








HW7

612415013 蕭宥羽

1. How to execute codes.

這次要跑三種不同的預測目標，而我分別嘗試了不同的方法，以下會詳細說明

 all_30_open.ipynb	2024/5/16 下午 06:38	Jupyter 來源檔案
 LSTM_Stock_Prediction.ipynb	2024/5/16 下午 04:50	Jupyter 來源檔案
 stt_15_volume.ipynb	2024/5/16 下午 06:33	Jupyter 來源檔案
 stt_15_volume_2.ipynb	2024/5/20 下午 02:29	Jupyter 來源檔案
 ual_stt_16_close.ipynb	2024/5/16 下午 05:50	Jupyter 來源檔案
 ual_stt_16_close_2.ipynb	2024/5/20 下午 02:56	Jupyter 來源檔案
 ual_stt_16_close_3.ipynb	2024/5/20 下午 03:16	Jupyter 來源檔案

原始程式碼說明

1. 將 stt 的資料切成訓練及測試

分別有這幾項數據 ['open', 'close', 'low', 'high', 'volume']

```
pivot = round(0.85 * len(df_stt))
print('pivot:', df_stt.loc[pivot, 'date'])

df_train = df_stt[:pivot].copy() # DataFrame
df_valid = df_stt[pivot:].copy() # DataFrame

FEATURES = ['open', 'close', 'low', 'high', 'volume']
std = StandardScaler().fit(df_train[FEATURES])
df_train[FEATURES] = std.transform(df_train[FEATURES])
df_valid[FEATURES] = std.transform(df_valid[FEATURES])

df_train.head(10)
```

2. 將數據分為 feature target

Feature : 9 天(window 天)為一筆資料 Target:後一天的資料

這樣就會產生多筆 9 天對 1 天的資料

由 9 天的['open', 'close', 'low', 'volume']這幾項作為特徵，預測['high']的值

```
class CompanyStockData(Dataset):
    def __init__(self, dataframe, window=10):
        super().__init__()
        self.df = dataframe
        self.window = window

    def __len__(self):
        return len(self.df) - self.window

    def __getitem__(self, idx):
        window_s = idx
        window_t = idx + self.window
        feature = self.df.iloc[window_s:window_t - 1]
        feature = feature[['open', 'close', 'low', 'volume']].values
        feature = torch.from_numpy(feature).float()
        target = self.df.iloc[window_t - 1]
        target = target['high'].item()
        target = torch.tensor([target]).float()
        return feature, target
```

3. 建構 lstm 模型

```
class LSTM(nn.Module):
    def __init__(self, input_size=4, hidden_size=32, num_layers=1, num_classes=1):
        super().__init__()
        self.lstm = nn.LSTM(
            input_size=input_size, hidden_size=hidden_size, num_layers=num_layers, batch_first=True
        )
        self.regressor = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        x, (h, c) = self.lstm(x)
        x = self.regressor(x[:, -1]) # We only takes last timestamps output of lstm
        return x
```

4. Training (使用 LSTM 模型進行訓練)

```
device = 'cuda'
model = LSTM(input_size=5, hidden_size=32, num_layers=1, num_classes=1).to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
```

5. 根據要預測的類別計算出 error

```
raw_open = normalize_open * std.scale_[0] + std.mean_[0]
raw_close = normalized_close * std.scale_[1] + std.mean_[1]
raw_low = normalized_low * std.scale_[2] + std.mean_[2]
raw_high = normalized_low * std.scale_[3] + std.mean_[3]
raw_volume = normalized_low * std.scale_[4] + std.mean_[4]
```

✚ 用 STT 前 15 天的資料預測第 16 天的交易量(volume) `stt_15_volume`

1. `stt_15_volume`

用這 4 項預測 volume

```
feature = feature[['open', 'close', 'low', 'high']].values
feature = torch.from_numpy(feature).float()
target = self.df.iloc[window_t - 1]
target = target['volume'].item()
```

2. `stt_15_volume2`

用這 5 項預測 volume

```
feature = feature[['open', 'close', 'low', 'high', 'volume']].values
feature = torch.from_numpy(feature).float()
target = self.df.iloc[window_t - 1]
target = target['volume'].item()
```

3. 根據 volume 所對應到 num=4，來計算 error

```
num = 4

preds, trues = evaluate(train_set)
preds = preds * std.scale_[num] + std.mean_[num]
trues = trues * std.scale_[num] + std.mean_[num]
```

✚ 使用其他同領域公司(ual)前 15 天的資料來預測 STT 第 16 天的收盤價(close) `ual_stt_16_close`

1. `ual_stt_16_close` `ual_stt_16_close2`

✚ 載入 ual 資料

```
df_all = pd.read_csv(csv_path_all)
df_ual = df_all[df_all['symbol'] == 'UAL']
df_ual = df_ual.sort_values(by='date')
df_ual.head(10)
```

✚ 將 ual 作為 train val data，並將 stt 做為測試資料

```
df_train = df_ual[:pivot].copy() # DataFrame
df_valid = df_ual[pivot:].copy() # DataFrame
df_test = df_stt.copy()
```

由 ual 的這 4 項及這 5 項來預測 ual 的 close

```
feature = feature[['open', 'close', 'low', 'volume']].values
feature = feature[['open', 'high', 'low', 'volume', 'close']].values
target = target['close'].item()
```

由 ual 的 data 訓練出來的模型去預測 stt 的 close

```
preds, trues = evaluate(test_set)
preds = preds * std.scale_[1] + std.mean_[1]
trues = trues * std.scale_[1] + std.mean_[1]
error = ((preds - trues) ** 2).mean()
fig, ax = plt.subplots()
ax.plot(trues, label='True')
ax.plot(preds, label='Pred')
ax.set_title(f'test Error={error:.3f}')
ax.legend()
plt.show()
```

2. ual_stt_16_close3

用 ual 作為前幾天的 data 去預測 stt 後 1 天的 close(我認為這樣比較符合題意)

```
class CompanyStockData(Dataset):
    def __init__(self, dataframe_ual, dataframe_stt, window=10):
        super().__init__()
        self.df_ual = dataframe_ual
        self.df_stt = dataframe_stt
        self.window = window

    def __len__(self):
        return len(self.df_ual) - self.window

    def __getitem__(self, idx):
        window_s = idx
        window_t = idx + self.window
        feature = self.df_ual.iloc[window_s:window_t - 1]
        feature = feature[['open', 'high', 'low', 'volume', 'close']].values
        feature = torch.from_numpy(feature).float()
        target = self.df_stt.iloc[window_t - 1]
        target = target['close'].item()
        target = torch.tensor([target]).float()
        return feature, target
```

使用整個股市的前 30 天資料來預測第 31 天的開盤價(open) all_30_open

用整個股市的資料

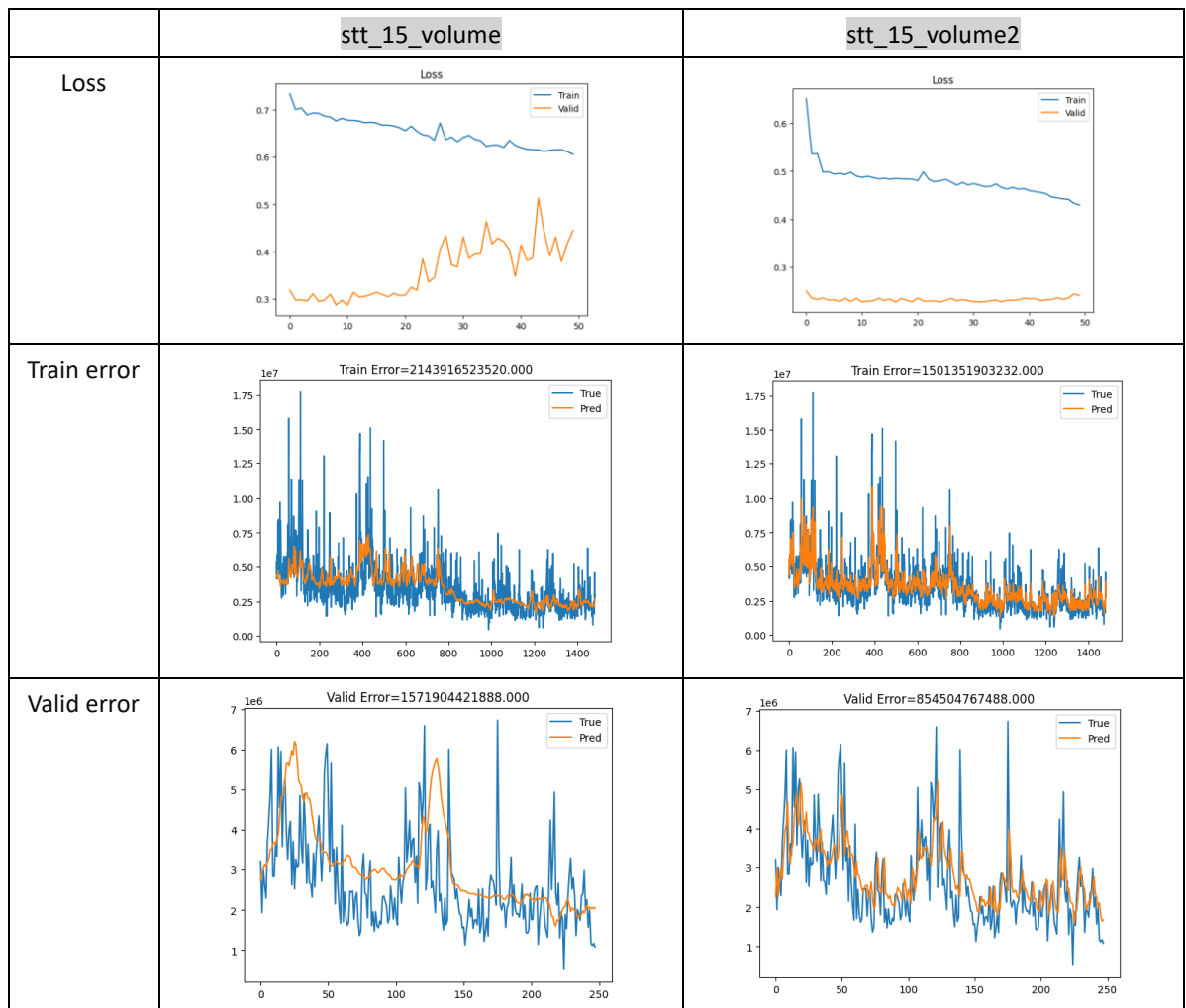
```
df_all = pd.read_csv(csv_path_all) # a DataFrame
df_all_stock = df_all[df_all['symbol'] == 'ALL']
df_all_stock = df_all_stock.sort_values(by='date')
df_all_stock.head(10)
```

由前 30 天去預測第 31 天

```
WINDOW = 31
train_set = CompanyStockData(df_train, window=WINDOW)
valid_set = CompanyStockData(df_valid, window=WINDOW)
```

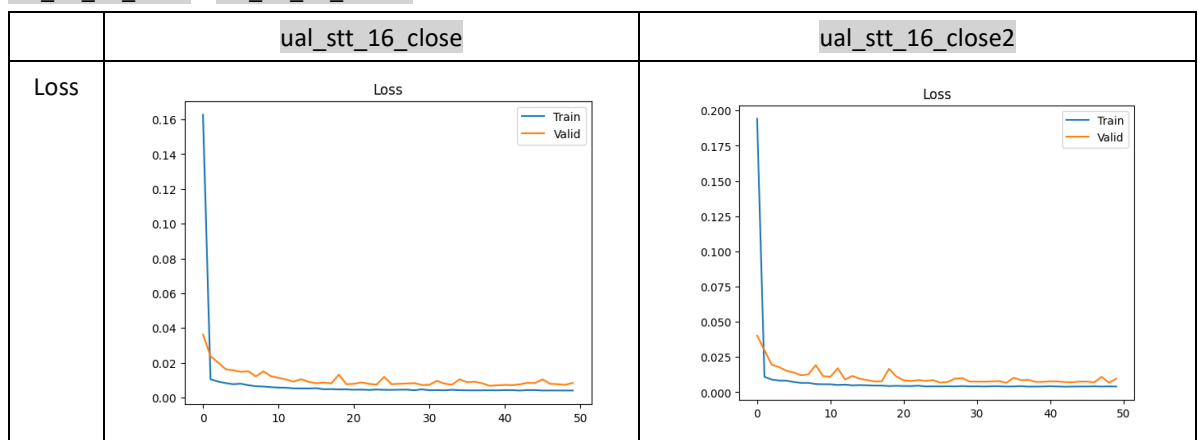
2. Experimental results

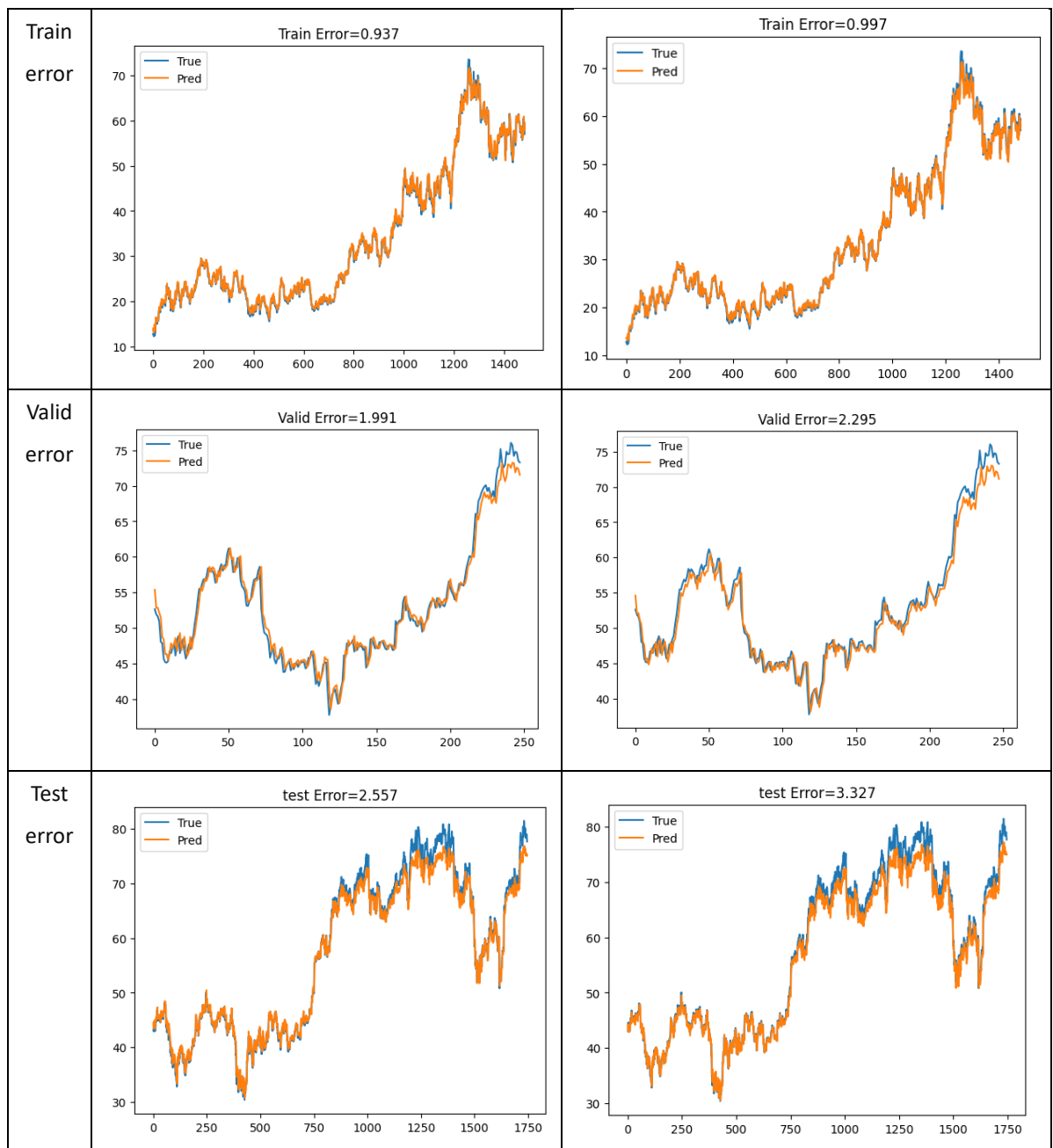
✚ 用 STT 前 15 天的資料預測第 16 天的交易量(volume)



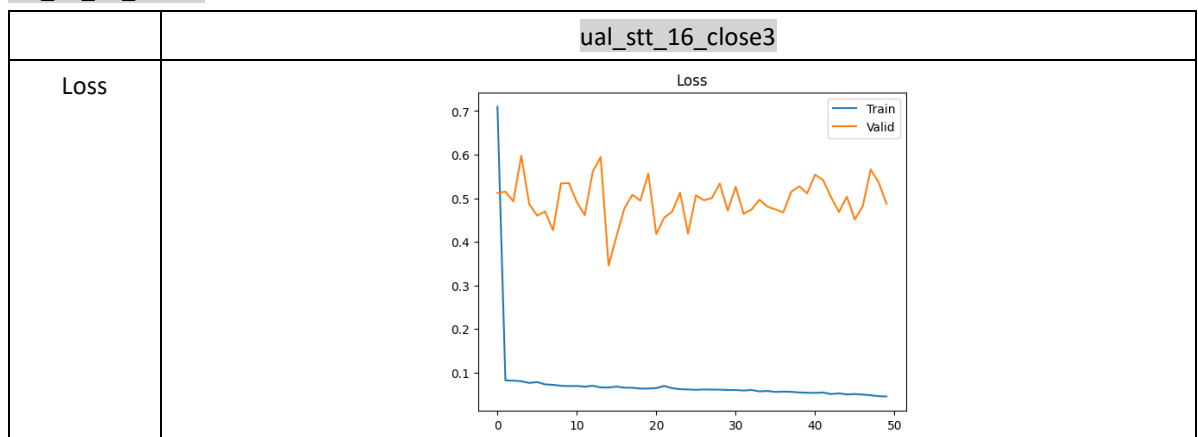
✚ 使用其他同領域公司(ual)前 15 天的資料來預測 STT 第 16 天的收盤價(close)

1. ual_stt_16_close ual_stt_16_close2



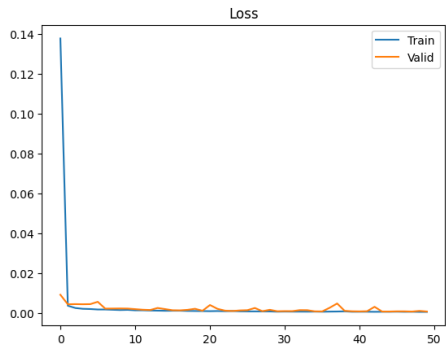
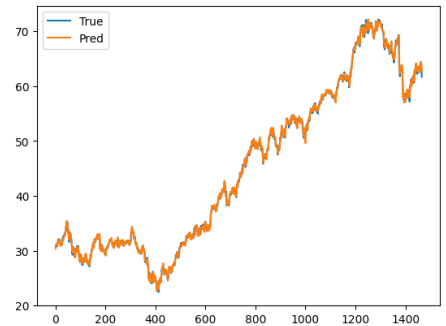


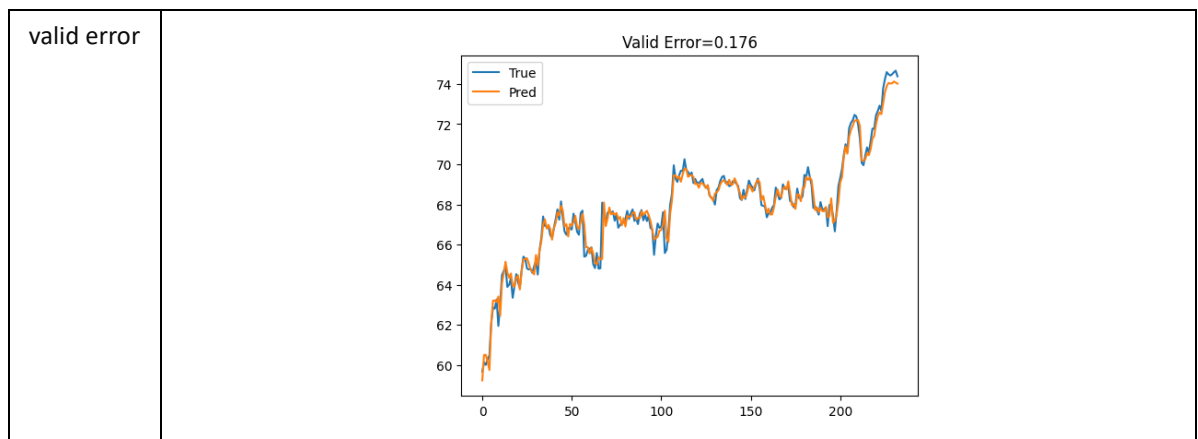
2. ual_stt_16_close3



train error	
valid error	

3. 使用整個股市的前 30 天資料來預測第 31 天的開盤價(open)

	all_30_open
Loss	
train error	



4. Conclusion

✚ 用 STT 前 15 天的資料預測第 16 天的交易量(volume)

```
feature = feature[['open', 'close', 'low', 'high']].values
feature = torch.from_numpy(feature).float()
target = self.df.iloc[window_t - 1]
target = target['volume'].item()

feature = feature[['open', 'close', 'low', 'high', 'volume']].values
feature = torch.from_numpy(feature).float()
target = self.df.iloc[window_t - 1]
target = target['volume'].item()
```

在 stt_15_volume 中，我們可以看到其驗證損失隨著訓練進行逐漸增大，並且波動較大，這表明模型存在過擬合現象，即模型在訓練數據上表現良好，但在驗證數據上效果不佳。此外，該模型的驗證誤差也顯示出較大波動，預測值與真實值之間存在顯著偏差，這進一步說明模型的泛化能力較差。

相比之下，stt_15_volume2 模型的驗證損失保持在較低且穩定的水平，我認為是因為將 volume 加入來做為特徵，所以預測 volume 會更加準確，但是總體的 error 非常高

相較 $\text{raw_high} = \text{normalized_low} * \text{std.scale_}[3] + \text{std.mean_}[3]$ 方式對 error 做處理，用 $\text{raw_volume} = \text{normalized_low} * \text{std.scale_}[4] + \text{std.mean_}[4]$ 的 error 值大很多

✚ 使用其他同領域公司(ual)前 15 天的資料來預測 STT 第 16 天的收盤價(close)

1. ual_stt_16_close ual_stt_16_close2

將 ual 作為 train val data，並將 stt 做為測試資料的方式，結果還不錯，而取這兩種不同特徵的結果也沒有差很多，可以推斷這整個領域的波動會互相影響，所以這樣的訓練方式得到的結果會還不錯，但是因為這樣的方法好像不符合提議，所以我試了另一個方法

2. ual_stt_16_close3

根據圖表我們可以看出，train 的 data 的走勢有符合整體的走向(也沒特別好)，但 val 的走勢就差非常多，從 loss 的那張圖也可以看出相關規律，我認為可能有以下原因：

- ✓ 資料相關性不足：每家公司的股價走勢受許多獨特因素影響，使用其他公司的資料作為輸入特徵可能與目標公司的股價沒有很強的相關性。
- ✓ 資料分佈差異：不同公司的股價分佈可能存在顯著差異，導致訓練集和驗證集之間存在分佈偏移 (distribution shift) 問題。

- ✚ 使用整個股市的前 30 天資料來預測第 31 天的開盤價(open)
- ✚ 跟前面的相比預測使用整個股市的前 30 天資料來預測第 31 天的開盤價(open)得到了較好的結果，我認為可能的原因如下：
 - ✓ 資訊來源更廣泛豐富
整個股市的行情資訊,涵蓋了各行各業的上市公司,能更好地反映整體市場走勢和氣氛。使用這些更全面的輸入資訊,模型就能抓住更多影響股價波動的潛在因素。
 - ✓ 捕捉行業關聯性
不同行業的股價往往存在一定的關聯性和傳導效應。使用整個市場的數據,模型可以更好地挖掘和學習這些行業間的內在聯繫,對單一股票的預測也會更準確。
 - ✓ 降低單一公司噪音影響
單一公司的股價波動,可能會受到一些特殊事件或消息的較大影響。但在整個市場的背景下,這些噪音的影響會被其他公司的數據所淨化和均攤,使預測模型更穩健。
 - ✓ 時間尺度拓展
30 天的時間範圍,比之前 15 天更長,能夠捕捉到一些更長週期的週期性規律,從而提高了模型的預測能力。

5. Discussion

做這份作業時對題意一開始有些不了解，所以試了好幾個方法，用了不同的資料去做預測，雖然結果的成效也沒有非常好，但在過程中我對 lstm 的模型有了更清楚的了解，而原始程式中還有提供不同的模型讓我們去做測試，但是我想說要使用統一的模型比較好觀察成果，所以就只有使用 lstm 的模型，之後我會再去嘗試其他模型或是用不同的資料去做訓練，甚至是改動模型的架構，我認為這會對我的學習有很大的幫助，這次的作業讓我學到很多，謝謝教授，謝謝助教。