人工智慧專題-
基於Transformer模型的市場價格預測
:應用於加密貨幣、S&P 500和台股指數的分析

組員: a1115513 劉沛辰、a1115530 劉柏鈞
a1115531 錢昱名、a1115534 林吟蓁、a1115538 賴芋安

1.專題背景與目標

在金融市場中，準確預測市場價格變動具有重要意義，能幫助投資者制定有效的投資策略。本專題的目標是開發一個基於Transformer深度學習模型的時間序列預測系統，用於預測不同市場（如加密貨幣、S&P 500指數和台股加權指數）的價格走勢。我們選擇Transformer模型，是因為它在處理長期依賴的序列數據方面表現出色，能夠更好地捕捉價格序列中的模式與趨勢。

2.資料收集、預處理與分析

- 數據收集**:**

  本專題使用三種市場資料進行分析，且均從trading view網站計算指標與數據

  - ❖ 加密貨幣**BTC/USD**：來自於Bitstamp交易所的比特幣價格數據。
  - ❖ **S&P 500**指數：標普500的日線價格數據。
  - ❖ 台股加權指數（**TAIEX**）：台股指數的日線價格數
  - ❖ 數據的使用：
    在訓練過程中，使用學習率調整和混合精度訓練技術，以提升模型的收斂速度和穩定性。
    - ➢ 80%的數據用於訓練
    - ➢ 20%的數據用於測試

- 預處理（含**EDA**分析）
  - ❖ 標準化處理**(Standardization):**所有數據皆經標準化處理，以確保不同特徵的尺度一致，避免模型偏向於高值特徵。
  - ❖ 滑動窗口（**Sliding Window**）**:**讓模型能夠以固定長度的時間序列數據進行訓練和預測。

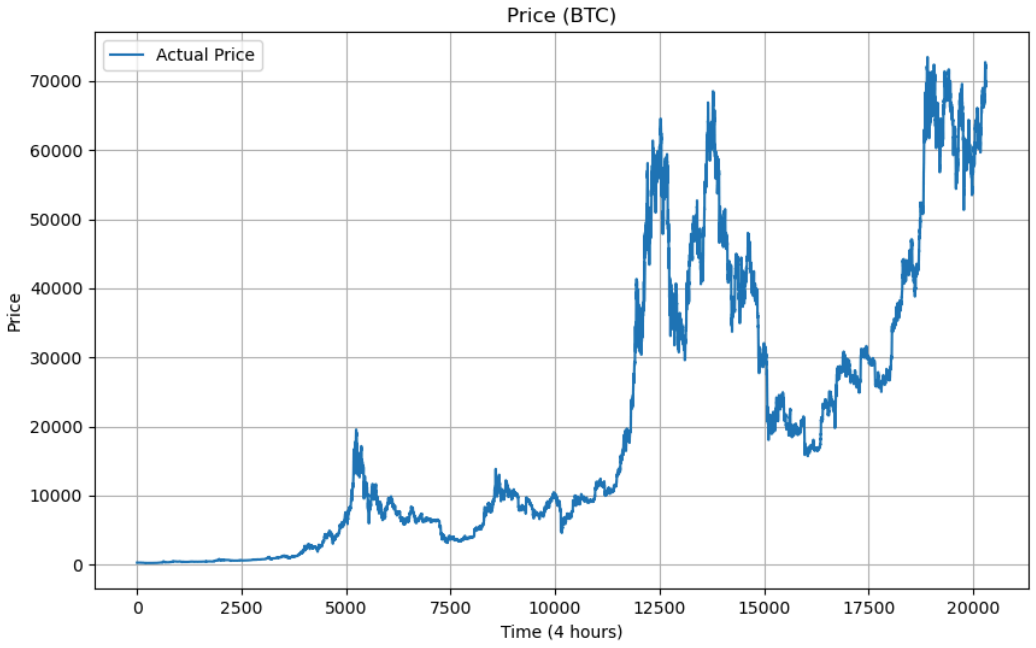
  找出趨勢、季節性模式、相關性

- 評估的步驟 **(**三組數據包含的特徵略有不同**):**
  - ❖ **BTC/USD**（**4**小時）**:** 使用最為完整，使用收盤價、為加斯通道(PMA12…PMA676)、隨機指標(KDJ)、RSI、時間加權MACD、QQE動能指標、Hull Moving Averag、成交量、成交量加權指標。
    - ➢ "features": "'close','PMA12', 'PMA144', 'PMA169', 'PMA576', 'PMA676', 'KD', 'J', 'RSI', 'MACD', 'Signal Line', 'Histogram', 'QQE

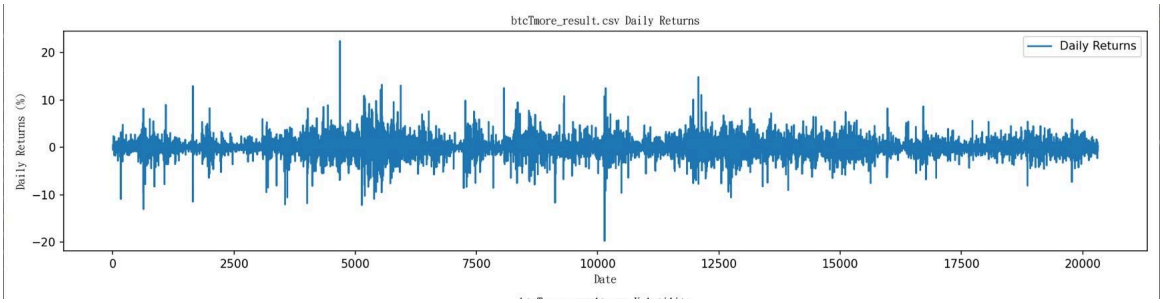
Line', 'Histo2','MHULL','SHULL','volume', 'Bullish Volume Trend','Bearish Volume Trend'"

- ❖ **S&P 500** 和 台股加權指數(每日)：除了收盤價(close)，還包含多個技術指標(如PMA、KD、RSI、MACD等)作為特徵，這些技術指標能夠提供更多市場情境的資訊，提升模型的預測準確性。
  - ➢ "features": "'close','PMA12', 'PMA144', 'PMA169', 'PMA576', 'PMA676', 'KD', 'J', 'RSI', 'MACD', 'Signal Line', 'Histogram', 'QQE Line', 'Histo2'"

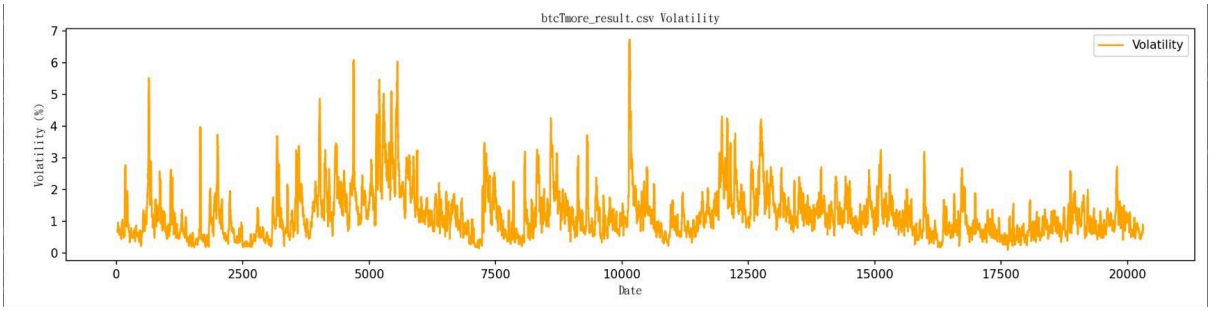
- 原始數據**performance indicators**
  - ❖ **BTC**
  - ➢ **Price**



- ➢ **Daily Returns**

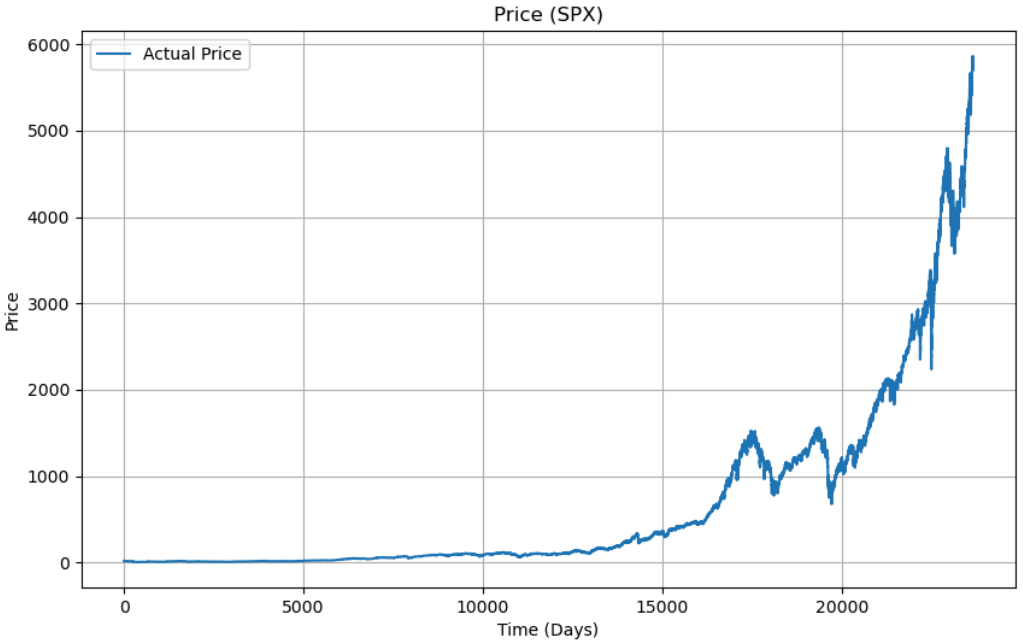


➢ **Volatility**

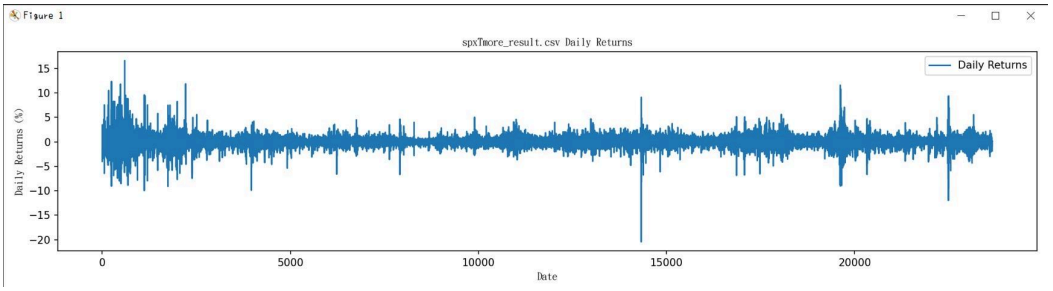


❖ **S&X**

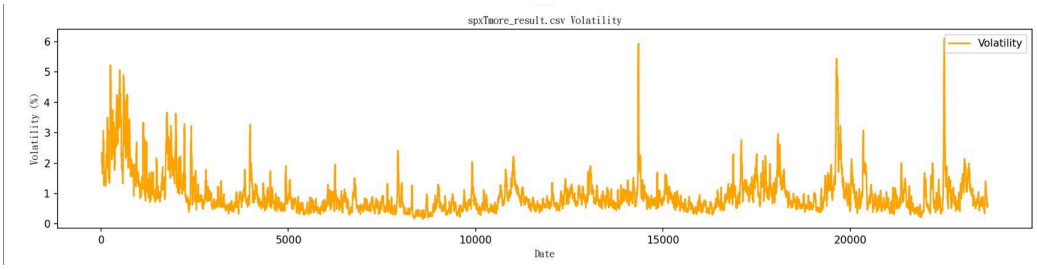
➢ **Price**



➢ **Daily Returns**



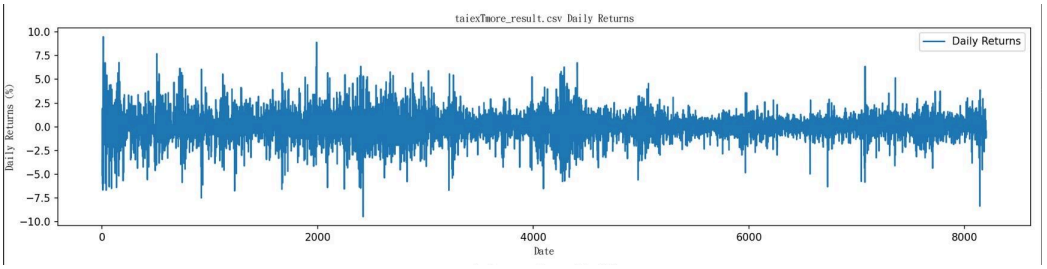
➢ **Volatility**



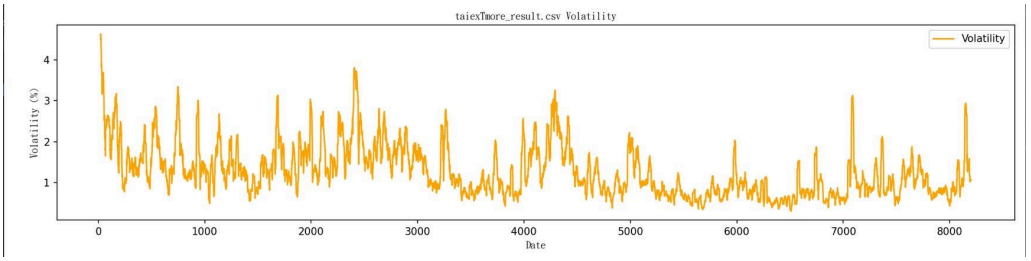
❖ **TAIEX**

➢ **Price**

➢ **Daily Returns**



➢ **Volatility**



3.模型架構與配置

在這個專題中，我們使用了Transformer模型來進行市場價格預測。選擇Transformer模型，是因為它在處理長期依賴的序列數據方面表現出色，能夠更好地捕捉價格序列中的模式與趨勢。模型的架構和參數配置如下：

- 模型結構:
  Transformer模型，由多層Encoder組成，每層包含多頭自注意力機制。

- 主要參數:
  - ➢ 序列長度(seq_len):
    設置於512，用於模型學習前幾個時間步的模式。
  - ➢ 隱藏層維度(hidden_dim):256。
  - ➢ 注意力頭數(nhead):4個頭，用於捕捉不同時間步的關聯性。
  - ➢ 2層數(num_layers):4層，用於增強模型的表現能力。
  - ➢ 學習率:0.0003。
  - ➢ 訓練批次大小(batch_size):
    設置為256。這樣的顯存使用10.7GB，使用RTX3080訓練

- 模型測試：
- ➢ 回測(**Backtesting**):

在每個epoch結束後，模型會針對測試數據進行回測，通過逐步預測的方式來 模擬實際的預測情境，並選取對於測試集損失最小的模型。最後用這個最好的模型把過去的訓練資料都回測一次。預測結果儲存於CSV文件中，以便後續進行結果分析和評估。結果分析。我們使用可視化和評估指標來分析模型的預測效果，並比較不同市場的預測準確度。

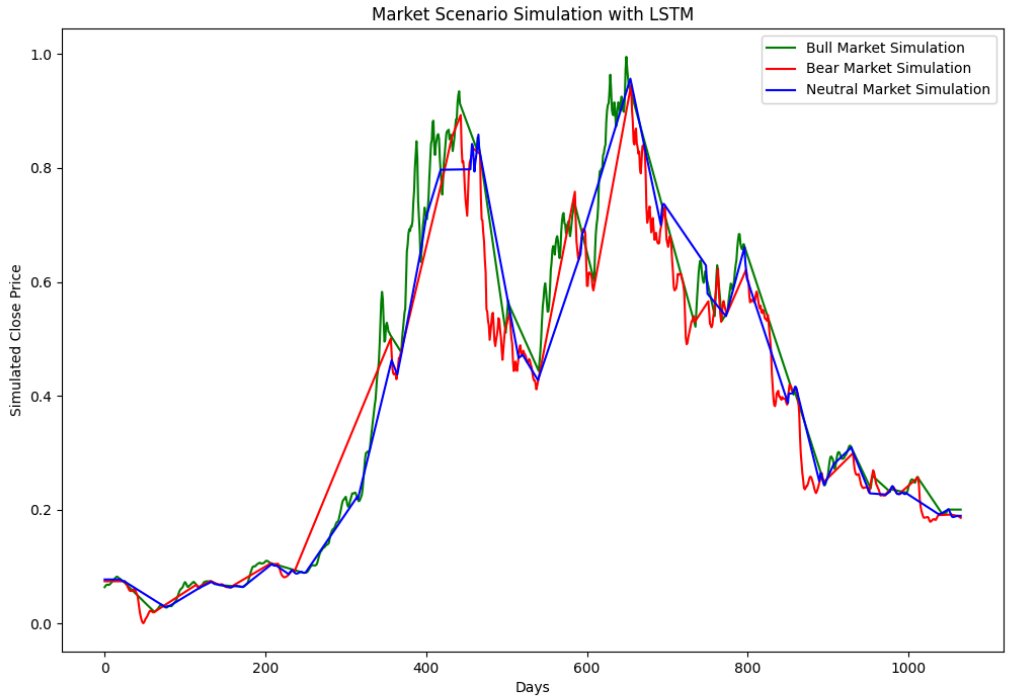
4.可視化與分析

透過draw_4hours.py和draw_day.py腳本，繪製了實際價格與預測價格的對比圖。不同市場的預測結果會顯示在同一張圖表上，讓我們可以直觀地看到模型預測與實際價格的吻合程度。

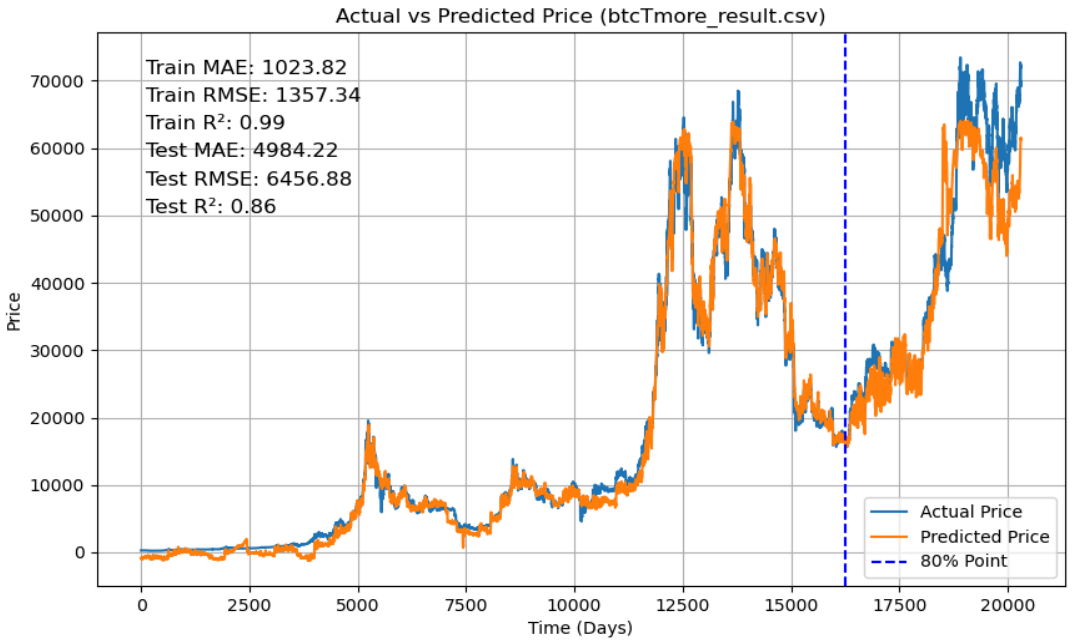
- 評估指標:
  - ➢ **MAE**(**Mean Absolute Error**):
    平均絕對誤差，用於衡量預測值與實際值的平均偏差。
  - ➢ **RMSE**(**Root Mean Square Error**):
    均方根誤差，反映模型在預測中出現的偏差大小。
  - ➢ **R^2**(**R-squared**):
    決定係數，用於評估模型的擬合效果。

- 結果解讀:

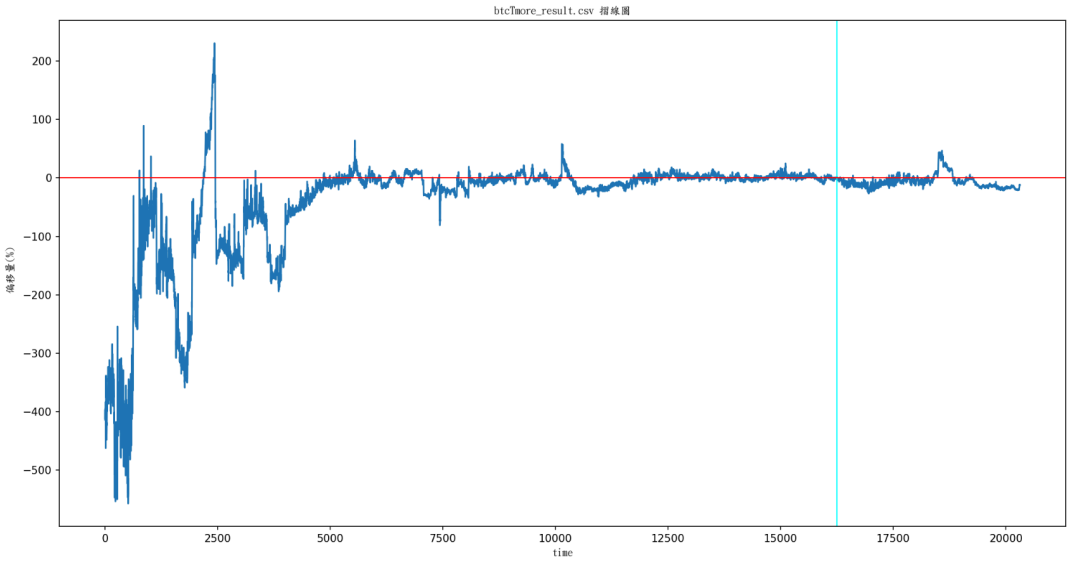
1. BTC



圖一：BTC(A) bull_bear



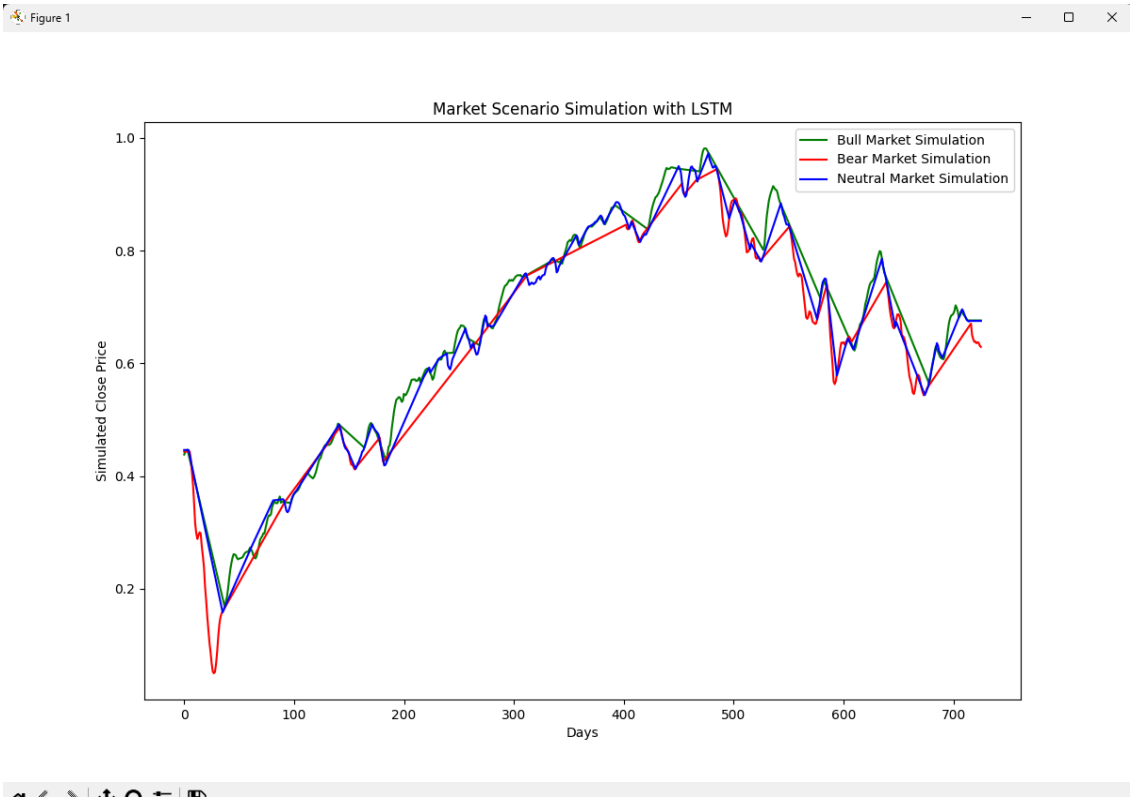
圖二：BTC(B)



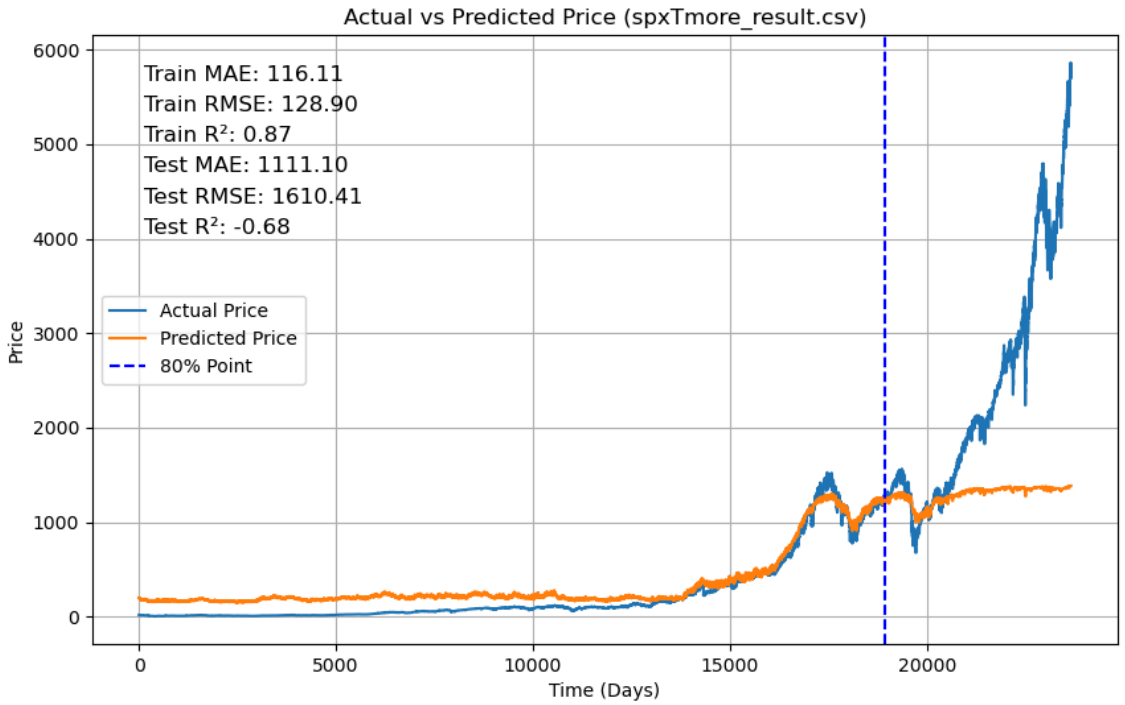
圖三：BTC(C)

- 圖A中顯示模擬的牛市和熊市情境下的價格波動較大，尤其在價格高點和低點處明顯看到不同情境之間的區別，在中性市場的模擬價格走勢相對於牛市和熊市平穩，但它也受到牛市和熊市的影響，因為市場往往會在兩者之間變化。
- 在比特幣的模擬結果 測試集$R2$有0.99的水準相當不錯，測試集有也很好的預測水準。但是模型似乎不記得訓練集前面發生了什麼。預測差的很遠，甚至預測到負的價格，推測是參數量不夠，記不起來。

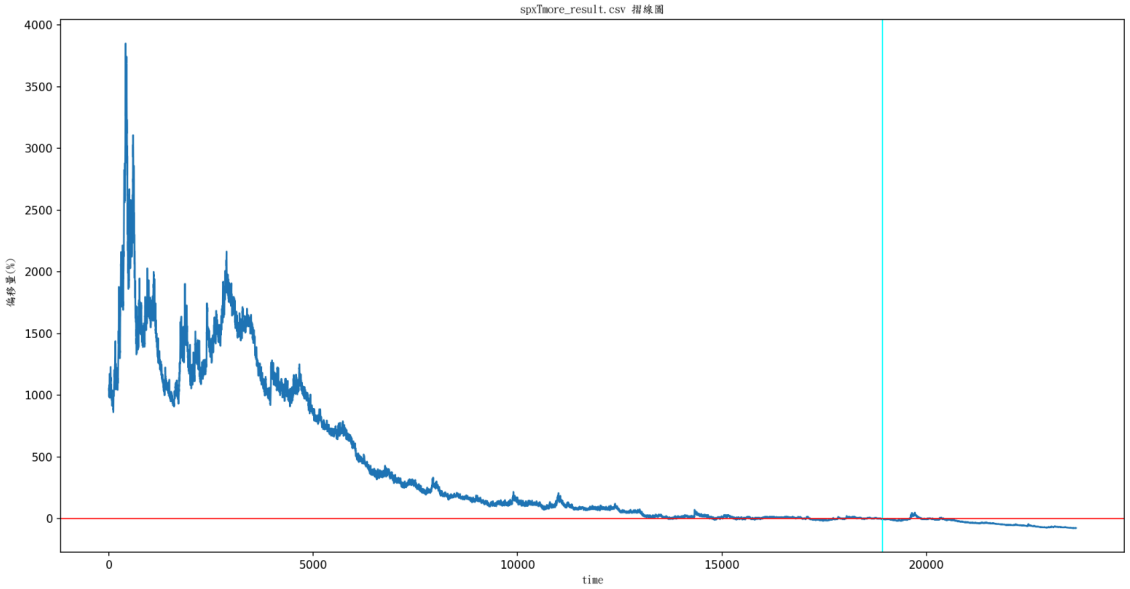
2. SPX



圖一：SPX(A) bull_bear



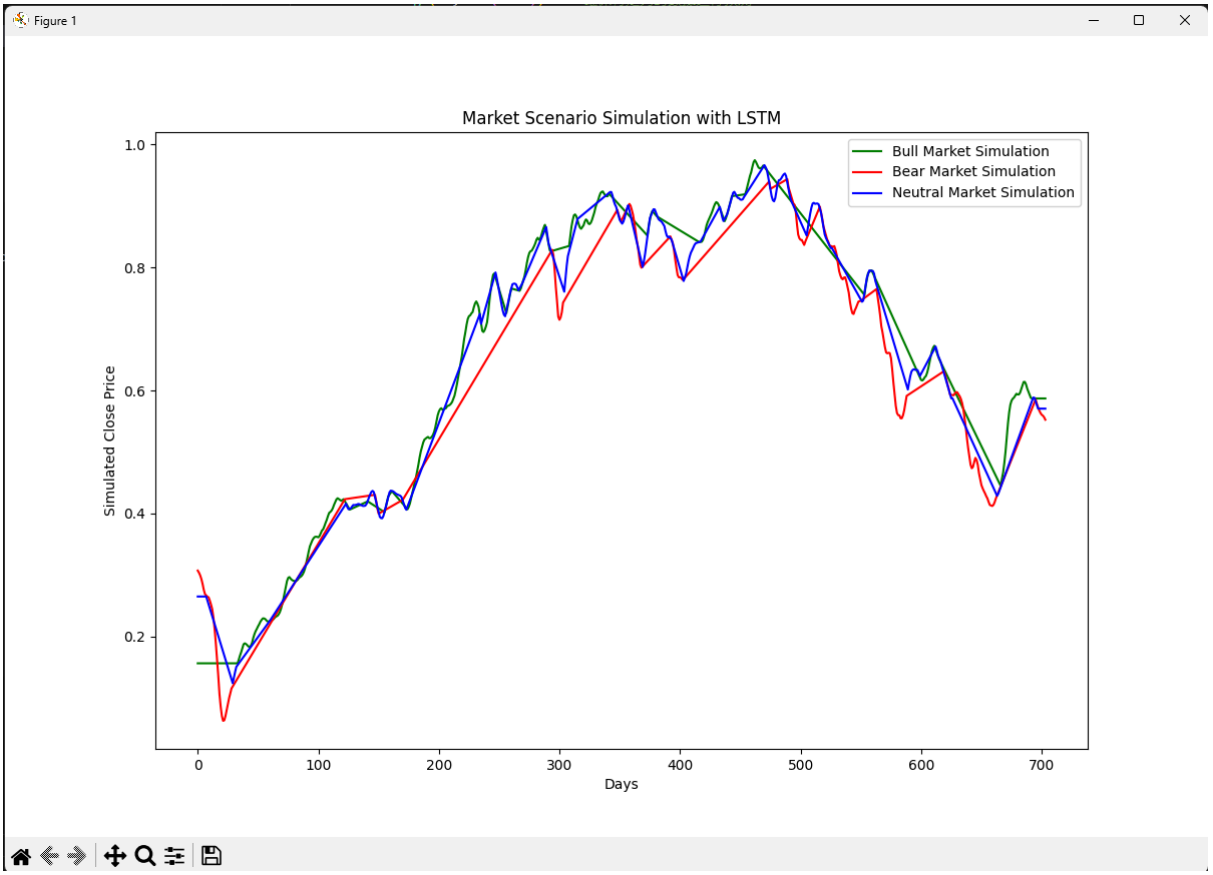
圖二：SPX(B)



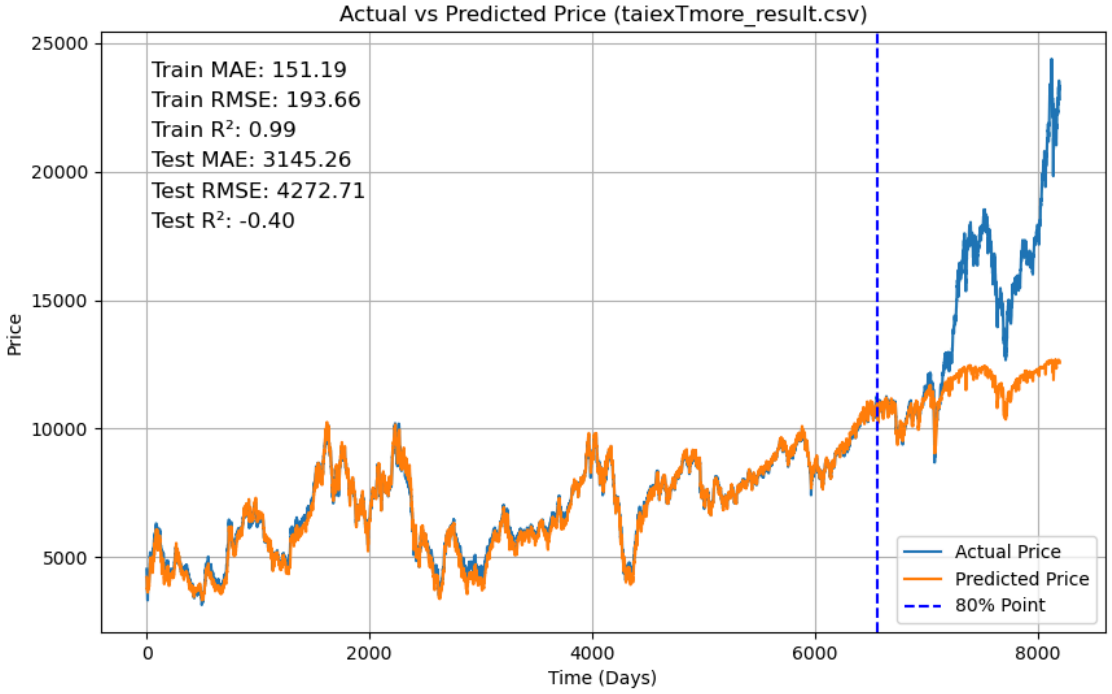
圖三：SPX(C)

- 從圖中可以看到，價格從起點約 **0.2** 左右開始，並在接近 **200** 天時開始快速上升，在 **400** 至 **500** 天左右達到高峰（約在 **0.8** 至 **1** 之間），隨後，模擬的價格開始呈現下跌趨勢，並且在 **700** 天左右穩定在較低的水平。

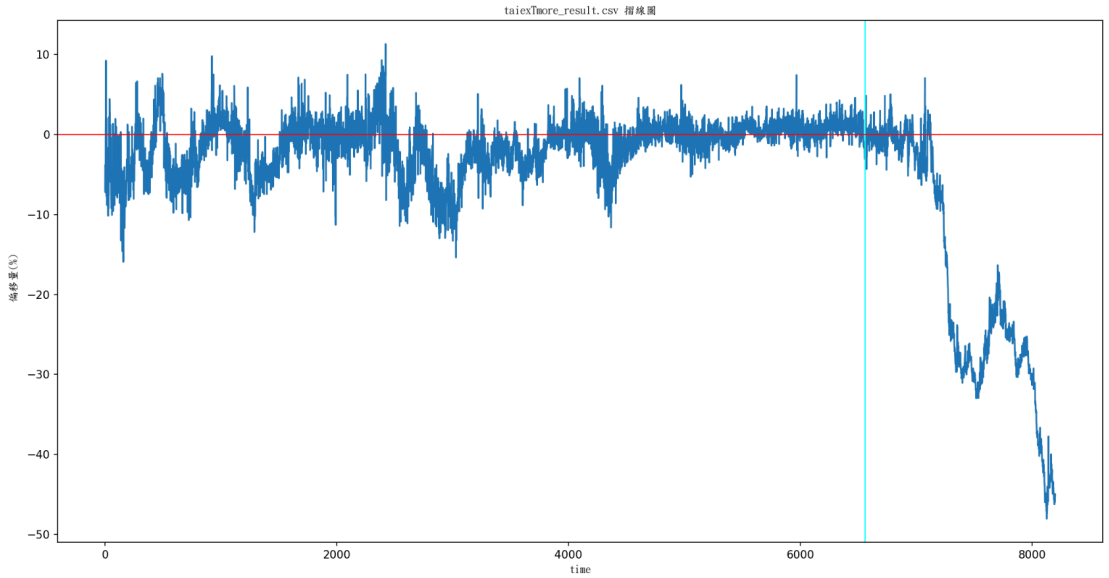
3.TAIEX



圖一：TAIEX(A) bull_bear



圖二：TAIEX(B)



圖三:TAIEX(C)

- 在約 **100-500** 天之間，三種情境下的模擬價格都處於上升期，且彼此之間的差異不大。這顯示了模型在不同市場情境下的價格模擬，特別是在強烈的趨勢時期，情境的差異並不明顯，在高峰過後的下跌期（約 **500** 天後），不同市場情境的線條差異變得更為明顯，熊市模擬的下跌速度最快，而牛市模擬相對穩定。
- 數據解讀
  對於BTC/USD數據，模型在4小時的預測粒度下表現尚可，能夠捕捉主要的價格變動趨勢，但在短期波動較大時預測準確性略有下降，
  對於S&P 500指數和台股指數，模型在日線下能較好地追蹤指數的趨勢走向。技術指標的加入似乎幫助模型更準確地預測市場走勢，尤其是在較大漲跌的情境下。

5.未來展望

本專題展示了Transformer模型在金融市場價格預測中的應用潛力。模型在多種市場（加密貨幣、S&P 500、台股指數）上的預測效果證明了其在捕捉長期依賴性和市場模式上的優勢。然而，模型在極端波動的情況下仍然存在一定的誤差，這可能是由於模型對短期變動的敏感度不足。

未來工作方向：

引入更多特徵：包括市場情緒(貪婪恐慌指數)、宏觀經濟數據等，以提升模型對市場情境的識別能力。
模型優化：嘗試使用更深層的Transformer或引入其他深度學習模型（如LSTM、GRU）進行組合，提升模型的泛化能力。
擴展到更多市場：測試其他金融市場（如黃金、大宗商品）的預測效果，以驗證模型的通用性。

6.生成式AI的協助

原文
第一句:可以幫我生成一個在電腦上使用***RTX3080***，***pytorch***、上面這個數據，預測下一根***K***線的價格的***transformer***模型
AI就生給我了一個可用的AI架構以及程式碼，我的程式碼是訓練與回測是分開的，這些參數要統一，所以就把一些超參數寫在config.json，叫他幫我應用進去，也很好完成了任務，後來我發現模型的收斂速度實在太慢，所以我就開始問:要怎麼加快收斂速度***?***
他就提出使用更好的優化器AdamW、標準化、將 MSE 損失函數替換為平滑的損失函數等等一頓優化，再結合之前每個epoch就儲存模型，這樣就可以隨時中斷訓練了。

這裡使用AI debug，詢問如何增加更多的特徵進去模型、分割訓練集跟測試集、要怎麼加上儲存最好的模型，再加上一些時間的訓練，就大致上完成了這個AI模型的訓練。