FinTech HW4

tags: `FinTech`

年級：資工碩二
學號：R08922125
姓名：張皓鈞

Problem 1.(i)

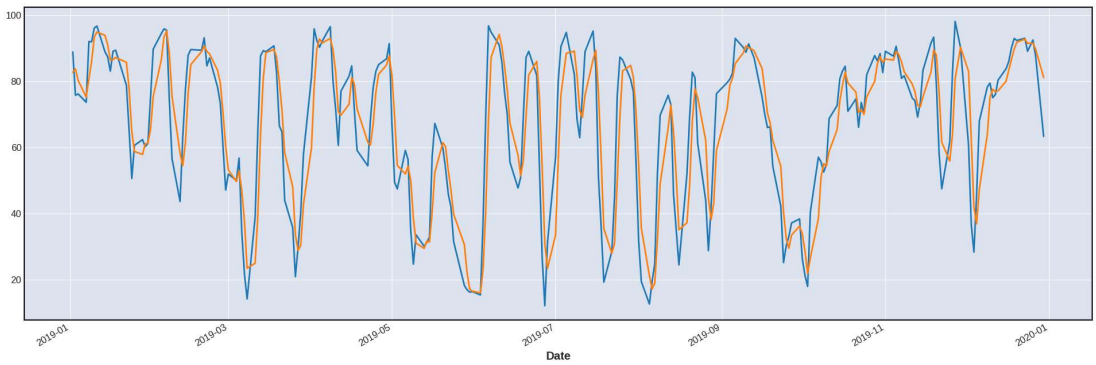
Plot S&P 500

a.) Candlestick chart with 2 moving average line (10 days and 30 days).
b.) KD line chart.
c.) Volume bar chart
Show your figures from 2019/1/1 to 2019/12/31

Candle stick with 10/30 MA and volume chart

- gray line= 10 MA
- blue line= 30 MA

KD line chart



- blue line= K line
- orange line = D line

Problem 1.(ii)

Data preprocessing

- add 4 features, MA 10 days, MA 30 days, K, D
- minMax normalizing

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

-

Date	open	high	low	close	volume	K	D	sma_5	sma_30
1994-01-03	0.009747	0.007927	0.010184	0.009468	0.022301	0.000000	0.000000	0.000000	0.000000
1994-01-04	0.009366	0.007909	0.010213	0.009985	0.027236	0.000000	0.000000	0.000000	0.000000
1994-01-05	0.009882	0.008241	0.010741	0.010221	0.033654	0.000000	0.000000	0.000000	0.000000
1994-01-06	0.010117	0.008661	0.011134	0.010067	0.030676	0.000000	0.000000	0.000000	0.000000
1994-01-07	0.009953	0.009111	0.011138	0.011060	0.027089	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...

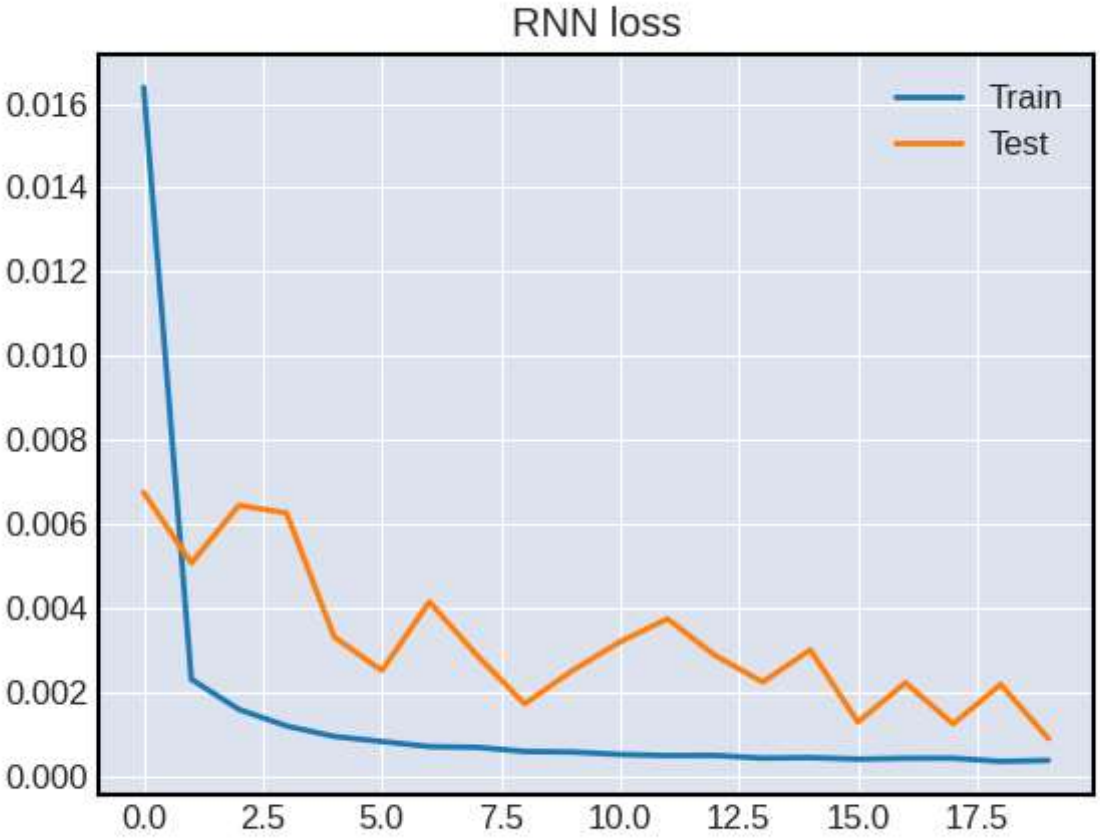
- train set: 1994~2017, test set: 2018~2019
- transform the data in oder to fit the model input size, using 30 days close value to predict the close value of next day.

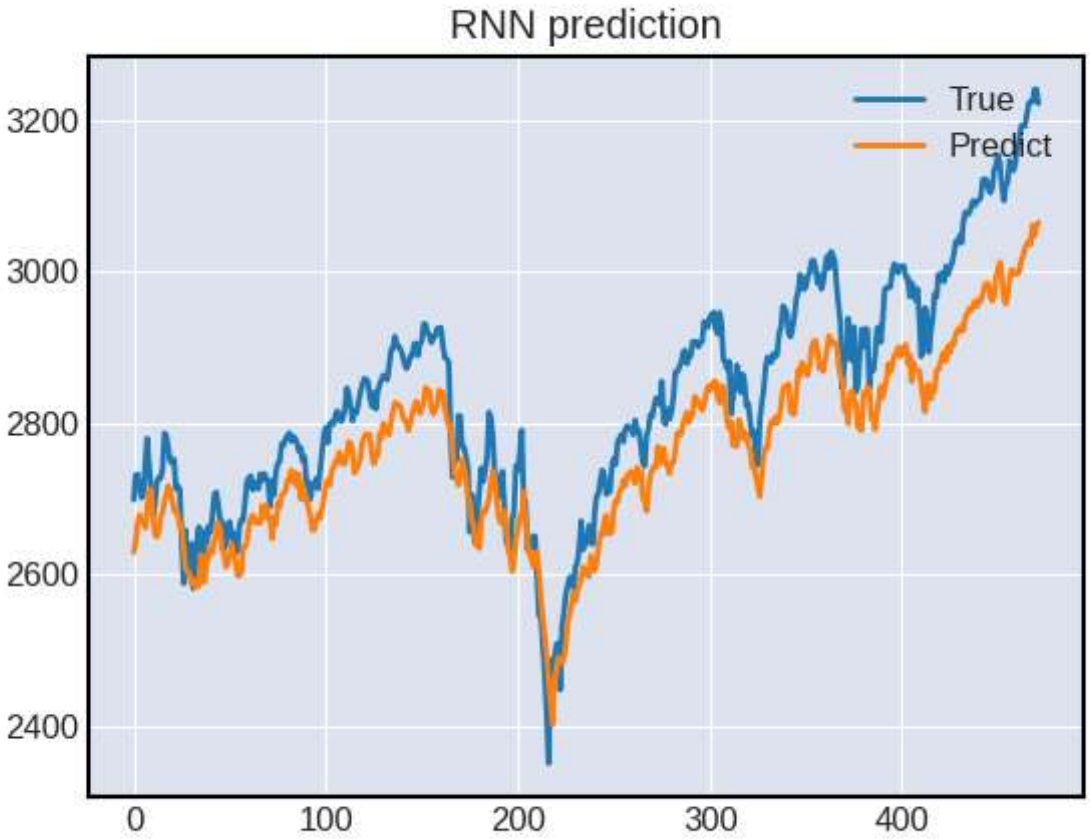
```
x_train = []   #預測點的前 30 天的資料
y_train = []   #預測點
x_test = []
y_test = []

for i in range(30, trainData.shape[0]):  # trainData.shape[0] 是訓練集總數
    x_train.append(trainData.iloc[i-30:i, :])
    y_train.append(y_trainData.iloc[i])
for i in range(30, testData.shape[0]):
    x_test.append(testData.iloc[i-30:i, :])
    y_test.append(y_testData.iloc[i]
                  )
x_train, y_train = np.array(x_train), np.array(y_train)  # 轉成numpy array的格式
x_test, y_test = np.array(x_test), np.array(y_test)
```

Problem 1.(iii) (iv) (v)

SimpleRNN with MSE

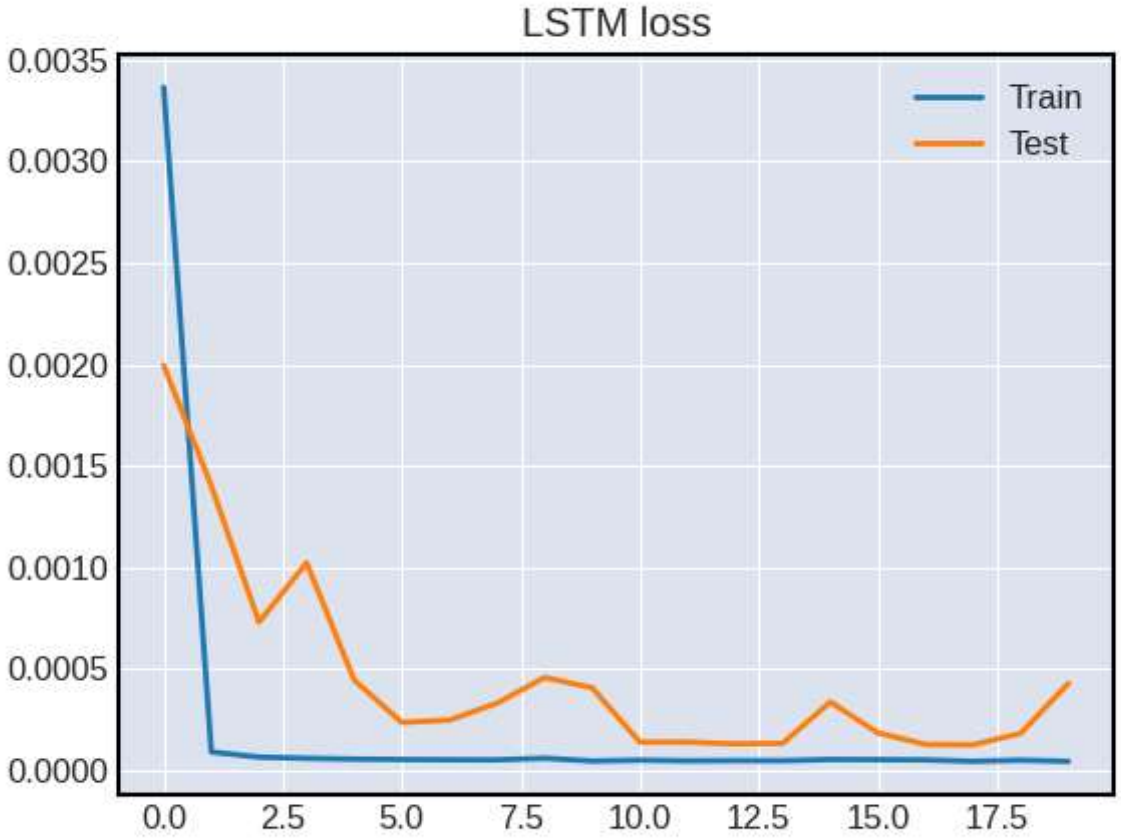

RNN loss
Train
Test
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

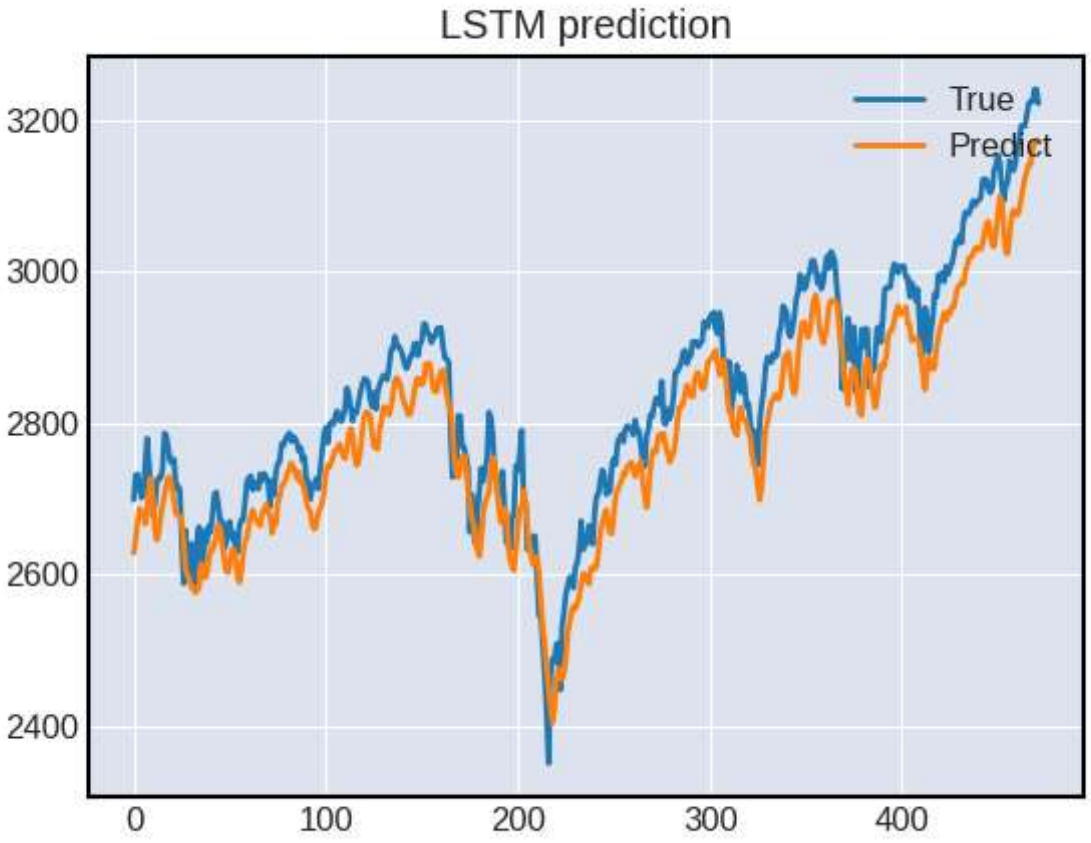

RNN prediction
True
Predict
3200
3000
2800
2600
2400
0
100
200
300
400

```
def create_RNN(x_train, y_train, x_test, y_test):
  batch_size = None
  steps = 30
  input_dim = 8
  epochs = 20
  model = Sequential()
  # 加 RNN 隱藏層(hidden layer)
  model.add(SimpleRNN(
      # 如果後端使用tensorflow，batch_input_shape 的 batch_size 需設為 None.
      # 否則執行 model.evaluate() 會有錯誤產生.
      batch_input_shape=(batch_size, steps, input_dim),
      units= 50,
      unroll=True,
  ))
  model.add(Dropout(0.2))
  model.add(Dense(10,activation='relu'))
  model.add(Dense(1))
  model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squa
  hist = model.fit(x_train, y_train,
            batch_size=batch_size, epochs=epochs,
            verbose=1, validation_data=(x_test, y_test))
  return hist, model
```

Problem 1.(vi)

LSTM cell


LSTM loss
Train
Test
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

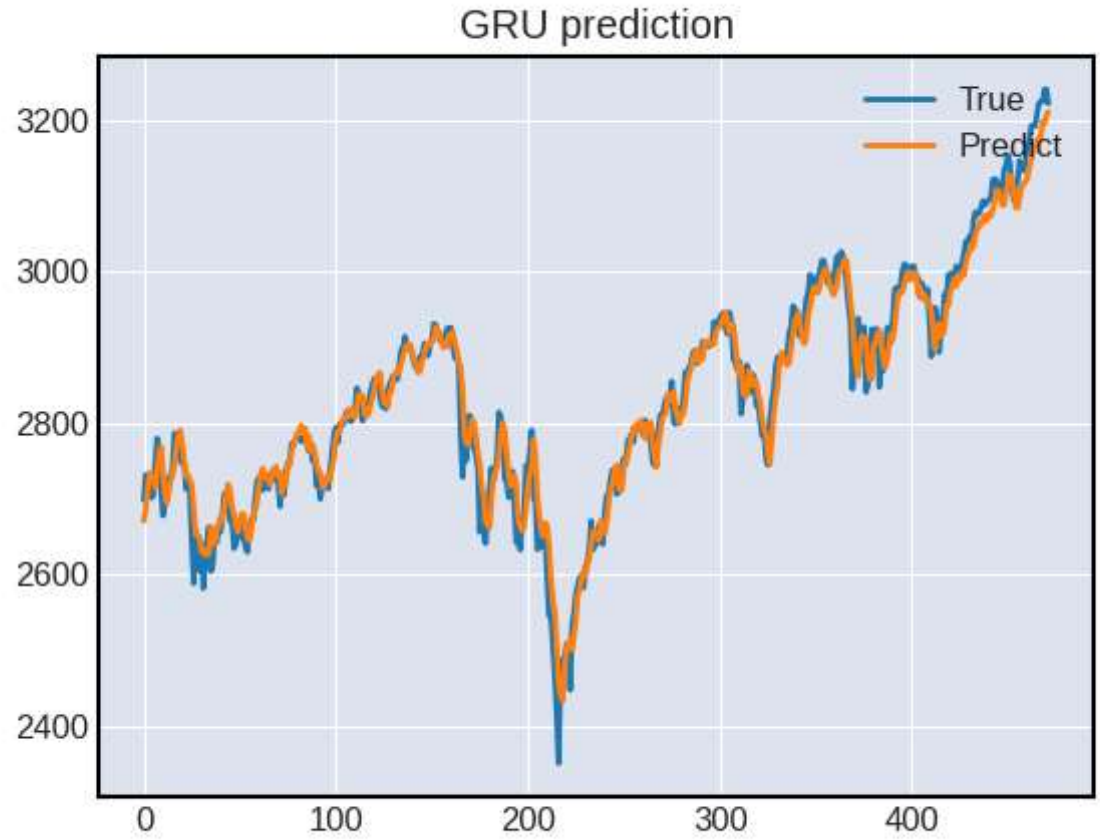

LSTM prediction
True
Predict
3200
3000
2800
2600
2400
0
100
200
300
400

```
def create_LSTM(x_train, y_train, x_test, y_test):
  batch_size = None
  steps = 30
  input_dim = 8
  epochs = 20
  lstm = Sequential()
  # 加 RNN 隱藏層(hidden layer)
  lstm.add(LSTM(
      # 如果後端使用tensorflow，batch_input_shape 的 batch_size 需設為 None.
      # 否則執行 model.evaluate() 會有錯誤產生.
      batch_input_shape=(batch_size, steps, input_dim),
      units= 50,
      unroll=True,
  ))
  lstm.add(Dense(10,activation='relu'))
  lstm.add(Dense(1))
  lstm.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squar
  hist = lstm.fit(x_train, y_train,
            batch_size=batch_size, epochs=epochs,
            verbose=1, validation_data=(x_test, y_test))
  return hist, lstm
```

Problem 1.(vi i)

GRU cell


GRU loss
Train
Test
0.008
0.006
0.004
0.002
0.000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


GRU prediction
True
Predict
3200
3000
2800
2600
2400
0
100
200
300
400

```
def create_GRU(x_train, y_train, x_test, y_test):
  batch_size = None
  steps = 30
  input_dim = 8
  epochs = 20
  model = Sequential()
  # 加 RNN 隱藏層(hidden layer)
  model.add(GRU(
      # 如果後端使用tensorflow，batch_input_shape 的 batch_size 需設為 None.
      # 否則執行 model.evaluate() 會有錯誤產生.
      batch_input_shape=(batch_size, steps, input_dim),
      units= 50,
      unroll=True,
  ))
  model.add(Dense(10,activation='relu'))
  model.add(Dense(1))
  model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squa
  hist = model.fit(x_train, y_train,
            batch_size=batch_size, epochs=epochs,
            verbose=1, validation_data=(x_test, y_test))
  return hist, model
```

Problem 1.(viii)

上面三個model我設的參數跟設計都一樣，epochs 30次，第一層為RNN/LSTM/GRU的layer，接下來一層10 units的network，最後產生output。再這樣的設計之後可以看出loss跟predict的表現效果如下: RNN < LSTM < GRU。
但股價預測若沒有考量到時事新聞等等添加預測因子，常常預測出的結果會比true value慢半拍。(如上面RNN及LSTM prediction的結果)

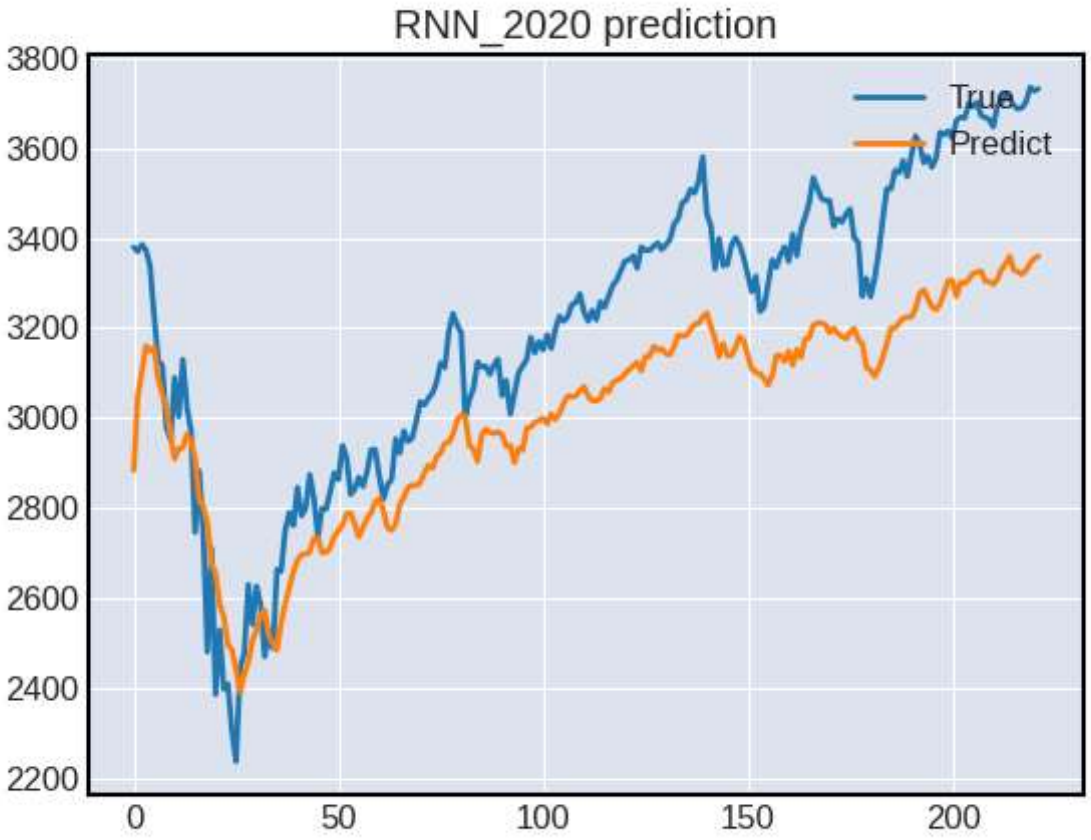
RNN雖然有考量的時間序列資料的前後關聯性，但對於長期時間性的資料來說效果不好，於是之後學者再提出LSTM來改善長期記憶的特性。
但LSTM也有執行速度慢、記憶體耗用高的缺點，雖然GRU改善了執行速度，但實務上該用哪個model還是需視情況而定。

在此S&P500的例子中，GRU prediction 的結果是相當接近true value的，整體來看GRU效果比LSTM好上不少。

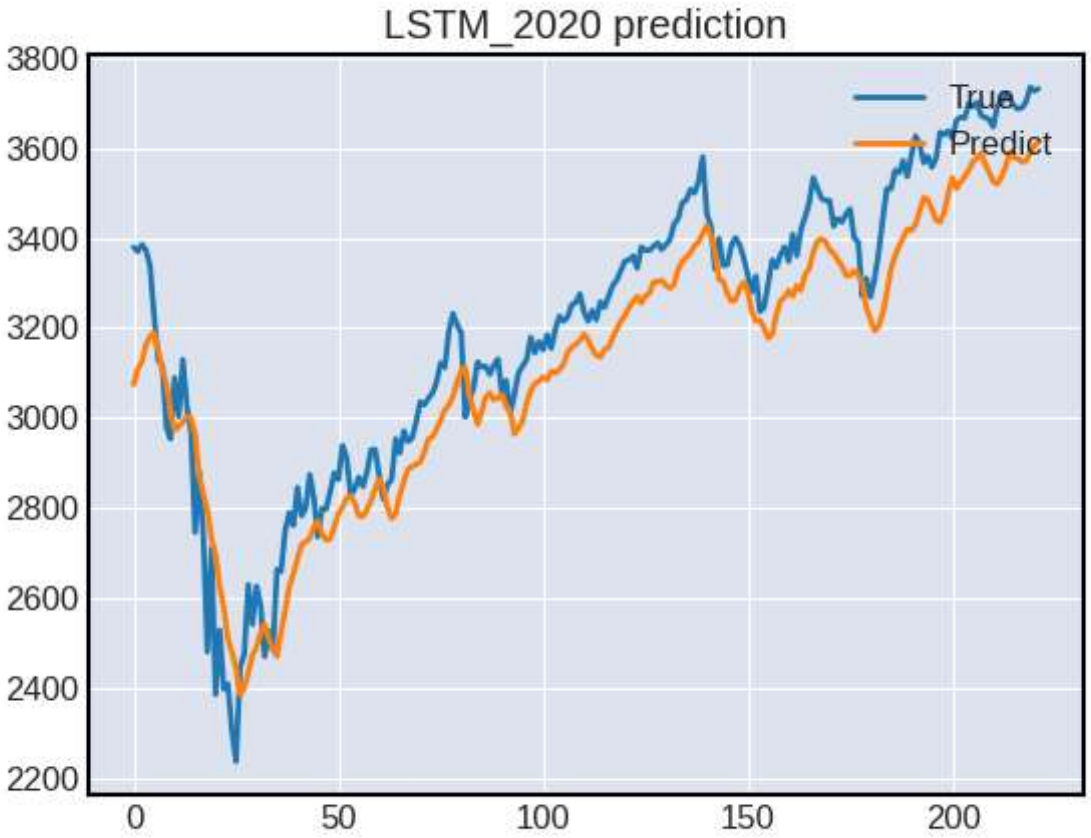
Problem 1.(ix)

Test on 2020 data, performance under COVID-19 condition

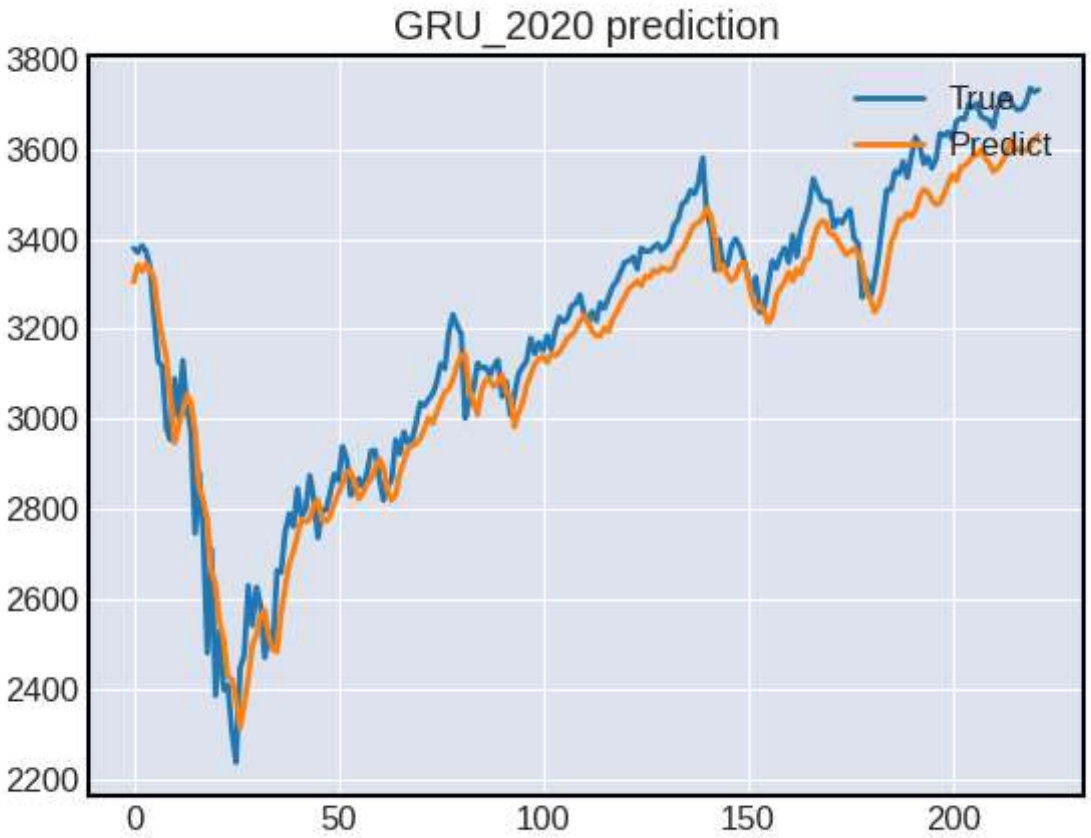
RNN


RNN_2020 prediction
True
Predict
3800
3600
3400
3200
3000
2800
2600
2400
2200
0
50
100
150
200

LSTM


LSTM_2020 prediction
True
Predict
3800
3600
3400
3200
3000
2800
2600
2400
2200
0
50
100
150
200

GRU


GRU_2020 prediction
True
Predict
3800
3600
3400
3200
3000
2800
2600
2400
2200
0
50
100
150
200

從上面三張圖來看，依舊是RNN < LSTM < GRU。
在2020年武漢肺炎的情況下，2020年true value的波動性也比較大，預測出的predict value都會比原先的test data再不準一點。但即使在這樣的狀況下，還是可以看出GRU的預測結果是滿貼近真實值的，但也會有先前提到預測慢半拍的狀況，畢竟波動性越大的data就是越越難預測(因為這邊的model都只考慮的historical data，但股價經常會受到當下的時事新聞去坐反應，因此預測結果都會慢半拍)。

Problem 1.(x)

Improvement

我認為若要提高預測準確度，需考量時事社會新聞，用NLP的方式去分析文本對於該檔股票的漲跌影響，再結合此作業的historical Data進行綜合性的預測，但如此一來的困難點會在NLP的部分。例如下述的問題：

- 同一則新聞對不同股票的影響不同，因此訓練一個General的Model不合理，應該針對每支股票建Model？
- 單一股票的波動，多少比例受大盤、市場影響？多少比例是因為個股的利多、利空資訊？
- 是否可以量化某些新聞資訊，比如外資買賣、財報狀況？
- 一則新聞出現後，反應時間是多久？
- 哪些新聞句子、描述是對哪些股票有影響的？
- 資料的粒度，應該切成句子、段落、還是整篇新聞？
- 同一間公司在不同時間遇到同樣的事件，是否會造成股價同樣的反應？

那這些NLP的內容就不是這門課提及的範圍了...