# 딥러닝1 : 회귀

## 1.환경준비

### (1) 라이브러리 로딩

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential
from keras.layers import Dense
from keras.backend import clear_session
from keras.optimizers import Adam
```

- 학습곡선 그래프

### (2) 데이터로딩

```
path = 'https://raw.githubusercontent.com/DA4BAM/dataset/master/boston.csv'
data = pd.read_csv(path)
data.head()
```

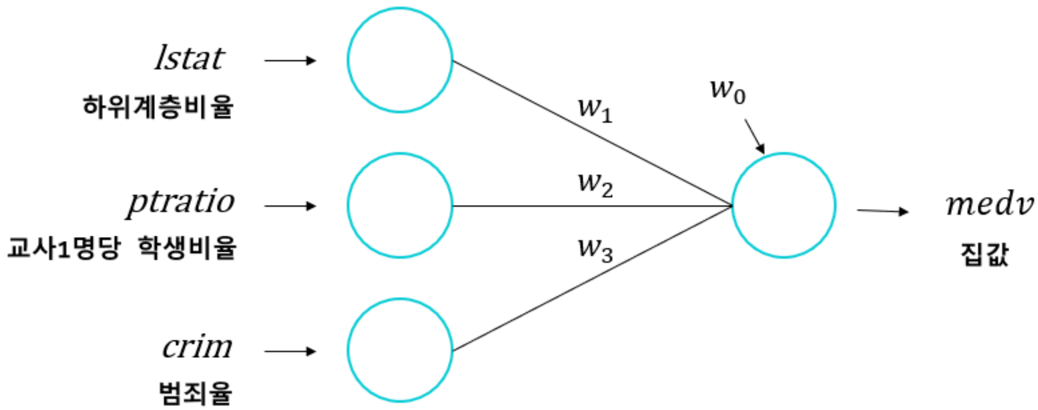
| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | lstat | medv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 4.98 | 24.0 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 9.14 | 21.6 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 4.03 | 34.7 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 2.94 | 33.4 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 5.33 | 36.2 |

Next steps: Generate code with data View recommended plots

| 변수 | 설명 |
|---|---|
| medv | 타운별 집값(중위수) |
| crim | 범죄율 |
| zn | 25,000 평방피트를 초과 거주지역 비율 |
| indus | 비소매상업지역 면적 비율 |
| chas | 찰스강변 위치(범주 : 강변1, 아니면 0) |
| nox | 일산화질소 농도 |
| rm | 주택당 방 수 |
| age | 1940년 이전에 건축된 주택의 비율 |
| dis | 직업센터의 거리 |
| rad | 방사형 고속도로까지의 거리 |
| tax | 재산세율 |
| ptratio | 학생/교사 비율 |
| lstat | 인구 중 하위 계층 비율 |

## 2.데이터 준비

lstat, ptratio, crim 만 이용하여 medv를 예측하는 모델을 만들어 봅시다.




## (1) 데이터 준비

- x, y 나누기
  - x : lstat, ptratio, crim
  - y : medv

```
target = 'medv'
features = ['lstat', 'ptratio', 'crim']
x = data.loc[:, features]
y = data.loc[:, target]
```

## (2) NaN 조치

## (3) 가변수화

## (4) 데이터분할

```
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

## (5) Scaling

```
# 스케일러 선언
scaler = MinMaxScaler()

# train 셋으로 fitting & 적용
x_train = scaler.fit_transform(x_train)

# validation 셋은 적용만!
x_val = scaler.transform(x_val)
```

# 3.딥러닝1 : 3개의 feature

## (1) 모델설계

```
# 분석단위의 shape
nfeatures = x_train.shape[1] #num of columns
nfeatures
```

```
3
```

```
# 메모리 정리
clear_session()

# Sequential 타입
model = Sequential(Dense(1, input_shape=(nfeatures,)))
                    # output            # input
# 모델요약
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 13

=================================================================
Total params: 13 (52.00 Byte)
Trainable params: 13 (52.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- compile
  - optimizer = 'adam' : 기본값으로 옵티마이저 사용(learning_rate = 0.001)
  - optimizer = Adam(lr = 0.1) : 옵션 값 조정 가능
    - lr과 learning_rate은 같지만, learning_rate 사용을 권장

```
model.compile(optimizer=Adam(0.1), loss='mse')
```

## (2) 학습

validation_split=0.2 : 학습시, 학습용 데이터에서 0.2 만큼 떼어 내서 검증셋으로 활용

```
history = model.fit(x_train , y_train, epochs = 20, validation_split=0.2).history
                                    # 20번 반복      # 20%를 검증셋 분리  # 가중치가 업데이트 되면서 그때그때마다의 성능을 측정하여 기록
```

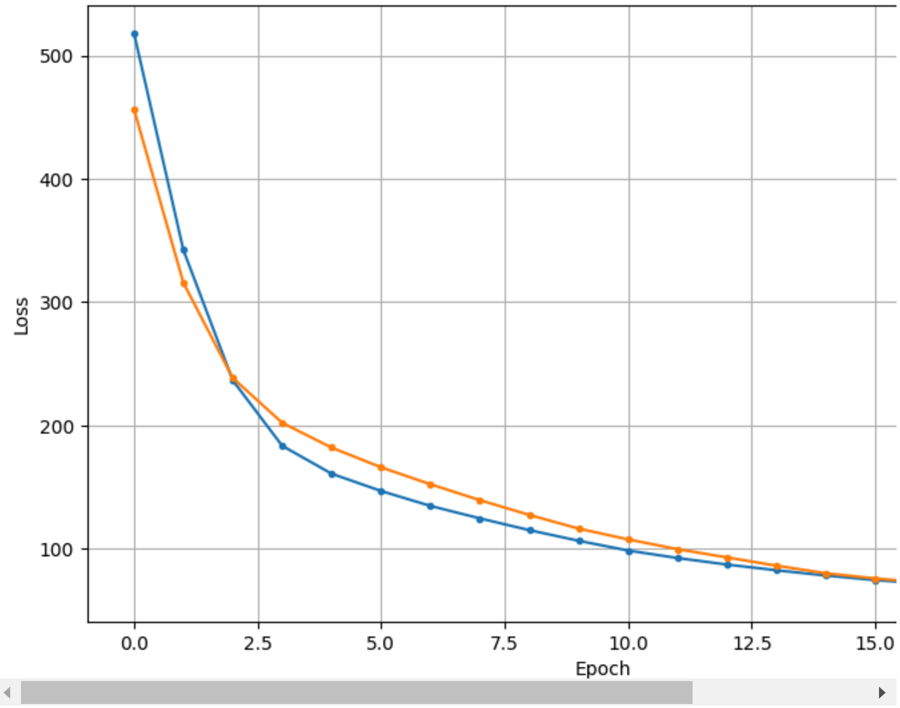
```
Epoch 1/20
11/11 [==============================] - 1s 22ms/step - loss: 518.1815 - val_loss: 456.4966
Epoch 2/20
11/11 [==============================] - 0s 7ms/step - loss: 342.1980 - val_loss: 315.6486
Epoch 3/20
11/11 [==============================] - 0s 8ms/step - loss: 236.4032 - val_loss: 238.3555
Epoch 4/20
11/11 [==============================] - 0s 7ms/step - loss: 183.3738 - val_loss: 202.0274
Epoch 5/20
11/11 [==============================] - 0s 6ms/step - loss: 160.7343 - val_loss: 181.8968
Epoch 6/20
11/11 [==============================] - 0s 7ms/step - loss: 146.7394 - val_loss: 165.9303
Epoch 7/20
11/11 [==============================] - 0s 7ms/step - loss: 134.6086 - val_loss: 152.1724
Epoch 8/20
11/11 [==============================] - 0s 9ms/step - loss: 124.5222 - val_loss: 139.3053
Epoch 9/20
11/11 [==============================] - 0s 8ms/step - loss: 114.9785 - val_loss: 127.3080
Epoch 10/20
11/11 [==============================] - 0s 7ms/step - loss: 106.3456 - val_loss: 116.1977
Epoch 11/20
11/11 [==============================] - 0s 8ms/step - loss: 98.4367 - val_loss: 107.4543
Epoch 12/20
11/11 [==============================] - 0s 8ms/step - loss: 92.3563 - val_loss: 99.5525
Epoch 13/20
11/11 [==============================] - 0s 8ms/step - loss: 87.0666 - val_loss: 92.8567
Epoch 14/20
11/11 [==============================] - 0s 7ms/step - loss: 82.4478 - val_loss: 86.1604
Epoch 15/20
11/11 [==============================] - 0s 7ms/step - loss: 78.1571 - val_loss: 79.9763
Epoch 16/20
11/11 [==============================] - 0s 8ms/step - loss: 74.4615 - val_loss: 75.7875
Epoch 17/20
11/11 [==============================] - 0s 8ms/step - loss: 71.6463 - val_loss: 72.4373
Epoch 18/20
11/11 [==============================] - 0s 8ms/step - loss: 69.0822 - val_loss: 68.6514
Epoch 19/20
11/11 [==============================] - 0s 7ms/step - loss: 67.1005 - val_loss: 66.0043
Epoch 20/20
11/11 [==============================] - 0s 8ms/step - loss: 65.2486 - val_loss: 63.6862
```

- 학습결과 그래프

```
# 함수로 만들어서 사용합시다.
def dl_history_plot(history):
    plt.figure(figsize=(10,6))
    plt.plot(history['loss'], label='train_err', marker = '.')
    plt.plot(history['val_loss'], label='val_err', marker = '.')

    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend()
    plt.grid()
    plt.show()
```

```
dl_history_plot(history)
```




### (3) 예측 및 검증

```
pred = model.predict(x_val)
```

```
4/4 [==============================] - 0s 3ms/step
```

```
print(f'RMSE : {mean_squared_error(y_val, pred, squared=False)}')
print(f'MAE  : {mean_absolute_error(y_val, pred)}')
print(f'MAPE : {mean_absolute_percentage_error(y_val, pred)}')
```

```
RMSE : 6.950386677183718
MAE  : 4.948413884405998
MAPE : 0.2519252652659587
```

## 4. 딥러닝2 : 전체 feature

- 이제 전체 데이터를 가지고 모델링을 시도해 보겠습니다.

### (1) 데이터 전처리

- 데이터 분할

```
target = 'medv'
x = data.drop(target, axis = 1)
y = data.loc[:, target]
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

- 스케일링

```
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_val = scaler.transform(x_val)
```

## (2) 모델링

- 모델 설계

```
nfeatures = x_train.shape[1]
nfeatures
```

```
12
```

```
# 메모리 정리
clear_session()

# Sequential 타입 모델 선언
model2 =Sequential(Dense(1, input_shape = (nfeatures,)))

# 모델요약
model2.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 13

=================================================================
Total params: 13 (52.00 Byte)
Trainable params: 13 (52.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- compile

```
model2.compile(optimizer=Adam(0.1), loss='mse')
```

- 학습

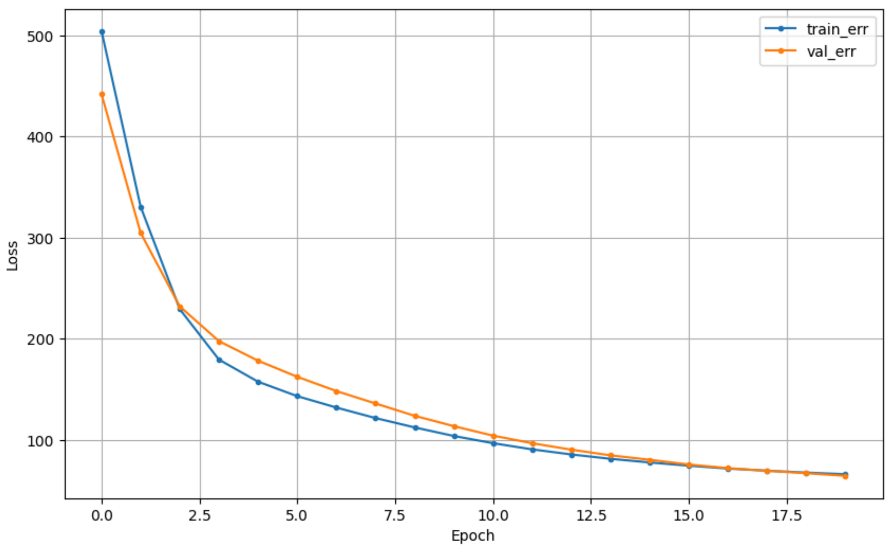
```
history = model2.fit(x_train , y_train, epochs = 20, validation_split=0.2).history
```

```
Epoch 1/20
11/11 [==============================] - 1s 31ms/step - loss: 504.4330 - val_loss: 442.0331
Epoch 2/20
11/11 [==============================] - 0s 11ms/step - loss: 330.4849 - val_loss: 304.7272
Epoch 3/20
11/11 [==============================] - 0s 10ms/step - loss: 229.2275 - val_loss: 231.9848
Epoch 4/20
11/11 [==============================] - 0s 10ms/step - loss: 179.4115 - val_loss: 197.6790
Epoch 5/20
11/11 [==============================] - 0s 12ms/step - loss: 157.5410 - val_loss: 178.3222
Epoch 6/20
11/11 [==============================] - 0s 10ms/step - loss: 143.2843 - val_loss: 162.4291
Epoch 7/20
11/11 [==============================] - 0s 9ms/step - loss: 131.9949 - val_loss: 148.3194
Epoch 8/20
11/11 [==============================] - 0s 8ms/step - loss: 121.5890 - val_loss: 136.0061
Epoch 9/20
11/11 [==============================] - 0s 7ms/step - loss: 112.3518 - val_loss: 123.8257
Epoch 10/20
11/11 [==============================] - 0s 8ms/step - loss: 103.8586 - val_loss: 113.5853
Epoch 11/20
```

```
11/11 [==============================] - 0s 6ms/step - loss: 96.7510 - val_loss: 104.2314
Epoch 12/20
11/11 [==============================] - 0s 6ms/step - loss: 90.6481 - val_loss: 96.7122
Epoch 13/20
11/11 [==============================] - 0s 6ms/step - loss: 85.6115 - val_loss: 90.3562
Epoch 14/20
11/11 [==============================] - 0s 6ms/step - loss: 81.2935 - val_loss: 84.7137
Epoch 15/20
11/11 [==============================] - 0s 6ms/step - loss: 77.7168 - val_loss: 80.3703
Epoch 16/20
11/11 [==============================] - 0s 6ms/step - loss: 74.4987 - val_loss: 75.6778
Epoch 17/20
11/11 [==============================] - 0s 4ms/step - loss: 71.6750 - val_loss: 72.0910
Epoch 18/20
11/11 [==============================] - 0s 6ms/step - loss: 69.5156 - val_loss: 69.4446
Epoch 19/20
11/11 [==============================] - 0s 6ms/step - loss: 67.6579 - val_loss: 67.0151
Epoch 20/20
11/11 [==============================] - 0s 8ms/step - loss: 66.0524 - val_loss: 64.4714
```

- 학습결과 그래프

```
dl_history_plot(history)
```




- 예측 및 평가

```
pred2 = model2.predict(x_val)

print(f'RMSE : {mean_squared_error(y_val, pred2, squared=False)}')
print(f'MAE  : {mean_absolute_error(y_val, pred2)}')
print(f'MAPE : {mean_absolute_percentage_error(y_val, pred2)}')
```

```
4/4 [==============================] - 0s 3ms/step
RMSE : 6.988840926795755
MAE  : 4.967330549277513
MAPE : 0.24785881013654562
```

## 5.실습!

- 위 4번에 이어서, 여러분은 다음을 조절할 수 있습니다.
    - epochs(반복횟수), learning_rate(학습율)
- 4번 코드를 그대로 보면서 작성하고 위 두가지를 조절하며 성능을 높여봅시다!

## (1) 데이터 전처리

- 데이터 분할

```
target = 'medv'
x = data.drop(target, axis = 1)
y = data.loc[:, target]
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

- 스케일링

```
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_val = scaler.transform(x_val)
```

## (2) 모델링

- 모델 설계

```
nfeatures = x_train.shape[1]
nfeatures
```

```
12
```

```
clear_session()
model = Sequential(Dense(1, input_shape = (nfeatures,)))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 13

=================================================================
Total params: 13 (52.00 Byte)
Trainable params: 13 (52.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- compile

```
model.compile(optimizer=Adam(0.2), loss='mse')
```

- 학습

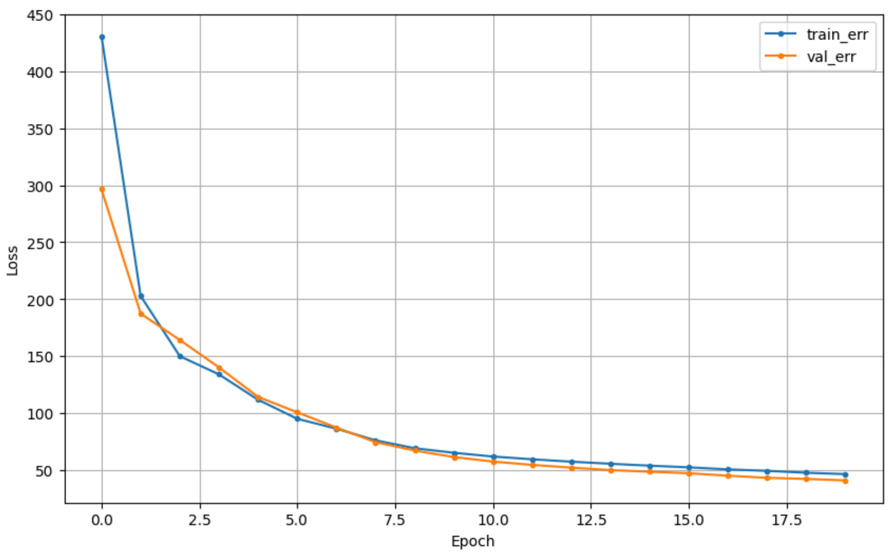
```
history = model.fit(x_train, y_train, epochs=20, validation_split=0.2).history
```

```
Epoch 1/20
11/11 [==============================] - 1s 22ms/step - loss: 430.6400 - val_loss: 296.4463
Epoch 2/20
11/11 [==============================] - 0s 7ms/step - loss: 202.8281 - val_loss: 187.4020
Epoch 3/20
11/11 [==============================] - 0s 8ms/step - loss: 150.0531 - val_loss: 164.2988
Epoch 4/20
11/11 [==============================] - 0s 7ms/step - loss: 134.1273 - val_loss: 140.2191
Epoch 5/20
11/11 [==============================] - 0s 6ms/step - loss: 111.8454 - val_loss: 114.3371
Epoch 6/20
11/11 [==============================] - 0s 6ms/step - loss: 95.1658 - val_loss: 100.8489
Epoch 7/20
11/11 [==============================] - 0s 8ms/step - loss: 86.4942 - val_loss: 87.4221
Epoch 8/20
11/11 [==============================] - 0s 6ms/step - loss: 76.2850 - val_loss: 74.6997
Epoch 9/20
11/11 [==============================] - 0s 8ms/step - loss: 69.3516 - val_loss: 67.3232
Epoch 10/20
11/11 [==============================] - 0s 8ms/step - loss: 65.3635 - val_loss: 61.6386
```

```
Epoch 11/20
11/11 [==============================] - 0s 8ms/step - loss: 62.0674 - val_loss: 57.6342
Epoch 12/20
11/11 [==============================] - 0s 8ms/step - loss: 59.6570 - val_loss: 54.7396
Epoch 13/20
11/11 [==============================] - 0s 8ms/step - loss: 57.5501 - val_loss: 52.3332
Epoch 14/20
11/11 [==============================] - 0s 10ms/step - loss: 55.7252 - val_loss: 50.1859
Epoch 15/20
11/11 [==============================] - 0s 8ms/step - loss: 54.0587 - val_loss: 48.7423
Epoch 16/20
11/11 [==============================] - 0s 10ms/step - loss: 52.5839 - val_loss: 47.4136
Epoch 17/20
11/11 [==============================] - 0s 11ms/step - loss: 50.7782 - val_loss: 45.2786
Epoch 18/20
11/11 [==============================] - 0s 11ms/step - loss: 49.5388 - val_loss: 43.3967
Epoch 19/20
11/11 [==============================] - 0s 11ms/step - loss: 47.8421 - val_loss: 42.4350
Epoch 20/20
11/11 [==============================] - 0s 10ms/step - loss: 46.6854 - val_loss: 41.1082
```

- 학습결과 그래프

```
dl_history_plot(history)
```




- 예측 및 평가

```
pred = model.predict(x_val)

print('RMSE:', mean_squared_error(y_val, pred, squared=False))
print('='*60)
print('MAE:', mean_absolute_error(y_val, pred))
print('='*60)
print('MAPE:', mean_absolute_percentage_error(y_val, pred))
```

```
4/4 [==============================] - 0s 3ms/step
RMSE: 5.961475859128757
============================================================
MAE: 4.457365878423055
============================================================
MAPE: 0.22097037018113536
```

## 5.딥러닝3 : hidden layer!

- 이제 레이어를 추가해 보겠습니다.

### (1) 데이터 전처리

- 데이터 분할

```
target = 'medv'
x = data.drop(target, axis = 1)
y = data.loc[:, target]

x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=.2, random_state = 20)
```

- 스케일링

```
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_val = scaler.transform(x_val)
```

### (2) 모델링

- 모델 설계

```
nfeatures = x_train.shape[1] #num of columns
nfeatures
```

```
12
```

```
# 메모리 정리
clear_session()

# Sequential 타입 모델 선언(입력은 리스트로!) # 레이터 2개 이상은 리스트로 작성
model3 = Sequential([  Dense(2, input_shape = (nfeatures,), activation = 'relu'),
                       Dense(1)   ])

# 모델요약
model3.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 26

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 29 (116.00 Byte)
Trainable params: 29 (116.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- compile

```
model3.compile( optimizer= Adam(learning_rate=0.1), loss = 'mse')
```

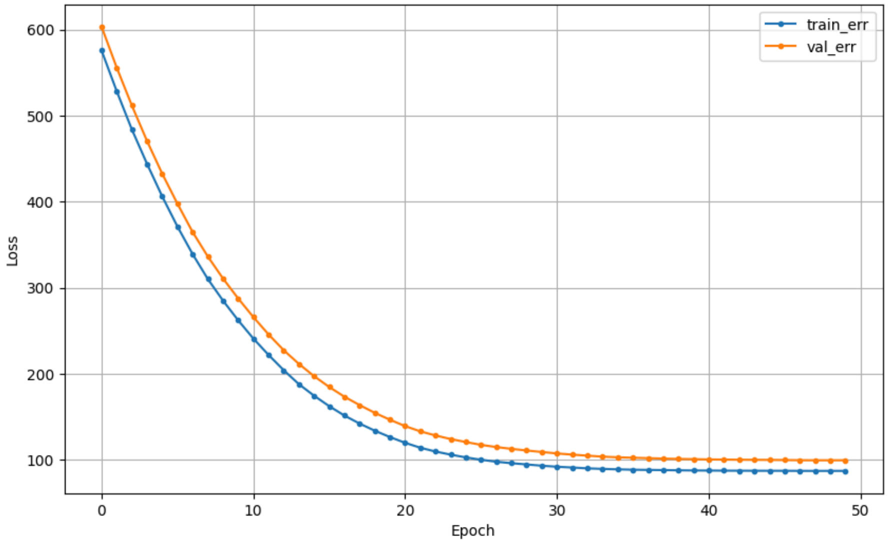
- 학습

```
hist = model3.fit(x_train, y_train, epochs = 50 , validation_split= .2 ).history
```

```
Epoch 24/50
11/11 [==============================] - 0s 5ms/step - loss: 105.9005 - val_loss: 124.0245
Epoch 25/50
11/11 [==============================] - 0s 5ms/step - loss: 102.6706 - val_loss: 120.5330
Epoch 26/50
11/11 [==============================] - 0s 5ms/step - loss: 99.9475 - val_loss: 117.2674
Epoch 27/50
11/11 [==============================] - 0s 5ms/step - loss: 97.6628 - val_loss: 114.6974
Epoch 28/50
11/11 [==============================] - 0s 6ms/step - loss: 95.9381 - val_loss: 112.6250
Epoch 29/50
11/11 [==============================] - 0s 6ms/step - loss: 94.5247 - val_loss: 110.7104
Epoch 30/50
11/11 [==============================] - 0s 7ms/step - loss: 93.0090 - val_loss: 108.9917
Epoch 31/50
11/11 [==============================] - 0s 7ms/step - loss: 91.8360 - val_loss: 107.3063
Epoch 32/50
11/11 [==============================] - 0s 9ms/step - loss: 90.9151 - val_loss: 105.9331
Epoch 33/50
11/11 [==============================] - 0s 8ms/step - loss: 89.9590 - val_loss: 104.7290
Epoch 34/50
11/11 [==============================] - 0s 7ms/step - loss: 89.3106 - val_loss: 103.5834
Epoch 35/50
11/11 [==============================] - 0s 8ms/step - loss: 88.8220 - val_loss: 102.7982
Epoch 36/50
11/11 [==============================] - 0s 7ms/step - loss: 88.3663 - val_loss: 102.3119
Epoch 37/50
11/11 [==============================] - 0s 7ms/step - loss: 88.1299 - val_loss: 101.8126
Epoch 38/50
11/11 [==============================] - 0s 8ms/step - loss: 87.9093 - val_loss: 101.3141
Epoch 39/50
11/11 [==============================] - 0s 6ms/step - loss: 87.7131 - val_loss: 100.8942
Epoch 40/50
11/11 [==============================] - 0s 8ms/step - loss: 87.4897 - val_loss: 100.6448
Epoch 41/50
11/11 [==============================] - 0s 7ms/step - loss: 87.4230 - val_loss: 100.3058
Epoch 42/50
11/11 [==============================] - 0s 6ms/step - loss: 87.2765 - val_loss: 100.1660
Epoch 43/50
11/11 [==============================] - 0s 5ms/step - loss: 87.2207 - val_loss: 99.9912
Epoch 44/50
11/11 [==============================] - 0s 5ms/step - loss: 87.1707 - val_loss: 99.8833
Epoch 45/50
11/11 [==============================] - 0s 7ms/step - loss: 87.1424 - val_loss: 99.8052
Epoch 46/50
11/11 [==============================] - 0s 7ms/step - loss: 87.1190 - val_loss: 99.6054
Epoch 47/50
11/11 [==============================] - 0s 5ms/step - loss: 87.0224 - val_loss: 99.4230
Epoch 48/50
11/11 [==============================] - 0s 7ms/step - loss: 86.9958 - val_loss: 99.3085
Epoch 49/50
11/11 [==============================] - 0s 5ms/step - loss: 87.0096 - val_loss: 99.3382
Epoch 50/50
11/11 [==============================] - 0s 6ms/step - loss: 87.0100 - val_loss: 99.3608
```

- 학습결과 그래프

```
dl_history_plot(hist)
```




- 예측 및 평가

```
pred3 = model3.predict(x_val)
print(f'RMSE : {mean_squared_error(y_val, pred3, squared=False)}')
print(f'MAE  : {mean_absolute_error(y_val, pred3)}')
print(f'MAPE : {mean_absolute_percentage_error(y_val, pred3)}')
```

```
4/4 [==============================] - 0s 3ms/step
RMSE : 8.049523194590314
MAE  : 5.496162048040652
MAPE : 0.3189378401797094
```

## (3) 실습1

- 다음의 summary를 보고 모델을 설계하시오.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense (Dense) | (None, 8) | 104 |
| dense_1 (Dense) | (None, 1) | 9 |

```
clear_session()
model = Sequential([Dense(8, input_shape=(nfeatures, )),
                    Dense(1, )])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 104

 dense_1 (Dense)             (None, 1)                 9

=================================================================
Total params: 113 (452.00 Byte)
Trainable params: 113 (452.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

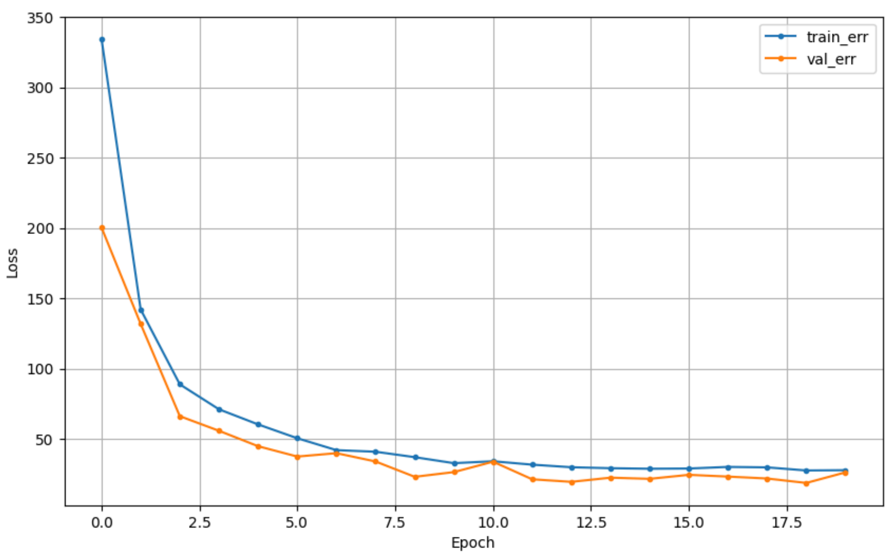
- 컴파일 + 학습

```
model.compile(optimizer=Adam(0.1), loss='mse')
hist = model.fit(x_train, y_train, epochs=20, validation_split=0.2).history
```

```
Epoch 1/20
11/11 [==============================] - 1s 22ms/step - loss: 334.6101 - val_loss: 200.3792
Epoch 2/20
11/11 [==============================] - 0s 7ms/step - loss: 142.0729 - val_loss: 131.7999
Epoch 3/20
11/11 [==============================] - 0s 8ms/step - loss: 88.8284 - val_loss: 66.1786
Epoch 4/20
11/11 [==============================] - 0s 8ms/step - loss: 71.0798 - val_loss: 55.7986
Epoch 5/20
11/11 [==============================] - 0s 7ms/step - loss: 60.3257 - val_loss: 44.8203
Epoch 6/20
11/11 [==============================] - 0s 7ms/step - loss: 50.4952 - val_loss: 37.4636
Epoch 7/20
11/11 [==============================] - 0s 7ms/step - loss: 42.0382 - val_loss: 39.8750
Epoch 8/20
11/11 [==============================] - 0s 8ms/step - loss: 40.8949 - val_loss: 33.9553
Epoch 9/20
11/11 [==============================] - 0s 5ms/step - loss: 37.0261 - val_loss: 23.0622
Epoch 10/20
11/11 [==============================] - 0s 6ms/step - loss: 32.7690 - val_loss: 26.4188
Epoch 11/20
11/11 [==============================] - 0s 7ms/step - loss: 34.1215 - val_loss: 33.8155
Epoch 12/20
11/11 [==============================] - 0s 5ms/step - loss: 31.7085 - val_loss: 21.2967
Epoch 13/20
11/11 [==============================] - 0s 4ms/step - loss: 29.8812 - val_loss: 19.4856
Epoch 14/20
11/11 [==============================] - 0s 5ms/step - loss: 29.1890 - val_loss: 22.4526
Epoch 15/20
11/11 [==============================] - 0s 6ms/step - loss: 28.7998 - val_loss: 21.6147
Epoch 16/20
11/11 [==============================] - 0s 5ms/step - loss: 28.9595 - val_loss: 24.4729
Epoch 17/20
11/11 [==============================] - 0s 6ms/step - loss: 30.1043 - val_loss: 23.2260
Epoch 18/20
11/11 [==============================] - 0s 6ms/step - loss: 29.7795 - val_loss: 21.8541
Epoch 19/20
11/11 [==============================] - 0s 6ms/step - loss: 27.5816 - val_loss: 18.7020
Epoch 20/20
11/11 [==============================] - 0s 7ms/step - loss: 27.7808 - val_loss: 26.2119
```

- 학습곡선

```
dl_history_plot(hist)
```




- 검증

```
pred = model.predict(x_val)
print('RMSE:', mean_squared_error(y_val, pred , squared=False))
print('MAE:', mean_absolute_error(y_val, pred))
print('MAPE: ', mean_absolute_percentage_error(y_val, pred))
```

```
4/4 [==============================] - 0s 3ms/step
RMSE: 4.509549114778724
MAE: 3.2778604025934257
MAPE:  0.16598983270966736
```

## (4) 실습2

- 다음의 summary를 보고 모델을 설계하시오.

| Layer (type) | Output Shape | Param # | 옵션 |
|---|---|---|---|
| dense (Dense) | (None, 8) | 112 | node, input_shape, activation |
| dense_1 (Dense) | (None, 4) | 36 | node, activation |
| dense_2 (Dense) | (None, 1) | 5 | node |

```
clear_session()
model = Sequential([Dense(8, input_shape = (nfeatures,), activation = 'relu'),
                    Dense(4, activation = 'relu'),
                    Dense(1)])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 104

 dense_1 (Dense)             (None, 4)                 36

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 145 (580.00 Byte)
Trainable params: 145 (580.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- 컴파일 + 학습

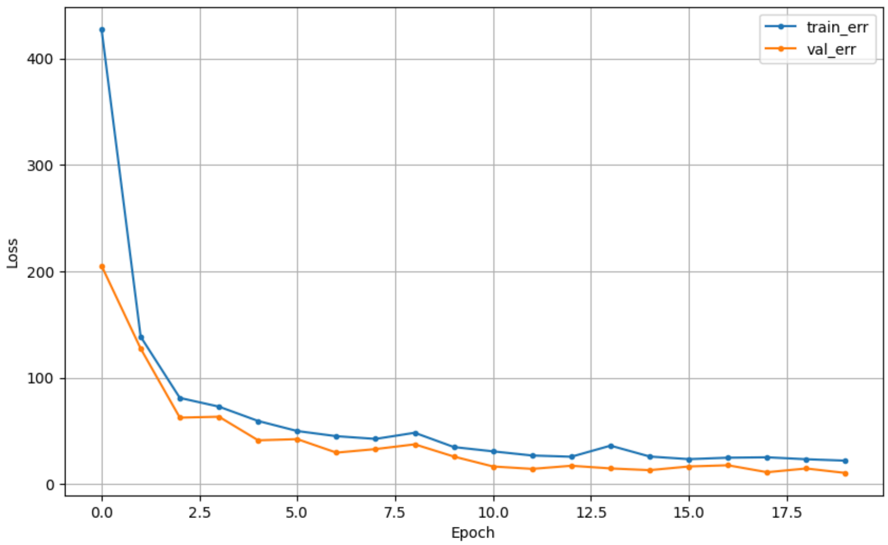
```
model.compile(optimizer=Adam(0.1), loss='mse')
hist = model.fit(x_train, y_train, epochs=20, validation_split=0.2).history
```

```
Epoch 1/20
11/11 [==============================] - 1s 19ms/step - loss: 427.6231 - val_loss: 205.2241
Epoch 2/20
11/11 [==============================] - 0s 5ms/step - loss: 138.5264 - val_loss: 127.3764
Epoch 3/20
11/11 [==============================] - 0s 5ms/step - loss: 81.0397 - val_loss: 62.4943
Epoch 4/20
11/11 [==============================] - 0s 7ms/step - loss: 72.8263 - val_loss: 63.3644
Epoch 5/20
11/11 [==============================] - 0s 7ms/step - loss: 59.3638 - val_loss: 41.2012
Epoch 6/20
11/11 [==============================] - 0s 7ms/step - loss: 49.9108 - val_loss: 42.3057
Epoch 7/20
11/11 [==============================] - 0s 7ms/step - loss: 45.0765 - val_loss: 29.6143
Epoch 8/20
11/11 [==============================] - 0s 5ms/step - loss: 42.5264 - val_loss: 32.9538
Epoch 9/20
11/11 [==============================] - 0s 5ms/step - loss: 48.2902 - val_loss: 37.4191
Epoch 10/20
11/11 [==============================] - 0s 8ms/step - loss: 34.8551 - val_loss: 25.9258
Epoch 11/20
11/11 [==============================] - 0s 8ms/step - loss: 30.8130 - val_loss: 16.6155
Epoch 12/20
11/11 [==============================] - 0s 7ms/step - loss: 26.9387 - val_loss: 14.3468
Epoch 13/20
11/11 [==============================] - 0s 10ms/step - loss: 25.7864 - val_loss: 17.2815
Epoch 14/20
11/11 [==============================] - 0s 7ms/step - loss: 36.1830 - val_loss: 14.7834
Epoch 15/20
11/11 [==============================] - 0s 8ms/step - loss: 25.9811 - val_loss: 13.1608
```

```
Epoch 16/20
11/11 [==============================] - 0s 8ms/step - loss: 23.5179 - val_loss: 16.6177
Epoch 17/20
11/11 [==============================] - 0s 8ms/step - loss: 24.8597 - val_loss: 17.7449
Epoch 18/20
11/11 [==============================] - 0s 7ms/step - loss: 25.2456 - val_loss: 11.2220
Epoch 19/20
11/11 [==============================] - 0s 7ms/step - loss: 23.4234 - val_loss: 14.7320
Epoch 20/20
11/11 [==============================] - 0s 8ms/step - loss: 22.0915 - val_loss: 10.5153
```

- 학습곡선

```
dl_history_plot(hist)
```




- 검증

```
pred = model.predict(x_val)
print('RMSE:', mean_squared_error(y_val, pred))
print('MAE:', mean_absolute_error(y_val, pred))
print('MAPE:', mean_absolute_percentage_error(y_val, pred))
```

```
4/4 [==============================] - 0s 4ms/step
RMSE: 17.968257299126716
MAE: 3.317481493482403
MAPE: 0.17787997410623801
```

## (5) 실습3

- 이번에는 여러분이 원하는 대로 설계하고, 학습해 봅시다.

```
nfeatures
```

```
12
```

```
clear_session()
model = Sequential([Dense(10, input_shape = (nfeatures,), activation = 'relu'),
                    Dense(5, activation = 'relu'),
                    Dense(2, activation = 'relu'),
                    Dense(1)])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 10)                130       
                                                                 
 dense_1 (Dense)             (None, 5)                 55        
                                                                 
 dense_2 (Dense)             (None, 2)                 12        
                                                                 
 dense_3 (Dense)             (None, 1)                 3         
                                                                 
=================================================================
Total params: 200 (800.00 Byte)
Trainable params: 200 (800.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- 컴파일 + 학습

```
model.compile(optimizer=Adam(0.01), loss='mse')
hist = model.fit(x_train, y_train, epochs=30, validation_split=0.2).history
```

```
Epoch 2/30
11/11 [==============================] - 0s 5ms/step - loss: 441.2220 - val_loss: 360.9224
Epoch 3/30
11/11 [==============================] - 0s 5ms/step - loss: 221.2153 - val_loss: 184.5843
Epoch 4/30
11/11 [==============================] - 0s 7ms/step - loss: 172.8001 - val_loss: 159.5147
Epoch 5/30
11/11 [==============================] - 0s 5ms/step - loss: 127.9323 - val_loss: 139.5051
Epoch 6/30
11/11 [==============================] - 0s 6ms/step - loss: 109.7522 - val_loss: 108.9406
Epoch 7/30
11/11 [==============================] - 0s 6ms/step - loss: 90.8513 - val_loss: 87.5680
Epoch 8/30
11/11 [==============================] - 0s 6ms/step - loss: 78.2152 - val_loss: 72.2928
Epoch 9/30
11/11 [==============================] - 0s 5ms/step - loss: 66.9805 - val_loss: 57.9580
Epoch 10/30
11/11 [==============================] - 0s 7ms/step - loss: 58.9006 - val_loss: 51.8733
Epoch 11/30
11/11 [==============================] - 0s 6ms/step - loss: 57.2095 - val_loss: 48.2587
Epoch 12/30
11/11 [==============================] - 0s 6ms/step - loss: 53.6062 - val_loss: 42.6373
Epoch 13/30
11/11 [==============================] - 0s 6ms/step - loss: 50.3853 - val_loss: 40.9081
Epoch 14/30
11/11 [==============================] - 0s 7ms/step - loss: 48.7626 - val_loss: 37.0658
Epoch 15/30
11/11 [==============================] - 0s 8ms/step - loss: 44.7278 - val_loss: 34.5931
Epoch 16/30
11/11 [==============================] - 0s 8ms/step - loss: 42.8035 - val_loss: 33.7240
Epoch 17/30
11/11 [==============================] - 0s 8ms/step - loss: 41.0996 - val_loss: 30.1669
Epoch 18/30
11/11 [==============================] - 0s 6ms/step - loss: 40.4547 - val_loss: 27.4050
Epoch 19/30
11/11 [==============================] - 0s 8ms/step - loss: 37.0391 - val_loss: 25.1136
Epoch 20/30
11/11 [==============================] - 0s 6ms/step - loss: 34.8819 - val_loss: 23.6709
Epoch 21/30
11/11 [==============================] - 0s 7ms/step - loss: 32.8164 - val_loss: 22.0197
Epoch 22/30
11/11 [==============================] - 0s 7ms/step - loss: 31.4117 - val_loss: 20.3374
Epoch 23/30
11/11 [==============================] - 0s 7ms/step - loss: 30.9939 - val_loss: 19.0910
Epoch 24/30
11/11 [==============================] - 0s 8ms/step - loss: 29.1643 - val_loss: 17.5947
Epoch 25/30
11/11 [==============================] - 0s 8ms/step - loss: 28.0950 - val_loss: 16.9769
Epoch 26/30
11/11 [==============================] - 0s 7ms/step - loss: 28.6024 - val_loss: 17.2496
Epoch 27/30
11/11 [==============================] - 0s 8ms/step - loss: 26.7104 - val_loss: 16.3261
Epoch 28/30
11/11 [==============================] - 0s 7ms/step - loss: 25.9790 - val_loss: 15.4216
Epoch 29/30
11/11 [==============================] - 0s 8ms/step - loss: 26.5143 - val_loss: 14.5859
Epoch 30/30
11/11 [==============================] - 0s 7ms/step - loss: 25.7653 - val_loss: 15.2795
```

- 학습곡선

```
dl_history_plot(hist)
```