一些比較基礎的交易策略：
不一定有好回報，但是有嘗試過了

台大財金三 柯宥圻

僅供TMBA程式交易部門社員申請用

Github連結：https://github.com/blackwingedkite/TMBA_100strategies_challenge/tree/master

Drive連結: https://drive.google.com/drive/folders/1Ne73WBq6KclvSrSj2E1LcNHww7MiJgEv?usp=sharing

目錄

1布林通道系列
2肯特爾通道系列
3KD系列
4RSI系列
5動能系列
6突破系列
7MACD系列
=> 7. MACD策略詳細研究

結論：回報好壞排名

1布林通道系列 +70%
2肯特爾通道系列 +15%
3KD系列 +120%
4RSI系列 +20%
5動能系列 +20%
6突破系列 -10%
7MACD系列 +80%
整體來看，可以考慮KD、MACD指標。
=> 詳細研究MACD 的策略表現，以及可能的成長方向

共用的策略資訊

- ✓ 商品：台指期
- ✓ 週期：60min K棒
- ✓ 交易時段: 早盤(0845-1245)
- ✓ In-sample: 2011-2018
- ✓ Out-sample: 2019
- ✓ 手續費用: 來回1200
- ✓ TMBA-台灣最優質的MBA社團

所有需要的資料前處理 (1)

04.ipynb (5) - JupyterLab

File Edit View Run Kernel Tabs Settings Help

01_BasicBoolean.ipynb | 04.ipynb | Program.ipynb | 02_BasicKeltner.ipynb | 03.ipynb

Code | Python 3 (ipykernel)

[1]:
```python
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import requests
import datetime as dt
import time
rule = '3T'
plt.style.use('ggplot')
pd.set_option('display.max_rows', None)

df = pd.read_csv('TWF_Futures_Minute_Trade.txt')
df.index = pd.to_datetime(df['Date'] + ' ' + df['Time'])
df['Hour'] = df.index.map(lambda x: x.hour)
df.columns = ['date', 'time', 'open', 'high', 'low','close','volume','hour']
Morning = df[(df['hour'] >= 8) & (df['hour'] <= 13)]
Morning.index += dt.timedelta(minutes=15) # 往前調整15mins
Morning.resample(rule=rule, closed='right', label='right').first()[['open']].iloc[0:3]

rule = '60T'
Morning = df[(df['hour'] >= 8) & (df['hour'] <= 13)]
Morning.index = Morning.index + dt.timedelta(minutes=15)
d1 = Morning.resample(rule=rule, closed='right', label='left').first()[['open']]
d2 = Morning.resample(rule=rule, closed='right', label='left').max()[['high']]
d3 = Morning.resample(rule=rule, closed='right', label='left').min()[['low']]
d4 = Morning.resample(rule=rule, closed='right', label='left').last()[['close']]
d5 = Morning.resample(rule=rule, closed='right', label='left').sum()[['volume']]
df_Morning = pd.concat([d1,d2,d3,d4,d5], axis=1)
df_Morning = df_Morning.dropna()
df_Morning.index = df_Morning.index - dt.timedelta(minutes=15)
```

Filter files by name

/

Name	Last Modified
01_BasicBo...	6 days ago
02_BasicKel...	a month ago
03.ipynb	6 days ago
04.ipynb	seconds ago
BTC_insma...	a month ago
data.csv	9 minutes ago
preprocessi...	4 days ago
Program.ip...	6 minutes ago
sample_bo...	6 days ago
settlement...	a month ago
stats.html	a month ago
TWF_Future...	a month ago

Simple 0 5 Python 3 (ipykernel) | Idle Mode: Command Ln 29, Col 17 04.ipynb

下午 11:24 2023/9/10

所有需要的資料前處理 (2)

04.ipynb (5) - JupyterLab

File Edit View Run Kernel Tabs Settings Help

01_BasicBoolean.ipynb | 04.ipynb | Program.ipynb | 02_BasicKeltner.ipynb | 03.ipynb

Code | Python 3 (ipykernel)

```python
rule = '60T'
Morning = df[(df['hour'] >= 8) & (df['hour'] <= 13)]
Morning.index = Morning.index + dt.timedelta(minutes=15)
d1 = Morning.resample(rule=rule, closed='right', label='left').first()[['open']]
d2 = Morning.resample(rule=rule, closed='right', label='left').max()[['high']]
d3 = Morning.resample(rule=rule, closed='right', label='left').min()[['low']]
d4 = Morning.resample(rule=rule, closed='right', label='left').last()[['close']]
d5 = Morning.resample(rule=rule, closed='right', label='left').sum()[['volume']]
df_Morning = pd.concat([d1,d2,d3,d4,d5], axis=1)
df_Morning = df_Morning.dropna()
df_Morning.index = df_Morning.index - dt.timedelta(minutes=15)

Night = df[(df['hour'] < 8) | (df['hour'] > 13)]
d1 = Night.resample(rule=rule, closed='right', label='left').first()[['open']]
d2 = Night.resample(rule=rule, closed='right', label='left').max()[['high']]
d3 = Night.resample(rule=rule, closed='right', label='left').min()[['low']]
d4 = Night.resample(rule=rule, closed='right', label='left').last()[['close']]
d5 = Night.resample(rule=rule, closed='right', label='left').sum()[['volume']]
df_Night = pd.concat([d1,d2,d3,d4,d5], axis=1)
df_Night = df_Night.dropna()

df_Day = pd.concat([df_Morning, df_Night], axis=0) #先日再夜
df_Day = df_Day.sort_index(ascending=True) #按照時間
df_Morning['Hour'] = df_Morning.index.map(lambda x: x.hour)
trainData = df_Morning[(df_Morning.index >= '2011-01-01 00:00:00') & (df_Morning.index <= '2018-12-31 00:00:00')].co
testData = df_Morning[(df_Morning.index >= '2019-1-1 00:00:00')].copy()
settlementDate_ = pd.read_csv('settlementDate.csv') #, encoding = 'ANSI'
settlementDate_.columns = ['settlementDate', 'futures', 'settlementPrice']
bool_ = [False if 'W' in i else True for i in settlementDate_['futures']] # if第三周->true, else: false
settlementDate = [pd.to_datetime(i).date() for i in list(settlementDate_[bool_]['settlementDate'])] #擷取第三個禮拜的
```

Filter files by name

/

Name	Last Modified
01_BasicBo...	6 days ago
02_BasicKel...	a month ago
03.ipynb	6 days ago
04.ipynb	seconds ago
BTC_insma...	a month ago
data.csv	9 minutes ago
preprocessi...	4 days ago
Program.ip...	7 minutes ago
sample_bo...	6 days ago
settlement...	a month ago
stats.html	a month ago
TWF_Future...	a month ago

Simple 0 5 Python 3 (ipykernel) | Idle Mode: Command Ln 29, Col 17 04.ipynb

下午 11:24 2023/9/10

策略01：布林通道+做多

預設參數: fund = 1000000, FEE = 600(買進賣出都要花600元), LENGTH = 20, NUMSTD = 1.5, K = 0.04 ，使用60MIN線

買入條件: 若某時刻的收盤價>布林通道上軌，且不在settlementDate ，則買入台指期

賣出條件: 若某時刻的收盤價<布林通道下軌，且在settlementDate的11點後，則賣出。損益是價格*200，作為損益。

策略01：布林通道+做多

```
BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []
for i in range(len(time_arr)):
  if i == len(df_arr)-1:
    break
  entryLong = df_arr[i,3] > df_arr[i,8]
  entryCondition = date_arr[i] not in settlementDate
  exitShort = df_arr[i,3] < df_arr[i,9]
  exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11
  if BS == 'B':
    stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-Kdown) #比進場價格少了4%
    stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+Kup) #比進場價格多了4%
  if BS == None:
    profit_list.append(0)
    profit_fee_list.append(0)
    if entryLong and entryCondition:
      BS = 'B'
      t = i+1 #下一期再買
      buy.append(t) #紀錄進場時間
  elif BS == 'B':
    profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
    profit_list.append(profit)
    if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
      pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
      profit_fee = profit - FEE*2
      profit_fee_list.append(profit_fee)
      sell.append(i+1)
      BS = None
      profit_fee_realized = pl_round - FEE*2
      profit_fee_list_realized.append(profit_fee_realized)
    else:
      profit_fee = profit
      profit_fee_list.append(profit_fee);

equity = pd.DataFrame({'profit':np.cumsum(profit_list), 'profitfee':np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

策略01：布林通道+做多

策略02：肯特林通道+做多

ATR指標全名為Average True Range(真實價格區間)，指的是股價真實的波動幅度。其算法是先計算每天的TR(True Range)，再以EMA（指數移動平均）之方式計算N日TR平均值以取得ATR。

TR的算法如下：TRt = max((Ht – Lt),abs(Ct-1 – Ht), abs(Ct-1 – Lt))

以EMA計算ATR，將TR做趨勢平滑，讓指標不易有暴起暴落的雜訊。

upline = EMA20+2*ATR

downline = EMA20-2*ATR

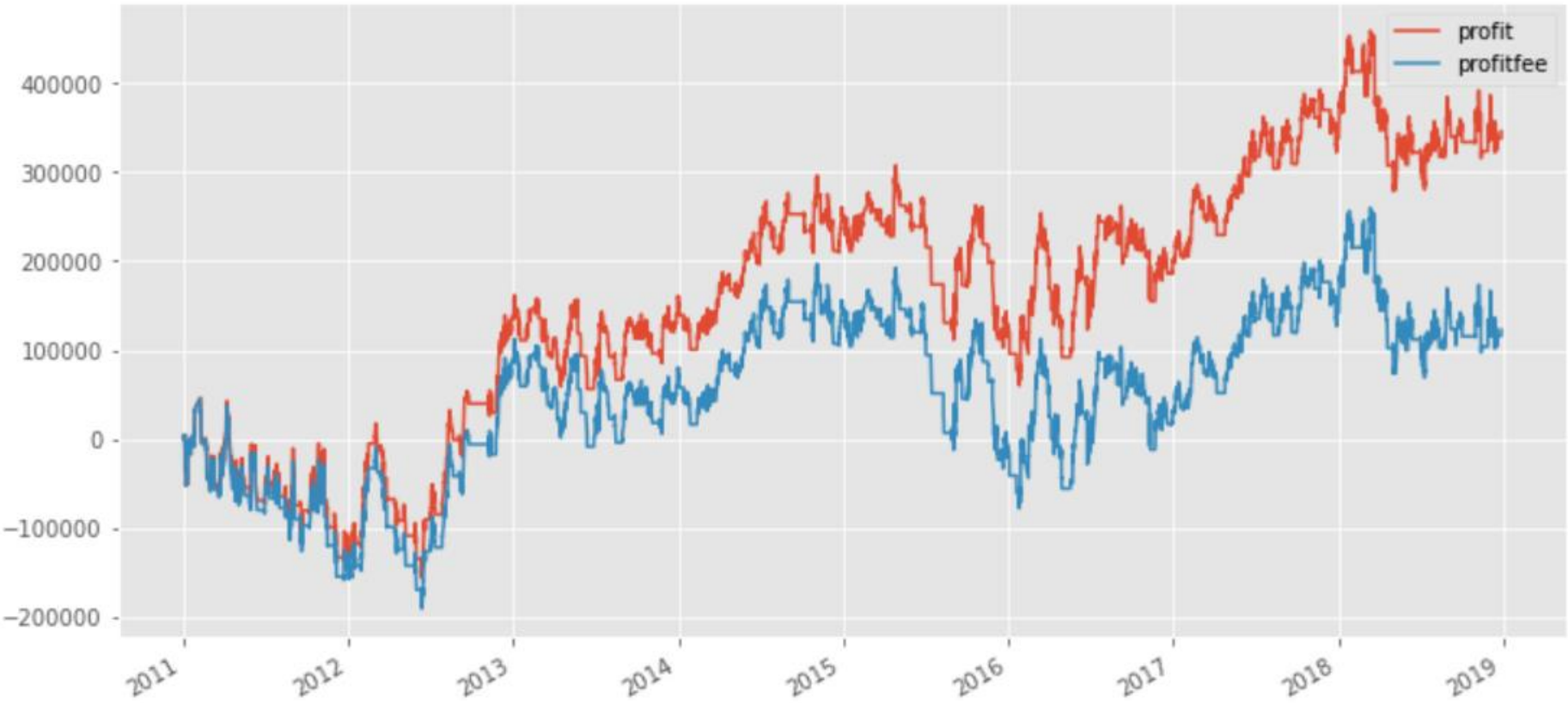
買入條件: 若某時刻的收盤價>上軌，且不在settlementDate ，則買入台指期

賣出條件: 若某時刻的收盤價<布林通道下軌，且在settlementDate的11點後，則賣出。損益是價格*200，作為損益。

策略02：肯特林通道+做多

```
BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []
for i in range(len(time_arr)):
    if i == len(df_arr)-1:
        break
    if i==0:
        lastTRt = 0
        lastEMA = 0
    max_min = df_arr[i,1] - df_arr[i,2] # 當日高低價之差額
    cl_max = abs(df_arr[i-1,3] - df_arr[i,1]) # 前日收盤與當日最高價之差額
    cl_min = abs(df_arr[i-1,3] - df_arr[i,2]) # 前日收盤與當日最低價之差額
    TRt = max(max_min, cl_max, cl_min) #有疑慮 一些人說的不一樣 - min(max_min, cl_max, cl_min)
    ATR = TRt*(2/21) + lastTRt*(19/21) #10日或20日
    EMA20 = df_arr[i,3]*(2/21) + lastEMA*(19/21)
    upline = EMA20+2*ATR
    downline = EMA20-2*ATR

    entryLong = df_arr[i,3] > upline
    entryCondition = date_arr[i] not in settlementDate
    exitShort = df_arr[i,3] < downline
    exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11
    if BS == 'B':
        stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-K) #比進場價格少了4%
        stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+K) #比進場價格多了4%

    if BS == None:
        profit_list.append(0)
        profit_fee_list.append(0)
        if entryLong and entryCondition:
            BS = 'B'
            t = i+1 #下一期再買
            buy.append(t) #紀錄進場時間

    elif BS == 'B':
        profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
        profit_list.append(profit)
        if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
            pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
            profit_fee = profit - FEE*2
            profit_fee_list.append(profit_fee)
            sell.append(i+1)
            BS = None
            profit_fee_realized = pl_round - FEE*2
            profit_fee_list_realized.append(profit_fee_realized)

        else:
            profit_fee = profit
            profit_fee_list.append(profit_fee);
    lastTRt = TRt
    lastEMA = EMA20
```

策略02：肯特林通道+做多



策略03：單純的KD

建立KD值的計算，因為就真的是KD所以不多做解釋，n=9.m=3

```
df['lowest_low'] = df['low'].rolling(n).min()
df['highest_high'] = df['high'].rolling(n).max()
df['rsv'] = (df['close'] - df['lowest_low']) / (df['highest_high'] - df['lowest_low']) *
100
df['K'] = df['rsv'].ewm(span=m).mean()
df['D'] = df['K'].ewm(span=m).mean()
```

買入條件: 若某時刻從K<D轉換成K>D，且不在settlementDate ，則買入台指期

賣出條件: 若某時刻從K>D轉換成K<D ，且在settlementDate的11點後，則賣出。

損益是價格*200，作為損益。

策略03：單純的KD

```
def calculate_kd(df, n=9, m=3):
    df['lowest_low'] = df['low'].rolling(n).min()
    df['highest_high'] = df['high'].rolling(n).max()
    df['rsv'] = (df['close'] - df['lowest_low']) / (df['highest_high'] - df['lowest_low']) * 100
    df['K'] = df['rsv'].ewm(span=m).mean()
    df['D'] = df['K'].ewm(span=m).mean()
calculate_kd(df)
df.tail(10)
df_arr = np.array(df) # 數據
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr] # datetime.date(2011, 2, 8)

BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []

for i in range(len(time_arr)):
    if i == len(df_arr)-1:
        break
    entryLong = (df_arr[i,9] >= df_arr[i,10]) and (df_arr[i+1,9] <= df_arr[i+1,10])
    entryCondition = date_arr[i] not in settlementDate
    entryLong = (df_arr[i,9] <= df_arr[i,10]) and (df_arr[i+1,9] >= df_arr[i+1,10])
    exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11
    if BS == 'B':
        stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-0.05) #5%
        stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+0.05) #5%
    if BS == None:
        profit_list.append(0)
        profit_fee_list.append(0)
        if entryLong and entryCondition:
            BS = 'B'
            t = i+1 #下一期再買
            buy.append(t) #紀錄進場時間

    elif BS == 'B':
        profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
        profit_list.append(profit)
        if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
            pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
            profit_fee = profit - FEE*2
            profit_fee_list.append(profit_fee)
            sell.append(i+1)
            BS = None
            profit_fee_realized = pl_round - FEE*2
            profit_fee_list_realized.append(profit_fee_realized)
        else:
            profit_fee = profit
            profit_fee_list.append(profit_fee);

equity = pd.DataFrame({'profit':np.cumsum(profit_list), 'profitfee':np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

策略03：單純的KD

策略04：單純的RSI

建立RSI值的計算 70以上則賣，30以下則買

```
def calculate_rsi(df, window=14):
    delta = df[ ‘close’ ].diff() # 首先，它計算了收盤價的變化（差異）
    gain = (delta.where(delta > 0, 0)).fillna(0) #將正數的變化保留，其餘的設為0
    loss = (-delta.where(delta < 0, 0)).fillna(0) #將負數的變化保留，其餘的設為0
    avg_gain = gain.rolling(window=window, min_periods=1).mean()計算平均漲幅。這裡使用移動窗口（rolling window）來計算移動平均值
    avg_loss = loss.rolling(window=window, min_periods=1).mean() 同上 計算跌幅
    rs = avg_gain / avg_loss 計算比率
    df['rsi'] = 100 - (100 / (1 + rs))
```

策略04：單純的RSI

策略04：單純的RSI

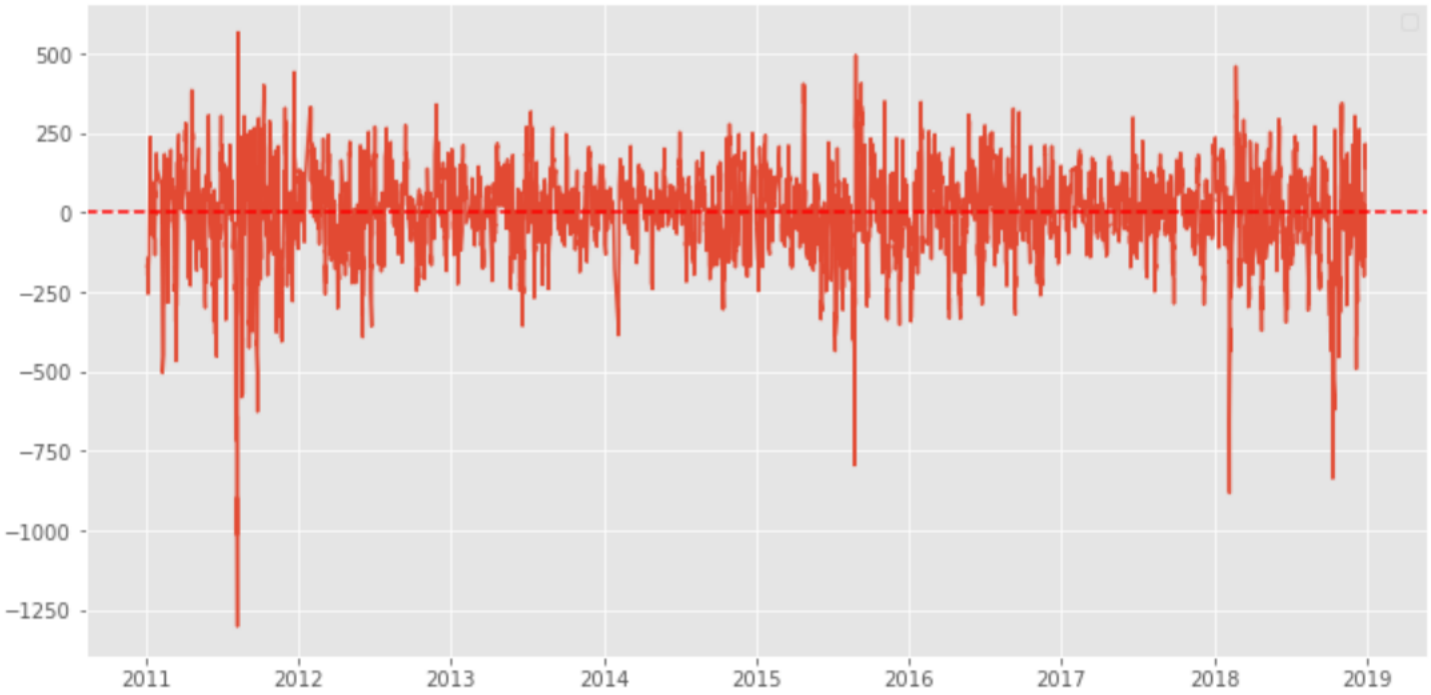
```
df_arr = np.array(df) # 數據
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr] # datetime.date(2011, 2, 8)

BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []

for i in range(len(time_arr)):
  if i == len(df_arr)-1:
    break
  entryLong = (df_arr[i,6] >= 30)
  entryCondition = date_arr[i] not in settlementDate
  exitLong = (df_arr[i,6] <= 70)
  exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11

  if BS == 'B':
    stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-0.05) #5%
    stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+0.05) #5%
  if BS == None:
    profit_list.append(0)
    profit_fee_list.append(0)
    if entryLong and entryCondition:
      BS = 'B'
      t = i+1 #下一期再買
      buy.append(t) #紀錄進場時間
Elif BS== 'B':
    profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
    profit_list.append(profit)
    if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
      pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
      profit_fee = profit - FEE*2
      profit_fee_list.append(profit_fee)
      sell.append(i+1)
      BS = None
      profit_fee_realized = pl_round - FEE*2
      profit_fee_list_realized.append(profit_fee_realized)
    else:
      profit_fee = profit
      profit_fee_list.append(profit_fee);

equity = pd.DataFrame({'profit':np.cumsum(profit_list), 'profitfee':np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

策略04：單純的RSI

策略05：Momentum動能策略

df['momentum'] = df['close'].diff(window)

Momemtum是動量，假設window=14，如果diff大於0則買入，失去動量則賣出。



策略05：Momentum動能策略

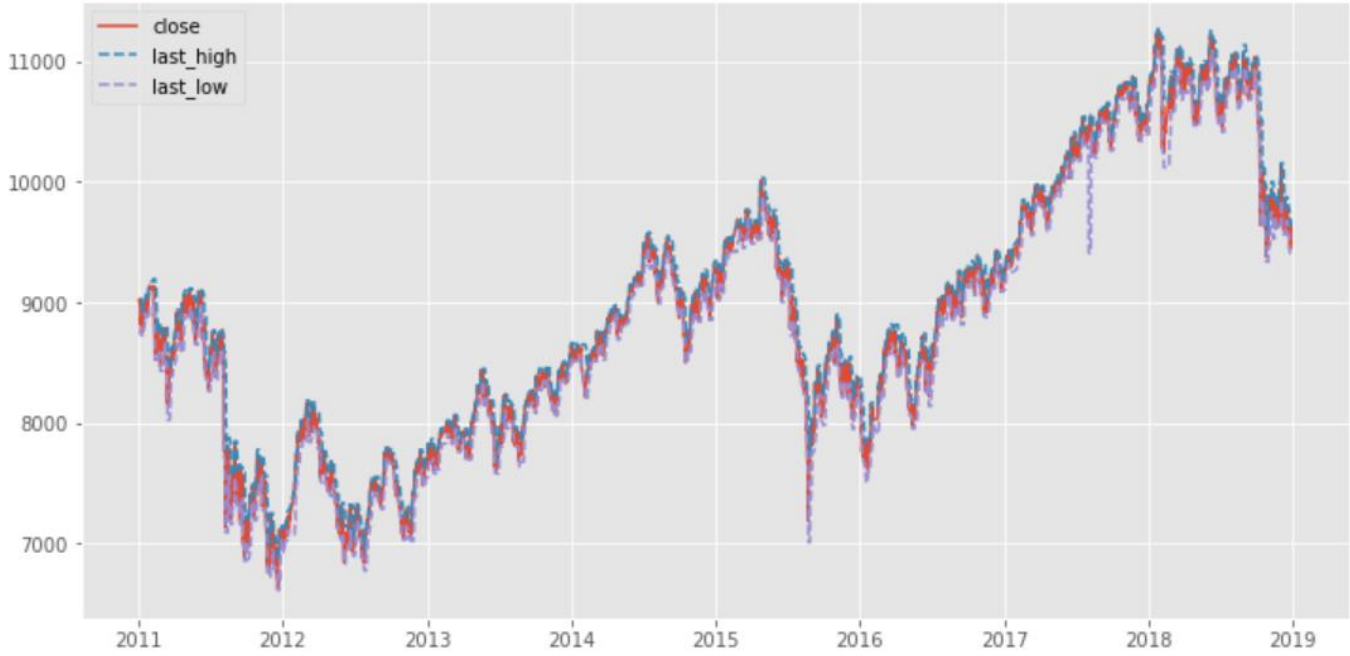
```
df_arr = np.array(df) # 數據
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr] # datetime.date(2011, 2, 8)
BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []
for i in range(len(time_arr)):
    if i == len(df_arr)-1:
        break
    entryLong = df_arr[i,6] > 0
    entryCondition = date_arr[i] not in settlementDate
    exitShort = df_arr[i,3] < 0
    exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11
    if BS == 'B':
        stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-0.05) #比進場價格少了4%
        stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+0.05) #比進場價格多了4%
    if BS == None:
        profit_list.append(0)
        profit_fee_list.append(0)
        if entryLong and entryCondition:
            BS = 'B'
            t = i+1 #下一期再買
            buy.append(t) #紀錄進場時間
    elif BS == 'B':
        profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
        profit_list.append(profit)
        if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
            pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
            profit_fee = profit - FEE*2
            profit_fee_list.append(profit_fee)
            sell.append(i+1)
            BS = None
            profit_fee_realized = pl_round - FEE*2
            profit_fee_list_realized.append(profit_fee_realized)
        else:
            profit_fee = profit
            profit_fee_list.append(profit_fee);

equity = pd.DataFrame({'profit':np.cumsum(profit_list), 'profitfee':np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

策略05：Momentum動能策略

策略06：Breakout Strategy (突破策略)

我們基於價格突破的前一高點或低點來生成買入和賣出信號。因此識別高點和低點，並根據突破來生成交易信號。

```
def calculate_breakout_levels(df, window=20):
    df['prev_high'] = df['high'].rolling(window=window).max()
    df['prev_low'] = df['low'].rolling(window=window).min()
```



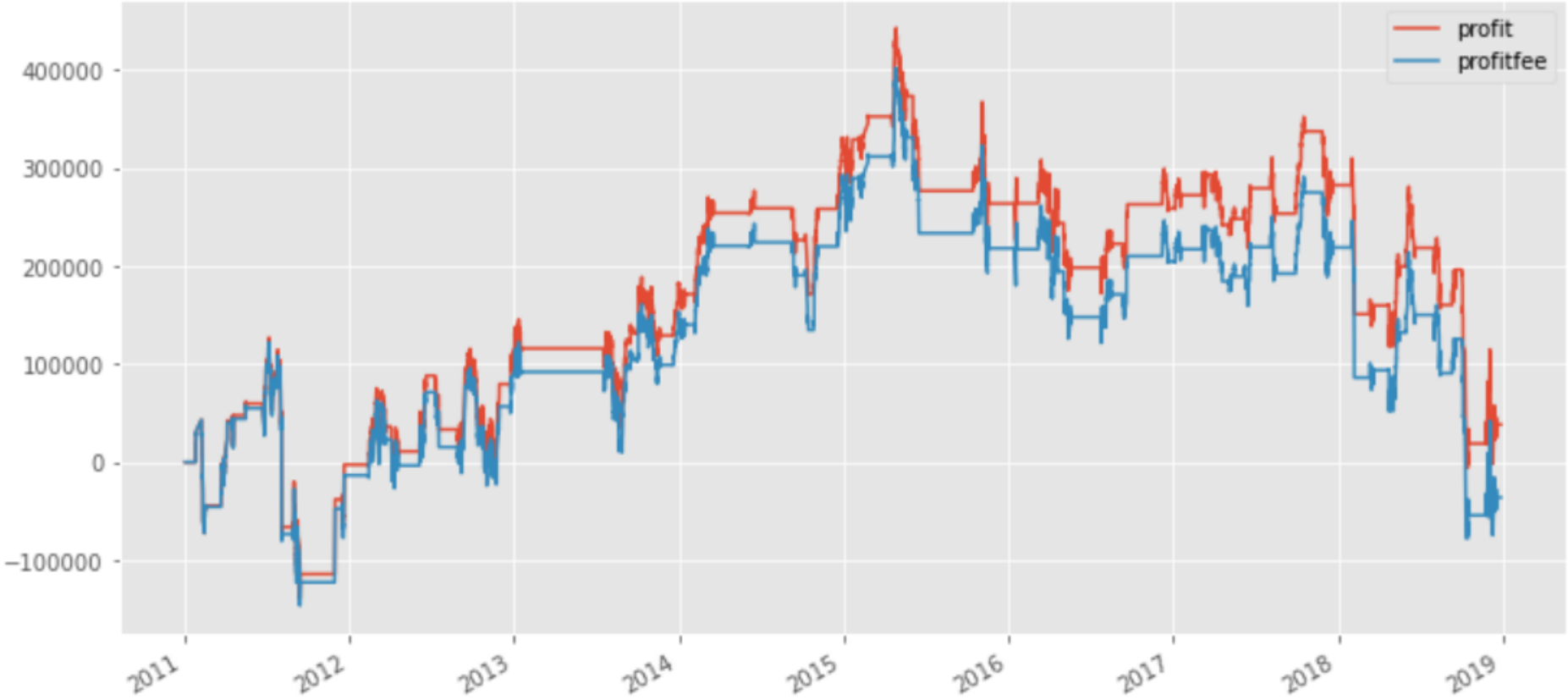
策略06：Breakout Strategy (突破策略)

```
df_arr = np.array(df) # 數據
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr] # datetime.date(2011, 2, 8)

BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []

for i in range(len(time_arr)):
  if i == len(df_arr)-1:
    break
  entryLong = (df_arr[i,3] >= df_arr[i,6])
  entryCondition = date_arr[i] not in settlementDate
  entryLong = (df_arr[i,3] <= df_arr[i,7])
  exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11

  if BS == 'B':
    stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-0.05) #5%
    stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+0.05) #5%
  if BS == None:
    profit_list.append(0)
    profit_fee_list.append(0)
    if entryLong and entryCondition:
      BS = 'B'
      t = i+1 #下一期再買
      buy.append(t) #紀錄進場時間
elif BS == 'B':
    profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
    profit_list.append(profit)
    if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
      pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
      profit_fee = profit - FEE*2
      profit_fee_list.append(profit_fee)
      sell.append(i+1)
      BS = None
      profit_fee_realized = pl_round - FEE*2
      profit_fee_list_realized.append(profit_fee_realized)
    else:
      profit_fee = profit
      profit_fee_list.append(profit_fee);

equity = pd.DataFrame({'profit':np.cumsum(profit_list), 'profitfee':np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

策略06：Breakout Strategy (突破策略)

✓ ☹☹

✓ ☹



策略07：MACD策略

MACD（Moving Average Convergence Divergence）指標是一個常用於技術分析的指標
可以幫助交易者識別資產的價格趨勢、趨勢的變化以及潛在的買入或賣出時機。
MACD指標主要由以下兩個部分組成：
快線（Short EMA）：通常以12個時期的指數加權移動平均來表示。
慢線（Long EMA）：通常以26個時期的指數加權移動平均來表示。

def calculate_macd(df, short_window=12, long_window=26, signal_window=9) #短期、長期、信號線
df['ShortEMA'] = df['close'].ewm(span=short_window, min_periods=1).mean()
來計算收盤價的短期指數加權移動平均值，**span** 參數指定了時間窗口的大小。
df['LongEMA'] = df['close'].ewm(span=long_window, min_periods=1).mean()
這行程式碼計算了收盤價的長期指數加權移動平均值。
df['MACD'] = df['ShortEMA'] - df['LongEMA']
這行計算了MACD指標，即短期EMA減去長期EMA的差異。
df['SignalLine'] = df['MACD'].ewm(span=signal_window, min_periods=1).mean()
這行程式碼計算了MACD的信號線，也是使用指數加權移動平均的方法。

當MACD>Signal_Line則買入，相反則賣出。

策略07：MACD策略

✓ **[‘SignalLine’]的線圖**

策略07：MACD策略

```
df_arr = np.array(df) # 數據
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr] # datetime.date(2011, 2, 8)
BS = None
buy = []
sell = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []
for i in range(len(time_arr)):
    if i == len(df_arr)-1:
        break
    entryLong = df_arr[i,8] > df_arr[i,9]
    entryCondition = date_arr[i] not in settlementDate
    exitShort = df_arr[i,8] < df_arr[i,9]
    exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11
    if BS == 'B':
        stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-Kdown) #比進場價格少了4%
        stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+Kup) #比進場價格多了4%
    if BS == None:
        profit_list.append(0)
        profit_fee_list.append(0)
        if entryLong and entryCondition:
            BS = 'B'
            t = i+1 #下一期再買
            buy.append(t) #紀錄進場時間
    elif BS == 'B':
        profit = 200 * (df_arr[i+1,0] - df_arr[i,0]) #單一期的損益
        profit_list.append(profit)
        if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
            pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0]) #下一期的一開始才結束交易，計算總損益
            profit_fee = profit - FEE*2
            profit_fee_list.append(profit_fee)
            sell.append(i+1)
            BS = None
            profit_fee_realized = pl_round - FEE*2
            profit_fee_list_realized.append(profit_fee_realized)
        else:
            profit_fee = profit
            profit_fee_list.append(profit_fee);

equity = pd.DataFrame({'profit':np.cumsum(profit_list), 'profitfee':np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

策略07：MACD策略

MACD策略深入研究

- 希望可以找出更好的return

MACD策略深入研究

✓ MACD的中文是「平滑異同移動平均線」，研究長周期均線和短周期均線的平均值來體現市場的動能。

✓ 如果交易的周期設置越小，其發出的信號越多，但準確性更低；周期設置得越大，發出的信號便會越少，但準確性更高。短期抓太多信號可能會抓到雜訊，而抓太少的話也有可能造成沒有抓到該抓的東西，算是一種tradeoff。 => 抓到最適合的參數是重點!(同時不能夠overfitting)

- 信號線是DIF快線差離值的指數移動平均：

  當其數值為正數時，表示短期平均價格大於長期平均價格，也就意味著市場行情處於上升中；

  當信號線值為負數時，表示短期平均價格低於長期平均價格，意味著市場行情處於下跌中。

✓ 因此，在MACD策略中，我們也可以適當選擇做空、多空多做，以及加碼等作為，來更好的捕捉股市型態。

✓ 原始利潤的數值：120% vs 80% (profit vs profitfee)

MACD策略深入研究

- ✓ 可能性1：參數最佳化：透過機器學習的嘗試讓參數能夠有最好的效果:
- ✓ 優點：看起來曲線更好看
- ✓ 缺點：可能會有overfitting的問題

這題比較不適合用機器學習的形式去探索最佳參數(因為這一點都不ML)，我們利用隨機搜索型態來找到最佳參數。

- ✓ 最佳參數: (10,53,17)、但是效果差不太多(增加10%績效)。

MACD策略深入研究

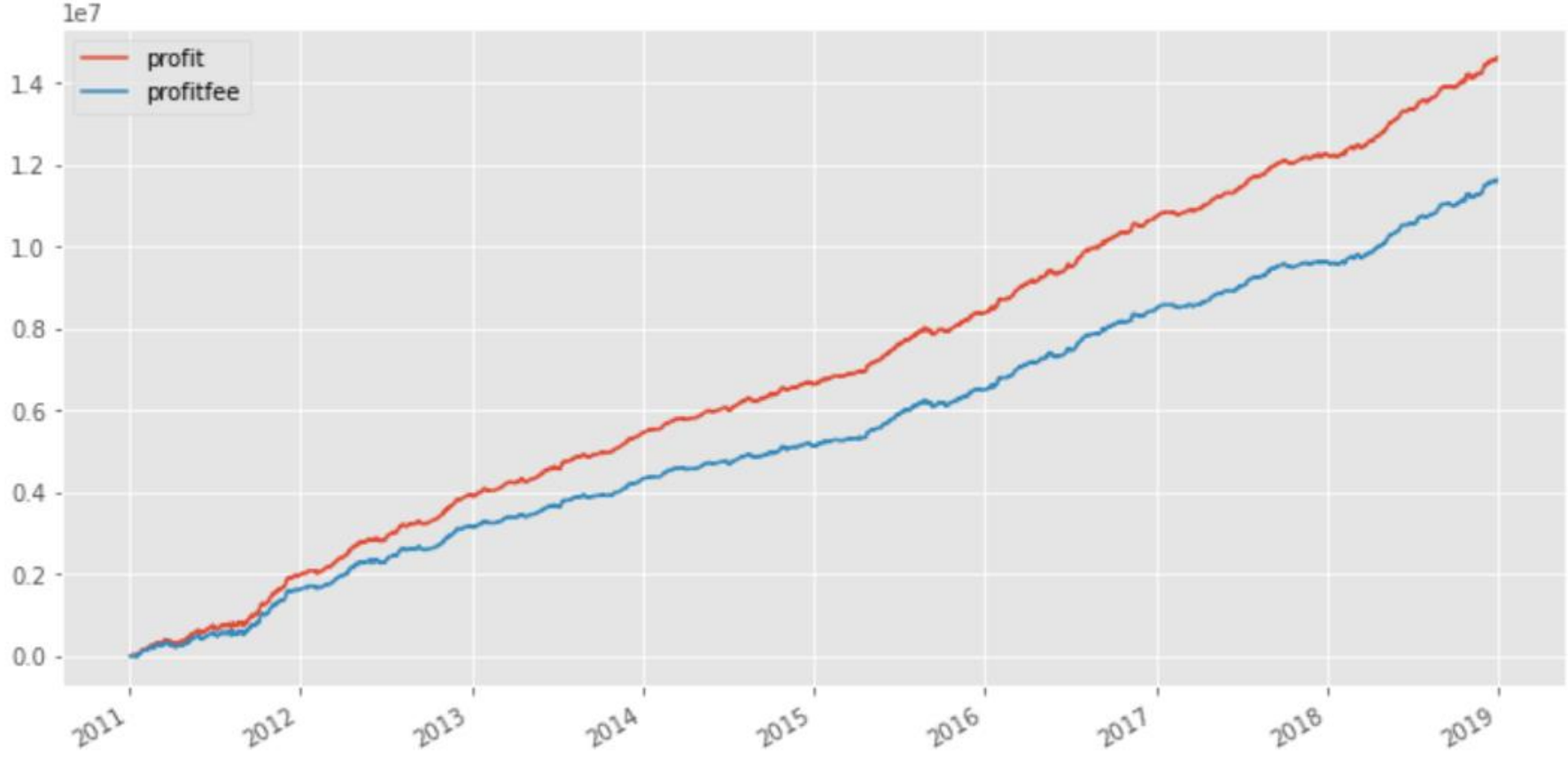
- ✓ 可能性2：多空都做
- ✓ 優點：把握空頭時間的波段、創造利潤
- ✓ 缺點：手續費用也由此增高

我參考了在7/31暑期社課中所提到的策略優化形式，同時增加了做多和做空的條件，判斷基礎基本相同，最後報酬率從80%升高到120%，代表此策略有顯著的成功。

MACD策略深入研究

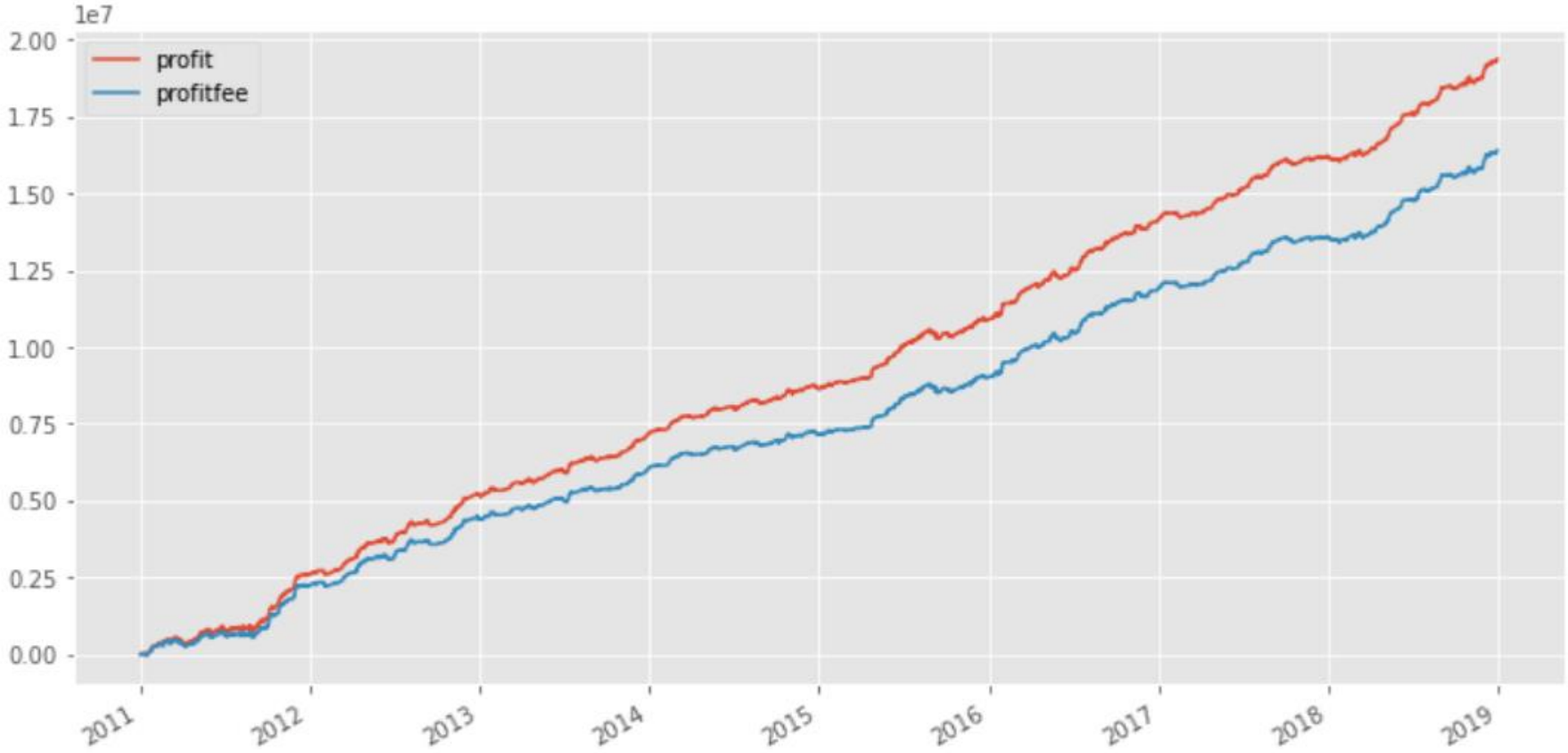
```
df_arr = np.array(df)
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr]
FEE = 600
BS = None
Buy, sell, profit_list, profit_fee_list, profit_fee_list_realized = [],[],[0],[0],[]
for i in range(len(time_arr)):
  if i == len(df_arr)-1:
    break
  entryLong = df_arr[i,8] > df_arr[i,9]
  entryShort = df_arr[i,8] < df_arr[i,9]
  entryCondition = date_arr[i] not in settlementDate
  exitShort = df_arr[i,8] < df_arr[i,9]
  exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11
  if BS == 'B':
    stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-0.05)
    stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+0.05)
  elif BS == 'S':
    stopLoss = df_arr[i,2] >= df_arr[t,0] * (1+0.05)
    stopProfit = df_arr[i,2] <= df_arr[t,0] * (1-0.05)
  if BS == None:
    profit_list.append(0)
    profit_fee_list.append(0)
    if entryLong and entryCondition:
      BS = 'B'
      t = i+1
      buy.append(t)
    elif entryShort and entryCondition:
      BS = 'S'
      t = i+1
      short.append(t)
  elif BS == 'B':
    profit = -200 * (df_arr[i+1,0] - df_arr[i,0])
    profit_list.append(profit)
    if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
      pl_round = 200 * (df_arr[i+1,0] - df_arr[t,0])
      profit_fee = profit - FEE*2
      profit_fee_list.append(profit_fee)
      sell.append(i+1)
      BS = None
      profit_fee_realized = pl_round - FEE*2
      profit_fee_list_realized.append(profit_fee_realized)
    else:
      profit_fee = profit
      profit_fee_list.append(profit_fee);
  elif BS == 'S':
    profit = -200 * (df_arr[i,2] - df_arr[i+1,0])
    profit_list.append(profit)
    if entryShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
      pl_round = 200 * (df_arr[t,0] - df_arr[i+1,0])
      profit_fee = profit - FEE*2
      profit_fee_list.append(profit_fee)
      buy.append(i+1)
      BS = None
      profit_fee_realized = pl_round - FEE*2
      profit_fee_list_realized.append(profit_fee_realized)
    else:
      profit_fee = profit
      profit_fee_list.append(profit_fee);
equity = pd.DataFrame({'profit': np.cumsum(profit_list), 'profitfee': np.cumsum(profit_fee_list)},
index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

MACD策略深入研究：投資績效



MACD策略深入研究

- ✓ 可能性2：多空都做+逢連續則加碼
- ✓ 策略：如果時間連續三天的指標都維持相同，則加倍進駐。(position:200 => 400)



最終版本code1

```
df_arr = np.array(df)
time_arr = np.array(df.index)
date_arr = [pd.to_datetime(i).date() for i in time_arr]
FEE = 600
BS = None
buy = []
sell = []
short = []
profit_list = [0]
profit_fee_list = [0]
profit_fee_list_realized = []
position = 200
count_B = 0
count_S = 0
for i in range(len(time_arr)):
    if i == len(df_arr)-1:
        break
    entryLong = df_arr[i,8] > df_arr[i,9]
    entryShort = df_arr[i,8] < df_arr[i,9]
    entryCondition = date_arr[i] not in settlementDate
    exitShort = df_arr[i,8] < df_arr[i,9]
    exitCondition = date_arr[i] in settlementDate and df_arr[i,5] >= 11

    if BS == 'B':
        stopLoss = df_arr[i,3] <= df_arr[t,0] * (1-0.05)
        stopProfit = df_arr[i,3] >= df_arr[t,0] * (1+0.05)
    elif BS == 'S':
        stopLoss = df_arr[i,2] >= df_arr[t,0] * (1+0.05)
        stopProfit = df_arr[i,2] <= df_arr[t,0] * (1-0.05)
    if BS == None:
        position = 200
        count_B = 0
        count_S = 0
        profit_list.append(0)
        profit_fee_list.append(0)
        if entryLong and entryCondition:
            BS = 'B'
            t = i+1
            buy.append(t)
        elif entryShort and entryCondition:
            count_B = 0
            BS = 'S'
            t = i+1
            short.append(t)
    elif BS == 'B':
        count_B += 1
        profit = -position * (df_arr[i+1,0] - df_arr[i,0])
        profit_list.append(profit)
        if exitShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
            count_B = 0
            pl_round = position * (df_arr[i+1,0] - df_arr[t,0])
            profit_fee = profit - FEE*2
            profit_fee_list.append(profit_fee)
            sell.append(i+1)
            BS = None
            profit_fee_realized = pl_round - FEE*2
            profit_fee_list_realized.append(profit_fee_realized)
            position =200
```

最終版本code2

```
else:
        profit_fee = profit
        profit_fee_list.append(profit_fee)
        if count_B ==3:
            position *= 2


  elif BS == 'S':
      count_S += 1
      profit = -position * (df_arr[i,2] - df_arr[i+1,0])
      profit_list.append(profit)
      if entryShort or exitCondition or stopLoss or stopProfit or i == len(df_arr)-2 :
          pl_round = position * (df_arr[t,0] - df_arr[i+1,0])
          profit_fee = profit - FEE*2
          profit_fee_list.append(profit_fee)
          buy.append(i+1)
          BS = None
          profit_fee_realized = pl_round - FEE*2
          profit_fee_list_realized.append(profit_fee_realized)
      else:
          profit_fee = profit
          profit_fee_list.append(profit_fee)
          if count_S == 3:
              position *= 2

equity = pd.DataFrame({'profit': np.cumsum(profit_list), 'profitfee': np.cumsum(profit_fee_list)}, index=trainData.index)
equity.plot(grid=True, figsize=(12,6));
```

回測績效：參考sample code使用的request形式

Profit : 1640200
Return : -673.2131147540983
Max DrawDown : 0.5
Caimar Ratio : -1346.4262295081967
Trade Times : 2646
Win Rate : 0.38492543329302703
Profit Factor : 0.8260905067528946

老實說我感覺得出來哪裡出錯了，但是圖表很漂亮，所以我就先擺者。

e

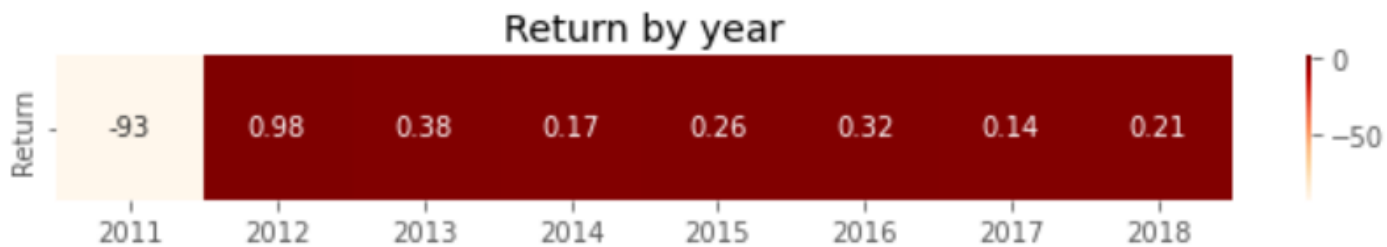
回測績效code

```
import seaborn as sns

import matplotlib.pyplot as py

plt.style.use('ggplot')

print(len(buy))

equity = equity[50:]

equity['equity'] = equity['profitfee']

equity['drawdown_percent'] = (equity['equity']/equity['equity'].cummax()) - 1

equity['drawdown'] = equity['equity'] - equity['equity'].cummax()

profit = equity['profitfee'].iloc[-1]

ret = equity['equity'][-1]/equity['equity'][0] - 1

mdd = abs(equity['drawdown_percent'].min())

calmarRatio = ret / mdd

tradeTimes = len(buy) + len(sell)

winRate = len([i for i in profit_fee_list_realized if i > 0]) / len(profit_fee_list_realized)

profitFactor = sum([i for i in profit_fee_list_realized if i>0]) / abs(sum([i for i in profit_fee_list_realized if i<0]))
```

```
print('Profit : ',profit)
print('Return : ',ret)
print('Max DrawDown : ',mdd)
print('Caimar Ratio : ',calmarRatio)
print('Trade Times : ',tradeTimes)
print('Win Rate : ',winRate)
print('Profit Factor : ',profitFactor)
```

回測績效：時間損益(年)

```
# 時間損益(年)
equity.index = pd.to_datetime(equity.index) #確保索引是datetime型態
years = ['2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018']
year_ret = []
for i in years:
    year_ret.append(equity[i]['equity'].iloc[-1]/equity[i]['equity'].iloc[0] - 1)

df = pd.DataFrame({'Return':year_ret},index = years)

# heatmap函式
py.figure(figsize=(10,1))
sns.heatmap(df.transpose(), annot=True, cmap='OrRd')
py.title('Return by year')
py.show()
print('')
```



回測績效：2012年到2018年(刪掉前後兩年看中間的平均績效)

- ✓ https://drive.google.com/file/d/1kn4zK-tvpY8tmoYf7zMHF-W6c57YucsH/view?usp=sharing
- ✓ (HTML檔案)

感謝聆聽，敬請指教