計算績效指標

製作人：黃宥輔

交易後 - KPI指標

- 從交易紀錄中，可以計算出各項指標來評估策略的好壞：

類別	指標
獲利	累計報酬率
	平均報酬率
風險	最大回落(MDD)
	獲利標準差
獲利/風險	獲利因子
	賺賠比
	期望值
	夏普指標
人性指標	交易次數
	勝率
	最大連續獲利與虧損次數

交易後 – 獲利指標

- 依據獲利金額計算出來的指標，數值越高表示獲利能力越佳

1. 累計報酬率

   ➢ (買入總支出 ÷ 賣出總收入 – 1) × 100%

2. 平均報酬率

   ➢ 單筆報酬率加總 ÷ 交易次數(處買賣視為一次)

交易後 – 風險指標

- 評估交易過程中所受的風險程度的指標，

  數值越低代表承受的風險較小

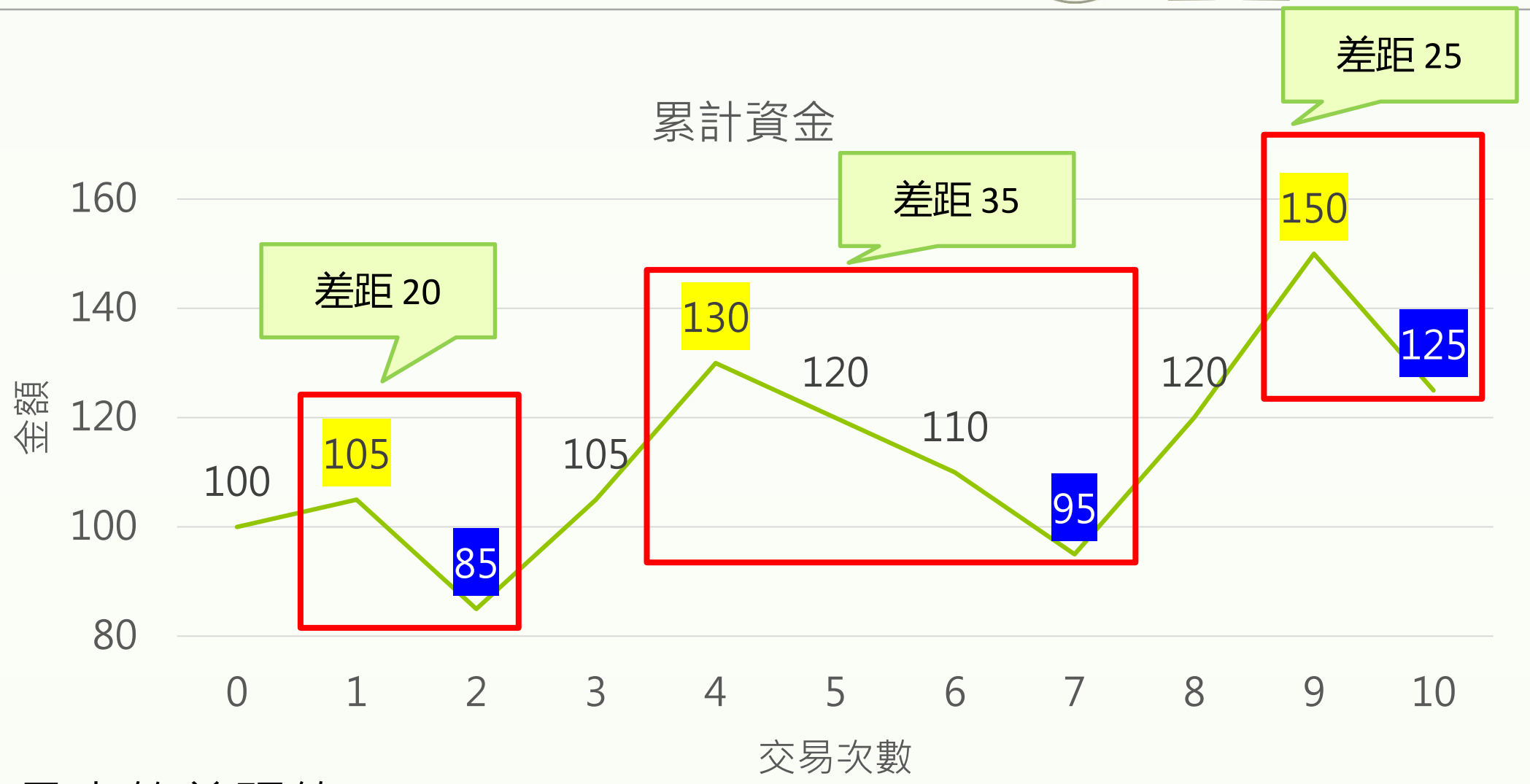
  1. 獲利標準差
     - ➢ 報酬率的標準差
  2. 最大回落(Maximum Drawdown，MDD)
     - ➢ 累計報酬率到達頂峰時，最多會回落多少

MDD範例

交易次數	累計資金	歷史高峰	最高峰差距	MDD
0	100	100	0	0
1	105	105	0	0
2	85	105	20	20
3	105	105	0	20
4	130	130	0	20
5	95	130	35	35
6	120	130	10	35
7	130	130	0	35
8	140	140	0	35
9	150	150	0	35
10	125	150	25	35

MDD範例

MDD範例



- 最大的差距值 = MDD = 35

交易後 – 報酬 / 風險指標

- 同時考慮獲利與風險的指標

1. 獲利因子
   - 在賠 1 元的情況下，期望能夠獲利多少錢
   - (賺錢的交易總金額 ÷ 賠錢的交易總金額) – 1
   - 獲利因子需要大於 1，此交易策略才有賺錢
2. 賺賠比
   - 賺錢的平均金額 ÷ 賠錢的平均金額
   - 假設賺賠機率相同，在承受 1 單位虧損的情況下，能夠有多少倍單位的獲利

交易後 – 報酬 / 風險指標

3. 期望值

- (平均賺錢的金額 × 平均獲利的機率) – (平均虧損的金額 × 平均虧損的機率)
- 計算每次交易可以獲利多少錢，平均獲利/虧損的機率可以使用勝率來計算 (人性指標中會說明勝率)

4. 夏普指標

- (平均報酬率 – 無風險報酬率) ÷ 報酬率標準差
- 無風險報酬率可以使用定存來當作計算標準
- 夏普指標的比率越高，代表相較於無風險報酬，每承受一單位風險獲得的風險溢酬越高
- 用於策略之間比較的參考值，類似於CP值

交易後 – 人性指標

- 數值高低的好壞因人而異，用來評估對交易策略的信心

1. 交易次數
   - 回測期間中，總共交易了幾次，本範定義例一買一賣作為一次完整的交易
   - 交易次數多，交易成本較高；交易次數低，交易策略有效度需要審慎評估
2. 勝率
   - 勝率 = 獲利的交易次數 ÷ 交易次數
   - 勝率高不一定賺錢，勝率低不一定賠錢
   - 如同買樂透，勝率極低，但報酬級高

交易後 – 人性指標

3. 最大連續獲利 / 虧損次數

- 計算過程中，連續獲利或虧損的次數
- 假設某交易策略過程中，每次賠 1%，連續賠 10 次，但後續一次獲利 20%，有可能因為連續虧損次數太多，使用人失去信心，在過程中更換交易策略，因此不會享受到後續的獲利

交易後 – 主程式

```
339  # 定義初始資金
340  cash = 1000000
341
342  fee_rate = 0.001425 # 手續費率，千分之1.425
343  min_fee = 20 # 不足20元以20元計價
344  tax_rate = 0.003 # 證交稅
345
346  trade(train_data, cash)
347  print(record_df)
348
349  print(proc_KPI(record_df, cash ))
350
351  trade(test_data, cash)
352  print(record_df)
353  print(proc_KPI(record_df, cash ))
```

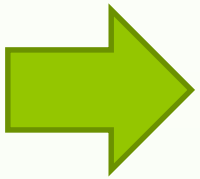
帶入訓練集交易結果與原始資金

帶入測試集交易結果與原始資金

交易後 – 整理交易紀錄

1. 一買一賣視為一次交易，故要將買賣交易進行配對
2. 到最後一日強制平倉，不留任何部位

	time	price	LS	num
0	2015-01-12 00:00:00	40.7	L	24535
1	2015-01-28 00:00:00	42.7	S	24535
2	2015-03-31 00:00:00	44	S	23600
3	2015-05-07 00:00:00	42.5	L	23600
4	2015-06-01 00:00:00	41.1	L	26055
5	2015-10-07 00:00:00	36.55	S	26055
6	2015-10-12 00:00:00	37.35	S	25272
7	2015-11-11 00:00:00	34.5	L	25272
8	2015-11-12 00:00:00	34.25	L	29584
9	2016-02-16 00:00:00	28.3	S	29584
10	2016-02-18 00:00:00	29	S	28615
11	2016-05-04 00:00:00	30.4	L	28615
12	2016-05-05 00:00:00	30.5	L	25817
13	2016-06-03 00:00:00	30.7	S	25817
14	2016-07-05 00:00:00	33.95	S	23140
15	2016-09-14 00:00:00	34	L	23140
16	2016-12-19 00:00:00	36.2	L	21608
17	2016-12-30 00:00:00	34.5	S	21608



	Buy_time	Buy_price	Sell_time	Sell_price	num
0	2015-01-12 00:00:00	40.7	2015-01-28 00:00:00	42.7	24535
1	2015-05-07 00:00:00	42.5	2015-03-31 00:00:00	44	23600
2	2015-06-01 00:00:00	41.1	2015-10-07 00:00:00	36.55	26055
3	2015-11-11 00:00:00	34.5	2015-10-12 00:00:00	37.35	25272
4	2015-11-12 00:00:00	34.25	2016-02-16 00:00:00	28.3	29584
5	2016-05-04 00:00:00	30.4	2016-02-18 00:00:00	29	28615
6	2016-05-05 00:00:00	30.5	2016-06-03 00:00:00	30.7	25817
7	2016-09-14 00:00:00	34	2016-07-05 00:00:00	33.95	23140
8	2016-12-19 00:00:00	36.2	2016-12-30 00:00:00	34.5	21608

交易後 – 整理交易紀錄


```
169  # 計算各項 KPI
170  def proc_KPI( record_df, origin_cash ):
171      trade_df = pd.DataFrame(columns = ["Buy_time","Buy_price", "Sell_time", "Sell_price", "num"])
172
173      Buy_index = 0
174      Sell_index = 0
175
176      for index, row in record_df.iterrows():
177          if row["LS"] == "L":
178              trade_df.at[Buy_index, "Buy_time"] = row["time"]
179              trade_df.at[Buy_index, "Buy_price"] = row["price"]
180              trade_df.at[Buy_index, "num"] = row["num"]
181              Buy_index+= 1
182          else:
183              trade_df.at[Sell_index, "Sell_time"] = row["time"]
184              trade_df.at[Sell_index, "Sell_price"] = row["price"]
185              trade_df.at[Sell_index, "num"] = row["num"]
186              Sell_index+= 1
187
188      # 將買入手續費視為買價的增加(成本增加)
189      trade_df["Buy_price"] = trade_df["Buy_price"] * 1.001425
190      # 將賣出手續費與證交稅視為賣價的減少(獲益減少)
191      trade_df["Sell_price"] = trade_df["Sell_price"] * (1 - 0.001425 - 0.003)
```

交易後 – 整理交易紀錄

函數帶入交易紀錄與初始資金

定義交易紀錄配對的資料表以及索引數

走訪交易紀錄，根據多空方向，記錄到對應的欄位

將買入手續費視為買價的加項(成本的增加)；
將賣出手續費與交易稅視為賣價的減少(獲益減少)

```
169  # 計算各項 KPI
170  def proc_KPI( record_df, origin_cash ):
171      trade_df = pd.DataFrame(columns = ["Buy_time","Buy_price", "Sell_time", "Sell_price", "num"])
172
173      Buy_index = 0
174      Sell_index = 0
175
176      for index, row in record_df.iterrows():
177          if row["LS"] == "L":
178              trade_df.at[Buy_index, "Buy_time"] = row["time"]
179              trade_df.at[Buy_index, "Buy_price"] = row["price"]
180              trade_df.at[Buy_index, "num"] = row["num"]
181              Buy_index+= 1
182          else:
183              trade_df.at[Sell_index, "Sell_time"] = row["time"]
184              trade_df.at[Sell_index, "Sell_price"] = row["price"]
185              trade_df.at[Sell_index, "num"] = row["num"]
186              Sell_index+= 1
187
188      # 將買入手續費視為買價的增加(成本增加)
189      trade_df["Buy_price"] = trade_df["Buy_price"] * 1.001425
190      # 將賣出手續費與證交稅視為賣價的減少(獲益減少)
191      trade_df["Sell_price"] = trade_df["Sell_price"] * (1 - 0.001425 - 0.003)
```

交易後 – 整理結果

	Buy_time	Buy_price	Sell_time	Sell_price	num
0	2015-01-12 00:00:00	40.7	2015-01-28 00:00:00	42.7	24535
1	2015-05-07 00:00:00	42.5	2015-03-31 00:00:00	44	23600
2	2015-06-01 00:00:00	41.1	2015-10-07 00:00:00	36.55	26055
3	2015-11-11 00:00:00	34.5	2015-10-12 00:00:00	37.35	25272
4	2015-11-12 00:00:00	34.25	2016-02-16 00:00:00	28.3	29584
5	2016-05-04 00:00:00	30.4	2016-02-18 00:00:00	29	28615
6	2016-05-05 00:00:00	30.5	2016-06-03 00:00:00	30.7	25817
7	2016-09-14 00:00:00	34	2016-07-05 00:00:00	33.95	23140
8	2016-12-19 00:00:00	36.2	2016-12-30 00:00:00	34.5	21608

可以明確知道交易的
買賣時間點、買賣成交價、成交量

	Buy_time	Buy_price	Sell_time	Sell_price	num
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608

將買賣手續費、證交稅
調整過後的資料
買價(Buy_Price)增加
賣價(Sell_price)減少

交易後 – 交易次數、單筆報酬率

AI.FINTECH
AI 金融科技中心

```
170 def proc_KPI( record_df, origin_cash ):
        ...
193     trade_times = len(record_df) # 交易次數
194
195     # 計算每一筆的報酬率
196     for index, row in trade_df.iterrows():
197         if row["Buy_time"] < row["Sell_time"]: # 做多
198             trade_df.at[index, "ROI"] = row["Sell_price"] / row["Buy_price"] - 1
199         else: # 做空
200             trade_df.at[index, "ROI"] = 1 - row["Buy_price"] / row["Sell_price"]
```

交易後 – 交易次數、單筆報酬率

AI.FINTECH
AI 金融科技中心

```
170  def proc_KPI( record_df, origin_cash ):
         ...
193      trade_times = len(record_df) # 交易次數
194
195      # 計算每一筆的報酬率
196      for index, row in trade_df.iterrows():
197          if row["Buy_time"] < row["Sell_time"]: # 做多
198              trade_df.at[index, "ROI"] = row["Sell_price"] / row["Buy_price"] - 1
199          else: # 做空
200              trade_df.at[index, "ROI"] = 1 - row["Buy_price"] / row["Sell_price"]
```

交易次數，即交易紀錄的資料數

計算每一筆的報酬率，根據時間先後，判定為做多或做空，分別進行處理

交易後 – 單筆報酬率

- 做多：買 100 元，賣 110 元，

  → 利益 10 元，成本 100 元

  → 報酬率 = (110 – 100) ÷ 100

  = (110 ÷ 100) – 1 = 10 %

- 做空：賣 100 元，買 90 元，

  → 利益 10 元，成本 100 元

  → 報酬率 = (100 – 90) ÷ 100

  = 1 – (90 ÷ 100) = 10 %

交易後 – 交易次數、單筆報酬率

	Buy_time	Buy_price	Sell_time	Sell_price	num
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608

交易次數(9次)

交易後 – 交易次數、單筆報酬率

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535	0.043011313
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600	0.028415226
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055	-0.115900566
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272	0.070877589
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584	-0.178549472
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615	-0.054435532
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817	0.000677395
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140	-0.007357409
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608	-0.052528669

計算完成後多出報酬率欄位
之後會使用到

交易後 – 平均報酬率、收益金額、累計報酬率


```
170 def proc_KPI( record_df, origin_cash ):
        ...
202     # 計算平均報酬率
203     avg_ROI = trade_df["ROI"].mean()
204
205     # 獲利標準差
206     ROI_std = trade_df["ROI"].std()
207
208     # 夏普指標，平均報酬率除以報酬率標準差
209     # 無風險利率以 111 年 6 月 22 日發布的郵政 3 年期未達 500 萬定存 1.270% 為準
210     sharpe = ( avg_ROI - 0.01270 ) / ROI_std
211     sharpe = round(sharpe, 2) # 四捨五入到小數點第二位
212
213     # 計算勝率，此處買賣視為一組交易
214     win_rate = (trade_df["ROI"] > 0).sum() / trade_times
215
216     # 計算每一筆的收益金額
217     trade_df["profit"] = (trade_df["Sell_price"] - trade_df["Buy_price"]) * trade_df["num"]
218
219     # 累計報酬率，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
220     acc_ROI = trade_df["profit"].sum() / origin_cash
```

交易後 – 報酬率衍生指標與獲利金額計算


```
170  def proc_KPI( record_df, origin_cash ):
         ...
202      # 計算平均報酬率
203      avg_ROI = trade_df["ROI"].mean()
204
205      # 獲利標準差
206      ROI_std = trade_df["ROI"].std()
207
208      # 夏普指標，平均報酬率除以報酬率標準差
209      # 無風險利率以 111 年 6 月 22 日發布的郵政 3 年期未達 500 萬定存 1.270% 為準
210      sharpe = ( avg_ROI - 0.01270 ) / ROI_std
211      sharpe = round(sharpe, 2) # 四捨五入到小數點第二位
212
213      # 計算勝率，此處買賣視為一組交易
214      win_rate = (trade_df["ROI"] > 0).sum() / trade_times
215
216      # 計算每一筆的收益金額
217      trade_df["profit"] = (trade_df["Sell_price"] - trade_df["Buy_price"]) * trade_df["num"]
218
219      # 累計報酬率，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
220      acc_ROI = trade_df["profit"].sum() / origin_cash
```

將每一筆報酬率平均，獲得平均報酬率

計算報酬率的標準差

平均報酬除減無風險利率除以標準差，取得夏普指標

篩選報酬率大於 0 的交易篩選出來並進行計數，除以交易次數，獲得勝率

交易後 – 報酬率衍生指標與獲利金額計算


```
170 def proc_KPI( record_df, origin_cash ):
        ...
202     # 計算平均報酬率
203     avg_ROI = trade_df["ROI"].mean()
204
205     # 獲利標準差
206     ROI_std = trade_df["ROI"].std()
207
208     # 夏普指標，平均報酬率除以報酬率標準差
209     # 無風險利率以 111 年 6 月 22 日發布的郵政 3 年期未達 500 萬定存 1.270% 為準
210     sharpe = ( avg_ROI - 0.01270 ) / ROI_std
211     sharpe = round(sharpe, 2) # 四捨五入到小數點第二位
212
213     # 計算勝率，此處買賣視為一組交易
214     win_rate = (trade_df["ROI"] > 0).sum() / trade_times
215
216     # 計算每一筆的收益金額
217     trade_df["profit"] = (trade_df["Sell_price"] - trade_df["Buy_price"]) * trade_df["num"]
218
219     # 累計報酬率，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
220     acc_ROI = trade_df["profit"].sum() / origin_cash
```

計算每一筆交易的實際收益

將收益進行加總並除以原始資金，獲得累計報酬率

交易後 – 平均報酬率、收益金額、累計報酬率

將報酬率欄位進行平均，
獲得平均報酬率
-2.95%

計算每一筆報酬金額
儲存成 profit 欄位

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535	0.043011313	43011.20443
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600	0.028415226	29375.805
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055	-0.115900566	-124290.1991
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272	0.070877589	66605.96709
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584	-0.178549472	-181173.4145
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615	-0.054435532	-44972.62168
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817	0.000677395	534.15373
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140	-0.007357409	-5754.426275
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608	-0.052528669	-41146.97998

將報酬金額加總，-257,881
除以原始資金 1,000,000 元，獲得累計報酬率 -25.78%

交易後 – 平均報酬率、收益金額、累計報酬率


```
170 def proc_KPI( record_df, origin_cash ):
        ...
222     # 將賺錢的資料與賠錢的資料分開
223     win_data = trade_df[ trade_df["profit"] > 0 ]
224     loss_data = trade_df[ trade_df["profit"] <= 0 ]
225
226     # 計算獲利因子，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
227     if len(loss_data) > 0:
228         profit_factor = win_data["profit"].sum() / loss_data["profit"].sum()
229         profit_factor = abs(profit_factor) # 計算後一定是負數，所以加上絕對值
230         profit_factor = round(profit_factor, 2) # 四捨五入到小數點第二位
231     else:
232         profit_factor = pd.NA
233
234     # 計算賺賠比，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
235     if len(loss_data) > 0:
236         profit_rate = win_data["profit"].mean() / loss_data["profit"].mean()
237         profit_rate = abs(profit_rate) # 計算後一定是負數，所以加上絕對值
238         profit_rate = round(profit_rate, 2) # 四捨五入到小數點第二位
239     else:
240         profit_rate = pd.NA
```

交易後 – 平均報酬率、收益金額、累計報酬率


```
170 def proc_KPI( record_df, origin_cash ):
        ...
222     # 將賺錢的資料與賠錢的資料分開
223     win_data = trade_df[ trade_df["profit"] > 0 ]
224     loss_data = trade_df[ trade_df["profit"] <= 0 ]
225
226     # 計算獲利因子，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
227     if len(loss_data) > 0:
228         profit_factor = win_data["profit"].sum() / loss_data["profit"].sum()
229         profit_factor = abs(profit_factor) # 計算後一定是負數，所以加上絕對值
230         profit_factor = round(profit_factor, 2) # 四捨五入到小數點第二位
231     else:
232         profit_factor = pd.NA
233
234     # 計算賺賠比，若沒有賠錢，分母為0，會出現ZeroDivisionError，則顯示空值
235     if len(loss_data) > 0:
236         profit_rate = win_data["profit"].mean() / loss_data["profit"].mean()
237         profit_rate = abs(profit_rate) # 計算後一定是負數，所以加上絕對值
238         profit_rate = round(profit_rate, 2) # 四捨五入到小數點第二位
239     else:
240         profit_rate = pd.NA
```

將獲利的資料
與虧損的資料分離

將獲利與虧損的
損益個別加總，
並加上絕對值，
計算出獲利因子

將獲利與虧損的
損益個別平均，
並加上絕對值，
計算出賺賠比

交易後 – 分離獲利與虧損資料

獲利資料

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535	0.043011313	43011.20443
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600	0.028415226	29375.805
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272	0.070877589	66605.96709
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817	0.000677395	534.15373

虧損資料

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055	-0.115900566	-124290.1991
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584	-0.178549472	-181173.4145
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615	-0.054435532	-44972.62168
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140	-0.007357409	-5754.426275
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608	-0.052528669	-41146.97998

交易後 – 分離獲利與虧損資料

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535	0.043011313	43011.20443
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600	0.028415226	29375.805
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272	0.070877589	66605.96709
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817	0.000677395	534.15373

加總：139,527
平均：34,482

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055	-0.115900566	-124290.1991
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584	-0.178549472	-181173.4145
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615	-0.054435532	-44972.62168
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140	-0.007357409	-5754.426275
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608	-0.052528669	-41146.97998

加總：-397,338
平均：-79,468

$$\text{獲利因子} = \left|\frac{139{,}527}{-397{,}338}\right| = 0.35$$

$$\text{賺賠比} = \left|\frac{34{,}482}{-79{,}468}\right| = 0.44$$

交易後 – 期望值、最大連續獲利與虧損


```
170 def proc_KPI( record_df, origin_cash ):
        ...
242     # 期望值
243     expected_rate = win_data["profit"].mean() * win_rate + \
244                     loss_data["profit"].mean() * (1 - win_rate)
245     expected_rate = round(expected_rate, 0) # 四捨五入到整數位
246
247     # 最大連續獲利與虧損
248     max_profit_times = 0
249     max_loss_times = 0
250     times = 0
251     for p in trade_df["profit"]:
252         if p >= 0 and times >= 0: # 累積連續獲利次數
253             times+= 1
254         elif p < 0 and times <= 0: # 累積連續虧損次數
255             times-= 1
256         elif p >= 0 and times < 0: # 轉而累積獲利次數
257             times = 1
258         elif p < 0 and times > 0: # 轉而累積虧損次數
259             times = -1
260
261         if times > max_profit_times:
262             max_profit_times = times
263         elif times < max_loss_times:
264             max_loss_times = times
265     max_loss_times = abs(max_loss_times)
```

交易後 – 期望值、最大連續獲利與虧損


```
170 def proc_KPI( record_df, origin_cash ):
        ...
242     # 期望值
243     expected_rate = win_data["profit"].mean() * win_rate + \
244                     loss_data["profit"].mean() * (1 - win_rate)
245     expected_rate = round(expected_rate, 0) # 四捨五入到整數位
246
247     # 最大連續獲利與虧損
248     max_profit_times = 0
249     max_loss_times = 0
250     times = 0
251     for p in trade_df["profit"]:
252         if p >= 0 and times >= 0: # 累積連續獲利次數
253             times+= 1
254         elif p < 0 and times <= 0: # 累積連續虧損次數
255             times-= 1
256         elif p >= 0 and times < 0: # 轉而累積獲利次數
257             times = 1
258         elif p < 0 and times > 0: # 轉而累積虧損次數
259             times = -1
260
261         if times > max_profit_times:
262             max_profit_times = times
263         elif times < max_loss_times:
264             max_loss_times = times
265     max_loss_times = abs(max_loss_times)
```

平均獲利 × 勝率 + 平均虧損 × 敗率

走訪利潤序列(trade_df[“profit”])
Time 以正數代表連續獲利次數
以負數代表連續虧損次數
走訪過程分四種情況探討

1. 獲利 並次數(time) >= 0，累計連續獲利次數
2. 虧損 並次數(time) <= 0，累計連續虧損次數
3. 獲利 並次數(time) < 0，重置次數為 1
4. 虧損 並次數(time) > 0，重置次數為 -1

最後判斷次數(time)是否有
超過最大連續獲利次數(max_profit_times) 或是
低於最大連續虧損次數(max_loss_times)

符合條件則更新最大連續獲利(虧損)次數
走訪完成將最大虧損次數加絕對值

交易後 – 最大回落(MDD)

AI.FINTECH
AI 金融科技中心

```
170  def proc_KPI( record_df, origin_cash ):
         ...
267      # 計算最大回落(MDD)
268      # 當下的平倉後的資金
269      trade_df["acuumulate_profit"] = trade_df["profit"].cumsum() + origin_cash
270
271      MDD = 0 # 最大回落
272      peak = 0 # 頂峰
273
274      # 走訪累計資金序列
275      for i in trade_df["acuumulate_profit"]:
276          if i > peak: # 當下的資金大於頂峰
277              peak = i # 更新頂峰
278          diff = (peak - i) / peak # 計算當下的累計資金與頂峰差距(比率)
279          if diff > MDD: # 差距大於當下的MDD，則更新MDD
280              MDD = diff
281
282      # 轉換為百分比文字
283      MDD = round(MDD * 100, 2)
284      MDD = f"{MDD}%"
```

交易後 – 最大回落(MDD)


```
170  def proc_KPI( record_df, origin_cash ):
         ...
267      # 計算最大回落(MDD)
268      # 當下的平倉後的資金
269      trade_df["acuumulate_profit"] = trade_df["profit"].cumsum() + origin_cash
270
271      MDD = 0 # 最大回落
272      peak = 0 # 頂峰
273
274      # 走訪累計資金序列
275      for i in trade_df["acuumulate_profit"]:
276          if i > peak: # 當下的資金大於頂峰
277              peak = i # 更新頂峰
278          diff = (peak - i) / peak # 計算當下的累計資金與頂峰差距(比率)
279          if diff > MDD: # 差距大於當下的MDD，則更新MDD
280              MDD = diff
281
282      # 轉換為百分比文字
283      MDD = round(MDD * 100, 2)
284      MDD = f"{MDD}%"
```

計算交易平倉後的累計資金

走訪平倉後的累計資金，尋找頂峰與最大回落

更動MDD的格式為百分比文字

交易後 – 最大回落(MDD)

AI.FINTECH
AI 金融科技中心

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit	acuumulate_profit
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535	0.043011313	43011.20443	43011.20443
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600	0.028415226	29375.805	72387.00943
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055	-0.115900566	-124290.1991	-51903.18964
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272	0.070877589	66605.96709	14702.77745
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584	-0.178549472	-181173.4145	-166470.637
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615	-0.054435532	-44972.62168	-211443.2587
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817	0.000677395	534.15373	-210909.105
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140	-0.007357409	-5754.426275	-216663.5312
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608	-0.052528669	-41146.97998	-257810.5112

累加獲利

交易後 – 最大回落(MDD)

	Buy_time	Buy_price	Sell_time	Sell_price	num	ROI	profit	acuumulate_profit
0	2015-01-12 00:00:00	40.7579975	2015-01-28 00:00:00	42.5110525	24535	0.043011313	43011.20443	1043011.204
1	2015-05-07 00:00:00	42.5605625	2015-03-31 00:00:00	43.8053	23600	0.028415226	29375.805	1072387.009
2	2015-06-01 00:00:00	41.1585675	2015-10-07 00:00:00	36.38826625	26055	-0.115900566	-124290.1991	948096.8104
3	2015-11-11 00:00:00	34.5491625	2015-10-12 00:00:00	37.18472625	25272	0.070877589	66605.96709	1014702.777
4	2015-11-12 00:00:00	34.29880625	2016-02-16 00:00:00	28.1747725	29584	-0.178549472	-181173.4145	833529.363
5	2016-05-04 00:00:00	30.44332	2016-02-18 00:00:00	28.871675	28615	-0.054435532	-44972.62168	788556.7413
6	2016-05-05 00:00:00	30.5434625	2016-06-03 00:00:00	30.5641525	25817	0.000677395	534.15373	789090.895
7	2016-09-14 00:00:00	34.04845	2016-07-05 00:00:00	33.79977125	23140	-0.007357409	-5754.426275	783336.4688
8	2016-12-19 00:00:00	36.251585	2016-12-30 00:00:00	34.3473375	21608	-0.052528669	-41146.97998	742189.4888

累計獲利加上初始資金，
形成平倉後累計資金

交易後 – 最大回落(MDD)

MDD = 1,072,387 – 742,189 = 330,198

→ MDD = 330,198 ÷ 1,072,387 = 30.79%

交易後 – 回傳資料


```
170 def proc_KPI( record_df, origin_cash ):
        ...
286     # 將數值進行四捨五入
287     avg_ROI, acc_ROI, win_rate, ROI_std = round(avg_ROI * 100, 2),\
288                                           round(acc_ROI * 100, 2),\
289                                           round(win_rate * 100, 2),\
290                                           round(ROI_std, 4)
291
292     # 將數值轉換為百分比文字
293     avg_ROI, acc_ROI, win_rate = f"{avg_ROI}%", f"{acc_ROI}%", f"{win_rate}%"
294
295     # 將各項指標整理成字典( dict )格式以便回傳運用
296     output_KPI = {
297         # 獲利指標
298         "累計報酬率" : acc_ROI,
299         "平均報酬率" : avg_ROI,
300         # 風險指標
301         "最大回落(MDD)" : MDD,
302         "獲利標準差" : ROI_std,
303         # 獲利/風險指標
304         "獲利因子" : profit_factor,
305         "賺賠比" : profit_rate,
306         "期望值" : expected_rate,
307         "夏普指標" : sharpe,
308         # 人性指標
309         "交易次數" : trade_times,
310         "勝率" : win_rate,
311         "最大連續獲利次數" : max_profit_times,
312         "最大連續虧損次數" : max_loss_times
313         }
314
315     return output_KPI # 回傳結果
```

交易後 – 回傳資料


```
170 def proc_KPI( record_df, origin_cash ):
        ...
286     # 將數值進行四捨五入
287     avg_ROI, acc_ROI, win_rate, ROI_std = round(avg_ROI * 100, 2),\
288                                           round(acc_ROI * 100, 2),\
289                                           round(win_rate * 100, 2),\
290                                           round(ROI_std, 4)
291
292     # 將數值轉換為百分比文字
293     avg_ROI, acc_ROI, win_rate = f"{avg_ROI}%", f"{acc_ROI}%", f"{win_rate}%"
294
295     # 將各項指標整理成字典( dict )格式以便回傳運用
296     output_KPI = {
297         # 獲利指標
298         "累計報酬率" : acc_ROI,
299         "平均報酬率" : avg_ROI,
300         # 風險指標
301         "最大回落(MDD)" : MDD,
302         "獲利標準差" : ROI_std,
303         # 獲利/風險指標
304         "獲利因子" : profit_factor,
305         "賺賠比" : profit_rate,
306         "期望值" : expected_rate,
307         "夏普指標" : sharpe,
308         # 人性指標
309         "交易次數" : trade_times,
310         "勝率" : win_rate,
311         "最大連續獲利次數" : max_profit_times,
312         "最大連續虧損次數" : max_loss_times
313         }
314
315     return output_KPI # 回傳結果
```

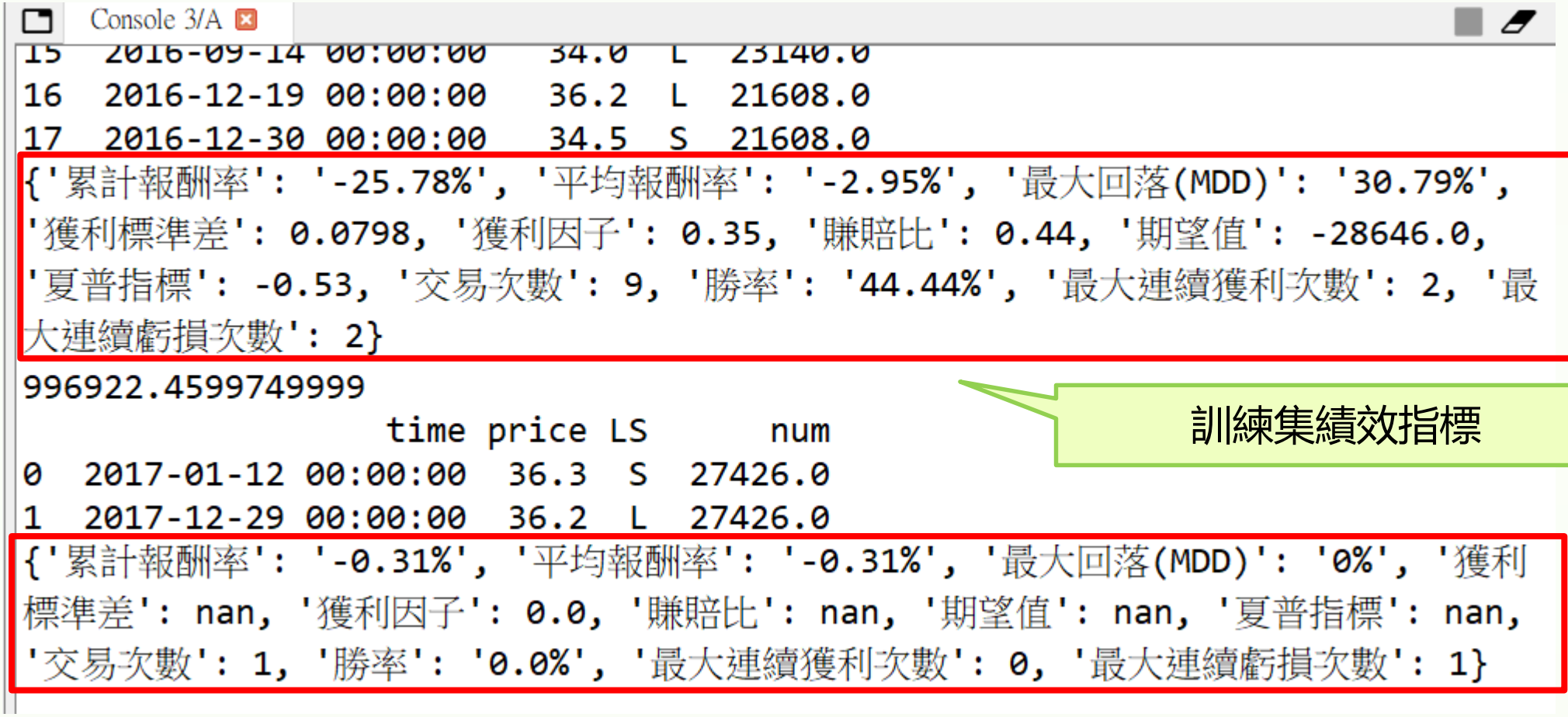
將數值百分比轉換以及四捨五入

將數值整理成帶百分比的文字

將所有指標轉整理成字典(dict)格式，以便回傳利用

回傳結果

執行結果



訓練集績效指標

測試集績效指標

章節到此結束，有任何問題歡迎提出來討論！