INTELLIGENT INTEGRATION
OF ENTERPRISE

利用長短期記憶演算法建立股票預測模型

Group 2

徐紫芸　梁茲晴

楊雨澄　張佳琳

CONTENTS

Scenario/topic

data

model : data-preprocessing
architecture
training process

generalizability

discussion

01

主題介紹

01 5W1H


進退場時機
Who
股票市場
what
When
投資客
Where
多對多，6年預測6年
多對多，6年預測1個月
單點分析，6年預測1天
Why
How
財富穩定增值
LSTM建構模型

目的/PURPOSE

利用2012-2017資料預測2018的股價

02

資料

02 資料

APPLE歷史股價，資料下載於Yahoo奇摩股市，如下圖所示

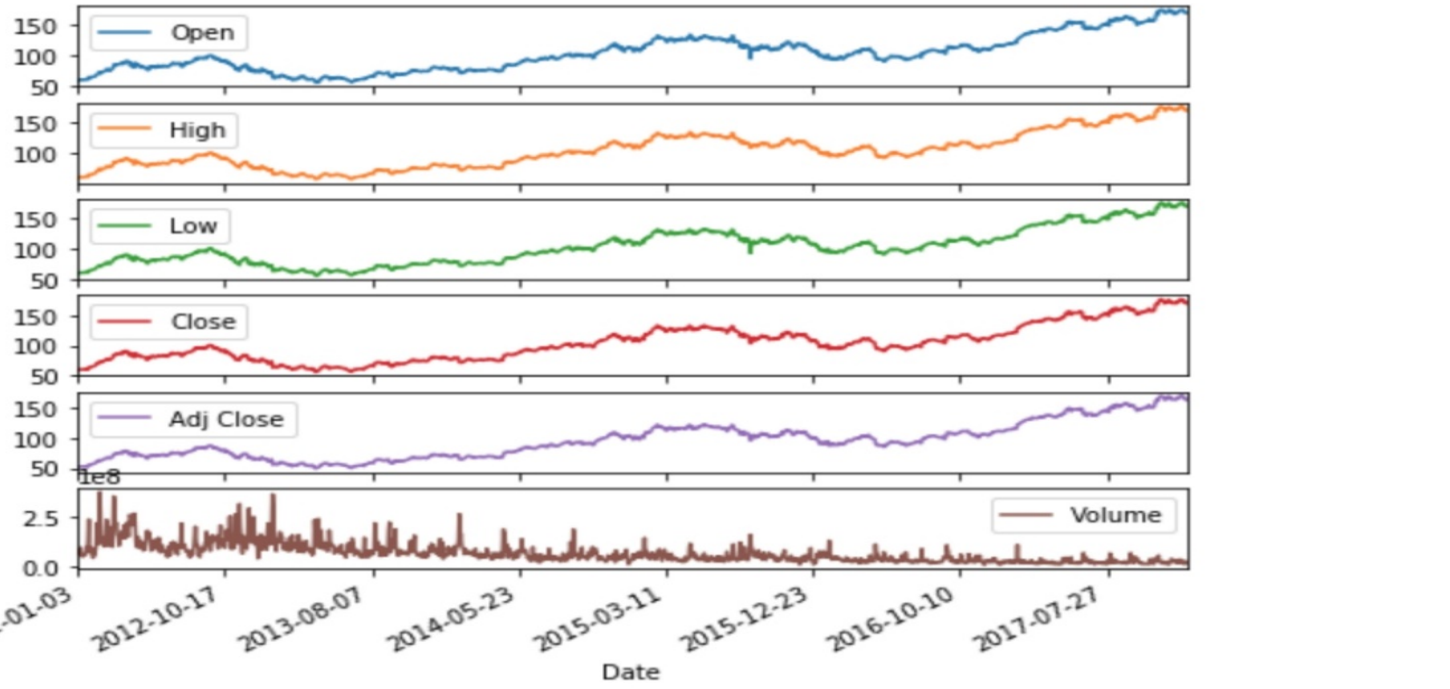
Date	Open	High	Low	Close	Adj Close	Volume
2012/1/3	58.48571	58.92857	58.42857	58.74714	51.11594	75555200
2012/1/4	58.57143	59.24	58.46857	59.06286	51.39065	65005500
2012/1/5	59.27857	59.79286	58.95286	59.71857	51.96119	67817400
2012/1/6	59.96714	60.39286	59.88857	60.34286	52.50438	79573200
2012/1/9	60.78571	61.10714	60.19286	60.24714	52.42109	98506100
2012/1/10	60.84428	60.85714	60.21429	60.46286	52.60879	64549100
2012/1/11	60.38286	60.40714	59.90143	60.36429	52.52303	53771200

02 資料

APPLE歷史股價，資料走勢圖，如下圖所示

```
features.plot(subplots=True)
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fdd9b4bc4e0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fdd9b403668>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fdd9b5e6f98>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fdd963ac438>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fdd96357898>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7fdd96389cf8>],
      dtype=object)
```



02 資料

- 黃金價，下載於「黃金價格資訊站」，如下圖所示

日期	最新	開市	高	低	成交量	更改%
2017年12月29日	1,362.60	1,362.60	1,362.60	1,362.60	0.02K	0.93%
2017年12月28日	1,350.10	1,346.00	1,346.40	1,343.30	0.13K	0.53%
2017年12月27日	1,343.00	1,341.00	1,341.00	1,341.00	0.00K	0.28%
2017年12月26日	1,339.20	1,337.00	1,337.90	1,337.00	0.00K	4.80%
2017年12月25日	1,277.90	1,278.90	1,279.50	1,277.90	-	-3.95%
2017年12月22日	1,330.50	1,322.00	1,329.10	1,322.00	0.00K	0.64%
2017年12月21日	1,322.00	1,319.50	1,319.50	1,319.50	0.03K	0.07%
2017年12月20日	1,321.10	1,317.90	1,321.80	1,317.90	0.02K	0.43%
2017年12月19日	1,315.50	1,315.50	1,315.50	1,315.50	0.01K	-0.06%

02 資料

油價，下載於「Investing.com」，如下圖所示

日期	最新	開市	高	低	成交量	更改%
2017年12月29日	60.42	59.91	60.51	59.82	464.48K	0.97%
2017年12月28日	59.84	59.53	59.94	59.44	345.96K	0.34%
2017年12月27日	59.64	59.79	59.93	59.33	402.40K	-0.55%
2017年12月26日	59.97	58.4	60.01	58.32	437.79K	2.36%
2017年12月25日	58.59	58.41	58.62	58.38	-	0.21%
2017年12月22日	58.47	58.21	58.5	57.87	324.55K	0.19%
2017年12月21日	58.36	58.02	58.38	57.63	405.32K	0.46%
2017年12月20日	58.09	57.66	58.12	57.44	420.84K	1.10%
2017年12月19日	57.46	57.3	57.64	57.16	36.29K	0.52%
2017年12月18日	57.16	57.37	57.78	56.82	137.15K	-0.24%

02 資料前處理

01 匯入套件

02 讀取2012-2017資料

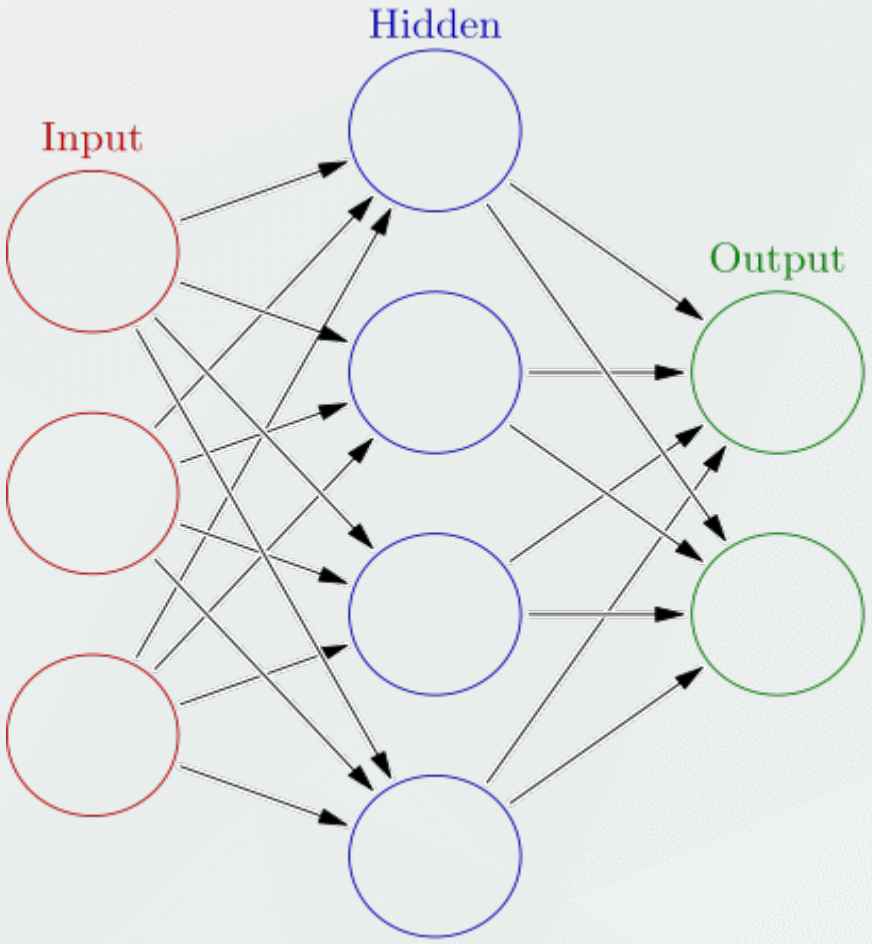
03 資料分組

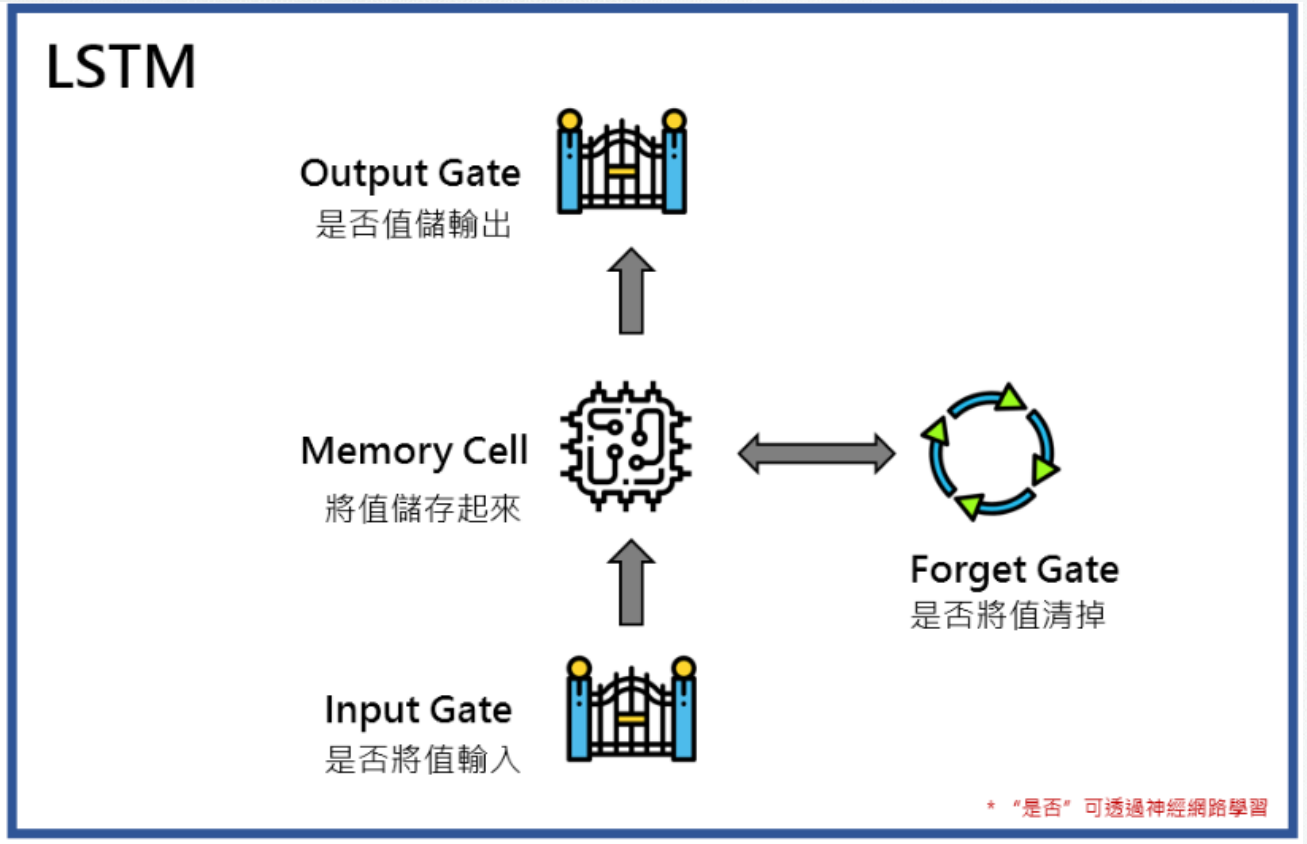
04 Normalization

05 Reshape Data


03
模型架構

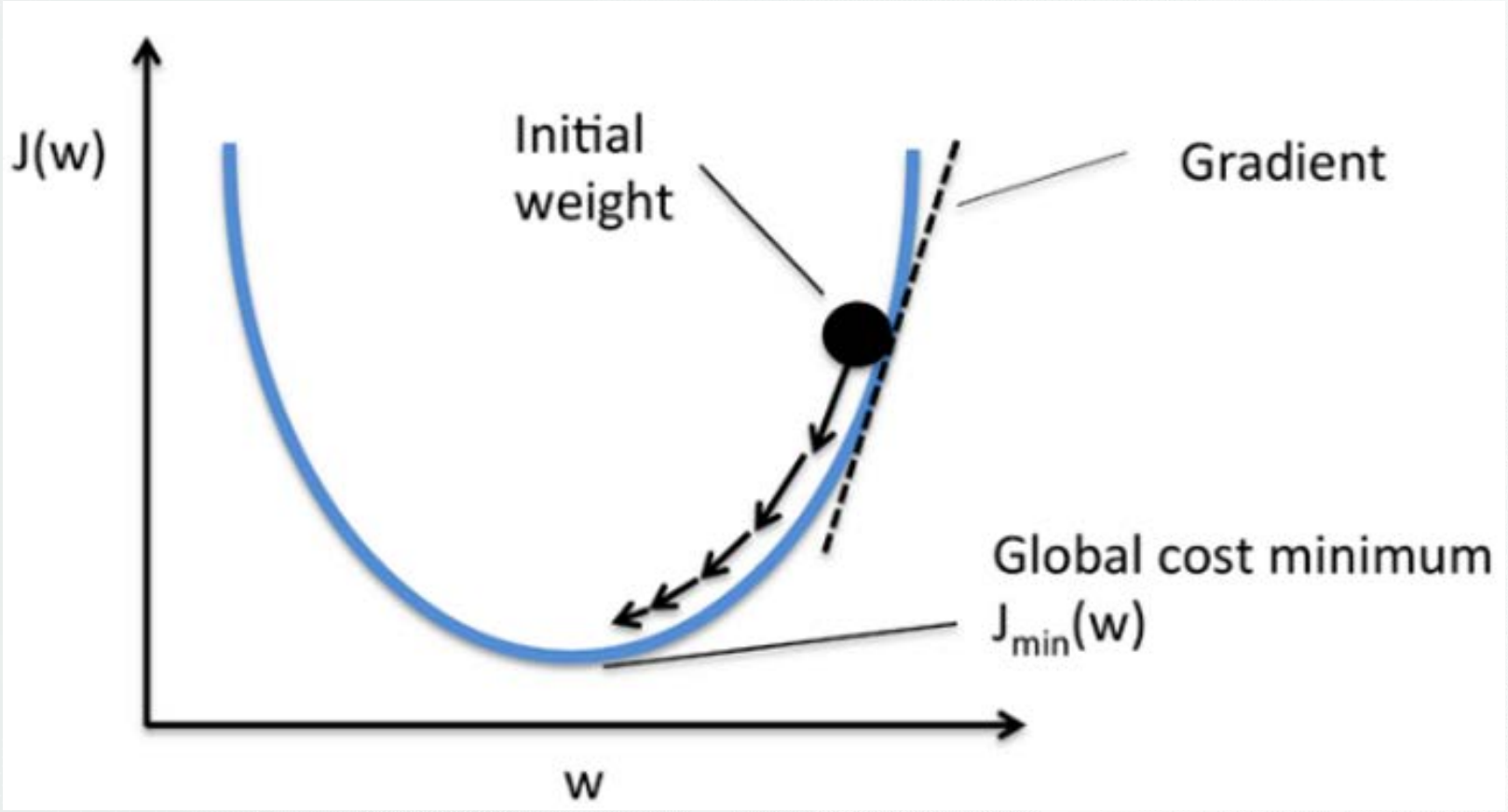
03 模型架構-RNN


Hidden
Input
Output


LSTM
Output Gate
是否值儲輸出
Memory Cell
將值儲存起來
Forget Gate
是否將值清掉
Input Gate
是否將值輸入
* "是否" 可透過神經網路學習

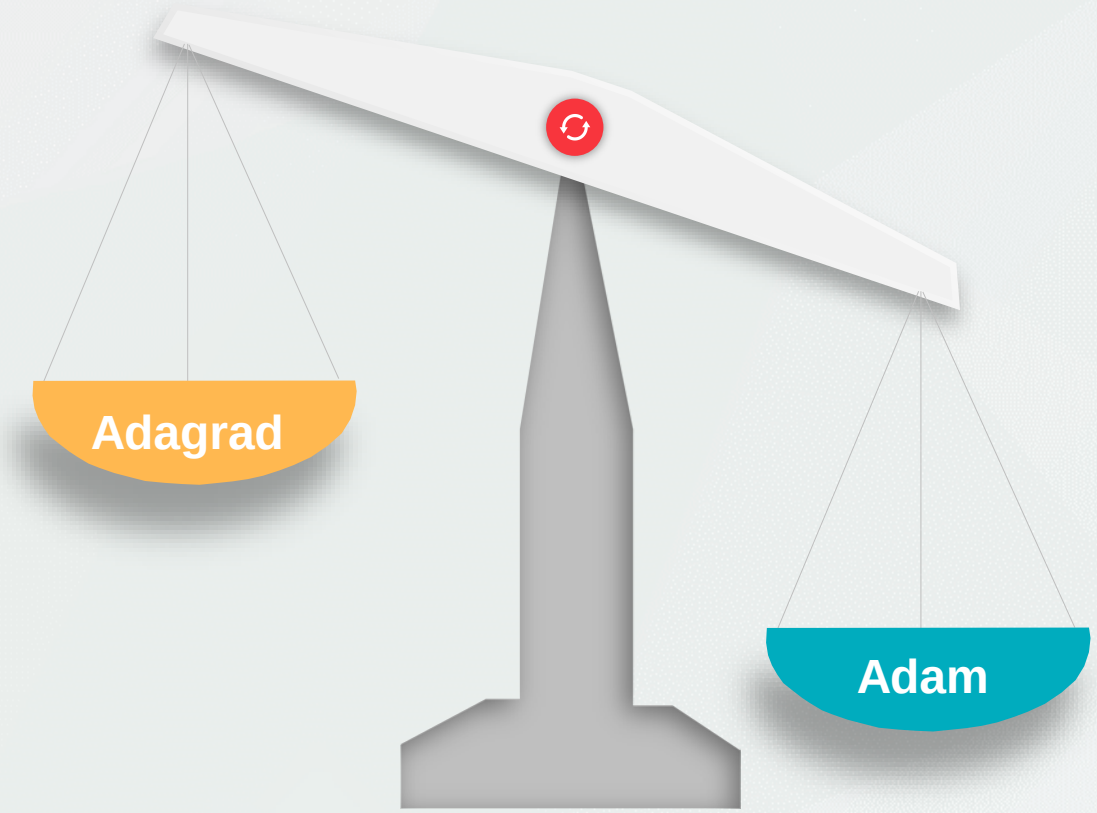
03 研究方法

長短期記憶模型 (Long Short-Term Memory , LSTM)



04 訓練過程

04 訓練過程-改善



Adam V.S Adagrad

為了讓學習率變佳，我們將optimizer由adagrad改成使用adam，它有做參數的「偏離校正」，使得每一次的學習率都會有個確定的範圍，會讓參數的更新較為平穩。

04 訓練過程

A

多對多，以6年預測6年

B

多對多，以6年預測1個月

多對多，以6年預測1年

C

單點分析，用6年預測1天-單變量

單點分析，用6年預測1天-多變量

04 訓練過程-程式碼

A 資料前處理

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
```

```
df = pd.read_csv("TSLA_2012-2017.csv")
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2012-01-03	28.940001	29.500000	27.650000	28.080000	28.080000	928100
1	2012-01-04	28.209999	28.670000	27.500000	27.709999	27.709999	630100
2	2012-01-05	27.760000	27.930000	26.850000	27.120001	27.120001	1005500
3	2012-01-06	27.200001	27.790001	26.410000	26.910000	26.910000	986300
4	2012-01-09	27.000000	27.490000	26.120001	27.250000	27.250000	897000
...	...	...	...	...	...	...	...
1504	2017-12-22	329.510010	330.920013	324.820007	325.200012	325.200012	4215800
1505	2017-12-26	323.829987	323.940002	316.579987	317.290009	317.290009	4378400
1506	2017-12-27	316.000000	317.679993	310.750000	311.640015	311.640015	4712100
1507	2017-12-28	311.750000	315.820007	309.540009	315.359985	315.359985	4316300
1508	2017-12-29	316.179993	316.410004	310.000000	311.350006	311.350006	3777200

1509 rows × 7 columns

```
[ ] values = df.iloc[:-1,1:-1] #取1508個而已
    values = values.values
    values
```

```
array([[ 28.940001,  29.5     ,  27.65    ,  28.08    ,  28.08    ],
       [ 28.209999,  28.67    ,  27.5     ,  27.709999,  27.709999],
       [ 27.76    ,  27.93    ,  26.85    ,  27.120001,  27.120001],
       ...,
       [323.829987, 323.940002, 316.579987, 317.290009, 317.290009],
       [316.      , 317.679993, 310.75    , 311.640015, 311.640015],
       [311.75    , 315.820007, 309.540009, 315.359985, 315.359985]])
```

```
[ ] #normalization 標準化
    values = (values - values.min()) / (values.max() - values.min())
```

```
[ ] values
```

```
array([[0.01716762, 0.01869363, 0.01365235, 0.0148241 , 0.0148241 ],
       [0.01517835, 0.01643186, 0.01324359, 0.01381584, 0.01381584],
       [0.0139521 , 0.01441535, 0.01147233, 0.01220809, 0.01220809],
       ...,
       [0.82074829, 0.82104808, 0.8009919 , 0.80292673, 0.80292673],
       [0.79941143, 0.80398944, 0.78510508, 0.78753039, 0.78753039],
       [0.7878301 , 0.79892094, 0.78180783, 0.79766738, 0.79766738]])
```

```
## data cleaning
xraw = []
yraw = []
for t in range(116):
  for i in range(13):
    if(i < 10):
      xraw.append(values[t + i])
    else:
      yraw.append(values[t + i])
```

```
[ ] xraw = np.array(xraw)
    yraw = np.array(yraw)

    xraw = xraw.reshape(116, 5, 10)
    yraw = yraw.reshape(116, 5, 3)

    print(xraw.shape, yraw.shape)
```

```
(116, 5, 10) (116, 5, 3)
```

04 訓練過程-程式碼

A 模型架構

```
[ ] from sklearn.model_selection import train_test_split #分測試跟訓練資料
    from keras.models import Sequential
    from keras.layers import Dense, LSTM, Dropout, Flatten, TimeDistributed
    from keras.optimizers import adam
    from keras.callbacks import ModelCheckpoint
    from keras.losses import mean_absolute_percentage_error
```

```
[ ] #作train跟test的切割
    #xraw
    #test_size0.2 trainingdata占0.8 116*0.8是training data
    #input跟output都分train跟test
    X_train, X_test, Y_train, Y_test = train_test_split(xraw, yraw, test_size = 0.2, random_state = 10)
```

```
#loss 選MAE 跟 MSE（直很小） MAE值比較大比較好看

model.compile(loss = mean_absolute_percentage_error,
              optimizer = adam(lr = 0.001),
             )
```

```
[ ] #checkpoint目的是訓練模型的過程中會有時候有高有低, 目的是存最佳模型
    #用validation loss , 一直存最低的val_loss的model

    checkpoint = ModelCheckpoint('to3.h5', monitor='val_loss', verbose=1, save_best_only=True,mode='min')
    callbacks_list = [checkpoint]

    k = model.fit(X_train ,Y_train, batch_size = 100, epochs = 500,
                  callbacks=[checkpoint],
            verbose = 1,
            validation_data = (X_test, Y_test))

    #MAE百分比轉換
```

```
[ ] model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 5, 100)            44400
_________________________________________________________________
lstm_8 (LSTM)                (None, 5, 80)             57920
_________________________________________________________________
lstm_9 (LSTM)                (None, 5, 50)             26200
_________________________________________________________________
dropout_3 (Dropout)          (None, 5, 50)             0
_________________________________________________________________
time_distributed_5 (TimeDist (None, 5, 50)             2550
_________________________________________________________________
time_distributed_6 (TimeDist (None, 5, 3)              153
=================================================================
Total params: 131,223
Trainable params: 131,223
Non-trainable params: 0
_________________________________________________________________
```

04 訓練過程-程式碼

A 訓練過程

```
#loss 選MAE 跟 MSE（直很小） MAE值比較大比較好看

model.compile(loss = mean_absolute_percentage_error,
              optimizer = adam(lr = 0.001),
             )
```

```
[ ] #checkpoint目的是訓練模型的過程中會有時候有高有低，目的是存最佳模型
    #用validation loss , 一直存最低的val_loss的model

    checkpoint = ModelCheckpoint('to3.h5', monitor='val_loss', verbose=1, save_best_only=True,mode='min')
    callbacks_list = [checkpoint]

    k = model.fit(X_train ,Y_train, batch_size = 100, epochs = 500,
                  callbacks=[checkpoint],
            verbose = 1,
            validation_data = (X_test, Y_test))

    #MAE百分比轉換
```

```
[ ] Epoch 495/500
92/92 [==============================] - 0s 445us/step - loss: 15.0185 - val_loss: 20.5819

Epoch 00495: val_loss did not improve from 20.44295
Epoch 496/500
92/92 [==============================] - 0s 432us/step - loss: 15.1188 - val_loss: 21.0937

Epoch 00496: val_loss did not improve from 20.44295
Epoch 497/500
92/92 [==============================] - 0s 435us/step - loss: 15.0165 - val_loss: 21.4229

Epoch 00497: val_loss did not improve from 20.44295
Epoch 498/500
92/92 [==============================] - 0s 423us/step - loss: 15.0327 - val_loss: 20.7326

Epoch 00498: val_loss did not improve from 20.44295
Epoch 499/500
92/92 [==============================] - 0s 488us/step - loss: 14.9136 - val_loss: 21.0083

Epoch 00499: val_loss did not improve from 20.44295
Epoch 500/500
92/92 [==============================] - 0s 433us/step - loss: 14.9652 - val_loss: 21.2540

Epoch 00500: val_loss did not improve from 20.44295
```

04 訓練過程

B 資料前處理

```
[ ] #import the libraries
    import numpy as np
    import matplotlib.pyplot as plt #for drawing
    import pandas as pd
```

```
[ ] #import the training set
    dataset_train = pd.read_csv('AAPL_2012-2017.csv') #read training data
    training_set = dataset_train.iloc[:,1:2].values #reach open
    dataset_train.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2012-01-03	58.485714	58.928570	58.428570	58.747143	51.115936	75555200
1	2012-01-04	58.571430	59.240002	58.468571	59.062859	51.390648	65005500
2	2012-01-05	59.278572	59.792858	58.952858	59.718571	51.961189	67817400
3	2012-01-06	59.967144	60.392857	59.888573	60.342857	52.504375	79573200
4	2012-01-09	60.785713	61.107143	60.192856	60.247143	52.421093	98506100

```
[ ] #Feature Scaling 限定到0-1
    from sklearn import preprocessing
    from sklearn.preprocessing import MinMaxScaler
    sc = MinMaxScaler(feature_range = (0,1))
    training_set_scaled = sc.fit_transform(training_set)
```

```
[ ] X_train = [] #預測點的前60天資料
    y_train = [] #預測點
    for i in range(60,1509): #1509是訓練集總數
        X_train.append(training_set_scaled[i-60:i,0])
        y_train.append(training_set_scaled[i,0])
    X_train, y_train = np.array(X_train), np.array(y_train) #轉成numpy array的格式，以利輸入RNN
```

```
[ ] X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))
```

```
[ ] X_train = [] #預測點的前60天資料
    y_train = [] #預測點
    for i in range(60,1509): #1509是訓練集總數
        X_train.append(training_set_scaled[i-60:i,0])
        y_train.append(training_set_scaled[i,0])
    X_train, y_train = np.array(X_train), np.array(y_train) #轉成numpy array的格式，以利輸入RNN
```

```
[ ] X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))
```

04 訓練過程

B 模型架構

```
[ ]  #import the Keras libraries and packages
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.layers import LSTM
     from keras.layers import Dropout
     from keras.optimizers import adam
     from keras.callbacks import ModelCheckpoint
     from keras.losses import mean_absolute_percentage_error

     #initialising the RNN
     regressor = Sequential()
```

```
     # Adding the first LSTM layer and some Droupout regrlarisation
     regressor.add(LSTM(units = 100, return_sequences = True, input_shape = (X_train.shape[1],1)))
     regressor.add(Dropout(0.3))

     #Adding a second LSTM layer and some Droupout regularisation
     regressor.add(LSTM(units = 100, return_sequences = True))
     regressor.add(Dropout(0.3))

     #Adding a third LSTM layer and some Droupout regularisation
     regressor.add(LSTM(units = 100, return_sequences = True))
     regressor.add(Dropout(0.3))

     #Adding a fourth LSTM layer and some Droupout regularisation
     regressor.add(LSTM(units = 100))
     regressor.add(Dropout(0.3))
```

```
[ ]  regressor.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 60, 100)           40800
_________________________________________________________________
dropout_5 (Dropout)          (None, 60, 100)           0
_________________________________________________________________
lstm_6 (LSTM)                (None, 60, 100)           80400
_________________________________________________________________
dropout_6 (Dropout)          (None, 60, 100)           0
_________________________________________________________________
lstm_7 (LSTM)                (None, 60, 100)           80400
_________________________________________________________________
dropout_7 (Dropout)          (None, 60, 100)           0
_________________________________________________________________
lstm_8 (LSTM)                (None, 100)               80400
_________________________________________________________________
dropout_8 (Dropout)          (None, 100)               0
=================================================================
Total params: 282,000
Trainable params: 282,000
Non-trainable params: 0
_________________________________________________________________
```

04 訓練過程

B 訓練過程

1個月

```
[ ]  #Adding the output layer
     #output layer units 設為1
     regressor.add(Dense(units = 1))
```

```
[ ]  #compiling
     regressor.compile(optimizer='adam', loss = 'mae')

     #train
     history=regressor.fit(X_train, y_train, epochs = 50, batch_size = 32)
```

```
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0195
Epoch 40/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0196
Epoch 41/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0214
Epoch 42/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0211
Epoch 43/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0199
Epoch 44/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0203
Epoch 45/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0203
Epoch 46/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0201
Epoch 47/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0214
Epoch 48/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0208
Epoch 49/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0214
Epoch 50/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0191
```

1年

```
[ ]  #Adding the output layer
     #output layer units 設為1
     regressor.add(Dense(units = 1))
```

```
[ ]  #compiling
     regressor.compile(optimizer='adam', loss = 'mae')

     #train
     history=regressor.fit(X_train, y_train, epochs = 50, batch_size = 32)
```

```
Epoch 42/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0260
Epoch 43/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0273
Epoch 44/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0263
Epoch 45/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0280
Epoch 46/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0263
Epoch 47/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0257
Epoch 48/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0263
Epoch 49/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0253
Epoch 50/50
1449/1449 [==============================] - 13s 9ms/step - loss: 0.0253
```

04 訓練過程

C 資料前處理

```
from __future__ import absolute_import, division, print_function, unicode_literals
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf

import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False
```

```
[7] df = pd.read_csv('AAPL_2012-2017.csv')
```

```
[8] df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2012-01-03	58.485714	58.928570	58.428570	58.747143	51.115936	75555200
1	2012-01-04	58.571430	59.240002	58.468571	59.062859	51.390648	65005500
2	2012-01-05	59.278572	59.792858	58.952858	59.718571	51.961189	67817400
3	2012-01-06	59.967144	60.392857	59.888573	60.342857	52.504375	79573200
4	2012-01-09	60.785713	61.107143	60.192856	60.247143	52.421093	98506100

```
[9] df.shape
```

```
(1509, 7)
```

```
[10] def univariate_data(dataset, start_index, end_index, history_size, target_size):
       data = []
       labels = []

       start_index = start_index + history_size
       if end_index is None:
         end_index = len(dataset) - target_size

       for i in range(start_index, end_index):
         indices = range(i-history_size, i)
         # Reshape data from (history_size,) to (history_size, 1)
         data.append(np.reshape(dataset[indices], (history_size, 1)))
         labels.append(dataset[i+target_size])
       return np.array(data), np.array(labels)
```

```
[11] TRAIN_SPLIT = 755
     print(TRAIN_SPLIT)
```

```
uni_data = df['Close']
uni_data.index = df['Date']
uni_data.head()
```

```
Date
2012-01-03    58.747143
2012-01-04    59.062859
2012-01-05    59.718571
2012-01-06    60.342857
2012-01-09    60.247143
Name: Close, dtype: float64
```

04 訓練過程

模型架構

```
[15] univariate_past_history = 60
     univariate_future_target = 0

     x_train_uni, y_train_uni = univariate_data(uni_data, 0, TRAIN_SPLIT,
                                                univariate_past_history,
                                                univariate_future_target)
     x_val_uni, y_val_uni = univariate_data(uni_data, TRAIN_SPLIT, None,
                                            univariate_past_history,
                                            univariate_future_target)
```

```
[16] print ('Single window of past history')
     print (x_train_uni[0])
     print ('\n Target stock price to predict')
     print (y_train_uni[0])
```

04 訓練過程

訓練過程

```
[26] simple_lstm_model = tf.keras.models.Sequential([
         tf.keras.layers.LSTM(8, input_shape=x_train_uni.shape[-2:]),
         tf.keras.layers.Dense(1)
     ])

     tf.keras.optimizers.Adam(lr=0.01)
     simple_lstm_model.compile(optimizer ='adam', loss='mae')
```

```
[27] for x, y in val_univariate.take(1):
         print(simple_lstm_model.predict(x).shape)
```

```
(256, 1)
```

```
EVALUATION_INTERVAL = 200
EPOCHS = 50

simple_lstm_model.fit(train_univariate, epochs=EPOCHS,
                      steps_per_epoch=EVALUATION_INTERVAL,
                      validation_data=val_univariate, validation_steps=50)
```

```
Epoch 45/50
200/200 [==============================] - 2s 8ms/step - loss: 0.0646 - val_loss: 0.5584
Epoch 46/50
200/200 [==============================] - 2s 9ms/step - loss: 0.0646 - val_loss: 0.5594
Epoch 47/50
200/200 [==============================] - 2s 8ms/step - loss: 0.0646 - val_loss: 0.5597
Epoch 48/50
200/200 [==============================] - 2s 8ms/step - loss: 0.0645 - val_loss: 0.5582
Epoch 49/50
200/200 [==============================] - 2s 8ms/step - loss: 0.0646 - val_loss: 0.5586
Epoch 50/50
200/200 [==============================] - 2s 8ms/step - loss: 0.0645 - val_loss: 0.5596
<tensorflow.python.keras.callbacks.History at 0x7fa6d90904e0>
```

05

訓練結果

05 訓練結果

A

多對多，以6年預測6年

B

多對多，以6年預測1個月

多對多，以6年預測1年

C

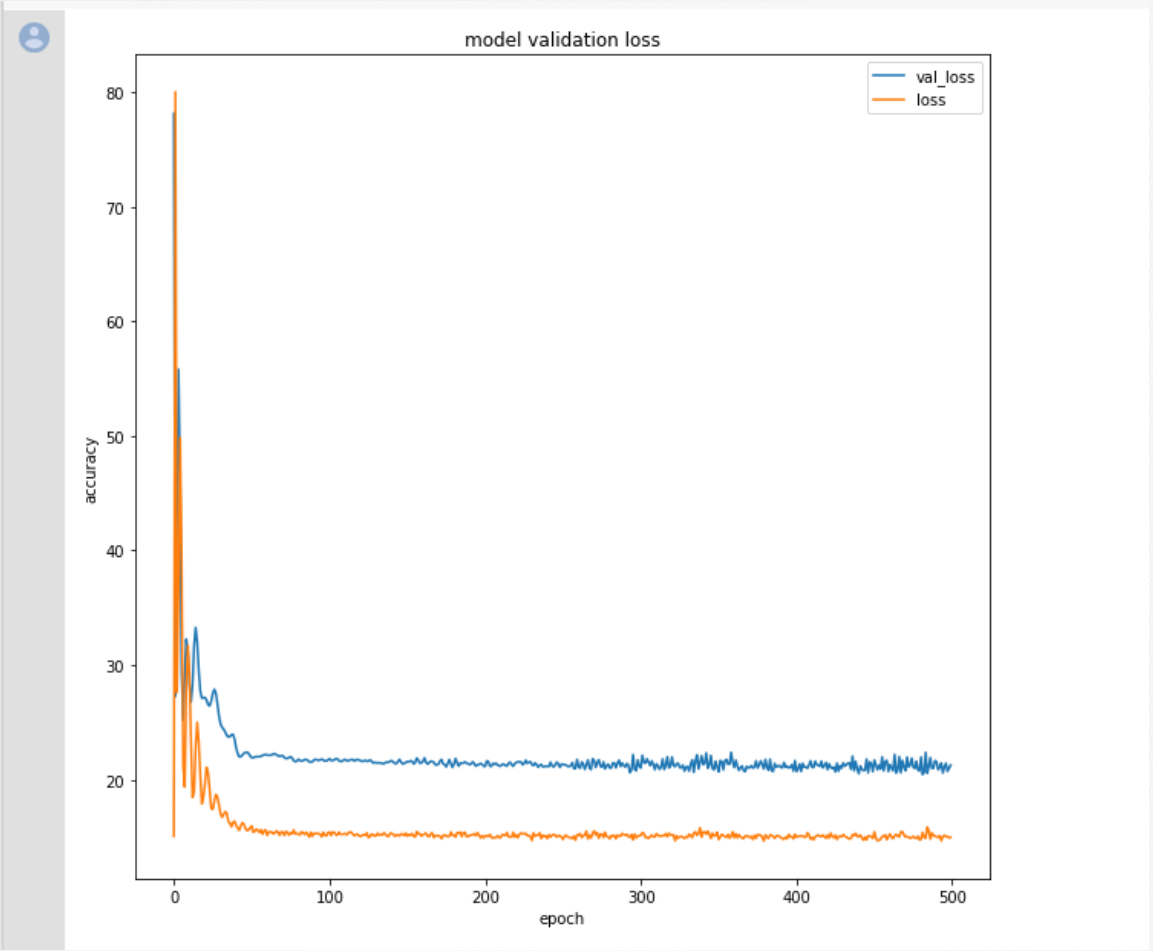
單點分析，用6年預測1天-單變量

單點分析，用6年預測1天-多變量

05 訓練結果

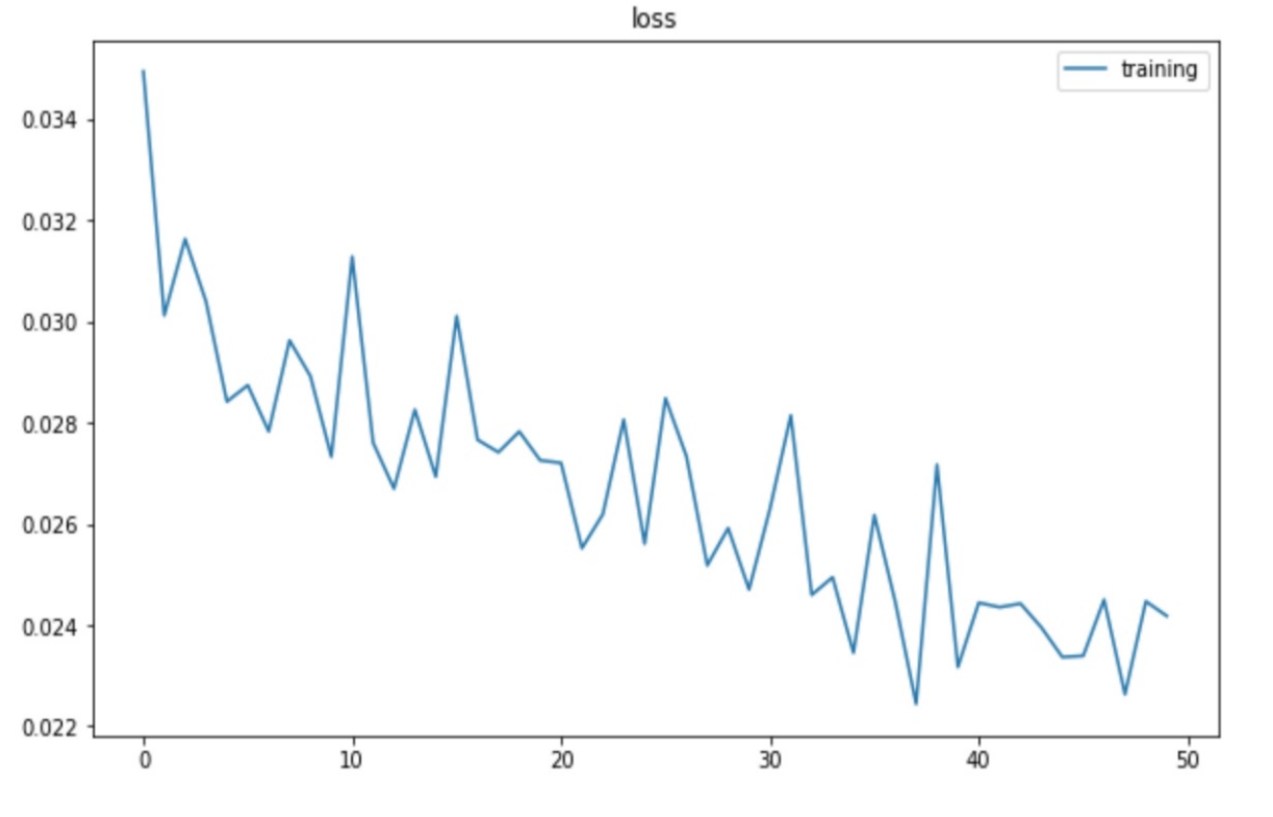
A 多對多，以6年預測6年

```
[ ]  plt.figure(figsize=(10, 10))
     plt.plot(k.history["val_loss"])
     plt.plot(k.history["loss"])

     plt.title('model validation loss')
     plt.ylabel("accuracy")
     plt.xlabel('epoch')
     plt.legend(['val_loss' , 'loss'], loc = 'best')
     plt.show()
```



05 訓練結果

A 多對多，以6年預測6年



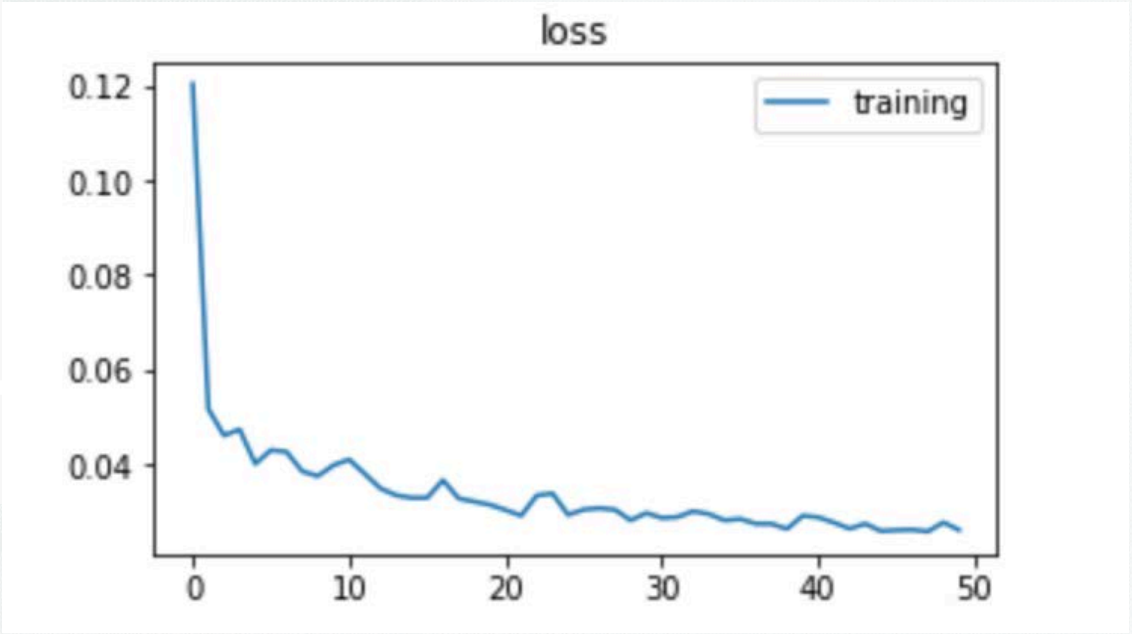
05 訓練結果

B 多對多，以6年預測1個月

```
[ ]  dataset_total = pd.concat((dataset_train['Open'],dataset_test['Open']), axis = 0)
     inputs = dataset_total[len(dataset_total)-len(dataset_test)-60:].values
     inputs = inputs.reshape(-1,1)
     inputs = sc.transform(inputs) #feature scaling

     X_test = []
     for i in range (60, 81): #timesteps一樣60, 81等於先前的60天+2018的未來21天資料
       X_test.append(inputs[i-60:i,0])
     X_test = np.array(X_test)
     X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1)) #reshape 成3-dimension
```

```
[ ]  predicted_stock_price = regressor.predict(X_test)
     predicted_stock_price = sc.inverse_transform(predicted_stock_price) #to get the original scale
```

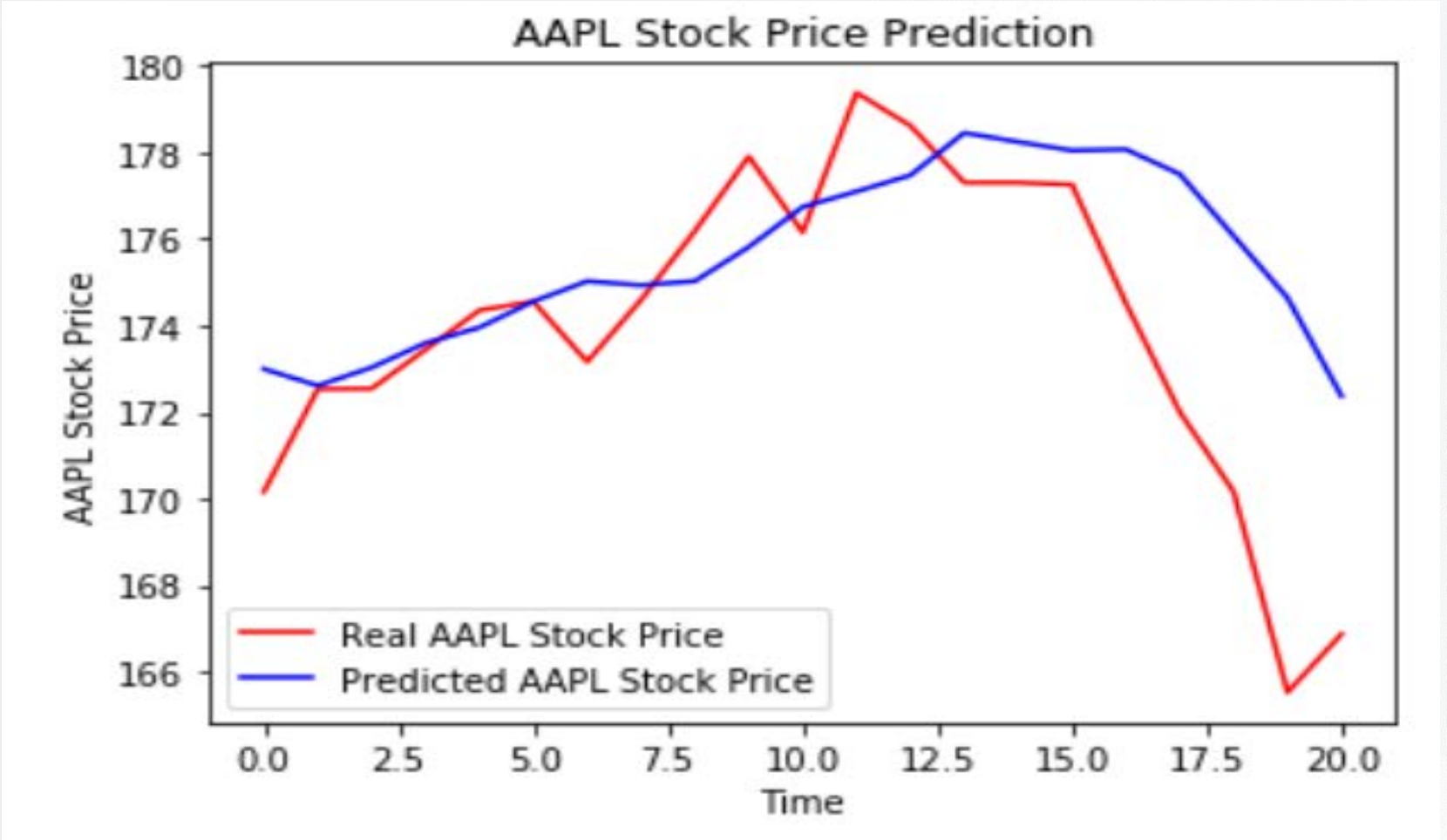
```
[ ] #visualising the results
    plt.plot(real_stock_price, color = 'red', label='Real AAPL Stock Price') #red line show the real stock price
    plt.plot(predicted_stock_price, color = 'blue', label='Predicted AAPL Stock Price') #blue line show the predict stock price

    plt.title('AAPL Stock Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('AAPL Stock Price')
    plt.legend()
    plt.show()
```



05 訓練結果

B

多對多，以6年預測1個月



05 訓練結果

B 多對多，以6年預測1個月

誤差百分比

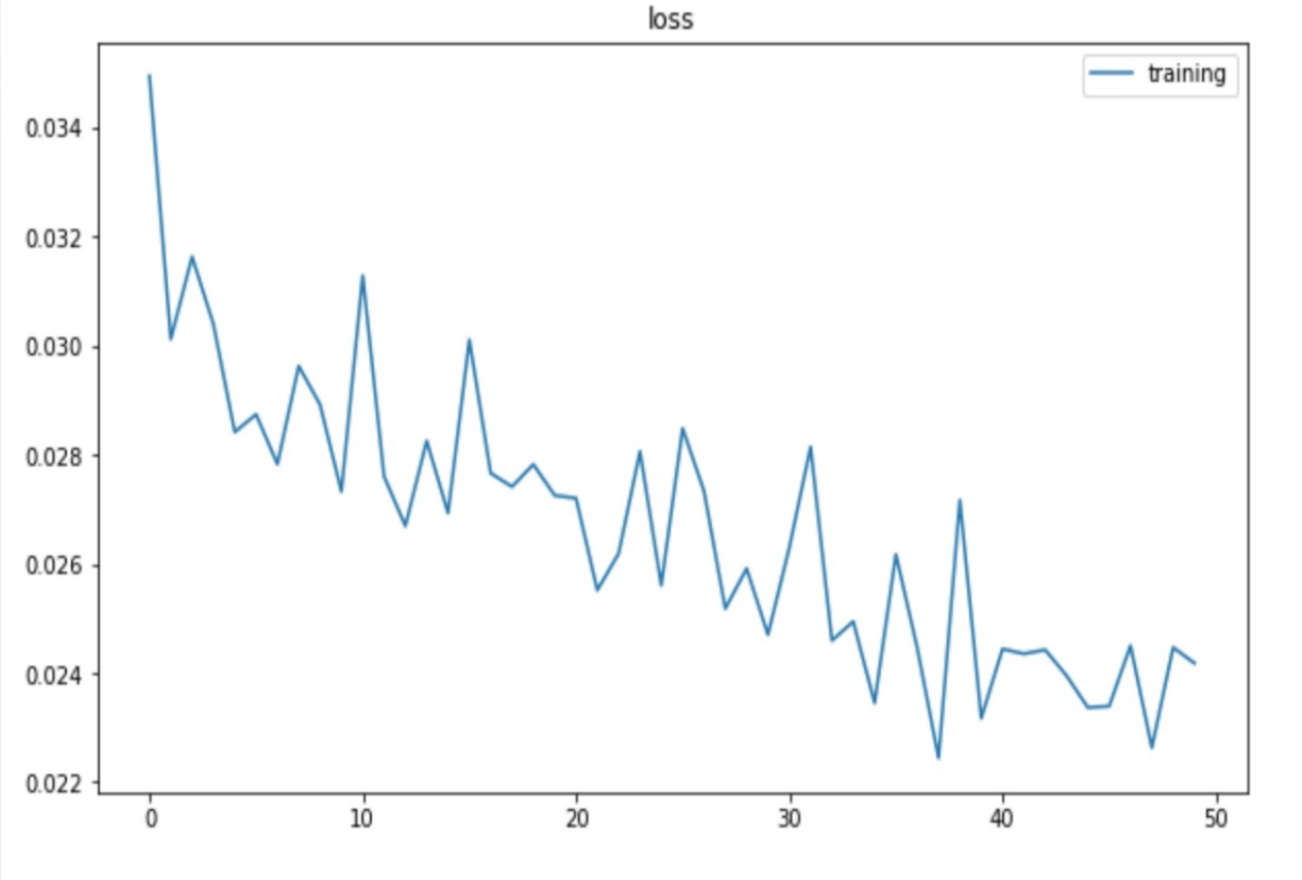
```
a = ((predicted_stock_price)-(real_stock_price))/(real_stock_price)

ACC = np.mean(a, axis=0)

print (abs(ACC))
```

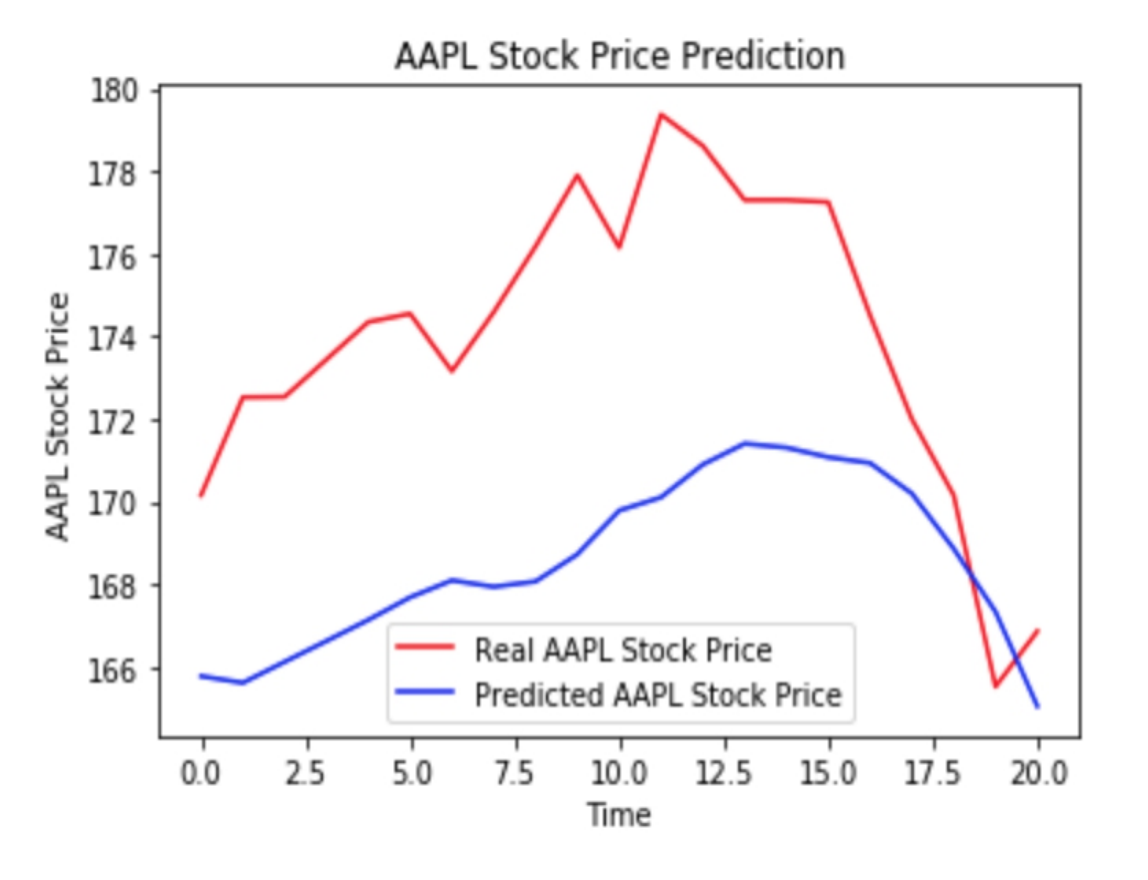
```
[0.00476125]
```

05 訓練結果-金價影響

B

多對多，以6年預測1個月





05 訓練結果

B 多對多，以6年預測1年

```
[ ] #get 201801 test data
    dataset_test = pd.read_csv('AAPL_2018-2019.csv')
    real_stock_price = dataset_test.iloc[:, 1:2].values
    dataset_test.shape
```

```
(251, 7)
```

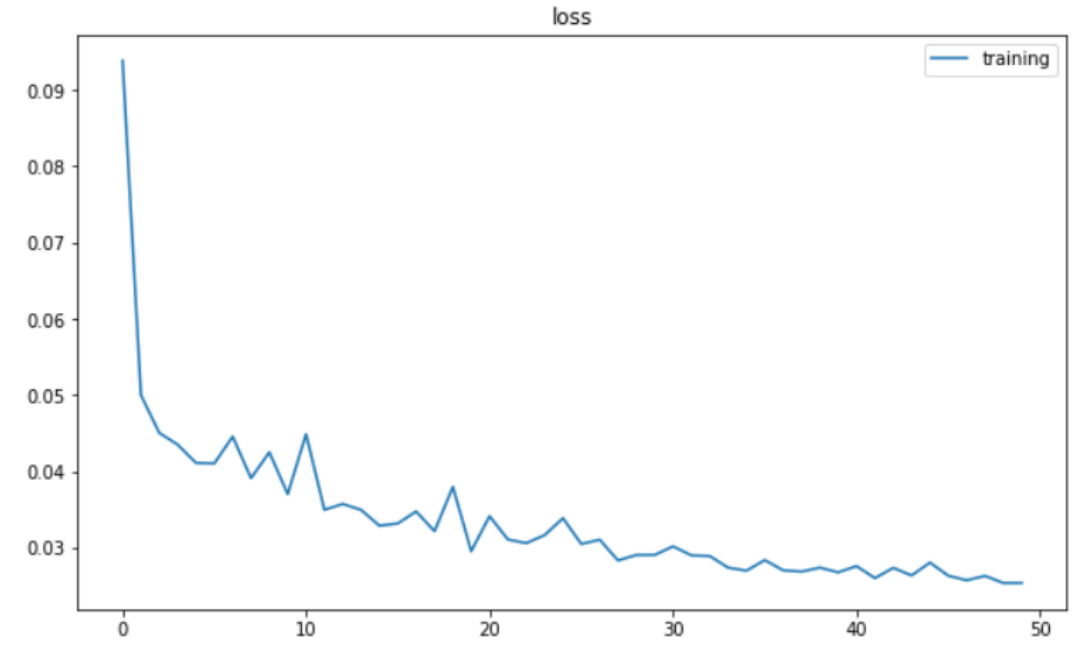
```
[ ] dataset_total = pd.concat((dataset_train['Open'],dataset_test['Open']), axis = 0)
    inputs = dataset_total[len(dataset_total)-len(dataset_test)-60:].values
    inputs = inputs.reshape(-1,1)
    inputs = sc.transform(inputs) #feature scaling

    X_test = []
    for i in range (60, 311): #timesteps一樣60, 331等於先前的60天+2018的251天資料
      X_test.append(inputs[i-60:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1)) #reshape 成3-dimension
```

```
[ ] predicted_stock_price = regressor.predict(X_test)
    predicted_stock_price = sc.inverse_transform(predicted_stock_price) #to get the original scale
```

```
[ ] #visualising the results
    plt.plot(real_stock_price, color = 'red', label='Real AAPL Stock Price') #red line show the real stock price
    plt.plot(predicted_stock_price, color = 'blue', label='Predicted AAPL Stock Price') #blue line show the predict stock price

    plt.title('AAPL Stock Price Prediction')
    plt.xlabel('Time')
    plt.ylabel('AAPL Stock Price')
    plt.legend()
    plt.show()
```

```
[ ] plt.figure(figsize=(10,6))
    plt.plot(history.epoch,history.history['loss'], label='training')
    plt.title('loss')
    plt.legend(loc='best')
```

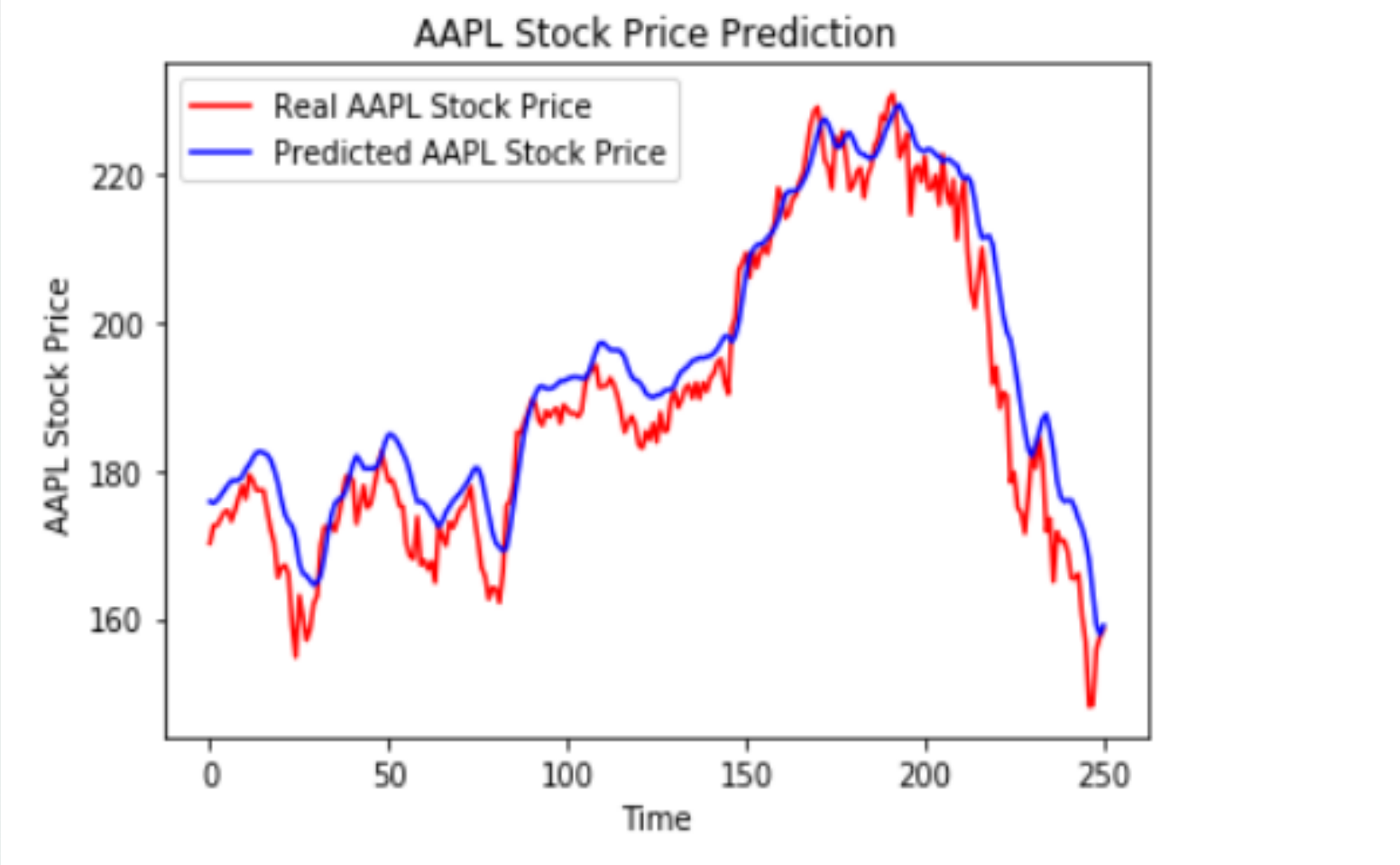
```
<matplotlib.legend.Legend at 0x7fdba935af60>
```



05 訓練結果

B

多對多，以6年預測1年



05 訓練結果

B 多對多，以6年預測1年

誤差百分比

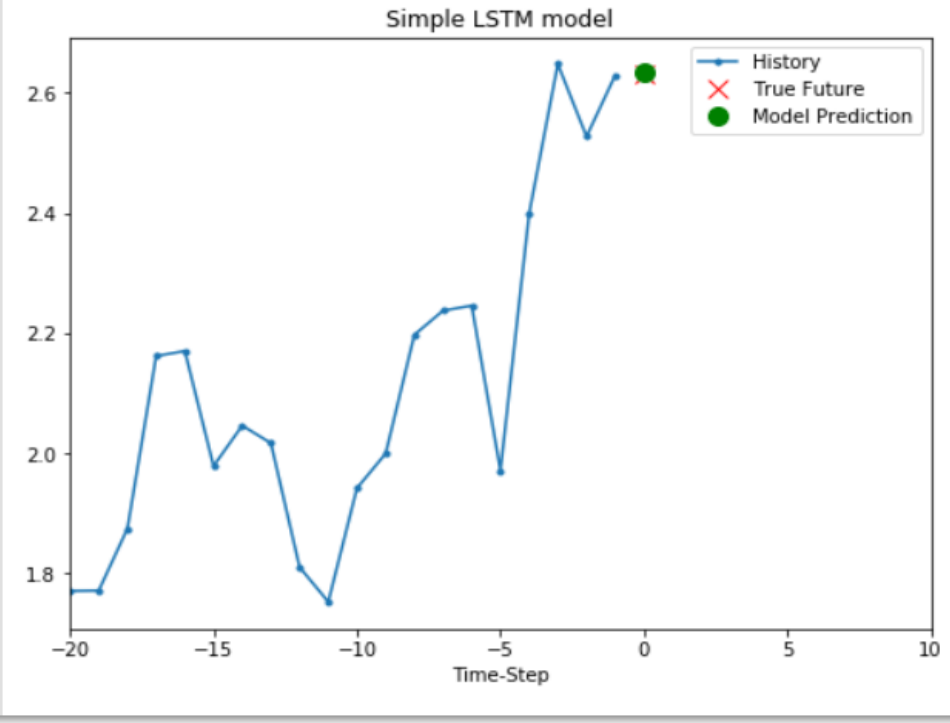
```
a = ((predicted_stock_price)-(real_stock_price))/(real_stock_price)

ACC = np.mean(a, axis=0)

print (abs(ACC))
```

```
[0.00040419]
```

05 訓練結果

單點分析，用6年預測1天-單變量

```
for x, y in val_univariate.take(1):
  plot = show_plot([x[0].numpy(), y[0].numpy(),
                    simple_lstm_model.predict(x)[0]], 0, 'Simple LSTM model')
  plot.show()
```



06

結果與討論

06 結論與未來展望

01

建構三種模型相輔相成

第一種模型可用於長期趨勢預測；第二種模型屬於預測未來的股價走勢，用於判斷股市上漲或下跌；最後一種用於單一天的股價預測，可以用來產出當天的預測值。

02

加入金價及油價做為參考

金價對於此股的影響較小，甚至有誤判得情形，負面影響較大

03

未來於模型中導入油價多變量分析長期模型

而油價的模型對與此股相較起來有正向影響，因此期望在未來與模型中導入多變量分析長期模型，以得到更精確的預測結果。

The End!