td>
<td>1823</td>
<td>10934</td>
</tr>
<tr>
<td>SZ</td>
<td>2579</td>
<td>9001</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>14253</strong></td>
<td><strong>152,452,357</strong></td>
</tr>
</tbody>
</table>

Table 5: Fact sheet of stock tick data collected in the database. *HKEx* stands for Hong Kong Stock Exchange, *JP* stands for stock exchanges in Japan, *SH* stands for Shanghai and *SZ* stands for Shenzhen.

calling the Yahoo! Finance API. The outline of the algorithm for downloading the stock tick data is shown in Algorithm 1.

**Algorithm 1: Download stock tick data**

**Data:** List of stock ticker symbols

**Result:** csv files with columns (Date, Open, High, Low, Close, Volume)

1. Import *pandas* and *yfinance*;
2. Load list of symbols into an array $S[1...n]$;
3. for symbol in $S[1...n]$ do
   4. data $\leftarrow$ Call the Yahoo! finance API;
   5. $df \leftarrow$ pandas.DataFrame(data);
   6. Export $df$ to a csv file;
4. end

An example output csv file of the 1-day tick data for a ticker symbol is shown in Figure 5. The *Date* is in YYYY-MM-DD format, the *Open*, *High*, *Low* and *Close* columns store floating point numbers, and the *Volume* column stores an integer. The first column would be replaced with *Datetime* if it is 1-minute tick data.

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>33.037451</td>
<td>33.367817</td>
<td>32.376717</td>
<td>32.376717</td>
<td>3194413</td>
</tr>
<tr>
<td>1</td>
<td>30.890019</td>
<td>31.385591</td>
<td>29.981523</td>
<td>30.146683</td>
<td>6058531</td>
</tr>
<tr>
<td>2</td>
<td>30.394473</td>
<td>30.559633</td>
<td>28.081818</td>
<td>28.659994</td>
<td>10440480</td>
</tr>
<tr>
<td>3</td>
<td>29.072963</td>
<td>29.403330</td>
<td>28.577392</td>
<td>29.238123</td>
<td>6049796</td>
</tr>
<tr>
<td>4</td>
<td>30.229264</td>
<td>30.724838</td>
<td>29.485929</td>
<td>29.485929</td>
<td>5195405</td>
</tr>
</tbody>
</table>

Figure 5: An extract of *hkex_0001.csv* which is the output file storing the 1-day tick of the stock "0001" listed in HKEx.

The 1-day price data for each ticker was downloaded since the initial public offering (IPO) of the company, and its 1-minute data was downloaded starting from 1 June 2020.
Build a Repository for Teaching Algorithmic Trading

Final Report

til the end of the project period. As both the 1-day and 1-minute stock tick data need to be continuously updated till the end of the project period, additional code has also been written so as to automate the process of downloading the data between the last updated date and the latest trading day. The pseudocode of the program is shown in Algorithm 2.

**Algorithm 2: Update stock tick data (in-place)**

<table>
<thead>
<tr>
<th>Data: Directory containing stock tick files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Result:</strong> Stock tick files with records til latest trading day</td>
</tr>
</tbody>
</table>

1. Import modules;
2. `dir_name ←` name of directory storing the stock tick files;
3. `pathlist ← Path(dir_name).rglob('*.csv');`
4. **for** `path in pathlist` **do**
5. `df ← Read file as csv;`
6. `last_date ← Get last date in df;`
7. `today ← Get today’s date;`
8. `dates ← list of dates between last_date and today;`
9. **if** `last_date != today` **then**
   10. `data ← Call Yahoo! Finance API with start=`dates[0], end=`dates[−1];`
   11. **if** `last_date == last date in data` **then**
       12. `# Today is not a trading day`
       13. `print(Data already up-to-date, no changes made);`
       14. **else**
       15. `| Append data to df;`
       16. **end**
   17. **else**
   18. `print(Data already up-to-date, no changes made);`
   19. **end**
20. **end**
21. `print success message;`

To update the stock tick data for a particular stock exchange, the following command could be run in the terminal (which would trigger the update program):

```
make update-<place/stock exchange abbreviation>-<day or min>
```

For example, if I would like to update the 1-minute price data for tickers listed in the Hong Kong Stock Exchange (HKEx), the required command would be `make update-hk-min`. Upon completion of the download, a success message would be displayed.

**Fundamentals Data**

All three types of financial statements (i.e. fundamentals data) including (1) Income Statement, (2) Balance Sheet and (3) Statement of Cash Flow were collected for each ticker symbol. The fact sheet in Table 6 summarises the total size and average number of rows of the collected fundamentals data.

As there does not exist any open-source API that enables direct crawling of the funda-
### Fundamentals data

<table>
<thead>
<tr>
<th></th>
<th>Number of tickers</th>
<th>Year range</th>
<th>Time period</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEx</td>
<td>1458</td>
<td>2016 to 2019</td>
<td>Annual</td>
<td>19.9 MB</td>
</tr>
<tr>
<td>JP</td>
<td>3733</td>
<td>2017 to 2020</td>
<td>Annual</td>
<td>54 MB</td>
</tr>
<tr>
<td>US</td>
<td>2088</td>
<td>2009 to 2019</td>
<td>Annual, quarterly</td>
<td>57.2 MB</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>7279</strong></td>
<td><strong>N/A</strong></td>
<td><strong>N/A</strong></td>
<td><strong>131.1 MB</strong></td>
</tr>
</tbody>
</table>

Table 6: Fact sheet of fundamentals data collected in the database. Note that US includes both NASDAQ and NYSE.

To collect fundamentals data, the `lxml` library was used to parse the HTML document and manually scrape the table from the page displayed on Yahoo! Finance (as shown in Figure 6). The outline of the algorithm for scrapping fundamentals data is shown in Algorithm 3.

![Figure 6: The table storing fundamentals data for a stock on the Yahoo! Finance page that is to be scraped using Python.](image)

Below shows the data featured in each type of financial statement respectively.
A ticker’s **income statement** reports the company’s income and expenditures. It contains the following columns:

- Date, Total Revenue,
- Income from Associates & Other Participating Interests,
- Special Income Charges, Other Non Operating Income Expenses,
- Pretax Income, Tax Provision, Net Income Common Stockholders,
- Diluted NI Available to Com Stockholders, Basic EPS, Diluted EPS,
- Basic Average Shares, Diluted Average Shares,
- Net Income from Continuing & Discontinued Operation,
- Normalized Income, Reconciled Depreciation,
- Net Income from Continuing Operation Net Minority Interest,
- Total Unusual Items Excluding Goodwill, Total Unusual Items,
- Tax Rate for Calcs, Tax Effect of Unusual Items,

A ticker’s **balance sheet** reports a company’s assets, liabilities and shareholders’ equity. It contains the following columns:

- Date, Total Assets, Total Liabilities Net Minority Interest,
- Total Equity Gross Minority Interest, Total Capitalization,
- Common Stock Equity, Net Tangible Assets, Invested Capital,
- Tangible Book Value, Total Debt, Net Debt, Share Issued,
- Ordinary Shares Number, Preferred Shares Number,
- Treasury Shares Number

A ticker’s **statement of cash flow** reports the amount of cash and cash equivalents held by a company. It contains the following columns:

- Date, Operating Cash Flow, Investing Cash Flow,
- Financing Cash Flow, End Cash Position, Capital Expenditure,
- Issuance of Capital Stock, Issuance of Debt, Repayment of Debt,
- Repurchase of Capital Stock, Free Cash Flow

**Stock Actions Data**  Stock actions data features information about distribution of dividends and stock splits for each ticker symbol. These data were collected via the same algorithm for downloading stock tick data (Algorithm 1), except that we call the Yahoo! Finance API with the parameter of `actions`. An example output file of the actions data is shown in Figure 7. The fact sheet in Table 7 summarises the total size and average number of rows of the collected stock actions data.
Figure 7: An extract of hkes_0005.csv which is the output file storing the actions data of the stock "0005" listed in HKEx.

<table>
<thead>
<tr>
<th>Date</th>
<th>Dividends</th>
<th>Stock Splits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0000-03-13</td>
<td>1.612</td>
<td>0.0</td>
</tr>
<tr>
<td>1 0000-08-14</td>
<td>0.150</td>
<td>0.0</td>
</tr>
<tr>
<td>2 0001-03-14</td>
<td>0.285</td>
<td>0.0</td>
</tr>
<tr>
<td>3 0000-08-22</td>
<td>0.190</td>
<td>0.0</td>
</tr>
<tr>
<td>4 0002-03-20</td>
<td>0.290</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 7: Fact sheet of stock actions data collected in the database.

**Company Info** The basic information of each listed company was collected via the same algorithm for downloading stock tick data (Algorithm 1), except calling the Yahoo! Finance API with the parameter of info. The information collected are in json format and they include the sector it belongs to, number of full-time employees, age of the company etc. An example of the company info json data is shown in Figure 8. The fact sheet in Table 8 summarises the total size and average number of rows of the collected stock actions data. (Note that not all ticker symbols are provided with basic info in Yahoo! Finance.)

<table>
<thead>
<tr>
<th>Company info data</th>
<th>Number of symbols with info</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKEx</td>
<td>169</td>
<td>721 KB</td>
</tr>
<tr>
<td>JP</td>
<td>6</td>
<td>22.2 KB</td>
</tr>
<tr>
<td>NASDAQ</td>
<td>2544</td>
<td>10.1 MB</td>
</tr>
<tr>
<td>NYSE</td>
<td>1883</td>
<td>7.82 MB</td>
</tr>
<tr>
<td>Total</td>
<td>4602</td>
<td>18.66 MB</td>
</tr>
</tbody>
</table>

Table 8: Fact sheet of company info collected in the database.
object {'sector': 'Communication Services', 'fullTimeEmployees': 24700, 'longBusinessSummary': 'PCCH Limited provides telecommunications and related services in Hong Kong, Mainland and other parts of China, Japan, and internationally. The company’s services include local telephony, local data and broadband, mobile and international telecommunications, and satellite-based and network-based telecommunications services; and outsourcing, consulting, and contact center services. It also provides technical support, electronics and communications engineering, and products and solutions, as well as free television, pay television program, and interactive multimedia services; sells advertising in various telephone directories and on the Internet; publishes directories; and sells mobile handsets and accessories. In addition, the company offers broadcasting and related services, management and engineering support services, customer relationship management and customer contact management solutions; content for various media, and outsourced call center and data center services; and over-the-top video services under the Viu brand, as well as sells customer-premises equipment and related solutions. Further, it engages in the sale, distribution, and marketing of telecommunication products; supply of broadband internet access solutions and web services; provision of data services; and software development, systems integration, consulting, and informatization activities; the provision of computer and IT related value-added services to business customers; property investment, development, management, and leasing, as well as hotel management activities; and ski operations. Additionally, the company offers entertainment over-the-top platforms; and digital, IT and business process outsourcing, cloud computing, hosting, managed, e-commerce, and IoT solutions. PCCH Limited was founded in 1925 and is headquartered in Quarry Bay, Hong Kong.

city: Quarry Bay
phone: 852 2888 2888
country: Hong Kong
companyOfficers: []
website: http://www.pcch.com
maxAge: 1
address1: PCCH Tower
fax: 852 2877 8877
industry: Telecom Services
address2: 42st Floor Taikoo Place 979 King's Road
previousClose: 4.64
regularMarketOpen: 4.45
twoHundredDayAverage: 4.5687323
trailingAnnualDividendYield: 0.07252252
payOutRatio: 3.5726081
volume24hr: null

Figure 8: An extract of hkek_0008.json which is the output file storing the company info data of the stock "0008" listed in HKEx.
Real-time Price  Although real-time prices are not collected as part of this repository’s database, it is useful in the practical setting and in more specialised fields such as intraday trading and high-frequency trading. Therefore, a code example is included in the repository in order to demonstrate how to scrape real-time prices for a particular stock ticker symbol using the Beautiful Soup library in Python.

Given the url to the Yahoo! Finance page for a particular stock ticker, a GET request could be made to obtain the HTML document. Then, Beautiful Soup could be used to parse the document and fetch the element that contains the real-time price:

```python
session = requests_html.HTMLSession()
r = session.get(url)  # url = the Yahoo! Finance page with the price
soup = BeautifulSoup(r.content, 'lxml')
try:
    price = soup.select('table td')[5].text.split(' ')[0]
    price = float(price)
except IndexError as e:
    price = None
```

The above code could further be wrapped in a for loop in order to obtain the real-time prices for a list of tickers. Note that the code might not work if Yahoo! Finance changes its layout, and thus will lead to changes in the HTML document structure (which is a disadvantage of scraping data with Beautiful Soup).
6.1.2 Macroeconomic Data

(The implementation of this part credits to Lee Kwanyoung.)

Macroeconomic data includes property transaction data and economic indicators data (both in Hong Kong only).

**Property Transaction Data**  Web scrapers were built with Beautiful Soup to collect transaction records of properties in Hong Kong from two real estate websites, which are Centaline Property and Midland Realty. These records are stored in different directories by districts, and each area has its own csv file (e.g. `kowloon/Lai_chi_kok.csv`). The fact sheet in Table 9 summarises the total size and average number of rows of the collected fundamentals data.

<table>
<thead>
<tr>
<th></th>
<th>Total number of rows</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centaline Property</td>
<td>251,660</td>
<td>21.7 MB</td>
</tr>
<tr>
<td>Midland Realty</td>
<td>166,481</td>
<td>26.2 MB</td>
</tr>
</tbody>
</table>

Table 9: Fact sheet of property transaction data collected in the database.

Concerning **Centaline Property**, the transaction data are categorised into Hong Kong Island, Kowloon, New Territories East and New Territories West districts. Records for each district could be obtained by passing the parameters of district id and registration period. The data spans over a date range of 2 Jan 2017 to 23 Dec 2020, and consists of 10 basic features describing the apartment involved in the transaction. The columns of the dataframe are as follows (in order):

- address, buildingAge, regDate, price, saleableArea, grossArea, upSaleableArea, upGrossArea, lasthold, gain

Concerning **Midland Realty**, the transaction data are categorised into Hong Kong Island, Kowloon, the New Territories districts. The data obtained spans over a date range of 2 Jan 2018 to 23 Dec 2020, with 20 features describing the apartment. Compared to the data collected from Centaline property, the address is further divided into separate columns and there are also additional features such as `floorL` (floor level) and `bedroom` (number of bedrooms) about the apartment. The columns of the dataframe are as follows (in order):

- region, subregion, district, estate, building, firstOpDate, floorL, bedroom, sittingroom, floor, flat, grossArea, saleableArea, price, regDate, lastRegDate, lastPrice, gain, lat, lon

Pre-processing and feature engineering has also been done before storing the transaction records data into the database. Additional features for instance as `buildingAge` was calculated using the first operating date of the building. Redundant features and columns with too many null values were dropped. In addition, missing values in each column were replaced with each column’s mean.
Economic Indicators  Economic indicators data were collected from the website of the Census and Statistic Department website. The list of indicators is shown in Table 10. All of them were manually downloaded from the website, and are stored in csv format. The data consist of indicators between the year of 2017 and 2021.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Indicator</th>
<th>Unit</th>
<th>Time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE001.csv</td>
<td>Population</td>
<td>thousands</td>
<td>Half-yearly</td>
</tr>
<tr>
<td>TABLE006.csv</td>
<td>Unemployment rate (seasonally adjusted)</td>
<td>%</td>
<td>Monthly</td>
</tr>
<tr>
<td></td>
<td>Unemployment rate (not seasonally adjusted)</td>
<td>%</td>
<td>Monthly</td>
</tr>
<tr>
<td>TABLE055.csv</td>
<td>Total export</td>
<td>HK$ million</td>
<td>Monthly</td>
</tr>
<tr>
<td></td>
<td>Total import</td>
<td>HK$ million</td>
<td>Monthly</td>
</tr>
<tr>
<td>TABLE030.csv</td>
<td>GDP</td>
<td>HK$ million</td>
<td>Quarterly</td>
</tr>
<tr>
<td></td>
<td>GDP per capita</td>
<td>HK$</td>
<td>Yearly</td>
</tr>
<tr>
<td>TABLE052.csv</td>
<td>Composite Consumer Price Index</td>
<td>-</td>
<td>Monthly</td>
</tr>
</tbody>
</table>

Table 10: The file and format of each economic indicator collected.
6.1.3 Sentiment Data

(The implementation of this part credits to Wu Xue.)

Sentiment data include financial news (Hong Kong and US), and tweets data i.e. Twitter posts (US only) for each ticker symbol. The fact sheet in Table 11 summarises the total size and number of rows of the collected sentiment data.

<table>
<thead>
<tr>
<th></th>
<th>Number of rows</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>HK news (aastock)</td>
<td>4861</td>
<td>4.7 MB</td>
</tr>
<tr>
<td>US news (finviz.com)</td>
<td>741,006</td>
<td>81.1 MB</td>
</tr>
<tr>
<td>Tweets (US)</td>
<td>1,989,362</td>
<td>240.1 MB</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2,765,229</strong></td>
<td><strong>325.9 MB</strong></td>
</tr>
</tbody>
</table>

Table 11: Fact sheet of property transaction data collected in the database.

**Financial News** Relevant financial news for each listed company in HKEX were collected from aastocks.com, which is a platform that centralises international news headlines relevant to Hong Kong shares.

Web scraping was done using Beautiful Soup. The page that contains all relevant news for a particular ticker symbol was first retrieved, and the `div` element that contains the table of headlines (as shown in Figure 9) was selected. The `div` element that stores the date of each news was also scraped, and was then concatenated with the table of news headlines. By wrapping this operation in a for loop, the news for each ticker could be collected accordingly.

![Figure 9: Table to be scraped that contains relevant news headlines for the ticker symbol 0005.HK on aastocks.com.](image)

Relevant financial news for each listed company in NASDAQ and NYSE were collected from finviz.com, which is a news platform that features latest financial news from major newsagents such as Yahoo! Finance and Bloomberg.

Similarly, a web scraper was built using the Beautiful Soup library to retrieve the page that features the news for a ticker symbol. Then, the table of news headlines was obtained selecting the element with `id="news-table"` (as shown in Figure 10). After that, these scraped tables could be stored as separate csv files.
Tweets Data  Relevant tweets for each company listed in the US (NYSE and NASDAQ) were collected via the Tweepy API. Take "AAPL" as an example, tweets with the cashtag \$AAPL or hashtag #AAPL (along with their post dates) were downloaded.

Below shows an extract of the tweets data collected for AAPL.

2021-02-04,'Apple and Hyundai-Kia pushing toward deal on Apple Car $AAPL'
2021-02-04,'RT @DCDOWORK: NEWS$ NIO (NYSE: NIO) Is Accelerating its Plans To Enter the American Market As New Circumstantial Evidence Emerges'

6.2 Part 1(b) - Backtester

A backtester has been built in Python, which could be used for testing a strategy given a particular ticker symbol and a selected date range.

6.2.1 Main Backtesting Function

Within the code/ directory, the file backtest.py contains the main backtesting function:

```python
Backtest(ticker, signals, df, initial_capital=float(100000.0))
```

The following is the description of its parameters:

- **ticker**: string
ticker symbol to be backtested, e.g. "0001.HK"
- **signals**: DataFrame or named Series
  the signals dataframe generated from a strategy
- **df**: DataFrame or named Series
  the dataframe that consists of a stock’s daily prices in a given date range
- **initial_capital**: float (optional)
  amount of initial capital, default = 100000

The function returns:

- **portfolio**: DataFrame or named Series
daily portfolio record in the given date range (example shown in Figure 11)
- **fig**: a matplotlib figure instance
  figure with the total portfolio value and trading signals plotted
6.2.2 Plotting Signals

The backtesting function makes use of the `matplotlib` library to plot the portfolio value and trading signals.

An example figure generated by the backtester is shown in Figure 12, where the closing price of the stock is plotted as a line; buying and selling signals are plotted with green and red markers on the line respectively.

Figure 12: Example figure of the stock’s closing price with buy and sell signals plotted by the backtester.

6.2.3 Evaluation Metrics

Several evaluation metrics have been implemented for analysing the strategy performance after calling the backtesting function. All of these functions are included in `/code/evaluate.py`. Below are the explanation and API description of each metric.
**Portfolio Return**  It is the return rate of the portfolio over the given date range.

\[ \text{PortfolioReturn} \text{(portfolio)} \]

Parameters:
- \text{portfolio} : DataFrame or named Series
  portfolio dataframe generated from Backtest()

Returns:
- \text{fig} : a matplotlib figure instance
  figure with portfolio return plotted over the date range of the dataframe

**Sharpe Ratio**  It is the return-to-risk ratio that describes how much excess return the investor could gain per unit of extra risk endured. The formula is:

\[ \text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \]

where \( R_p \) = expected return of the portfolio, \( R_f \) = risk-free rate (assume = 0), \( \sigma_p \) = standard deviation of the portfolio’s excess return.

\[ \text{SharpeRatio} \text{(portfolio)} \]

Parameters:
- \text{portfolio} : DataFrame or named Series
  portfolio dataframe generated from Backtest()

Returns:
- \text{sharpe_ratio} : figure with portfolio return plotted over the date range of the dataframe

**Maximum Drawdown (MDD)**  This metric measures the downside risk of a stock over a specific time period. The formula is:

\[ \text{Maximum Drawdown} = \frac{P - L}{P} \]

where \( P \) = peak value before largest drop, \( L \) = lowest value at trough before a new peak is attained.

\[ \text{MaxDrawdown} \text{(portfolio, window=252)} \]

Parameters:
- \text{portfolio} : DataFrame or named Series
  portfolio dataframe generated from Backtest()
- \text{window} : integer (optional)
  time window for calculating the maximum drawdown, default = 252

Returns:
fig : a matplotlib figure instance
figure with maximum drawdown plotted over the date range of the dataframe
max_daily_drawdown : float
maximum drawdown of the portfolio in the time window
daily_drawdown : float
daily drawdown of the portfolio

**Compound Annual Growth Rate (CAGR)**  It computes the rate of return required for an investment to grow from its initial value to its ending value, assuming that all profits generated were reinvested at the end of each year. The formula is:

\[
CAGR = \left(\frac{\text{Ending value}}{\text{Starting value}}\right)^{\frac{\text{days}}{n}} - 1
\]

where \( n \) = number of days in the time period.

CAGR(portfolio)

Parameters:

portfolio : DataFrame or named Series
portfolio dataframe generated from Backtest()

Returns:

cagr : float
CAGR of the portfolio

**Standard Deviation**  It measures the risk of a portfolio. The formula is:

\[
s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}}
\]

where \( x_i \) = each data point, \( \bar{x} \) = the average of \( x_i \), and \( n \) = number of data points. Note that the formula for sample standard deviation is applied.

StandardDeviation(portfolio)

Parameters:

portfolio : DataFrame or named Series
portfolio dataframe generated from Backtest()

Returns:

sd : float
Annualised standard deviation of the portfolio
6.3 Part 2(a) - Trading Strategies

There are namely two categories of strategies - (1) Conventional strategies i.e. technical and fundamental analysis that generate signals solely based on microeconomic data; and (2) Integrated strategies that generate signals based on all three dimensions (microeconomic, macroeconomic, sentiment) of data inputs.

All of these strategies have been implemented in Python. One of the technical strategies (Moving Average) has also been implemented in Julia, which serves to demonstrate the basic syntax of writing a strategy in Julia.

6.3.1 Technical Analysis

Technical Indicators  A total of 16 technical indicators out of four different categories (trend, momentum, volume, volatility) have been implemented in Python. Table 12 shows the full list of indicators featured in the repository. Each indicator has its own class, and could be imported from the `technical-analysis_python/strategy` directory. Every indicator also comes along with an example code file prefixed with `main_` which demonstrates how to call the methods for each indicator.

Take Moving Average Crossover as an example:

```python
class MovingAverageCrossover(Indicator):
    def __init__(self, df, short_window=40, long_window=100):
        self.df = df
        self.short_window = short_window
        self.long_window = long_window
    def gen_signals(self):
        ...
        return signals
```

An instance of `MovingAverageCrossover` could be created by passing in the stock tick dataframe `df`, and the `gen_signals` method could be called to generate the `signals` dataframe (example output shown in Figure 13).

<table>
<thead>
<tr>
<th>signal</th>
<th>short_mavg</th>
<th>long_mavg</th>
<th>positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-12-24</td>
<td>1.0</td>
<td>264.221056</td>
<td>235.930897</td>
</tr>
<tr>
<td>2019-12-26</td>
<td>1.0</td>
<td>265.398771</td>
<td>236.904727</td>
</tr>
<tr>
<td>2019-12-27</td>
<td>1.0</td>
<td>266.574494</td>
<td>237.841293</td>
</tr>
<tr>
<td>2019-12-30</td>
<td>1.0</td>
<td>267.656605</td>
<td>238.774814</td>
</tr>
<tr>
<td>2019-12-31</td>
<td>1.0</td>
<td>268.616614</td>
<td>239.686144</td>
</tr>
</tbody>
</table>

Figure 13: An extract of the signals dataframe generated by the Moving Average Crossover function.
<table>
<thead>
<tr>
<th>Category</th>
<th>Indicator(s)</th>
</tr>
</thead>
</table>
| Trend    | • Moving Average Crossovers  
          | • Moving Average Convergence Divergence (MACD)  
          | • Parabolic Stop and Reverse (Parabolic SAR) |
| Momentum | • Commodity Channel Index (CCI)  
          | • Relative Strength Index (RSI)  
          | • Rate of Change (ROC)  
          | • Stochastic Oscillator (STC)  
          | • True Strength Index (TSI)  
          | • Money Flow Index (MFI)  
          | • Williams %R |
| Volume   | • Chaikin Oscillator  
          | • On-Balance Volume (BOV)  
          | • Volume Rate of Change |
| Volatility | • Bollinger Bands  
          | • Average True Range (ATR)  
          | • Standard Deviation |

Table 12: List of technical indicators implemented in the repository.
In general, the steps for running a technical analysis strategy is shown in Algorithm 4. 

**Algorithm 4: Run & backtest a technical analysis strategy**

<table>
<thead>
<tr>
<th>Data:</th>
<th>Stock tick data for a specified ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>png files of figures generated and backtesting results printed on terminal</td>
</tr>
</tbody>
</table>

1. Import required libraries;
2. \( df \leftarrow \) Load stock tick data for specified ticker symbol;
3. Select time range to backtest;
4. \( ticker \leftarrow \) Set ticker symbol to test;
5. \( strategy \leftarrow \) Create an instance of the strategy class by passing \( df \) as argument;
6. \( signals, figure \leftarrow \) Call `gen_signals` and `plot_signals` methods in strategy class;
7. Call the Backtester function by passing \( ticker, signals \) and \( df \) as arguments;
8. Call evaluation metric functions as needed (e.g. sharpe ratio, maximum drawdown);

The figures generated by the backtester in the example main files will be saved in the `technical-analysis_python/figures` directory, and all absolute values (e.g. final portfolio value and number of trade signals generated) will all be printed in the terminal.

**Julia Example** In addition to the implementation of technical indicators in Python, we have coded one of the strategies (Moving Average Crossover) in Julia. We would like to feature this programming language in the code repository, since it is one of the fastest growing languages designed for quantitative finance. Below are some highlights of the advantages in using Julia:

- It has a legible syntax and is easy to learn.
- It incorporates vector notations and DataFrames as part of the language.
- It compiles codes in advance and thus is designed to be fast.

All Julia code examples (for exploratory data analysis and technical analysis) are contained in the `./code/technical-analysis_julia/` directory. The workflow of running the trading strategy follows from the one in Python (Algorithm 4). The evaluation metrics implemented in Julia also share the same parameters and output with those in Python (refer to Section 6.2.3).

By installing the Julia kernel for Jupyter, the code could also be run in Jupyter Notebook as shown in Figure 14. Example notebooks of running the strategy in Julia could be found in the repository.
6.3.2 Fundamental Analysis

Stock Screening  The three types of financial statements (income statement, balance sheet, cash flow statement) collected in the database are used to compute various financial ratios in order to evaluate the performance of a listed company. Based on the performance of a company relative to others, an investor could filter out the best performing tickers to buy (and presumably hold for a long period of time) by configuring certain criteria. By way of example, a valid criteria could be to filter out companies with earnings per share (EPS) higher than the overall average. This entire workflow of selecting stocks is known as "stock screening", and a Jupyter Notebook file ratio-analysis.ipynb is featured in the repository to demonstrate how the whole process could be carried out in programming.

The major steps of conducting stock screening is illustrated in Algorithm 5.

Algorithm 5: Carry out stock screening

**Data:** Financial statements for tickers in a stock exchange  
**Result:** filtered_df, A list of filtered stocks desirable for investment

1. Import required libraries;
2. merged_df ← Merge dataframes of three types of financial statements;
3. ratios_df ← Compute the financial ratios according to the equations (which are available on the documentation website);
4. masks ← Create screening masks;
5. filtered_df ← Apply the masks to the dataframe;
6. Find the name and sector of the company by matching with the ticker symbol in the master list

Note that there are 4 categories of financial ratios (short-term solvency ratios, turnover ratios, financial leverage ratios, profitability ratios), and they respectively assess different...
aspects of a company. The list of most commonly used financial ratios are shown in Table 13. One could configure the screening masks based on his/her investment preferences by selecting the relevant ratios.

<table>
<thead>
<tr>
<th>Category</th>
<th>Ratio(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Short-term solvency ratios</strong></td>
<td>• Current ratio&lt;br&gt;• Quick ratio&lt;br&gt;• Cash ratio&lt;br&gt;• Networking capital to current liabilities</td>
</tr>
<tr>
<td><strong>Turnover ratios</strong></td>
<td>• Average collection period&lt;br&gt;• Inventory turnover ratios&lt;br&gt;• Receivable turnover&lt;br&gt;• Fixed asset turnover&lt;br&gt;• Total asset turnover</td>
</tr>
<tr>
<td><strong>Financial leverage ratios</strong></td>
<td>• Total debt ratio&lt;br&gt;• Debt to equity&lt;br&gt;• Equity ratio&lt;br&gt;• Long-term debt ratio&lt;br&gt;• Times interest earned ratio</td>
</tr>
<tr>
<td><strong>Profitability ratios</strong></td>
<td>• Gross profit margin&lt;br&gt;• Net profit margin&lt;br&gt;• Return on assets (ROA)&lt;br&gt;• Return on equity (ROE)&lt;br&gt;• Earning per share (EPS)</td>
</tr>
</tbody>
</table>

Table 13: List of common financial ratios used for fundamental analysis.

**Bankruptcy Prediction** Not only are fundamentals data useful for stock screening, but they could also be used for predicting the bankruptcy of a listed company. Based on the Simple Analysis of Failure (SAF2002) model (Shirata, 2003), Altman (2013) and Beaver (1966), the following ratios are chosen as input variables for the bankruptcy
prediction machine learning model.

\[ X_1 = \text{working capital} \div \text{total assets} \]
\[ X_2 = \text{retained earnings} \div \text{total assets} \]
\[ X_3 = \text{earnings before interest and taxes (EBIT)} \div \text{total assets} \]
\[ X_4 = \text{total equity (book)} \div \text{total assets} \]
\[ X_5 = \text{net income} \div \text{total assets} \]
\[ X_6 = \text{total liabilities} \div \text{total assets} \]
\[ X_7 = \text{cash flow from operation} \div \text{total liabilities} \]

The machine learning model was first trained with the labelled dataset\(^4\) collected by the UCLA School of Law, with records of more than 200 public companies in the US. A company that has gone bankrupt within 3 years is labelled with 0, and those which have survived are labelled with 1.

The dataset was split into training set and test set (with train-test ratio = 7:3), so that we could achieve unbiased evaluation (i.e. the model is evaluated with fresh data during inferencing). With regards to the selection of machine learning model, four different models have been coded and be used for testing, which are (1) Support Vector Machine (SVM), (2) Decision Tree, (3) Random Forest and (4) K-Nearest Neighbours (KNN). They are all models typically used for classification. The accuracy score for each model is equal to the percentage of predictions that is correct.

Concerning the SVM model, the soft margin constant C, Gamma and kernel function are hyperparameters to be set. The soft margin constant is used to control error, and gamma is used for giving curvature weight to the decision boundary. During training, C was set as \([0.1, 1, 10, 100, 1000]\) and gamma was set as \([1, 0.1, 0.01, 0.001, 0.0001]\). The \((C, \text{gamma})\) pair that gives the best result was chosen via grid search. The kernel function was configured as the radial basis function (RBF).

Concerning the Decision Tree model, the number of trees in the forest \((n\_estimators)\) was set as 100.

Concerning the KNN model, the number of neighbours \((n\_neighbours)\) was set as 1.

Table 14 summarises the accuracy scores of the different types of model after training.

<table>
<thead>
<tr>
<th>Model</th>
<th>1-year</th>
<th>3-year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine (SVM)</td>
<td>73%</td>
<td>65%</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>69%</td>
<td>54%</td>
</tr>
<tr>
<td>Random Forest</td>
<td>88%</td>
<td>65%</td>
</tr>
<tr>
<td>K-Nearest Neighbour (KNN)</td>
<td>77%</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 14: Accuracy scores of different machine learning models in predicting 1-year and 3-year bankruptcy of a listed company.

\(^4\)https://lopucki.law.ucla.edu/
From Table 14, the Random Forest model has the highest accuracy (88%) in predicting whether a company would be bankrupt in one year; and both SVM and Random Forest have the highest accuracy (65%) in predicting the three-year bankruptcy of a company.

### 6.3.3 Macroeconomic Analysis

*(The implementation of this part credits to Lee Kwanyoung.)*

Trends in economic indicators, monthly average housing prices and stock prices are intertwined. To establish an integrated strategy that takes macroeconomic data as inputs, we have conducted exploratory data analysis in order to investigate the correlation between these three.

**Economic Indicator Analysis** We first explored how the economic indicators of Hong Kong affect the Hang Seng Index (HSI). Both univariate analysis and bivariate analysis were carried out to analyse each indicator in order find the relationship between the indicator and HSI.

In **univariate analysis**, the distribution of numerical features was examined by calling the pandas `Dataframe.describe()` function, which produces a statistical summary such as mean, standard deviation, min, and max of the dataframe. In addition, the `seaborn.distplot()` function was used to visualise the results with histograms. Figure 15 shows the code snippet for univariate analysis.

```python
def univariate_analysis(feature_name):
    # Statistical summary
    print(df[feature_name].describe())

    # Histogram
    plt.figure(figsize=(10,5))
    sns.distplot(df[feature_name], axlabel=var);
```

Figure 15: Code snippet for univariate analysis *(Credits to Lee Kwanyoung).*

In **bivariate analysis**, the correlations between different indicators and HSI were studied. The correlation between two features was visualised via a scatter plot and a regression line (as shown in Figure 16). The regression line in the figure has a positive slope, which indicates that the value of imports is positively correlated to the HSI. Then, the `pandas.DataFrame.corr()` and `seaborn.heatmap()` functions were used to compute the pairwise correlation of features and visualise the correlation matrix. The resulted matrix is shown in Figure 17.
Figure 16: An example of a scatter plot with a regression line (Credits to Lee Kwanyoung).

Figure 17: Correlation heatmap of macroeconomic indicators and HSI (Credits to Lee Kwanyoung).
From Figure 17, we can see that the following economic indicators are positively related to HSI (in descending order of correlation):

- GDP
- Housing price
- Population
- Imports
- Year
- Total exports
- Composite Consumer Price Index

And the following are negatively related to HSI (in descending order of correlation):

- Seasonally adjusted unemployment rate
- Non-seasonally adjusted unemployment rate

Applying similar methods for bivariate analysis, we also examined the correlation between the economic indicators and the monthly average housing price per saleable area in Hong Kong. The correlation matrix is shown in Figure 18.

![Correlation heatmap of macroeconomic indicators and the average monthly price per saleable area in Hong Kong (Credits to Lee Kwanyoung).](image)

From Figure 18, we can see that the following economic indicators are positively related...
to the monthly average price per saleable area (in descending order of correlation):

- Population
- GDP
- Composite Consumer Price Index
- Year
- Month
- Imports
- Total exports

Note that both seasonally adjusted and non-seasonally adjusted unemployment rates are not correlated with the average housing price per saleable area.

**Transaction Data Analysis**  As shown in above, the housing price in Hong Kong has a strong positive correlation with the HSI. In fact, the properties and construction sector accounts for over 10% of weighting in the HSI, and thus the real estate market data could be a source of volatility in the Hong Kong stock market.

Hence, in this part, we move on to examine the relationship between an apartment’s features and the individual housing prices of Hong Kong. We used similar methodology applied in economic indicator analysis to conduct univariate analysis and bivariate analysis respectively. Transaction records from Midland Realty were used for the analysis.

In **univariate analysis**, the distribution of the individual housing prices was examined. The housing price of Hong Kong has a mean of 9 million HKD and a standard deviation of 13 million HKD. The skewness and kurtosis are 26.9 and 1526.4 respectively, showing that the housing price of Hong Kong is skewed positively to a very high extent. In order to obtain better result for the bivariate analysis, outliers were removed by using the standard deviation.

In **bivariate analysis**, the correlation coefficient between the features describing an apartment and the individual housing price was computed. Figure 19 shows the correlation map of the top 7 features that has the highest correlation with the housing price, which are (in descending order of correlation):

- Saleable area
- Last price
- Gross area
- Bedroom
- Floor
- Region
- Building age

**Property Price Prediction**  Based on the transaction data analysis, property price prediction models were created. The data were split with a ratio of 8:2 and used to train the machine learning model. The input variables were the top 7 features selected from the analysis, and the output feature was the individual housing price. Before training the model, log transformation was used to normalise the highly skewed price data.
Figure 19: Correlation heatmap of the top 7 features selected in the property transaction data (Credits to Lee Kwanyoung).
We have tested with 4 different types of models, which are - (1) XGBoost, (2) Lasso, (3) Random Forest, and (4) Linear Regression. The performance of each model was evaluated by the root mean square log error (RMSLE). The RMSLE was used since the price values are too large, and the RMSLE could prevent penalising the large differences between actual and predicted prices. Table 15 shows the RMSLE values of each trained model.

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSLE</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGBoost</td>
<td>0.1608</td>
<td>0.1645</td>
<td></td>
</tr>
<tr>
<td>Lasso</td>
<td>0.2640</td>
<td>0.2652</td>
<td></td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.3077</td>
<td>0.3071</td>
<td></td>
</tr>
<tr>
<td>Linear Regression</td>
<td>0.2630</td>
<td>0.2643</td>
<td></td>
</tr>
</tbody>
</table>

Table 15: RMSLE values of the housing price prediction models.

According to Table 15, XGBoost achieves the best performance among the 4 models. We hypothesise that this is because XGBoost uses a more accurate implementation of gradient boosting algorithm and optimised regularisation, which induces it to give a better result than other models. Figure 20 shows the graph of the actual and predicted housing price for XGBoost. It tends to give smaller errors for the range of prices which has a larger size of training data.

Figure 20: The graph of actual and predicted housing price for XGBoost (Credits to Lee Kwanyoung).
6.3.4 Sentiment Analysis

(The implementation of this part credits to Wu Xue.)

The VADER and TextBlob libraries have been used to generate sentiment labels for the news headlines and tweets collected in the database.

**VADER Sentiment Analyser**  
VADER (Valence Aware Dictionary and sEntiment Reasoner) (Hutto and Gilbert, 2014) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to analysing social media texts. It is designed to be sensitive to both polarity (whether the sentiment is positive or negative) and intensity (the extent of positivity or negativity) of emotions. According to the experiment conducted by Hutto and Gilbert (2014) on analysing 4200 tweets, it achieves an F1-score of 0.96.

An example of the output from the VADER sentiment analyser is as the following:

```
{'neg': 0.0, 'neu': 0.326, 'pos': 0.674, 'compound': 0.7351}
```

The *compound score* is a metric that sums up all the lexicon ratings (i.e. the score of each vocabulary) which have been normalised between -1 (most extreme negative) and +1 (most extreme positive). The sentiment labels are derived from the compound score according to the following rules:

- **Positive sentiment** \((= 2)\): compound score \(\geq 0.01\)
- **Neutral sentiment** \((= 1)\): compound score \(> -0.01\)
- **Negative sentiment** \((= 0)\): compound score \(\leq -0.01\)

**TextBlob Sentiment Analyser**  
TextBlob (Loria, 2018) is a lexicon-based tool that leverages the NLTK library to achieve sentiment analysis. It measures the polarity and subjectivity (amount of personal opinion and factual information contained in the text) of a sentence. The output from the analyser is a weighted average sentiment score over all the words in a sentence. The advantage of this approach is that it can diffuse out the effect of widely varying polarities between words, for example as having ‘glad’ and ‘but’ simultaneously in a sentence. An example output from the analyser is as follows:

```
tweet = TextBlob("Today is the first really strong day for CN tech ADRs since the correction started in mid-February.")

>>> tweet.sentiment
Sentiment(polarity=0.34166666666667, subjectivity=0.53333333333333)
```

We applied the same rules (used for processing the VADER compound score) to obtain the sentiment labels for each news headline and tweet collected in the database.
6.4 Part 2(b) - Integrated Strategies

Integrated strategies refer to strategies that take all three dimensions of data (microeconomic, macroeconomic, sentiment) as input in order to generate trading signals. All of the code examples for integrated strategies could be found in the code/integrated-strategy directory.

We have implemented three different integrated strategies in Python:

- **Baseline model** - Filter out trading signals generated by technical analysis strategies using macroeconomic and sentiment data
- **Single-feature LSTM** - End-to-end trading signal prediction with stock price series as the only input
- **Multi-feature LSTM** - End-to-end trading signal prediction with stock price, technical indicators, macroeconomic data and sentiment labels as inputs

The following provides the explanation of each strategy’s working mechanism.

6.4.1 Baseline Strategy

The baseline model makes use of the trading signals generated from a technical analysis strategy, and filter out those that are conflicting with the macroeconomic trend or sentiment label. The steps for running the baseline strategy is shown in Algorithm 6.

---

Algorithm 6: Baseline strategy

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Import required libraries;</td>
</tr>
<tr>
<td>2</td>
<td>df ← Load stock tick data for specified ticker symbol;</td>
</tr>
<tr>
<td>3</td>
<td>Select time range to backtest;</td>
</tr>
<tr>
<td>4</td>
<td>ticker ← Set ticker symbol to test;</td>
</tr>
<tr>
<td>5</td>
<td>strategy ← Create an instance of the strategy class by passing df as argument;</td>
</tr>
<tr>
<td>6</td>
<td>signals ← Call gen_signals method in strategy class;</td>
</tr>
<tr>
<td>7</td>
<td>s_gdp, s_unemploy, s_property ← Call GetSensitivity(df);</td>
</tr>
<tr>
<td>8</td>
<td>signals ← Call GetMacrodata(signals);</td>
</tr>
<tr>
<td>9</td>
<td>Apply macroeconomic filter on signals;</td>
</tr>
<tr>
<td>10</td>
<td>Apply sentiment filter on signals;</td>
</tr>
<tr>
<td>11</td>
<td>Call the Backtester function by passing ticker, signals and df as arguments;</td>
</tr>
<tr>
<td>12</td>
<td>Call evaluation metric functions as needed (e.g. sharpe ratio, maximum drawdown);</td>
</tr>
</tbody>
</table>

---

In Line 7, the GetSensitivity(df) function returns the stock price’s sensitivities to GDP, unemployment rate and average property price respectively.

In Line 8, the GetMacrodata(signals) function appends the normalised economic indicators data to the signals dataframe.
In the macroeconomic filter, the adjusting factor is calculated in the following way:

\[
adj\_factor = GDP \times s_{GDP} + \text{unemployment rate} \times s_{unemployment} + \text{average property price} \times s_{property}
\]

Note that GDP, unemployment rate and average property price are all normalised, and the resulting value of \(adj\_factor\) lies within the range of \((-1,1)\). We apply it on the signals dataframe simply by adding it to the trading signal column and rounding it off to the nearest integer.

In the sentiment filter, the sentiment labels generated from the VADER analyser is first concatenated with signals dataframe to generate a \texttt{merged\_df}. Conflicting signals are then eliminated using the following logic:

```python
# when there is a buy signal but a -ve sentiment label
merged_df[(buy_signal) & (neg_label)]['filtered_signal'] = 0.0

# when there is a sell signal but a +ve sentiment label
merged_df[(sell_signal) & (pos_label)]['filtered_signal'] = 0.0
```

Eventually, these filtered signals are passed to the backtester function to analyse the strategy performance.

### 6.4.2 Trading Signal Prediction with LSTM

Instead of relying on rule-based logic like the baseline strategy, this model makes use of a machine learning model (Long-Short-Term Memory (LSTM) Network) to generate trading signals.

**LSTM Network** LSTM (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) that is useful for sequence prediction problems. As opposed to vanilla RNNs, LSTMs are specifically designed to capture long-term dependencies in time series data.

In the repository, we made use of the \texttt{PyTorch} machine learning library to create the LSTM model. The model is defined as follows making use of the \texttt{nn.module}:

```python
nn.LSTM(input_dim, hidden_dim, num_layers, dropout, batch_first=True)
```

\texttt{input_dim} and \texttt{hidden_dim} are hyperparameters to be configured. The \texttt{dropout} rate is the probability that the layer outputs are ignored or “dropped out” during training. It is a regularisation method the helps prevent overfitting (Srivastava et al., 2014). A common value of the dropout rate is about 0.4-0.8 for hidden layers.

**Single-feature Prediction with LSTM** The LSTM model takes only the stock price data series as input, and predicts the stock price at the next time step. In order to generate trading signals, we’ll then compare the actual and predicted prices to check if
the stock is undervalued or overvalued. We configured the LSTM model with `input_dim = 1`, `hidden_dim = 32`, `num_layers = 2`, `output_dim = 1`.

The steps of running the strategy is outlined in Algorithm 7.

**Algorithm 7: Trading signal prediction with single-feature LSTM model**

<table>
<thead>
<tr>
<th>Data:</th>
<th>Stock tick data for a specified ticker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result:</td>
<td>png files of figures generated and backtesting results printed on terminal</td>
</tr>
</tbody>
</table>

1. Import required libraries;
2. `df ← Load stock tick data for specified ticker symbol;`
3. Select time range to backtest;
4. Scale the `Close` column with `MinMaxScaler;`
5. Make training and testing sets in torch;
6. Set hyperparameters;
7. for `t in range(num_epochs)` do
   8. Pass training data into LSTM model;
   9. Call backpropagation;
   10. Update parameters;
8. end
9. Calculate root mean squared error (RMSE) with testing set;
10. `signals ← gen_signal(y_pred, y, dates);`
11. Save signals as csv file;

Note that `y_pred` would be predicted prices of the stock, and `y` represents the actual prices. The `gen_signal` function generates a buy or sell signal according to the following logic:

```python
for p, a in zip(y_pred, y):
    if (abs(p - a) < 1.0):
        signal.append(0)

    # shows that current price is overvalued, sell the stock
    elif (p > a):
        signal.append(-1)

    # shows that current price is undervalued, buy the stock
    elif (p < a):
        signal.append(1)
```

**Multi-feature Prediction with LSTM**  As opposed to the single-feature model, this LSTM model takes the stock price, technical indicator, economic indicator data and sentiment labels as inputs to predict the price, which will then be converted to trading signals by the `gen_signals` function. We configured the LSTM model with `input_dim = 7`, `hidden_dim = 64`, `num_layers = 4`, `output_dim = 7`.

The steps of running the multi-feature model is largely the same as Algorithm 7, except that we load the merged dataframe (with 7 feature columns) instead of taking the closing
price as a single feature.

In the code file **LSTM-train_wrapper.py**, the following line gets the merged dataframe:

```python
df, scaled, scaler = merge_data(symbol, data_dir,
                                sentiment_data_dir, 'macd-crossover')
```

An example of the dataframe output `df` (with multiple feature columns) from the `merge_data` function is shown in Figure 21.

![Figure 21: An extract of the example merged dataframe.](image)

To backtest these signals dataframes generated by the LSTM model, we could simply run the **output_backtester.py** file in the repository that loads the csv files in the output directory and pass them to the backtester function for performance evaluation.
6.5 Part 2(c) - Paper Trading

The repository contains code that demonstrates the execution of trading orders and request for data via the Interactive Brokers API. The installation guide of the API and setup instructions for connecting to the Interactive Brokers (IB) Trader Workstation (TWS) are featured in the documentation website. The code examples for Paper Trading could be found in the /code/paper-trading/ directory.

The five example files illustrate different usages of the IB API:

- **main_order_stock.py** - create, modify, cancel and submit orders (Stock)
- **main_order_FX.py** - create, modify, cancel and submit orders (FX pair)
- **main_reqMktData.py** - request streaming market data for a ticker symbol
- **main_reqHistoricalData.py** - request historical market data for a ticker symbol
- **main_reqAccountSummary.py** - get the account summary

For instance, **main_order_stock.py** covers the following steps in showing how to place an order for the stock with ticker symbol 0001.HK:

```python
# Establish a connection to the IB Trader Work Station
app = App()
app.connect('127.0.0.1', 7497, 0)
app.nextorderId = None

# Start the socket in a thread
api_thread = threading.Thread(target=run_loop)
api_thread.start()

data_time.sleep(1)

# Create contracts
contract = Contract()
contract.symbol = "1"
contract.secType = "STK"
contract.exchange = "SEHK"
contract.currency = "HKD"

# Create the order
order = Order()
order.action = 'BUY'
order.totalQuantity = 500
order.orderType = 'MKT'

# Place the order
app.placeOrder(app.nextorderId, contract, order)
data_time.sleep(5)

# Disconnect to the app
```
After running `main_order_stock.py`, we could check our portfolio in the IB Trader Work Station. As shown in Figure 22, the 500 shares of 0001.HK is included in the "My Investments" tab.

![Figure 22: Interface of the Interactive Brokers Trader Work Station.](image)

**Daily Paper Trading** Putting together all topics covered in the repository, we have also implemented the "Daily trading strategy" that allows the user to generate a trading signal via a trained LSTM model, then pass it to the IB API for placing an order in a Paper Trading Account. All of the code files could be found in the `code/integrated-strategy/` directory.

The pipeline works as follows:

**Step 1**: run `LSTM-train_daily.py` - train the multi-feature LSTM model for a selected ticker and save the model.

**Step 2**: run `daily_trading_strategy.py` - load the trained LSTM model and save the daily trading signal to a csv file in the database.

**Step 3**: run `daily_trading_order.py` - load the file with the daily trading signal and make the order in the IB Paper Trading account.

Note that Step 1 is mostly just a reuse of Algorithm 7, and Step 3 is largely similar to the example of making an order given in `main_order.py` (in the Paper Trading part). The methodology of Step 2 is outlined in Algorithm 8.
Algorithm 8: Daily trading signal generation

Data: selected ticker symbol, trained LSTM model
Result: csv file containing the daily trading signal

1 Import required libraries;
2 Collect news data for ticker by calling collect_news();
3 Collect sentiment labels by calling collect_individual_sentiment();
4 df ← get_price(ticker) # gets dataframe with price and sentiment labels;
5 df ← collect_macro_data(df, dir_name, ticker);
6 model ← load saved LSTM model;
7 Get prediction by passing scaled df to model (i.e. inferencing);
8 Save predicted daily signal as csv file;

This complete pipeline illustrates how we could connect everything together and conduct trading in a practical setting.
6.6 Part 3 - Documentation Website

The documentation website goes hand in hand with the code implementation in the repository to guide learners in digesting the code and understanding relevant financial concepts at the same time. We have built the website using Sphinx, which is an open-source Python documentation generator. The website has been hosted on readthedocs, can be accessed at https://algo-trading.readthedocs.io/en/latest/. The homepage of the website is shown in Figure 23.

![Figure 23: The homepage of the documentation website.](image)

6.6.1 Layout Design

Each page of the website follows a consistent layout that renders the content to be easy to follow and reader-friendly.

**Learning Goals** At the beginning of each page, the learning goals are first listed out in bullet points (as shown in Figure 24) so that the learner could be geared up for what to learn in the upcoming tutorial.

**Highlight Box** All important points such as definitions and tips are placed in highlight boxes of various colours (as shown in Figure 25), so that they remain catchy.

**Disclaimer** A legal disclaimer is featured at the end of each page (as shown in Figure 26) which serves to remind users that the content of the website is only for informational purposes, and thus does not constitute investment advice or investment recommendation.
Figure 24: The bullet points of learning goals listed at the beginning of each page.

Figure 25: The highlight boxes that feature important definitions and tips in the tutorial.

Figure 26: The legal disclaimer featured at the end of each page.
6.6.2 Tutorial Walkthrough

The tutorials in the website are divided into three major parts, namely:

- **Part 1:** Intro to Algo Trading
- **Part 2:** Core Trading Strategies
- **Part 3:** Machine Learning

These parts are arranged in ascending order of difficulty, so the user could first learn the essentials of data science and writing a simple trading strategy, then move on to learn about advanced strategies with more inputs that involve the use of machine learning models.

Each part of the tutorial features explanation of basic financial concepts as well as mathematical equations for various strategies implemented in the code base. They are all coupled with code examples and descriptions so that it is easy for beginners (especially those without any financial background) to follow, as illustrated in Figure 27.

---

**Figure 27:** A page of the documentation website featuring explanation of the Commodity Channel Index, the equation for calculating the indicator and a code example.
7. Experiment & Results

7.1 Vanilla Technical Analysis Strategies

Making use of the backtester and technical indicators implemented in the repository, we have carried out experiments to analyse the strength and characteristics of different indicators. We first suggested a scenario based on a realistic investment decision, then conducted the experiment with the aim of answering the question prompt. A total of 5 scenarios have been evaluated, and the following would go through the findings in each scenario one by one.

**Scenario 1**  Given a ticker symbol and a strategy, we would like to investigate what the optimal time span (long-term or short-term) of the investment is.

- **Independent variable(s):** Time span
- **Controlled variable(s):** Ticker = 0005.HK, Strategy = MACD crossover

![Figure 28: Line plot that illustrates how the portfolio value changes with the time span (varying from 10 years to 6 months), with 0005.HK as the chosen ticker and applying the MACD crossover strategy.](image)

Figure 28: Line plot that illustrates how the portfolio value changes with the time span (varying from 10 years to 6 months), with 0005.HK as the chosen ticker and applying the MACD crossover strategy.

Having 0005.HK as an example, from Figure 28, it is suggested that the portfolio return has a "wave-like" pattern with a period of roughly four years. In other words, a crest or trough is formed for every four-year reduction on the time span (there is a crest when time span = 10 years, 6 years, 2 years).

On the other hand, Figure 29 shows the relationship between the number of trading signals generated by MACD crossover and the duration of the time span. The number of trading signals decreases almost linearly with the time duration.
Scenario 2  Given a fixed time span and a strategy, we would like to know the general performance of the strategy by backtesting on different tickers.

- **Independent variable(s):** Ticker
- **Controlled variable(s):** Time period = 2015-01-01 to 2019-01-01, Strategy = RSI

<table>
<thead>
<tr>
<th>Ticker</th>
<th>Portfolio return</th>
<th>Sharpe ratio</th>
<th>Number of trades</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>1477.000000</td>
<td>1477.000000</td>
<td>1477.000000</td>
</tr>
<tr>
<td>mean</td>
<td>1203.303318</td>
<td>-0.075937</td>
<td>-0.000303</td>
</tr>
<tr>
<td>std</td>
<td>1160.275490</td>
<td>1.118188</td>
<td>0.594079</td>
</tr>
<tr>
<td>min</td>
<td>1.000000</td>
<td>-12.249520</td>
<td>-2.246904</td>
</tr>
<tr>
<td>25%</td>
<td>400.000000</td>
<td>-0.091619</td>
<td>-0.378983</td>
</tr>
<tr>
<td>50%</td>
<td>885.000000</td>
<td>-0.001992</td>
<td>-0.018826</td>
</tr>
<tr>
<td>75%</td>
<td>1580.000000</td>
<td>0.045920</td>
<td>0.330247</td>
</tr>
<tr>
<td>max</td>
<td>7300.000000</td>
<td>25.106627</td>
<td>3.175077</td>
</tr>
</tbody>
</table>

Figure 30: Statistical summary of portfolio performance when RSI is applied on all tickers in HKEx given a 4-year period.

Figure 30 shows the results of applying the Relative Strength Index (RSI) strategy on all tickers listed in HKEx over the period of 2015-2019. Referring to the results, the average portfolio return generated by the RSI strategy is about -0.0759% and the standard deviation is approximately 1.118%, which is relatively low. The mean number of trading signals generated in a 4-year period is about 63.96.
Scenario 3  Given a ticker symbol and a fixed time span, we would like to study which indicator has the best performance.

- **Independent variable(s):** Strategy
- **Controlled variable(s):** Ticker = 0005.HK, Time period = 2017-01-01 to 2019-01-01

![Heat map](image1)

Figure 31: Heat map that illustrates how the number of trading signals changes for different strategies, given 0005.HK as the ticker and a fixed time range between 2017 and 2019.

![Heat map](image2)

Figure 32: Heat map that illustrates how the sharpe ratio varies for different strategies, given 0005.HK as the ticker and a fixed time range between 2017 and 2019.

From Figure 31, it could be inferred that Stochastic oscillator (STC oscillator) has the best performance out of the 9 indicators, as it has the highest sharpe ratio (i.e. return-to-risk ratio).

Observing both Figure 31 and 32, it is also interesting to see that the sharpe ratio of the strategy seems to have a positive relationship with the number of trading signals generated. While here it is assumed that transaction costs are negligible in the backtester,
in reality the number of trading signals could imply that higher transaction costs would be incurred and thus the net sharpe ratio would be lower.

**Scenario 4** Given a ticker symbol, we vary both the strategy and the time span to examine the correlation between the two.

- **Independent variable(s):** Strategy, time span (start date)
- **Controlled variable(s):** Ticker = 0005.HK

![Heat map that illustrates how the sharpe ratio varies for different strategies and time spans (2019-01-01 set as the fixed end date), given 0005.HK as the ticker.](image)

From the heat map shown in Figure 33, the following hypotheses could be inferred:

- Trend indicators (MACD crossover, MA crossover, Parabolic SAR) tend to perform relatively better in short-term (from 3 months to 1 year).
- Momentum indicators (Relative Strength Index (RSI), STC oscillator, Money Flow Index (MFI)) tend to improve as the time span increases, with Williams %R as an exception.
- Volume indicators (Chaikin Oscillator, Volume rate of change (ROC)) appear to be the worst-performing indicator among the three types.

Nevertheless, as the experiment was only carried out with one ticker, it will be essential to conduct experiments on a larger number of tickers (e.g. randomly sample 1000 tickers out of all tickers listed in different places) so as to empirically verify the above listed hypotheses. Alternatively, the hypotheses might only be true when certain assumptions have been made (e.g. the ticker belongs to the banking sector).
Scenario 5  We want to compare and contrast different strategies’ performance on Hong Kong stock tickers between the time periods 2017-2019 and 2019-2021.

- **Independent variable(s):** Strategy
- **Controlled variable(s):** Ticker = Set of all tickers in HKEx, Time span (2017-2019 and 2019-2021)

A total of 9 different technical analysis strategies were tested with all tickers in HKEx (1421 symbols). In the following, we would compare and analyse the heatmaps for each variable one by one.

Firstly, concerning the average **number of trading signals** for all tickers, the relative number of trading signals appears to remain unchanged regardless of the selected time period. As shown in Figures 34 and 35, in both cases, STC oscillator has the highest number of trading signals, and MA crossover has the lowest number.

![Figure 34: Heat map that illustrates the average number of trading signals for different strategies, given a fixed time range of 2017 to 2019.](image)

![Figure 35: Heat map that illustrates the average number of trading signals for different strategies, given a fixed time range of 2019 to 2021.](image)
Secondly, concerning the average **portfolio return** for all tickers, Figures 36 and 37 show that the strategies have almost reverse performance in the two periods. While Bollinger Bands and RSI have the best performances in the period of 2017-2019, they have the lowest returns in the period of 2019-2021. Similarly, the STC oscillator (which has the worst performance in 2017-2019) also exhibits the same pattern. This shows that the performance of technical indicators could be affected by macroeconomic cycles (such as expansion and recession), as 2019 was a turning point for Hong Kong’s economy. It would be important to first identify which phase of the cycle the stock market is in before analysing which strategy could guarantee higher portfolio returns.

Thirdly, concerning the average **sharpe ratio** for all tickers, Figures 38 and 39 show that the strategies have a consistent performance throughout the time period. As previously we have seen that the strategies have largely different performance in terms of portfolio return across the periods, this implies that the risk of holding the stocks has changed.
build a repository for teaching algorithmic trading final report

accordingly which made the sharpe ratio (i.e. return-to-risk ratio) constant throughout the time horizon. This also aligns with the principle of "risk-return trade-off" that says higher risk guarantees a probability of greater return.

figure 38: heat map that illustrates the average sharpe ratio for different strategies, given a fixed time range of 2017 to 2019.

figure 39: heat map that illustrates the average sharpe ratio for different strategies, given a fixed time range of 2019 to 2021.

7.1.1 summary

these 5 example scenarios indicate how the backtester, indicators and historical data in the repository could be used together to perform research and study on the relationship between different variables. we have studied the following characteristics of the technical analysis strategies:

- the sensitivity of MACD’s portfolio return with the time span (long-term vs short-term).
- the relationship between number of trading signals generated by MACD and the time span.
• The average performance of RSI applied on all tickers in HKEx given a fixed time range.
• The best strategy for a particular ticker (taken 0005.HK as an example)
• The correlation between different strategies and different time spans (taken 0005.HK as an example).
• The average performance of technical indicators across the time periods of 2017-2019 and 2019-2021, and how 2019 as a turning point in Hong Kong’s economy has influenced the stock market.

7.2 Integrated Strategies

In these experiments, we aim to demonstrate how we could make use of the classes and modules in the repository to build an algorithmic trading pipeline that takes inputs which examine the market in multiple perspectives.

To assure the objectivity of the experiment results, we have tested all integrated strategy models with the same set of tickers and the same date range of start_date = 2020-06-10 to end_date = 2021-03-03. The set of selected tickers is shown in Table 16. These 29 selected tickers are companies with the largest market capitalisation in each industry in Hong Kong.

7.2.1 Baseline Model

We have respectively used MACD crossover, Relative Strength Index (RSI) and Stochastic Oscillator (STC) as the technical analysis strategy to generate signals and apply the macroeconomic and sentiment filters afterwards. The output dataframe (which consists of trading signals) is then passed to the backtester, which could automatically compute the evaluation metrics including portfolio return and sharpe ratio.

As shown in Table 17, within the three baseline models, Baseline with MACD achieves the highest portfolio return and highest sharpe ratio for the selected set of tickers. On the other hand, the Baseline with STC model has the lowest portfolio return standard deviation (i.e. risk).

7.2.2 Trading Signal Prediction with LSTM

Moving on to the LSTM model, we have tested with both the single-feature model (that takes only price data as input) and multi-feature model (that takes microeconomic, macroeconomic, sentiment data as inputs) respectively.

Single-feature LSTM The single-feature model was trained with epochs = 100 and batch_size = 32, and learning_rate = 0.01.

Multi-feature LSTM The multi-feature model was trained with epochs = 100 and batch_size = 72, and learning_rate = 0.01.

Referring to Table 17, among the multi-feature LSTM models, M_LSTM with RSI achieves the highest portfolio return and highest sharpe ratio. On the other hand,
<table>
<thead>
<tr>
<th>Industry</th>
<th>Ticker</th>
<th>Company name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer Discretionary</td>
<td>00669</td>
<td>TECHTRONIC IND</td>
</tr>
<tr>
<td></td>
<td>00175</td>
<td>GEELY AUTO</td>
</tr>
<tr>
<td></td>
<td>01211</td>
<td>BYD COMPANY</td>
</tr>
<tr>
<td>Consumer Staples</td>
<td>02319</td>
<td>MENGNIU DAIRY</td>
</tr>
<tr>
<td></td>
<td>00168</td>
<td>CHINA RES BEER</td>
</tr>
<tr>
<td>Conglomerates</td>
<td>00001</td>
<td>CKH HOLDINGS</td>
</tr>
<tr>
<td></td>
<td>00004</td>
<td>WHARF HOLDINGS</td>
</tr>
<tr>
<td></td>
<td>00019</td>
<td>SWIRE PACIFIC A</td>
</tr>
<tr>
<td>Information Technology</td>
<td>00700</td>
<td>TENCENT</td>
</tr>
<tr>
<td>Properties &amp; Construction</td>
<td>00823</td>
<td>LINK REIT</td>
</tr>
<tr>
<td></td>
<td>00016</td>
<td>SHK PPT</td>
</tr>
<tr>
<td></td>
<td>01113</td>
<td>CK ASSET</td>
</tr>
<tr>
<td>Finance</td>
<td>1299</td>
<td>AIA</td>
</tr>
<tr>
<td></td>
<td>0005</td>
<td>HSBC HOLDINGS</td>
</tr>
<tr>
<td></td>
<td>0939</td>
<td>CCB</td>
</tr>
<tr>
<td>Utilities</td>
<td>0002</td>
<td>CLP HOLDINGS</td>
</tr>
<tr>
<td></td>
<td>0003</td>
<td>HK &amp; CHINA GAS</td>
</tr>
<tr>
<td></td>
<td>2688</td>
<td>ENN ENERGY</td>
</tr>
<tr>
<td>Telecommunications</td>
<td>0941</td>
<td>CHINA MOBILE</td>
</tr>
<tr>
<td></td>
<td>0762</td>
<td>CHINA UNICOM</td>
</tr>
<tr>
<td>Industrials</td>
<td>0968</td>
<td>XINYA SOLAR</td>
</tr>
<tr>
<td></td>
<td>0868</td>
<td>XINYI GLASS</td>
</tr>
<tr>
<td></td>
<td>2382</td>
<td>SUNNY OPTICAL</td>
</tr>
<tr>
<td>Materials</td>
<td>2899</td>
<td>ZIJIN MINING</td>
</tr>
<tr>
<td></td>
<td>1818</td>
<td>ZHAOJIN MINING</td>
</tr>
<tr>
<td></td>
<td>2689</td>
<td>ND PAPER</td>
</tr>
<tr>
<td>Energy</td>
<td>0883</td>
<td>CNOOC</td>
</tr>
<tr>
<td></td>
<td>0386</td>
<td>SINOPEC CORP</td>
</tr>
<tr>
<td></td>
<td>0857</td>
<td>PETROCHINA</td>
</tr>
</tbody>
</table>

Table 16: The list of selected tickers for testing the baseline model and LSTM models.
M_LSTM with MACD has the lower portfolio return standard deviation and smallest number of trading signals generated.

<table>
<thead>
<tr>
<th>Baseline model</th>
<th>ret</th>
<th>$\sigma_{\text{ret}}$</th>
<th>sharpe</th>
<th>$\sigma_{\text{sharpe}}$</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline with MACD</td>
<td>5.458%</td>
<td>12.157%</td>
<td>0.677</td>
<td>1.002</td>
<td>7.724</td>
</tr>
<tr>
<td>Baseline with RSI</td>
<td>2.998%</td>
<td>9.927%</td>
<td>0.543</td>
<td>1.226</td>
<td>14.931</td>
</tr>
<tr>
<td>Baseline with STC</td>
<td>3.782%</td>
<td>9.423%</td>
<td>0.604</td>
<td>0.946</td>
<td>34.586</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single-feature LSTM model</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S_LSTM</td>
<td>3.010%</td>
<td>14.757%</td>
<td>0.016</td>
<td>0.164</td>
<td>15.535</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-feature LSTM model</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>M_LSTM with MACD</td>
<td>1.837%</td>
<td>4.722%</td>
<td>0.266</td>
<td>1.300</td>
<td>4.143</td>
</tr>
<tr>
<td>M_LSTM with RSI</td>
<td>4.817%</td>
<td>14.802%</td>
<td>1.190</td>
<td>0.129</td>
<td>4.464</td>
</tr>
<tr>
<td>M_LSTM with STC</td>
<td>-2.718%</td>
<td>7.554%</td>
<td>-0.433</td>
<td>1.046</td>
<td>47.214</td>
</tr>
</tbody>
</table>

Max. 5.458% 14.802% 1.190 1.300 47.214
Min. -2.718% 4.722% -0.433 0.129 4.143
Mean 2.741% 10.477% 0.409 0.830 14.085

Table 17: Performance results of the integrated strategy models, including the baseline model, single-feature LSTM and multi-feature LSTM models. Note that $\text{ret}$ stands for portfolio return, $\text{sharpe}$ stands for sharpe ratio and $n$ refers to number of trade signals generated.

7.2.3 Summary

Overall, with respect to the selected set of tickers and date range, here are the major findings in the experiments with different integrated strategy models:

- Baseline models perform better than M_LSTM models on average, and is on par with the S_LSTM model.
- Considering sharpe ratio, the M_LSTM with RSI model achieves the highest sharpe ratio among all models, which is nearly double of what the baseline models could achieve.
- Although the RSI does not perform well in the baseline model, as an input to the LSTM model it achieves the best results among the three indicators. This shows that using an technical indicator as a machine learning input feature could be more advantageous than having it to generate signals based on quantitative rules.

We hypothesise that the reason why some indicators perform better when coupling with the LSTM while some becomes worse is contingent on their correlation with the macroeconomic data and sentiment labels. It might be the case that the RSI captures information or patterns in stock prices that could not be captured by other features (i.e.
macroeconomic data and sentiment label), and thus induces better performance when used as a machine learning input compared to other technical indicators.

8. CONCLUSION

To conclude, we have built a comprehensive, reader-friendly code and data repository that serves to teach students without any financial background to go from zero to hero in algorithmic trading. According to our research, a majority of tertiary institutions in Hong Kong currently does not have any established syllabus for algorithmic trading and research, especially for undergraduates. We advocate the initiative of launching an algorithmic trading course within the Department of Computer Science at HKU, as it aligns with the goals of the BASc(FinTech) degree and is a skill of high demand in the industry. We envision that students could learn how to translate financial concepts (and thus other domain knowledge) into code and algorithms through the course. This repository would assist students in achieving such a goal.

We have included the code implementations of conventional strategies (technical & fundamental analysis) as well as more advanced ones that involve the use of machine learning (e.g. the LSTM trade signal prediction model) in the repository. Alongside with the explanation and mathematical formulae featured on our documentation website, they serve to show students concrete examples of writing a strategy and the workflow of having it evaluated in a backtester. In addition, example code snippets of using the Interactive Brokers API also help students learn to execute the trade in a practical setting.

We have created a well-organised database that consists of microeconomic, macroeconomic and sentiment data. It is anticipated that this database will be useful for teaching and research purposes at the university. To enable maintenance of the database in the long run, we have also built pipelines for downloading and updating the data.

All in all, we have completed the goals outlined in the Project Plan and have successfully created an all-in-one pocket guide to algorithmic trading.

9. FUTURE WORKS

To improve the content on the website and the code in the repository, it would be essential to collect more feedback from the users so as to evaluate the ease in understanding the tutorials we have written. Additionally, we hope to enlarge the community of contributors so that the database and repository could be maintained in the long run. We also hope to add more examples (e.g. technical indicators) and add more Julia code snippets in the repository so as to enhance its comprehensiveness.
BIBLIOGRAPHY


