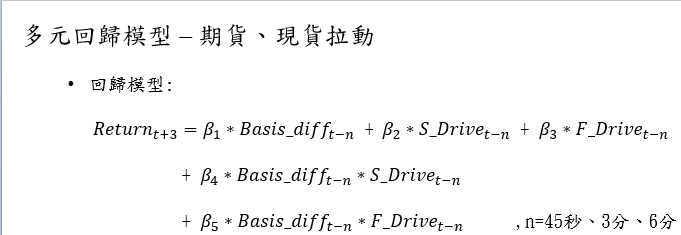
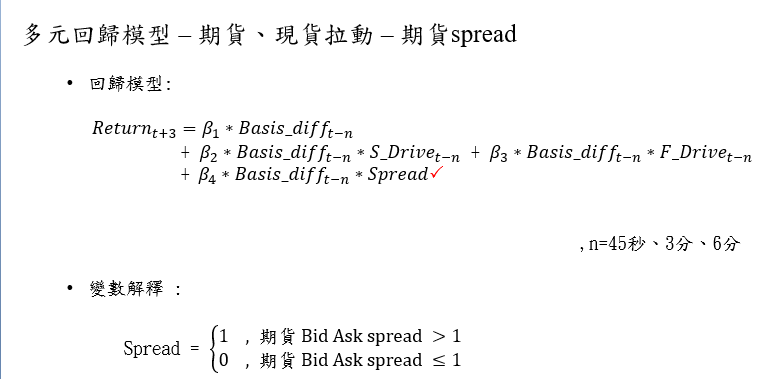
2/24



簡單線性回歸 -> 一個一個挑進來 (為何不先挑一個，接著2.3.4…….)

共線性問題? Why use linear regression?

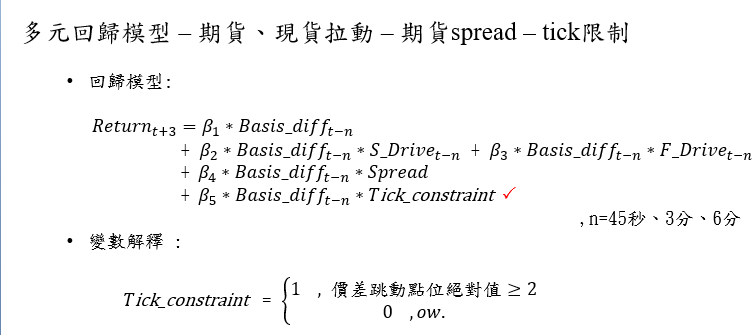
現貨拉動? 期貨拉動? S\_Drive、F\_Drive



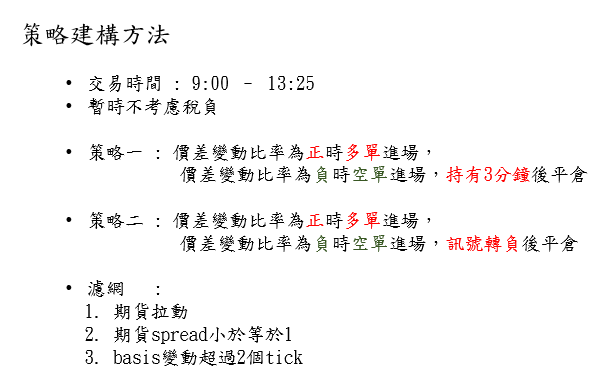
期貨spread? 為何要納入這個指標

Q -> 訊號出現後，我們假設可以在下一筆成交資料進場，如果**進場當下期貨spread變寬**，則會**現賠變寬的點數?**

P16 -> basis只變動一個tick時，報酬分配較為分散，**容易頻繁進場又出場**；而tick = 2時，75%以上報酬為正，tick = -2時，75%報酬為負 -> 設定一個跳動區間為 2~7



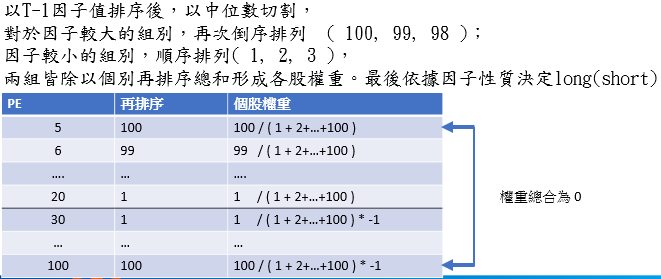
整理好重要因子後(**期貨拉動、期貨spread <= 1、basis變動 > 2 tick**) -> 開始回測



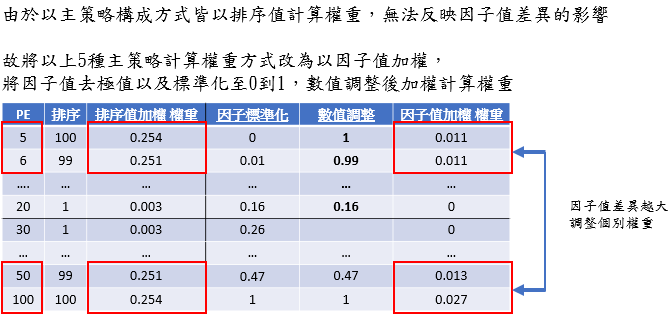
專案主題3：日間因子投組研究

主策略1 -> 5種

主策略1-3 排序權重法 -> 意義? (重新排序權重，更標準?)

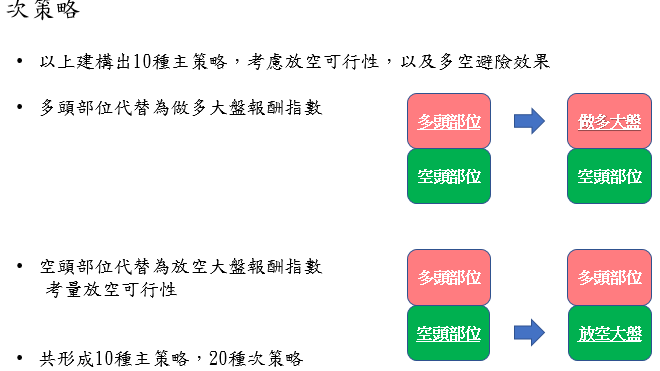


主策略2 因子值加權 (策略1的5種)



Total: 10種主策略

次策略



Total: 20種次策略(多頭 -> 做多大盤；空頭 -> 放空大盤)

2/25

門密碼

12324565

Data

基本資料：**成交量、收盤價、總市值、成交金額**

Code錯誤改正 -> df\_V\_melt -> df\_V\_melt\_new (line47)

產業數據：**產業代號、台灣發行量加權股價報酬指數(大盤指數)**

<https://whhnote.blogspot.com/2011/01/python-strtime-strftime.html>

(時間格式轉換方法)

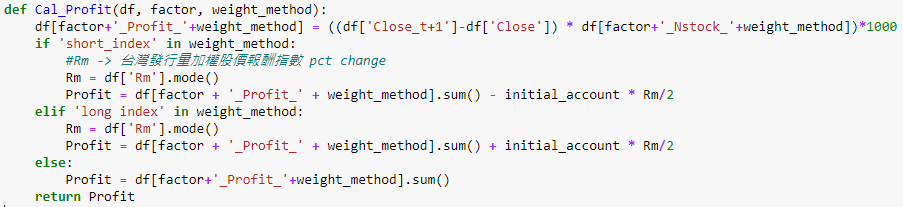
[datetime.strftime(x, format='%Y%m%d') for x in df\_RM['Date'] ]

df\_merge1\_new更正

CMoney data：**PE、PB、外資買賣超(for)、券資比(rgz)、外資買賣超金額(for\_N)、投信買賣超(invest\_N)、主力買賣超張數、股本(sharecapital)**

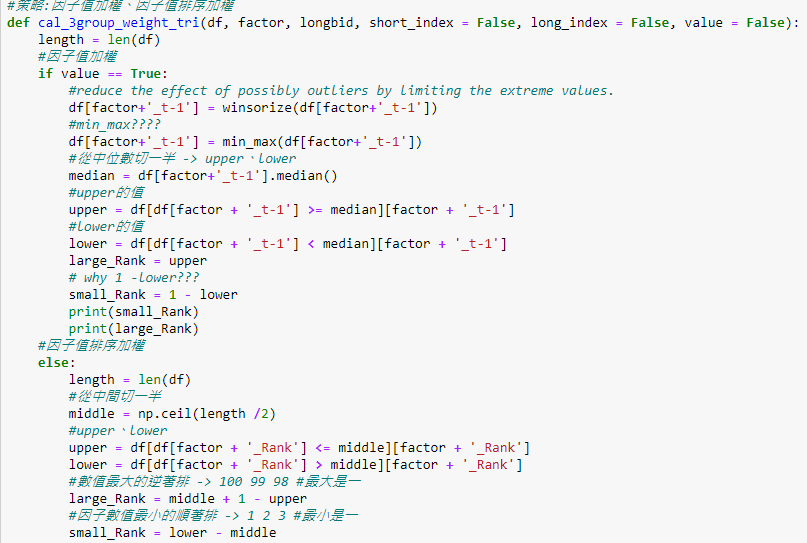
其他資料：**VWAP(成交量加權平均價)、SALES、div(股利)**

Df\_merge3\_new 更正



Df[‘RM’.mode ??、why initial account/2，分一半? -> 一半short/long大盤，一半long/short台股

因為是取每日資料，基本上RM只有一個值 -> 取眾數沒差 (solved)



**因子值加權**、**因子值排序加權**

2/26

策略想法

篩選：平均日成交金額 >= 5000萬 、以T-1日市值排序 -> 選取前200大個股(加上**挑選前10大產業**)

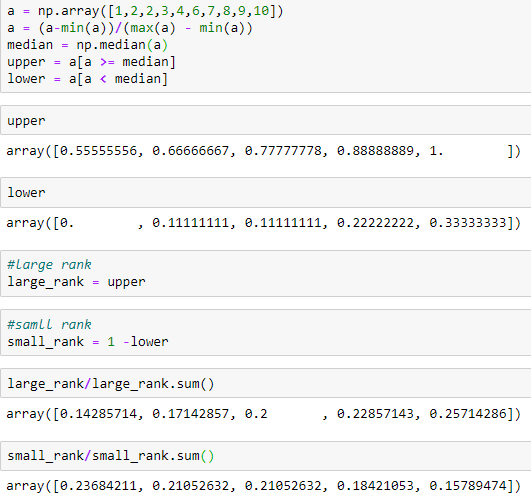
排序權重法(**考慮產業**) + 因子值加權 ex: PE ratio

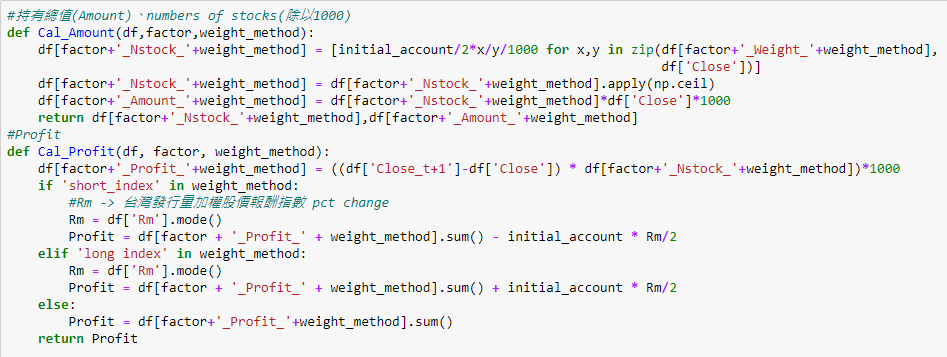
排序PE -> 因子去極值、標準化 -> 從中位數切一半

-> **Upper 、 lower** -> large rank = upper、lower rank = 1- lower

-> 依照策略計算weight

買(賣)因子大的，賣(買)因子小的 (一買一賣 -> **delta對消的概念**)

策略1 因子值加權法



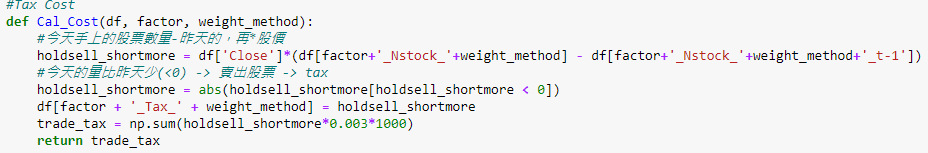
每天10億 -> 5億long/5億short -> initial amount / 2

問題：**沒有考慮到account剩下多少錢的問題**?

* 可能是只hold一天，每天進出場?
* 不過**隔天又會有initial account的一半的錢**可以花，有點奇怪

計算equity value的方法

最後initial value + **逐天累加的porfits** – 逐天累加的tax



依據前面的function(Cal\_Amount)來看，感覺每天的買賣是獨立的，因此以此為標準來判斷是否有賣出股票有點奇怪 (已解決)

Ex:昨天某張股票買300張，今天的算法得出要買200張 -> 賣出100張?

* 今天和昨天的價差去計算損益(profits)，昨天20今天21，if long -> 賺1\*N\_stocks
* 最後逐天累加，再加上初始金額 -> 最後的總equity

進度:line372 strategy 2

3/2

篩選台股標準

1. 以平均日成交金額5000萬為篩選標準
2. 總市值前200大

* temp\_m

1. 選出前10大產業產業(數量前10多的)

* temp\_i

1. 平均權重法 考慮產業 -> Ind\_CAP\_Weight

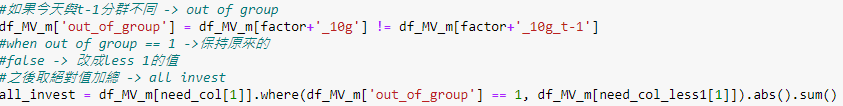
策略2

**問題**

1. 不懂取out of group的意思

Out of group -> 昨天與今天的分群不同 (ex:PE變化大時)

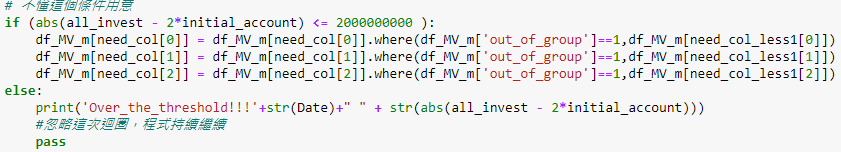
* 如果PE變化過大時，仍然買昨天的amount



1. |all invest – 2 \* initial value| <= 2 \* initial value (20億)

差異過大時(t、t-1期分在不同群) -> 將t期改成t-1期的stocks、amounts、weight

Where(out of group == 1 -> 改成 less1)



策略1 因子值排序加權 (tri)

1. 挑選總市值前200大
2. 將t-1期的PE排序，給予排名 1~200
3. 給予權重

策略2 因子值排序加權 + 10組固定 (tri\_10g)

策略3 因子值排序加權 + 產業百分比(scale) (tri\_ind%)

1. 挑選總市值前200大
2. 將t-1期的PE排序
3. 將其去極值並標準化(scaling) -> PE\_t-1
4. 又因太多0、1 -> 依照PE\_t-1\_%排序後，如有相同，再依照原本的PE\_t-1去排序
5. 給予排名 1~ 200
6. 給予權重

策略4 因子值排序加權 + 產業百分比 + 10組固定 (tri\_ind%\_10g)

策略5 因子值排序加權 + 產業標準化 (tri\_indstd)

同策略3，唯獨標準化的方式為std

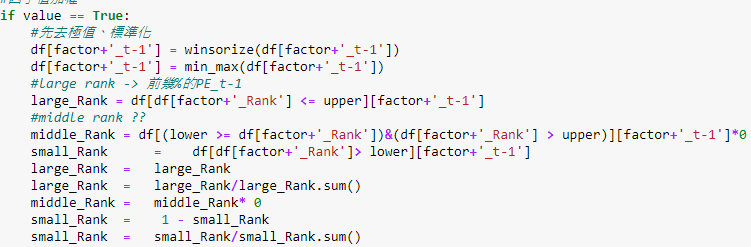
策略6因子值排序加權 + 產業標準化 + 10組固定 (tri\_indstd\_10g)

**back\_test\_multiway\_PE**

問題

1. middle\_Rank ??

* 這裡不配給權重，中間不作買賣 ex: PE買後10%賣前10%



3/3

計算權重(因子值加權->以因子值去作加權、因子值排序加權->以因子值的rank去作加權)

1. 排序權重法

2. 平均權重法

large rank、small rank ex. 取前後10%，並給予equal weight

main strategy

#因子值排序加權

1. 排序權重法(tri)

初始的function

1. 排序權重法 考慮前10大產業(tri\_ind)

依照各個產業類別分別去平均分配權重、排序rank

上述權重最後再除上本身產業的個數 -> 真正的權重 (總權重和=1)

1. 排序權重法 考慮前10大產業 考慮產業市值

依照各個產業類別分別去平均分配權重、排序rank

上述權重最後再\*先前算出的Ind\_CAP\_Weight(前10產業市值之權重)

1. 平均權重法(eq)
2. 平均權重法 考慮前10大產業(eq\_ind)

ex. long/short因子值最大的2檔個股，並short/long因子值最小的2檔個股

，並給予equal weight

上述權重最後再除上本身產業的個數 -> 真正的權重 (總權重和=1)

1. 以上結合放空指數or做多指數
2. 將上述全改成#因子值加權(value)

**Select factor by machine learning**

因子: 過去一年的 -> PE(取倒數)、PB(取倒數)、ROE、市值(取對數 ~ Normal)、3 months beta

Response: 下一個月的報酬

regression coefficient (factor return) is calculated by using the monthly factor value of each stock of the **past 1 year (before the t-phase) as the explanatory variable** and by using **the end of the next month (t-phase) return as the target variable**. We have calculated the future stock return rate (expected return) of each stock by **multiplying the resultant regression coefficient (factor return) with the factor value (factor weight) of each stock** at the end of the test period.

we considered **first quantile** of the five quantiles portfolio made on the basis of the resultant expected return as **long portfolio** and **fifth quantile** as **short portfolio**, and calculated the difference in return (spread return) of the long portfolio and the short portfolio when held

until the next month. We rolled this on a monthly basis and accumulated the obtained returns.

問題

Out of group那幾個策略，用t、t-1的資訊去買賣t期的股票有點奇怪

* 改 t-1、t-2 (自己看錯，solved)

作業

1. 預測一檔股票的中價 (bid+ask)/2 using machine learning algorithms

<https://www.groundai.com/project/machine-learning-for-forecasting-mid-price-movement-using-limit-order-book-data/1>

3/4

Machine learning for forecasting mid price movement using limit order (限價單) book data

* a limit order is a type of order to buy or sell a specific number of shares within a set price.

Data: financial time-series data

* inherently noise and non-stationary (不適合用統計模型 -> 因為常需要分配的假設等等), machine learning algorithms usually take no such distribution assumption.
* NN、SVM、deep learning methods (capable of modeling highly non-linear and complex data)

\*limit order data changes distribution frequently even within one day.

\*feature extractions (ex. NN、bag-of-features)

Three classifiers are evaluated using combinations of these sets of features on two different evaluation setups and three prediction scenarios.

Three scenarios are assessed regarding the span of time for which predictions are made: a scenario where (1) **the movement of the mid price of the immediately succeeding sample in time is predicted,** one where (2) **the average mid price movement of the next five samples is predicted** and, last, one where (3) **the average movement of the mid price for the next ten samples is predicted**.

\*Past work

Ex. Using NN SVM MLP -> using two diff windows, long and short term

and it is shown that **using too many samples** that span too far into the past can degrade the prediction quality.

In a similar fashion to this work, kercheval2015modelling uses several handcrafted **time sensitive and insensitive features**, extracted from the limit order book. These features include **bid-ask spreads** and **mid prices**, **price differences**, **mean prices** and **volumes**, along with derivatives of the price and volume, average and relative intensity indicators, totaling to 144 different features. However, in kercheval2015modelling the proposed methods are evaluated on a very small dataset that contains about 400,000 order book rows. In contrast, in our case a large-scale dataset that contains information for **10 days and 5 stocks** is used, with the raw data being more than **4 million samples**.

\*Features from high frequency limit order book

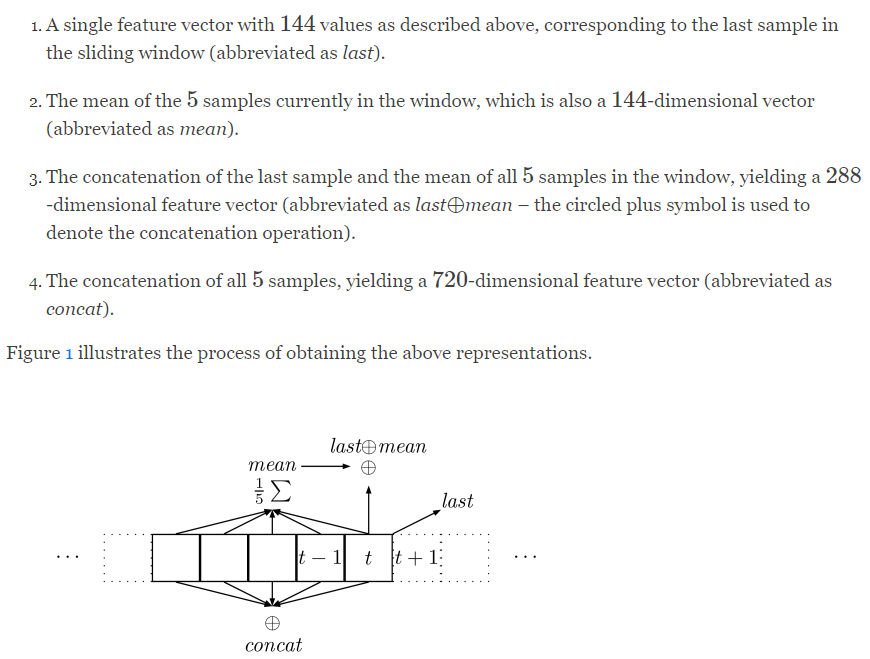
Bia、ask、bid volumes、ask\_volumes

The data used in this work consists of **10 orders for each side (bid, ask) of the LOB**. Each order is described by **2 values, the price and the volume**, yielding a **total of 40 values for each time step**. The time period used for collecting that data ranges from the 1st to the 14th June 2010 (only business days are included), and the data is provided by the Nasdaq Nordic data feeds siikanen2016limit (); siikanen2017drives (). The dataset is made up of

10 days for 5 different stocks and the total number of messages is about 4.5 million with equally many separate depths.

More specifically, first, a basic set of features which includes the **prices and volumes** for every level of t**he ask and bid side of the order book** is extracted. This information yields **40 values** at each time step. Then, time-insensitive features describing **the spread, mid-price, price and accumulated price differences between the bid and ask orders** of each depth level, and price and volume spreads are extracted. Finally, time-sensitive features are extracted corresponding to the **average intensity for trades, orders, cancellations, deletion, execution of visible limit orders and execution of hidden limit orders**. This set of features also includes **the price and volume average values at each level of the LOB, the average intensity per trading type as well as comparisons between the intensities and limit activity acceleration** (derivatives of average intensities). Because of the non-linear nature of time in the LOB data, we follow an event-based inflow.

Feature extracted methods

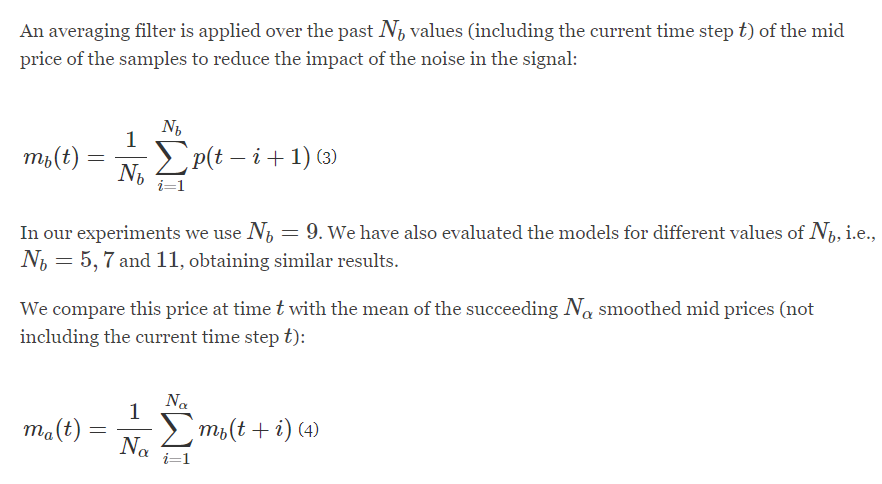


Using this sliding window approach allows for **temporal information to be incorporated** into the representations, which are subsequently used to predict the movement of the stock’s mid price. By averaging over the five entries contained in the window, some of the inherent noise of the features is alleviated. Concatenating (聯集) different representations, e.g., the last feature vector and the average of the last 5 feature vectors, allows for more temporal information to be introduced in the final feature vector.

Predition labels

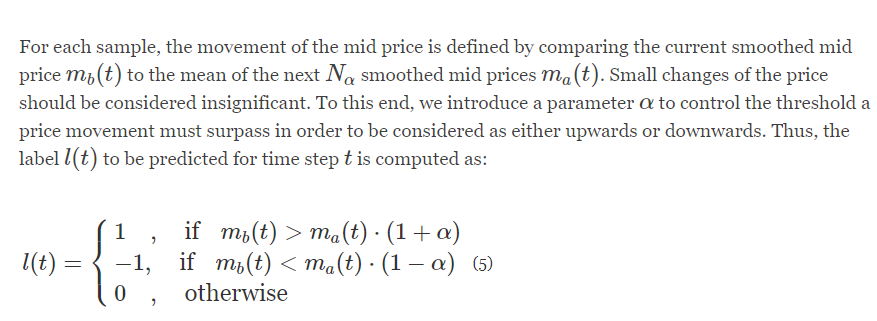
Of the values accompanying LOB data, **the tick price,** which is the price of the last executed trade, typically **varies wildly between the two sides of the margins**, introducing great amounts of noise to the prediction labels. The so-called market micro-structure **noise can be partially reduced by using mid-prices**, i.e. the mean of the best ask and best bid prices, **instead of transaction prices**.

Another advantage of using mid-prices instead of transaction prices is that **they are observable every time as long as there are orders on both bid and ask sides** while transaction prices are updated only at transactions.



N\_alpha = 1, 5, 10

Classification: up, down, no change



Alpha = 0.0001, 0.0002, 0.0003

Ex. mb(9) > ma(9) -> 上漲

lob\_ML\_3

Feature Engineering

**handcrafted features** and **fully automated feature extraction** methods

Since the problem under consideration in the present thesis focuses on financial

data (i.e., the mid-price prediction based on high-frequency data), relevant features/-

factors are related to financial concepts. More specifically, these features will be **LOB**

**features,** **technical indicators, quantitative analysis, econometrics, and fully automated**

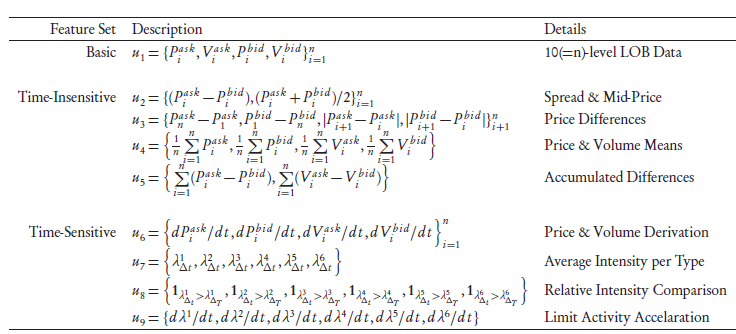
**financial processes.** The following list of features is by no means exhaustive,

demonstrating only a few exemplary cases.

LOB features

extraction is based on the prediction of mid-price. There are three main

categories of feature set whose description can be seen on below



Technical indicator

ADX, PSAR, LRL

Quantitative Analysis

Ex. Autocorrelation and Partial Correlation、Cointegration

Econometric

Ex. Vol, realized variance, realized kernel, realized pre-averaged variance

Fully Automated Feature Extraction

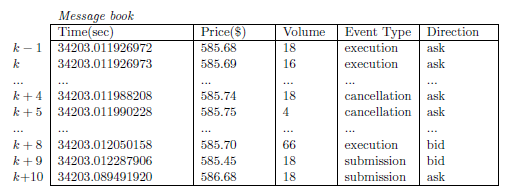
Ex. Autoencoders (AE)

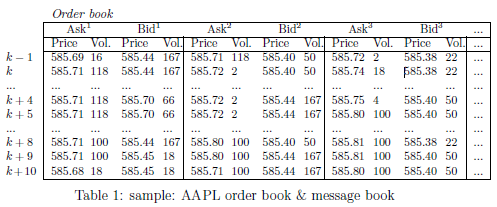
問題

Bid、ask price 有沒有1~10個levels的資料

Prediction多為mid price(t+1)的漲/跌/平 (classification)

Lob\_SVM





Event type: **limit order submission**, **limit order cancellation**, or **market order**

**execution**

It can also be observed from the message book that **multiple trading events** could arrive **within milliseconds** as demonstrated from Row k - 1 to k + 10, leading to **drastic fluctuation of prices**

**and volumesin** the order book. Although a variety of “metrics’’ have been designed to capture the price fluctuation, in this paper we select as metrics (a) the occurrence and direction of mid-price movement, and (b) the occurrence and direction of bid-ask spread crossing, described below.

These two metrics can work independently or together to provide guidance for

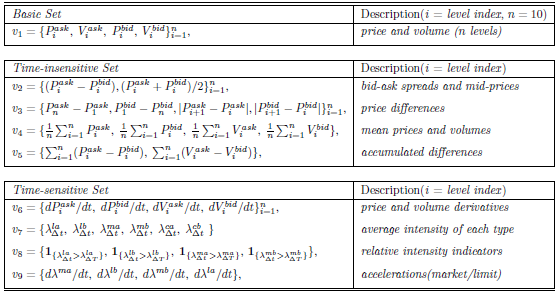
trading strategies depending on particular scenarios. Of course, the premise for the

arbitrage opportunities described above is that the directions of mid-price movement

and price spread crossing can be predicted accurately, which is the main task addressed

in this paper.

Feature Extraction and Training Data Preparation



7 days train 3 days test

3/9

10 levels bid ask price

**Basic set**

1. each bid ask price and their volume (OK)

**time insensitive**

1. bid-ask spread and mid price from 10 levels (OK)
2. bid, ask price diff (1-10、10 -1) and their abs value (OK)
3. bid, ask, their volume’s mean (OK)
4. accumulate diff (OK)

**time sensitive**

1. price and volume’s derivatives (per 5 sec) (OK)
2. average intensity: limit/ market/ cancellation ask/bid orders (per 5 sec) average arrival rate
3. relative intensity indicators: if 10sec’s rate > 900sec’s rate -> 1 else 0 (limit/ market)
4. v7’s derivative (per 5 sec) (limit/ market)

**需要的data**: 前10筆買賣價、event type(submission, execution, cancellation)、個股期貨資料

問題: rolling window(固定長度or累積?) roll幾筆? Test set 預測一筆就好? Response的表達方式

Tensor-based regression

Convolutional Neural Network (CNN)-based regression

Recurrent Neural Network (RNN)-based regression

結果: random forest ACC大概70%

3/10

Forecasting vix futures

LSTM RNN: 時間給予權重，愈遠的權重愈小 (long-short-term-memory RNN)

Contango: 遠期交割價 > 即期交割價

Backwardation: 遠期交割價 < 即期交割價

Input variables

1. term structure spreads: 不同到期日之vix future’s spread
2. vol surface skewness: The skew is the difference in implied volatility between the two strikes at a particular maturity.
3. Vix spot index & MA signals

(vix spot index have 14, 50, 100days)

If index cross upper MA -> +1, cross lower ma -> -1, others -> 0

1. VVIX(Vix of Vix): take its close price and its high-low intraday spread

3/11

防火牆擋住(http error 0000)解決方法

conda config --set ssl\_verify no

環境

<https://towardsdatascience.com/installing-keras-tensorflow-using-anaconda-for-machine-learning-44ab28ff39cb>

conda update --all

cudatoolkit ver: 10.1.243

解決cuda版本問題

<https://blog.csdn.net/carolynlmk/article/details/86630865>

LSTM

<https://medium.com/@daniel820710/%E5%88%A9%E7%94%A8keras%E5%BB%BA%E6%A7%8Blstm%E6%A8%A1%E5%9E%8B-%E4%BB%A5stock-prediction-%E7%82%BA%E4%BE%8B-1-67456e0a0b>

看test

<https://www.kaggle.com/amarpreetsingh/stock-prediction-lstm-using-keras>

best one!

<https://github.com/Danjtchen/LSTM_stock_example/blob/master/LSTM_example.ipynb>

LSTM variable selection

<https://stats.stackexchange.com/questions/191855/variable-importance-in-rnn-or-lstm>

3/16

LSTM 整理

<https://zhuanlan.zhihu.com/p/31783805>

高頻交易 LSTM

<https://github.com/blockchain99/stock-predict-by-RNN-LSTM>

lstm early stopping

<https://blog.csdn.net/silent56_th/article/details/72845912>

deal with overfitting: early stop, drop out, regularization, batch normalization

drop out

<https://machinelearningmastery.com/how-to-reduce-overfitting-with-dropout-regularization-in-keras/>

response改成預測**報酬**

3/18

卡爾曼慮波 配對交易

Lstm normalized data will help the machine converge and get the features equal weighted

It will be beneficial to normalize your training data. Having different features with widely different scales fed to your model will cause the network to weight the features not equally. This can cause a falsely prioritisation of some features over the others in the representation.

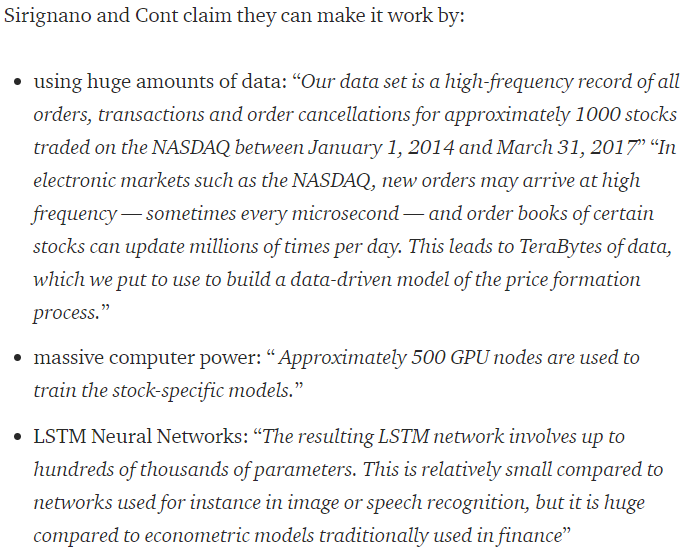
Despite that the whole discussion on data preprocessing is controversial either on when exactly it is necessary and how to correctly normalize the data for each given model and application domain there is a general consensus in Machine Learning that running a Mean subtraction as well as a general Normalization preprocessing step is helpful.

In the case of Mean subtraction, the mean of every individual feature is being subtracted from the data which can be interpreted as centering the data around the origin from a geometric point of view. This is true for every dimensionality.

Normalizing the data after the Mean subtraction step results in a normalization of the data dimensionality to approximately the same scale. Note that the different features will loose any prioritization over each other after this step as mentioned above. If you have good reasons to think that the different scales in your features bear important information that the network may need to truly understand the underlying patterns in your dataset, then a normalization will be harmful. A standard approach would be to scale the inputs to have mean of 0 and a variance of 1.

Better to predict in high freq dataset

1. with no noise, no large jump



Feature selection idea

Leave one out and compare the outcome

Upgrade colab’s RAM

<https://towardsdatascience.com/upgrade-your-memory-on-google-colab-for-free-1b8b18e8791d>

return -> 不變動的改成 後面往前補 -> 看看lstm會不會比較明顯

ex

mid: 126.75 126.75 126.75 126.75 127.25

return: ln(127.25/126.75) …………….

3/23

賣 0.15% 手續費

以天為單位(重新讀data fame 日期不要drop掉) 每天重新開始

權益圖 return \* 1000

1 0 -1 if 多且一直 1 -> 維持部位1 空訊號進入 1 -> 0 再空訊號 0 -> -1 以此類推

Equity = pd.dataframe……….

for i in range(len(df)):

if signal = 1 …. 0…. -1

……

3/24

空 -> 預測跌0.1%再空

多 -> 預測漲0.2%再買

**問題**

當某個tick的deal quantity = 0時，**只能買在ask1 / 賣在bid1** ?

處理方法?

如果deal qty =0 and side有變化 -> 將actual return 改成

ASK1/DEAL(t-1) or BID1/DEAL(t-1) or

ASK1/MID(t-1) or BID1/MID(t-1)

上述可以直接加在strategy裡面 (多一個if qty=0 and else:~~~~)

或是return算法直接全部改成 ask1/mid(t-1) or bid1/mid(t-1)

策略問題!!

When I =0 -> 應該只有side沒有return

之後試試預測mid price 並看其return的變動當訊號

4/13

Long side: 0.003, 0.0025, 0.002 with Short side = -0.001 perform best in test set.

逐筆資料

Bid1沒有為0的值，但Ask1有不少(1660筆)為0，也都發生在train set(前10天)，而其中1660筆的Ask2都不為0

問題

1. 當ask1為0時，如何去算中價

目前方法，mid1以bid1來代替。

另外當side = 1要買時，如果ask1=0，以買ask2為替代。

1. 是否可以使用bid1 = mid1這個方法
2. 當ask1 = 0時，買不到!!! return怎麼算? -> 買在ask2

4/14

1. 逐筆搓合的data performs bad in test set，價格低估，但趨勢有跟上。
2. 5秒搓的結果，in test set , when long side = 0.0015, short side = -0.001 performs the best(1倍多)
3. 逐筆搓的結果，in test set , when (0.0015, -0.003), (0.001, -0.003) performs the best(3.5%)
4. Return 作為response (5秒搓)

Results: long side = 0.0001, short side = -0.0001 performs the best (超過2倍)

1. Return 作為response (逐筆搓)

(1.6e-05, -5e-05)

(1.2e-05, -4e-05)

(1.2e-05, -5e-05)

Almost 5% in test set

1. Variable selection

注意：避免Function的window寫錯，dataset最後一行都要放response(mid1 or return)

漲停，ask不會有值

Ask1 = 0 -> 漲停! -> 買不到要注意

正確的data 3227\_2.csv

逐筆搓合的都要改

4/15

Data 3227\_2

有時會出現有bid1/ask1，但是bidqty1/askqty1 = 0 (各223/178筆)

但是這種時候會有成交價(剛好為bid1/ask1)、量

* 應該買不到那個價，不過可改可不改，影響不大

資料少了4/13的資料 (我先drop掉4/14的data)

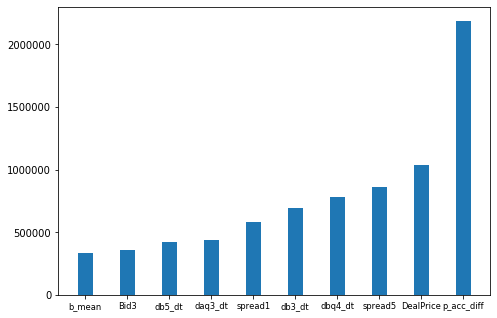
問題

收盤漲停或跌停，空單或多單結不掉怎麼辦? 留到下一天?

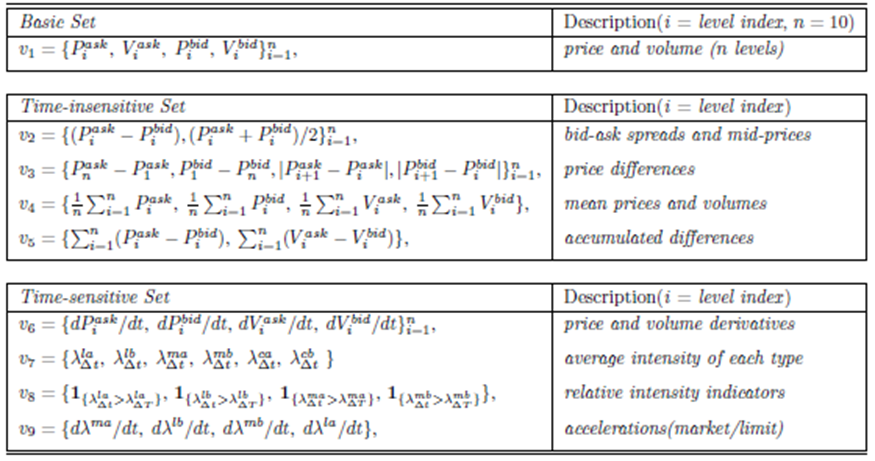
目前不影響，因為不會在收盤且漲停時出場

下一個工作：新增其他指標(duration, max drawdown, sharp ratio………..等等)、試試其他標的

4/20



Variable importance





1. importance
2. 策略修改return方式
3. 算return std sharp odd ratio mdd skew……

4/21

交易策略修正

1. 損益的算法
2. 出場的策略

Performance

問題

績效差

看了transaction history

全部用機器學習來判斷進出場(side)

發現出場都太慢了，等到要出場時，多單已經跌光 / 空單已經漲上去，所以虧很多錢

進場用side判斷，出場策略另改其他方法?

Hard stop: mid1(i-1) / ask1(買價) or mid1(i-1) / bid1(賣價) \* (-1) 虧到多少就出場

新策略

進場看side的變化

出場用hard stop(賺2.5% 跌2.5%出場)

下一個工作

1. 加上未實現損益

損益 = 前一筆已實現 + 未實現(當已時現時，未實現歸零 -> 損益 = 已實現損益 + 0)

1. 解釋variable importance

LSTM bad performance in stock price

https://cloud.tencent.com/developer/article/1395797

4/22

1. 損益(已實現、未實現)
2. Performance
3. 解釋variable importance

Max drawdown 指帳戶淨值從最高點的滑落程度，

意義在於，從任一時間點進場可能遇到的最糟狀況。

新的工作

1. 增加新的參數 (不只是橫向的差，也考慮看看縱向的)

Ex: 一段時間的價差

1. 看144個參數train出來的結果

4/27

64 paras

0915 ~ 1039

New data (1 month)

Previous results in test set gave some bad performance because of over-fitting.

Hence, I turned a parameter, which is called “patience”, into 5 that is when validation loss did not decrease after 5 steps then the model stopped.

After that, the performance in the test set got much better.

Train

trades: 22 (21 long, 1 short) return: 8%

Test

Trades: 4 (4 long) return: -7.68%

注意：a21~a54有錯要改，改完看unique() OK (lstm\_all 記得改)

拿到一個月的資料後，data處理完先去看各個features的unique()，看有沒有都是同樣的，要drop掉

4/28

問題

1. 目前是用return(mid(t) / mid(t-1))來判斷是否要進場

之後可以用10個tick的momentum之類的看嗎? Ex t ~ t-10 ticks的動量

1. 內外盤的量可以當因子，去判斷是否進場?

3406 data 616、617 (2020-03-23)

時間從9:22跳到9:41

3/23 3/24 4/13資料有缺值 不能用 drop 掉

設更好的停利點

沒辦法在peak出場 -> 達到peack後 如果下降一點就出場

Ex 停利的hard stop 設大一點 (例如3%)

Trail stop 設 unrealized return達到peak後 下降多少就出

Ex (side 1 -> -1 and 跌幾% 停損) or (side 1 -> -1 and 長幾%) 停利

目前是前10盤 預測下一盤

但是發現這樣子都買不到起點

Ex 改成前10盤 去預測 後第10盤的值

除了用tick去分，也可以用秒來區分

4/29

目標

1. OK

1 -> -1: 結掉多單之外，還要再空一筆。

-1 -> 1: 結掉空單之外，還要再多一筆。

改為用前10盤，預測下一個10盤

Ex: 1~10 -> 21

Maybe 1~10 -> 11~15 ， 6~15 -> 16~20 ?

不一定要以tick的劃分，也可以用秒來分

Ex: 用前一秒tick的所有特徵值，預測下一秒最後一個tick的價格

判斷side(1.0.-1)的基準，進出場可以不同

Ex: 進場的side(1.0.-1)用0.003、-0.0015

出場的side(1.0.-1)用0.002、-0.001

100個para

940 ~ 13:40

新發現

反過來做賺錢

MA

tick(i) > past 10 tick MA ~% -> side = -1

tick(i) < past 10 tick MA ~% -> side = 1

結掉多單，再空一筆：賺1% + side反轉(1 -> -1)

結掉空單，再多一筆：賺1% + side反轉(-1 -> 1)

結單hard stop：賺2% or 虧1%

一般的pred\_return

tick(i) > tick(i-1) ~% -> side = -1

tick(i) < tick(i-1) ~% -> side = 1

結掉多單，再空一筆：賺1% + side反轉(1 -> -1)

結掉空單，再多一筆：賺1% + side反轉(-1 -> 1)

結單hard stop：賺2% or 虧1%

猜測: tick跟tick之間假如達到條件(ex: 上升0.3%)，有可能代表已經漲得差不多了，即將反轉

MA的概念同上

5/4

方法

1. 反向做 (train set 賺)
2. Using MA正向做 -> didn’t work
3. Forecast futures 5 ticks -> didn’t work

新工作

Lstm R squared? MSE ….等看模型好壞的方法

預測報酬(response change to return) 拉遠一點

Ex 1~10 -> 20，再拿10跟20去比

5/5

1. 1~10 tick -> 10 (return or mid1) -> didn’t work

只預測下一根的情形會發生太晚才進場/出場的狀況，因為一根一根去看除非某tick之間有大漲/大跌才會有訊號，常常到了高點才進場，虧很多才出場。

1. 1~10 tick -> 10 (mid1) + MA -> didn’t work
2. 1~10 tick -> 11~15 (mid1)-> work! All train set have profit, test set signal 太少 沒啥買賣 但賺錢
3. 1~10 tick -> 20/10的return，再於11判斷進場或出場

嘗試改成直接預測報酬，用1~10個tcik的資料去預測，第10、20個tcik之間的報酬，判斷於第11個tick是否要進場或出場。 (Train set 7個賺 test set 都賠)

Train, test的比例要重新分

原本是用日期去7:3，但發現train set太少(總筆數3:1 -> 調成8:2)，training學不到空頭的訊號

導致在多頭的時候(train set)可以賺20%以上，trade 20次

空頭的時候(test set)只成交個位數次，雖然有賺(3%左右)，但交易次數太少

或是拉長一點資料 4/24~5/5

新工作

(1) #setting parameters

import itertools

long\_side, short\_side, stop\_1 = [0.0015, 0.0017, 0.002], [-0.0037, -0.004, -0.0045], [-0.01, -0.015, -0.02, -0.025]

paras = list(itertools.product(long\_side, short\_side, stop\_1))

paras

進出場的side最好設一樣 -> [0.0015, 0.0017, 0.002], [-0.0015, -0.0017, -0.002]

設一個 1->0 or -1 -> 0的點

Ex: - 0.0015 / 2

注意：一天理想的trades次數大概為10次，所以要增加交易次數，較為可信

(2)

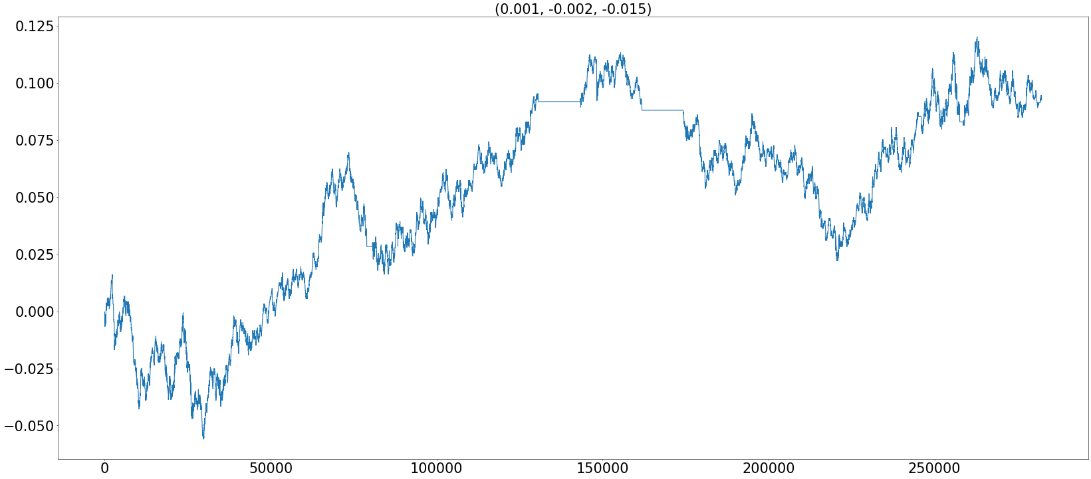
新增判斷模型好壞的標準

Ex: R-squared

5/6

**預測下5根mid (only side + hard stop) in train set**

Return > 0, trades 最多的

(0.001, -0.002, -0.015)

return: 0.09244541667498148

std: 0.04603817506174504

sharp ratio: 1.8950667909397039

trades: 39.0

odds ratio: 0.46153846153846156

mdd: 0.07413817837358117

skewness: -0.526419565339529

trades最多次的，return最差

(0.001, -0.001, -0.015)

return: -0.2091557029840436

std: 0.06283702812275517

sharp ratio: -3.4112960047265344

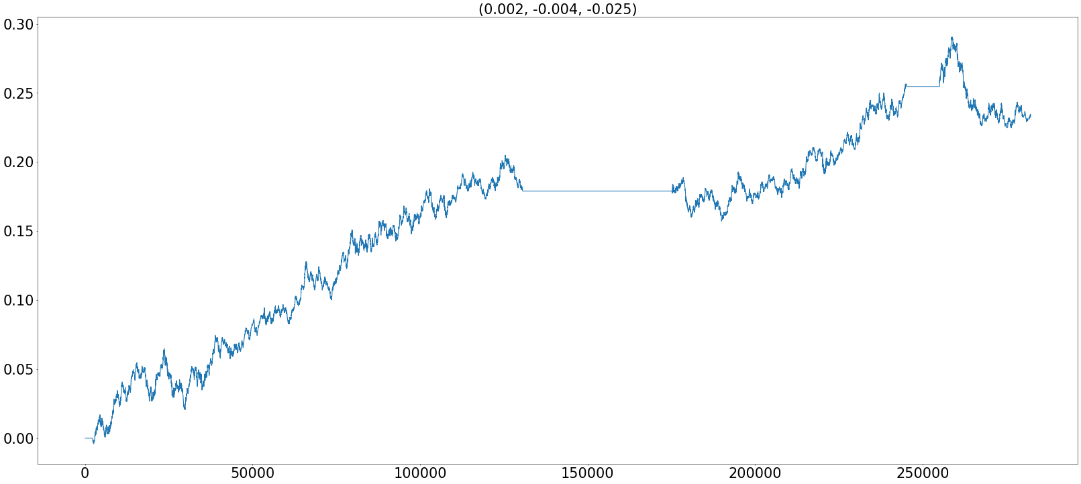
trades: 90.0

odds ratio: 0.3

mdd: 0.2187779864280577

skewness: -0.359426620893958

績效最好

(0.002, -0.004, -0.025)

return: 0.23245996103219385

std: 0.0718101575831413

sharp ratio: 3.1647327993825085

trades: 13.0

odds ratio: 0.6153846153846154

mdd: 0.027586956521739092

skewness: -0.5381430922746622

**預測下5根mid (only side + hard stop) in test set**

接下來我將在train set return > 0且交易次數 > 30與績效最好的一組參數，放到test set看其表現

(0.002, -0.004, -0.025)

return: 0.03982231404958675

std: 0.0188681881375156

sharp ratio: 1.8349570079146729

trades: 1.0

odds ratio: 1.0

mdd: 0.0015

skewness: -0.817343570275094

(0.0015, -0.002, -0.015)

return: 0.030016275649718982

std: 0.02523345655467455

sharp ratio: 0.9834671518722914

trades: 8.0

odds ratio: 0.625

mdd: 0.037004578109592956

skewness: -0.4091385636839937

r-squared problems in time-series, especially in stock price or exchange rates

Let’s say, I predict that tomorrows Israeli Shekel - US Dollar exchange rate will equal to the rate today. If I collect enough data, and compare my predictions with the actual numbers, I will get a pretty high R2. Why is that? Because there is an obvious connection between tomorrow’s and today’s exchange rates — they are intrinsically dependent one on another.

If you have time series data and your response variable and a predictor variable both have significant trends over time, this can produce very high R-squared values. You might try a time series analysis, or including time related variables in your regression model, such as lagged and/or differenced variables. Conveniently, these analyses and functions are all available in Minitab statistical software.

pip報錯，解決方式

pip install --upgrade setuptools --trusted-host pypi.python.org --trusted-host files.pythonhosted.org --trusted-host pypi.org

pip install --upgrade pip --trusted-host pypi.python.org --trusted-host files.pythonhosted.org --trusted-host pypi.org

pip install torch torchvision --trusted-host pypi.python.org --trusted-host files.pythonhosted.org --trusted-host pypi.org

5/11

預測下5根mid (side + hard stop/轉換 1 ->0 or -1 -> 0) in train set

交易次數普遍增加，然而獲得正報酬的仍是交易次數較少的。

績效最好，交易次數不多

(0.002, -0.004, -0.025)

return: 0.2199952739765666

std: 0.05810455812167254

sharp ratio: 3.6967026498468396

trades: 14.0

odds ratio: 0.5714285714285714

mdd: 0.01993670259000596

skewness: -1.1422594793016863

return >0，trades最多次的

(0.0015, -0.002, -0.015)

return: 0.04622189860742379

std: 0.03167364851205474

sharp ratio: 1.2951428248567949

trades: 40.0

odds ratio: 0.375

mdd: 0.07856295713464312

skewness: -0.33516867284914403

test set

(0.002, -0.004, -0.025)

return: 0.03982231404958675

std: 0.0188681881375156

sharp ratio: 1.8349570079146729

trades: 1.0

odds ratio: 1.0

mdd: 0.0015

skewness: -0.817343570275094

(0.0015, -0.002, -0.015)

return: 0.030016275649718982

std: 0.02523345655467455

sharp ratio: 0.9834671518722914

trades: 8.0

odds ratio: 0.625

mdd: 0.037004578109592956

skewness: -0.4091385636839937

目前想法

要增加交易的次數，直接用side判斷基準/2 \* (-1)來當作把單結掉的方式可能不是很好

要想其他更好的停利點

預測下10根tick，績效全部為負報酬

目標

1. 提升預測return的準確度(r-squared)
2. 增加交易次數，同時報酬為持 > 0

Dig into lstm

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

<http://resuly.me/2017/08/16/keras-rnn-tutorial/>

timedistributed

<https://blog.csdn.net/u012193416/article/details/79477220>

flatten

<https://blog.csdn.net/qq_33221533/article/details/82256531>

通常賺最多的，都是標準設最窄的，也就是說交易次數會較少 -> 僥倖賺到?

改進

1. 標準調寬(原本調寬會導致虧損) + 增加進場點判斷標準 ex: ask bid 的量，看偏哪一邊，才進場

3/25 ~ 5/5 (3406)

Data mining、爬蟲、SQL、BERT (李宏毅)

Data mining project BERT伺服器

10.220.26.55:8888

密碼: 1234

5/12

試試看：進場標準放寬，並新增判斷進場標準來篩選

Bid/ask Qty Ratio

Odds ratio下降、Return部分上升，部分下降

Data mining 工作

1. Tag list(不要重複)存成一個list -> csv
2. 嘗試去分類

3/25 ~ 5/5 (3406)

Lstm mid 5t 2\_1

挑在train set中，return > 0、trades > 28 ，且其中odds ratio最高的參數丟進去test set

(0.001, -0.0025, -0.025)

return: 0.05936454665265439

std: 0.04475572666366394

sharp ratio: 1.2102260579902335

trades: 29.0

odds ratio: 0.4827586206896552

mdd: 0.0860475683245886

skewness: -0.9906911477236108

In test set

(0.001, -0.0025, -0.025)

return: 0.01605752288924983

std: 0.02579792057686661

sharp ratio: 0.42086814155811997

trades: 11.0

odds ratio: 0.45454545454545453

mdd: 0.03850070200093014

skewness: 0.6649502140199656

lstm mid 5t\_2\_2

為了增加交易次數，此為2\_1再加上1 -> 0 / -1 -> 0的標準出場

挑在train set中，return > 0、trades > 28 ，且其中odds ratio最高的參數丟進去test set

(odds ratio普遍低，丟進test後虧錢)

Lstm mid 5t\_2\_3

試試看：進場標準放寬，並新增判斷進場標準來篩選

Bid/ask Qty Ratio

Odds ratio下降、Return部分上升，部分下降

嘗試加上bid ask qty ratio當作判斷標準，不過doesn’t work

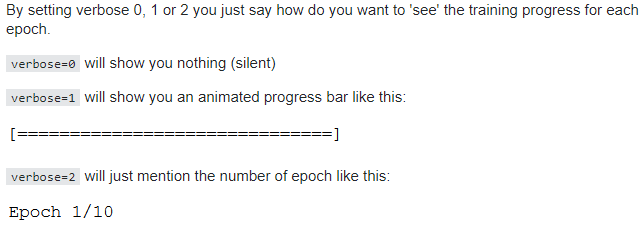
5/13

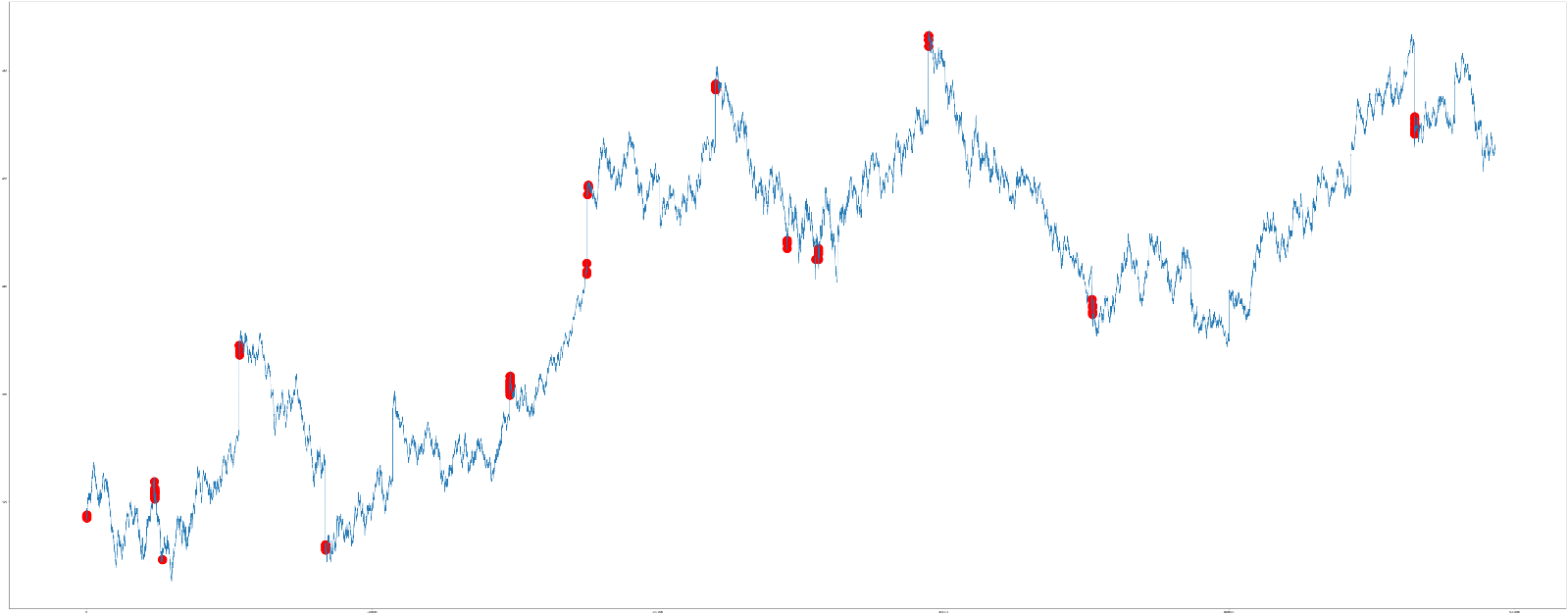
統計各個tags出現的次數 (done)

Variable importance最高的變數: 'p\_acc\_diff'

嘗試將這個變數納入策略考量

Verbose





上圖為mid1 price以及p\_acc\_diff(sum(ask1~5-bid1~5))的疊圖

可以發現的是，p\_acc\_diff > 25的多發生在兩個情形

1. 漲到局部高點了，Ask掛更高賣出(賭追高?)
2. 已經跌到局部低點了，Bid掛更低買入(賭還會再下探?)

策略想法

(1) 50根mid1去做MA，如果前一盤的mid1 > MA(50) -> 上漲趨勢，否則是下跌趨勢

(2) p\_acc\_diff，把p\_acc\_diff > 25的index找出來

將1、2去結合，

如果**p\_acc\_diff == 1 & mid1 > MA(50)** -> short side

如果**p\_acc\_diff == 1 & mid1 < MA(50)** -> long side

return: 0.1601908531659454

std: 0.04850987618508289

sharp ratio: 3.195037080173091

trades: 34.0

odds ratio: 0.29411764705882354

mdd: 0.04192852532920327

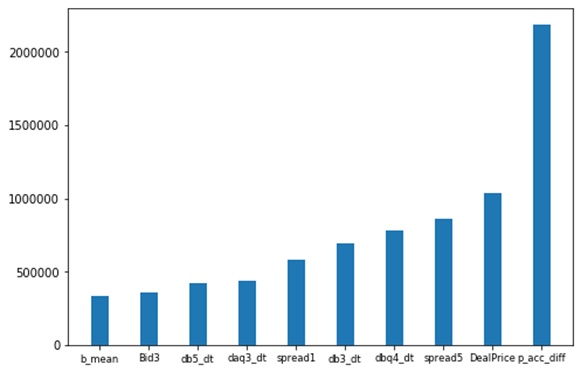
skewness: -0.4164332666918991

也許可以嘗試

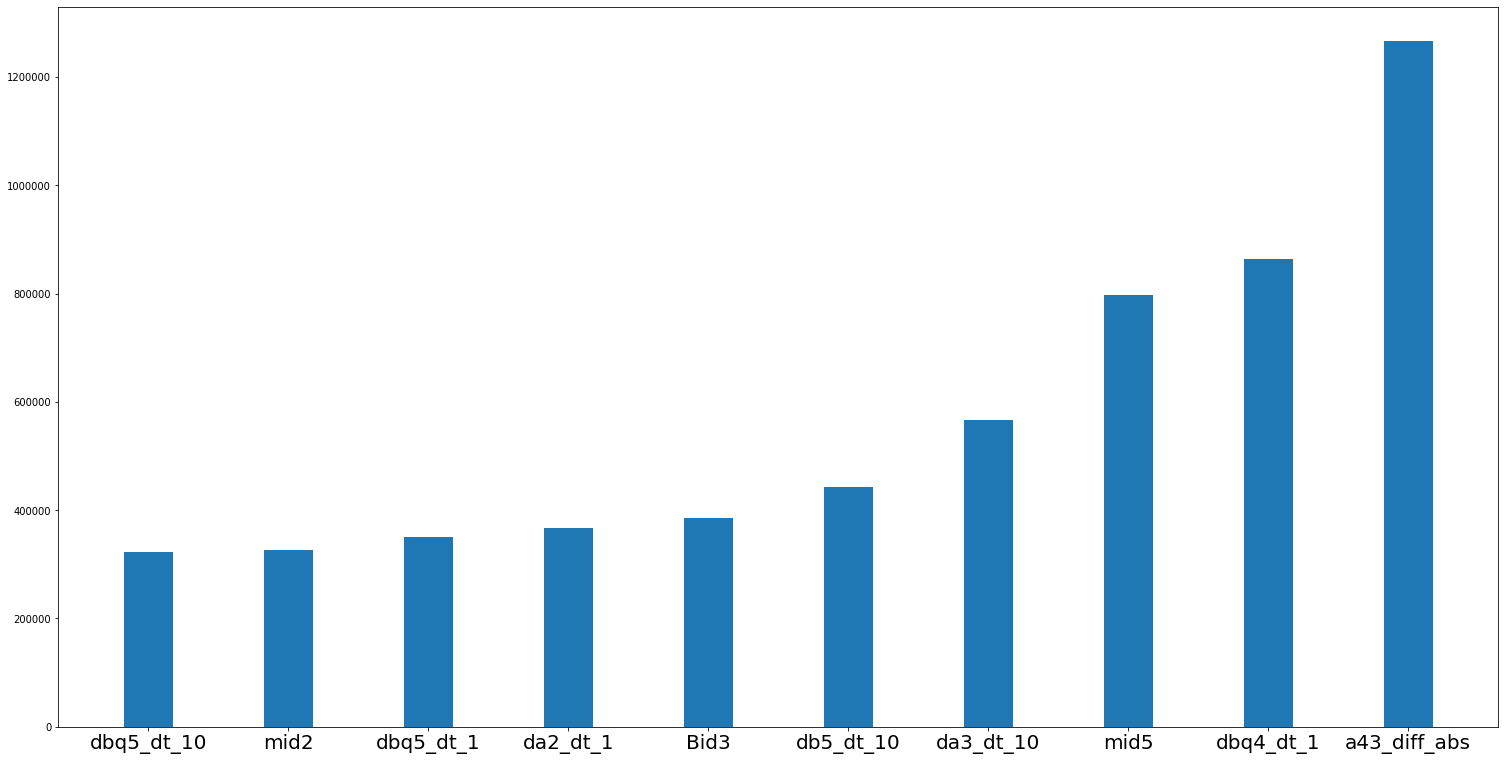
原始策略設寬鬆的標準 + p\_acc\_diff去決定進出場

5/18

3227



3406



Try to tune parameters to increase return’s r-squared

Train set: 0.1 -> 0.12

5/19

1. return\_10t + MA(50)

side == 1 and ma\_side == 0 -> long

1. try mid\_5t + MA(50)
2. p\_acc\_diff + MA(50) train test spli (6:4)
3. **response: return, 將deal price等股價轉換成報酬形式試試看**

**先fill 0 with previous value**

**之後有關price的變數，全部轉為過去10筆的return形式**

**Ex: 10~19 (X) -> return(1~10) … return(10~19) (X) -> return\_mid(20~29) Y**

1. try 進場(0->1/-1) side + MA 出場 side

NLP

1. 嘗試不影響acc的情形下，增加FP，降低TN
2. 中文字文本，保留標點符號，績效是否提高
3. BERT，512字限制，擷取新聞開頭+結尾，drop中間(maybe不重要)，而不是只簡單取前512字

5/20

std、its shift(1 5 10)

<http://10.220.26.189:8888/tree>

選因子 伺服器