70쪽 소스 제일 위 (추가)

#2. 모델 구성

55쪽 결과 중간부분 (수정)

acc : 0.908497….

mse : 0.908497…

65쪽 첫번째 소스 하단 (추가)

#RMSE 구하기

from sklearn.metrics import mean\_squared\_error

def RMSE(y\_test, y\_predict):

return np.sqrt(mean\_squared\_error(y\_test, y\_predict))

print("RMSE : ", RMSE(y\_test, y\_predict))

# R2 구하기

from sklearn.metrics import r2\_score

r2\_y\_predict = r2\_score(y\_test, y\_predict)

print("R2 : ", r2\_y\_predict)

72쪽 중간부분 (수정)

mse = model.evaluate(x\_test, y\_test, batch\_size=3)

mse = model.evaluate(x\_test, y\_test, batch\_size=1)

(batch의 크기를 evaluate시 변경하는 것은 상관은 없으나, 책의 본문의 문맥상 batch를 1로 잡는 것이 나을 것 같습니다.)

78쪽 밑에서 3번째 줄 (수정)

마지막을

마지막으로

99쪽 첫번째 소스 하단 (수정)

x\_predict = np.array(11,12,13)

x\_predict = np.array([11, 12, 13])

113쪽 첫번째 소스 하단 (수정)

x\_predict = np.array(11,12,13)

x\_predict = np.array([11, 12, 13])

116쪽 첫번째 소스 (수정)

\_train



x\_train

121쪽 2번째 소스 하단에 (추가)

model = Model(inputs=input1, outputs=output1)

129쪽 2번째 소스 밑에서 2번째 줄(수정)

dense1 = Dense(7)(dense1\_2)

dense1 = Dense(7)(dense1)

130쪽 마지막 소스 2군데 (수정)

output1 = Dense(3)(output1)

output1 = Dense(2)(output1)

output2 = Dense(3)(output2)

output2 = Dense(2)(output2)

171쪽 소스 상단 (추가)

#3. 훈련

model.compile(loss='mse', optimizer='adam', metrics=['mse'])

model.fit(x\_train, y\_train, epochs=100, batch\_size=1)

174쪽 소스 중간 (수정)

#2. 두 번

#2. 모델 구성

186쪽 첫번째 소스 (수정)

verbose=0

verbose=1

234쪽 마지막소스 마지막 줄 (수정)

print(x\_test.shape)



print(x\_pred.shape)

252쪽 중간 부분 (수정)

그렇다면 위 모델은

그렇다면 위 모델을

263쪽 3번째 소스 상단 (추가)

#2. 모델 구성

from keras.models import Sequential

from keras.layers import Dense, LSTM

=================== 2022년 1월 7일 추가 ======================

32쪽 2번째 줄

from keras.models import Sequential,

->

from keras.models import Sequential ‘,’ 삭제

43쪽 소스 밑에서 4번째 줄

validation\_data = (x\_train, y\_train))

->

validation\_data = (x\_test, y\_test)

163쪽 첫째 줄

(None, 3, 1) -> (None, 5, 1)

====================== 2022년 3월 11일 추가 ===================

35쪽

35페이지 예제는 사실 회귀모델이어서 acc를 쓸 경우 정확하게 판정이 나지 않습니다. 그러나 초심자분들의 이해를 쉽게 하기 위해 당시 acc를 사용하여 코딩하였고, 집필시점의 버전에서 출력시에는 acc=1.0으로 출력 확인과 검수까지 마친 상태였습니다.

그러나 현재 확인하니 acc=0으로 나오는 증상이 확인이 되어, 소스를 아래와 같이 수정하여 실습 진행해 주시기 부탁드립니다.

35페이지

import numpy as np

x = np.array([1,2,3,4,5,6,7,8,9,10])

y = np.array([1,2,3,4,5,6,7,8,9,10])

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(10, input\_dim=1, activation='relu'))    # 수정

model.add(Dense(1))                                     # 추가

model.compile(loss='mse', optimizer='adam', metrics=['acc'])

model.fit(x, y, epochs=500, batch\_size=1)

loss, acc = model.evaluate(x, y, batch\_size=1)

print('loss : ', loss)

# print('acc : ', acc)                      # 삭제

y\_pred = np.round(model.predict(x), 0)      # 추가

# print(y\_pred)

from sklearn.metrics import accuracy\_score  # 추가

acc = accuracy\_score(y, y\_pred)            # 추가

print('acc : ', acc)                       # 추가

추가 설명을 드리겠습니다.

모델 구성시

model.add(Dense(1, input\_dim=1, activation='relu'))

위와 같이 단일 레이어로 구성되어 있던 부분에

model.add(Dense(10, input\_dim=1, activation='relu'))    # 수정

model.add(Dense(1))                                     # 추가

히든레이어(노드10개)를 추가하여 조금 더 깊게 구성하도록 추가하였습니다.

Accuracy를 출력하는 부분을 삭제하고, x값을 predict하여 y\_pred를 출력합니다.

y\_pred에는 출력값이 소수점으로 출력됩니다.(회귀모델이기 때문입니다.) 이것을 반올림(np.round)하여 정수값으로 바꿉니다.

이후 sklearn에서 제공하는 accuracy\_score를 import한 후 accuracy\_score에 원래 y값과, 이번에 x값을 넣어서 예측한 y\_pred를 비교하여 acc를 도출해 냅니다.

Epoch 500/500

1/10 [==>...........................] - ETA: 0s - loss: 0.000010/10 [==============================] - 0s 2ms/step - loss: 2.4670e-12 - acc: 0.1000

1/10 [==>...........................] - ETA: 0s - loss: 6.878110/10 [==============================] - 0s 1ms/step - loss: 2.3533e-12 - acc: 0.1000

loss : 2.3533176275736256e-12

acc : 1.0

43페이지

import numpy as np

x\_train = np.array([1,2,3,4,5,6,7,8,9,10])

y\_train = np.array([1,2,3,4,5,6,7,8,9,10])

x\_test = np.array([101,102,103,104,105,106,107,108,109,110])

y\_test = np.array([101,102,103,104,105,106,107,108,109,110])

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(50, input\_dim=1, activation='relu'))    # 수정

model.add(Dense(25))                                    # 수정

model.add(Dense(1))                                     # 수정

model.compile(loss='mse', optimizer='adam', metrics=['acc'])

model.fit(x\_train, y\_train, epochs=100, batch\_size=1,

          validation\_data = (x\_test, y\_test))

loss, acc = model.evaluate(x\_test, y\_test, batch\_size=1)

print('loss : ', loss)

# print('acc : ', acc)                      # 삭제

y\_pred = np.round(model.predict(x\_test), 0)      # 추가

# print(y\_pred)

from sklearn.metrics import accuracy\_score  # 추가

acc = accuracy\_score(y\_test, y\_pred)            # 추가

print('acc : ', acc)                       # 추가

책에서는 모델구성시 각 레이어당 5, 3, 1로 노드를 주었으나, 정확도를 높이기 위해

노드의 개수를 50, 25, 1 로 수정하였습니다. 가장 마지막 레이어의 activation=’relu’는 삭제하였습니다.

Accuracy 평가 부분은 위 35페이지 소스와 동일합니다.

Epoch 100/100

10/10 [==============================] - 0s 7ms/step - loss: 1.1260e-04 - acc: 0.1000 - val\_loss: 0.0551 - val\_acc: 0.0000e+00

10/10 [==============================] - 0s 2ms/step - loss: 0.0551 - acc: 0.0000e+00

loss : 0.05508648231625557

acc : 1.0