Final Report

Building a Repository for Teaching Algorithmic Trading

Supervisor: Dr. RuiBang Luo

Lee Kwan Young (3035347392)

April 19, 2021

Abstract

In this project, we built an open-source code and data repository for teaching algorithmic trading. For the first phase of the project, we built a database that contains microeconomic, macroeconomic, and sentimental data, which had been continuously updated till the end of the project period. Using the collected data, we created indicators from various analysis, which includes microeconomic analysis – technical and fundamental analysis, macroeconomic analysis, as well as sentiment analysis. In the second phase of the project, we integrated the created indicators together to predict the stock trends in Hong Kong and generate the trade signals. The performances of different strategies used for indicator integration were then evaluated by running a backtester. In addition, we connected our strategy with a real-time trading platform to trade automatically via Python code. To achieve the educational objectives of this project, we documented all the codes used in different strategies and analyses with detailed explanations on a website.

From technical skills in developing algorithmic trading strategies to basic financial concepts to understand, this repository shall enable the learners to acquire a deep understanding of algorithmic trading. Moreover, our website will serve as a complete step-by-step guide for algorithmic trading beginners. The repository and the documentation website are available at https://github.com/awoo424/algotrading and https://algo-trading.readthedocs.io/ respectively.

Acknowledgements

I would like to express my special thanks of gratitude to my project supervisor Dr Rui Bang Ruo for providing his invaluable support, guidance and comments throughout the project.

I would also like to thank my teammates Angel Chung and Snow Wu for the brilliant ideas and valuable information provided by them in their respective fields.

Contents

List of Figures

List of Tables

Note that all the figures and tables used in Part 1 (a) - Microeconomic Analysis and Part 1 (c) - Sentiment Analysis are created by the other teammates - Angel and Snow

1. Introduction

1.1 Project Background

Algorithmic trading, also referred to as “algo”, “robo” or “black-box” trading, 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). In other words, it is a combination of code and technical analysis that allows investors to abstract their trading goals and automate entering and exiting trades according to a predetermined set of parameters. As algo trading allows users to place the order accurately and instantly without adding human emotion, it is getting more popular nowadays. In fact, algo trading is already empowering a lot of organizations to optimize trade executions, and by now, it is accounted for around 90% of the total trading (Melin, 2017).

Despite its popularity, algo trading is still not approachable for numerous individual learners. The reason is that algo trading covers multiple disciplines that require a considerable degree of finance, statistics, and programming knowledge. Learning algorithmic trading is not an easy task, especially for an individual investor or student who is not familiar with either one of the disciplines.

Furthermore, the university currently does not have a course focusing purely on algo trading nor database for researching purposes.

Answering a need for an all-in-one repository that combines all the interdisciplinary data and code required for teaching algo trading, we dedicated our efforts to bring this to reality throughout these two semesters.

1.2 Project Motivation

The challenge of learning algo trading Having looked through the list of learning materials available online, including blog posts, online courses and websites like Investopedia, we summarized the problem of the existing materials below:

- The materials are scattered, and most of them only cover shallow contents.
- The materials focus only on one discipline, i.e. focus on either finance or programming, which make the learner difficult to get the complete picture of algo trading.
- Most of the materials are based on the US market, and hence they are irrelevant to the local background, i.e., the Hong Kong stock market.

Consequently, it takes time for learners to find various learning materials to pick up each concept in order to get a complete and comprehensive understanding of algo trading. Besides, the financial setting for each country is different, and hence different analyses or strategies should be applied. The above facts brought our team a mind for building a repository that contains all the interdisciplinary knowledge on algo trading, which could also align with the financial setting of Hong Kong.

The challenge of applying algo trading In the real world, there are various factors affecting the stock market and causing market fluctuation. However, a lot of existing approaches analyse the stock market with indicators from a single source (e.g. technical analysis). Therefore it would be advantageous to collect the data from various sources in order to look over the market from different perspectives.

Technical and fundamental indicators are the two standard indicators that are essential in predicting the market trends. At the same time, human factors such as public mood could also be an important factor. According to Zhang et al. (2011) and Bollen et al. (2011), there was a better prediction made for some stock market indices (e.g. Dow Jones Industrial Average (DJIA) and S&P 500) when the public mood dimensions derived from Twitter feeds was incorporated together with technical indicators. The use of sentiment indicators created from Tweets could help predict the direction of stock price movement with an accuracy of up to 82.7 percent (Rao and colleagues, 2012).

Apart from sentiment indicators, we hypothesise that indicators generated based on housing market data also play an important role in predicting the stock market. In fact, the properties and construction sector accounts for over 10% of the weighting in the Hang Seng Index (Hang Seng Indexes Company Limited, 2020), and thus could potentially become a source of volatility in the Hong Kong stock market. Nneji et al. (2013) have verified empirically that house prices are responsive to fluctuations in macroeconomic variables in certain regimes.

Stock movement prediction with multi-source features is certainly a powerful extension, and there are some papers experimenting on machine learning models that combines indicators from different sources. Deng et al. (2011) have used Multiple Kernel Learning (MKL), and Wu et al. (2012) have used other machine learning analysis, in order to integrate technical and sentiment indicators. However, there is still a need for more researches on stock movement prediction with features from more than two sources, as well as researches that are focused on the Hong Kong stock market.

2. Objectives

This project has educational and research objectives, which are further described below.

2.1 Educational Objectives

The educational purpose of this project is to combines all the interdisciplinary knowledge, including financial concepts, data science skills and algorithmic design techniques, that are required for teaching algorithmic trading in one repository. The contents of the repository will be organized in a clear and understandable manner, allowing the learners to have a thorough understanding of strategies and analyses applied in algo trading.

2.2 Research Objectives

The research objective of this project is to analyse and evaluate the Hong Kong stock market from different perspectives - microeconomic, macroeconomic, and sentimental. Within each perspective, indicators to analyse the financial market will be created. The indicators created from each analysis will be integrated together to predict the future stock price in Hong Kong. Moreover, a consolidated database that consists of the historical stock tick, property prices data, and social media data will be available for future research purposes at the University of Hong Kong.

3. Project Overview

This project is divided into two parts - Part 1 and Part 2 and the two parts were completed in Semester one and Semester two respectively. The works done in each part is illustrated in Figure 1.

Part 1 began with preparing the data sources required for building a trading strategy. We tried to look at the market from different aspects, and hence historical data from microeconomic, macroeconomic, and sentimental were collected. Moreover, the collected data was used to create indicators that analyse the financial market. Moving on to Part 2, experiments were carried out on algorithms that integrate different indicators created from Part 1. The algorithms were used to predict stock price movements of Hong Kong, and the performance was analysed with backtesters. Finally the algorithm used to predict the stock price trend was connected to a real-time trading platform. All the strategies and analyses were documented with detailed explanations on our website.

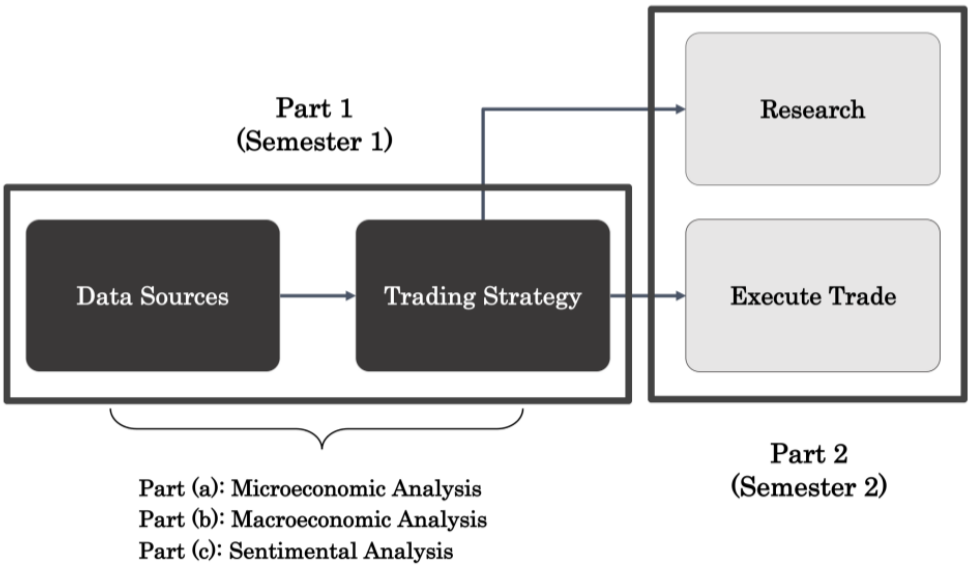


Figure 1: Flow chart showing the overall workflow of the project

4. Methodology

This section presents the workflow of the whole project.

4.1 Part 1 (a) - Microeconomic Analysis

Beginning with the microeconomic aspect, a historical stock tick database was created. Then, technical and fundamental indicators were created in order to study the past market action and interpret the performance of a company respectively.

4.1.1 Data Collection

Stock tick data A complete list of ticker symbols was first downloaded from the stock exchange official websites in order to collect the historical stock tick for each of the listed tickers. The list of ticker symbols is shown in Figure 2. This list was then loaded as an array into a Jupyter Notebook (denoted as ticker_list). Subsequently, the Yahoo! Finance API was called to import data for each ticker.

	symbol	name
0	0001	Cheung Kong (Holdings) Ltd
1	0002	CLP Holdings Ltd
2	0003	Hong Kong and China Gas Co. Ltd
3	0004	Wharf (Holdings) Ltd
4	0005	HSBC Holdings plc

Figure 2: 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).

Figure 3 shows an example of a stock tick data output csv file for a ticker. The Date column is formatted as YYYY-MM-DD, the Open, High, Low, and Close columns contain floating point numbers, and the Volume column contains an integer. If the data is 1-minute tick data, the first column would be Datetime, instead of Date.

	Date	Open	High	Low	Close	Volume
0	2000-01-04	33.037451	33.367817	32.376717	32.376717	3194413
1	2000-01-05	30.890019	31.385591	29.981523	30.146683	6058531
2	2000-01-06	30.394473	30.559633	28.081818	28.659994	10440480
3	2000-01-07	29.072963	29.403330	28.577392	29.238123	6049796
4	2000-01-10	30.229264	30.724838	29.485929	29.485929	5195405

Figure 3: An extract of *hkex_0001.csv* which is the output file storing the 1-day tick of the stock "0001" listed in HKEx.

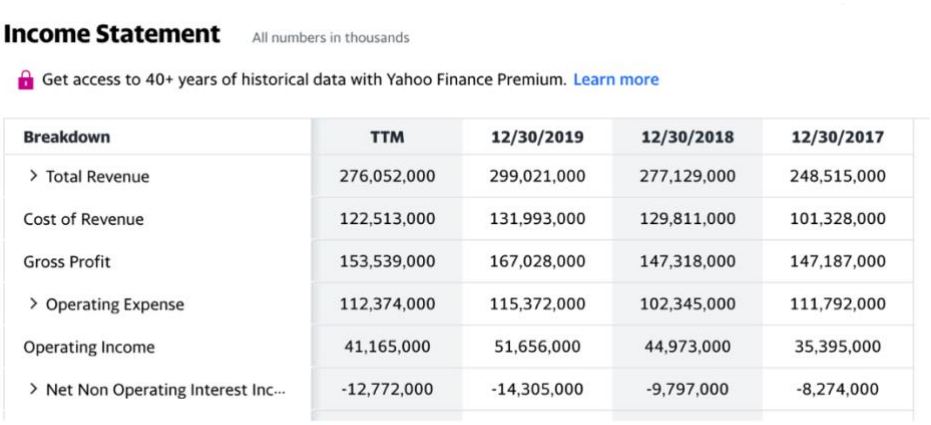
For each ticker, 1-day price data since the company's initial public offering (IPO) was collected, and also its 1-minute data since June 1, 2020 was collected. The 1-day and 1-minute stock tick data was constantly updated until the project's end. To ease the updating process, a code for automating the process of collecting the data between the last updated date and the most recent trading day was written.

The following command could be run in the terminal to update the stock tick data for a specific stock exchange:

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

For example, by running the command make update-hk-min, 1-minute price data for tickers listed on the Hong Kong Stock Exchange (HKEx) will be updated, and the progress message will be shown on the terminal.

Fundamentals data There is no open-source API that allows for direct crawling of fundamentals data, and hence the lxml library was used to parse the HTML document. The useful data was manually scraped from the tables on Yahoo! Finance web page, which is shown in Figure 4.



Income Statement All numbers in thousands

Get access to 40+ years of historical data with Yahoo Finance Premium. Learn more

Breakdown	TTM	12/30/2019	12/30/2018	12/30/2017
› Total Revenue	276,052,000	299,021,000	277,129,000	248,515,000
Cost of Revenue	122,513,000	131,993,000	129,811,000	101,328,000
Gross Profit	153,539,000	167,028,000	147,318,000	147,187,000
› Operating Expense	112,374,000	115,372,000	102,345,000	111,792,000
Operating Income	41,165,000	51,656,000	44,973,000	35,395,000
› Net Non Operating Interest Inc···	-12,772,000	-14,305,000	-9,797,000	-8,274,000

Figure 4: The table storing fundamentals data for a stock on the Yahoo! Finance page that is to be scraped using Python.

Real-time price Real-time price delivers up-to-the-minute information and are usually used for building intra-day trading and high-frequency trading strategy. In this project, the real-time price was not collected. Nevertheless, a code that uses Beautiful Soup library in Python was written and stored in the repository, so that it could be used for crawling the real-time price for a specific stock ticker symbol.

To get the HTML document that contains the real-time price, a GET request with the URL of Yahoo! Finance page could be sent. Then, Beautiful Soup could be used to obtain the element containing the current price from the HTML document.

```
<span class="Trsdu(0.3s) Fw(b) Fz(36px) Mb(-4px) D(ib)" data-reactid="32">
    56.000
</span>
```

For example, the price 56.000 shown in the above element could be extracted using the code below:

```
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
```

In order to obtain the real-time price for a list of tickers, the above code could be ran in a for loop. However, it's worth noting that the code may have to be updated if Yahoo! Finance changes its layout of the HTML content.

4.1.2 Technical Analysis

In total, 16 technical indicators were implemented in Python, and the list of indicators is shown in Table 1. The indicators are classified into 4 different categories, which are trend, momentum, length, and volatility. For each indicator, a class was built in technical-analysis_python/strategy directory and an example code showing how to import the class was written in a file named main_{indicator name}.py. Moreover, a backtester was built in order to analyse the performance of the strategies. When running the backtester, figures will be generated and stored in technical-analysis_python/figures directory. In addition, all absolute values including final portfolio value and the number of trade signals will be printed in the terminal.

Category	Indicator
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 1: List of technical indicators implemented in the repository.

4.1.4 Fundamental Analysis

Stock screening The three types of financial statements - income statement, balance sheet, and cash flow statement, compiled in the database were used to compute different financial ratios, which are shown in Table 2. The financial ratios are classified into 4 different categories, which are short-term solvency ratios, turnover ratios, financial leverage ratios, profitability ratios. All the financial ratios were used to analyse the performance of a company.

An investor could filter out the best performing tickers based on the relative performance of a company. This process is often referred to as "stock scanning", and the implementation of the whole process could be found in a Jupyter Notebook file named ratio-analysis.ipynb .

Category	Ratio(s)
Short-term solvency ratios	• Current ratio • Quick ratio • Cash ratio • Networking capital to current liabilities
Turnover ratios	• Average collection period • Inventory turnover ratios • Receivable turnover • Fixed asset turnover • Total asset turnover
Financial leverage ratios	• Total debt ratio • Debt to equity • Equity ratio • Long-term debt ratio • Times interest earned ratio
Profitability ratios	• Gross profit margin • Net profit margin • Return on assets (ROA) • Return on equity (ROE) • Earning per share (EPS)

Table 2: List of common financial ratios used for fundamental analysis.

Bankruptcy prediction Fundamentals data can be used not only for stock screening, but also for forecasting the bankruptcy of a company. Based on the Simple Analysis of Failure (SAF2002) model (Shirata, 2003), Altman (2013), and Beaver (1966), seven ratios were selected to be the input variables for training the bankruptcy prediction machine learning model.

The seven input variables are shown below:

X1 = working capital ÷ total assets

X2 = retained earnings ÷ total assets

X3 = Earnings before interest and taxes (EBIT) ÷ total assets

X4 = total equity (book) ÷ total assets

X5 = net income ÷ total assets

X6 = total liabilities ÷ total assets

X7 = cash flow from operation ÷ total liabilities

A labelled dataset compiled by the UCLA School of Law containing records from over 200 public companies in the United States was used to train the model. The companies that have gone bankrupt within three years and the companies that have survived were labelled with 0 and 1 respectively.

The dataset was first split into training and test datasets with the ratio of 7:3. Then, the dataset was trained into 4 different machine learning models, which are Support Vector Machine (SVM), Decision Tree, Random Forest, and K-nearest neighbours. The classification result was obtained by calculating the percentage of data with a correct prediction. The accuracies of the four models are summarized in Table 3.

Model	**1-year**	**3-year**
Support Vector Machine (SVM)	73%	65%
Decision Tree	69%	54%
Random Forest	88%	65%
K-Nearest Neighbour (KNN)	77%	50%

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

Referring to the classification result, we could conclude that Random Forest model performs the best when predicting the one-year bankruptcy of a company. On the other hand, both

SVM and Random Forest perform the best when predicting the three-year bankruptcy of a company.

4.2 Part 1(b) - Macroeconomic Analysis

Proceeding on to the macroeconomic aspect, the Hong Kong real estate market was analysed. Hong Kong residential market transactions data was collected from the websites of different estate agents. Then, the collected data was analysed in different analyses. Moreover, the data was used to study the interdependence between the real estate market and stock market in Hong Kong. All the codes for the macroeconomic analysis could be found in macroeconomic-analysis/ directory.

4.2.1 Data Collection

Hong Kong residential market transaction records Using web scraping APIs and open-source libraries, web scrapers were built to collect Hong Kong residential market transaction records from two real estate websites which are Centaline Property and Midland Realty.

Originally for Centaline Property, the English version of the Hong Kong residential market transaction data was collected. As the GET request with parameters of district id and registration period returns the whole HTML page, the useful data had to be extracted from the content by using BeautifulSoup. Figure 5 shows the data structure of transaction records extracted from Centaline Property. The data contains the transaction records from 02/01/2017 to 23/12/2020, with 10 basic features describing the property.

address	buildingAge	regDate	price	saleableArea	grossArea	upSaleableArea	upGrossArea	lasthold	gain
FLAT 7 26/F BLOCK A SMITHFIELD TERRACE	34	2020-12-23	5300000.0	262	357	20229	14846	3394	89
FLAT H 41/F THE FOREST HILLS	12	2020-12-23	7380000.0	439	627	16811	11770	2793	50
FLAT 5 (NO. 26) 1/F MAN YUE MANSION (BUILDING)	56	2020-12-23	3980000.0	480	-	8292	-	-	NaN
FLAT 12 30/F CHUN HONG HOUSE (BLOCK E) TIN MA ...	34	2020-12-23	4800000.0	-	461	-	10412	-	NaN
NO. 3 4/F GOLDEN PHOENIX BUILDING	55	2020-12-23	3350000.0	381	-	8793	-	7718	329

Figure 5: The data structure of transaction record (Centaline Property - English)

However, the Centaline Property website was renewed, and hence the original API that retrieves English data cannot be used anymore. Currently, only Chinese version of data is available on the website. Therefore, the API that returns the Chinese version of data was found and used to retrieve the data. The POST request with the body containing region type code and registration period was sent in order to retrieve the array of transaction records for

the specific registration period. The structure of transaction records (Chinese version) extracted from Centaline Property is shown in Figure 6. The data contains transaction records from 23/04/2018 to 12/04/2021 with 14 features describing the property.

address	bedroom	building	district	estate	flat	floor	grossArea	price	regDate	region	saleableArea	upGrossArea	upSaleableArea
渣華道3-5號	NaN	永光閣	北角	NaN	C室	11樓	468.0	5500000.0	2021-12-04	港島	296	11752.0	18581.0
欣景路8號	2.0	7座	寶琳	2期	E室	21樓	617.0	7220000.0	2021-12-04	九龍	441	11702.0	16372.0
福澤街38號	NaN	2座	大角咀	利奧坊·曦岸	D室	6樓	NaN	6019890.0	2021-12-04	九龍	257	NaN	23424.0
碧街44-48號	NaN	祥興大廈	油麻地	NaN	1室	8樓	NaN	5800000.0	2021-12-04	九龍	550	NaN	10545.0
福澤街38號	NaN	2座	大角咀	利奧坊·曦岸	F室	16樓	NaN	6367560.0	2021-12-04	九龍	257	NaN	24776.0

Figure 6: The data structure of transaction record (Centaline Property - Chinese)

For Midland Realty, sending a GET request with parameters of district id and registration period returns an array of transaction records for the specific registration period. The data structure of transaction records extracted from Midland Realty is shown in Figure 7 and 8. The data contains the transaction records from 07/01/2018 to 13/04/2021 with 20 features describing the house. Compared to the data from Centaline Property, the address is further divided into sub-features, and also there are some additional features such as floor level and the number of bedrooms.

region	subregion	district	estate	building	firstOpDate	floorL	bedroom	sittingroom	floor
Hong Kong Island	Eastern	Heng Fa Chuen (Chai Wan)	Heng Fa Chuen	Block 31	1988-03-19	L	2.0	NaN	NaN
New Territories	Sha Tin	Shatin	City One Shatin	Block 31	1983-08-13	M	2.0	1.0	NaN
Kowloon	Kowloon City	Hung Hom	Royal Peninsula	Block 1	2000-12-14	H	3.0	NaN	NaN
New Territories	Kwai Tsing	Tsing Yi	Cheung Fat Estate	Block 1 (King Fat House)	1989-09-01	H	NaN	NaN	NaN
New Territories	Yuen Long	Yuen Long	Grand Del Sol	Block 01	1997-12-05	H	3.0	NaN	NaN

Figure 7: The data structure of transaction record (Midland Realty) - Part 1.

flat	grossArea	saleableArea	price	regDate	lastRegDate	lastPrice	gain	lat	lon
6	597.0	498.0	8250000.0	2021-01-07	2003-01-17	1700000.0	385.29	22.278636	114.239373
D	395.0	327.0	5400000.0	2021-01-07	NaT	NaN	0.00	22.386003	114.204891
B	1251.0	952.0	21000000.0	2021-01-07	2008-01-18	9680000.0	116.94	22.304435	114.184404
21	629.0	485.0	3950000.0	2021-01-07	NaT	NaN	0.00	22.361853	114.103990
E	883.0	740.0	7830000.0	2021-01-07	1997-08-06	4953900.0	58.06	22.440296	114.034537

Figure 8: The data structure of transaction record (Midland Realty) - Part 2.

As the English version of transaction records from Centaline Property has fewer feature columns but have a greater number of records covering a longer period of time compared to that of Midland Realty, it was used for economic indicator analysis. On the other hand, the transaction records from Midland Realty have more feature columns compared to that of

Centaline Property. Therefore, it was used for transaction data analysis. Lastly for further research, the Chinese version of Centaline Property data was used for macroeconomic analysis of the Hong Kong stock market. The details of APIs used for macroeconomic data collection is provided in Appendix.

Economic indicator Economic indicators from the Census and Statistic Department website were collected to study the determinants of Hong Kong's housing prices. 8 different types of data were collected including population, unemployment, total import/export, GDP, and composite consumer price index. Data were manually downloaded from the website, cleaned and stored in csv format. The data contains economic indicators from 2016 to the present. Table 4 shows the file and data structure of economic indicators.

File Name	Data	Unit	Time Period
TABLE001.csv	Population	thousands	Half-yearly
TABLE006.csv	Unemployment rate (seasonally adjusted)	%	Monthly
	Unemployment rate (not seasonally adjusted)	%	
TABLE055.csv	Total export	HK$ million	Monthly
	Total import	HK$ million	
TABLE030.csv	GDP	HK$ million	Quarterly
	GDP per capita	HK$	Yearly
TABLE052.csv	Composite Consumer Price Indices	-	Monthly

Table 4: The file and data structure of economic indicators.

4.2.2 Data Pre-processing

Before analysing the collected data, the data was pre-processed. Firstly, some useful features were derived from the existing features. For example, building age was calculated from the first operating date of the building. Then, unmeaningful features and features with too many missing values were dropped. After getting all the feature columns ready, missing values were

handled by replacing NAN with a mean value of a feature. Lastly, categorical features such as region and district were label encoded.

4.2.3 Data Exploration

Economic indicator analysis The main focus of economic data analysis was to explore how the economic indicators affect the monthly average house price per saleable area in Hong Kong. The reason why the monthly average house price per saleable area was used is because there is no correlation between economic indicators and individual house prices in Hong Kong.

After calculating the monthly average house price per saleable area from the Centaline Property transaction records, it was joined with economic indicators by year and month. Then, both univariate analysis and bivariate analysis were carried out to analyse each feature and find the relationship between the features.

In univariate analysis, the distribution of the numerical features was examined by calling pandas Dataframe.describe() function. By calling this function, statistical summary such as mean, standard deviation, min, and max of a data frame can be viewed. For a better understanding of the statistics summary, seaborn.distplot() function was used to visualize the results with histograms. Figure 9 shows the code snippet of univariate analysis.

```
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 9: Code snippet of univariate analysis

In bivariate analysis, correlations between the features were studied. Using scatter plot and regression line, the relationship between two features were visualized. An example of a scatter plot with a regression line is shown in Figure 10. The regression line in the figure has a negative slope, indicating that the unemployment rate (seasonally adjusted) is negatively correlated to the monthly average price per saleable area.

Then, pandas.Dataframe.corr() and seaborn.heatmap() functions were used to compute a pairwise correlation of features and visualize the correlation matrix. Table 5 shows the correlation coefficient of economic indicators and the monthly average house price per saleable area in Hong Kong. According to the table below, GDP per capita, GDP, composite consumer price index, population, year, imports, month, and total exports are positively correlated to the

monthly average house price per saleable area in Hong Kong, while both seasonally adjusted unemployment rate and not seasonally adjusted unemployment rate are negatively correlated to the monthly average house price per saleable area in Hong Kong.

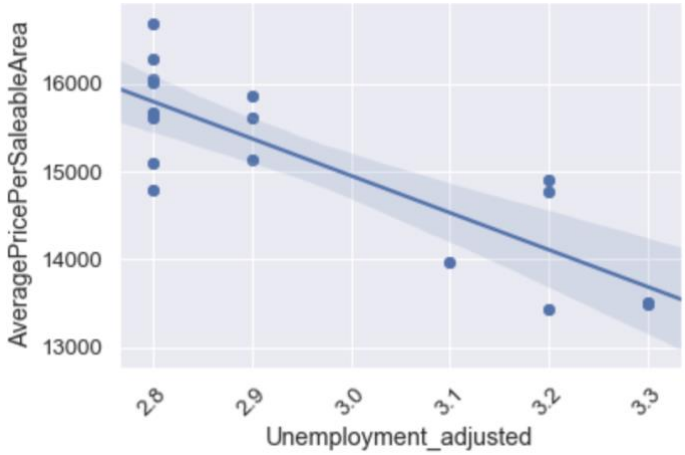


Figure 10: An example of a scatter plot with a regression line

Feature	Correlation coefficient
GDP per capita	0.82
GDP	0.70
Composite consumer price index	0.69
Population	0.68
Year	0.68
Imports	0.43
Month	0.38
Total exports	0.37
Unemployment rate (not seasonally adjusted)	-0.65
Unemployment rate (seasonally adjusted)	-0.84

Table 5: the correlation coefficient of economic indicators and the monthly average house price per saleable area in Hong Kong.

Transaction data analysis The objective of transaction data analysis was to examine the relationship between features describing the house and the individual house prices of Hong Kong. Using the same method as for economic indicator analysis, univariate analysis and bivariate analysis were carried out. As mentioned above, the transaction records from Midland Realty was used for this analysis.

In univariate analysis, the distribution of Hong Kong's house price was examined. The house price of Hong Kong has a mean of 9 million HKD and a standard deviation of 13 million HKD. The skewness and kurtosis were 26.9 and 1526.4 respectively, showing that the house price of Hong Kong is skewed positively to a very high degree. In order to get a better result for the bivariate analysis, outliers were removed by using standard deviation.

In bivariate analysis, the correlation coefficient between the features describing the house and the house price was computed, and 7 features with the highest correlation were selected. The correlation heatmap of the top 7 features selected is shown in Figure 11. According to the figure, the house price in Hong Kong has 1) a strong positive correlation with saleable area; 2) a moderate positive correlation with last transaction price; 3) a moderate positive correlation with gross area; 4) a moderate positive correlation with number of bedrooms; 5) a weak positive correlation with floor; 6) a weak negative correlation with region; and 7) a weak negative correlation with building age.

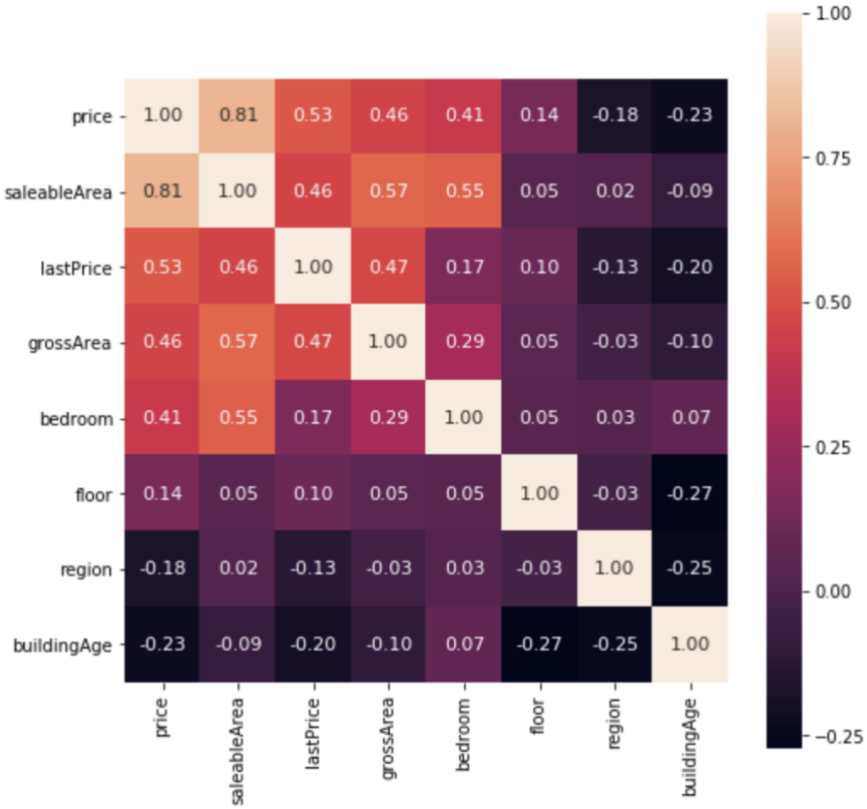


Figure 11: The correlation heatmap of the top 7 features selected

Macroeconomic analysis of stock market The objective of macroeconomic analysis was to examine how the macroeconomic indicators affect the stock prices in Hong Kong. In this analysis, the Chinese version of data collected from Centaline Property was used. Also, monthly Hang Seng Index (HSI) was used to represent the Hong Kong stock market.

First of all, the monthly average house price was calculated. Then, it was joined with other macroeconomic indicators and monthly Hang Seng Index data frames. Then, same as the above two analyses, bivariate analysis was used to study the interdependence between macroeconomic indicators and the monthly Hang Seng Index.

To make the research more interesting, the whole data frame was split into two parts:

1. Data before the pandemics (April 2018 to December 2019)
2. Data after the pandemics (January 2020 to March 2021)

Before the pandemics, it was found that monthly average house price has the highest positive correlation with HSI, whereas GDP has the highest negative correlation with HSI. Other macroeconomic indicators also showed weak correlations with HSI. Figure 12 shows the correlation heatmap of macroeconomic indicators and HSI before the pandemics. Usually, GDP is positively correlated to the stock trend. The reason why it is showing a negative correlation for this period might be because Hong Kong's economy experienced an annual contraction in 2019.

On the other hand, the result was completely different for the data collected after the pandemics. This time, GDP has the highest positive correlation with HSI. The correlation coefficient between the monthly average house price and HSI has dropped. Moreover, the correlation coefficient between the unemployment rate and HSI has changed from negative to positive. Having a positive correlation between the unemployment rate and HSI is an unexpected result, and it might due to the effect of the pandemics. In fact, the pandemics caused the surge in the unemployment rate.

There were also changes in correlation coefficient for other macro indicators as well. For example, the interdependence between import/export and HSI has decreased. The correlation heatmap of macroeconomic indicators and HSI after the pandemics is shown is Figure 13.

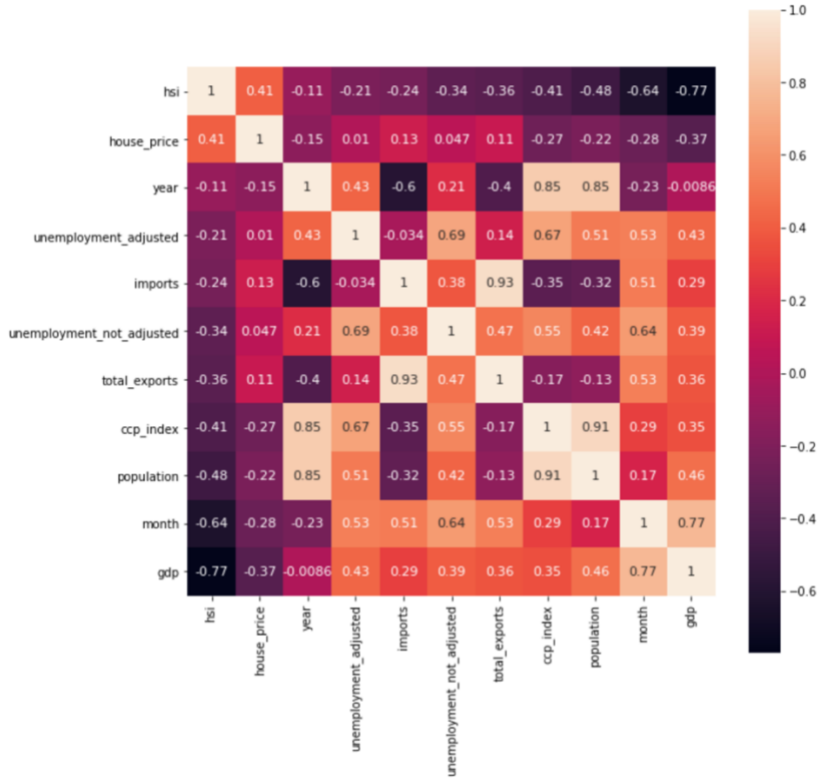


Figure 12: The correlation heatmap of macroeconomic indicators and HSI before the pandemics.

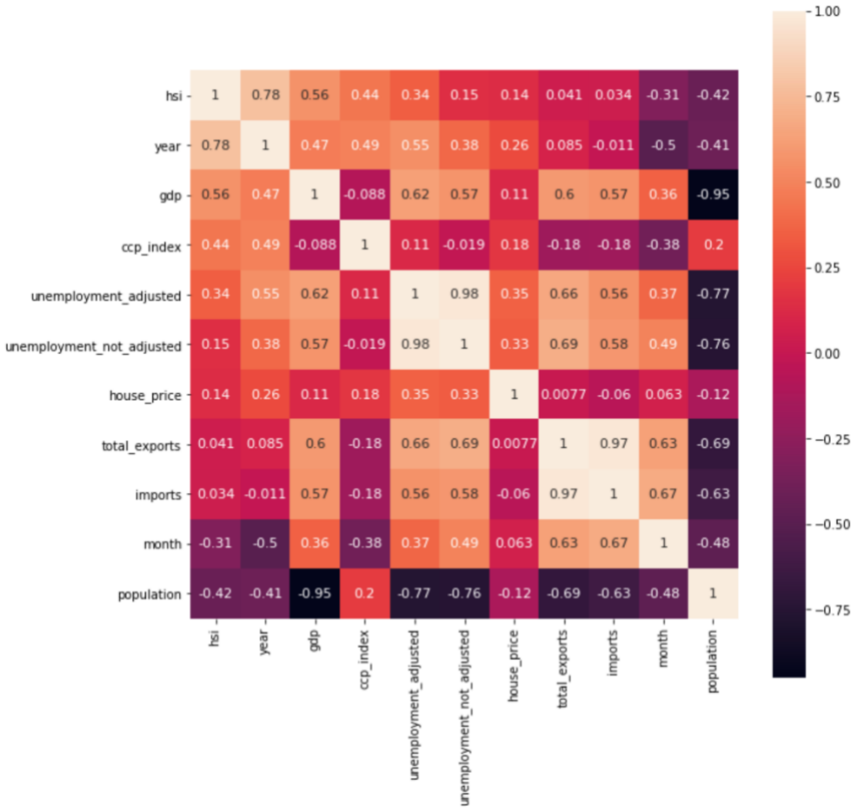


Figure 13: The correlation heatmap of macroeconomic indicators and HSI after the pandemics.

4.2.4 Property Price Prediction

Based on the result of transaction data analysis, house price prediction models were built. The data was split with the ratio of 8:2 and was used to train and test the model. The input variables were the top 7 features selected from the analysis, and the output feature was the house price. Before training the model, log transformation was used to normalize the highly skewed price data. In this way, the dynamic range of Hong Kong's house prices can be reduced.

In total, 4 different types of models were built - XGBoost, Lasso, Random Forest, and Linear Regression. Then, the performance of each model was evaluated by root mean square log error (RMSLE). The reason why RMSLE was used is that the price values are too big, and RMSLE prevents penalizing large differences between actual and predicted prices. Table 6 shows the RMSLE value of the models.

Model	**RMSLE**	
	Train	**Test**
XGBoost	0.1608	0.1645
Lasso	0.2640	0.2652
Random Forest	0.3077	0.3071
Linear Regression	0.2630	0.2643

Table 6: RMSLE values of the housing price prediction models.

As shown in the evaluation table, the model with the best performance is XGBoost. XGBoost uses a more accurate implementation of gradient boosting algorithm and optimised regularisation, and hence, it gives a better result than other models. However, in this case, the result shows that the model is overfitting the train data. Figure 14 shows the graph of the actual and predicted house price for XGBoost. It tends to give less error for the range of price which has a larger number of training data.

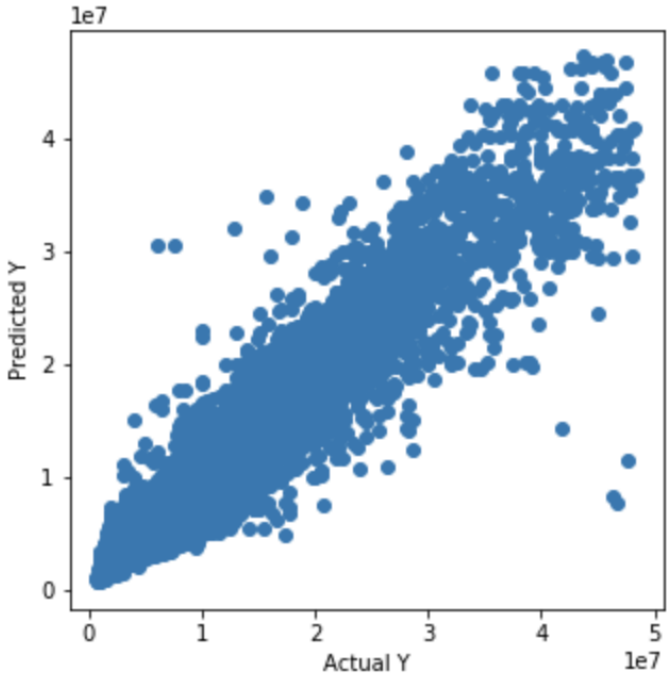


Figure 14: The graph of actual and predicted house price for XGBoost

4.3 Part 1(c) - Sentiment Analysis

Moving on to the sentimental analysis, data was collected from various social media websites and news articles on the stock market using text mining technologies. The extracted data was used to analyse the local market sentiment.

4.3.1 Financial News Headline Collection and Update

Different websites have different front-end architecture design, and hence the code for scraping data had to be specifically designed for each website using Beautiful Soup library.

Starting with the US market, financial news for listed companies in New York Stock Exchange (NASDAQ) and New York Stock Exchange (NYSE) were collected from finviz.com, which is a browser-based stock market that allows users to find the most up-to-date financial news from various sources such as Yahoo Finance, Accesswire, and Newsfile.

To collect the data, a GET request with finviz_url was sent to obtain all the relevant financial news for a particular ticker. Then, headlines stored in the div tag block with the id name of news-table were extracted and stored in corresponding csv file. Figure 15 shows a code function that was used for getting the news from finviz.com.

```
finviz_url = 'https://finviz.com/quote.ashx?t='
# get news from finviz.com
def get_finviz(ticker,day):
    try:
        url = finviz_url + ticker
        req = Request(url=url,headers={'user-agent': 'my-app/0.0.1'})
        resp = urlopen(req)
        html = BeautifulSoup(resp, features="lxml")
        news_table = html.find(id='news-table')
        news_tables[ticker] = news_table
        df = news_tables[ticker]
        path=os.path.join(dir_name,'data-news/data-finviz/'+'data-'+ticker+'-finviz.csv')
        with open(path,'w') as f:
            print(ticker)
            writer = csv.writer(f)
            for info in df.findAll('tr'):
                text=info.a.get_text()
                date_scrape= info.td.text.split()
                if(len(date_scrape)==1):
                    time=date_scrape[0]
                else:
                    date= date_scrape[0]
                    time=date_scrape[1]
                    news_time_str= date+" "+time;

                date_time_obj = datetime.datetime.strptime(news_time_str, '%b-%d-%y %I:%M%p')
                date_time=date_time_obj.strftime('%Y-%m-%d')

                if(datetime.datetime.now()-date_time_obj).days<=day:
                    print(date_time)
                    print(text)
                    writer.writerow([date_time,text])

    except Exception as e:
        print(e)
        pass
```

Figure 15: Code function for updating news from finviz.com

For the Hong Kong market, aastocks.com was used to gather the financial news headlines related to the listed companies. This website has been one of Hong Kong's top financial information platforms. It provides real-time data on Hong Kong stocks that can be used to analyse the sentiment of a company, and also the local market trends.

Similarly, a GET request with a URL was sent to obtain all the relevant financial news for a particular ticker. For this website, new headlines are not stored in a table format, but instead, stored in separate div blocks. Therefore, the collected news headlines were joined together as a table first, then stored in the corresponding csv file. Figure 16 shows a code function that was used for updating the news from aastocks.com.

```
prefix_url='http://www.aastocks.com/en/stocks/analysis/stock-aafn/'
postfix_url='/0/all/1'
# get news from aastock.com
def get_news_aastock(ticker,day):
    try:
        fill_ticker=ticker.zfill(5)
        url=prefix_url+fill_ticker+postfix_url
        print(url)
        req = Request(url=url,headers={'user-agent': 'my-app/0.0.1'})
        resp = urlopen(req)
        html = BeautifulSoup(resp, features="lxml")
        dates=html.findAll("div", {"class": "inline_block"})
        news=html.findAll("div", {"class": "newshead4"})
        idx=0
        path=os.path.join(dir_name,'data-news/data-aastock/'+'data-'+ticker+'-aastock.csv')
        with open(path,'w') as f:
            writer = csv.writer(f)
            for i in dates:
                if "/" in str(i.get_text()):
                    date=str(i.get_text())
                    if "Release Time" in date:
                        date=date[13:23]
                    else:
                        date=str(date[:10])
                        text=news[idx].get_text()
                        date_time_obj = datetime.datetime.strptime(date, '%Y/%m/%d')
                        date_time=date_time_obj.strftime('%Y-%m-%d')
                        if(datetime.datetime.now()-date_time_obj).days<=day:
                            category=hkex.loc[ticker]['Category']
                            print(text)
                            writer.writerow([date_time,text])
                        idx+=1
    except Exception as e:
        print(e)
        pass
```

Figure 16: a code function that was used for updating the news from aastocks.com

4.3.2 Tweets Collection and Update

The Tweepy API was used to collect and update the Tweet data. Relevant tweets for the listed companies in NASDAQ and NYSE were extracted and stored. To obtain the tweets that hashtag a particular stock, the Tweepy API with that particular company's ticker symbol was sent. Then, the retrieved tweets and dates were gathered and appended to the corresponding csv file. Figure 17 shows a code function that was used for updating the tweets.

```
# get data from  the cashtag
def get_tagtweets(name,day):
    allhashtag=[]

    hashtags=tweepy.Cursor(api.search,q=name,lang='en',tweet_mode='extended').items(300)
    outttags=[]
    try:
        path=os.path.join(dir_name,'data-tweets/')
        with open(path+'data-'+name+'-tweets.csv','a') as f:
            writer = csv.writer(f)
            for status in hashtags:
                if(datetime.datetime.now()-status.created_at).days<=day:
                    tweet_text = status.full_text.encode("utf-8")
                    dates=str(status.created_at)[:10]

                    print(dates)
                    print(tweet_text)
                    writer.writerow([dates,tweet_text])
            sleep(2000)
    except:
        print("error")
        pass
```

Figure 17: A code function that was used for updating the tweets.

4.3.3 Data Pre-processing

Before training the model, data files from the data-news directory were combined to create a single dataframe that contains both Hong Kong and US news for the stock trend analysis. Similarly, the data files in the data-tweets directory were also combined to create a dataframe for the Twitter sentiment analysis. Figure 18 shows the file structure of sentiment analysis data in the repository.

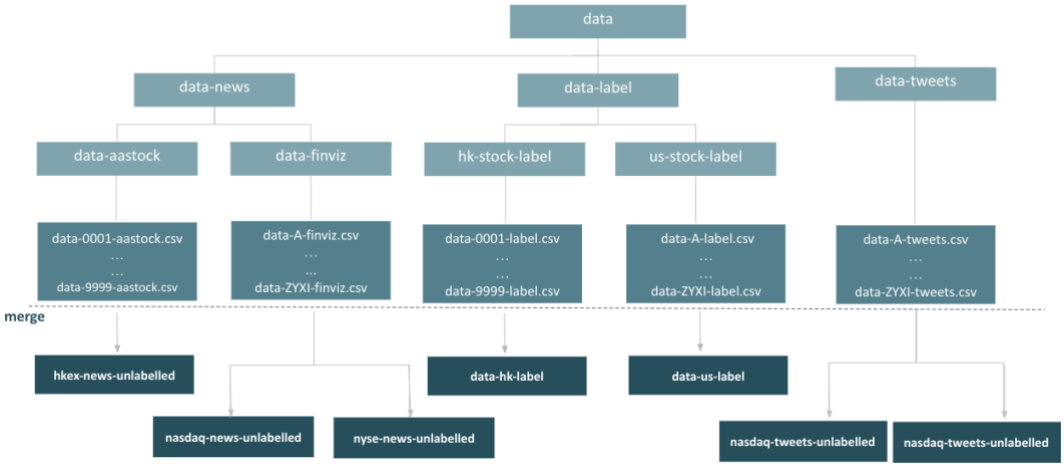


Figure 18: File structure of sentiment analysis data in the repository.

4.3.4 Machine Learning Analysis

VADER unsupervised learning VADER model (Hutto and Gilbert, 2014) is a model that is used for analysing the social media text and identifying the market sentiment based on certain rules. To verify the performance of the model, an experiment was done on the model by feeding a well-labelled dataset of 5000 positive tweets and 5000 negative tweets from the NLTK library. The model had a high accuracy with the given dataset, and hence it was used to label our own dataset of new headlines and tweets. Figure 19 summarises the steps designed for Vader Training.

For the labels, the VADER Compound score (Hutto and Gilbert, 2014) was used, which is a metric that sums up all the lexicon ratings for each tweet. The score is normalised between -1 and +1, where -1 represents the most extreme negative and +1 represents the most extreme positive. Based on the score and the rules below, the sentiment labels were generated.

- **Positive sentiment** (= 2): compound score ≥ 0.01
- **Neutral sentiment** (= 1): compound score > −0.01
- **Negative sentiment** (= 0): compound score ≤ −0.01

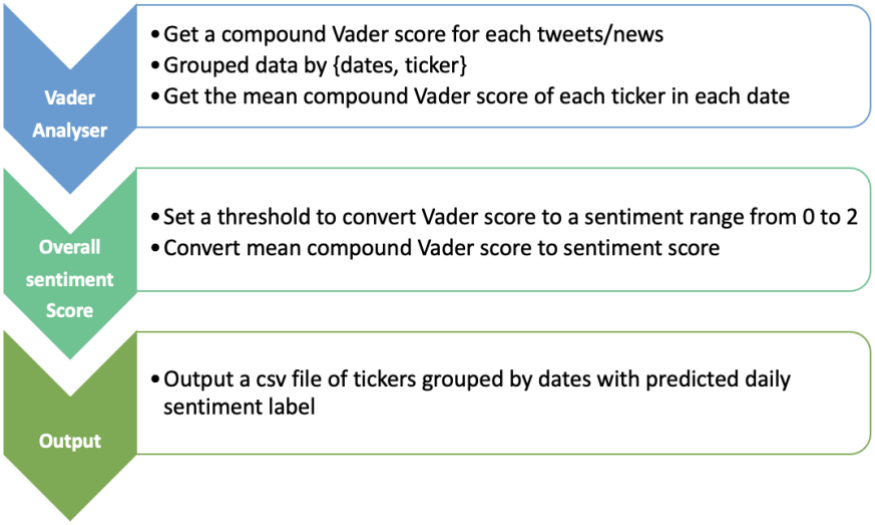


Figure 19: High-level workflow of Vader training.

BERT Machine Learning Model Proceeding on, the Bidirectional Encoder Representations from Transformers (BERT) model and BERT tokenizer were used to analyse the daily sentiment of tweets and news headlines and get the sentiment labels (Devlin et al., 2018). BERT is a model for pre-training natural language representation that is developed by Google. A sample of tweet sentiment analysis output is shown in Figure 20.

dates	ticker	compound_vader_score	vader_label	tweets	bert_label	actual_label
...	...	...	...	...	...	...
2021-01-08	ZS	0.178829	Positive	b'@sentivcapital Same here, $AMRN was my bigge...	Negative	Neutral
	ZSAN	0.087868	Positive	b'@sentivcapital @juliaskripkaser @buysidebio ...	Neutral	Positive
	ZUMZ	0.000000	Neutral	b"$ARPO Cool basic science + some clinical val...	Neutral	Positive
	ZYNE	0.000000	Neutral	b"FREE TRIAL\nSign up and be ready to receive ...	Negative	Neutral
	ZYXI	0.074583	Positive	b'$ARPO $MYAV $ERAGE is 0.982 https://t.co/Beu...	Negative	Negative

Figure 20: Sample of tweets sentiment analysis output for NASDAQ from 2020-12-28 to 2021-01-08.

4.4 Part 2 – Integration Strategy and Trade Execution

In part 2, we integrated all the parts together based on the results and analyses made in Part 1. In addition, a code for executing the paper trade via calling the Interactive Broker's API was written.

4.4.1 Stock Movement Prediction with Multi-source Features

Upon finish creating indicators from the above three subparts – microeconomic, macroeconomic and sentiment analysis, experiments were carried out in order to integrate those indicators together to predict the stock price movement of Hong Kong. In total, three models were built and backtested.

Feature selection From microeconomic analysis, Moving Average Convergence Divergence was selected, as it is the most popularly used technical indicator which is proved to give an efficient and reliable analysis. From macroeconomic analysis, GDP, unemployment rate, and monthly average house price were selected. The reason for this selection is that the three indicators showed relatively high correlation to HSI, and also they are the most accessible macroeconomic indicators. Lastly from sentiment analysis, the sentiment labels generated from financial news was used. This is because there was a lack of tweets that were relevant to Hong Kong tickers.

Baseline model A baseline model that integrates all the seven features was built and stored in integrated-strategy/baseline.py. In this model, the trading signals are first generated by applying MACD crossover strategy. The signals are then filtered by macroeconomic data to remove the bias. Again, the signals are filtered by sentiment labels. Finally, the filtered signals are used to backtest the baseline model. The detailed implementation of the baseline strategy is shown in Algorithm 1.

Algorithm 1: Baseline strategy

Data: Ticker list, Price data, Macro data, Sentiment data
Result: File containing backtest result

```
Import modules;
for ticker in Ticker list do
    df ← Load price data;
    start_date, end_date ← Select time range for trading;
    // Technical analysis
    df ← price data between start_date and end_date;
    signals ← Apply MACD crossover strategy and generate signal;
    // Macroeconomic analysis
    Calculate ticker's sensitivity to macro data;
    Append signals with macro data;
    signals ← Calculate adjusting factor and update the signal;
    // Sentiment analysis
    if sentiment label contrasting with buy/sell signals then
        signals ← Update the signal;
    // Backtest and evaluate
    Backtest with signals;
```

Besides the baseline model, Long Short-Term Memory (LSTM) models were implemented for further experiments. LSTM learns the contextual information that is required for making a prediction, and therefore it is a widely used mechanism for time-series forecasting, especially for stock price prediction.

Single feature LSTM model To start with, only price ('Close') was used as the input feature to train the LSTM model. The code implementation of the single feature LSTM model could be found in integrated-strategy/LSTM-train_price-only.py.

First, the price data was loaded and normalized with minmax scaler. Then, the data was split into train and test datasets based on the train to test ratio and the sequence length. The train and test datasets were converted to tensor before feeding to the model. After training the model, the predictions were made, which were inversed in order to get the real price value. To evaluate the model performance, the root mean square error (RMSE) was calculated. With the help of RMSE, the model was optimised. The less the error, the better the performance of the model. The hyperparameter setting for the final model is shown below:

- input_dim = 1
- hidden_dim = 32
- num_layers = 2
- output_dim = 1
- num_epochs = 100
- learning_rate = 0.01

Based on the actual price and the predicted price, the buy/sell signals were generated. There are two rules in generating the signals:

1. Generate signals by trend:
   - If the price trends for both the actual and the predicted price are the same, then the signal is 0, which indicates neutral.
   - Else if the price trend of predicted price is negative and the price trend of actual price is positive, it indicates that the current price is overvalued, therefore the signal is -1 (sell).
   - Else if the price trend of predicted price is positive and the price trend of actual price is negative, it indicates that the current price is undervalued, therefore the signal is 1 (buy).
2. Generate signals by abs price:
   - If the absolute price difference between the actual price and the predicted price is less than 1.0, then the signal is 0 (neutral).
   - Else if the predicted price is larger than the actual price, it indicates that the current price is overvalued, therefore the signal is -1 (sell).
   - Else if the predicted price is less than the actual price, it indicates that the current price is undervalued, therefore the signal is 1 (buy).

Finally, the generated signals were stored in csv file and used to backtest the LSTM model.

Multi-feature LSTM model The procedure of training multi-feature LSTM model is similar to that of training single feature LSTM model. However, in multi-feature LSTM model, seven different input variables were fed into the model to predict the stock price and generate the signals. The input variables are 'Close', 'technical_signal', 'GDP', 'Unemployment rate', 'Property price', 'Vader label', and 'Textblob label'. In addition, batching was implemented so that the model could be trained with different batch size. The code implementation of the multi-feature LSTM model could be found in integrated-strategy/LSTM-train_wrapper.py.

During model optimisation, the model was evaluated by RMSE. The hyperparameters were adjusted to get a better result, and the following are the final adjusted hyperparameter:

- input_dim = 7
- hidden_dim = 64
- num_layers = 4
- output_dim = 7
- num_epochs = 100
- learning_rate = 0.01
- batch_size = 72

With the above hyperparameter setting, the model was trained and the predictions were made. By comparing the prediction and the actual price trend, the buy/sell signals were generated and stored in csv file, which were in turn used to backtest the multi-feature LSTM model.

Performance Analysis In total, there were four different integration strategies used to generate the buy/sell signals:

1. Baseline Strategy
2. Single feature LSTM model – by trend
3. Single feature LSTM model – by abs price
4. Multi-feature LSTM model

In order to evaluate the performance of the strategies, 28 tickers were selected (based on weighting in HSI and market capitalization) from 11 different industries. Each ticker was used to generate the signals with the above strategies. Then, the signals were backtested within the time rage between 10 June 2020 to 3 March 2021, with an initial capital of 100,000. The performance was evaluated with three evaluation metrics - Portfolio return, Sharpe ratio and Compound Annual Growth Rate (CAGR).

The results were analysed based on:

1. Portfolio return:

The tickers were grouped according to their industry and portfolio return. Figure 21 shows the group of tickers with poor portfolio returns. In general, all the strategies that we built did not work well for the tickers from Energy, Material, Conglomerates, Consumer Staples, and Telecommunication industry.

In more detail, it showed that the tickers from Energy and Material industry have almost zero portfolio return, whereas the tickers from Conglomerates, Consumer Staples, and Telecommunication industry have relatively more dynamics in the portfolio return. Moreover, it was found that the baseline strategy tends to work better with this group of tickers, whereas the single feature LSTM model (by price trend) tends to give negative portfolio return, with a few exceptions (eg. 0168.HK).

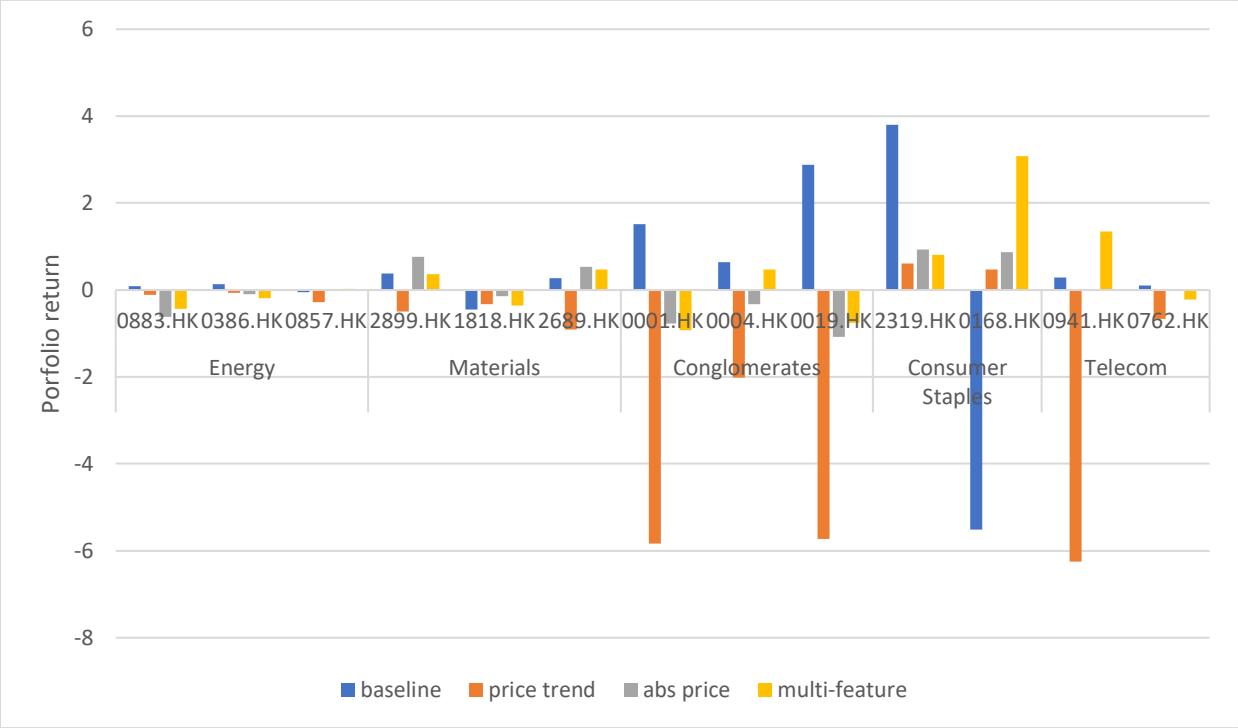


Figure 21. The graph showing the group of tickers with poor portfolio return.

The second group of tickers is the tickers with moderate portfolio return, and they are shown in Figure 22. The second group of tickers includes the tickers from Property and Construction, Financial and Utilities industry.

This group usually had portfolio returns of around -10 to 15. It was found that for Properties and Construction industry, the baseline strategy tends to give a positive portfolio return. For Utilities industry, it was shown that the single feature LSTM model (by price trend) tends to give a positive portfolio return. However, for Financial industry, there was no pattern in portfolio return.

The last group is the tickers with good portfolio return, and is shown in Figure 23. This group consists of tickers from Industrials, Consumer Discretionary and Information Technology industry.

This group gave a maximum portfolio return of 72.3 and a minimum of -21.2. Although there are few results giving negative portfolio returns, most of them gave a positive portfolio return. Especially for Information Technology industry, all strategies worked well to give high portfolio returns except for the multi-feature LSTM model strategy.

In general, all the strategies tend to give positive portfolio returns for the tickers from these three industries, except for some cases when the single feature LSTM model (by price trend) and the multi-feature LSTM model strategies give negative portfolio returns.

Figure 22. The graph showing the group of tickers with moderate portfolio return.

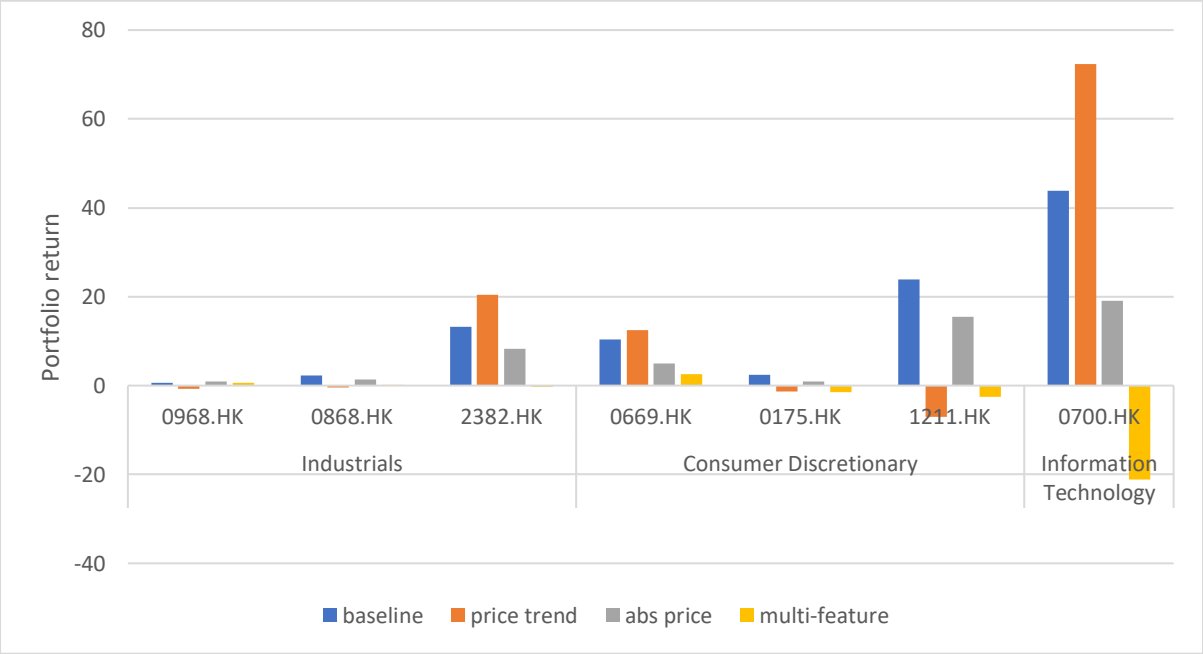


Figure 23. The graph showing the group of tickers with good portfolio return.

2. Average portfolio return, Sharpe ratio, CAGR and trade signal number

To have a look at the performance of each strategy, the average portfolio return, Sharpe ratio, CAGR, and trade signal number were calculated. The averaged result for each strategy is shown in Table 7.

Focusing on the average portfolio return and CAGR, the first three strategies have positive portfolio returns. Among all the strategies, the baseline strategy has the best portfolio return, followed by single feature LSTM and multi-feature LSTM.

Moving on to the average Sharpe ratio, it was shown that all the strategies have an average Sharpe ratio that are less than 1. It means that all the strategies are sub-optimal with a low degree of expected return for a relatively low amount of risk.

Lastly, for the number of trade signal, the table shows that the multi-feature LSTM model generates larger trade signals compared to other strategies. This may be because the multi-feature LSTM model is more sensitive to the input features, and hence more signals are generated. However, more number of signals does not mean that it gives a good result. As the multi-feature LSTM model is giving a negative portfolio return and CAGR, it indicates that signals generated are not accurate.

Strategy	Average Portfolio return	Average Sharpe ratio	Average CAGR	Trade signal Number
Baseline Strategy	4.00	0.65	0.038	7.6
Single feature LSTM by trend	3.01	0.02	0.004	15.6
Single feature LSTM by abs price	1.84	0.27	0.017	4.1
Multi-feature LSTM	-0.51	0.03	-0.005	46.5

Table 7: The averaged result for each strategy

In conclusion, the baseline strategy works the best among all the strategies. The reason might be because it first generated the signal based on the technical indicators, then tried to remove bias and reduce inaccuracy by filtering with macroeconomic and sentiment indicators. And the fact that the single feature LSTM model works better than the multi-feature LSTM model tells us that it might be not a good idea to feed all the indicators together to generate the signals.

4.4.2 Execution of Trade

To simulated trades, but without risking real money, we tried to create a paper trade account and explore the ways of executing the paper trade. Furthermore, we implemented a paper trade execution code that applies our own trading strategy.

Paper Trading with Interactive Brokers Interactive Brokers (IB) provides Python native APIs that enable users to execute the trades automatically via Python code. A simple instruction for setting up a connection to IB Trader Work Station (TWS) was written on the documentation website. Moreover, basic codes required for executing a paper trade were written and stored in paper-trading/ directory. The files in the directory include the codes for:

1. Establishing an API connection
2. Creating basic contracts
3. Requesting streaming market data
4. Requesting historical market data
5. Managing orders
6. Requesting account summary

Apply daily trading strategy to real-time trading The trading strategy that was built in Part 2(a), together with Interactive Brokers APIs were used to execute a paper trade. The code implementation could be found in integrated-strategy/daily_trading_order.py.

In daily_trading_order.py, a connection IB TWS is made. Then, a daily trading strategy is run in order to capture the buy/sell signals. Based on today's signal, the Contract object is created and an order is sent. The detailed algorithm of the daily trading strategy is shown is Algorithm 2.

Algorithm 2: Daily trading strategy

Data: Ticker, Macro data
Result: File containing daily signal

Import modules;
Collect daily news data of *Ticker*;
df ← Collect individual sentiment label;
df ← Collect daily price data of *Ticker*;
df ← Join with macroeconomic data;
df ← Normalise *df* with Minmax scaler;
model ← Load the saved multi-feature LSTM model;
Predict the daily price using *model*;
Generate *signals*;
Save *signals* in csv file

4.5 Documentation Website

A website was built with Sphinx to document all the strategies and the analysis implemented throughout the project. The website is hosted on Readthedocs, and the screenshot of the documentation webpage is shown in Figure 24.

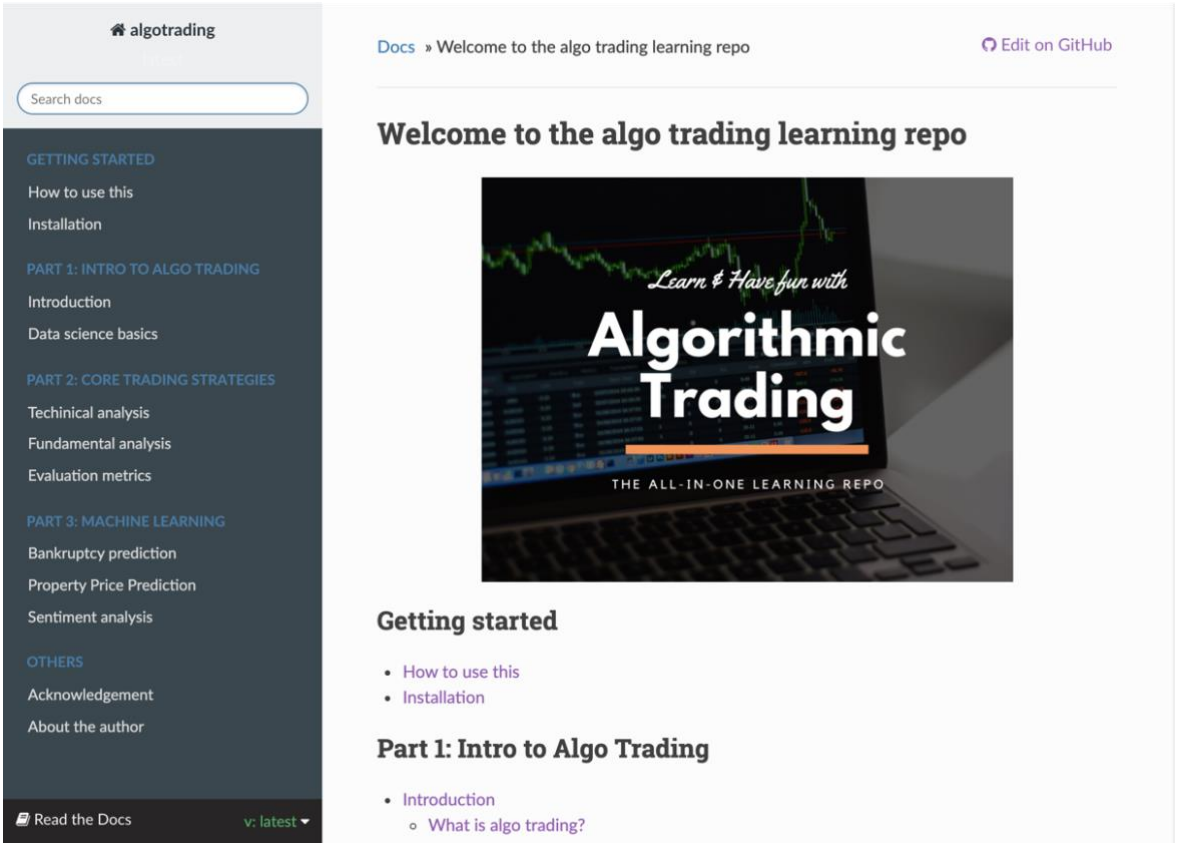


Figure 24: The homepage of the documentation website.

The website consists of tutorials that will serve as a step-by-step guide for new learners. There are three parts in the tutorials:

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

Part 1 explains the basic financial concepts. Part 2 elaborates on different trading strategies with code examples and mathematical equations. Part 3 includes various machine learning

models that are useful in analysing the market from different perspectives. Figure 25 shows a screenshot of the documentation website featuring an explanation of the Commodity Channel Index, the equation for calculating the indicator and a code example.

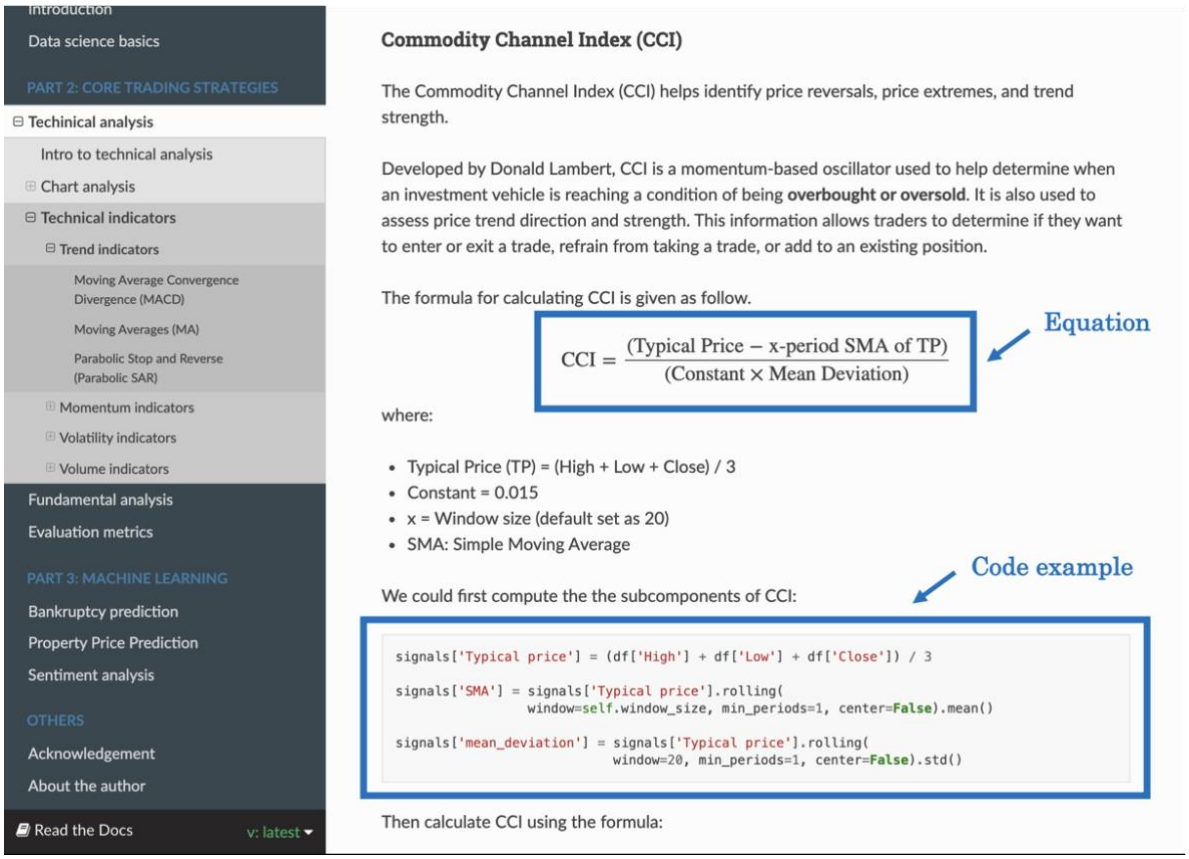


Figure 25: A page of the documentation website featuring explanation of the Commodity Channel Index, the equation for calculating the indicator and a code example.

5. Conclusion

The purpose of this project is to build an open-source code and data repository for teaching algorithmic trading. The project was initiated to implement a pipeline for collecting microeconomic, macroeconomic, and sentimental data. The data had been continuously updated till the end of the project period. Using the collected data, indicators were created that could analyse the market from various perspectives. Then, the indicators were integrated together using different strategies to predict the stock trends in Hong Kong and generate trade signals. The performances of trading strategies were evaluated by running back-tester. In addition, the trading strategy was connected with the real-time trading platform to trade automatically via Python code. To achieve the educational objectives of this project, all the codes are documented with detailed explanations on a website.

The final product of this project is an all-in-one repository that contains all the relevant information required for teaching algorithmic trading. From technical skills in developing algorithmic trading strategies to basic financial concepts to understand, this repository shall enable the learners to acquire a deep understanding of algorithmic trading. Moreover, our website will serve as a complete step-by-step guide for algorithmic trading beginners.

5.1 Difficulties Encountered

There were three difficulties encountered during the data collection:

1. Websites usually set the upper threshold on calling APIs, and hence there was difficulty in collecting data. However, it was overcome by introducing a pause between calls.
2. The original website for collecting data (Centaline Property) was renewed, and the way of collecting data was totally changed. Although the original API and the code for collecting data cannot be used, a new API was found and used to collect a new set of data.
3. There was an error when fetching data by sending some POST API calls to the new Centaline Property website. Not very frequent, but sometimes the API returns an empty string, which resulted in missing data. The error seems to be generated due to a server issue, so it was inevitable.

There was a difficulty encountered when executing a paper trade

1. There were some functions restricted to subscribed users only. For example, users have to subscribe to market data in order to get the data. However, we overcame this problem by calling other available API to get the market data.

5.2 Improvement and Future Work

The followings are the possible work that could be done for improvement:

1. Currently, macroeconomic data are manually downloaded from Census and Statistic Department website. It could be automated by finding an API to collect the macroeconomic data.
2. The performance of the multi-feature LSTM model was not satisfactory. The model could be optimized further, or another machine learning model (such as regression model) could be introduced to improve the accuracy in generating the buy/sell signals.

Bibliography

Altman, E. I. (2013). Predicting financial distress of companies: revisiting the z-score and zeta® models. In Handbook of research methods and applications in empirical finance. Edward Elgar Publishing.

Beaver, W. H. (1966). Financial ratios as predictors of failure. Journal of accounting research, pages 71–111.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. Journal of computational science, 2(1):1–8.

Deng, S., Mitsubuchi, T., Shioda, K., Shimada, T., and Sakurai, A. (2011). Combining technical analysis with sentiment analysis for stock price prediction. In 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pages 800–807. IEEE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Hang Seng Indexes Company Limited (2020). Hang seng index factsheet. https://www.hsi.com.hk/static/uploads/contents/en/dl_centre/factsheets/hsie.pdf.

Hendershott, T., Riordan, R., et al. (2009). Algorithmic trading and information. Manuscript, University of California, Berkeley.

Hutto, C. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Proceedings of the International AAAI Conference on Web and Social Media, volume 8.

Melin, M. (2017). Aqr: Computers don’t replace human stock pickers, they augment them.

Nneji, O., Brooks, C., and Ward, C. W. (2013). House price dynamics and their reaction to macroeconomic changes. Economic Modelling, 32:172–178.

Rao, T., Srivastava, S., et al. (2012). Analyzing stock market movements using twitter sentiment analysis.

Shirata, C. (2003). The bankruptcy prediction model–saf 2002 model. Chuo-Keizai Sha.

Sohangir, S., Wang, D., Pomeranets, A., and Khoshgoftaar, T. M. (2018). Big data: Deep learning for financial sentiment analysis. Journal of Big Data, 5(1):3.

Statista (2020). Hong kong social network penetration. https://www.statista.com/statistics/412500/hk-social-network-penetration/.

Wu, J.-L., Su, C.-C., Yu, L.-C., and Chang, P.-C. (2012). Stock price predication using combinational features from sentimental analysis of stock news and technical analysis of trading information. International Proceedings of Economics Development and Research.

Zhang, X., Fuehres, H., and Gloor, P. A. (2011). Predicting stock market indicators through twitter “i hope it is not as bad as i fear”. Procedia-Social and Behavioral Sciences, 26:55–62.

Appendix

The details of APIs used in macroeconomic data collection is provided below:

1. Centaline Property (English data): macroeconomic-analysis/webscrap_centaline.py
The website has been renewed and hence API cannot be used anymore.

2. Centaline Property (Chinese data): macroeconomic-analysis/webscrap_centaline_chinese.py
A POST request should be sent with:

- URL = https://hk.centanet.com/findproperty/api/Transaction/Search
- Headers
    - Host
    - Origin
    - Referer
    - User-Agent
    - Content-Type
    - Content-Length
    - Connection

- Body
    - day – the registration period (e.g. “Day30”, “Day90”, “Day180”, “Day365”, “Day1095”)
    - mtrs – default: []
    - offset – offset of data (should start from 0)
    - order – the order of the data (e.g. “desc”, “asc”)
    - pageSource – default: “search”
    - postType – the type of the transaction (e.g. “Sale”, “Rent”, or “Both”)
    - primarySchoolNets – default: []
    - size – size of the return array (maximum 30)
    - sort – default: “InsOrRegDate”
    - typeCodes – an array of the type code for regions

Sending the request with above headers and body, it will return an array containing the relevant transaction records of the specific registration period. Up to recent 3 years of data can be retrieved, and the data beyond the array size 10000 cannot be retrieved.

In order to update the database, get_property_list(region_name, region_list, reg_period) should be ran. For example, if the Centaline database hasn’t been updated for less than a month, the following code should be ran.

```
get_property_list("hk_island", region_hk, 'Day30')
get_property_list("kowloon", region_kowloon, 'Day30')
get_property_list("new_east", region_new_east, 'Day30')
get_property_list("new_west", region_new_west, 'Day30')
```

The type codes for the regions:
HK Island:

Kennedy_town_sai_ying_pun	19-HMA111, 19-HMA047, 19-HMA056, 19-HMA012
Bel_air_sasson	19-HMA063
South_horizon	19-HMA200
Aberdeen_ap_lei_chau	19-HMA155, 19-HMA127, 19-HMA045, 19-HMA093, 19-HMA151, 19-HMA082, 19-HMA122
Mid_level_west	19-HMA028, 19-HMA053
Peak_south	19-HMA025, 19-HMA141, 19-HMA121, 19-HMA118, 19-HMA123, 19-HMA105, 19-HMA064, 19-HMA018, 19-HMA048, 19-HMA132
Mid_level_central	19-HMA076, 19-HMA026
Happy_valley_mid_level_east	19-HMA130, 19-HMA070, 19-HMA071, 19-HMA126
Wanchai_causeway_bay	19-HMA160, 19-HMA144
North_point	19-HMA041, 19-HMA032, 19-HMA014
Mid_level_north_point	19-HMA042
Quarry_bay_kornhill	19-HMA110, 19-HMA113
Taikoo_shing	19-HMA034
Sai_wan_ho	19-HMA057, 19-HMA195
Shau_kei_wan_chai_wan	19-HMA163, 19-HMA094, 19-HMA020
Heng_fa_chuen	19-HMA061

Kowloon:

Olympic_station	19-HMA134
Kowloon_station	19-HMA003
Mongkok_yaumatei	19-HMA074, 19-HMA068, 19-HMA015, 19-HMA033
Tsimshatsui_jordan	19-HMA172, 19-HMA052, 19-HMA059
Lai_chi_kok	19-HMA043
Nam_cheong	19-HMA081
Ho_man_tin_kings_park	19-HMA065, 19-HMA058
To_kwa_wan	19-HMA013, 19-HMA108
Whampoa_laguna_verde	19-HMA133, 19-HMA095
Tseung_kwan_o	19-HMA148, 19-HMA112, 19-HMA060, 19-HMA159, 19-HMA114
Meifoo_wonderland	19-HMA128, 19-HMA099, 19-HMA098, 19-HMA092, 19-HMA089
Cheung_sha_wan_sham_shui_po	19-HMA171, 19-HMA049, 19-HMA077
Yau_yat_chuen	19-HMA010
Kowloon_tong	19-HMA152, 19-HMA004
Lam_tin_yau_tong	19-HMA156, 19-HMA075
Kowloon_bay_ngau_chi_wan	19-HMA005, 19-HMA040
Kwun_tong	19-HMA161, 19-HMA051
Diamond_hill_wong_tai_sin	19-HMA147, 19-HMA131, 19-HMA066, 19-HMA135, 19-HMA137, 19-HMA162, 19-HMA002, 19-HMA115, 19-HMA039
Hung_hum	19-HMA090, 19-HMA091
Kai_tak	19-HMA117

New Territory East:

Sai_kung	19-HMA055, 19-HMA046, 19-HMA054, 19-HMA119, 19-HMA017
Tai_wai	19-HMA188
Shatin	19-HMA170, 19-HMA106, 19-HMA062, 19-HMA176, 19-HMA021
Fotan_shatin_kau_to_shan	19-HMA187, 19-HMA001
Ma_on_shan	19-HMA107
Tai_po_mid_level_hong_lok_yuen	19-HMA177, 19-HMA180, 19-HMA169
Tai_po	19-HMA184, 19-HMA182, 19-HMA185, 19-HMA181
Sheung_shui_fanling_kwu_tung	19-HMA179, 19-HMA168, 19-HMA166, 19-HMA165, 19-HMA164, 19-HMA189

New Territory West:

Discovery_bay_other_islands	19-HMA125, 19-HMA019, 19-HMA078, 19-HMA067, 19-HMA178
Fairview_park_palm_spring_the_vineyard	19-HMA150, 19-HMA136
Yuen_long	19-HMA085, 19-HMA084, 19-HMA191, 19-HMA087, 19-HMA083, 19-HMA190, 19-HMA030, 19-HMA029, 19-HMA009, 19-HMA008, 19-HMA007, 19-HMA006, 19-HMA149
Tuen_mun	19-HMA138, 19-HMA139, 19-HMA037, 19-HMA038, 19-HMA153, 19-HMA036, 19-HMA035, 19-HMA050, 19-HMA157, 19-HMA086
Tin_shui_wai	19-HMA031
Tsuen_wan_belvedere_garden	19-HMA016, 19-HMA143, 19-HMA103, 19-HMA104, 19-HMA193, 19-HMA100, 19-HMA101, 19-HMA102, 19-HMA158
Kwai_chung	19-HMA140, 19-HMA192
Tsing_yi	19-HMA079
Ma_wan_park_island	19-HMA109
Tung_chung_islands	19-HMA174, 19-HMA183
Sham_tseng_castle_peak_road	19-HMA023, 19-HMA116, 19-HMA011, 19-HMA073, 19-HMA120, 19-HMA080

3. Midland Realty - macroeconomic-analysis/webscrap_midland.py

A GET request should be sent with:

- URL = https://data.midland.com.hk/search/v1/transactions?
- Headers
    - Host
    - Origin
    - authorization
    - Referer
    - User-Agent
    - X-Requested-With

- Params
    - hash – default: ‘true’
    - lang – default: ‘en’

- currency – default: ‘HKD’
- unit – default: ‘feet’
- search_behavior – default: ‘normal’
- dist_ids – the id of the districts
- tx_type – the type of transaction (e.g. ‘s’ for sales)
- tx_date – the registration period (e.g. ‘30days’, ‘90days’, ‘180days’, ‘1year’, ‘3year’)
- page – the page of the data (start from 1)
- limit – the size of the return array (max 50)

Sending the request with above headers and parameters, it will return an array containing the relevant transaction records of the specific registration period. Same as Centaline Property, up to recent 3 years of data can be retrieved, and the data beyond the array size 10000 cannot be retrieved.

In order to update the database, get_property_list(region_name, region_list, reg_period) should be ran. For example, if the Midland database hasn’t been updated for less than 3 months, the following code should be ran.

```
get_property_list("hk_island", region_hk, '90days')
get_property_list("kowloon", region_kowloon, '90days')
get_property_list("new_territory", region_new_territory, '90days')
```

The id of the districts:
HK island:

- Chai_wan: 100404
- Heng_fa_chuen: 100407
- Shau_kei_wan: 100406
- Sai_wan_ho_tai_koo: 100405
- Quarry_bay: 100403
- North_point_fortress_hill: 100401
- Braemar_hill_north_point_mid_level: 100402
- Jardines_lookout_tai_hang: 100201
- Happy_valley_mid_level_east: 100202
- Wan_chai_causeway_bay: 100203
- Tin_hau: 100204
- Central_mid_level_admiralty: 100101
- Sheung_wan_central: 100104
- Hong_kong_west: 100102
- Western_mid_levels: 100103
- The_peak: 100105
- Residence_bel_air_pokfulam: 100303
- Ap_lei_chau: 100305
- Aberdeen_wong_chuk_hang: 100304
- Repulse_bay_shou_son_hill: 100301
- Tai_tam_shek_o: 100306
- Stanley: 100302

Kowloon:

- Tsim_sha_tsui: 200501
- Kowloon_station: 200504
- Yau_ma_tei: 200507
- Kingspark: 200503
- Mongkok: 200502
- Tai_kok_tsui: 200506
- Olympic: 200505
- Lai_chi_kok: 200601
- Mei_foo: 200604
- Cheung_sha_wan_sham_shui_po: 200602
- Yau_yat_tsuen: 200603
- Kowloon_tong_beacon_hill: 200903
- Ho_man_tin: 200902
- Hung_hum: 200901
- To_kwa_wan: 200904
- Kai_tak: 200906
- Kowloon_city: 200905
- Wong_tai_sin_lok_fu: 200801
- Diamond_hill_san_po_kong_ngau_chi_wan: 200802
- Kowloon_bay: 200701
- Kwun_tong: 200703
- Lam_tin_yau_tong: 200702
- Lohas_park: 20100
- Tiu_keng_leng: 201004
- Hang_hau: 201001
- Po_lam_tseung_kwan_o_station: 201002

New Terrirory:

- Sai_kung_clear_water_bay: 301003
- Shatin: 301702
- Kau_to_shan_fotan: 301701
- Ma_on_shan: 301703
- Tai_po: 301601
- North: 301502
- Sheung_shui_fanling: 301501
- Hung_shui_kiu: 301403
- Fairview_palm_springs_the_vineyard: 301404
- Tin_shui_wai: 301401
- Yuen_long: 301402
- Tuen_mun: 301301
- Tsuen_wan: 301101
- Sham_tseng: 301102
- Ma_wan: 301103
- Kwai_chung: 301201
- Tsing_yi: 301202
- Discovery_bay: 301802
- Tung_chung: 301803
- Lan_tau_island: 301801