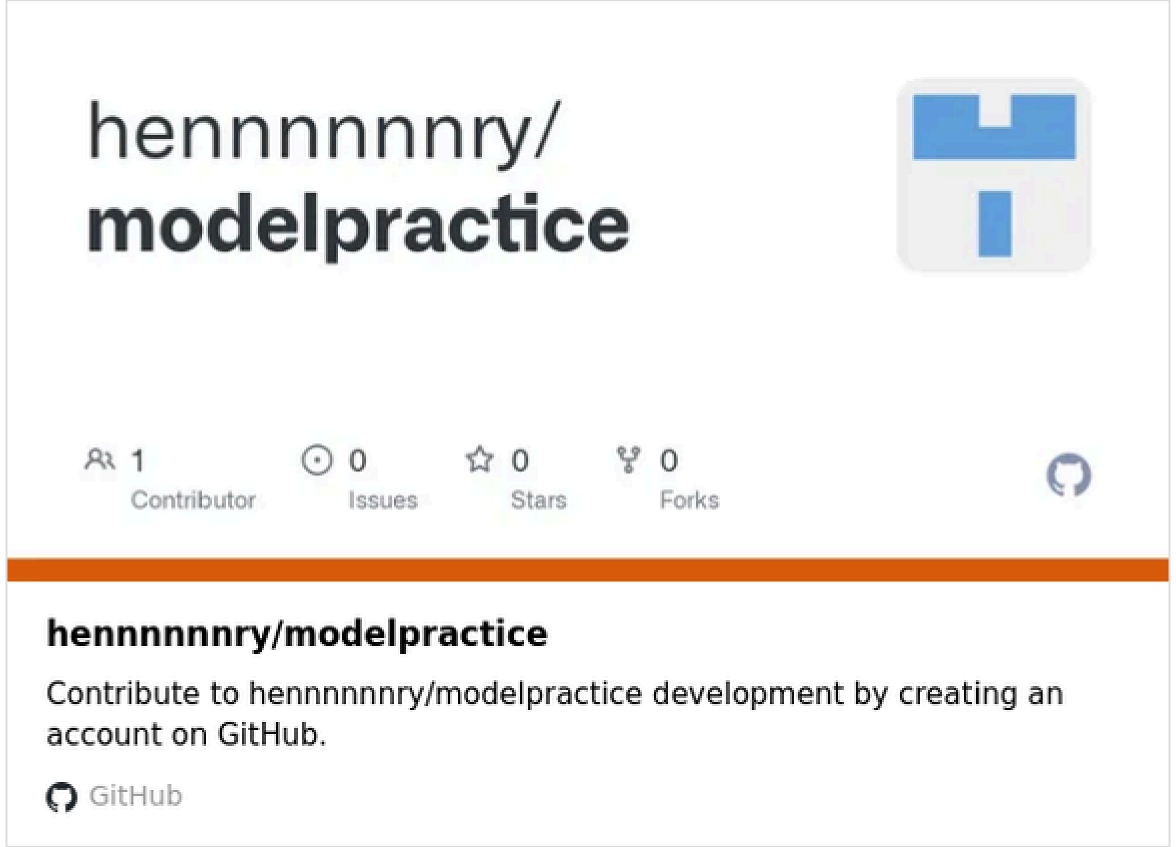
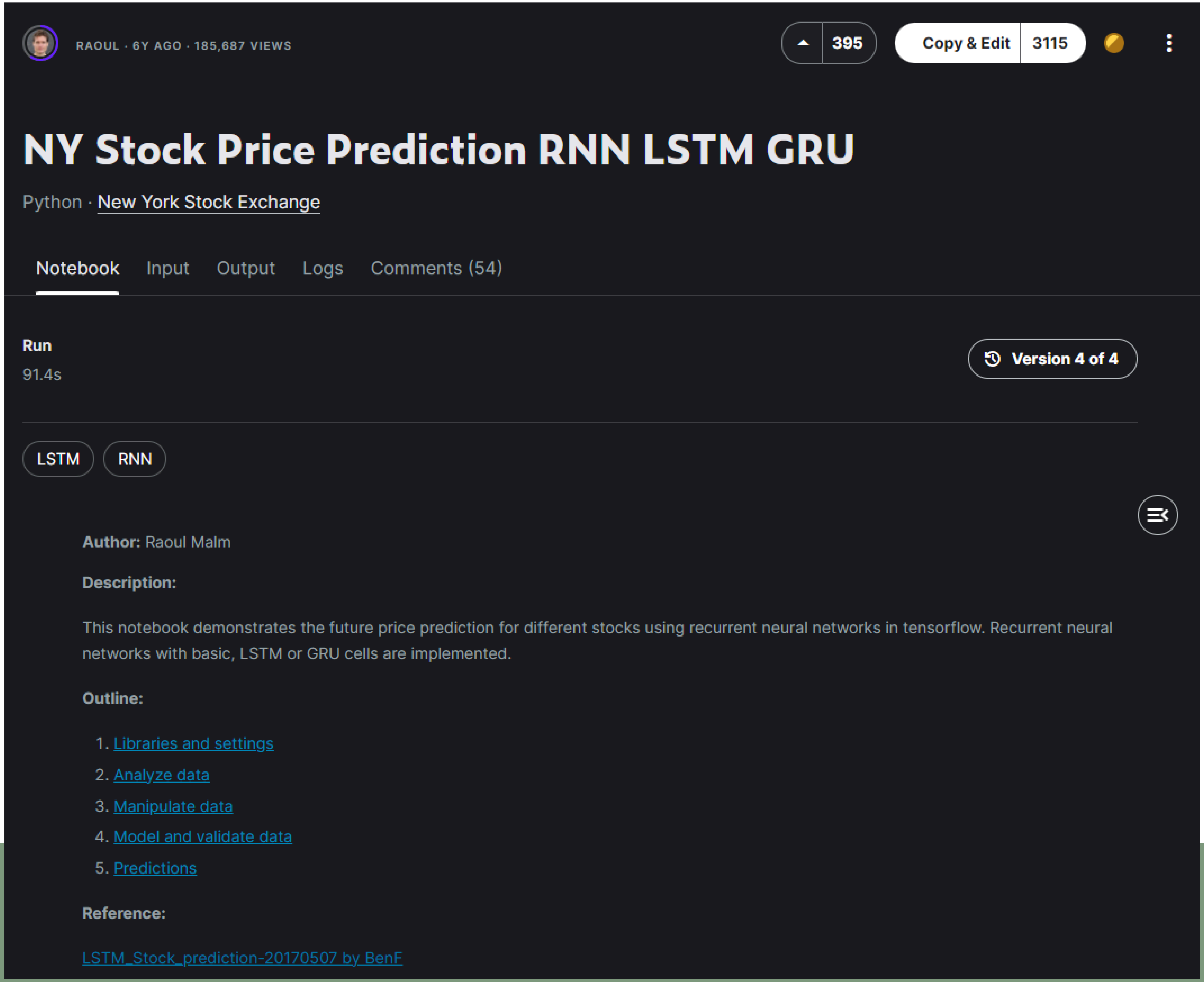
LSTM
模型嘗試

簡報者：陳弘蒼

目錄



GitHub詳細程式碼



範例參照

參考kaggle上的紐約股價預測範例，使用2330台積電十年的資料，資料切分並使用LSTM訓練模型。

來源網址：

步驟概述

• 資料蒐集

233010.csv

1 to 10 of 2444 entries Filter

date	stock_id	Trading_Volume	Trading_money	open	max	min	close	spread	Trading_turnover
2014-05-07	2330	33276563	3931216686	118	119	117.5	118.5	0.5	6945
2014-05-08	2330	30096125	3594265403	119.5	120	119	119.5	1	5599
2014-05-09	2330	23287585	2785144700	120	120	119	120	0.5	6405
2014-05-12	2330	37048901	4390300238	119.5	120	117.5	118.5	-1.5	6664
2014-05-13	2330	24046010	2894574200	120	121	119.5	120.5	2	7767
2014-05-14	2330	32903344	3991162751	121	122	120.5	122	1.5	9719
2014-05-15	2330	16813508	2040805195	121	122	120.5	122	0	5040
2014-05-16	2330	28679719	3497688718	122	123	121	122	0	7047
2014-05-19	2330	20829956	2524459168	122	122.5	121	121	-1	5325
2014-05-20	2330	25402168	3088614323	122	122	121	121	0	7109

Show 10 per page 1 2 10 100 200 240 245

透過FinMind獲取 2330台積電10年資料

• 資料處理

```
columns_to_drop = ['stock_id', 'Trading_Volume', 'Trading_money',
                   'spread', 'Trading_turnover']
df.drop(columns=columns_to_drop, axis=1, inplace=True)
```

drop掉不需要的參數

• 正規化函數

```
def  normalize_data(df):
     min_max_scaler  =  sklearn.preprocessing.MinMaxScaler()
     df['open']  =  min_max_scaler.fit_transform(df['open'].values.reshape(-1,  1))
     df['max']  =  min_max_scaler.fit_transform(df['max'].values.reshape(-1,  1))
     df['min']  =  min_max_scaler.fit_transform(df['min'].values.reshape(-1,  1))
     df['close']  =  min_max_scaler.fit_transform(df['close'].values.reshape(-1,  1))
     return  df
```

防止模型過擬合

• 套件使用

scikit-learn：計算判定係數

tensorflow ：模型建置相關

matplotlib ：圖表呈現

步驟概述

• 資料切分函數

```python
def load_data(stock, seq_len, valid_set_size_percentage=20, test_set_size_percentage=10):
    data_raw = stock.to_numpy()
    data = []

    for index in range(len(data_raw) - seq_len):
        data.append(data_raw[index: index + seq_len])

    data = np.array(data)
    valid_set_size = int(np.round(valid_set_size_percentage / 100 * data.shape[0]))
    test_set_size = int(np.round(test_set_size_percentage / 100 * data.shape[0]))
    train_set_size = data.shape[0] - (valid_set_size + test_set_size)

    x_train = data[:train_set_size, :-1, :]
    y_train = data[:train_set_size, -1, :]

    x_valid = data[train_set_size:train_set_size + valid_set_size, :-1, :]
    y_valid = data[train_set_size:train_set_size + valid_set_size, -1, :]

    x_test = data[train_set_size + valid_set_size:, :-1, :]
    y_test = data[train_set_size + valid_set_size:, -1, :]

    return [x_train, y_train, x_valid, y_valid, x_test, y_test]
```

定義 load_data 函數切分成訓練集70%、測試集20%、驗證集10%

• 函數代入

```python
df_norm = df.copy()
df_norm = normalize_data(df_norm)

seq_len = 20
x_train, y_train, x_valid, y_valid, x_test, y_test = load_data(df_norm, seq_len)
```

數據正規化，序列長度選擇20天

步驟概述

- **函數建立**

```python
index_in_epoch = 0
perm_array = np.arange(x_train.shape[0])
np.random.shuffle(perm_array)

def get_next_batch(batch_size):
    global index_in_epoch, x_train, perm_array
    start = index_in_epoch
    index_in_epoch += batch_size

    if index_in_epoch > x_train.shape[0]:
        np.random.shuffle(perm_array) # shuffle permutation array
        start = 0 # start next epoch
        index_in_epoch = batch_size

    end = index_in_epoch
    return x_train[perm_array[start:end]], y_train[perm_array[start:end]]
```

建立獲取下一批序列為20天的函數

- **參數調適**

```python
n_steps = x_train.shape[1]
n_inputs = x_train.shape[2]
n_outputs = y_train.shape[1]
n_neurons = 200
n_layers = 3
```

```python
learning_rate = 0.001
batch_size = 50
n_epochs = 250
train_set_size = x_train.shape[0]
test_set_size = x_test.shape[0]
```

步驟概述

- **建立模型**

```
model = tf.keras.Sequential()
for _ in range(n_layers):
    model.add(tf.keras.layers.LSTM(units=n_neurons, activation='relu', return_sequences=True))
    model.add(tf.keras.layers.Dropout(rate=0.1)) # Lowered dropout rate
model.add(tf.keras.layers.LSTM(units=n_neurons, activation='relu'))
model.add(tf.keras.layers.Dense(units=n_outputs, kernel_regularizer=tf.keras.regularizers.l2(0.01)))
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), loss='mean_squared_error')
```

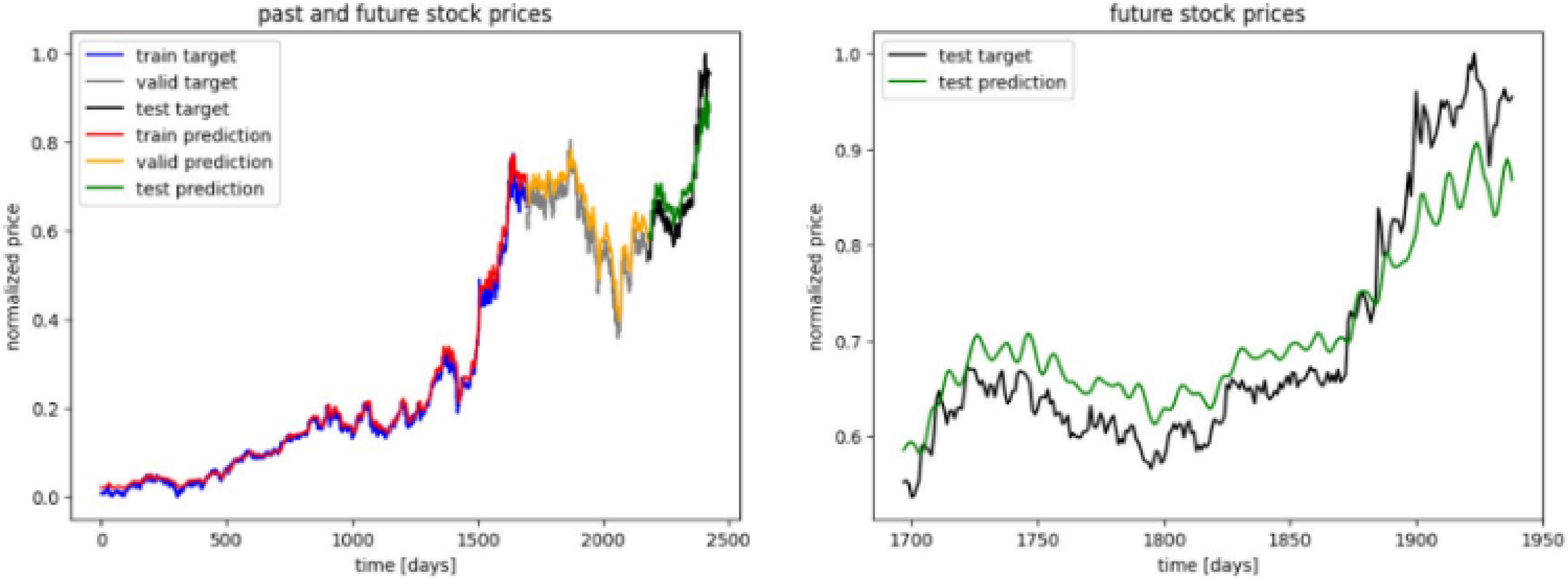
使用LSTM，激活函數使用relu，加入Dropout層和L2正則化防止模型過擬合，損失函數使用MSE

- **訓練模型**

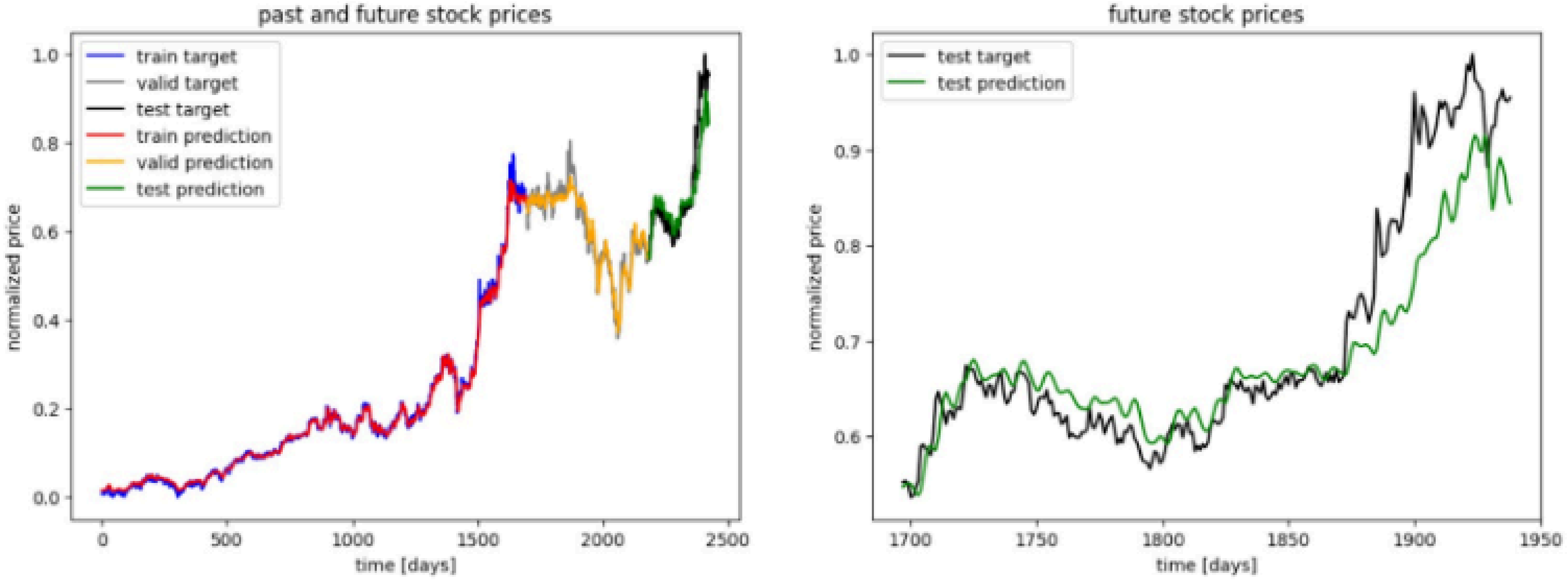
```
for epoch in range(n_epochs):
    for iteration in range(train_set_size // batch_size):
        x_batch, y_batch = get_next_batch(batch_size)
        model.train_on_batch(x_batch, y_batch)

    if epoch % 5 == 0:
        mse_train = model.evaluate(x_train, y_train, verbose=0)
        mse_valid = model.evaluate(x_valid, y_valid, verbose=0)

        y_train_pred = model.predict(x_train)
        y_valid_pred = model.predict(x_valid)

        corr_price_development_train = np.sum(np.equal(np.sign(y_train[:, 1] - y_train[:, 0])
                        , np.sign(y_train_pred[:, 1] - y_train_pred[:, 0])).astype(int)) / y_train.shape[0]
        corr_price_development_valid = np.sum(np.equal(np.sign(y_valid[:, 1] - y_valid[:, 0])
                        , np.sign(y_valid_pred[:, 1] - y_valid_pred[:, 0])).astype(int)) / y_valid.shape[0]
        print(f'{epoch} epochs: MSE train/valid = {mse_train:.6f}/{mse_valid:.6f}')
        print(f'Correct sign prediction train/valid: {corr_price_development_train:.2f}/{corr_price_development_valid:.2f}')

y_train_pred = model.predict(x_train)
y_valid_pred = model.predict(x_valid)
y_test_pred = model.predict(x_test)
```

每5個epoch做評估

模型演化與結語



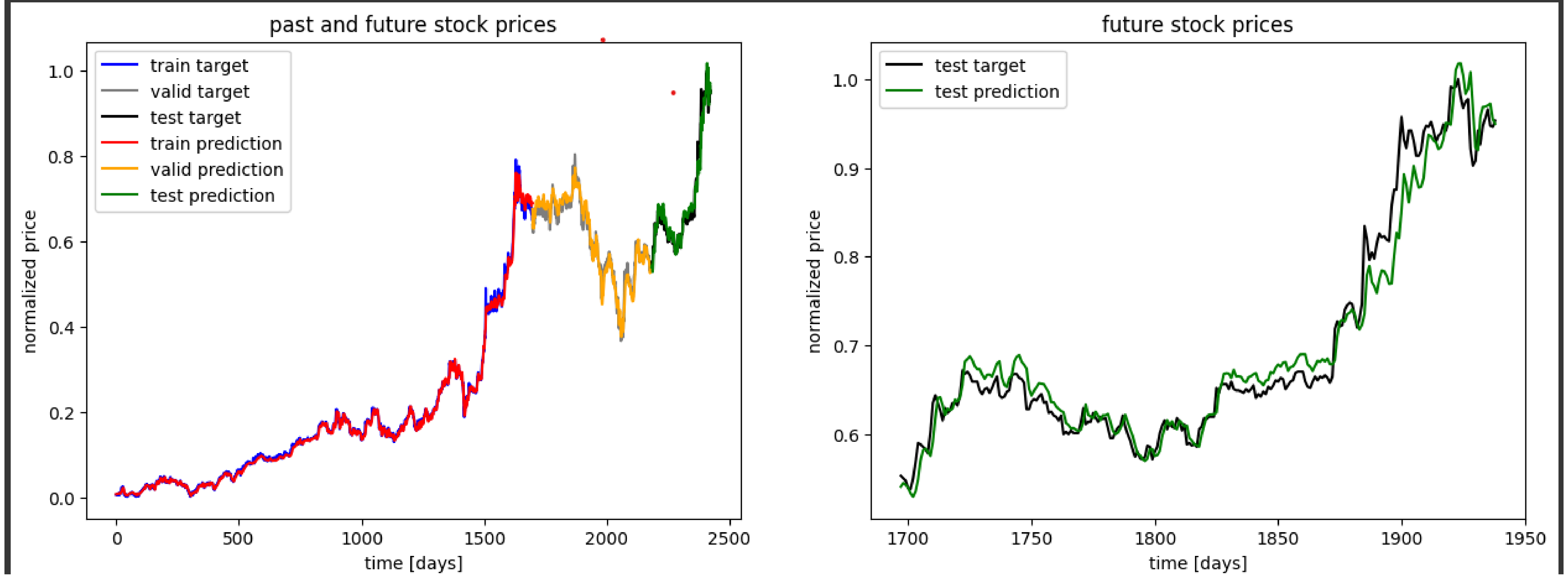
epochs = 100，layers = 4



epochs = 150，layers = 4

模型演化與結語

```
230 epochs: MSE train/valid = 0.000068/0.000350
Correct sign prediction train/valid: 0.53/0.54
235 epochs: MSE train/valid = 0.000088/0.000451
Correct sign prediction train/valid: 0.51/0.54
240 epochs: MSE train/valid = 0.000094/0.000346
Correct sign prediction train/valid: 0.47/0.54
245 epochs: MSE train/valid = 0.000101/0.000605
Correct sign prediction train/valid: 0.52/0.54
MSE on test set: 0.000651
R²: 0.9587683379345342
correct sign prediction for close - open price for train/valid/test: 0.45/0.54/0.54
```



epochs = 250，layers = 3

此模型為使用前20天的序列來預測第21天的數據，最後的嘗試為迭代250次、層數3層，判定係數為0.9587，模型有抓到整體趨勢。

練習操作模型的過程，我碰到的難關為各參數的調適，此步驟需要多方嘗試及經驗，因此節省運算資源和參數的選擇，有助於提升整體專案推進速度。
以上使用 Google colab 運算模型。

THE END