Multi Agent System for Crypto-currency Trading Using Case Based Reasoning

Naoya Ohara, CFA, CAIA, A0197178L
(NUS Master of computing, general track)

A REPORT SUBMITTED FOR THE CAPSTONE PROJECT OF
MASTER OF COMPUTING, GENERAL TRACK,
NATIONAL UNIVERSITY OF SINGAPORE

2021

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Naoya Ohara, CFA, CAIA
13 September 2021

Table of Contents

Summary

As a capstone internship of Master of Computing (General Track), NUS, I created a multi agent system (MAS) for crypto-currency trading using case-based reasoning (CBR).

The purpose of creating this system is to offer the body-of-knowledge in algorithmic trading, to show how we can implement MAS and CBR in real world data, and to show how we can utilize the machine learning techniques such as PCA and kNN into real world's financial data, mainly aiming to the learning purpose for the future students and corporate sponsors at FinTech Lab (Please refer below picture too).

(Source: NUS Fintech Lab, Naoya Ohara)

For example, in the process of implementing this system, I introduced a train/test split using a sliding window for back-testing of financial data, the methodology that is often used by industry practitioners and in quantitative finance literature. Also, I implemented position sizing and risk management using Value at Risk (VaR). Further, I demonstrated the data preprocessing to utilize daily financial time-series data for trading buy/sell decisions.

In addition, I introduced how we can process natural language processing (NLP) and apply Fuzzy Logic to text data a.k.a alternative data. The topic includes the data preprocessing of text data and data conversion into quantified sentiment for buy/sell trading decisions. Throughout this process, one can learn the basics regarding how we can utilize alternative data in the context of investment decisions.

Regarding the machine learning techniques, I demonstrated how we can utilize Principal Component Analysis (PCA) for dimensionality reduction i.e. summarization of many macroeconomic data. Also, I utilized k-nearest neighbours (kNN) to implement CBR.

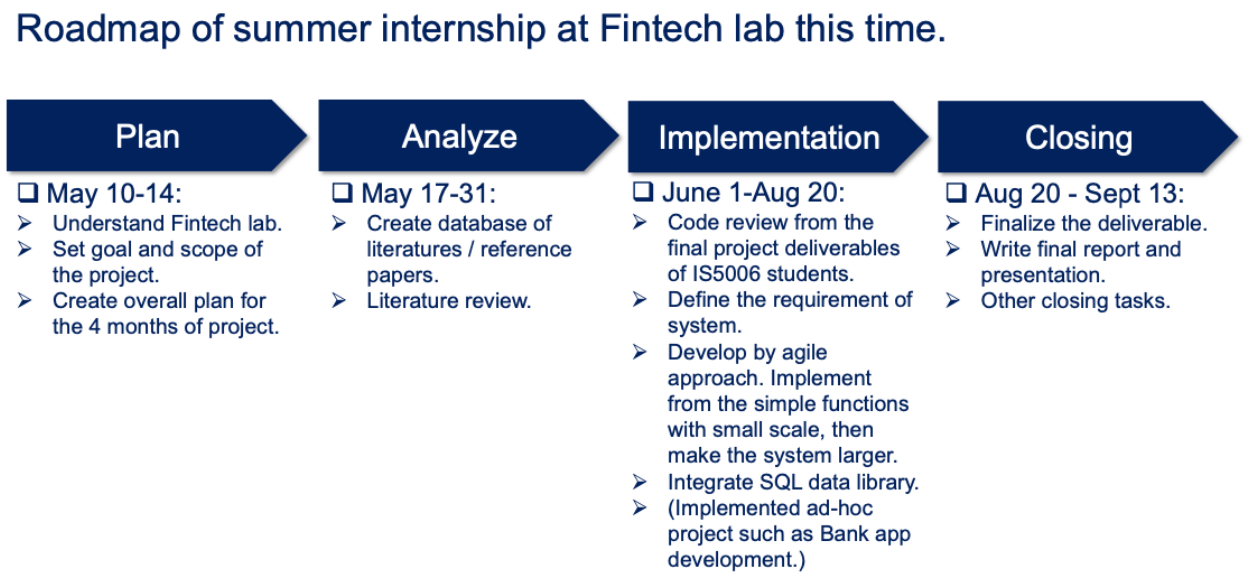
As a result, by learning through this system, one can learn how to implement MAS and CBR in the real world, how algorithmic trading works and its domain knowledge in finance, and how we can apply machine learning into the real world of financial market analysis.

Roadmap and development methodology of the project

1.Roadmap of the project

With regard to the development roadmap of multi-agent system at the capstone internship is as follows. It is a 4 months of full-time internship project. First part is to establish the database of relevant literature and reference papers gained from the students' teams projects at IS5006. In this process, I established the database for relevant 200+ of literature and research papers. Then, I also reviewed codes from the students' teams projects at IS5006 and extracted the best practices and pros/cons regarding the output from each team. Then, I started development of MAS for cryptocurrency trading. Meanwhile, I

also engaged in ad-hoc projects such as the development of a mobile Bank app in the NUS Fintech Lab, collaborating with other internship students at NUS Fintech Lab. After the latter part of August, I started the finalization and closing of the project.

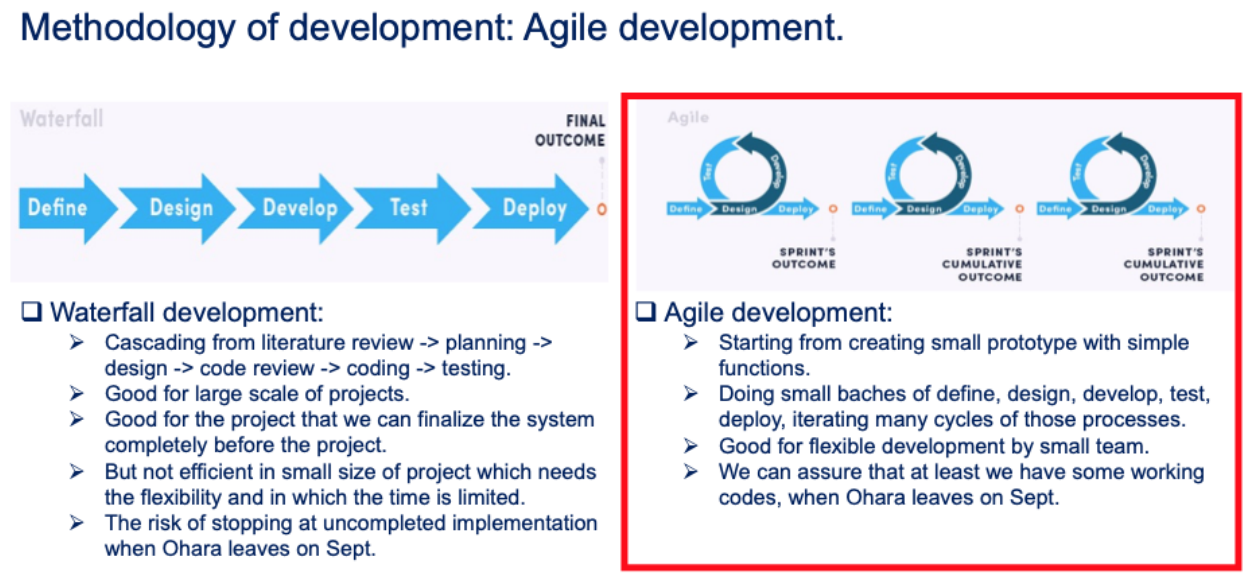


(Source: NUS Fintech Lab, Naoya Ohara)

2.Development methodology

With regard to the methodology of MAS development, I adopted Agile development. The main reason for adopting Agile approach is as follows:

1. The project this time should be done by one person with limited time-frame.
2. The system specification can change during the project such that I needed the flexibility in the project.



(Source: https://selleo.com/blog/agile-software-development-process-everything-you-need-to-know, Naoya Ohara)

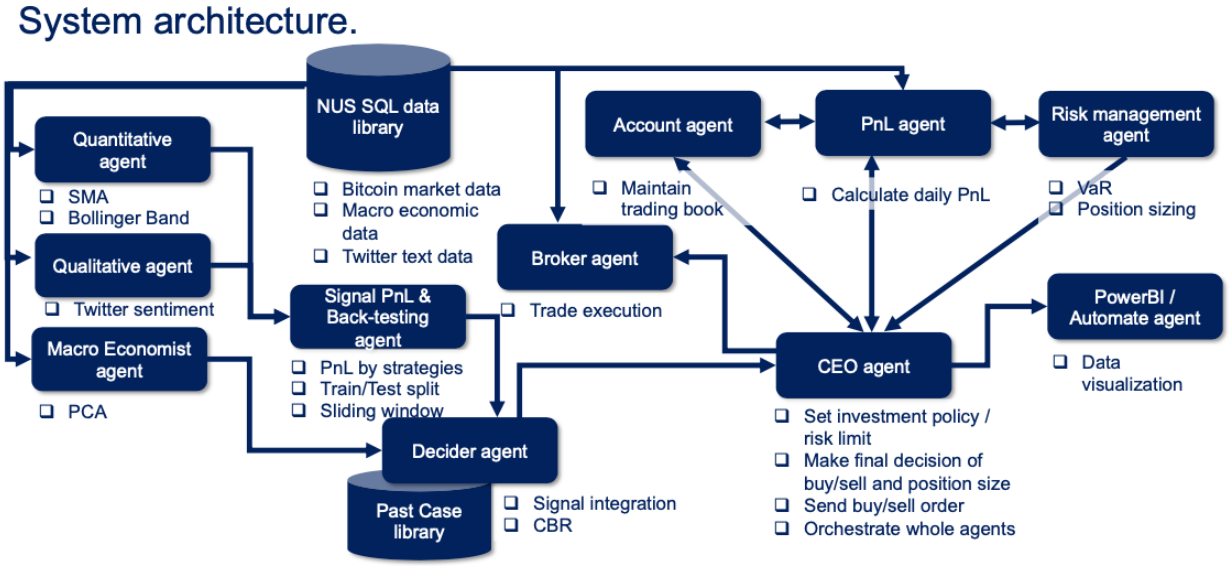
Regarding the development methodology, traditional Waterfall development is sometimes said to be completely obsolete or out-of-date. However, when the project size is large and we can plan the final state of the system from the beginning, still waterfall development can be an effective methodology. On the

other hand, the project this time is small sized with one person and the system specification can change over the time of development. In such a case, an Agile approach can be a more appropriate methodology. Nowadays, the consumer's preference and effective system specification can change in a relatively short period of time, such that many projects apply the Agile approach.

System architecture

1.Whole picture of system architecture

The system consists of different agents as follows.



(Source: NUS Fintech Lab, Naoya Ohara)

2.Brief explanation of each agent

Below is the list of agents that the system incorporates. From the next chapter, I will explain more details for each agent.

- **Account agent:** Account agent launches and manages account book data i.e. buy/sell transaction data, PnL data, and Net Asset Value (NAV) data etc.
- **Broker agent:** It establishes the connection with market data. Also, it enables the buy/sell trading execution.
- **Quantitative signaling agent:** It recommends buy and sell decisions based on simple moving average (SMA) and bollinger band.
- **Qualitative signaling agent:** It acquires and analyzes twitter text regarding bitcoin from major news sources and influential person's accounts (e.g. I picked Elon Musk who often mentions bitcoin), utilizing natural language processing (NLP). Then, the system converts the results from NLP into sentiment indicators then buy and sell recommendations using fuzzy logic.
- **Macro Economist agent:** Also, to understand market circumstances at the time, the system utilizes data from macro economics and other asset classes, such as data from equity market, interest rate, commodity market, VIX index and so on. Macro economist agent receives those data and summarizes the market situation with 3 factors, using Principal Component Analysis (PCA).

- **Signal PnL & Backtesting agent:** Those agents calculate the trading performance by each signaling agents, i.e. SMA, Bollinger band, and Twitter sentiment. Also, it implements back-testing utilizing train/test data split with sliding-window.

- **Decider agent:** Then, the decider agent aggregates those recommendations and macro economic information and makes the final recommendation to the CEO agent. In this process, the system utilizes case-based reasoning (CBR) to store and utilize historical prices, technical signals, and macro factor data with the resulting price direction on the following days. By referring to "same or similar combinations of quantitative and qualitative agent's buy/sell signal under similar macro economic circumstance", the CBR allows the trading system to generate better final recommendation. For the implementation of CBR, I utilized k-nearest neighbours (kNN), which is a popular method in machine learning and for implementing CBR.

- **PnL Agent:** This agent calculates daily PnL movement and returns it to the Account book and CEO agent.

- **PowerBI / Automate agent:** It can establish the connection with PowerBI, which enables us the data visualization of daily Account book and market data. Also, it can enable the automation of sending monthly reports via email by using Power Automate.

- **Risk management agent:** It implements Value at Risk (VaR), which is a popular method in market risk management. Also, it can support appropriate position sizing based on VaR metrics.

- **CEO agent:** It receives final investment recommendation from decider agent and VaR risk metric report from risk management agent, and decides the final action (buy/sell/do nothing/terminate trading) and position size of trading, based on investment and risk management policy that we set up before the trading as CEO.

Account agent (In the code: class Account(object))

```
Account.getTradeInfoDf().tail()
```

	time	side	exec_price	last_price	cost_price	pnl_costprice	quantity	num_coins	coin_value	cash	nav
207	2021-07-27	no-trade	0.0	39406.941406	0.0	0.0	0.0	0.0	0.0	972308.639463	972308.639463
208	2021-07-28	no-trade	0.0	39995.906250	0.0	0.0	0.0	0.0	0.0	972308.639463	972308.639463
209	2021-07-29	no-trade	0.0	40008.421875	0.0	0.0	0.0	0.0	0.0	972308.639463	972308.639463
210	2021-07-30	no-trade	0.0	42235.546875	0.0	0.0	0.0	0.0	0.0	972308.639463	972308.639463
211	2021-07-31	no-trade	0.0	41626.195312	0.0	0.0	0.0	0.0	0.0	972308.639463	972308.639463

	nav	dailyPnL	dailyPnLPct	totalPnL	totalPnLPct	VaR_annualized
972308.639463	0.0	0.0	-27691.360537	-0.027691	-0.0
972308.639463	0.0	0.0	-27691.360537	-0.027691	-0.0
972308.639463	0.0	0.0	-27691.360537	-0.027691	-0.0
972308.639463	0.0	0.0	-27691.360537	-0.027691	-0.0
972308.639463	0.0	0.0	-27691.360537	-0.027691	-0.0

(Source: NUS Fintech Lab, Naoya Ohara)

Account agent creates and maintains the account book shown above (Account.trade_info_df). It records daily trading activity (buy/sell/no-trade), purchased/sold price, # of bitcoin holding, NAV (net asset value = cash amount + market value of holding bitcoin), daily PnL, and annualized Value at Risk (VaR), etc. We can regard Account agent as the miniature of the back-office function in asset management companies.

Broker agent (In the code: class Broker(object))

Broker agent handles market data acquisition and buy/sell trading execution.

1.Market data acquisition

In terms of collecting market information, Broker class has the ability to get both historical and real time cryptocurrency price and trade information. When I started the system development, I gathered price data from yahoo finance and market exchange such as hitbtc using ccxt library. Then, NUS Fintech Lab decided to launch a SQL data library, such that I implemented a broker agent such that the system obtains data from MySQL data library. So far, this data library obtains daily OHLCV (Open-High-Low-Close-Volume) data from yahoo finance.

(Note: Through market exchanges using ccxt, we can obtain only recent data from around 2017. On the other hand, we can obtain bitcoin OHLCV data from 2014 by yahoo finance. For daily market analysis, at least we need 5-6 years of data. With such a background, yahoo finance is used as the data source this time.)

Broker agent maintains price data as Broker.longhist_price_df shown below. The system only uses closing prices, such that this dataframe just stores closing price data and daily volume. Then, the system adds day to day percent change and log-return too. (Note: For back-test purposes, industry practitioners often use log return, because it's easy to calculate cumulative return just by adding daily log return.)

```
Broker.longhist_price_df.head()
```

Date	close	volume	pct_change	log_return
2014-09-17	457.334015	21056800.0	NaN	NaN
2014-09-18	424.440002	34483200.0	-0.071926	-0.074643
2014-09-19	394.795990	37919700.0	-0.069843	-0.072402
2014-09-20	408.903992	36863600.0	0.035735	0.035111
2014-09-21	398.821014	26580100.0	-0.024659	-0.024968

(Source: NUS Fintech Lab, Naoya Ohara)

2.Trading execution

In terms of executing decisions, Broker agent has the ability to initialize the trade information stored in the Account Class, execute the trade according to the final decision made by the CEO agent, and update the account information.

With regard to the trading execution, as the simulation logic, the system assumed 0.1% of spread from closing price, i.e. we need to buy 0.1% higher than market close, while we need to sell 0.1% lower than market price. In the future, if NUS Fintech Lab establishes simulated market exchange or directly trades real money at market exchange, a successor can modify Broker agent by which the system enables us to buy/sell at the simulated or real market exchange.

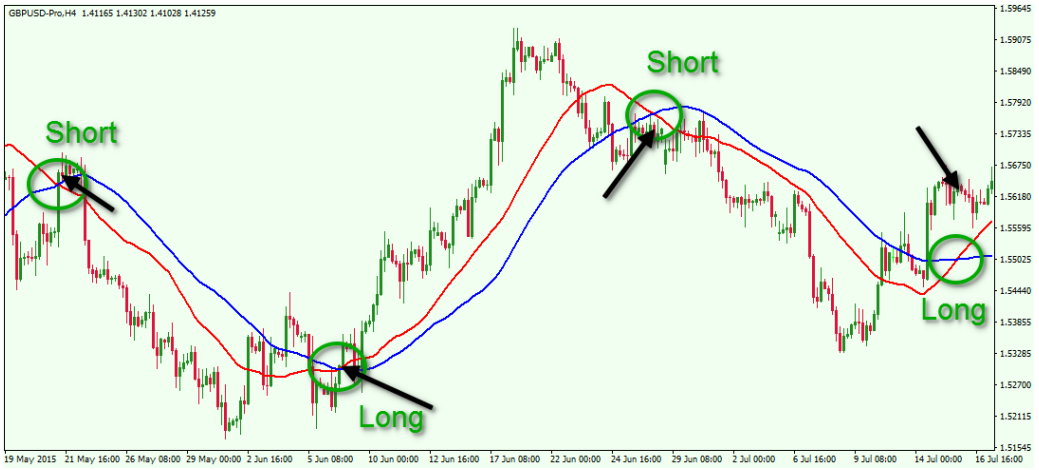
PnL agent (In the code: class PNLAgent(object))

Based on the raw data collected and recorded by the Account and Broker agent, PNL agent is the one who calculates and updates NAV and PnL information in the account book which is created and maintained by Account agent.

Quantitative signaling agent (In the code: class quantsSignal())

As a quantitative signaling agent, I implemented two different trading strategies, using different technical indicators. One is SMA crossover strategy, and the other is Bollinger bands strategy.

1.Simple Moving Average (SMA)



(Source: https://forextrainninggroup.com/anatomy-of-popular-moving-averages-in-forex/)

Simple Moving Average (SMA) is one of the most popular and common technical indicators in the technical analysis field. SMA is the average price over a given number of time periods. Then, we defined the golden-cross and death-cross. The golden-cross occurs when the short-term moving average (short-term MA) price goes over long-term moving average (long-term MA) to the upside and is interpreted as buying opportunities. Similarly downside moving average crossover constitutes the death-cross and is understood to signal a downturn in a market. SMA trading strategy is recognized as one of the most popular "trend-following strategies". Trend-following strategy can perform well if the underlying asset class often shows strong and long-term upward and downward trends. On the other hand, trend-following strategies can perform poorly if the market is within a narrow range i.e. consolidation. Daily trading signal data can be stored at the dataframe of self.agent1_hist_signal_df, and the decider agent will receive trading signal data from this dataframe.

2.Bollinger Bands

(Source:https://www.tradingwithrayner.com/bollinger-bands-trading-strategy/)

Bollinger Bands® are a technical analysis tool developed by John Bollinger for generating oversold or overbought signals. There are three lines that compose Bollinger Bands: A simple moving average (middle band) and an upper and lower band. The upper and lower bands are typically 2 standard deviations +/- from a 20-day simple moving average, but can be modified [1].

If the price goes up beyond the upper band, it is recognized as expensive such that it is a “sell” opportunity. As opposed, if the price goes down below the lower band, it is recognized as cheap such that it is a “buy” opportunity. We can set exit rules differently. For example, many practitioners set exit points when the price reverts back to the moving average. Or, we can also set exit points when the price reverts back between upper and lower bands. The system applies the latter rule for the exit.

As you can see, the above trading rule is based on the assumption of “mean reversion”, i.e. after the price goes up extremely it can go down and vice versa. Hence, this strategy can work well, when the market is in range bound i.e. in consolidation. On the other hand, when there exists a strong and long-term upward or downward trend, such a mean-reverting strategy cannot work well.

$$\text{BOLU} = \text{MA}(\text{TP}, n) + m * \sigma[\text{TP}, n]$$
$$\text{BOLD} = \text{MA}(\text{TP}, n) - m * \sigma[\text{TP}, n]$$

where:

$\text{BOLU} = \text{Upper Bollinger Band}$
$\text{BOLD} = \text{Lower Bollinger Band}$
$\text{MA} = \text{Moving average}$
$\text{TP (typical price)} = (\text{High} + \text{Low} + \text{Close}) \div 3$
$n = \text{Number of days in smoothing period}$
$m = \text{Number of standard deviations}$
$\sigma[\text{TP}, n] = \text{Standard Deviation over last } n \text{ periods of TP}$

(Source: https://www.investopedia.com/trading/using-bollinger-bands-to-gauge-trends/)

Above formulas are the formal notation of bollinger bands. In practice, TP can be simplified just by close price, such that the system applies this practically simple approach. In the back-test optimization, the system implements greedy search from the search space of n = [10, 20] and m = [2.0, 3.0].

Daily trading signal data can be stored at the dataframe of self.agent2_hist_signal_df, and the decider agent will receive trading signal data from this dataframe.

3.(For reference) Other technical indicators

In addition to the SMA and Bollinger Bands, several different technical indicators are invented and used by industry practitioners. For example, in Wikipedia, we can find 44 different technical indicators shown as follows.

Category:Technical indicators

From Wikipedia, the free encyclopedia

Pages in category "Technical indicators"

The following 44 pages are in this category, out of 44 total. This list may not reflect recent changes (learn more).

- Technical indicator

A
- Absolute currency strength
- Accumulation/distribution index
- Advance-Decline Data
- Average directional movement index
- Average true range

B
- Bollinger Bands

C
- Commodity channel index

D
- Detrended price oscillator
- Donchian channel
- Double exponential moving average

E
- Ease of movement

F
- Force index

I
- Ichimoku Kinkō Hyō

K
- Keltner channel

M
- MACD
- Mass index
- Momentum (technical analysis)
- Money flow index
- Moving average crossover
- Moving average envelope

N
- Negative volume index

O
- On-balance volume
- Oscillator (technical analysis)

P
- Parabolic SAR
- Put/call ratio

R
- Rahul Mohindar oscillator
- Range expansion index
- Relative currency strength
- Relative strength index
- Rising moving average

S
- Smart money index
- Stochastic oscillator

T
- KST oscillator
- Triple exponential moving average
- Trix (technical analysis)
- True strength index

U
- Ulcer index
- Ultimate oscillator

V
- Volume analysis
- Volume–price trend
- Vortex indicator

W
- Williams %R

Z
- Zero lag exponential moving average

(Source: Wikipedia)

You can search and use those indicators. However, you can also remember that the technical indicators are not science, but the metrics that are developed with practical needs to capture some aspect of the market and price movement. Still technical indicators are widely used even by institutional investors, such that we should know popular technical indicators as part of domain knowledge when we engage in the financial market. However, nowadays, those main usage is just for reference purposes and we can be aware of this point. The industry trends moves from searching for holy grail by technical analysis, toward applying more advanced statistical analysis and machine learning techniques. When you try to choose and use technical indicators, you can check each indicator in terms of following points:

- Does this indicator work as a trend-following, or mean-reversion algorithm?
- Which kind of price movement this indicator tries to capture to make profit?
- What does this indicator assume regarding price movement? (For example, price can keep going up or down with long-term trend, or price can revert to mean relatively in a short period of time, etc.)

One technical indicator rarely shows robust performance over a long period of time. Also, parameter tuning and optimization based on historical data tend to fall into overfitting. When we use technical indicators, we can be aware of those points.

4.(For reference) Normal class vs staticmethod

You may have noticed that **class quantsSignal()** is implemented by the usual class method, while previous Account, Broker, and PnL agents are implemented by so-called staticmethod with @staticmethod. I will briefly explain the difference between normal class and staticmethod, then how both are used in the system.

Usual class implementation (left) and staticmethod (right)

(Source: NUS Fintech Lab, Naoya Ohara)

As you can see, in the usual class implementation on the left, we can see the constructor of __init__ and we use the class by instantiation such as carToyota = Car(100,30). On the other hand, in the staticmethod on the right, we do not do instantiation and we call functions directly by Car.run() etc. Pros and cons for each implementation can be shown as follows:

The usual class implementation

Pros:

- It's normal. In many cases, codes are implemented as such.
- When we would like to create multiple instances based on the class, a normal class method with a constructor can be useful.

Cons:

- We need instantiation to utilize the class.

The staticmethod implementation

Pros:

- We do not need instantiation to utilize the class. We can call it directly.
- We can improve readability in some cases. For example, the reader can understand that **def run():** is used only under class **Car(object):** in the example.
- Especially, when we do not need to create many instances from the class and we just keep one whole object as one class, it's easy to handle and maintain.

Cons:

- It may not be a normal implementation. Some people may not know staticmethod.

- When we would like to create multiple instances based on the class, a normal class method with a constructor can be more useful.

Reflecting pros and cons of staticmethod and normal class, the whole system is implemented based on below criteria:

- If multiple instances should be created and maintained based on the class, the class is implemented as a usual class method. (This is the reason that quantitative agent is implemented by a normal class.)
- If only one object is required and we do not need to produce multiple instances by instantiation, the class is implemented as staticmethod. (This is why Account, Broker, and PnL agents are implemented by staticmethod.)

Qualitative signaling agent (Twitter agent) (In the code: class twitterSignal())

Qualitative anget is the agent which formulates buy/sell signals based on qualitative data such as text. In our implementation, we try to formulate buy/sell signals by analyzing twitter text sentiment and converting those into trading buy/sell signals, by implementing Twitter agent.

1.Data acquisition

First of all, we should obtain text data regarding bitcoin from twitter. As the text scraping tool for twitter, it looks that “tweepy” is widely used by many students in past IS5006 class. However, it has the limitation of data acquisition, such as the limitation of available data only for recent 7 days, etc. Considering such limitations, this time, the system applied the “snscrape” library to obtain tweet data several years ago.

Next, we need to set the scope of data acquisition. While we may be able to reach great insight by collecting all worldwide tweets regarding bitcoin and cryptocurrencies, gathering such tremendous amounts of data is computationally too expensive. Also, as we gather tweets from less popular and unreliable accounts, the quality of information can saturate and the amount of noise in text information can increase. With regard to this point, the sentiment of popular twitter news accounts such as NYTimes, Bloomberg, CNN News, etc, have high correlation with market price [2-4]. Based on those surveys, this time, we have chosen Wall Street Journal (WSJ), Bloomberg, Investing.com, CNN, NYTimes, Financial Times (FT), and Elon Musk. Elon Musk is not the news media but an individual, but nowadays his tweet has influential power toward bitcoin and other cryptocurrency’s price, such that I included his twitter account within data gathering scope. I collected data from 2014 with 7612 tweets, but many of the tweets are recent one after 2018, when bitcoin price showed a bubble and burst with large price movement. Then, we added those data into SQL data library. The system obtains those data from SQL database then stores data into the dataframe of self.tweets_hist_df.

2.Data preprocessing

In text sentiment analysis, data preprocessing takes a very important part. First of all, we need to clean up twitter text. For example, we remove strings starting with “@”, “#”, “$”. Also, we remove hyperlinks and URL, video, and “RT”, etc, the words which are not important to evaluate the sentiment. Also, we can eliminate stop-words. Stop-words means meaningless words such as “a”, “the”, “of”, and so on. The system executes those tasks by the functions of clean_tweet_text(self, tweetText) and removeStopwords(self, text), when twitterSignal class executes the function of dataPreprocessing(self). When we deal with stopwords, nltk (Natural Language ToolKit) library is popularly used, so I introduced

it in the system implementation. Those preprocessing can remove noise in text data, such that it affects the quality of sentiment analysis significantly.

3.Data transformation

In addition to the data preprocessing, we can transform text data to convert those into quantified sentiment data. For example, I introduced sentiment measurement of "polarity" and "subjectivity" by using the textblob library. Polarity is a float within the range [-1.0, 1.0], where 1 means positive statement and -1 means a negative statement. Subjectivity is a float within the range [0.0, 1.0] where 0 is objective and 1.0 is subjective. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information [5]. Obtaining polarity and subjectivity by using textblob is a very simple, but powerful way to start text sentiment analysis.

To move one step ahead for the text sentiment analysis, the system utilized Google Text Analytics in the function of googleSentiment(self, df). By utilizing this function, we can obtain text sentiment metrics such as "score" and "magnitude" [6].

- **Score:** It ranges between -1.0 (negative) and 1.0 (positive) and corresponds to the overall emotional leaning of the text.
- **Magnitude:** It indicates the overall strength of emotion (both positive and negative) within the given text, between 0.0 and +inf. Each expression of emotion within the text (both positive and negative) contributes to the text's magnitude (so longer text blocks may have greater magnitudes).

In addition, by using Google Text Analytics, we can obtain other information such as the category of the topic that the text mentions (e.g. finance, leisure, etc). As a result, we can gather data shown as follows.

testTwitter.tweets_df

	index	crypto	url	date	id	username	content	text_length	content_processed	polarity	subjectivity	label	text_processed_length
0	1799	bitcoin	https://twitter.com/business/status/1340922472...	2020-12-21 07:30:05	1340922472351330307	business	JPMorgan says the odds of a Bitcoin correction...	161	jpmorgan says odds bitcoin correction would in...	0.037500	0.637500	4	112
1	7472	btc	https://twitter.com/Investingcom/status/134099...	2020-12-21 12:11:41	1340993341916393472	Investingcom	Here are https://t.co/HMaYPuSux7's #Top5Things...	272	monday deal agreed new strain found uk st...	0.136364	0.454545	4	114
2	7471	btc	https://twitter.com/Investingcom/status/134102...	2020-12-21 14:06:15	1341022172014325761	Investingcom	*Bitcoin slumps 6% as new COVID-19 strain upse...	119	bitcoin slumps new covid strain upsets wider ...	0.136364	0.454545	4	59
3	5508	bitcoin	https://twitter.com/Investingcom/status/134102...	2020-12-21 14:06:15	1341022172014325761	Investingcom	*Bitcoin slumps 6% as new COVID-19 strain upse...	119	bitcoin slumps new covid strain upsets wider ...	0.136364	0.454545	4	59
4	164	bitcoin	https://twitter.com/WSJ/status/134159061918200...	2020-12-23 03:45:03	1341590619182002177	WSJ	Ripple said it will defend itself against a la...	199	ripple said defend lawsuit sec claims company ...	-0.050000	0.300000	2	121

score	magnitude	google_language	google_entities	google_categories
-0.2	0.2	en	OTHER,LOCATION,ORGANIZATION	/Finance/Investing/Currencies & Foreign Exchange
0.0	1.0	en	ORGANIZATION,OTHER,LOCATION,PERSON	
0.0	0.4	en	OTHER,NUMBER,LOCATION	None
0.0	0.4	en	OTHER,NUMBER,LOCATION	None
-0.5	0.5	en	OTHER,ORGANIZATION	/Finance/Investing

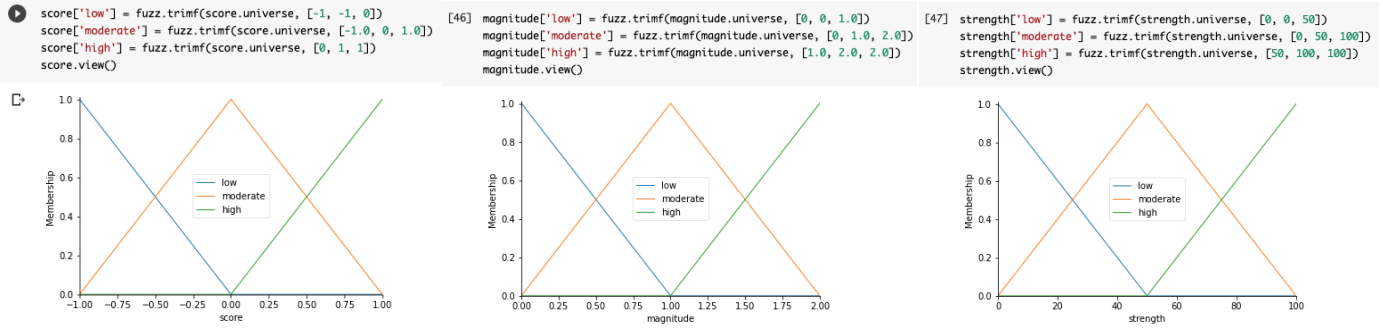
(Source: NUS Fintech Lab, Naoya Ohara)

4.Fuzzy logic

After obtaining sentiment metrics by using google text analytics, the system implements the fuzzy logic to convert "score" and "magnitude" into one integrated sentiment indicator of "strength".

First, we set membership functions for score, magnitude, and strength. For example, by setting membership functions for score and magnitude, we can define "how much of the score is regarded as low score i.e. negative, how much is mediocre i.e. neutral, and how much is positive" in a "fuzzy" manner.

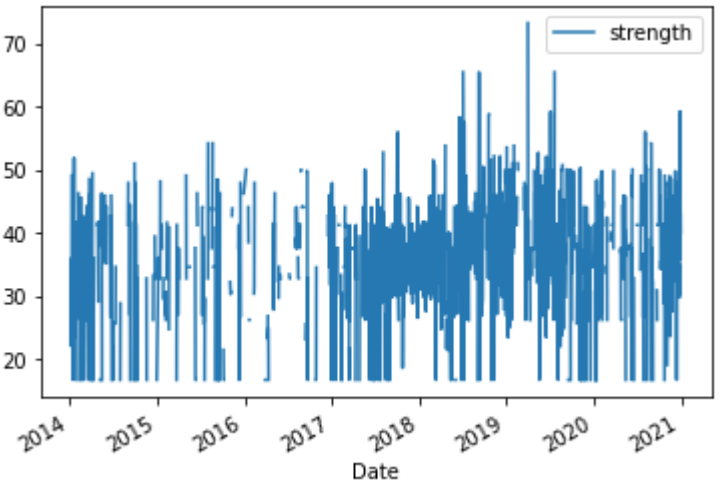
Then, the fuzzy logic can output the final integrated sentiment outcome as “strength” by combining “score” and “magnitude”.



(Source: NUS Fintech Lab, Naoya Ohara)

Fuzzy-logic is a relatively old idea which was introduced around 1965 and used for the temperature control of rice cookers and electric ovens. However, it is still useful when we deal with “vague” recognition, such as “hot” or “cold” for rice cookers and “positive” and “negative” in text tone. By utilizing fuzzy logic, we can deal with the vagueness in the use of language such as “positive” or “negative” tone in text.[7]

Finally, we can obtain a “strength” sentiment indicator between 0-100, shown below in the graph.



(Source: NUS Fintech Lab, Naoya Ohara)

By utilizing this indicator, we can make buy and sell decisions. For example, if the 5 days moving average of strength falls below 20, it shows negative sentiment such that we sell bitcoin. On the other hand, when the 5 days moving average of strength goes above 40, it indicates positive sentiment such that we buy bitcoin. The Twitter agent passes those decisions to the Decider agent.

Macro Economist agent (In the code: class macroEconomist(object))

From the past, many industry practitioners have combined technical indicators and macroeconomic data for better understanding of financial market circumstances and better trading decisions [8]. In this section, I introduce how we can utilize macroeconomic data for trading with the combination of technical indicators and twitter sentiment.

1.Macro economic data

I extracted macro economic data from FRED (https://fred.stlouisfed.org/) as follows. While some students tried to find the relationship between cryptocurrency price and minor macro economic data or

commodities such as soybeans etc, I chose popular macro and market indices that many industry professionals check market conditions of "risk-on" and "risk-off" sentiment. Following are the data that I gathered and utilized this time.

Date	NASDAQCOM	DGS10	TEDRATE	T10Y2Y	BAA10Y	DCOILWTICO	GOLDAMGBD228NLBM	VIXCLS
2019-01-01	6635.277	2.69	0.41	0.21	2.45	45.15	1281.65	25.42
2019-01-02	6665.938	2.66	0.42	0.16	2.45	46.31	1287.20	23.22
2019-01-03	6463.504	2.56	0.44	0.17	2.48	46.92	1287.95	25.45
2019-01-04	6738.855	2.67	0.43	0.17	2.45	47.76	1290.35	21.38
2019-01-05	6738.855	2.67	0.43	0.17	2.45	47.76	1290.35	21.38

(Source: FRED, NUS Fintech Lab, Naoya Ohara)

- **'NASDAQCOM' :** NASDAQ Composite Index, which is a major USA technology equity market index.
- **'DGS10' :** 10-Year Treasury Constant Maturity Rate. Interest rate of US government bonds with 10 years of maturity is often referenced as a major reference rate as risk free rate.
- **'TEDRATE' :** TED Spread, the spread between 3-month LIBOR and Treasury bills, which indicates perceived credit risk. LIBOR is the benchmark interest rate of London's inter-bank lending/borrowing. When credit crunch happens in the financial sector, LIBOR can increase rapidly, compared with the interest of 3 month US Treasury bills.

  (Note: While US dollar LIBOR is disclosed until 2023, it will not be used after that. We need to find an alternative index to check the credit situation in the banking sector.)

- **'T10Y2Y' :** The 10-year minus 2-year Treasury (constant maturity) yields: Positive values may imply future growth, negative values may imply economic downturns. Especially, if this figure becomes negative, it is called "inverse yield". In many cases, within 2 years after inverse yield, economic downturns and equity market crashes happened in the past.
- **'BAA10Y' :** Moody's Seasoned Baa Corporate Bond Yield Relative to Yield on 10-Year Treasury Constant Maturity. It shows the corporate sector's funding and credit circumstances. In the "risk-on" market, it decreases i.e. companies can easily issue corporate bonds with low interest rates, while in the "risk-off" market, it expands i.e. companies can feel difficulty in funding by issuing corporate bonds even with high interest rates.
- **'DCOILWTICO' :** Crude Oil Prices, West Texas Intermediate (WTI) - Cushing, Oklahoma. It is a major index of oil price. And oil price represents overall commodity price movements and global inflation. When the inflation rate goes up, oil prices tend to go up synthetically.
- **'GOLDPMGBD228NLBM' :** Gold price. Gold also constitutes an important portion in commodity asset classes. Gold price represents the "value of physical currency", compared with the US dollar which represents the "value of paper currency". When the risk of an emergency (such as war) rises, the gold price tends to rise. Also, when people expect high inflation, i.e. the depreciation of paper money's value, people tend to buy gold such that the gold price tends to rise.
- **'VIXCLS' :** CBOE Volatility Index: VIX. VIX measures market expectation of near term volatility conveyed by stock index option prices. It shows overall market sentiment of equity market participants. When many market participants expect that the equity market can be stable

in the near future, VIX stays at low range. When many market participants expect that the equity market can become unstable in the near future, VIX can go up.

There are many other market indices and macroeconomic indices. You can search for those and can utilize those for trading and better understanding of macro economy and market situation. When you try to do so, you can take note followings:

- **Understand and consider economic meaning:** Just searching for economic indices which show high correlation with bitcoin and applying this or that economic indices for trading are not reasonable ways. Spurious correlation can exist among many market indices. If you search hundreds of economic indices and check correlation or execute regression in different time periods, some of those can show high correlation in certain data periods just by chance, without reasonable economic meaning. For example, when you apply soybean futures for the analysis of bitcoin, you need to consider what is the relationship between soybean price and bitcoin price with regard to economic sense and investor behavior.
- **Data availability:** While all above indices are priced and disclosed everyday, some economic indices are not. For example, the US unemployment rate is a very useful macro economic indicator, but it is disclosed once every month. Also, GDP is a very popular macro economic statistics, but it is disclosed only quarterly and the number is revised after the first disclosure of the preliminary report. For daily trading purposes, indices with daily pricing and disclosure can be desirable in terms of timeliness and freshness of information. All above indices are chosen in terms of such data availability viewpoint.

2.Data transformation

After gathering the above data, the system does preprocessing and then implements data transformation by Principal Component Analysis (PCA).

First, the system does data filling. Macro economic data is usually priced and disclosed on Monday-Friday and holiday is closed i.e., around 250 trading days annually. On the other hand, bitcoin is traded every day i.e. 365 trading days annually. To fit both data comparable, the system converts annual 250 days of trading data into annual 365 days trading data, then executes fillna to fill by previous days data if the macroeconomic data is not available.

Then, the system calculates 30 days rolling % changes of macroeconomic data, using the function of rollingReturnCalc(preprocessed_df,days=30). The reason for taking 30 days % change and not taking daily % change is that the daily % changes of macroeconomics data is too noisy. By taking 1 month % change of macroeconomic data, the system can capture macroeconomic sentiment in a reasonable manner for practitioners as well.

Next, the system executes PCA by the function of pcaDimReduction(day, macro_pct_df). But before executing PCA, the system implements another data transformation, data normalization of MinMax scalar shown on the left hand side of the formula.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \qquad z = \frac{x - \mu}{\sigma}$$

μ = Mean
σ = Standard Deviation

0-1 (Min-Max) normalization (left) and z-normalization (right)
(Source:Wikipedia, Naoya Ohara)

Data normalization intends that normalized values allow the comparison of corresponding normalized values for different datasets in a way that eliminates the effects of certain gross influences. There are

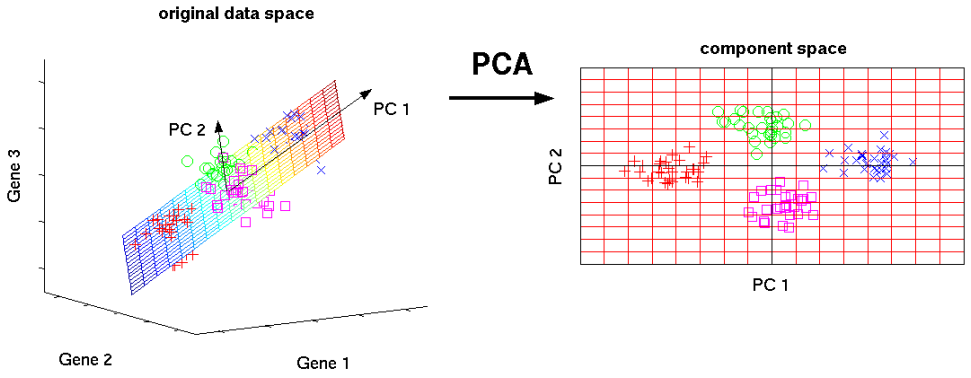
several ways to do data normalization, above 0-1 normalization (left hand side) and z-normalization (right hand side) are often used in data analytics.

- **0-1 (or Min-Max) normalization:** It can normalize a certain data set within the range between 0 -1, by using minimum value and maximum value in the dataset. By this normalization, the minimum data takes 0, while the maximum data takes 1.
- **Z-normalization:** It takes mean and standard deviation of the dataset, then converts raw data x into standardized data z in the above right hand side of the formula. In this case, most data can range between -3 to +3, by assuming normal distribution of the dataset. Often, more than +3 and less than -3 are truncated into +3 and -3 to eliminate outlier data, in the community of financial practitioners.

Regarding which way of normalization is used, it depends on the situation. It looks that the machine learning community likes 0-1 normalization, while financial practitioners such as quants analysts like z-normalization. Statistically, z-normalization can work well when we can assume the bell-curve i.e. normal distribution of the dataset. Also, if the analyst truncates the value beyond +3 and -3 into +3 and -3, the information of outlier can be lost. So, when we can assume a bell-curve of the dataset's distribution and we would like to eliminate outliers from analysis, z-normalization can work nicely. On the other hand, 0-1 normalization can work even when the dataset does not follow the normal distribution. Also, the information of outliers still remains in standardized 0-1 data. Therefore, if the data may not follow the normal distribution and the information of outliers can be important in the analysis, we can use MinMax normalization.

Normalization can be an important preprocessing step for many machine learning algorithms. If a feature has a variance that is orders of magnitude larger than others, it might end up dominating the estimator, which might not learn well from other features. In the system, as I explained, MinMax normalization is implemented.

3.Principal Component Analysis (PCA)



(Source:https://www.analyticsvidhya.com/blog/2016/03/pca-practical-guide-principal-component-analysis-python/)

Principal Component Analysis (PCA) is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. It is popularly used to reduce the dimensionality of the dataset. As PCA is interested in the components that maximize the variance, if one component varies less than another because of their respective scales, PCA can work wrongly, such that data normalization is executed beforehand, as mentioned above.

Finally, 8 macroeconomic data can be summarized by 3 factors, shown in the following picture. Macroeconomic agent pass those data to the Decider agent, such that the Decider agent can utilize those

macroeconomic data to capture the market circumstance for the day and reflect this insight when the system implements case based reasoning (CBR) as I will explain later.

Date	macro_factor1	macro_factor2	macro_factor3
2019-01-01	-0.891191	0.096499	0.378480
2019-01-02	-0.858942	0.064011	-0.299344
2019-01-03	-0.523940	0.178067	0.417479
2019-01-04	-0.141390	-0.660551	0.554137
2019-01-05	-0.179610	1.024015	0.176716

(Source: FRED, NUS Fintech Lab, Naoya Ohara)

Signal PnL and Backtesting agents (In the code: class stratPnL() and class backTest(object))

1.Signal PnL Agent: PnL Recording function for each signal agent

Signal PnL agent (in the system, class stratPnL()) is the agent to calculate profit and loss for each trading signal (i.e. SMA, Bollinger Bands, and Twitter sentiment) under certain levels of profit-taking and stop loss. For example, by receiving daily trading signals of Simple Moving Average strategy from quantitative agent, this agent calculates the PnL of a certain training and test period in the back test and daily trading activity. Below is the snapshot of the data.

```
CEO.movingAvg_strat.df.head()
```

Date	close	long_signal	short_signal	log_return	long	short	long_costprice	short_costprice	long_dailypnl_pct
2021-01-01	29374.152344	1.0	0.0	NaN	1.0	0.0	29403.526496	0.0	-0.001000
2021-01-02	32127.267578	1.0	0.0	0.089590	1.0	0.0	29403.526496	0.0	0.089590
2021-01-03	32782.023438	1.0	0.0	0.020175	1.0	0.0	29403.526496	0.0	0.020175
2021-01-04	31971.914062	1.0	0.0	-0.025022	1.0	0.0	29403.526496	0.0	-0.025022
2021-01-05	33992.429688	1.0	0.0	0.061280	1.0	0.0	29403.526496	0.0	0.061280

short_dailypnl_pct_nega	total_dailypnl_pct_posi	total_dailypnl_pct_nega	long_sumprofit	long_sumloss	short_sumprofit
-3.029772	9.430534	-7.093114	6.325383	-4.063342	3.105151
0.000000	0.089590	0.000000	6.414973	-4.063342	3.105151
0.000000	0.020175	0.000000	6.435148	-4.063342	3.105151
0.000000	0.000000	-0.025022	6.435148	-4.088365	3.105151
0.000000	0.061280	0.000000	6.496428	-4.088365	3.105151

long_cumsignal_pertrade	short_cumsignal_pertrade	long_cumpnl_pct	short_cumpnl_pct	long_recovperiod	short_recovperiod
0.0	0.0	-0.001000	0.0	0.0	0.0
1.0	0.0	0.088591	0.0	0.0	0.0
2.0	0.0	0.108766	0.0	0.0	0.0
3.0	0.0	0.083743	0.0	1.0	0.0
4.0	0.0	0.145023	0.0	0.0	0.0

short_drawdown	total_drawdown	long_recovery_flag	short_recovery_flag	total_recovery_flag	long_position	short_position
0.0	0.000000	0	0	0	NaN	NaN
0.0	0.000000	0	0	0	0.0	0.0
0.0	0.000000	0	0	0	0.0	0.0
0.0	-0.025022	1	0	1	0.0	0.0
0.0	0.000000	0	0	0	0.0	0.0

short_dailypnl_pct	total_dailypnl_pct	long_dailypnl_pct_posi	long_dailypnl_pct_nega	short_dailypnl_pct_posi
0.0	-0.001000	6.325383	-4.063342	3.105151
0.0	0.089590	0.089590	0.000000	0.000000
0.0	0.020175	0.020175	0.000000	0.000000
0.0	-0.025022	0.000000	-0.025022	0.000000
0.0	0.061280	0.061280	0.000000	0.000000

short_sumloss	total_sumprofit	total_sumloss	long_PF	short_PF	total_PF	long_cumpnl_pct_pertrade	short_cumpnl_pct_pertrade
-3.029772	9.430534	-7.093114	1.556695	1.02488	1.329534	-0.001000	0.0
-3.029772	9.520124	-7.093114	1.578743	1.02488	1.342164	0.088591	0.0
-3.029772	9.540300	-7.093114	1.583708	1.02488	1.345009	0.108766	0.0
-3.029772	9.540300	-7.118137	1.574015	1.02488	1.340280	0.083743	0.0
-3.029772	9.601580	-7.118137	1.589004	1.02488	1.348889	0.145023	0.0

total_recovperiod	total_cumpnl_pct	long_cumpnl_max	short_cumpnl_max	total_cumpnl_max	long_drawdown
0.0	-0.001000	-0.001000	0.0	-0.001000	0.000000
0.0	0.088591	0.088591	0.0	0.088591	0.000000
0.0	0.108766	0.108766	0.0	0.108766	0.000000
1.0	0.083743	0.108766	0.0	0.108766	-0.025022
0.0	0.145023	0.145023	0.0	0.145023	0.000000

(Source: NUS Fintech Lab, Naoya Ohara)

While I do not explain all columns one by one, the summary of the above PnL calculation process is shown as follows.

- First, this agent receives buy/sell signals from quantitative or qualitative agent (column long_signal and short_signal).
- When the long (short) signal turns from 0 to 1, trade starts.
- When any of the following criteria is met, trade is exited.
    - The long (short) signal moves back to 0.
    - Opposite signal arises i.e., when buying, short_signal turns from 0 to 1 and vice versa.
    - The price reaches a certain profit taking or loss cut point. (For example, if the system sets (profit_taking, loss_cut) = (0.2, 0.1), the trade is closed when the price increases by 20% from cost price or the price decreases by -10% from cost price.)
- This agent records cumulative profit of profitable trades and cumulative losses from loss-making trades. Then, this agent calculates "Profit Factor" for long position, short position, and long/short total. (Regarding the Profit Factor, I will explain it later.)
- Also, this agent records "Drawdown", the loss from its peak performance. (Regarding the details, I will explain later.)
- Transaction cost of buying/selling is assumed as 0.1% of the price for each trade. (Regarding the notion of "transaction cost", I will explain details later.)

In the next chapter, I will explain some important concepts in financial trading, such as "Profit Factor", "Drawdown", and "transaction cost".

2.Profit Factor

As mentioned, stratPnL() class calculates Profit Factor (PF) of each strategy. From the definition by [9], PF is shown as follows.

$$Overall\ PF = \frac{\sum pips\ won}{\sum pips\ lost} - 1$$

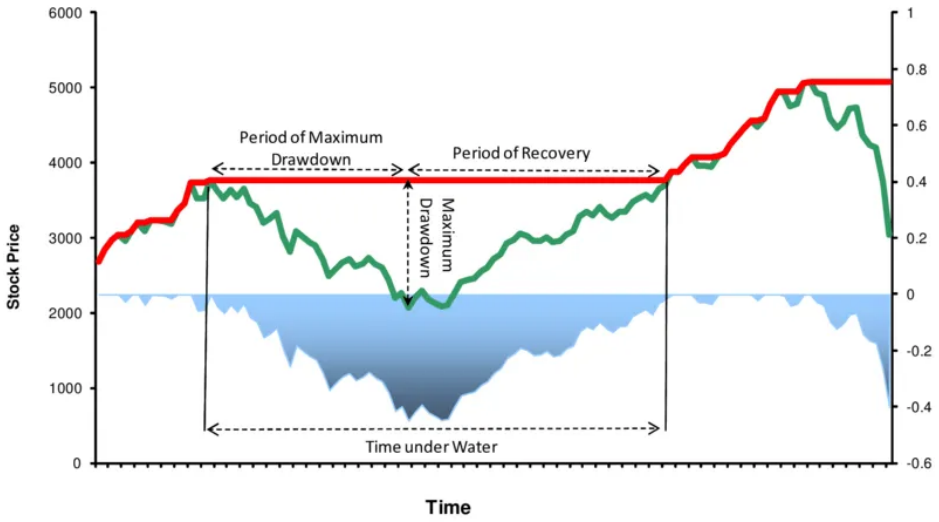
$$Long\ PF = \frac{\sum pips\ won\ when\ predicting\ up}{\sum pips\ lost\ when\ predicting\ up} - 1$$

$$Short\ PF = \frac{\sum pips\ won\ when\ predicting\ down}{\sum pips\ lost\ when\ predicting\ down} - 1$$

However, in the actual workplace, PF is often calculated just with the gross profit divided by the gross loss (including commissions) for the entire trading period i.e. "-1" in the formula can be omitted and PF can take positive value (if profit is zero and only loss, PF=0. If there are tons of profit with very small loss, PF = very large number) [10]. To prevent PF taking extremely large values, the system capped PF <= 5.0. PF measures the profit per unit of risk, with PF >= 1.0 indicating a profitable trading strategy. The system utilize PF as the profitability measurement of each trading strategy (i.e. SMA, Bollinger Bands, and Twitter Sentiment), such that the system determine the weightage of how much we rely on each

strategy's recommendation at Case Based Reasoning in the Decider agent (I will explain the calculation of this weightage using PF late at the section of the Decider agent).

3.Drawdown



(Source: https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/maximum-drawdown/)

In the system, stratPnL() class also calculates the "drawdown" of each strategy. "Drawdown" means the amount of loss from its peak performance. "Time under water" a.k.a. "Recovery period" shows total period of time from the day when the fund/strategy began to make a loss from peak, to the day when the fund/strategy could recover its loss (i.e. "going back above water"). Of course, a shallow drawdown with fewer days of Time Under Water is better for investors, such that industry practitioners in fund industry often check those figures when they evaluate the performance of funds/trading strategies. The stratPnL() class calculates those figures to visualize them into a graph.

4.Transaction cost (and its estimation)

In the PnL calculation for each trading strategy, transaction cost is estimated as 0.1% of the closing price. However, formally, the "transaction cost" can be decomposed as follows:

- Exchange fee
- Bid-Ask spread / Price
- Market impact i.e. the cost arising from the situation where we should take a more expensive ask than best ask (when buying) and a cheaper bid than best bid (when selling) if the buy/sell amount becomes larger.
- Opportunity cost i.e. cost regarding orders not fulfilled.

And Trading cost can be modeled by following formula [11].

$$Tcost = x + y + MI + OC$$

$$Tcost = a + c \cdot \sigma \cdot \sqrt{\frac{V_{trade}}{V_{daily}}} + OC$$

(Where:
Tcost = total transaction cost,
x = exchange fee,
y = bid-ask spread/price,
MI = Market Impact Cost=c*σ*√Vtrade/Vdaily,
OC = Opportunity Cost, in transaction cost estimate, usually 0.0 is used because we assume all trade can be executed in the transaction cost estimation.
a = x+y (i.e. observable cost),
c = market impact metrics (usually, 1.0 is used),
σ = Daily market volatility,
Vtrade = Volume of this fund's/strategy's trade in a day,
Vdaily = daily market volume.)

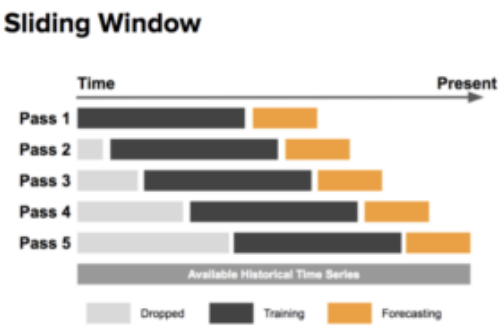
(Source: Reference [11])

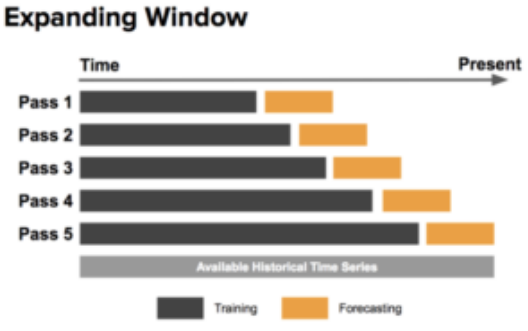
While it looks difficult with math equation, the key point is as follows:

- Total transaction cost can be decomposed into **observable cost** i.e. exchange fee x + bid-ask spread/price y, and **unobservable cost** i.e. Market Impact cost and Opportunity Cost.
- In unobservable cost, while we assume Opportunity Cost as zero in the transaction cost estimation, market impact cost cannot be zero in the real world.
- Market Impact cost can rise by 1.volatility of the market increases, and 2.the fund size becomes larger and transaction volume i.e. Vtrade increases, compared with market liquidity Vdaily.

In a rigorous sense, I have to note that we should estimate transaction cost by the above model for transaction cost estimation (Actually, institutional investors usually estimate transaction cost by using a model like this.). However, as a starting point of learning algorithmic trading, the above estimation sounds too complicated for most small individual investors who do not have to worry so much about market impact cost (i.e. Vtrade / Vdaily is very close to zero for small investors), such that I just simplified the transaction cost as a fixed number of 0.1%. If you start your career at a fund management company, I recommend you to calculate transaction cost estimation in a rigorous manner using models like the above, because the market impact cost can become an important matter if you work as an institutional investor with a large amount of funds.

5.Train/Test split, using sliding window





(Source: https://stackoverflow.com/questions/56601488/is-there-a-way-to-get-a-sliding-nested-cross-validation-using-sklearn)

Until the last section of explanation regarding the **class stratPnL()**, now we can calculate PnL for each trading strategy.

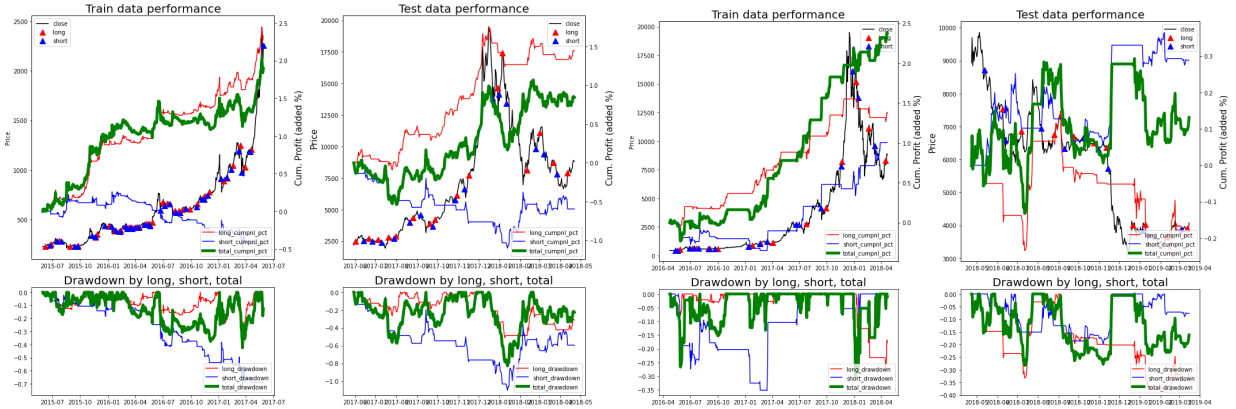
Then, to implement back-test correctly, we need to implement the "Sliding Window" (aka "Work Forward") shown in the above picture. The system implemented it at **class backTest(object)**.

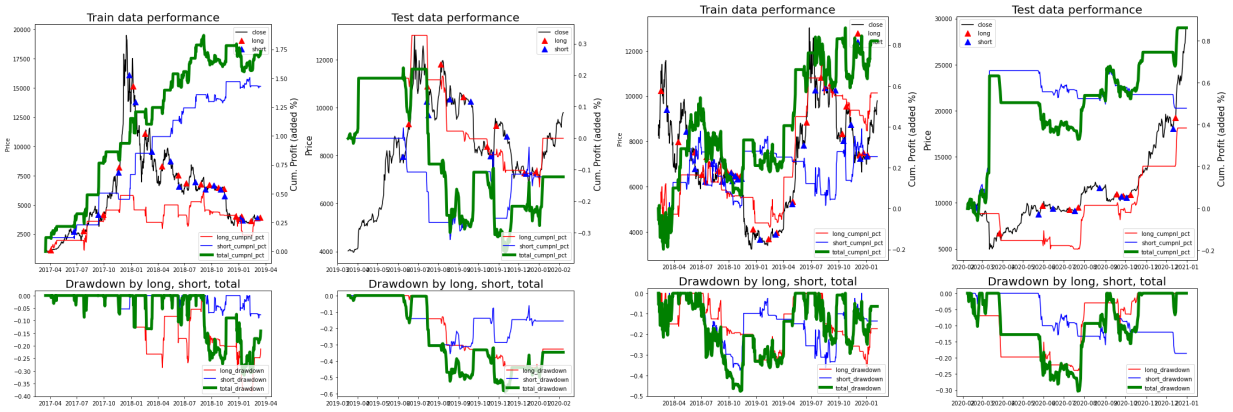
By training data, we optimize the parameters of strategy (For example, in SMA, day combination of short-term moving average and long-term moving average) to obtain best return. Then, in the test or forecasting period, we check whether an optimized strategy can work in an out-of-sample period. By doing so, we can avoid overfitting to past data and can check whether this strategy can work in the future. Train/test split using sliding windows is important to do back-test and build machine learning models for time-series data. With regard to how to slide train/split windows, there are 2 ways.

- **Sliding window:** One is the sliding window, shown on the left hand side of the picture below. In this case, both training periods are fixed (i.e. training period: 2 years, test period: 1 year, for example). Then, we slide train/test period.
- **Expanding window:** The other is the expanding window, shown on the right hand side of the picture below. In this case, the training period spans from the starting time until immediate before the test period. It means that the period of training data becomes longer and longer, as we test recent data.

While some quants' finance books suggest the expanding window (For example, [12]), many industry practitioners and quantitative finance literature (for example, [13]) apply the sliding window as industry standard. We followed industry standards by applying the sliding window in the system.

Below are the examples of performance in the training period and test period with 4 sets of sliding windows for the SMA strategy. Former 2 years of data is used as a training period to optimize parameters of short term moving average and long term moving average. Also, with the optimized parameters, the last 1 year is traded as a test period. Then, the system slides a dataset of 2 years of training data and 1 year of test data, by 1 year ahead. As you can see, in the training data period on the left hand side of the graph, the performance is great i.e. total cumulative profit of the green line goes upward in a constant manner because it is the performance with the optimized parameters after try-and-errors by greedy search. However, in the test data period on the right hand side, the performance is not as great as was in the training data period.





(Source: NUS Fintech Lab, Naoya Ohara)

Above result shows typical reality in algorithmic trading. Many traders optimize the parameter by past data such that the back-testing performance i.e. training data's performance looks great. However, in reality, when we apply the optimized parameter into future data for real trading, the performance cannot be as good as was in the test data. It is a popular phenomena called "overfitting" toward training data. Many trading systems just show great performance in the training data period, but most of them cannot be reproducible in the real trading for the future coming data. Searching for the robust trading model which shows constant performance in the both training and test period can be the important theme for the quantitative researchers, while it is very challenging.

6.Parameter optimization by sharpe ratio

Next, we should determine the criteria for the parameter optimization. One way is just to maximize the return in the training period. For example, we can choose the combination of short-term moving average days and long-term moving average days (like short-term MA = 10 day, long-term MA = 20 days), the combination by which we can maximize the return in the training period and apply those parameters into the test period. However, it is not the industry standard. When we consider the return, we also take into account how much we take the risk to obtain this return. In this context, Sharpe Ratio is popularly used in the financial industry to evaluate the performance of certain trading strategies or funds, shown as follows. In the system, **def generateBenchmarkComp(df, annualtradedays=365)** in **class backTest(object)** calculates it.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

where:

R_p = return of portfolio

R_f = risk-free rate

σ_p = standard deviation of the portfolio's excess return

(Source: https://www.investopedia.com/terms/s/sharperatio.asp)

There are some tips and industry customs when we calculate Sharpe ratio.

- **Annual return:** Usually, Rp is stated as an annual return. Just summing up the daily log returns for 1 years, it can become the annual return.
- **Annual volatility:** Also, σp is stated as an annual volatility. By converting daily volatility based on daily % changes into annual volatility, the formula is σ_annual = σ_day * sqrt(trading days for 1 year i.e. 365 in bitcoin's case). When it comes to equity which we do not trade Saturday/Sunday and national holiday, usually 250 days are used i.e. σ_annual = σ_day * sqrt(250).[14]
- **Treatment of Rf:** Risk-free rate can be often omitted and σp can be just calculated as the portfolio's volatility (not the volatility of the portfolio's excess return, i.e. σ(Rp - Rf)) in industry custom, because nowadays the risk free rate tends to be low and because the calculation of Rp /

σp can be just simpler. By following such industry custom, the system calculates the Sharpe Ratio of trading strategy as Rp / σp, by omitting Rf.

The system optimizes parameters of each strategy to maximize the annualized Sharpe Ratio by greedy search. For example, in SMA strategy, the system tries (short-term MA, long-term MA) = (5,10),(5,20),(10,20) and calculates the sharpe ratio by using each (short-term MA, long-term MA) under training data period. Then, the system applies the parameter with best performance at train data into test data.

7.Multi threading

While it is a technical matter, I introduced multi-threading at the implementation of **def backtestWithSlidingWindow(start,end)** in **class backTest(object).** By multithreading, the system is able to process multiple threads of codes or functions concurrently. With regard to multi threading, it can be fast to understand with the below example.

- Noodle cooking code: Below 2 functions work to boil noodles and prepare soup with given seconds of preparation time.

```
def boil_noodle(seconds):
  print('Boiling noodle for {} secondes\n'.format(seconds))
  time.sleep(seconds)
  print('Noodle is boiled \n')

def prepare_soup(seconds):
  print('Preparing soup for the noodle for {} seconds \n'.format(seconds))
  time.sleep(seconds)
  print('Soup is prepared \n')
```

(Source:NUS Fintech Lab, Naoya Ohara)

- No multi-threading vs With multi-threading

Then, we can see the difference between without multithreading and with multithreading. Left hand side below shows the operation without multithreading. The system operates boil_noodle for 3 seconds and the noodle is boiled, then the system runs prepare_soup for 2 seconds. In such a way, noodles get soggy and are not tasty.

On the other hand, Right hand side below shows the operation with multithreading. First, we create threads by setting target=function name and args=(variables to pass to the function). Then, the system starts threads by threads.start() and waits until both threads are completed by threads.join(). With the multithreading, we can see that boiling noodles and preparing soup are operated concurrently, such that noodles can be boiled immediately after the soup is prepared and we can enjoy tasty noodles.

```
print('Start cooking noodle')
boil_noodle(3)
prepare_soup(2)
print('Put noodle and soup together in the bowl')
print('Ramen is done!')
```

```
Start cooking noodle
Boiling noodle for 3 secondes

Noodle is boiled

Preparing soup for the noodle for 2 seconds

Soup is prepared

Put noodle and soup together in the bowl
Ramen is done!
```

```
print('Start cooking noodle')
#Creating threads
thread1 = threading.Thread(target=boil_noodle, args=(3,))
thread2 = threading.Thread(target=prepare_soup, args=(2,))
#Starting thread 1
thread1.start()
#Starting thread 2
thread2.start()
#Wait until the execution of thread 1 is done
thread1.join()
#Wait until the execution of thread 2 is done
thread2.join()
#Both thread 1, 2 are done
print('Put noodle and soup together in the bowl')
print('Ramen is done!')
```

```
Start cooking noodle
Boiling noodle for 3 secondes

Preparing soup for the noodle for 2 seconds

Soup is prepared

Noodle is boiled

Put noodle and soup together in the bowl
Ramen is done!
```

Pictures: WITHOUT multithreading (left hand side) and WITH multithreading (right hand side)
(Source:NUS Fintech Lab, Naoya Ohara)

As the bottom line, by using multithreading, we can run many functions concurrently. In algorithmic trading, there can be the situation when we need to run multiple agents together, especially if the data time-horizon becomes shorter than a day, i.e. trading with 5 minutes of price tick data. In such a situation, we can utilize multithreading.

Note: Under the Python environment, the system operates each thread just with a single core CPU by switching between threads. So, multithreading by Python may not contribute to the faster operation. Rather, it can become slower due to the additional tasks of switching operations among the working threads. If we would like to implement very short term trading by using minute ticks or shorter, such operational speed can become the issue and we may need to implement the algorithm by Java (faster than Python, but slower than C++) or C++ (faster than Python and Java, but the complexity of implementation can rise and there may not be so much available engineers who can write C++). In the area of HFT (High Frequency Trading) in which the algorithmic system trades equities just by microseconds, mainly C++ or FPGA are used.

Decider agent(In the code: class cbrDecider(object))

1.Overall function

The decider agent receives daily buy/sell trading signals from quantitative agent (SMA and Bollinger Bands) and qualitative agent (Twitter sentiment strategy). Also, the decider agent receives daily 3 macroeconomic factor data calculated by the macro economist agent. Then, the decider creates the final recommendation based on signaling agents, macro economic factors, and CBR's case retrieval. Finally, the CEO agent will receive the final recommendation from the decider agent to take final action.

2.Logic of combining buy/sell decisions from 3 different agents

First, the Decider agent receives buy/sell recommendations of SMA, Bollinger Bands, and Twitter sentiment from quantitative and qualitative agents. To merge different trading signals into one final recommendation, I defined the following formula.

$$finalrecommendation = \sum_{i=1}^{n} w_i * (SignalLong_i - SignalShort_i)$$

Where:

$$n = number\ of\ signals.\ In\ the\ system, n = 3.$$
$$i = each\ signal.\ SMA = 1, Bollinger\ Bands = 2, Twitter\ sentiment = 3$$
$$SignalLong_i = 1\ if\ long\ signal\ triggered, else\ 0$$
$$SignalShort_i = 1\ if\ short\ signal\ triggered, else\ 0$$

$$w_i = \frac{PF(total)_i}{\sum_{i=1}^{n} PF(total)_i}$$

$PF(total)_i$
$= signaling\ agent\ i's\ total\ (i.e.\ including\ both\ long\ and\ short)\ Profit\ Factor$

$$\sum_{i=1}^{n} PF(total)_i = sum\ of\ PF(total)_i, such\ that\ \sum_{i=1}^{n} w_i = 1$$

(Source:NUS Fintech Lab, Naoya Ohara)

Final recommendation is the weighted average of (SignalLong - SignalShort), in terms of the weight w calculated by the Profit Factor, which is calculated at the section of **Signal PnL and Backtesting agents, 2.Profit Factor**. Above equation looks scary, but it just means that, if the Profit Factor from a certain agent is high, the weightage of this trading strategy becomes high.

For example, in a certain day, we assume below data is obtained:

SMA agent has PF=2.0 (very profitable)
Bollinger Bands has PF=1.0 (zero profit or loss)
Twitter sentiment has PF=0.5 (loss is larger than profit)

Then, weightages for each strategy are shown as follows:

Weightage of SMA = 2.0 / (2.0+1.0+0.5) = 0.5714
Weightage of Bollinger Bands = 1.0 / (2.0+1.0+0.5) = 0.2857
Weightage of Twitter sentiment = 0.5 / (2.0+1.0+0.5) = 0.1428

Below charts shows the example of the movements for PF and weights regarding each agent (agent1=SMA, agent2=Bollinger Bands, agent3=Twitter sentiment).

(Lhs: Example of PF movement for each agent, Rhs: Example of weights movement for each agent)
(Source:NUS Fintech Lab, Naoya Ohara)

And today, for example, if SMA shows buy, Bollinger Band shows neutral (both buy/sell signals are 0), and Twitter sentiment shows sell, then the finalrecommendation will become as follows:

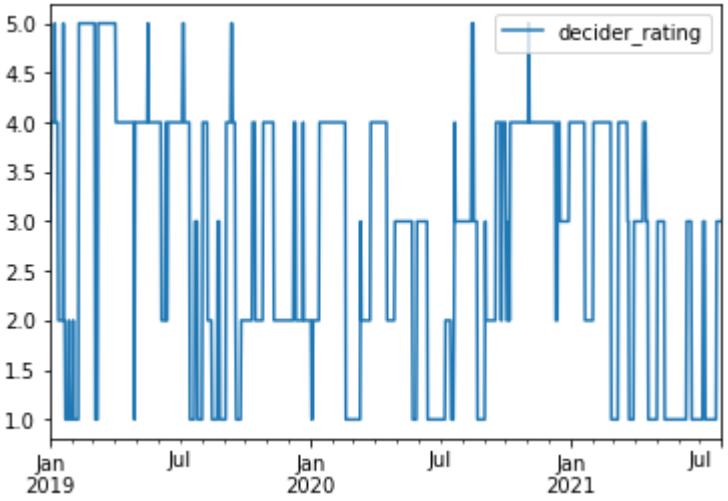
finalrecommendation = 0.5714 * (1 - 0) + 0.2857 * (0 - 0) + 0.1428 (0 - 1)
= 0.5714 + 0 - 0.1428 = 0.4286

As you can infer, when all trading signals show buy recommendation, finalrecommendation is equal to 1.0, while when all trading signals show sell recommendation, finalrecommendation is equal to -1.0. Also, when all signals are neutral, finalrecommendation = 0.0.

Then, the system convert "finalrecommendation" into 5 (strong buy), 4 (buy), 3 (neutral), 2 (sell), and 1 (strong sell) with following criteria. In above example of 0.4286, it becomes Buy (4) rating:

finalrecommendation > 0.5: Strong buy 5
0.1 < finalrecommendation <= 0.5: Buy 4
0.1 < finalrecommendation <= 0.1: Neutral 3
-0.5 < finalrecommendation <= -0.1: Sell 2
finalrecommendation <= -0.5: Strong sell 1

Below chart shows the trading decisions made by the Decider, which reflects PF and weights for each agent shown in previous charts.



(Example of 1-5 rating)

(Source:NUS Fintech Lab, Naoya Ohara)

Those calculations are implemented at the functions of **def receiveSignal(agent1_hist, agent2_hist, agent3_hist),def mergeSignal(agent1_hist, agent2_hist, agent3_hist),def recommendationConverter(x),** and **def deciderRecommendation(allsignals_df)** in **class cbrDecider(object)**.

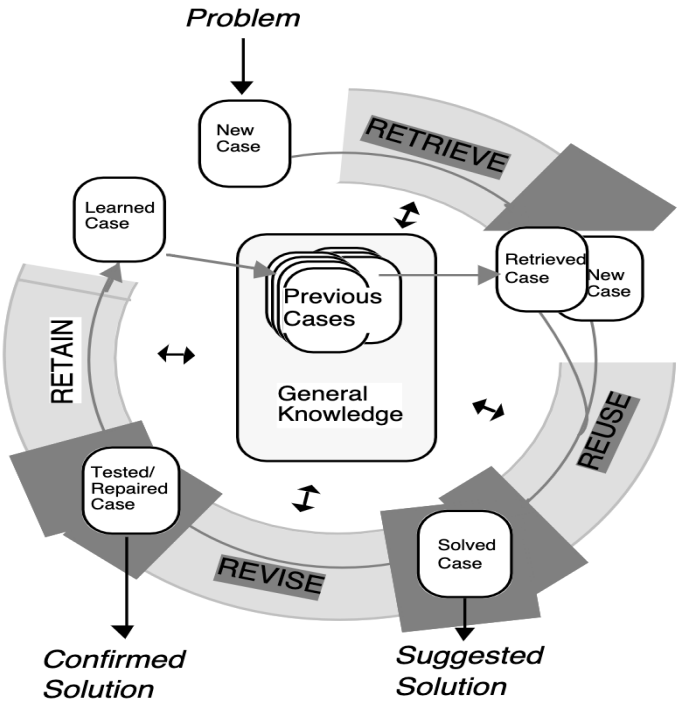
Note: Above formula around the "finalrecommendation" are original implementation from the author of this paper , and not written in finance text. However, once you learn basic concepts of finance and programming, you can produce such an original formula or indicators on your own. When you create your own formula or indicators, you can note that it's important for us to think about the logical or economic meaning of your formula or indicators. Such an original idea can become your differentiating factor in the financial market.

3. Case Based Reasoning (CBR)

In the previous section, we could calculate the "final recommendation" by integrating 3 signals from SMA, Bollinger Bands, Twitter Sentiment.

However, so far, we are not sure whether the certain day's signaling combination (e.g. SMA=buy, Bollinger=neutral, Twitter=sell in previous example) showed great performance after the day (let's say, t+1 to t+10 days from certain day t). Also, the performance can depend on the macroeconomic environment at a certain day t. To confirm such points, the system then applies Case Based Reasoning (CBR), with adding 3 macroeconomic factors data obtained by macro economist agent.

CBR is a relatively mature methodology in the Artificial Intelligence domain. The roots of case-based reasoning in AI is found in the works of Roger Schank and his students such as Janet Kolondner at Yale University in the early 1980s [15]. In the 1990s, interest in CBR grew internationally, as evidenced by the establishment of an International Conference on Case-Based Reasoning in 1995.

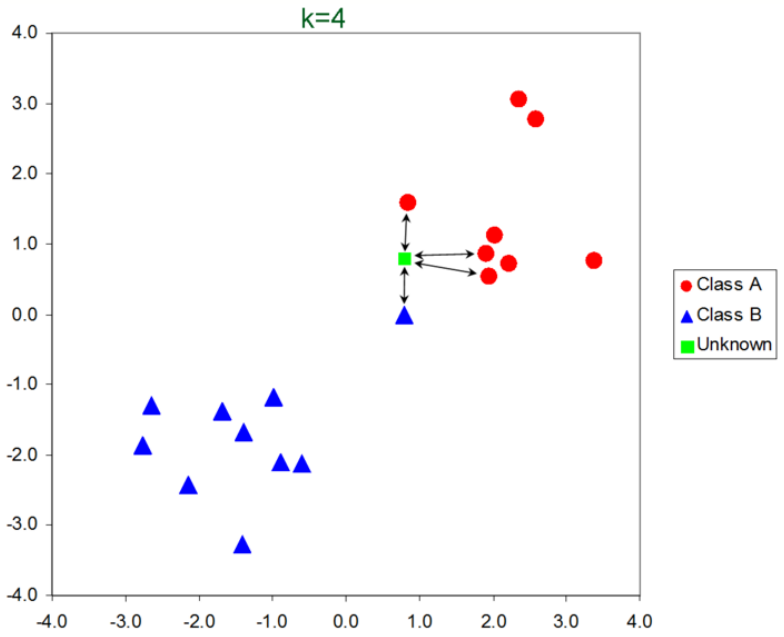


(Source: Reference [15])

Central tasks of CBR are to deal with identifying the current problem situation, finding past cases similar to the new one (Retrieve), using those similar cases to suggest a solution to the current problem (Reuse), evaluating the proposed solution (Revise), and updating the system by learning from this experience (Retain). Literature [15] covers whole topics regarding CBR nicely, and the above picture from [15] can help us to understand how CBR works in problem solving. CBR can be widely used for the area such as past case's utilization in helpdesk function and case-law technical diagnosis and so on. Also, in the financial market, searching for "past similar market circumstances with today" to find some insights for the future is a natural way of thinking for many market practitioners. The purpose of CBR application toward algorithmic trading is to mimic such market practitioners' methodology.

4. k-Nearest Neighbors (kNN)

To implement the case retrieval in CBR system, k-nearest neighbour (kNN) algorithm and its advanced application with the further improvement and/or the combination with other algorithms such as genetic algorithm (GA) are widely used in many literatures (See [16-21]). To illustrate how kNN works in the context of CBR, I implemented simple kNN this time in the system.



(Source:http://scholarpedia.org/article/K-nearest_neighbor)

K Nearest Neighbours (kNN) is the algorithm to take the distance among certain feature vectors and extract top kth nearest vectors in terms of distances. For example, in the above picture, k = 4, and 3 data from the red circle and 1 data from blue are taken for the new coming data of green. With regard to taking distance, the Minkowski distance below is popular. Especially, when we take q=2, it becomes Euclidean distance and this is popularly used in the kNN algorithm.

$$d(X,Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q},$$

Where:

X, Y: two different feature vectors i.e. dataset, X=(x1, x2,...xn), Y=(y1, y2,...yn)

n: # of features in X and Y

q: if q=1, d(X, Y) is called the Manhattan distance. If q=2, d(X, Y) is called the Euclidean distance.

(Source: Reference [22])

You can take note that the kNN performs better with a lower number of features. If the number of features increases, it requires more data to perform kNN appropriately. Increase in dimension of the feature vector can also lead to the problem of overfitting. We call such problems, which is due to the higher dimension, the “Curse of dimensionality”. To avoid the curse of dimensionality, we often use PCA to reduce the dimensionality of data, as shown in the Macro Economist agent section.

5. Implementation of Case Retrieval in CBR by kNN

We can apply kNN to implement the CBR of cryptocurrency trading in the same manner. In the above mentioned graph, green unknown data corresponds to today's newly coming market data. Then, the system can search k nearest past market data that correspond to red circles and blue triangulars. By doing so, the system can implement the "Retrieve" section of CBR.

In our system, data shown below picture is stored in the past case data storage (i.e. dataframe of **cbrDecider.case_data_df**), and we can extract the top 30 (i.e. k=30) of the most similar datasets compared with the certain day, by using kNN.

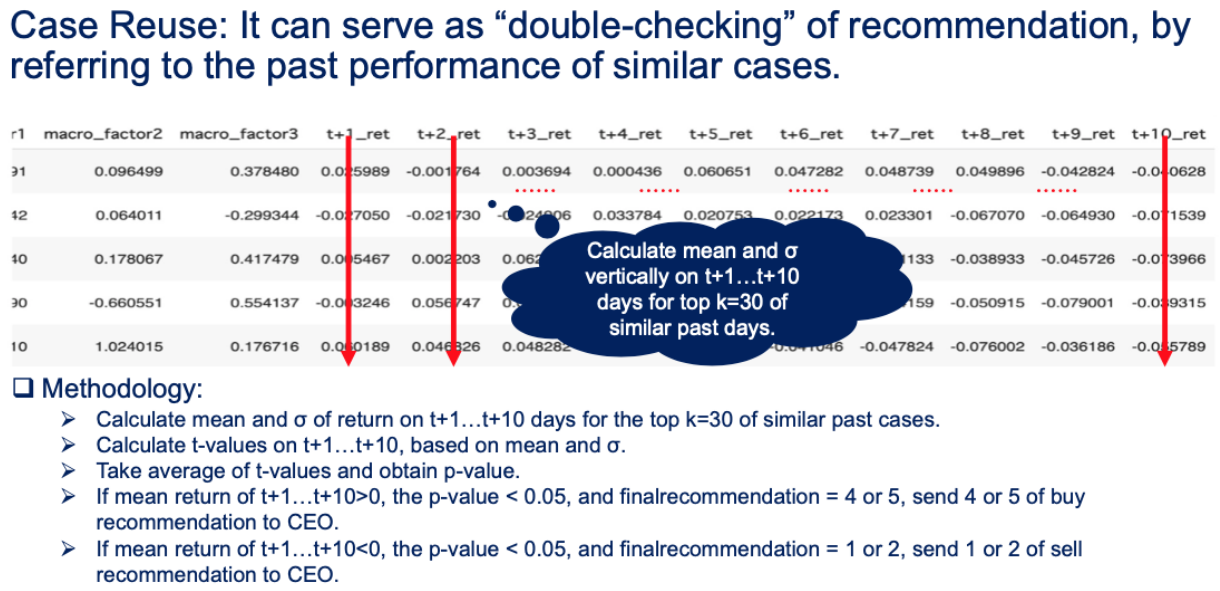
```
cbrDecider.case_data_df
```

	agent1_long	agent1_short	agent2_long	agent2_short	agent3_long	agent3_short	macro_factor1	macro_factor2	macro_factor3
2019-01-01	1.0	0.0	0.0	0.0	1.0	0.0	-0.891191	0.096499	0.378480
2019-01-02	1.0	0.0	0.0	0.0	0.0	0.0	-0.858942	0.064011	-0.299344
2019-01-03	1.0	0.0	0.0	0.0	0.0	0.0	-0.523940	0.178067	0.417479
2019-01-04	1.0	0.0	0.0	0.0	0.0	0.0	-0.141390	-0.660551	0.554137
2019-01-05	1.0	0.0	0.0	0.0	0.0	0.0	-0.179610	1.024015	0.176716

(Source:NUS Fintech Lab, Naoya Ohara)

As a result, the CBR system can obtain the data of "the most similar past market condition" in terms of the buy/sell recommendation from 3 signaling agents and the macroeconomic environment summarized by 3 macroeconomic factors. To avoid the curse of dimensionality mentioned above, macroeconomic data is summarized into 3 factors, using PCA in the previous section. Usually, the performance of kNN can improve by the dimensionality reduction using PCA (or other methodologies of dimensionality reduction of data), such that you can utilize the combination of PCA => kNN as one of your tools in your arsenal when you implement data analytics including but not limited to the financial market data analysis.

6. Logic of Case Reuse, Revise, and Retain part in CBR



(Source:NUS Fintech Lab, Naoya Ohara)

Next, I will show how the system implements the "Reuse" part of CBR. After taking top k=30 of the "market days of the most similar past market condition", the system calculates mean and standard deviation (i.e. volatility) for each day. Then, the system calculates t-values on t+1...t+10 days based on each mean and standard deviations. Next, the system takes the average of t-values and obtains p-value.

If the mean return of t+1...t+10 is positive, p-value is less than 0.05 (i.e. statistically significantly different from 0% return at positive side), and finalrecommendation obtained at previous sections is 4 or 5 (i.e. buy or strong buy), we send 4 or 5 rating to the CEO agent. On the other hands, if the mean return of t+1...t+10 is negative, p-value is less than 0.05 (i.e. statistically significantly different from 0% return at minus performance side), and finalrecommendation obtained at previous section is 1 or 2 (i.e. sell or strong sell recommendation), we send 1 or 2 rating to the CEO agent. If p-value > 0.05, it means that the past return of t+1...t+10 is not statistically significant, such that the Decider just send 3 (i.e. neutral) recommendation to the CEO agent, even though finalrecommendation obtained at previous section shows buy or sell recommendation. Whole above process is implemented at the function of **def dailyReportToCEO(day, k=30, shreshold_p = 0.05)** in the **class cbrDecider(object).**

Above process means that the CBR of Reuse part can serve as the "double checking" function to confirm whether the recommendation can fit to the past actual performance in terms of the past similar cases.

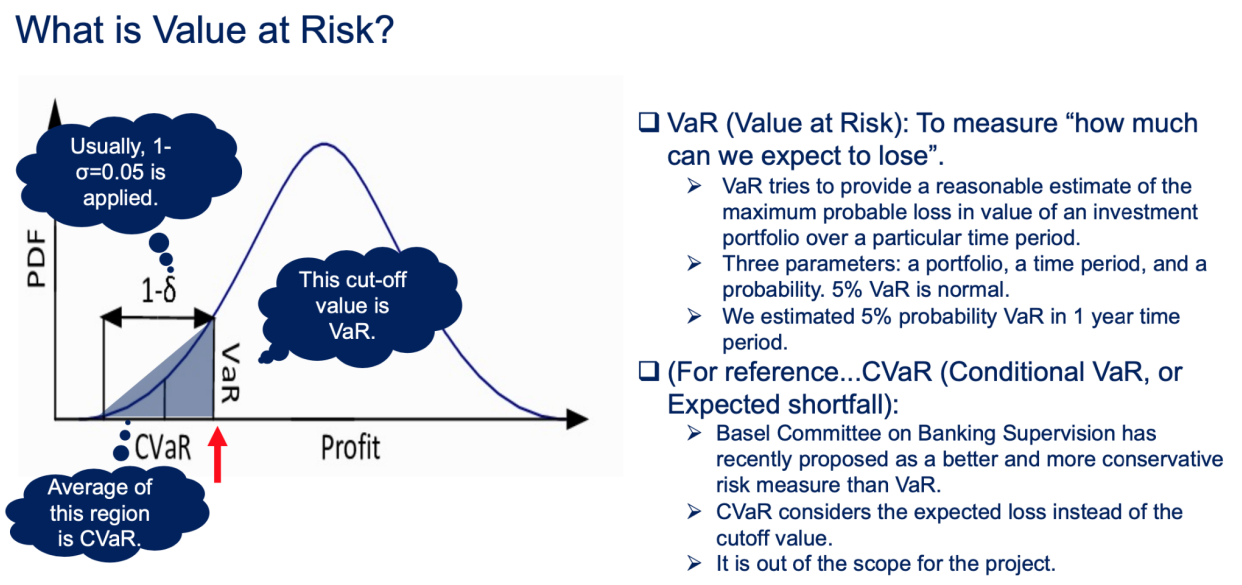
After the Decider sends this double-checked recommendation to the CEO agent, the Decider agent updates the past case as of today to execute the "Revise" and "Retain" part in CBR. By doing so, new cases are updated daily such that the Decider can utilize recent cases for the next day. The function of **def caseUpdate(day)** in the **class cbrDecider(object)** is the implementation of those parts.

Risk management agent(In the code: class riskManagement(object))

I implemented an independent risk management agent, based on Value at Risk (VaR). Basically, institutional investors (i.e. such as hedge-fund) determine the position size based on "how much we can make loss". The legendary hedge-fund manager George Soros also mentioned this point as "Survival first, then make profit". Also, in private wealth management (i.e. such as private banking), this approach can be used. This chapter introduces the concept of VaR. The CEO agent will set the maximum limit of annualized 5% VaR, i.e. possible annual loss with 5% of probability as initial risk management policy. Then, the position size is determined based on this annualized VaR limit.

1.Value at Risk (VaR)

In the risk management agent, I introduced an important concept of risk management called Value at Risk (VaR). VaR tries to provide a reasonable estimate of the maximum probable loss in value of an investment portfolio over a particular time period. There are 3 parameters in VaR, a portfolio, a time period, and a probability. Basically, practitioners calculate 5% VaR in a certain period (1 day, 1 month, or 1 year depending on the situation) is normal. Also, if you prefer more conservative risk management criteria, you can also apply CVaR, i.e. Conditional VaR, which considers the expected loss with 5% of probability, rather than cut-off value in VaR.



(Source: Advanced Analytics with Spark, NUS Fintech Lab, Naoya Ohara)

There are 3 methods to implement VaR (or CVaR) as follows:

- **Variance-Covariance method:** Variance-covariance is by far the simplest and least computationally intensive method. Its model assumes that the return of each instrument is normally distributed, which allows deriving an estimate analytically just by calculating volatility of the portfolio.
- **Historical simulation method:** Historical simulation extrapolates risk from historical data by using its distribution directly instead of relying on summary statistics. For example, to determine a 95% VaR for a portfolio, we might look at that portfolio's performance for the last 100 days and estimate the statistic as its value on the fifth-worst day as "1 day 5% VaR". A drawback of this method is that historical data can be limited and fails to include all market scenarios such as market collapse.
- **Monte Carlo Simulation method:** Monte Carlo simulation tries to weaken the assumptions in the previous 2 methods by simulating the portfolio under random conditions. While this method is widely used in the financial risk management at banks and asset management companies, it requires much computational resources. We can utilize distributed computing such as Spark to calculate Monte Carlo VaR [23], while it is out of the scope for this project.

In the risk management agent, the agent calculates estimated 5% probability VaR annually i.e. in 1 year time period by variance-covariance method, because of its simple implementation and computationally cheap and fast execution. When we apply the variance-covariance method, we need to calculate the portfolio volatility i.e. standard deviation of σ (Note: variance is the square of standard deviation). The portfolio volatility for 2 securities is shown as follows:

$$\sigma_{portfolio} = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2\rho_{1,2}\sigma_1\sigma_2}$$

Where:

w_1	=	Proportion of the portfolio invested in Asset 1
w_2	=	Proportion of the portfolio invested in Asset 2
σ_1	=	Asset 1 standard deviation of returns
σ_2	=	Asset 2 standard deviation of returns
$\rho_{1,2}$	=	Correlation coefficient between the returns of Asset 1 and Asset 2

(Source:https://medium.com/python-data/assessing-the-riskiness-of-a-portfolio-with-python-6444c727c474)

In our case, w1=cash, w2=bitcoin. Also, cash has zero volatility i.e. $\sigma_1 = 0$. Therefore, above equation can bi simplified as $\sigma\_portfolio = \sqrt{w_2^2\sigma_2^2} = w_2\sigma_2$. At the functions of **def coinVaRCalc(day, var_pct=0.05)** and **def portVaRCalc(day, var_pct=0.05)** in **class riskManagement(object),** the portfolio volatility is calculated as such.

Note: If the portfolio can consist of multiple securities, portfolio volatility is calculated by matrix algebra shown as follows:

The formula for portfolio volatility is:

$$\sigma_{Portfolio} = \sqrt{w_T \cdot \Sigma \cdot w}$$

- $\sigma_{Portfolio}$: Portfolio volatility
- Σ: Covariance matrix of returns
- w: Portfolio weights (w_T is transposed portfolio weights)
- · The dot-multiplication operator

(Source:https://stackoverflow.com/questions/59462628/is-there-a-way-to-vectorize-the-portfolio-standard-deviation-in-python-pandas)

For example, above portfolio volatility of 2 securities can be shown in context of matrix algebra, as follows:

$$\sigma_p^2 = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} \\ \sigma_{2,1} & \sigma_2^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1\sigma_1^2 + w_2\sigma_{2,1} & w_1\sigma_{1,2} + w_2\sigma_2^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$
$$= w_1^2\sigma_1^2 + w_1w_2\sigma_{2,1} + w_1w_2\sigma_{1,2} + w_2^2\sigma_2^2$$
$$= w_1^2\sigma_1^2 + 2w_1w_2\sigma_{1,2} + w_2^2\sigma_2^2$$

(Source:https://medium.com/python-data/assessing-the-riskiness-of-a-portfolio-with-python-6444c727c474)

While dealing with multiple securities or multi asset classes is out of scope this time, we can implement VaR with multiple securities or multi asset classes using the above matrix algebra formula relatively easily with Python. Also, while CVaR and monte carlo simulation methods are out of the scope in this project, it is worth introducing as those are often used by industry practitioners, such that I also introduced those just for reference.

2.Position sizing using VaR

After we can calculate the portfolio of VaR, we can decide the position size using VaR. For example, the CEO agent sets the maximum VaR limit = 0.2 (20% of loss in 1 year with 5% of probability), we can take the position as such, by calculating σ_portfolio = weight of bitcoin in total NAV * σ_bitcoin, i.e. the weight of bitcoin in total NAV = σ_portfolio / σ_bitcoin. The system implements such a function at **def targetPositionFromVaR(day, VaRLimit, var_pct=0.05)** in in **class riskManagement(object).** In this function, using above logic, the target bitcoin's position size is calculated by Target Coin Value = NAV * VaR_limit / VaR_bitcoin.

As you can infer, the bitcoin volatility σ_bitcoin can change over time as the degree of market fluctuation of bitcoin changes, such that our portfolio volatility σ_portfolio can change over time. The CEO function can decide the position sizing using this function, and can rebalance the portfolio using this function, i.e. if the VaR becomes too large at the current position size in terms of VaR limit, the system can slush the position size, and vice versa. It is the operation that many institutional investors actually do in daily portfolio management, such that I implemented such a function to introduce such basic portfolio management activities in the financial and investment management industries.

PowerBI / RPA agent (In the code: dataVisualRPA(object))

Data visualization and RPA (Robotic Process Automation) are also important parts in business operational flow.

By data visualization with several graphs and charts, we can recognize what's going on in the market and our portfolio easily and can make decisions toward the market timely (such as the human decision making of whether we should stop working the trading system etc).

Also, we can improve efficiency of daily operation by automating repetitive operations such as sending monthly reports by email and storing attached files in email to specific data storage (such as One Drive).

This agent takes those tasks such as data visualization and RPA. I introduced PowerBI for data visualization and Power Automate for RPA. Those Microsoft products are widely used, such that we can utilize those in our workplace to improve operational efficiency.

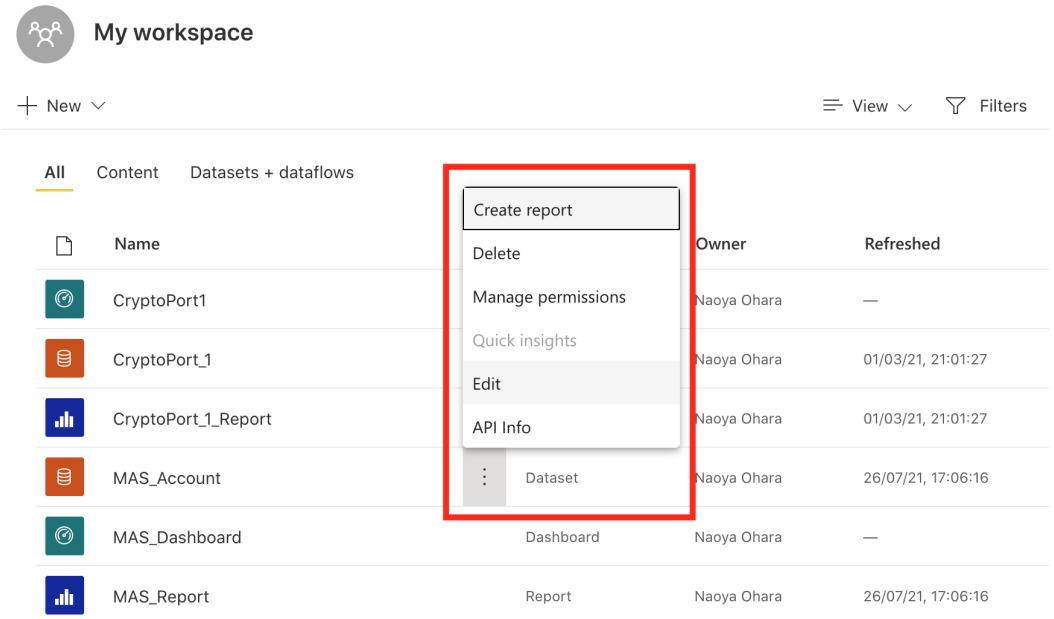
1.Data visualization by PowerBI

In the data visualization tool, Tableau and PowerBI are popularly used in business corporations, and the PowerBI is the one which is released by Microsoft. With regard to the basic usage and how-to, we can refer to Microsoft's website as follows. You can learn how to create and manage Report and Account in Power BI to visualize our data with graphs and charts.

Get started using Power BI
https://docs.microsoft.com/en-us/users/microsoftpowerplatform-5978/collections/k8xidwwnzk1em

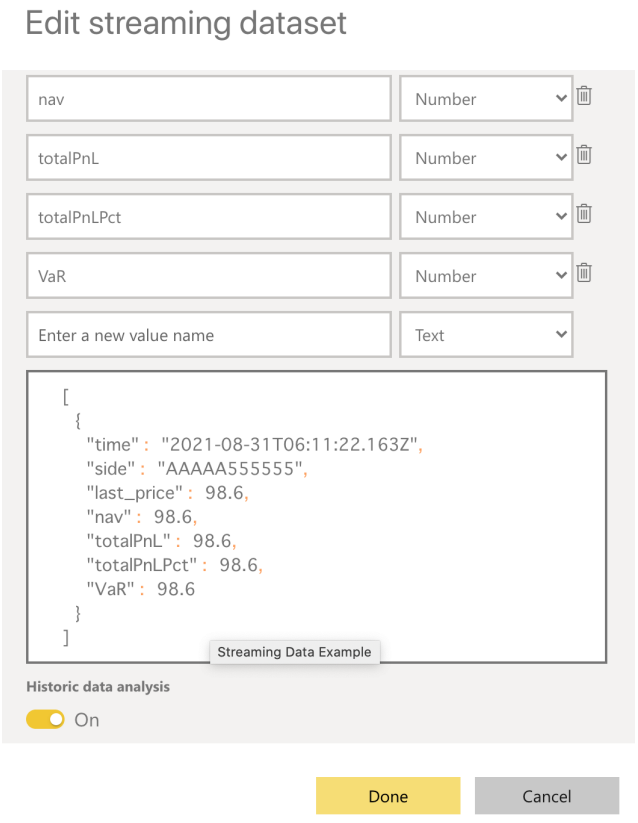
By utilizing API (Application Programming Interface) of Power BI, we can send our data, which are generated by Python's dataframe in google Colab notebook, into Power BI. I will explain the steps for using Power BI API briefly.

- **Step 1:** Create Account in "My workspace", then click "more options (vertical "...")", then click "Edit" as is in the picture below.



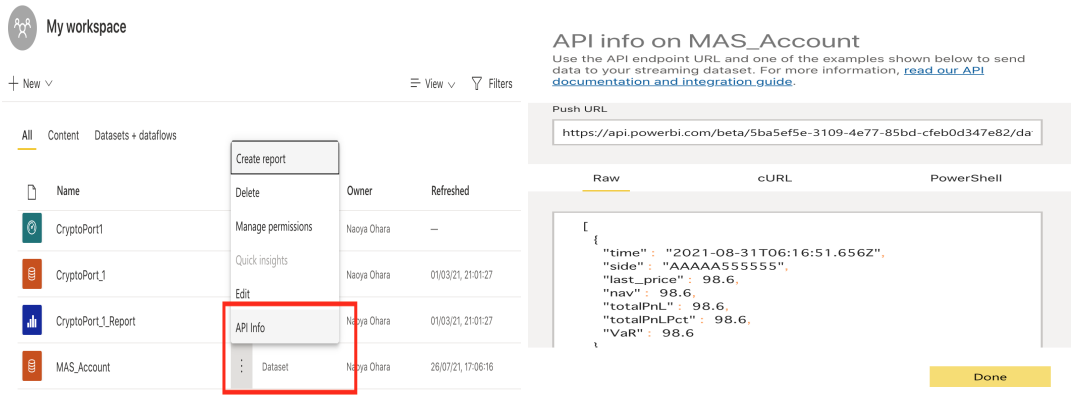
(Source: Microsoft, NUS Fintech Lab, Naoya Ohara)

- **Step 2:** Edit data properties shown as follows. Historic data analysis is "On" and click "Done", such that Power BI stores historical daily trading data.



(Source: Microsoft, NUS Fintech Lab, Naoya Ohara)

- **Step 3:** Check "API Info" as we can see in the pictures below.



(Source: Microsoft, NUS Fintech Lab, Naoya Ohara)

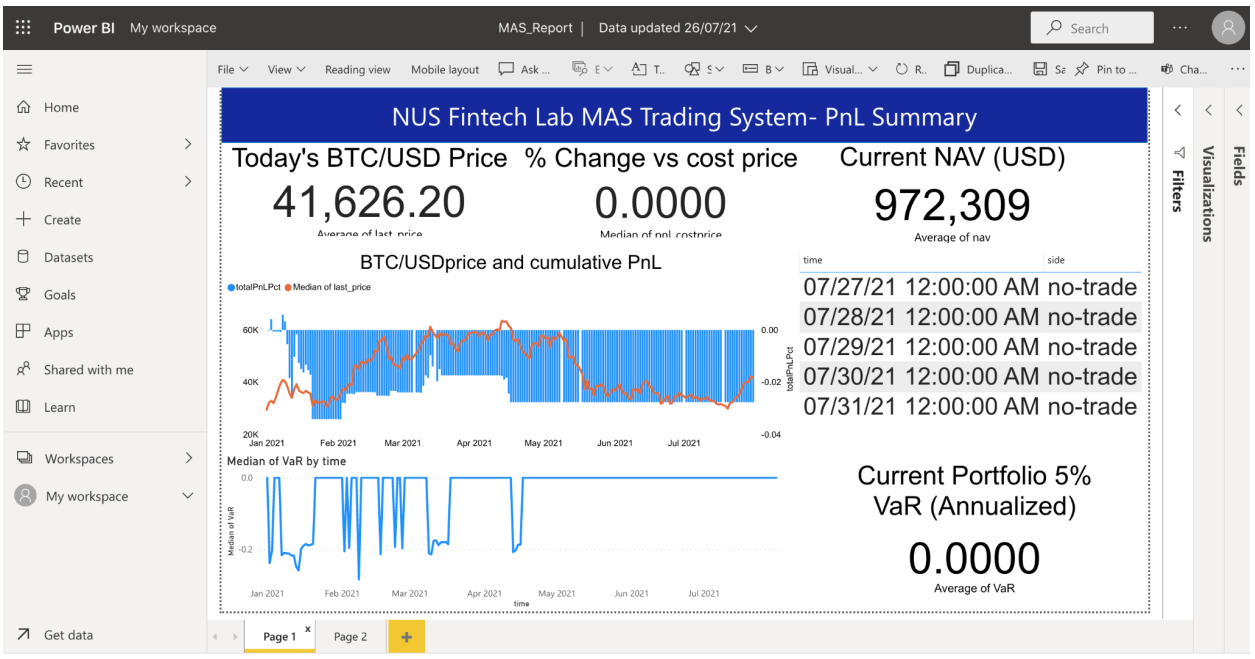
- **Step 4:** Copy and paste url in the code shown as follows. Set sending data and API configurations like follows.

```
class dataVisualRPA(object):

  # Visualization in PowerBI
  @staticmethod
  def visualizationPowerBI():
    #use your own PowerBI url
    url ='https://api.powerbi.com/beta/5ba5ef5e-3109-4e77-85bd-cfeb0d347e82/datasets/fec8de5

    # Prepare all the data
    day =Account.getTradeInfoDf()['time'].iloc[-1].strftime('%Y-%m-%d %H:%M:%S')
    side =Account.getTradeInfoDf()['side'].iloc[-1]
    exec_price =Account.getTradeInfoDf()['exec_price'].iloc[-1].astype('float64')
    last_price =Account.getTradeInfoDf()['last_price'].iloc[-1].astype('float64')
    quantity =Account.getTradeInfoDf()['quantity'].iloc[-1].astype('float64')
    num_coins = Account.getTradeInfoDf()['num_coins'].iloc[-1].astype('float64')
    coin_value = Account.getTradeInfoDf()['coin_value'].iloc[-1].astype('float64')
    cash = Account.getTradeInfoDf()['cash'].iloc[-1].astype('float64')
    nav = Account.getTradeInfoDf()['nav'].iloc[-1].astype('float64')
    dailyPnL = Account.getTradeInfoDf()['dailyPnL'].iloc[-1].astype('float64')
    dailyPnLPct = Account.getTradeInfoDf()['dailyPnLPct'].iloc[-1].astype('float64')
    totalPnL = Account.getTradeInfoDf()['totalPnL'].iloc[-1].astype('float64')
    totalPnLPct = Account.getTradeInfoDf()['totalPnLPct'].iloc[-1].astype('float64')
    VaR_annualized = Account.getTradeInfoDf()['VaR_annualized'].iloc[-1].astype('float64')
```

```
data = [
        {"time": day,
         "side": side,
         "last_price": last_price,
         "nav": nav,
         "totalPnL": totalPnL,
         "totalPnLPct": totalPnLPct,
         "VaR": VaR_annualized
         }
        ]

headers = {"Content-Type": "application/json"}
response = requests.request(
  method="POST",
  url= url,
  headers=headers,
  data=json.dumps(data)
)
```

(Source: Microsoft, NUS Fintech Lab, Naoya Ohara)

Then, we can see the data updated like the following picture at the "Report" which is linked with the "Account" file.



(Source: Microsoft, NUS Fintech Lab, Naoya Ohara)

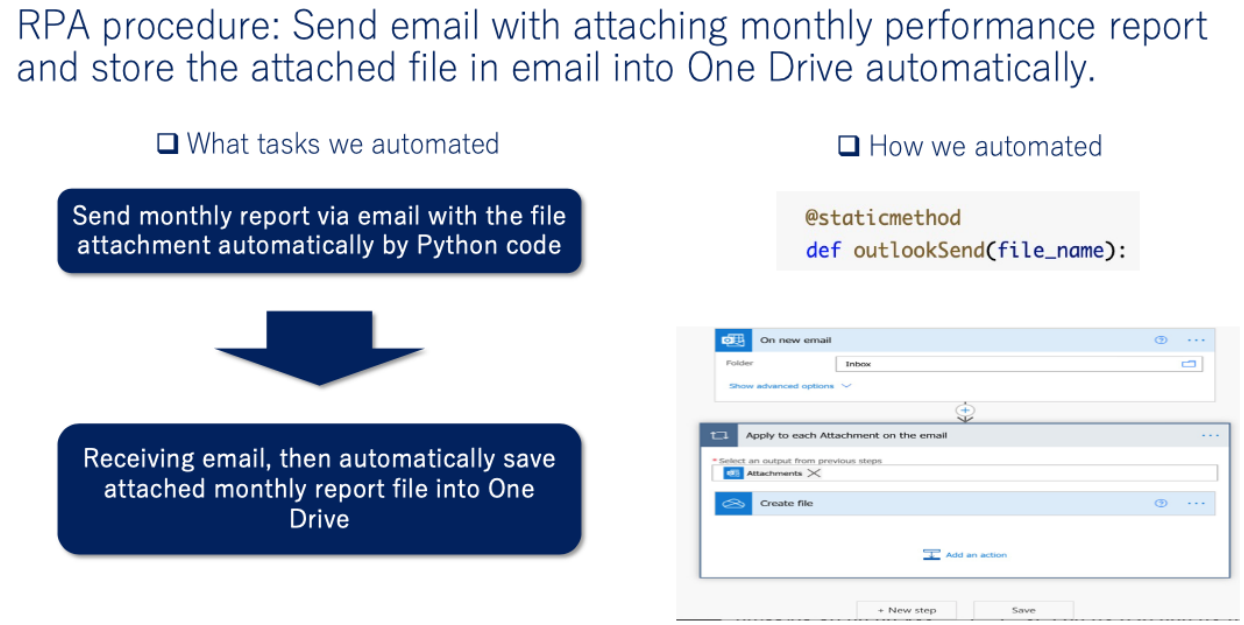
Data visualization by Power BI or Tableau is often used in the daily workplace. And we can see many data scientists' job descriptions requiring Power BI or Tableau, such that it's valuable for us to learn those tools as industry practitioners.

2.RPA by PowerAutomate

Robotic process automation (RPA) is a technology that mimics and automates the way humans interact with software to perform high-volume, repeatable tasks. While there are many RPA software and services available, Microsoft Power Automate is a convenient solution for RPA, especially as we would like to automate operations around softwares and services of Microsoft family (i.e. Office 365, One Drive etc). With regard to the basics of how-to and what we can do by Power Automate, we can refer to the following tutorials from Microsoft.

Get started with Power Automate
https://docs.microsoft.com/en-us/power-automate/getting-started

On the other hand, unfortunately, Power Automate cannot set Google Drive and google Colab as the trigger of doing automated action. Also, with regard to the frequently used function such as sending email automatically, we can write code to send email automatically. So, I introduced the code to send email automatically at **def outlookSend(file_name)** in **class dataVisualRPA(object)**. As a result, the system automated the monthly report sending and storing tasks by Python code and Power Automate shown in the following picture. We can reduce the cost and improve daily efficiency significantly by RPA, such that we can utilize this skill at the daily workplace.



(Source: Microsoft, NUS Fintech Lab, Naoya Ohara)

CEO agent (In the code: class CEO(object))

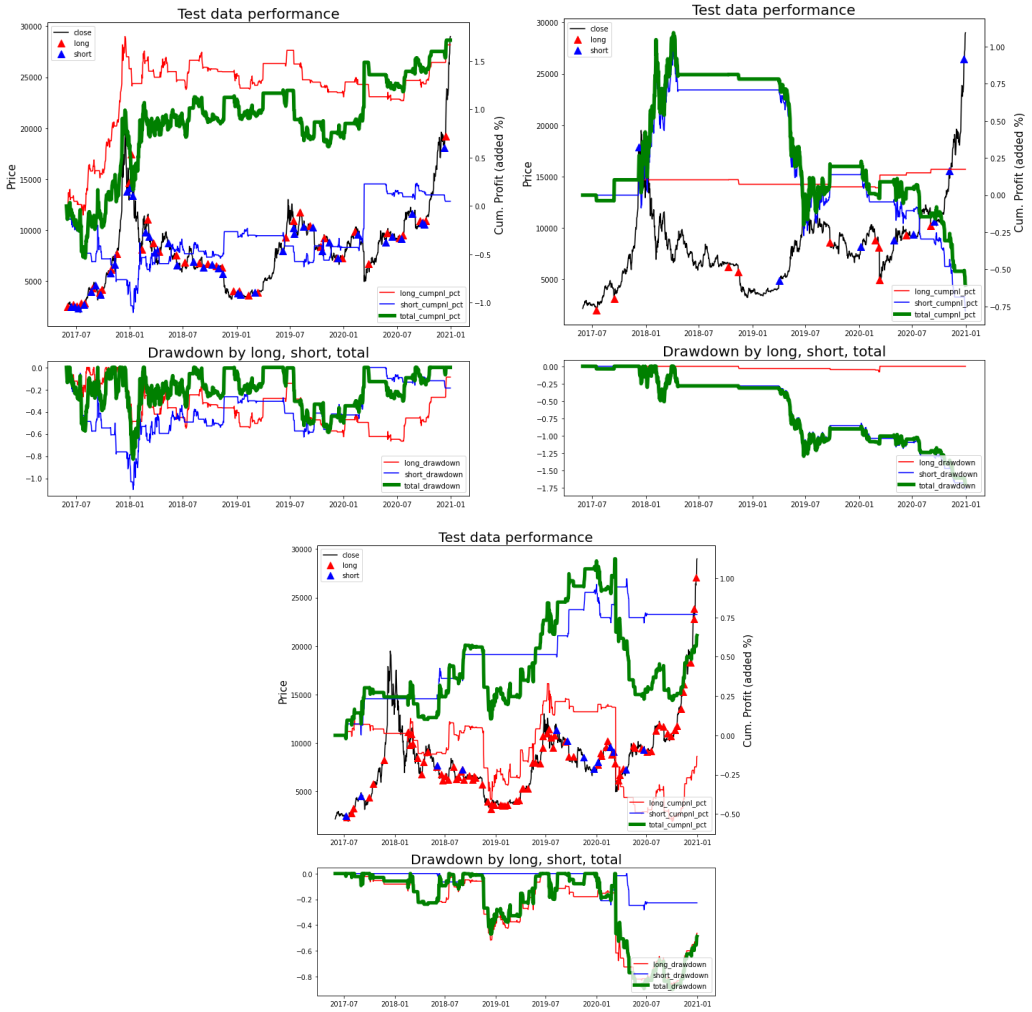
Finally, we reached the final agent, the CEO agent. First, the CEO agent establishes investment and risk management policy at ceoInitialSetUp() function. CEO sets up initial fund amount, starting day, risk management policy such as annualized VaR limit, and profit target/maximum loss tolerance when the fund terminates trading if it reaches to that point. Such investment and risk policy are set before we start daily trading and fund management activities in the real world of fund management too. Then, the CEO agent orchestrates the buy/sell recommendation from the Decider agent, Account recording by Account

agent, risk management from Risk management agent, and makes final trading decisions and asks trade execution to the Broker agent.

Simulated trading performance and its insight

1.Backtest performance of 3 each strategies (SMA, Bollinger Band, and Twitter sentiment)

Below charts show the performance of SMA signal, Bollinger Band signal, and Twitter sentiment signal, calibrated by train / test split with the sliding window. Below data is the connected performance data of 4 test data periods, based on the 4 training data and its parameter optimization. The parameter optimizations were executed in training periods, and those parameters were applied in the test period. Below charts consist only of test data, such that we could avoid the overfitting issue by train/test split and sliding window approach.



The test data performance of SMA (upper left), Bollinger Band (upper right), and Twitter sentiment (bottom)

(Source: NUS Fintech Lab, Naoya Ohara)

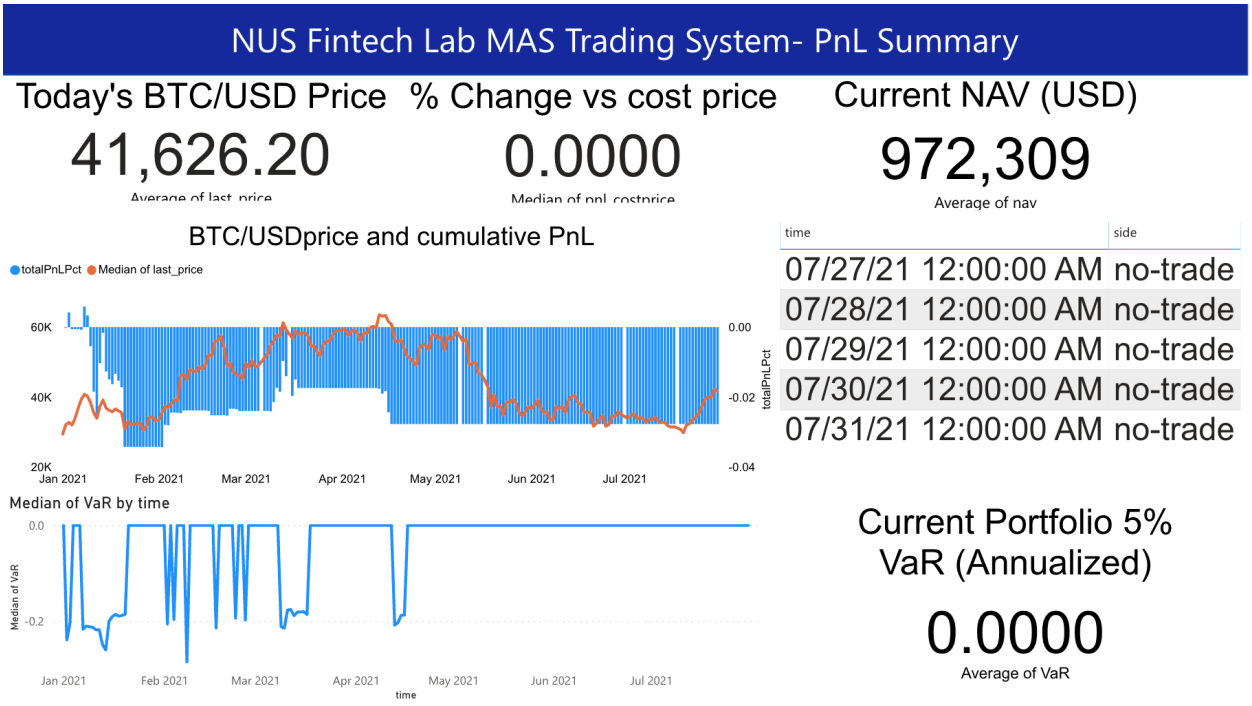
- **SMA works well:** SMA strategy is very traditional and simple technical analysis. However, as we can see from the upper left chart of SMA performance, SMA could make relatively nice performance, while the drawdown in the market collapse in 2018 was relatively tough. This result implies that there exists a long-lasting price trend in the bitcoin market, and the trend-following strategy can work relatively well. Simple strategy can work sometimes relatively well, and

complexity does not necessarily mean better performance [24]. Successors can try to find the better algorithms by improving trend-following strategies.

- **Bollinger Bands did not work well:** Unfortunately, the Bollinger Bands did not work well at all in the past test data. In particular, the recommendation of short-selling suffered a lot, i.e. the price went up further after the system recommended to sell. The Bollinger Bands is the popular strategy as the counter-trend strategy, i.e. mean-reverting strategy after the price went up/down rapidly. The failure of this strategy can show the same implication as the SMA strategy: The bitcoin price tends to have a long-lasting price trend. Probably, searching for counter-trend/mean-reverting strategies for bitcoin may not be a wise approach for quantitative researchers.
- **Twitter sentiment moderately works:** The twitter sentiment showed moderately plus performance, mainly contributed by short-selling recommendation, while the performance is less than that of SMA strategy. This result suggests there can be rooms for improvement in text sentiment analysis. For example, while we picked up twitter data mainly from popular news media such as Bloomberg, NYT, CNN, FT etc, those news media can be "lagged" indicators. When the news media talks about bitcoin in a bullish manner, it is possible that most of the price surge is ending, and vice versa. Also, we can improve the methodology of obtaining text sentiment scores from texts, by developing our own NLP algorithms too. However, in total, text sentiment strategy can be worth investigating for improvement than the mean-reverting algorithms such as Bollinger Bands, as this result suggests.

2.Performance of simulated trading in 2021 (Jan-July), by integrating 3 signals using CBR and VaR risk management

Next, the below shows the performance of simulated trading in 2021 from Jan to July, by integrating 3 signals using CBR implemented in the Decider agent, and by enhancing risk management with VaR in the Risk Management agent, those which were orchestrated by the CEO agent.



(Source: NUS Fintech Lab, Naoya Ohara)

As a result, while at the time of the bitcoin price plunge the system could safeguard the performance with CBR function and VaR risk management system, still the performance was -2.8% YTD

(Year-To-Date), slightly negative performance which reduced 1 mio USD of NAV to 972,309 USD. To improve the performance, first, we need to find more profitable buy/sell signals as in particular the Bollinger Bands was not profitable shown above. Then, we can think about how we can improve the Decider function and risk management as well.

Future improvement

I believe that the system and this final paper can offer the body-of-knowledge in algorithmic trading and real world's MAS, CBR, NLP, and Machine learning applications to it, in addition to domain knowledge in finance such as risk management by VaR and back-testing using train/test split with sliding window. On the other hand, due to the time constraints and other factors, there also exist rooms for future improvement. I will list some of them.

1.Time horizon of data: From day to minutes, second or less than that

This time, I implemented the whole system with day OHLCV data for several reasons, such as follows:

- Data availability. Day OHLCV data can be available since around 2014 easily by yahoo finance. On the other hand, 1 minutes or 5 minutes data can be obtained by market exchanges such as binance and hitbit. However, those market exchanges do not offer several years of 1-5 minutes tick data for back-testing for free.
- Nice data to understand long-term historical movement in bitcoin. By checking daily price data of bitcoin more than 5 years, we can understand the basic history and characteristics of bitcoin price movement appropriately, i.e in year 2018 there was the price boom and bust in bitcoin, the annualized volatility can be more than 100% (price double or half in one year is normal for bitcoin), etc.
- Appropriate complexity in system development as the showcase at NUS Fintech lab. If we deal with 1 or 5 mins data, the system is required to deal with more complex matters such as the data synchronization issue and the whole system can become exponentially complex. I've heard from the supervisor at NUS Fintech Lab that learners at NUS Fintech Lab can include non computing background of people, such that after the discussion with the supervisor, we concluded that daily data can be appropriate this time.

However, in the real world of trading, short-term trading within some minutes using minute tick data is not so special. In equity and currency markets with thick liquidity, second or even less than mili-second of trading is also popular, while we need to implement Java, C++ or FPGA to implement such a very fast trading and such implementation is out of the scope for this project.

2.Searching and picking up more profitable technical indicators at quantitative agents

I picked the SMA and Bollinger band, because SMA is the basic technical indicator for trend-following and the bollinger band is also very popular in using the basic mean-reverting indicator. However, we can search for more profitable technical indicators in the future. Also, we can pursue more advanced statistical analysis to generate better buy/sell recommendations.

3.The rooms for improvement in qualitative agent, NLP of twitter text

I believe that I can introduce important topics in NLP with qualitative agents, the topics such as data preprocessing of stop-words and data transformation into sentiment indicators using Fuzzy Logic. On the other hand, there are additional topics for the improvement in twitter text analysis shown as follows.

- This time, we did not use twitter information such as the number of likes and retweets, due to the constraints of data availability within free and easy data acquisition. The number of likes and retweets shows the importance and popularity from many people, such that such information can have significant value for better analysis.
- We did not include analysis of reply and retweet of tweets. In the twitter context, each tweet does not stand alone. In reality, to understand the context of conversation and its tone, we should

analyze whole communication flows of reply and retweet. While it could increase the complexity of text sentiment analysis exponentially such that I didn't implement such analysis this time, analyzing such a whole context of conversation in twitter can become an important topic for future improvement.

- We can increase or change data sources too. This time, I just obtained twitter from popular news' twitter accounts such as Bloomberg, FT, NewYork Times and Elon Musk. While limiting data source into such major accounts assures relatively low noise and high quality in data, especially official news accounts may just mention the past and may not predict about the future. By including less popular but future-insightful accounts to data source, we may be able to improve the predictability of twitter sentiment analysis.
- We can utilize advanced machine learning techniques or originally developed NLP tools. This time, we mainly utilized google sentiment analysis tool for natural language processing. However, if we'd like to differentiate the quality of twitter sentiment by the improvement of the NLP algorithm, we may need to develop it by ourselves, utilizing advanced machine learning techniques such as BERT, by diving deeper into NLP itself (or by hiring the professionals of NLP).

4. Utilizing more advanced concepts in finance domain

This time, to make the system tractable for the students including the people without finance background, I implemented some finance concepts in a simple way.

For example, I omitted the calculation of market impact estimate such that the transaction cost is just fixed as 0.1% per one buy/sell trade.

Also, I just implemented VaR by the simplest method i.e. mean-variance approach. However, we can also develop it in more advanced ways such as monte-carlo method. Also, VaR has more advanced relatives such as CVaR (Conditional VaR), which can be more conservative and a better way for risk management. In the future, the sophistication of VaR implementation can become one of the important topics to improve the performance of a system.

In addition, this time, I just implemented a "cash or bitcoin" portfolio, to simplify the portfolio management process. However, we can introduce multi currency portfolio or multi asset class portfolio including equity and fixed income in the future, the implementation which can become more realistic in the actual institutional portfolio management.

By improving the above points, the system can become closer to the institutional, professional investor level.

5. Utilizing advanced machine learning

This time, I thought much of offering the body-of-knowledge in algorithmic trading and the application of basic machine learning into real world's financial data analysis, such that I implemented PCA and kNN with scikit learn, which is the basic but powerful tool to implement simple machine learning algorithms.

However, with regard to CBR, some literature offer more advanced methodologies to improve the quality of CBR [16-21]. Also, many market practitioners are now trying to apply more advanced machine learning (such as deep learning and reinforcement learning) into the algorithmic trading arena [25,26]. How we can improve the trading performance by applying advanced machine learning with TensorFlow or PyTorch can become the next topic for future improvement.

With regard to the application of advanced machine learning into financial trading and portfolio management, I recommend referring and studying [12, 27] written by Marcos Lopez De Prado, who is the quant practitioner who worked at global quant hedge-funds such as AQR capital management and Guggenheim Partners. [12, 27] contain advanced topics, but you can follow many of the topics covered by those books, after learning topics covered in this paper.

6. Connection with simulated exchange or real money trading

The main purpose of creating the system this time is to show the body-of-knowledge in algorithmic trading and real-world application of machine learning, such that I implemented the system as the simulated environment without the real trading in market exchange. However, in the future, if the NUS Fintech lab could launch the simulated market exchange, we can connect the system with its simulated

market exchange. Also, after the trial at simulated market exchange and feasibility study for real money trading, we may proceed to the real money trading to make profit in the future, while it requires more stringent standards regarding the system robustness, several risk management, and other safeguards not to lose money by several risks including system failure risk.

References

[1] https://www.investopedia.com/terms/b/bollingerbands.asp
[2]D.V.Cruz, V.F.Cortez, A.L.Chau, R.S.Almazan. Does Twitter Affect Stock Market Decisions_Financial Sentiment Analysis During Pandemics_A Comparative Study of the H1N1 and the COVID-19 Periods, 2021
[3] C.Kearney, S. Liu. Textual Sentiment Analysis in Finance_A Survey of Methods and Models, 2013
[4] Zhang.W, Skiena.S. Trading strategies to exploit blog and news sentiment, 2010
[5] https://textblob.readthedocs.io/en/dev/api_reference.html#textblob.blob.TextBlob.sentiment
[6] https://cloud.google.com/natural-language/docs/basics
[7] Adrian A. Hopgood. Intelligent Systems for Engineers and Scientists Third Edition, CRC Press, 2012
[8] Lam.M. Neural Network Techniques For Financial Performance Prediction Integrating Fundamental And Technical Analysis, 2004
[9] Rui Pedro Barbosa, Orlando Belo. Algorithmic Trading Using Intelligent Agents, 2008
[10] https://www.investopedia.com/articles/fundamental-analysis/10/strategy-performance-reports.asp
[11] Barclays Global Investors, Fumio Nakakubo et al., Everything about quantitative active investment: its theory and practice (Written by Japanese), Kinyuu Zaisei Jijo Kenkyukai, 2008
[12] Marcos Lopez De Prado. Advances in Financial Machine Learning. John Wiley & Sons, 2018
[13] Wang.X, SykoraM.D, Archer.R, Parish.D, BezH.E.Case based reasoning approach for transaction outcomes prediction on currency markets, 2009
[14] John C Hull. Options, Futures, and Other Derivatives (10th Ed). Pearson Education, 2018
[15] Aamodt Agnar, Enric Plaza.Case-Based Reasoning, Foundational Issues Methodological Variations and System Approaches, 1994
[16] Huseyin.I. Short term stock selection with case-based reasoning technique, 2014
[17] LiS.T, HoH.F. Predicting financial activity with evolutionary fuzzy case-based reasoning, 2009
[18]Campillo-Gimenez.B, Jouini.W, Bayat.S, Cuggia. M. K-Nearest Neighbour algorithm coupled with logistic regression in medical case-based reasoning systems. Application to prediction of access to the renal transplant waiting list in Brittany, 2013
[19] Chun.S.H, Park.Y.J. A New Hybrid Data Mining Technique Using A Regression Case Based Reasoning Application To Financial Forecasting, 2006
[20] Qi.J, Peng.Y, Hu.J. A New Adaptation Method Based On Adaptability Under K-nearest neighbors For Case Adaptation In Case-based Design, 2012
[21] Wang.F, CheungD.W. Combining Technical Trading Rules Using Particle Swarm Optimization, 2014
[22] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques, 2009.
[23] Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills. Advanced Analytics with Spark. O'Reilly, 2015
[24] Cliff.D, Rollins.M. Methods Matter A Trading Agent with No Intelligence Routinely Outperforms AI-Based Traders, 2020
[25] Zhang.Z, Wang.D. EAQR A Multiagent Q-Learning Algorithm For Coordination Of Multiple Agents, 2018
[26] Korczak.J, Hemes.M. Deep learning for financial time series forecasting in A-Trader system, 2017
[27] Marcos Lopez De Prado. Machine Learning for Asset Managers. Cambridge University Press, 2020

Disclaimer

This document is provided for information purposes only to eligible recipients. **This document shall not constitute an offer to sell or the solicitation of any offer to buy any interest**. **The author is not responsible for any loss or damage arising from any investment or any other activities based on any information contained here.**