Data science final report

Stock Price Prediction Based on the Amount of Five Ticks

利用台股五檔資訊來預測股票漲跌

Team member :

羅翊展 f05942036

王冠驊 r05942073

動機 :

瞬息萬變的金融市場，股價的變化讓人難以捉摸，如何在高點買進，低點賣出是眾人所追求的聖杯。然而，若單以人類的能力，我們很難在短時間內精確的判斷下一個時刻是否是一個好的買點。

有鑒於現今電腦運算速度的提升，我的得以利用複雜的電腦系統挾帶速度優勢，搭配演算法，進行非常快速的證券交易。

方法 :

相較於其他股票市場，台股有個特點是會提供 5 檔的資訊來讓投資者判斷是否是一個好買點。5 檔是市場的即時動向資料，可以讓投資者知道現在每一檔價格有多少人想買入或賣出。

我們將這個題目視為一個回歸問題。我們想利用現今各種熱門的機器學習模型讓我們能夠在極短的時間內，準確的預測下一個時刻股市的股市走向，藉此抓住獲利的契機。在實驗的過程中，我們除了會比較不同機器學習模型的效果，亦會測試對於不同的輸入資訊與預測準確度的關聯。

說明範例

假設有一檔股票，上一個成交價為 43.9 元，交易所便會把「目前未成交的買單，價格最高的五檔（上圖左側的報價）」，以及「目前未成交的賣單，價格最低的五檔（上圖右側的報價）」揭示出來，合計稱為「五檔報價」。

以上圖範例來說

最佳五檔買價（術語為 內盤），分別為:

43.75，105 張；43.70，224 張；43.65，187 張；43.60，399 張；43.55，256 張

最佳五檔賣價（術語為 外盤），分別為:

43.90，18 張；43.95，23 張；44.00，50 張；44.05，77 張；44.10，20 張

未成交的買單（內盤，圖中左側的報價）

每個檔位（Tick）皆有幾百張，反之，未成交的賣單（外盤，圖中右側的報價)每個檔位，只有幾十張，傳統分析上來說，這反映出交易者對於這檔股票的心態，「買方比賣方積極」等一下股價走高的機率，會判斷比往下殺，還高出很多！

反之，上圖範例 2，這檔股票的 外盤（委賣，上圖右側的報價）賣壓龐大，每一檔都有幾百張，內盤（委買，上圖左側的報價）卻很空虛，每一檔都只有幾十張，在這樣的情況下，賣方隨時可能按耐不住，大舉把單子抽掉，往內盤去砍，讓股價走低，所以在心態上，假設你今天本來就要賣這檔股票，「最好趕快把它賣掉」，免得等一下股價越走越低。而我們這次研究的目標，就是希望能讓機器自己去判斷，這種極短線的漲跌變化。

實驗內容：

爬蟲資料如下

時間	成交價	成交量	累計	最佳五檔（賣價）	最佳五檔（賣量）	最佳五檔（買價）	最佳五檔（買量）
09:00:31	129	9	1115	129.50_130.00_130.50_131.00_131.50_	1120_682_376_365_300_	129.00_128.50_128.00_127.50_127.00_	1431_1643_860_703_363_
09:00:31	129	9	1115	129.50_130.00_130.50_131.00_131.50_	1120_682_376_365_300_	129.00_128.50_128.00_127.50_127.00_	1431_1643_860_703_363_
09:00:31	129	9	1115	129.50_130.00_130.50_131.00_131.50_	1120_682_376_365_300_	129.00_128.50_128.00_127.50_127.00_	1431_1643_860_703_363_
09:00:37	129	1	1116	129.50_130.00_130.50_131.00_131.50_	1120_682_377_366_300_	129.00_128.50_128.00_127.50_127.00_	1432_1643_860_703_363_
09:00:37	129	1	1116	129.50_130.00_130.50_131.00_131.50_	1120_682_377_366_300_	129.00_128.50_128.00_127.50_127.00_	1432_1643_860_703_363_
09:00:42	129	1	1117	129.50_130.00_130.50_131.00_131.50_	1120_683_377_367_300_	129.00_128.50_128.00_127.50_127.00_	1445_1642_860_703_365_
09:00:47	129.5	17	1134	129.50_130.00_130.50_131.00_131.50_	1103_683_377_368_300_	129.00_128.50_128.00_127.50_127.00_	1518_1641_860_703_365_
09:00:47	129.5	17	1134	129.50_130.00_130.50_131.00_131.50_	1103_683_377_368_300_	129.00_128.50_128.00_127.50_127.00_	1518_1641_860_703_365_
09:00:57	129.5	7	1143	129.50_130.00_130.50_131.00_131.50_	1103_685_383_368_307_	129.00_128.50_128.00_127.50_127.00_	1529_1637_857_704_366_
09:00:52	129	2	1136	129.50_130.00_130.50_131.00_131.50_	1102_684_382_368_305_	129.00_128.50_128.00_127.50_127.00_	1531_1636_857_703_366_
09:00:57	129.5	7	1143	129.50_130.00_130.50_131.00_131.50_	1103_685_383_368_307_	129.00_128.50_128.00_127.50_127.00_	1529_1637_857_704_366_
09:00:57	129.5	7	1143	129.50_130.00_130.50_131.00_131.50_	1103_685_383_368_307_	129.00_128.50_128.00_127.50_127.00_	1529_1637_857_704_366_
09:01:18	129.5	1	1165	129.50_130.00_130.50_131.00_131.50_	1093_690_381_368_309_	129.00_128.50_128.00_127.50_127.00_	1657_1643_856_704_366_
09:01:07	129.5	2	1149	129.50_130.00_130.50_131.00_131.50_	1109_685_382_368_307_	129.00_128.50_128.00_127.50_127.00_	1631_1642_856_704_365_
09:01:12	129.5	15	1164	129.50_130.00_130.50_131.00_131.50_	1094_686_381_368_308_	129.00_128.50_128.00_127.50_127.00_	1653_1643_856_704_365_
09:01:18	129.5	1	1165	129.50_130.00_130.50_131.00_131.50_	1093_690_381_368_309_	129.00_128.50_128.00_127.50_127.00_	1657_1643_856_704_366_
09:01:23	129.5	12	1177	129.50_130.00_130.50_131.00_131.50_	1071_801_381_369_310_	129.00_128.50_128.00_127.50_127.00_	1657_1643_856_704_365_
09:01:28	129.5	1	1178	129.50_130.00_130.50_131.00_131.50_	1069_808_381_369_310_	129.00_128.50_128.00_127.50_127.00_	1657_1643_856_704_366_
09:01:23	129.5	12	1177	129.50_130.00_130.50_131.00_131.50_	1071_801_381_369_310_	129.00_128.50_128.00_127.50_127.00_	1657_1643_856_704_365_

●前處理

○*處理重複資料和缺失的資料*

因為我們是用爬蟲的方法來抓取股票的五檔資訊，所以會因為證卷交易所網頁傳送了重複的資料，或是少資料的情形發生，我們在這裡的作法是將重複的資料(時間、以及五檔變化相同)合併為同一筆資料，而時間間隔太長的資料則延續前筆出現資料(當作沒有變化發生)

○抽取所需的欄位

在我們的預測系統中，我們使用目前成交價、買和賣的五檔價格及單量共 11 欄資訊作為我們的原始的輸入

○將抽取出的資料轉成向量作為模型輸入

我的的模型所輸入的向量大小皆為 17 維的向量，表示如下:

Input = [價差, five_ticks_vector(bid), five_ticks_vector(ask)], len(input)=17

價差代表的是與上一個成交價的差額，也是我們模型所希望預測出的結果，而因為五檔的價格在少部分的例子中，並非都是等間隔，所以我們將買賣的五檔以 16 維的向量表示，每維的所代表的意義是與成交價的的差價(例如目前成交價是 100 買方五檔中的價格為 100.5[5] 101[10] 101.5[15] 102[20] 103[30]， 我們就記為 5 10 15 20 0 30)

●模型

我們擷取了 2015 年整年度有開市的五檔變化共 136651 筆資料，其中 116651 作為模型的訓練使用，後 2000 筆資料作為模型的測試資料。我們的預測模型使用了深度學習中，三中不同的方法來做為比較，分別是最直接的方法，回歸模型; 以及考慮時間性的 LSTM，和先用 CNN 抽取特徵再丟入 RNN 中學習的 CRNN 模型，架構詳列如下:

○Logistic Regression

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 170)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               21888
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 14)                910
=================================================================
Total params: 31,054
Trainable params: 31,054
Non-trainable params: 0
_________________________________________________________________
Train on 116651 samples, validate on 20000 samples
```

○LSTM

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 10, 17)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 128)           74752
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_1 (Dense)              (None, 14)                1806
=================================================================
Total params: 208,142
Trainable params: 208,142
Non-trainable params: 0
_________________________________________________________________
Train on 116651 samples, validate on 20000 samples
```

○CRNN

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 10, 17, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 17, 128)       1280
_________________________________________________________________
elu_1 (ELU)                  (None, 10, 17, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 8, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 5, 8, 128)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 8, 128)         835712
_________________________________________________________________
elu_2 (ELU)                  (None, 5, 8, 128)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 4, 128)         0
_________________________________________________________________
reshape_1 (Reshape)          (None, 8, 128)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 8, 128)            131584
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_1 (Dense)              (None, 14)                1806
=================================================================
Total params: 1,101,966
Trainable params: 1,101,966
Non-trainable params: 0
_________________________________________________________________
Train on 116651 samples, validate on 20000 samples
```

實驗結果：

我們除了使用五檔的變化來作為輸入的資料，也考慮使用更少的 3 檔輸入做為比較對象，另外，還考慮股價常有微擾的問題(上下微幅跳動，總是回到原點)，我們會將一段範圍內的三筆資料做移動平均來消弭這個問題，實際操作上就是改以預測後面第三筆的資料；除此之外，我們也在訓練模型時，會對預測結果 weighting，讓預測結果是 0 的 weight 小一些，使最終的預測出來不要都是 0，三種不同模型的，預測準確率如下:

2330(台積電) 準確率預測表格，分別是操作變因分別為五檔變化、三檔變化、價格平均、以及三種不同模型的比較

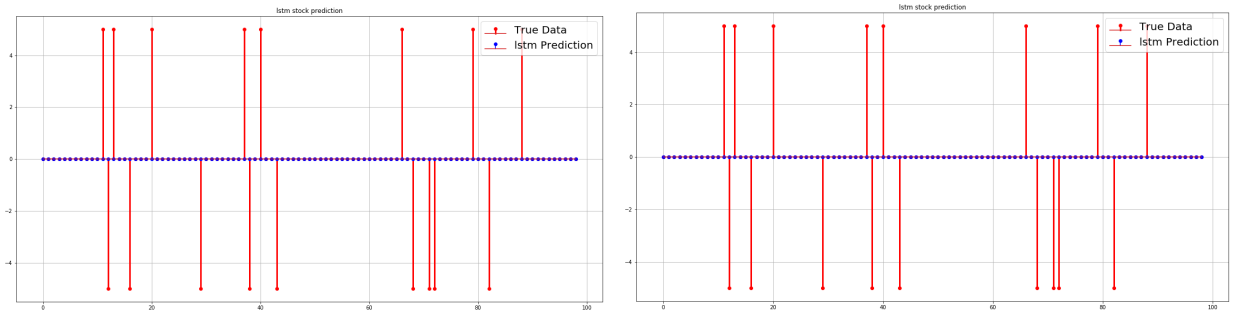
	Logistic regression	CRNN	LTSM
Accuracy(五檔)	0.4625	0.6719	0.6713
Accuracy(三檔)	0.4327	0.4534	0.6755
Accuracy(五檔,平均)	0.6166	0.6166	0.6172
Accuracy(三檔,平均)	0.4108	0.6173	0.6171

3008(大立光) 準確率預測表格，分別是操作變因分別為五檔變化、三檔變化、價格平均、以及三種不同模型的比較

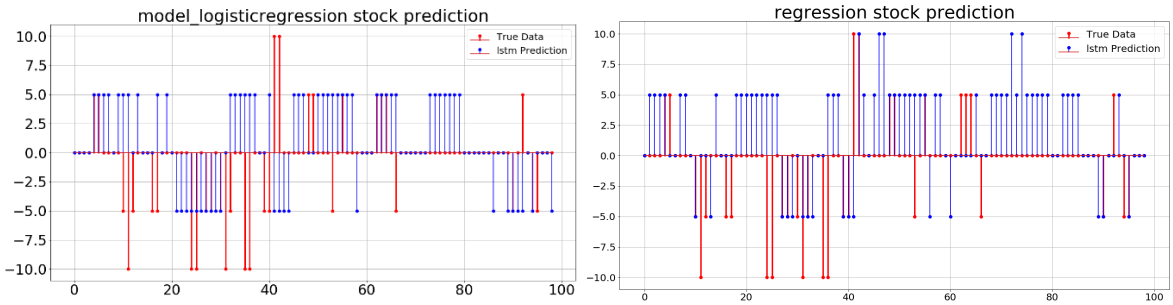
	Logistic regression	CRNN	LTSM
Accuracy(五檔)	0.4007	0.7255	0.7255
Accuracy(三檔)	0.2935	0.3116	0.7255
Accuracy(五檔 ,平均)	0.3585	0.5885	0.5885
Accuracy(三檔 , 平均)	0.3178	0.5885	0.5885

比較五檔和三檔:

我們取出 2000 比測試資料中的 100 出來作圖，觀察結果。其實會發現訓練資料是 5 檔或 3 檔其實差距並不大，2330 lstm 準確率分別是 0.671 和 0.675；3008 logistic regression 分別是 0.4007 和 0.2935 。從 2000 筆測試資料取出 100 筆資料比較如下:



Lstm 2330 五檔(左) 和三檔(右)的比較結果圖一



Logistic regression 3008 五檔(左) 和三檔(右)的比較結果圖二

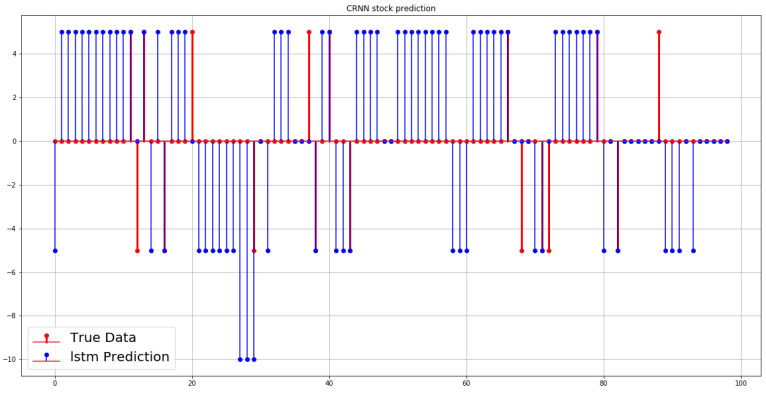
從上圖一會發現，因為股價幾乎沒有甚麼變化，造成 lstm 和 CRNN 預測的常常結果都是 0，如上圖 100 個預測結果都是 0，相較來說 logistic regression 比較沒這種問題(圖二)。

對資料作平均(Unbalance sampling):

檢討發現在訓練資料中，其實股價多是沒變化的，有變化的佔很小一部分，訓練時 sample 到的點多半是 0，所以預測才會都是 0，因此我們針對這點 unbalance sampling 做改進，具體在 code 中如下:

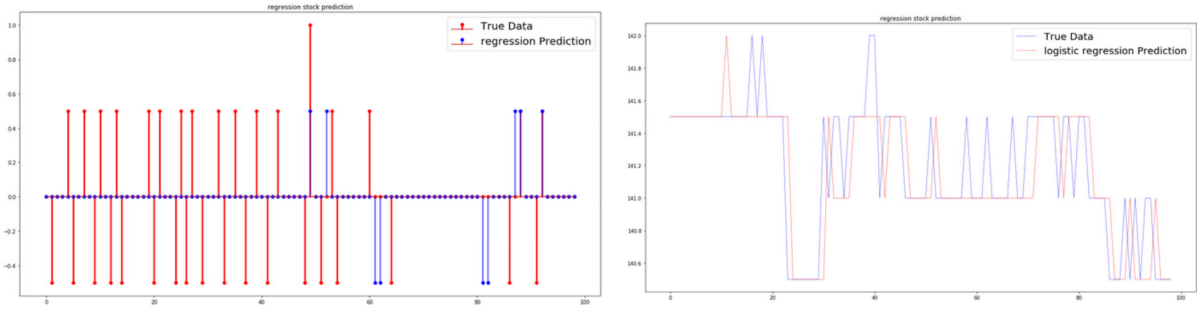
```python
def get_class_weights(y, smooth_factor=0.01):
    """
    Returns the weights for each class based on the frequencies of the samples
    :param smooth_factor: factor that smooths extremely uneven weights
    :param y: list of true labels (the labels must be hashable)
    :return: dictionary with the weight for each class
    """
    counter = Counter(y)

    if smooth_factor > 0:
        p = max(counter.values()) * smooth_factor
        for k in counter.keys():
            counter[k] += p

    majority = max(counter.values())

    return {cls: float(majority / count) for cls, count in counter.items()}
```

我們 sample 時會改變每個 sample 的 weight，如果像 0 出現了很多次，我們就將他的 weight 條小，使他對 model 的影響減少，讓 model 能學到資料(股價)有變化的部分。出來的結果，有好一些，但是常會變成如下圖，變成錯誤很多，因此我們認為怎麼取得平衡很重要。所以加入了平均，來預測後面的第三筆資料(原本是後面的下一筆)

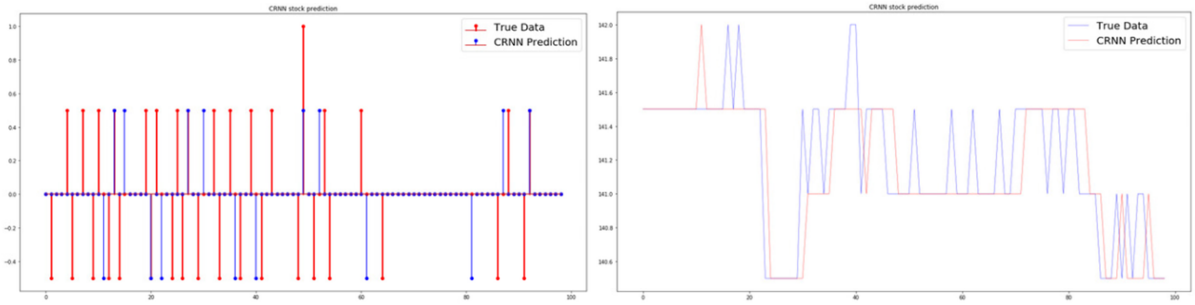


對資料作平均 CRNN 2330 3 檔變化為 input 的結果圖

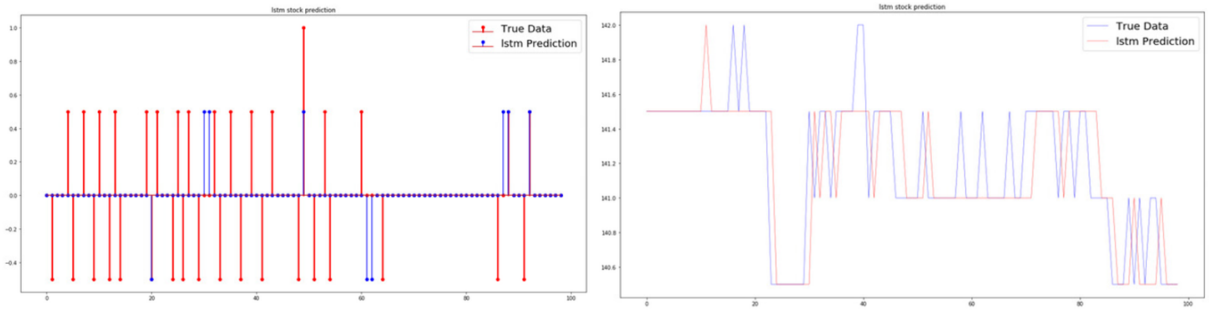
最後我們將準確度較高的預測拿出來看，有兩種作圖方式，一種是原始預測的變化結果，另一種是將成交價加上的還原價格圖(三中模型依序分別是 regression, crnn, lstm)如下:



Regression 預測變化與還原價格圖



crnn 預測變化與還原價格圖



lstm 預測變化與還原價格圖

結論:

在這次的實驗當中，我們測試了不同的方法來比較，最後我們發現其實預測的準確度並沒有辦法到很準，但至少比隨便亂猜來的好一些，而在不同的 constrain 下，我們發現使用五檔以及使用三檔的結果，雖然差不多，但五檔似乎有稍微高一點，推測原因，雖然三檔更能反映即時的股市狀況，但距離較遠的第四、第五檔還是有些微幫助的，例如假單出現的情形，就比較有可能掛在後面一點，也許模型也會將這種狀況加入預測。

而在平均方面，會發現結果很容易就會出現 0，因此加入了 weight 來預測盡量不要都沒變化(0)，最後相互配合的結果可以讓結果更加合理。

此外，因為我們這次測試的兩隻股票多為權值股，因此股價長期走勢受外資及政府政策影響，但這種都是中長期的變化，是我們這種模型無法預測的，也許以後可以加入考量(分點的主力)當作訓練時的輸入之一。此外如果有機會也可以將這些架構拿來測試股性更為活潑不穩定的股票，也許會有更好的預測。但這就會和我們一開始的動機不太一樣，我們初衷是想選一支看漲的穩定股票讓機器去盯盤看五檔變化，判斷適合的買點。

總結來說，我們訓練完的模型能提供投資人判斷看好一支股票長期的走勢後，提供買賣即時的參考，找到適合的買點，增加投資者獲利，不用一直盯盤觀察。理論上，只要準確率高於 50%，此模型就能夠作為比亂猜更好的依據。目前台灣對於五檔的方分析方式仍舊十分欠缺，希望藉由這次的 project 能夠在這塊金融市場建立能量。