

第二屆「生成式 AI 創新應用競賽」計畫書

一、問題說明

財金系在實務上常需要大量數據分析和回測，但由於某些條件限制，回測在 Excel 上比較難處理，而且 Excel 數據在 100 萬筆以上時容易不穩定。因此，在進行量化分析時，經常需要借助程式來輔助處理。然而，對於非資工科系的學生來說，學習程式的成本較高。而生成式 AI 技術，如 ChatGPT，則非常適合自動生成程式碼，因此我們希望利用財金系的邏輯，結合 AI 來生成適合的程式。

具體目標

使用 AI 寫出一個 Python 程式來回測一個股票策略，對比其短期、中期和長期策略，並進行簡單的分析和圖表化。

使用工具

- ChatGPT
- Bing

二、設計流程

擬定目標

1. 擬定基本投資策略

- 當 MACD 為正且 RSI 連續 2 次比前一次強時買入 1 單位。
- 當 MACD 為負且 RSI 連續 2 次比前一次強時賣出 1 單位。
- 在任何時間內，最高持有一單位股票，並進行基本紀錄，包含勝率、交易次數、總獲益。

2. 時間設定

- 分別設定分鐘、日線、周線、月線交易及 Buy and Hold 交易策略。

3. 指定股票及交易時間

- 選定特定的股票和交易時間段進行回測。

4. 優化並視覺化

- 使用 AI 幫忙設計視覺化的圖表，展示基本量化資料和解釋程式碼的運作方式。

5. 具體結果-主程式

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
import matplotlib.font_manager as fm

# 設置中文字體
```

```

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 更換為您的系統字體
plt.rcParams['axes.unicode_minus'] = False # 確保減號正常顯示

def macd_strategy(data):
    data['EMA_12'] = data['Close'].ewm(span=12, adjust=False).mean()
    data['EMA_26'] = data['Close'].ewm(span=26, adjust=False).mean()
    data['MACD'] = data['EMA_12'] - data['EMA_26']
    data['Signal_Line'] = data['MACD'].ewm(span=9, adjust=False).mean()

    delta = data['Close'].diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
    rs = gain / loss
    data['RSI'] = 100 - (100 / (1 + rs))

    return data

def generate_signals(data):
    data = macd_strategy(data)
    data['Signal'] = 0
    data['Signal'][((data['MACD'] > data['Signal_Line']) &
                     (data['RSI'].diff(1) > 0) &
                     (data['RSI'].diff(2) > 0))] = 1
    data['Signal'][((data['MACD'] < data['Signal_Line']) &
                     (data['RSI'].diff(1) > 0) &
                     (data['RSI'].diff(2) > 0))] = -1
    data['Position'] = data['Signal'].replace(to_replace=0,
method='ffill').shift()
    return data

def resample_data(data, frequency):
    resampled_data = data.resample(frequency).agg({
        'Open': 'first',
        'High': 'max',
        'Low': 'min',
        'Close': 'last',
        'Volume': 'sum'
    })

```

```

    }).dropna()
    return resampled_data

def backtest(data):
    data['Market_Return'] = data['Close'].pct_change()
    data['Strategy_Return'] = data['Market_Return'] * data['Position']
    data['Cumulative_Market_Return'] = (1 +
data['Market_Return']).cumprod()
    data['Cumulative_Strategy_Return'] = (1 +
data['Strategy_Return']).cumprod()

    return data

def analyze_performance(data, period):
    total_trades = data['Signal'].abs().sum()
    winning_trades = data[data['Strategy_Return'] >
0]['Strategy_Return'].count()
    win_rate = winning_trades / total_trades
    total_return = data['Cumulative_Strategy_Return'].iloc[-1]

    print(f"{period} - 總交易次數: {total_trades}")
    print(f"{period} - 勝率: {win_rate:.2%}")
    print(f"{period} - 總獲益: {total_return:.2f}")
    print(f"{period} - 詳細交易資料:")
    print(data.loc[data['Signal'] != 0, ['Signal', 'Close']])

def visualize(data, period):
    plt.figure(figsize=(14,7))
    plt.plot(data['Cumulative_Market_Return'], label='市場累積回報')
    plt.plot(data['Cumulative_Strategy_Return'], label='策略累積回報')

    # 添加交易點
    buy_signals = data[data['Signal'] == 1]
    sell_signals = data[data['Signal'] == -1]
    plt.scatter(buy_signals.index,
data.loc[buy_signals.index]['Cumulative_Strategy_Return'], marker='^',
color='g', label='買入點')

```

```

plt.scatter(sell_signals.index,
data.loc[sell_signals.index]['Cumulative_Strategy_Return'], marker='v',
color='r', label='賣出點')

plt.legend()
plt.title(f'{period} 股票策略回測結果')
plt.xlabel('日期')
plt.ylabel('累積回報')
plt.show()

def get_stock_data(ticker, start, end):
    stock_data = yf.download(ticker, start=start, end=end)
    return stock_data

if __name__ == "__main__":
    # 輸入股票代號和交易期間
    ticker = input("請輸入股票代號: ")
    start_date = input("請輸入開始日期 (格式: YYYY-MM-DD): ")
    end_date = input("請輸入結束日期 (格式: YYYY-MM-DD): ")

    data = get_stock_data(ticker, start_date, end_date)

    for period, freq in [('日線', 'D'), ('周線', 'W'), ('月線', 'M')]:
        resampled_data = resample_data(data, freq)
        resampled_data = generate_signals(resampled_data)
        resampled_data = backtest(resampled_data)
        analyze_performance(resampled_data, period)
        visualize(resampled_data, period)

```

6. 具體結果-執行結果（以 1101，執行時間 2022 至今）

日線 - 總交易次數: 397
 日線 - 勝率: 121.41%
 日線 - 總獲益: 0.87
 日線 - 詳細交易資料:

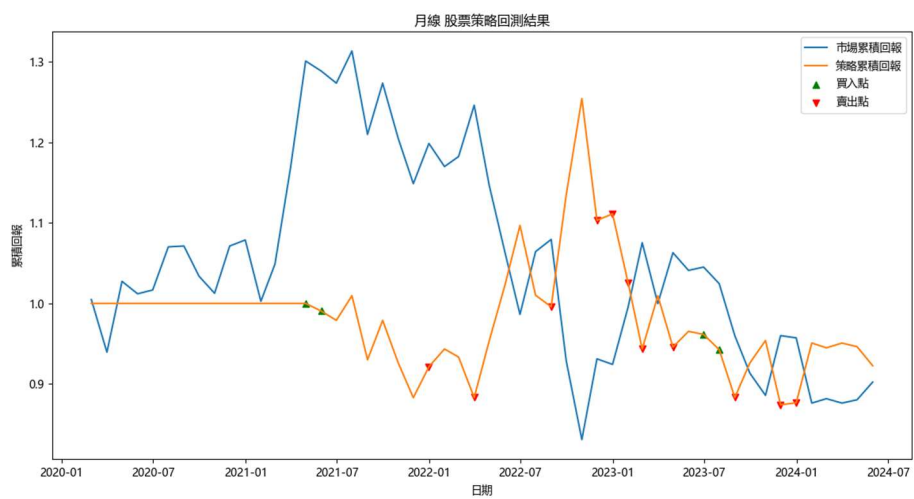
Date	Signal	Close
2020-02-04	-1	36.711792
2020-02-06	-1	37.231297
2020-02-13	1	37.361176
2020-02-19	1	37.577637
2020-02-21	1	37.447758
...
2024-05-10	1	33.299999
2024-05-15	1	33.299999
2024-05-16	1	33.599998
2024-05-17	1	33.900002
2024-05-22	1	34.349998

周線 - 總交易次數: 71
 周線 - 勝率: 133.80%
 周線 - 總獲益: 0.58
 周線 - 詳細交易資料:

Date	Signal	Close
2020-04-19	-1	36.365456
2020-05-10	1	37.231297
2020-06-07	1	37.794098
2020-06-21	1	36.841667
2020-06-28	1	37.101421
...
2024-04-28	1	31.799999
2024-05-05	1	32.400002
2024-05-12	1	33.299999
2024-05-19	1	33.900002
2024-05-26	1	34.049999

月線 - 總交易次數: 15
 月線 - 勝率: 113.33%
 月線 - 總獲益: 0.92
 月線 - 詳細交易資料:

Date	Signal	Close
2021-04-30	1	47.366726
2021-05-31	1	46.912151
2021-12-31	-1	43.639210
2022-03-31	-1	45.366596
2022-08-31	-1	39.299999
2022-11-30	-1	33.900002
2022-12-31	-1	33.650002
2023-01-31	-1	36.250000
2023-02-28	-1	39.150002
2023-04-30	-1	38.700001
2023-06-30	1	38.049999
2023-07-31	1	37.299999
2023-08-31	-1	34.950001
2023-11-30	-1	34.950001
2023-12-31	-1	34.849998



三、說明運用生成式 AI 協助解決問題的過程

使用程式：ChapGPT(GPT 3.5)和 Bing(GPT 4)

具體流程：在上面設定好目標後，分析目標給予以下指令

指令：

可以幫我依上面程式，補足成為一個新程式，他需要具備以下條件

1. 使用 `yfinance`，`pandas`，`numpy`，`backtrader` 包，如有需要也可以使用其他包
2. 使用 `backtrader` 框架
3. 輸入開始日期和結束日期
4. 輸入股票代號
5. 第一個子程式
6. 當此股票在時間範圍內，日線發生 `macd` 為正且 `rsi` 連續 2 次比前一次強時買入 1 單位，並且在 `macd` 為負且 `rsi` 連續 2 次比前一次強時賣出 1 單位
7. 同時間內，最高至持有一單位
8. 計算期間總獲利，並輸出
9. “日線交易”
10. “總獲利:” 期間總獲利
11. “總交易次數:” 期間總交易次數
12. “勝率:” 獲利為正的次數/總交易次數
13. 第二個子程式
14. 當此股票在時間範圍內，周線發生 `macd` 為正且 `rsi` 連續 2 次比前一次強時買入 1 單位，並且在 `macd` 為負且 `rsi` 連續 2 次比前一次強時賣出 1 單位
15. 同時間內，最高至持有一單位
16. 計算期間總獲利，並輸出
17. “周線交易”
18. “總獲利:” 期間總獲利
19. “總交易次數:” 期間總交易次數
20. “勝率:” 獲利為正的次數/總交易次數
21. 第三個子程式
22. 當此股票在時間範圍內，月線 `macd` 為正且 `rsi` 連續 2 次比前一次強時買入 1 單位，並且在 `macd` 為負且 `rsi` 連續 2 次比前一次強時賣出 1 單位

23. 同一時間內，最高至持有一單位
24. 計算期間總獲利，並輸出
25. “月線交易”
26. “總獲利:” 期間總獲利
27. “總交易次數:” 期間總交易次數
28. “勝率:” 獲利為正的次數/總交易次數

29. 第四個子程式
30. 開始日開盤買入，終止日收盤賣出
31. 計算期間總獲利，並輸出
32. “Buy and Hold 交易”
33. “總獲利:” 期間總獲利
34. “總交易次數:” 期間總交易次數
35. “勝率:” 獲利為正的次數/總交易次數

得到初步結果(擷取部分資訊)



要實現這個程式，我們需要使用Python並搭配一些金融數據分析的工具。例如，`pandas`用於數據處理，`yfinance`用於獲取股票數據，`talib`用於計算技術指標（如MACD和RSI）。

首先，請確保你已經安裝了所需的Python包：

```
sh  
  
pip install yfinance pandas ta-lib
```

複製程式碼


接下來，我們可以開始編寫程式。以下是程式的完整實現：

```
python  
  
import yfinance as yf  
import pandas as pd  
import talib as ta  
  
def fetch_data(ticker, start, end, interval):  
    data = yf.download(ticker, start=start, end=end, interval=interval)  
    return data
```


複製程式碼

然後反覆修正異常和調整顯示資料(資料過多，以兩張為例)

修正包異常

 您

No module named 'talib'

 Copilot

看來您在安裝 TA-Lib 時遇到了一些問題。TA-Lib 是一個 C 語言編寫的技术分析庫，因此在 Python 中使用它之前，您需要先安裝 C 語言庫。以下是在不同操作系統中安裝 TA-Lib 的一些指導：

Windows:

1. 下載 `ta-lib-0.4.0-msvc.zip` 並解壓縮到 `C:\ta-lib`。
2. 如果您使用的是 64 位 Python，您需要建立一個 64 位版本的庫。
3. 下載並安裝 Visual Studio 社區版（2015 或更高版本），記得選擇 [Visual C++] 特性。
4. 從 Windows 開始菜單啟動 [VS2015 x64 原生工具命令提示符]。

調整圖片文字異常

 顯示不出來中文，中文部分是方塊

 中文顯示為方塊通常是因為缺少相應的中文字體。以下是改進後的版本，確保你有合適的中文字體，並且正確設置路徑。

假設你已經下載了 SimHei 字體並放置在程式的當前目錄中。這裡是一個完整的 Python 程式，展示如何使用 Matplotlib 和 PIL 庫來生成並顯示包含中文的圖像。

首先，確保你的系統已經安裝了 Matplotlib 和 Pillow 庫：

```
bash
```

```
pip install matplotlib pillow
```

複製程式碼

進行優化(以視覺化為例)

可以在圖上添加交易點嗎



當然可以，在可視化結果上添加交易點以標示每次交易的位置。我們可以在策略產生交易信號時將相應的日期和價格記錄下來，然後在圖上標示出來。

下面是更新後的程式碼，包含在圖上添加交易點的部分：

python

複製程式碼

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
import matplotlib.font_manager as fm

# 設置中文字體
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 更換為您的系統字體
plt.rcParams['axes.unicode_minus'] = False # 確保減號正常顯示
```