Interim Report

Building a Repository for Teaching Algorithmic Trading

Supervisor: Dr. RuiBang Luo
Woo Chung Yu, Angel
Wu Xue, Snow
Lee Kwanyoung

January 28, 2021

Abstract

In this project, we propose to build an open-source code and data repository that puts together the relevant financial concepts and technical skills for learning algo trading. Over the span of four months, we have built pipelines for collecting stock price, company financial statements, property price, social media and financial news data. This database will be updated continuously til the end of the project period. In addition to the database, we have also implemented indicators for microeconomic, macroeconomic and sentiment analysis, predominantly in Python, 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. In response to the educational objective 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. These tutorials are coupled with equations and code examples. Having made satisfactory progress in the last phase, we plan to move on to investigate methods to predict stock trends with multi-source features, and to have execution of trade in the context of Hong Kong as an additional topic in our repository. The repository and the documentation website are respectively available at https://github.com/awoo424/algotrading and https://algo-trading.readthedocs.io/.

Contents

List of Figures

List of Tables

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). With the advantages of high efficiency and easier management, algo trading has become increasingly dominant in the market, and by now it is gauged that up to 90% of total trading is entirely algorithmic (Melin, 2017). The fact that it is prevalent in the market, however, does not necessarily imply that it is a skill easy for individual investors to acquire and apply.

By studying various learning resources available (including blog posts, online courses and websites like Investopedia) on the Internet, we concluded that they suffer from the following problems in general:

- 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.
- Majority of the resources do not fit the local context (e.g. about trade execution in Hong Kong) and are merely focusing on the US market.

Ideally, we want an all-in-one, comprehensive educational resource that puts together the interdisciplinary knowledge necessary for learning algo trading (i.e. data science, coding and basic finance concepts), and aligns with the financial setting of Hong Kong. Being a group of students who are majoring in Computer Science and Finance at the university, we are eager to dedicate our efforts into bringing this to reality and eliminate the stumbling block in the learning journey of individual investors or students who are intrigued to pick up the skill.

2. Objectives

This project aims to achieve several educational and research objectives in support of teaching algorithmic trading and stock movement prediction based on multi-source features.

2.1 Educational Objectives

The primary educational objective of this project is to build an open-source repository that puts together the financial concepts, data science skills and algorithmic design techniques that are crucial to learning algorithmic trading; and thus to present them in a clear, understandable manner. The contents of the repository aim 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.

[1]Also referred to as “algo”, “robo” or “black-box” trading in the literature

2.2 Research Objectives

The research objective of this project is to experiment and design an algorithm that integrates indicators of different dimensions in order to predict price movement of stocks listed in the Hong Kong and United States (US) markets. These indicators include technical, fundamental, macroeconomic and market sentiment, and they intend to cover the majority of factors affecting the stock market. The correlation between various indicators and stock market trends will be analysed and evaluated. In addition, a consolidated database that consists of historical stock tick, property prices data and social media data will also be constructed to serve for future research purposes at the University of Hong Kong.

3. Project Overview

The project is divided into two parts, as shown in Figure 1. Part 1 has been completed in Semester 1, and Part 2 is to be commenced in Semester 2.

The first part looks into the data sources essential for forming a trading strategy. We have examined the market from three perspectives - microeconomic, macroeconomic and sentimental. With respect to each perspective, historical data have been collected and indicators to analyse the financial market have been created.

In the second part, we will experiment on algorithms that leverage data from all these three aspects to predict stock price movements. Additionally, we will simulate virtual investment by opening a *Paper Trading*[2] account and study the trading setting in Hong Kong. The final product is envisioned to be a self-contained repository which consists of the modules to build, test and evaluate an algorithmic trading strategy.

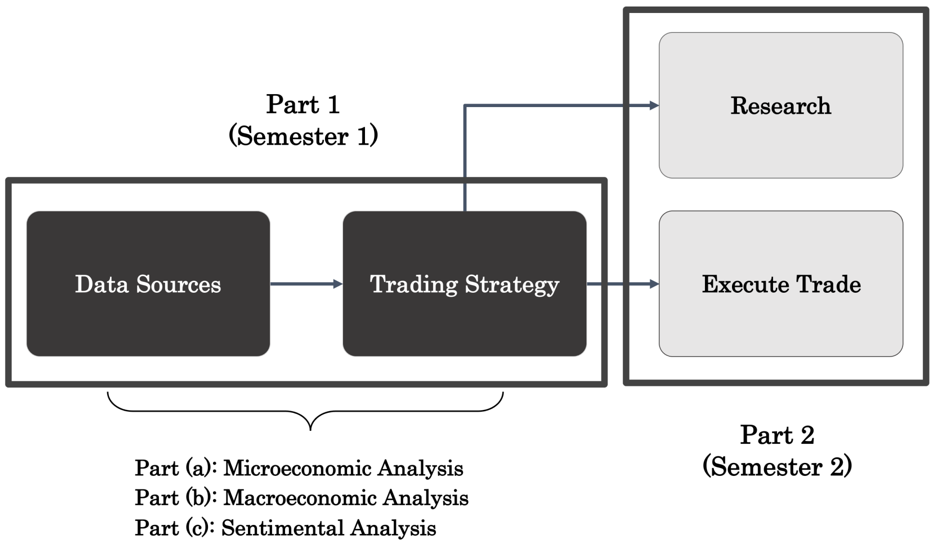

Part 1
(Semester 1)
Data Sources
Trading Strategy
Research
Execute Trade
Part 2
(Semester 2)
Part (a): Microeconomic Analysis
Part (b): Macroeconomic Analysis
Part (c): Sentimental Analysis

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

[2]Simulated trade that allows an investor to practise buying and selling without risking real money

4. Methodology

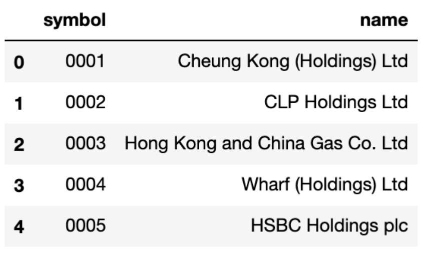
In the following, we will elaborate on the methodology for each subpart in Part 1.

4.1 Part 1(a) - Microeconomic Analysis

Microeconomic analysis consists of three major components - (1) Data collection, (2) Technical analysis, and (3) Fundamental analysis. A database that features historical stock prices data and fundamentals data has been built. Moreover, technical indicators that scrunitise different properties such as momentum and volatility of a security has been implemented. Code tutorials that illustrate how to exploit the fundamentals data to carry out stock screening and bankruptcy prediction have also been included in the repository.

4.1.1 Data Collection

Stock tick data To collect the historical price data (i.e. stock tick) for each listed company, the complete list of ticker symbols (example shown in Figure 2) is first downloaded from the stock exchange official websites respectively. With this list, it could then be loaded 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 calling the Yahoo! Finance API. The outline of the algorithm for downloading the stock tick data is shown in Algorithm 1.

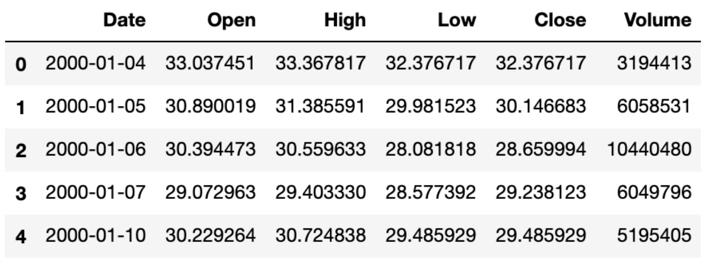


	symbol	name
0	0001	Cheung Kong (Holdings) Ltd
1	0002	CLP Holdings Ltd
2	0003	Hong Kong and China Gas Co. Ltd
3	0004	Wharf (Holdings) Ltd
4	0005	HSBC Holdings plc

Figure 2: The fist few rows in the csv file storing the complete list of ticker symbols that could be traded on Hong Kong Stock Exchange (HKEx).

```
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;
7 end
```

An example output csv file of the stock tick data for a ticker is shown in Figure 3. 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.



	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.

The 1-day price data for each ticker were downloaded since the initial public offering (IPO) of the company, and its 1-minute data were downloaded starting from 1 June 2020. As both the 1-day and 1-minute stock tick data needs to be continuously updated til the end of the project period, I have also written code so as to automate the process of downloading the data between the last updated date and the latest trading day. The pseudocode for the program is shown in Algorithm 2.

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

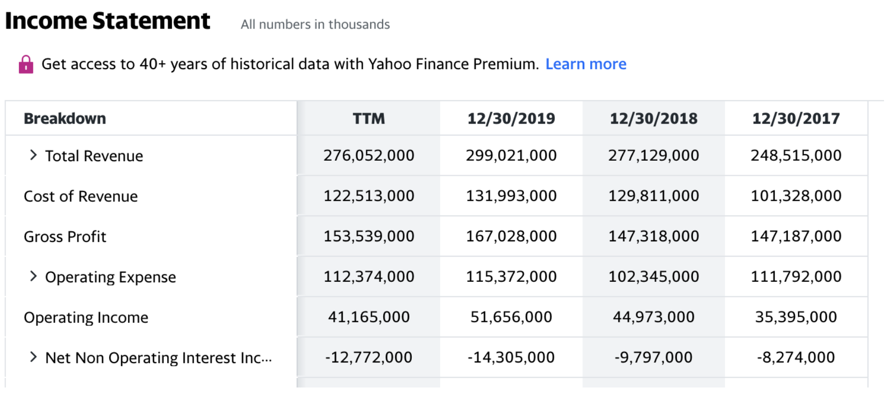
```
Data: Directory containing stock tick files
Result: Stock tick files with records til latest trading day
Import modules;
dir_name ← name of directory storing the stock tick files;
pathlist ← Path(dir_name).rglob('*.csv');
for path in pathlist do
    df ← Read file as csv;
    last_date ← Get last date in df;
    today ← Get today's date;
    dates ← list of dates between last_date and today;
    if last_date != today then
        data ← Call Yahoo! Finance API with start=dates[0], end=dates[−1];
        if last_date == last date in data then
            # Today is not a trading day
            print(Data already up-to-date, no changes made);
        else
            Append data to df;
        end
    else
        print(Data already up-to-date, no changes made);
    end
end
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 the completion of the download, a success message would be displayed.

Fundamentals data As there does not exist any open-source API that enables crawling of the fundamentals data directly, 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 4). The outline of the algorithm for scrapping fundamentals data is shown in Algorithm 3.



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.

Algorithm 3: Scrape fundamentals data

```
   Data: List of stock ticker symbols
   Result: csv files with fundamentals data
1  Import modules;
2  Load list of symbols into an array $S[1...n]$;
3  for $symbol$ in $S[1...n]$ do
4      $page \leftarrow$ Fetch the Yahoo! Finance page for $symbol$;
5      $tree \leftarrow$ Parse the page with lxml;
6      $table\_rows \leftarrow$ Fetch all div elements which have class $D(tbr)$;
7      assert $\text{len}(table\_rows) > 0$;
8      $df \leftarrow$ parse $table\_rows$;
9      $df \leftarrow$ clean the data in $df$;
10     Export df to a csv file;
11 end
```

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 intra-day 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.

With 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:

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

For example, to scrape the price `56.000` shown in the element above that is obtained from the HTML document, the code to crawl the data is as follows:

```
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 leads to changes in the HTML document (which is a disadvantage of scraping data with Beautiful Soup).

4.1.2 Technical Analysis

A total of 16 technical indicators out of four different categories (trend, momentum, volume, volatility) have been implemented in Python. Table 1 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 class for each indicator. In addition to the indicators, a backtester has been implemented in Python so that the performance of the strategy could be evaluated with historical price data.

Category	**Indicator(s)**
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.

The major steps in running a technical analysis strategy is shown in Algorithm 4.

Algorithm 4: Backtest a technical analysis strategy

```
Data: Stock tick data for a specified ticker
Result: png files of figures generated and backtesting results printed on terminal
1 Import required libraries;
2 df ← Load stock tick data for specified ticker symbol;
3 Select time range to backtest;
4 ticker ← Set ticker symbol to test;
5 strategy ← Create an instance of the strategy class by passing df as argument;
6 signals, figure ← 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 including final portfolio value and number of trade signals will all be printed in the terminal.

4.1.3 Technical Indicators Performance Analysis

The implementation of indicators coupling with the backtester is useful in evaluating which indicator is suitable for a certain ticker symbol, and to compare the performance of different strategies. In the following, I will show the results which I have obtained from experiments that aim to provide insights for an investment decision from different angles, and discuss what could be analysed from each of the example scenarios.

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

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

Having 0005.HK as an example, from Figure 5, it is suggested that the portfolio return has a "wave-like" pattern with a period of roughly about 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 6 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.

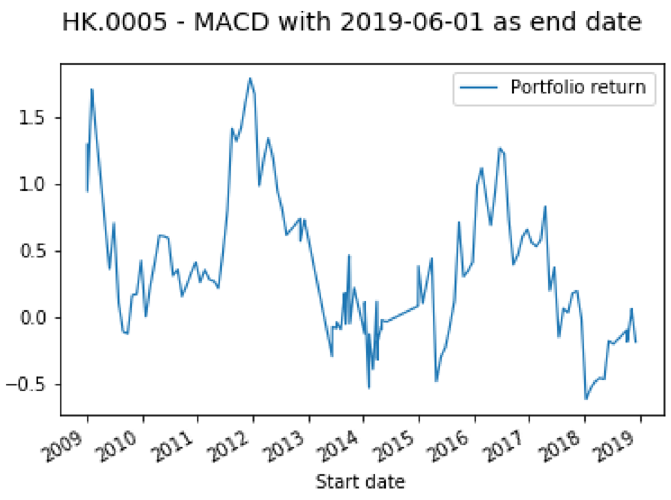


Figure 5: Line plot that illustrates how the portfolio value changes with the time span (varying from 10 years to 6 months), with 0005.HK as the chosen ticker and applying the MACD crossover strategy.

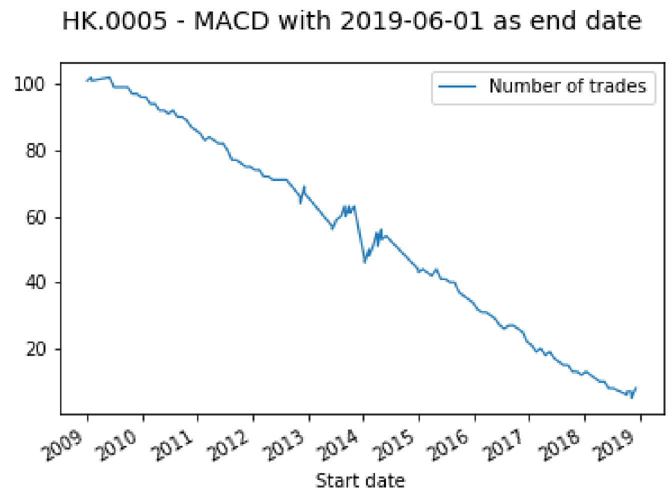
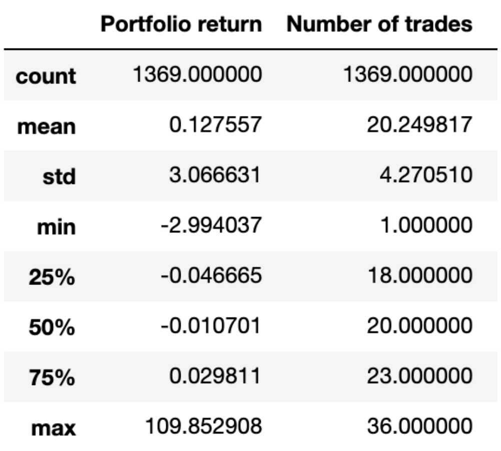


Figure 6: Line plot that illustrates how the number of trading signals changes with the time span (varying from 10 years to 6 months), with 0005.HK as the chosen ticker and applying the MACD crossover strategy.

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 = 2017-01-01 to 2019-01-01, Strategy = MACD crossover



	Portfolio return	Number of trades
count	1369.000000	1369.000000
mean	0.127557	20.249817
std	3.066631	4.270510
min	-2.994037	1.000000
25%	-0.046665	18.000000
50%	-0.010701	20.000000
75%	0.029811	23.000000
max	109.852908	36.000000

Figure 7: Summary of statistics when MACD crossover is applied on all tickers in HKEx given a 2-year period

Figure 7 shows the results of applying the MACD crossover strategy on all tickers listed in HKEx. Referring to the results, the average portfolio return generated by the MACD crossover strategy is about 12.8% and the standard deviation is approximately 3.1, which is quite high. The mean number of trading signals generated in a 2-year period is about 20.

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

From Figure 8, 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 8 and 9, 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.

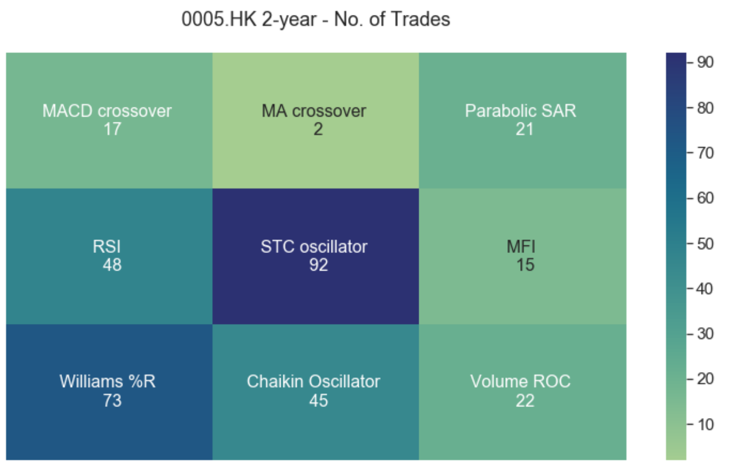


Figure 8: Heat map that illustrates the number of trading signals changes for different strategies, given 0005.HK as the ticker and a fixed time range between 2017 and 2019.

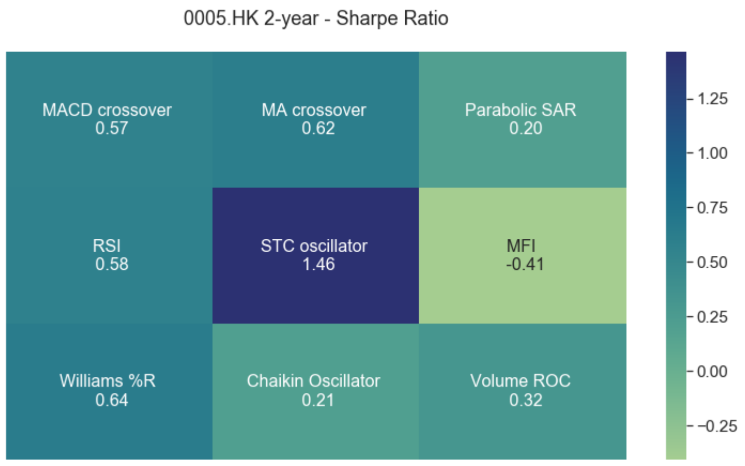


Figure 9: Heat map that illustrates how the sharpe ratio varies for different strategies, given 0005.HK as the ticker and a fixed time range between 2017 and 2019.

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

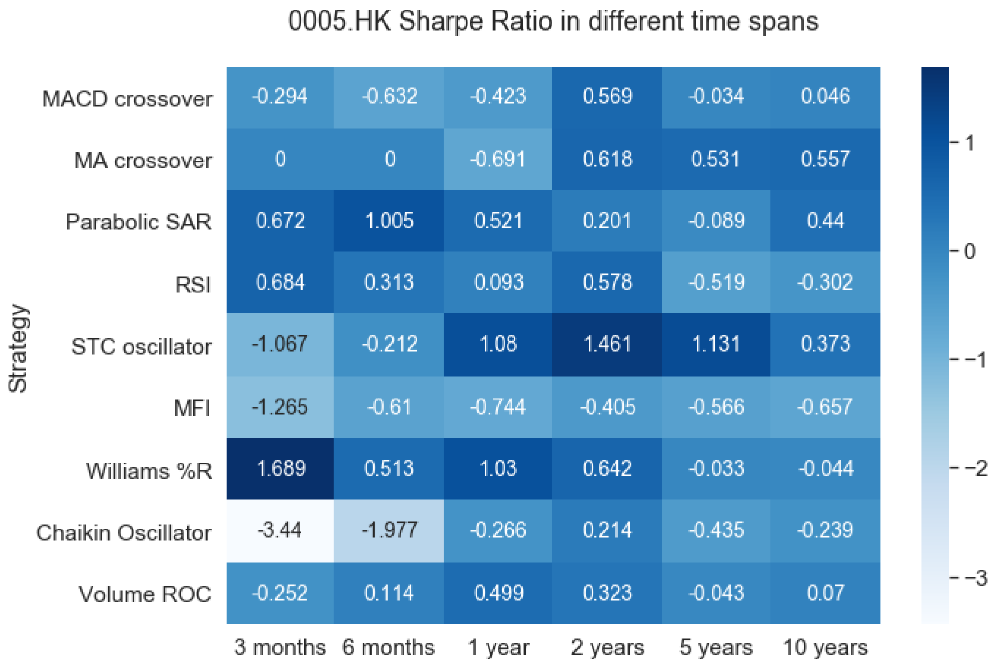


Figure 10: Heat map that illustrates how the sharpe ratio varies for different strategies and time spans (2019-01-01 set as the fixed end date), given 0005.HK as the ticker.

From the heat map shown in Figure 10, 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), so more experiments are yet to be carried out.

In summary, the 4 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.

4.1.4 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 by configuring certain criteria (e.g. with earnings per share (EPS) higher than the overall average). This entire workflow is also 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 in 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 $merged\_df \leftarrow$ Merge dataframes of three types of financial statements;
3 $ratios\_df \leftarrow$ Compute the financial ratios according to the equations (which are
    available on the documentation website);
4 $masks \leftarrow$ Create screening masks;
5 $filtered\_df \leftarrow$ 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 2. One could configure the screening masks based on his/her investment preferences by selecting the relevant ratios.

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.

$$X1 = \text{working capital} \div \text{total assets}$$
$$X2 = \text{retained earnings} \div \text{total assets}$$
$$X3 = \text{Earnings before interest and taxes (EBIT)} \div \text{total assets}$$
$$X4 = \text{total equity (book)} \div \text{total assets}$$
$$X5 = \text{net income} \div \text{total assets}$$
$$X6 = \text{total liabilities} \div \text{total assets}$$
$$X7 = \text{cash flow from operation} \div \text{total liabilities}$$

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.

The machine learning model is first trained with the labelled dataset[3] 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 is split into training set and test set (the 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, I have experimented with four different models - Support Vector Machine (SVM), Decision Tree, Random Forest and K-nearest neighbours, which are all models typically used for classification. The accuracy score for each model is the percentage of data that its prediction is correct. Table 3 summarises the inferencing results of the experiments with the training dataset.

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.

From Table 3, it could be concluded that Random Forest 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 three-year bankruptcy of a company.

[3]https://lopucki.law.ucla.edu/

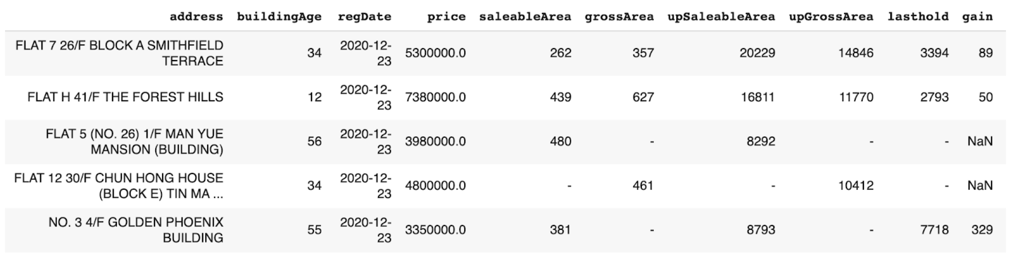
4.2 Part 1(b) - Macroeconomic Analysis

Proceeding on to the macroeconomic aspect, we will look into the Hong Kong real estate market. Hong Kong residential market transactions data will be collected from the websites of different estate agents. The collected data will be analysed over geographies of any kind (e.g. districts) in order to study the interdependence between the real estate market and stock market in Hong Kong.

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.

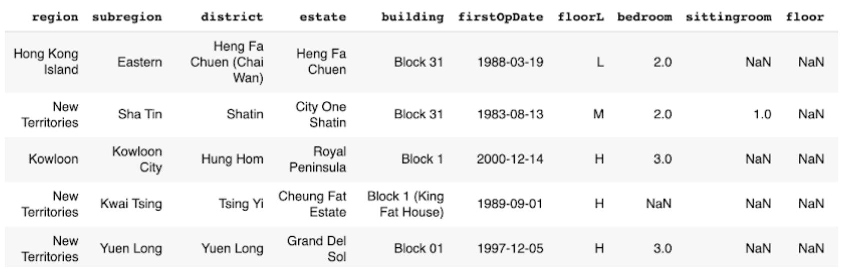
For Centaline Property, the GET request with parameters of district id and registration period returns the whole HTML page, therefore, useful data was extracted from the content by using Beautiful Soup. Figure 11 shows the data structure of transaction records extracted from Centaline Property. The data contains the transaction records from 02/01/2017 to 23/12/2020, with 10 basic features describing the apartment.



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 11: The data structure of transaction record (Centaline Property).

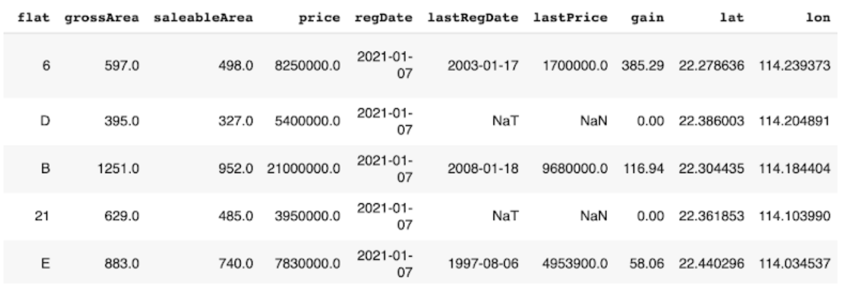
For Midland Realty, the GET request with parameters of district id and registration period returns an array of transaction records for a certain registration period. The data structure of transaction records extracted from Midland Realty is shown in Figure 12 and 13. The data contains the transaction records from 0/01/2018 to 23/12/2020 with 20 features describing the flat. 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 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 12: The data structure of transaction record (Midland Realty) - Part 1.

As the transaction records from Centaline property has fewer feature columns but have a



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

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.

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, and stored in csv format. The data contain economic indicators from 2017 to the present. Table 4 shows the file and data structure of the economic indicators.

Filename	Data	Unit	Time period
TABLE001.csv	Population	thousands	Half-yearly
TABLE006.csv	Unemployment rate (seasonally adjusted)	%	Monthly
	Unemployment rate (not seasonally adjusted)	%	Monthly
TABLE055.csv	Total export	HK$ million	Monthly
	Total import	HK$ million	Monthly
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

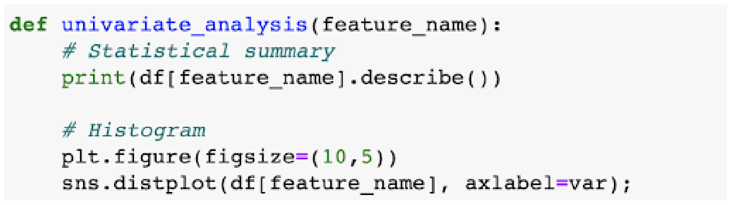
Before analysing the collected data, the data was pre-processed. Firstly, some useful features were derived from existing features. For example, building age was calculated from the first operating date of the building. Then, unmeaningful features and features with too many missing values were dropped. 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 transactioin 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 were 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 visualise the results with histograms. Figure 14 shows the code snippet for 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 14: Code snippet for univariate analysis.

In bivariate analysis, correlations between the features were studied. Using scatter plot and regression line, the relationship between two features were visualised. An example of a scatter plot with a regression line is shown in Figure 15. 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.

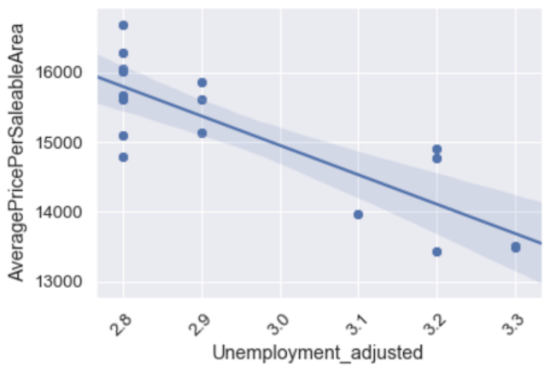


Figure 15: An example of a scatter plot with a regression line.

Feature	**Correlation coefficient**
GDP per capita	0.82
GDP	0.70
Composite consumer price index	0.69
Population	0.68
Year	0.68
Imports	0.68
Month	0.38
Total exports	0.37
Unemployment rate (not seasonally adjusted)	-0.65
Unemployment rate	-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 housing 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 housing 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 housing 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 16. According to the figure, the housing 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.

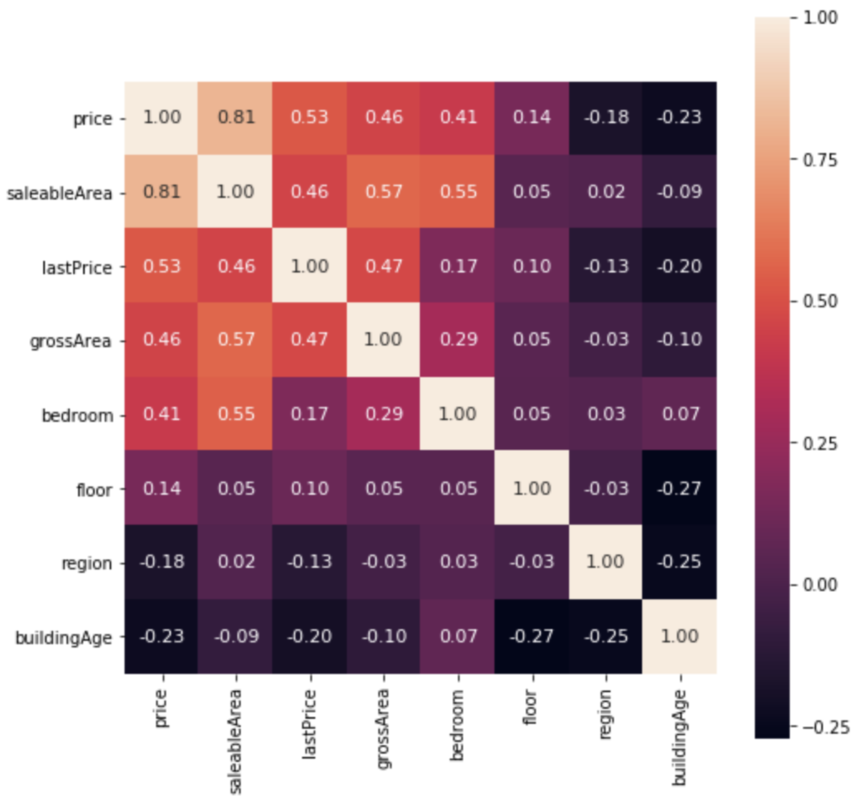


Figure 16: Correlation heatmap of the top 7 features selected in the property transaction data.

4.2.4 Property Price Prediction

Based on the transaction data analysis, property 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 normalise the highly skewed price data. In this way, the dynamic range of Hong Kong's property price 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 because the price values are too big, and RMSLE prevents penalising 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. XG-Boost 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 17 shows the graph of actual and predicted property price for XGBoost. It tends to give smaller error for the range of price which has a larger number of training data.

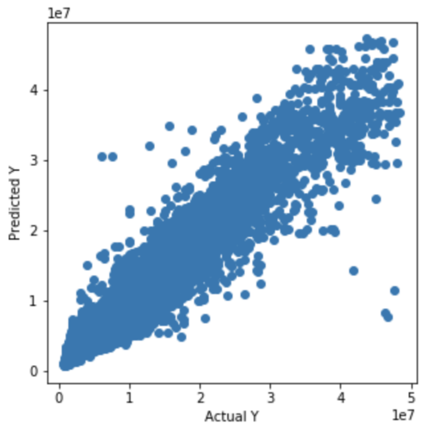


Figure 17: The graph of actual and predicted house price for XGBoost.

4.3 Part 1(c) - Sentiment Analysis

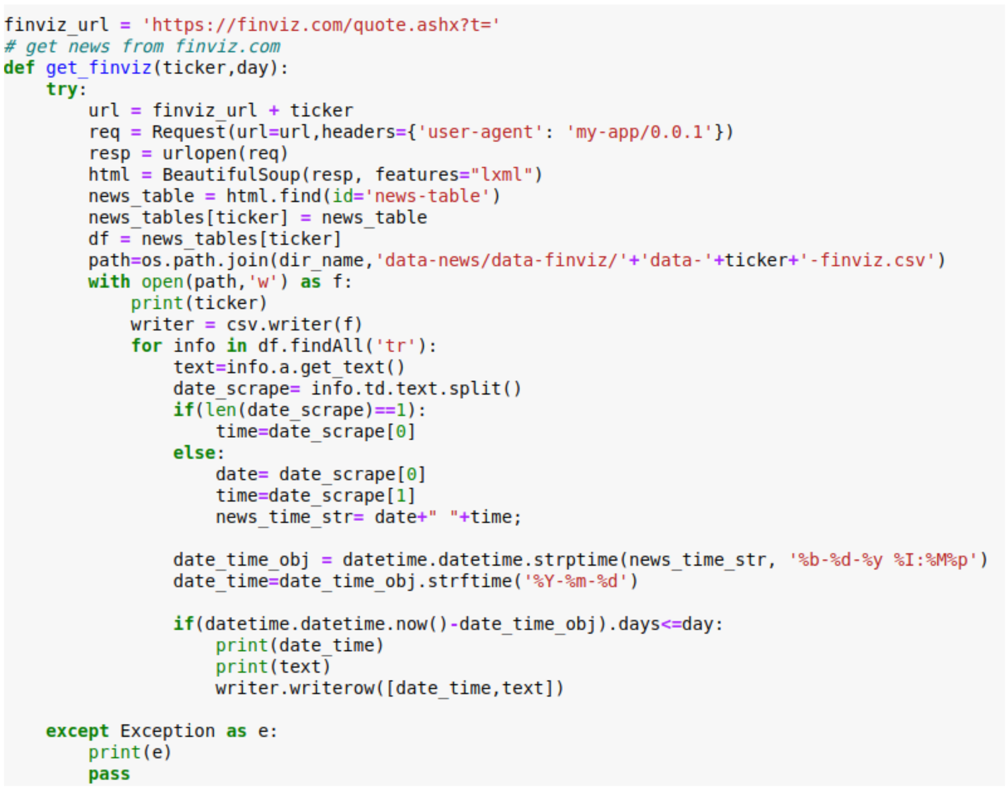
Regarding the sentimental aspect, market sentiment will be analysed through data collected from different social media platforms and from news articles relevant to respective stock markets. As the popularity of social media platforms varies across countries, text mining technologies will be applied to extract data that has the highest relevance to the local market.

4.3.1 Financial News Headline Collection and Update

In order to account for the difference in front-end architecture in different websites, various code designs were implemented to scrape data from the web using the Beautiful Soup library.

With regards to the US market, relevant financial news for each listed company in NASDAQ and NYSE were collected from finviz.com. It is a browser-based stock market research platform that allows visitors to see the latest financial news collected from different major newsagents such as Yahoo! Finance, Accesswire, and Newsfile.

The code snippet shown in Figure 18 demonstrates the program designed for updating news from finviz.com. By passing in the ticker name on a required day, a GET request was sent to the URL in the ticker's page that features all the relevant news. News headlines were extracted under the div tag with the id name of `news-table` and stored in the ticker's corresponding csv file.

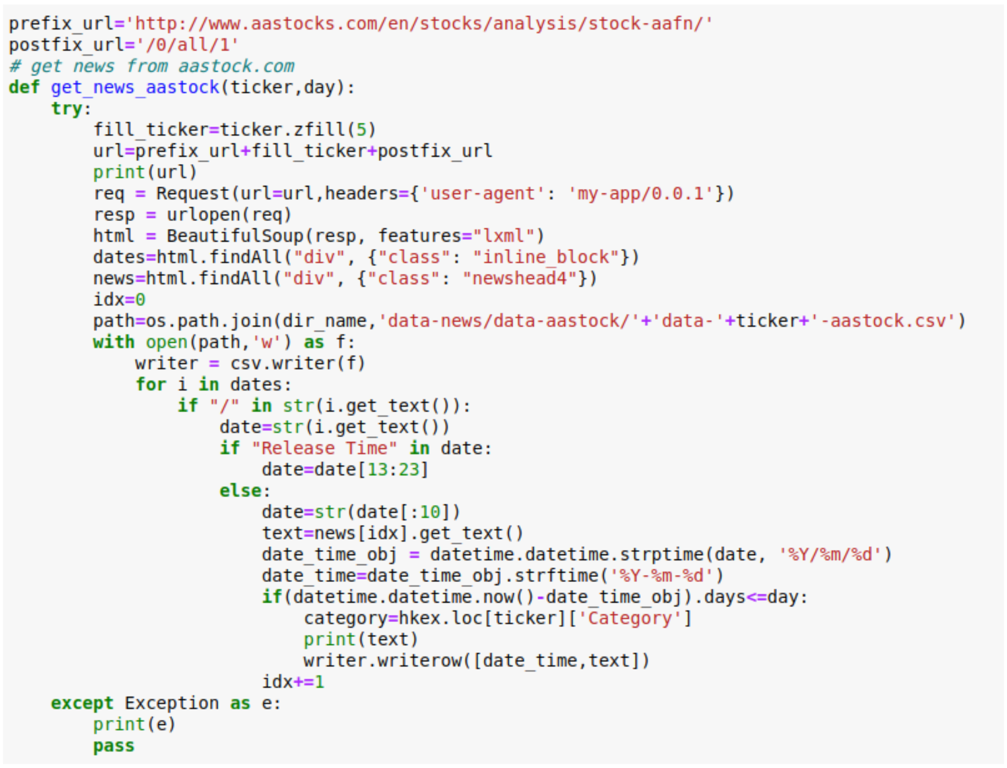


```python
finviz_url = 'https://finviz.com/quote.ashx?t='
# get news from finviz.com
def get_finviz(ticker,day):
    try:
        url = finviz_url + ticker
        req = Request(url=url,headers={'user-agent': 'my-app/0.0.1'})
        resp = urlopen(req)
        html = BeautifulSoup(resp, features="lxml")
        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 18: Code snippet for updating news from finviz.com.

Regarding the Hong Kong market, financial news headlines relevant to each listed company were collected from aastock.com. The website has been one of the highest-ranking financial information platform in Hong Kong for more than a decade. It offers real-time international information relevant to Hong Kong shares, which are useful for analysing sentiment and trends in the local market.

The code snippet shown in Figure 19 demonstrates the code for updating the news from aastock.com. By passing in the ticker name on a required day, a GET request will be sent to the URL in the ticker's page that features all the relevant news. As opposed to data from finviz.com, note that the news headlines and dates were not well organised in a table form. As headline text and dates were stored in chronological order in separate `div` elements, text and dates were collected separately and then joined as a table (by matching the datetime) at the last stage.



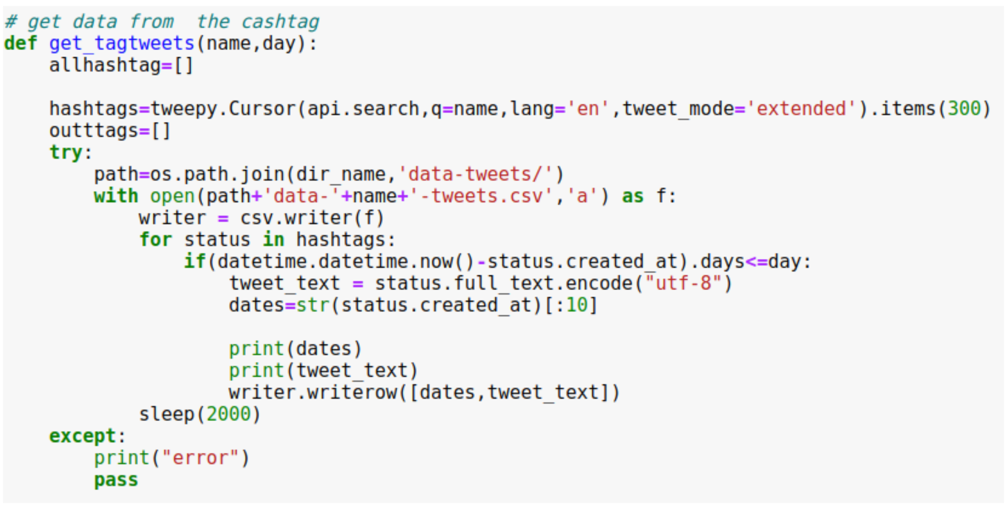
```python
prefix_url='http://www.aastocks.com/en/stocks/analysis/stock-aafn/'
postfix_url='/0/all/1'
# get news from aastock.com
def get_news_aastock(ticker,day):
    try:
        fill_ticker=ticker.zfill(5)
        url=prefix_url+fill_ticker+postfix_url
        print(url)
        req = Request(url=url,headers={'user-agent': '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 19: Code snippet for updating news from aastock.com.

4.3.2 Tweets Collection and Update

The downloading and updating of Tweets were completed through the Tweepy API. Relevant tweets about each company listed in the US (NYSE and NASDAQ) were extracted and stored. The company's ticker symbol is passed as the API input to retrieve the tweets with hashtags mentioning the stock. Corresponding tweets and dates were collected and appended to the company's csv file that was created initially.

The code in Figure 20 indicates the function for updating tweets from Twitter. By passing in the ticker name on a required day, its relevant tweets were presented in the extended mode (i.e. having the full text of the tweets). The data was then updated to the data repository after the text has been encoded in the utf-8 mode.



```
# 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 20: Code snippet for updating tweets using the API.

4.3.3 Data Pre-processing

Referring to Figure 21, data files within the data-news directory were merged to build a single dataframe that aggregates all Hong Kong news or US news for stock trend analyses. Data files within the data-tweets directory were merged to build a dataframe for the Twitter sentiment analysis in the US during the model training process.

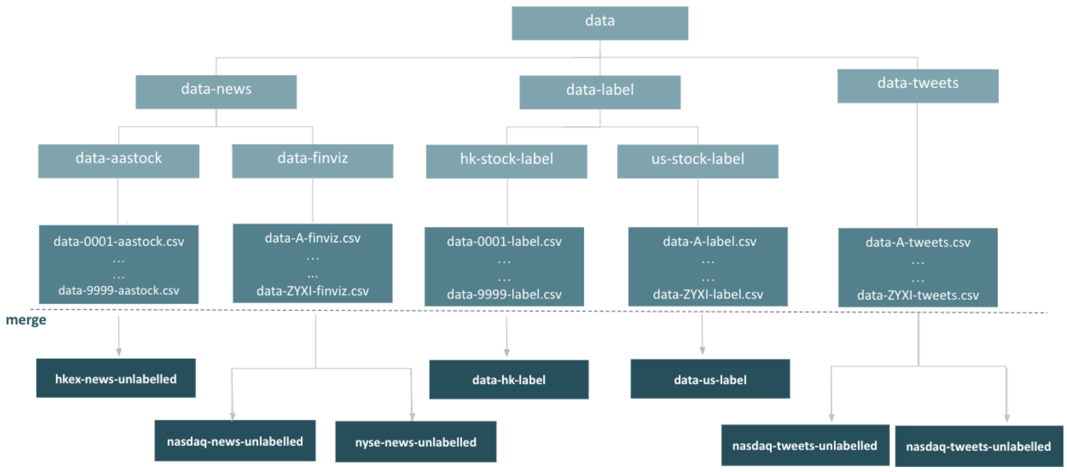


Figure 21: File structure of sentiment analysis data in the repository.

4.3.4 Machine Learning Analysis

VADER unsupervised learning Firstly, we experimented with the VADER model (Hutto and Gilbert, 2014) which is a simple rule-based model for sentiment analysis of social media text. Results show that it achieves a high accuracy on a well-labeled dataset that features 5000 positive tweets and 5000 negative tweets from the NTLK library. The *VADER Compound score* (Hutto and Gilbert, 2014) is a metric that sums up all the lexicon ratings (for each tweet) which have been normalised between -1 (most extreme negative) and +1 (most extreme positive).

Thus, the VADER model was employed to label the dataset of tweets and news headlines in the data repository, with the steps shown in Figure 22. The sentiment labels are generated from the VADER Compound score according to the following rules:

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

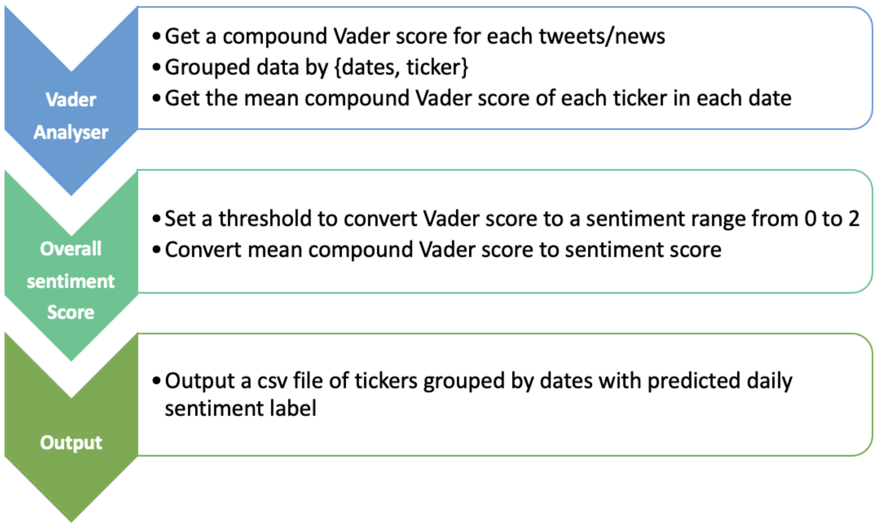
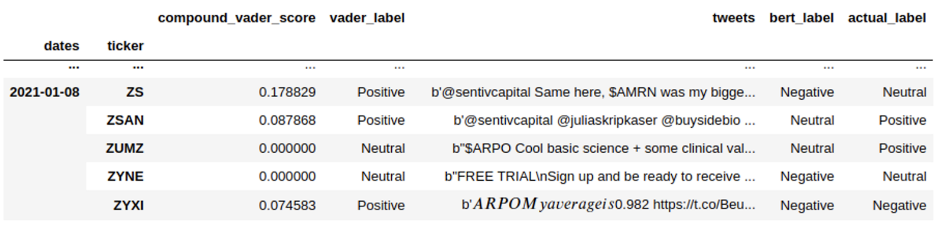

Vader Analyser
• Get a compound Vader score for each tweets/news
• Grouped data by {dates, ticker}
• Get the mean compound Vader score of each ticker in each date
Overall sentiment Score
• Set a threshold to convert Vader score to a sentiment range from 0 to 2
• Convert mean compound Vader score to sentiment score
Output
• Output a csv file of tickers grouped by dates with predicted daily sentiment label

Figure 22: High-level workflow of Vader training.

BERT Machine Learning Model Moving on, we also exploited the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) model for sentiment analysis, which is the state-of-the-art neural network based-technique developed by Google. It is used for natural language processing pre-training, and in this project, a pre-trained machine learning model (bert-base-uncased[4]) and a BERT tokeniser were employed to capture the daily sentiment of tweets and news. Each input is fed into the BERT tokeniser before passing into the model. In the last step, the softmax layer was applied to get the predictions and the argmax function was used to determine the sentiment label. An example of the final output is shown in Figure 23.

[4]https://huggingface.co/bert-base-uncased

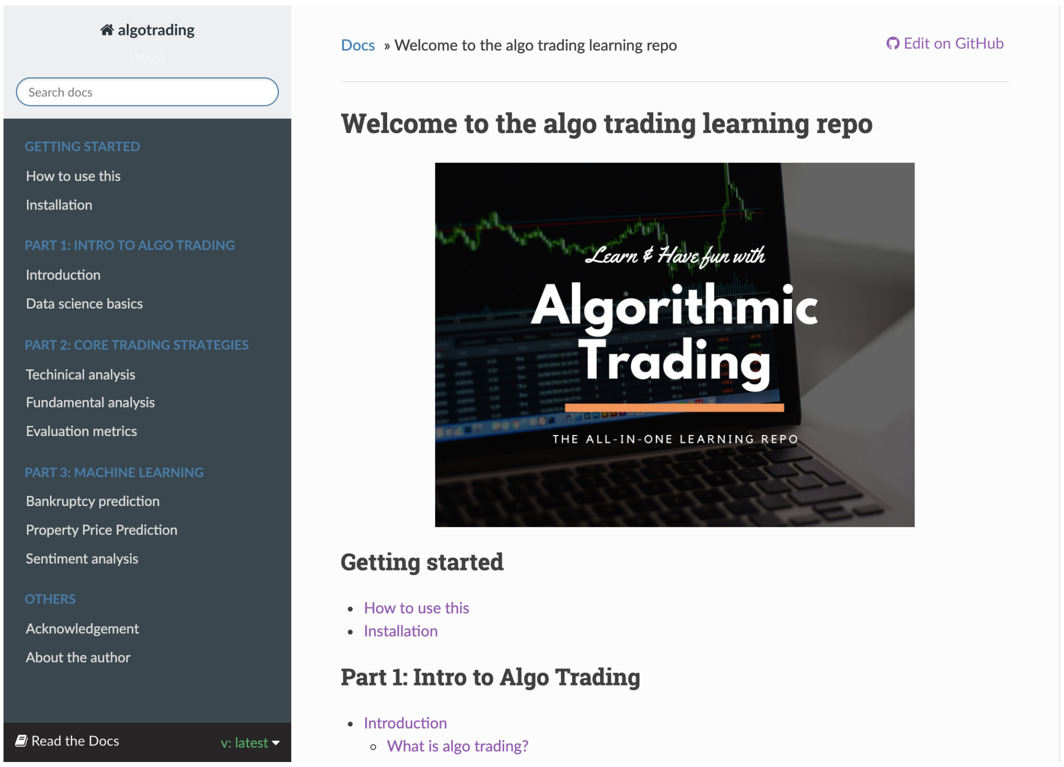


dates	ticker	compound_vader_score	vader_label	tweets	bert_label	actual_label
...	...	...	...	...	...	...
2021-01-08	ZS	0.178829	Positive	b'@sentivcapital Same here, $AMRN was my bigge...	Negative	Neutral
	ZSAN	0.087868	Positive	b'@sentivcapital @juliaskripkaser @buysidebio ...	Neutral	Positive
	ZUMZ	0.000000	Neutral	b"$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'*ARPOM yaverageis*0.982 https://t.co/Beu...	Negative	Negative

Figure 23: Sample of tweets sentiment analysis output for NASDAQ from 2020-12-28 to 2021-01-08.

4.4 Documentation Website

In response to the educational aim of this project, all the code implementations have been (and some parts will be) documented in a website hosted on Readthedocs (https://algo-trading.readthedocs.io/). It is built with Sphinx, which is a documentation generator especially designed for writing technical documentations. The homepage of the website is shown in Figure 24.



algotrading
latest
Search docs

GETTING STARTED
How to use this
Installation

PART 1: INTRO TO ALGO TRADING
Introduction
Data science basics

PART 2: CORE TRADING STRATEGIES
Techinical analysis
Fundamental analysis
Evaluation metrics

PART 3: MACHINE LEARNING
Bankruptcy prediction
Property Price Prediction
Sentiment analysis

OTHERS
Acknowledgement
About the author

Read the Docs v: latest

Docs » Welcome to the algo trading learning repo
Edit on GitHub

Welcome to the algo trading learning repo

Getting started
- How to use this
- Installation

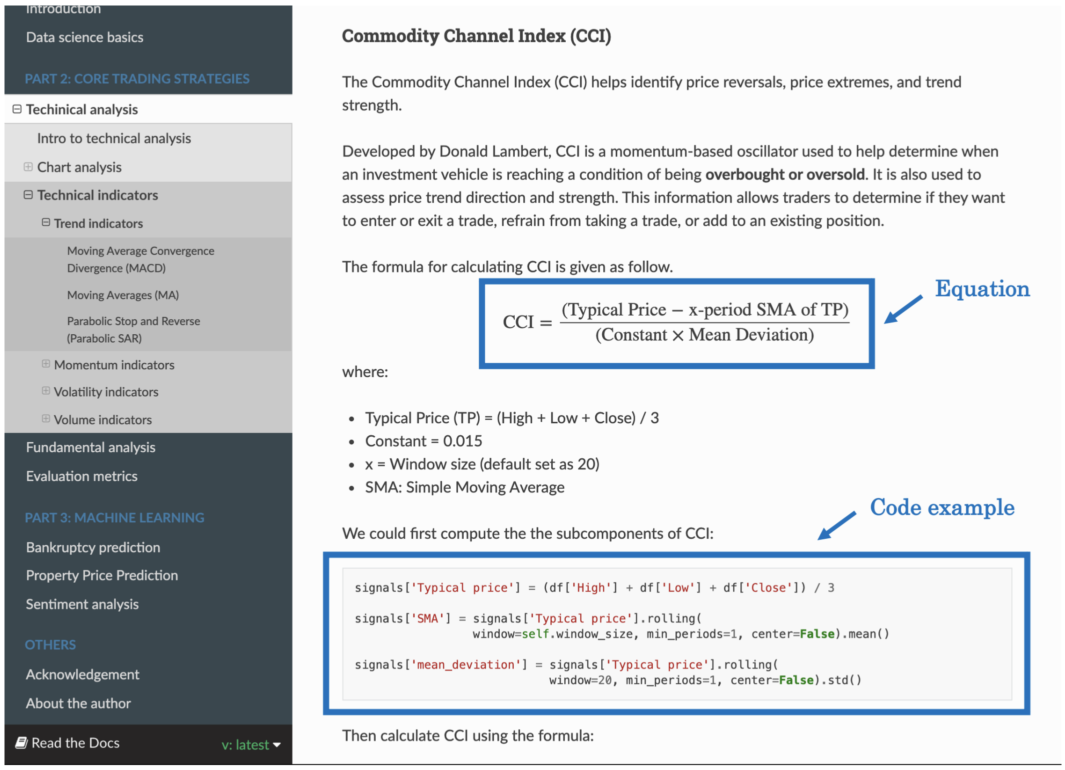
Part 1: Intro to Algo Trading
- Introduction
  - What is algo trading?

Figure 24: The homepage of the documentation website.

The tutorials in repository are divided into three parts, namely:

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

Each part features explanations of basic financial concepts as well as mathematical equations for various strategies included in the repository. They are all coupled with code examples and descriptions so that it is easy for beginners to follow, as illustrated in Figure 25. Concerning the Machine Learning part, we will discuss example strategies that are based on the correlation between certain macroeconomic indicators, news sentiment and stock prices, and to inspire users to build their own trading pipeline.



Commodity Channel Index (CCI)

The Commodity Channel Index (CCI) helps identify price reversals, price extremes, and trend strength.

Developed by Donald Lambert, CCI is a momentum-based oscillator used to help determine when an investment vehicle is reaching a condition of being **overbought or oversold**. It is also used to assess price trend direction and strength. This information allows traders to determine if they want to enter or exit a trade, refrain from taking a trade, or add to an existing position.

The formula for calculating CCI is given as follow.

$$\text{CCI} = \frac{(\text{Typical Price} - \text{x-period SMA of TP})}{(\text{Constant} \times \text{Mean Deviation})}$$

where:

- Typical Price (TP) = (High + Low + Close) / 3
- Constant = 0.015
- x = Window size (default set as 20)
- SMA: Simple Moving Average

We could first compute the the subcomponents of CCI:

```
signals['Typical price'] = (df['High'] + df['Low'] + df['Close']) / 3

signals['SMA'] = signals['Typical price'].rolling(
                window=self.window_size, min_periods=1, center=False).mean()

signals['mean_deviation'] = signals['Typical price'].rolling(
                            window=20, min_periods=1, center=False).std()
```

Then calculate CCI using the formula:

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. Progress Evaluation

Overall, the progress of the project has been satisfactory as we have successfully completed the majority of Part 1 in accordance with the timeline set in the Project Plan. Table 7 summarises the progress of each subpart in the project.

Item	Progress
Microeconomic analysis • Data collection • Data pre-processing • Technical & fundamental analysis	• **Data collection**: pipeline completed, database continuously updated til the end of project period • **Python code**: completed • **Julia code**: in progress, 80% completed
Macroeconomic analysis • Data collection • Data pre-processing • Machine learning analysis	• **Data collection**: pipeline completed, database continuously updated til the end of project period • **Machine learning analysis**: coding completed, analysis of results in progress
Sentiment analysis • Data collection • Data pre-processing • Machine learning analysis	• **Data collection**: pipeline completed, database continuously updated till the end of project period • **Machine learning analysis**: completed, analysis of results in progress
Documentation website • Feature tutorials and explanation of code implementations in the repository	• **Hosting & set-up**: completed • **"Part 1: Intro to algo trading"**: completed • **"Part 2: Core trading strategies"**: in progress, 70% completed • **"Part 3: Machine learning"**: to-be-completed

Table 7: Summary of the current status of different subparts in the project.

We initially encountered difficulties in collecting the data due to the upper threshold imposed on the API calls or GET requests. However, we resolved the problem by manually adding pauses in the program as a workaround to avoid hitting the maximum limit. Now with the data collection pipeline successfully tested and built, we could continue to update the database efficiently til the end of the project period.

6. Future Plan

In the upcoming semester, we will begin with Part 2 and work on integrating the findings in the three subparts (Microeconomic, Macroeconomic, Sentiment analysis). Moreover, we will work together to carry out virtual trading as well as to predict stock price movements with the mult-dimensional indicators. At the same time, we will continue to polish and update the documentation website and make amendments according to feedback we have received during the presentation. We also aim to refine the repository by refactoring some of the code and adding comments, so as to make sure everything featured in the repository is clean and organised for the users to read.

Table 8 briefly summarises the tasks to be completed in the next phase of the project.

Tasks	**Target deadline**
Execution of trade • Create a Paper Trading account • Make trading orders via calling the Broker's API	mid-February 2021
Stock movement prediction with multi-source features • Construction of a pipeline to build, test and evaluate the combined features • Experiment on algorithms to weigh and learn from features of different sources	mid-March 2021
Prepare deliverables • Polish and update documentation website • Code refactoring & Add comments • Final report & presentation	18th April 2021

Table 8: Tasks to be completed in Part 2 of the project.

7. Conclusion

To conclude, we have made one step forward towards the goal of democratising education and inducing algorithmic trading to be a more approachable topic. Currently, the university does not have any established syllabus nor database for algorithmic trading and research. Thus, we highlight the need for a database that consists of historical tick data, property prices data and sentiment analysis data. We have built a pipeline for downloading the data, so that the database could be updated continuously til the end of the project period. In addition to the database, the code implementation for each subpart also serves to exemplify how users could abstract trading logic into code. Furthermore, machine learning has also been covered in our repository in order to demonstrate how to leverage its power to make predictions about company status, stock trends and prices. All of the code comes along with explanation and equations in the documentation website that we have created.

Moving on, we will continue to explore machine learning models in predicting stock trends with the different data sources (microeconomic, macroecnomic, sentiment) we have collected. We will also add materials about the execution of trade orders in Hong Kong within the repository, so as to provide users a full picture of algorithmic trading in a practical setting.

We anticipate that this repository could fulfill our objectives in supporting the teaching and learning of algo trading. We have designed the code and explanation in the repository to be intuitive, easy to follow and reader-friendly. We hope that the repository could serve as an educational resource in the long term, and to assist learners to go from zero to hero in algorithmic trading.

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.

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*.

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.

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