201901001 김민기 실습과제12 보고서

1. 문제

▪ 주어진 학습 데이터를 data-02-stock_daily.csv를 불러오고 전처리를 수행한 다음 RNN 모델을 학습한다.

1. 아래 RNN 모델 구조를 따라 훈련데이터를 학습하고 수행한다.

✓ SimpleRNN 한 층인 모델.

✓ input_shape=[7, 5]

✓ n_units = 256, epochs=30, optimizer, loss는 자유

✓ return_sequences=False

✓ 출력 층은 노드가 1개인 Dense 층으로 구성.

- 학습된 모델을 통해 테스트 데이터에 대해 예측값과 실제값을 plot으로 나타냄.

2. (1)의 모델 구조에서 다음의 내용으로 변경하여 훈련한 후 (1)과 같이 plot으로 나타냄.

✓ SimpleRNN을 LSTM 모델로 변경.

3. (1)의 모델 구조에서 다음의 내용으로 변경하여 훈련한 후 (1)과 같이 plot으로 나타냄.

✓ 다층 구조 SimpleRNN을 학습.

✓ n_units = 34을 가진 4개의 층을 구성.

✓ 네 번째를 제외한 층을 return_sequences=True으로 설정.

4. 위 과제를 수행한 코드와 plot을 캡쳐하여 보고서 작성.

2. 코드 및 결과 화면

MinMaxScaler 함수 정의

```
19]  import numpy as np

     def MinMaxScaler(data) :
       numerator = data - np.min(data, 0)
       denominator = np.max(data, 0) - np.min(data, 0)
       return numerator / (denominator + 1e-7)
```

데이터 전처리

```
]  timesteps = seq_length = 7
   data_dim = 5
   output_dim = 1

   xy = np.loadtxt('/content/drive/MyDrive/ML_Lab/report/12/data-02-stock_daily.csv', delimiter = ',')
   xy = xy[::-1]
   xy = MinMaxScaler(xy)
   x = xy
   y = xy[:, [-1]]

   dataX = []
   dataY = []

   for i in range(0, len(y) - seq_length) :
     _x = x[i:i + seq_length]
     _y = y[i + seq_length]
     # print(_x, "->", _y)
     dataX.append(_x)
     dataY.append(_y)
```

train, test 나누기 (0.7, 0.3)

```
train_size = int(len(dataY) * 0.7)
test_size = len(dataY) - train_size
trainX, testX = np.array(dataX[0:train_size]), np.array(dataX[train_size:len(dataX)])
trainY, testY = np.array(dataY[0:train_size]), np.array(dataY[train_size:len(dataY)])
```

SimpleRNN 층 하나로 구성된 모델 학습

```
import tensorflow as tf
n_units = 256
simpleRNN_model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(units = n_units, return_sequences = False, input_shape = [7, 5]),
    tf.keras.layers.Dense(1)
])

simpleRNN_model.compile(optimizer = 'adam', loss = 'mse')
simpleRNN_model.fit(trainX, trainY, epochs = 100)
```

```
16/16 [==============================] - 0s 11ms/step - loss: 9.4461e-04
Epoch 57/100
16/16 [==============================] - 0s 12ms/step - loss: 8.9734e-04
Epoch 58/100
16/16 [==============================] - 0s 10ms/step - loss: 9.3702e-04
Epoch 59/100
16/16 [==============================] - 0s 12ms/step - loss: 9.7580e-04
Epoch 60/100
16/16 [==============================] - 0s 12ms/step - loss: 8.9188e-04
Epoch 61/100
16/16 [==============================] - 0s 10ms/step - loss: 0.0010
Epoch 62/100
16/16 [==============================] - 0s 10ms/step - loss: 9.6039e-04
Epoch 63/100
```

plot 함수 정의

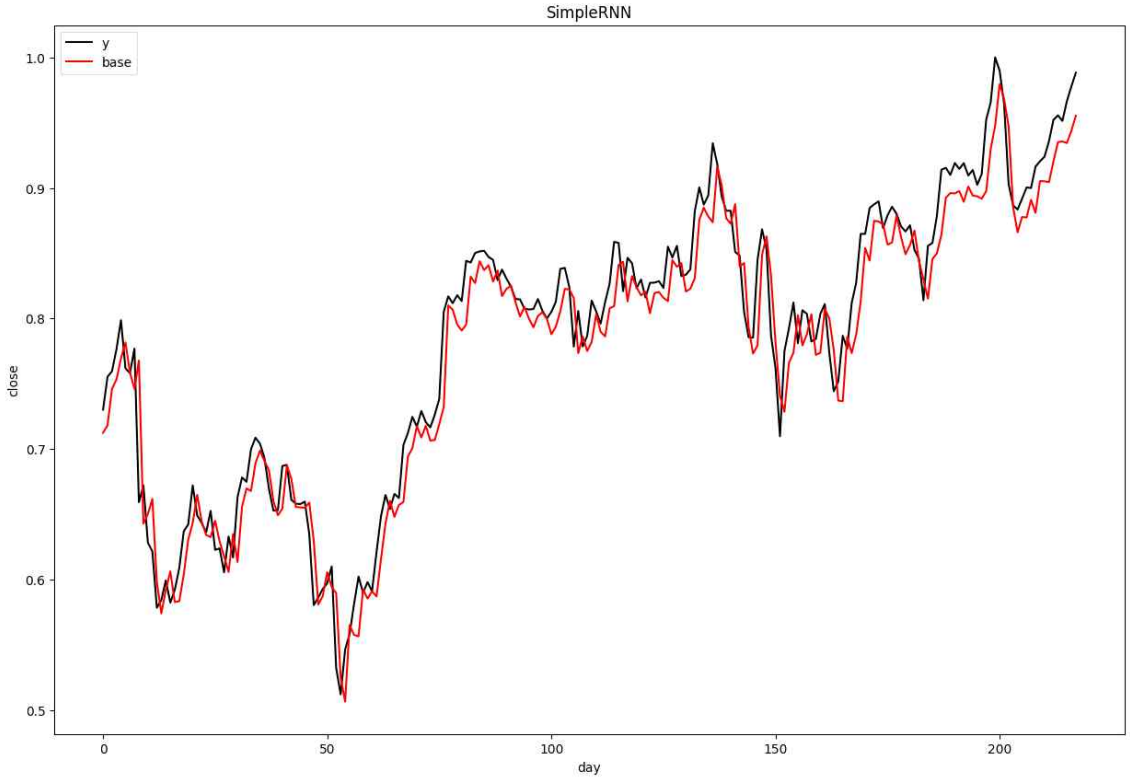
```
import matplotlib.pyplot as plt

def plot(output, title) :
  color = ["black", "red", "blue", "magenta"]
  label = ["y", "base", "LSTM", "base + layer"]
  plt.figure(figsize = (15, 10))
  for i, y in enumerate(output) :
    plt.plot(y, label = label[i], color = color[i])

  plt.title(title)
  plt.xlabel('day')
  plt.ylabel('close')
  plt.legend()
  plt.show()
```

SimpleRNN 층이 하나인 예측값

```
simpleRNN_pred = simpleRNN_model.predict(testX)
output = [testY, simpleRNN_pred]
plot(output, "SimpleRNN")
```



LSTM 층이 하나인 model 구성

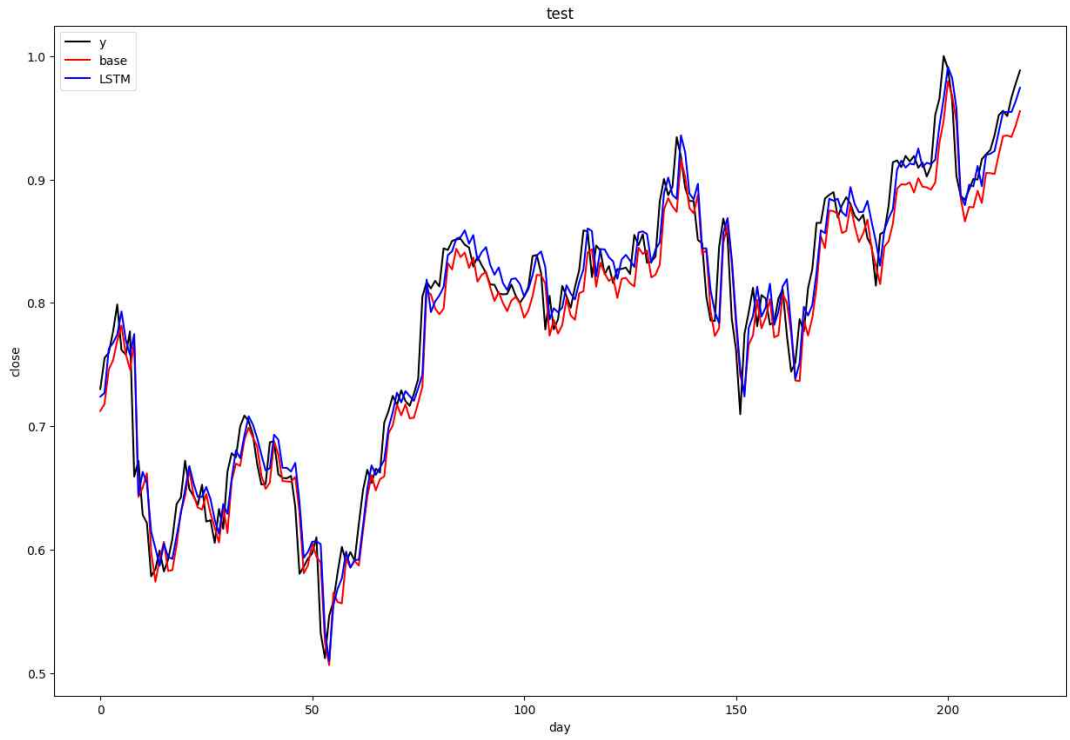
```
LSTM_model = tf.keras.Sequential([
    tf.keras.layers.LSTM(units = n_units, return_sequences = False, input_shape = [7, 5]),
    tf.keras.layers.Dense(1)
])

LSTM_model.compile(optimizer = 'adam', loss = 'mse')
LSTM_model.fit(trainX, trainY, epochs = 100)
```

```
Epoch 50/100
16/16 [==============================] - 0s 20ms/step - loss: 0.0013
Epoch 51/100
16/16 [==============================] - 0s 19ms/step - loss: 0.0012
Epoch 52/100
16/16 [==============================] - 0s 19ms/step - loss: 0.0011
Epoch 53/100
16/16 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 54/100
16/16 [==============================] - 0s 25ms/step - loss: 0.0012
Epoch 55/100
16/16 [==============================] - 0s 25ms/step - loss: 0.0012
Epoch 56/100
16/16 [==============================] - 0s 24ms/step - loss: 0.0011
Epoch 57/100
16/16 [==============================] - 0s 24ms/step - loss: 0.0011
```

LSTM 예측값 그래프

```
LSTM_pred = LSTM_model.predict(testX)
output = [testY, simpleRNN_pred, LSTM_pred]
plot(output, "test")
```



층이 4개인 RNN 학습

```
RNN_model = tf.keras.Sequential([
    tf.keras.layers.SimpleRNN(units = 34, return_sequences = True, input_shape = [7, 5]),
    tf.keras.layers.SimpleRNN(units = 34, return_sequences = True),
    tf.keras.layers.SimpleRNN(units = 34, return_sequences = True),
    tf.keras.layers.SimpleRNN(units = 34),
    tf.keras.layers.Dense(1)
])

LSTM_model.compile(optimizer = 'adam', loss = 'mse')
LSTM_model.fit(trainX, trainY, epochs = 100)
```

```
Epoch 63/100
16/16 [==============================] - 0s 20ms/step - loss: 8.2597e-04
Epoch 64/100
16/16 [==============================] - 0s 22ms/step - loss: 8.0816e-04
Epoch 65/100
16/16 [==============================] - 0s 20ms/step - loss: 7.5918e-04
Epoch 66/100
16/16 [==============================] - 0s 20ms/step - loss: 8.4529e-04
Epoch 67/100
16/16 [==============================] - 0s 20ms/step - loss: 7.7479e-04
Epoch 68/100
16/16 [==============================] - 0s 19ms/step - loss: 7.8487e-04
```

```
RNN_pred = RNN_model.predict(testX)
output = [testY, simpleRNN_pred, LSTM_pred, RNN_pred]
plot(output, "test")
```