1. Tensorflow 버전 확인(사용할 라이브러리 버전 확인)

In [158…

```
# 그래프, 연산처리
import numpy as np # tensorflow와 많이 충돌할 수 있으므로 버전 확인 필수
import pandas as pd
import matplotlib.pyplot as plt

# Tensorflow 처리
import tensorflow as tf
from tensorflow import keras

# 학습 시간 확인
import time

# Tensorflow 모델 설계
from tensorflow.keras import optimizers
from tensorflow.keras.layers import Dense, Input
```

2. 버전 확인

In [22]:

```
# 버전 확인
print("tensorflow : ", tf.__version__)
print("numpy : ", np.__version__)
```

```
tensorflow :  2.16.1
numpy :  1.26.4
```

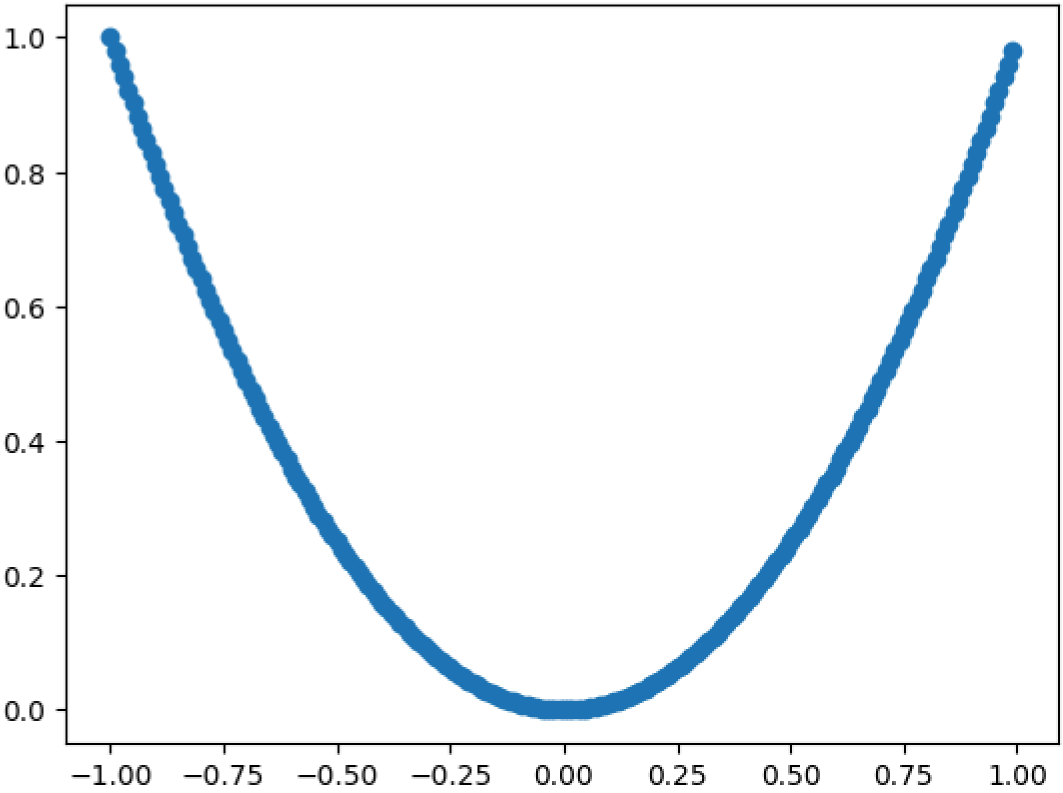
3. 그래프 작성

In [89]:

```
x = np.arange(-1, 1, 0.01)
y = x**2

plt.scatter(x, y)
```

Out[89]:

```
<matplotlib.collections.PathCollection at 0x1d7306a5150>
```



4. 데이터 구조 확인

In [90]:

```
print("데이터 크기(x) : ", len(x))
print("데이터 구조(x) : ", x.shape)
```

```
데이터 크기(x) :  200
데이터 구조(x) :  (200,)
```

5. 간단한 Tensorflow 모델 생성

In [103…

```
# 모델 정의
model = keras.Sequential() # 보통 함수로 생성
model.add(Input([1]))
model.add(Dense(10, activation='tanh')) # 노드가 1개일 경우 학습 불가
model.add(Dense(10, activation='tanh')) # 히든레이어
model.add(Dense(10, activation='tanh')) # 1. 히든레이어가 없는 경우 -> 결과 = 상
model.add(Dense(10, activation='tanh')) # 2. 노드를 절반으로 줄인 경우(10 -> 5)
model.add(Dense(1))
```

In [104…

```
# 컴파일
model.compile(optimizer = 'SGD', loss = 'mse')
```

In [107…

```
# 시간 체크
start_time = time.time() # 시작

# 학습
model.fit(x, y, epochs=500, verbose=0, batch_size=20) # batch_size는 학습에 영향

# 학습 종료
print("학습 시간 : {}".format(time.time() - start_time)) # 현재시간 - 시작시간 =
학습 시간 : 13.661322116851807
```

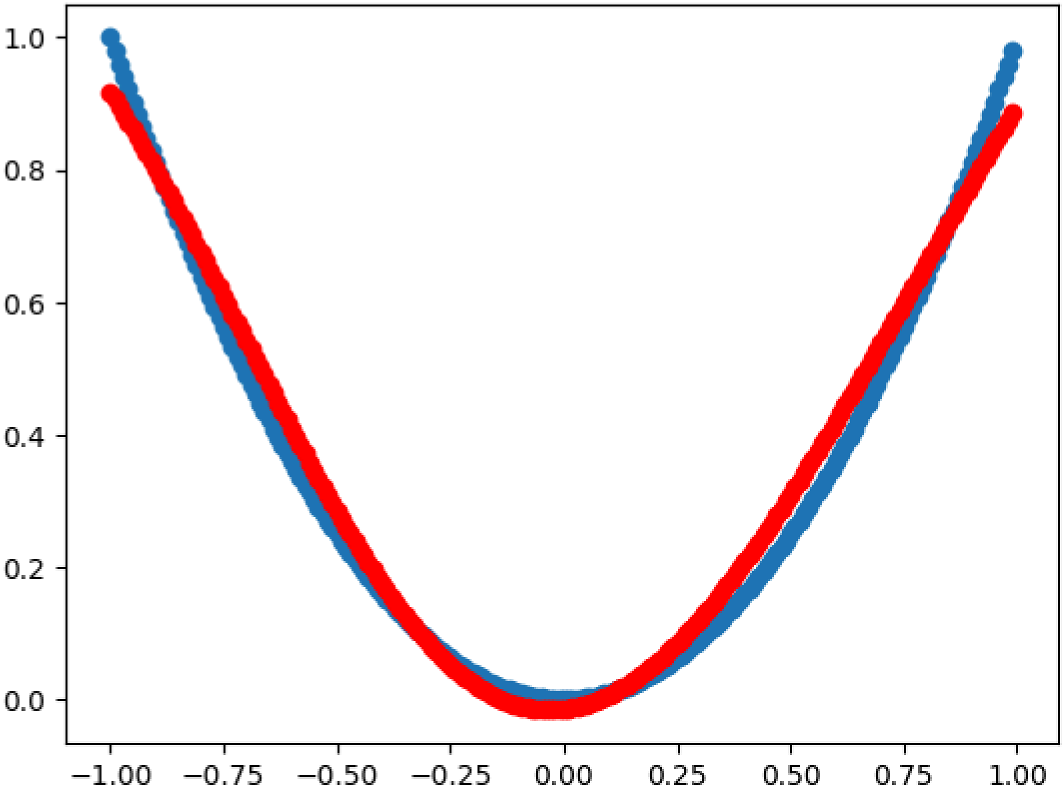
In [108…

```
# 예측
rst = model.predict(x)
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
```

In [109…

```
# 결과를 그래프로 확인
plt.scatter(x, y) # 실제 그래프(정답)
plt.scatter(x, rst, color='r') # 예측 결과값
plt.savefig('wo_hiddenLayer.png') # 그래프를 저장할 때 먼저 저장 후 plt.show()
plt.show()
```



테스트셋으로 평가

In [49]:

```
x = np.arange(-1, 1, 0.01)
np.random.shuffle(x) # 데이터를 섞어야 좋은 결과가 나옴
y = x**2
```

In [55]:

```
# 데이터 분리
# 데이터 양이 많은 경우 -> train, test, valid(validation data)
split_index = int(x.shape[0]*0.7) # 7 : 3으로 나눌 때
split_index
```

Out[55]:

```
140
```

In [57]:

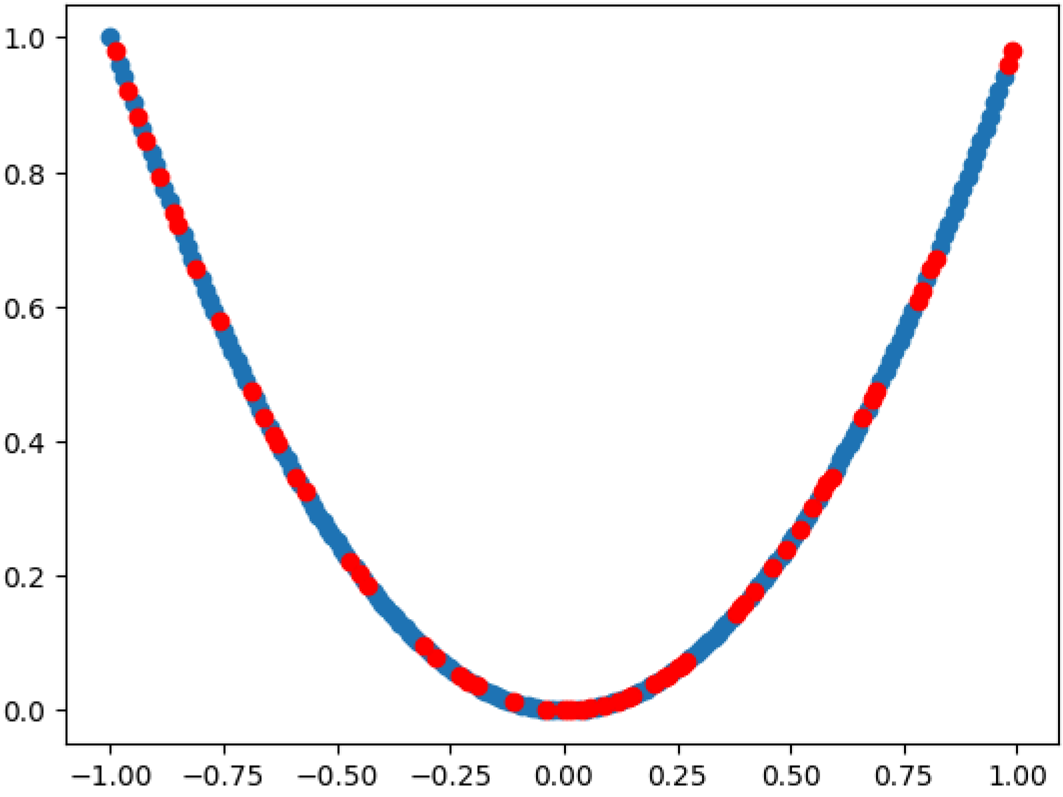
```
# 데이터 분리 처리
train_x, test_x = x[:split_index], x[split_index:]
train_y, test_y = y[:split_index], y[split_index:]
train_x.shape, test_x.shape, train_y.shape, test_y.shape
```

Out[57]:

```
((140,), (60,), (140,), (60,))
```

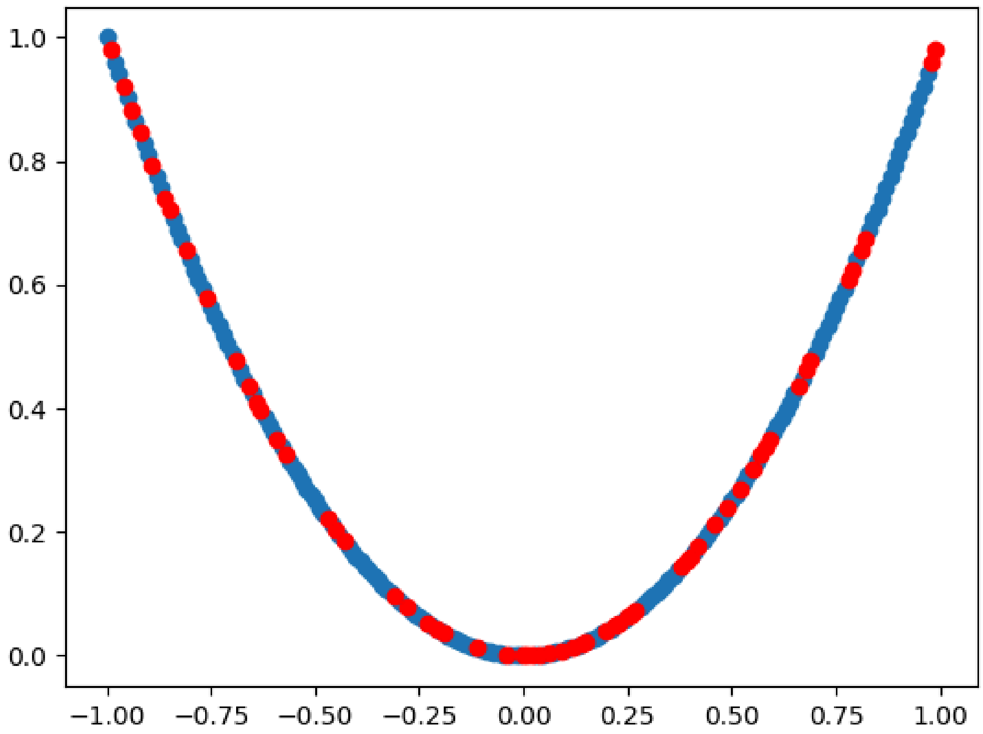
In [67]:

```
# 데이터를 섞었을 때 데이터가 어떻게 표시되는지 확인
plt.scatter(train_x, train_y)
plt.scatter(test_x, test_y, color='r')
# 그래프 저장
plt.savefig('img_test.png') # 그래프를 저장할 때 먼저 저장 후 plt.show()
plt.show()
```



In [72]:

```
# 저장된 이미지를 불러와서 출력
# Ipython -> 1줄씩 출력
from IPython.display import Image
display(Image("test_png/img_test.png"))
```



매직 명령어 사용

In [85]:

```
%time
print("######################################################################
%ls
print("######################################################################
%pwd
```

```
CPU times: total: 0 ns
Wall time: 0 ns
##########################################################################
 D 드라이브의 볼륨: 백업디스크
 볼륨 일련 번호: 080F-4620

 D:\ai_exam\cnn_exam 디렉터리

2024-05-20  오전 11:32    <DIR>          .
2024-05-20  오전 11:32    <DIR>          ..
2024-05-20  오전 11:15    <DIR>          .ipynb_checkpoints
2024-05-17  오후 05:30             1,743 0.jpg
2024-05-17  오후 05:30             1,424 1.jpg
2024-05-17  오후 05:30             1,812 2.jpg
2024-05-17  오후 05:30             1,843 3.jpg
2024-05-17  오후 05:30             2,133 4.jpg
2024-05-17  오후 05:30             1,944 5.jpg
2024-05-17  오후 05:30             2,093 6.jpg
2024-05-17  오후 05:30             1,489 7.jpg
2024-05-17  오후 05:30             2,449 8.jpg
2024-05-17  오후 05:30             2,099 9.jpg
2024-05-20  오전 11:21            23,225 img_test.png
2024-05-17  오후 05:24           773,200 mnist_cnn_20240517_epochs_50_9876.h5
2024-05-17  오후 03:35            32,294 model.png
2024-05-20  오전 09:50    <DIR>          nbtpy
2024-05-17  오후 01:33             2,598 requirement.txt
2024-05-17  오후 05:46           312,536 tensorflow_cnn.ipynb
2024-05-17  오후 01:12           111,006 tensorflow_linear.ipynb
2024-05-20  오전 11:32           327,668 tensorflow_module.ipynb
2024-05-20  오전 09:02                 0 test.py
2024-05-20  오전 11:24    <DIR>          test_png
              18개 파일           1,601,556 바이트
               5개 디렉터리  30,431,707,136 바이트 남음
##########################################################################
```

Out[85]: 'D:\\ai_exam\\cnn_exam'

학습 시간 출력

In [79]:

```
model = keras.Sequential()
model.add(Dense(10, activation='tanh', input_shape=(1,)))
model.add(Dense(10, activation='tanh'))
model.add(Dense(1))

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

```
C:\Users\hi\anaconda3\envs\p310_cnn\lib\site-packages\keras\src\layers\core\dens
e.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [87]:

```
# 시간 체크
start_time = time.time() # 시작

# 모델 학습
model.fit(train_x, train_y, epochs=500, verbose=0, batch_size=20) # verbose -> 0
```

```
# 학습 종료
print("학습 시간 : {}".format(time.time() - start_time)) # 현재시간 - 시작시간 =
```

```
학습 시간 : 12.712069988250732
```

데이터를 sin으로 변경해서 처리

In [111…

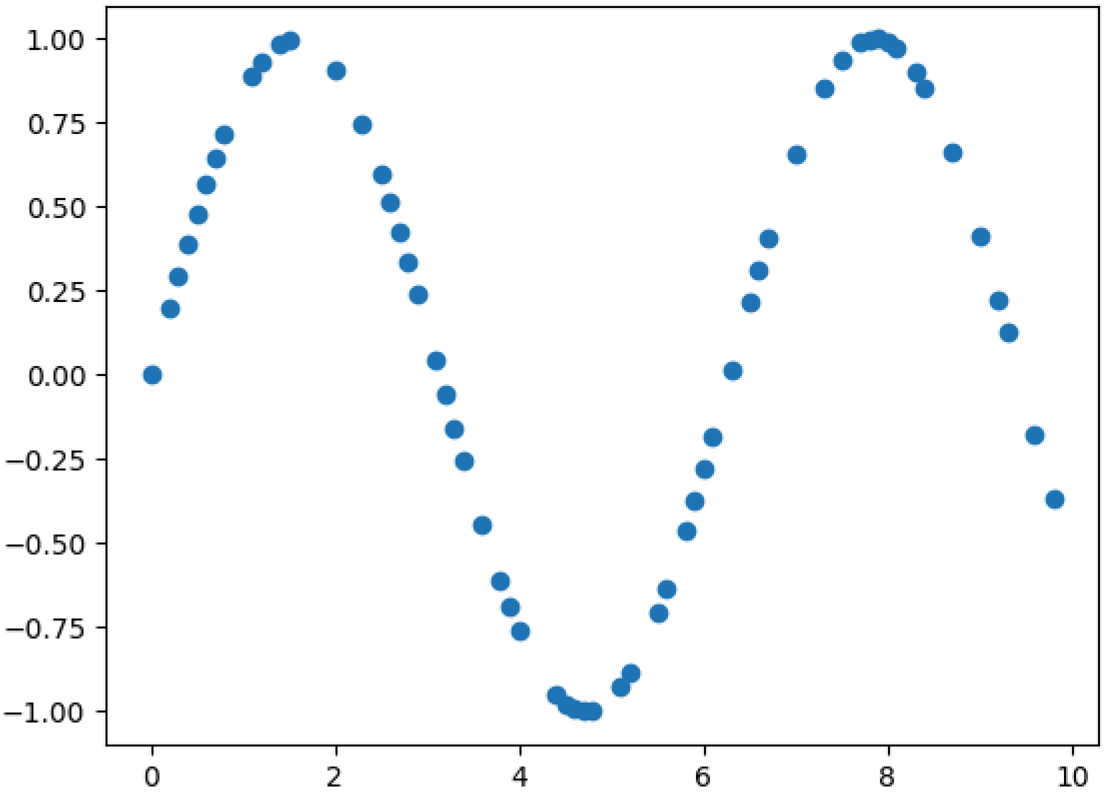
```
# 데이터 sin 처리하는 함수
def get_sin_data(start = 0, end = 10, step = 0.1):
    x = np.arange(start, end, step)
    np.random.shuffle(x)
    y = np.sin(x)

    split_index = int(x.shape[0]*0.6)

    train_x, test_x = x[:split_index], x[split_index:]
    train_y, test_y = y[:split_index], y[split_index:]

    return (train_x, train_y), (test_x, test_y) # 그룹으로 묶어서 return
```

In [113…

```
(train_x, train_y), (test_x, test_y) = get_sin_data(start=0, end=10, step=0.1)
```

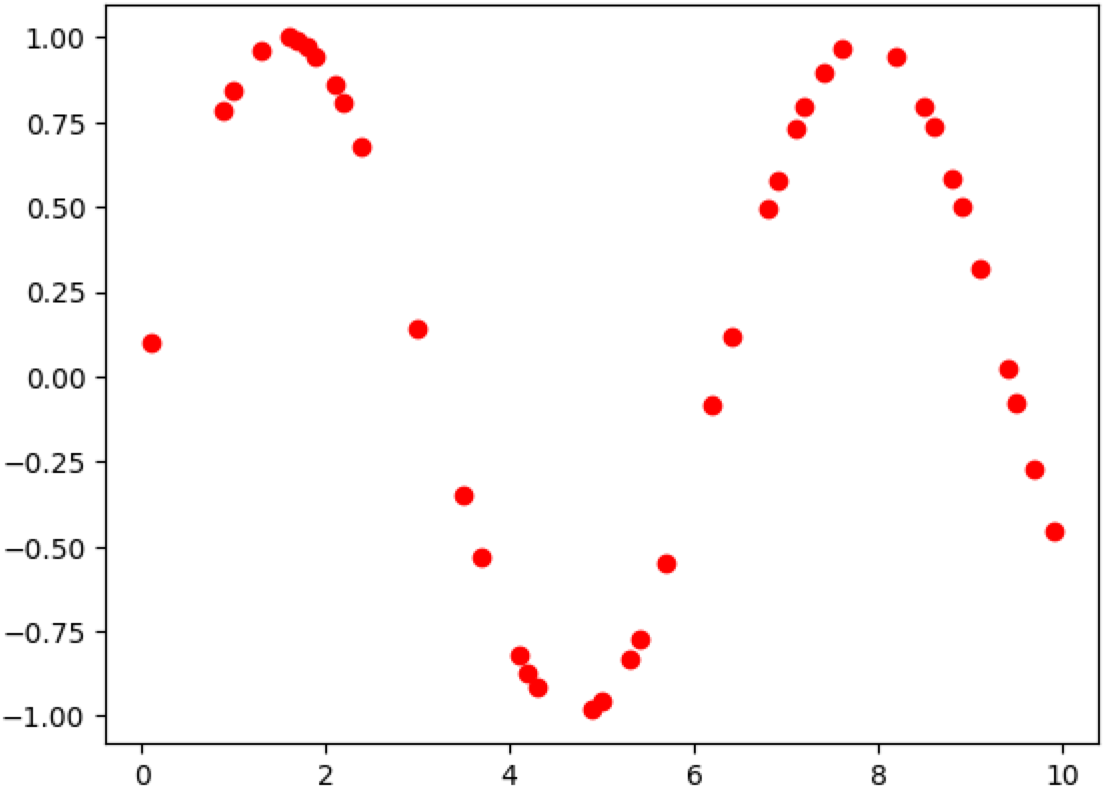
In [114…

```
plt.scatter(train_x, train_y)
plt.show()
```



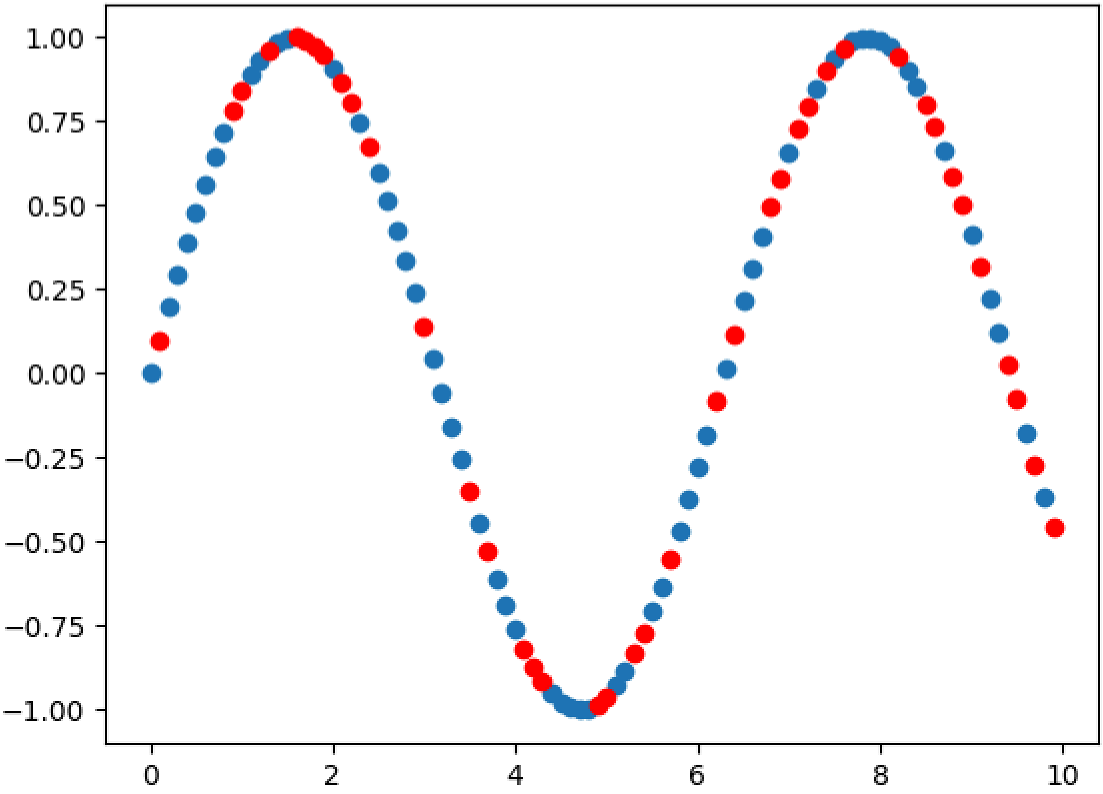
In [116…

```
plt.scatter(test_x, test_y, color = 'r')
plt.show()
```



In [117…

```
plt.scatter(train_x, train_y)
plt.scatter(test_x, test_y, color = 'r')
plt.show()
```



In [119…

```
# 모델 신규 생성
model1 = keras.Sequential()
model1.add(Dense(10, activation='tanh', input_shape=(1,)))
model1.add(Dense(10, activation='tanh'))
model1.add(Dense(1))
```

```
model1.compile(optimizer='SGD', loss='mse', metrics=['mse'])
model1.summary()

start_time = time.time()

model1.fit(train_x, train_y, epochs=1000, verbose=0, batch_size = 20)
print("학습시간 : {}".format(time.time() - start_time))
```

Model: "sequential_14"

Layer (type)	Output Shape
dense_43 (Dense)	(None, 10)
dense_44 (Dense)	(None, 10)
dense_45 (Dense)	(None, 1)

```
Total params: 141 (564.00 B)
Trainable params: 141 (564.00 B)
Non-trainable params: 0 (0.00 B)
학습시간 : 23.009048223495483
```

결과 보기

In [120…

```
loss, mse = model1.evaluate(test_x, test_y)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step - loss: 0.1926 - mse: 0.1926
```

In [121…

```
print("loss = ", loss)
print("mse = ", mse)
```

```
loss =  0.18958239257335663
mse =  0.18958239257335663
```

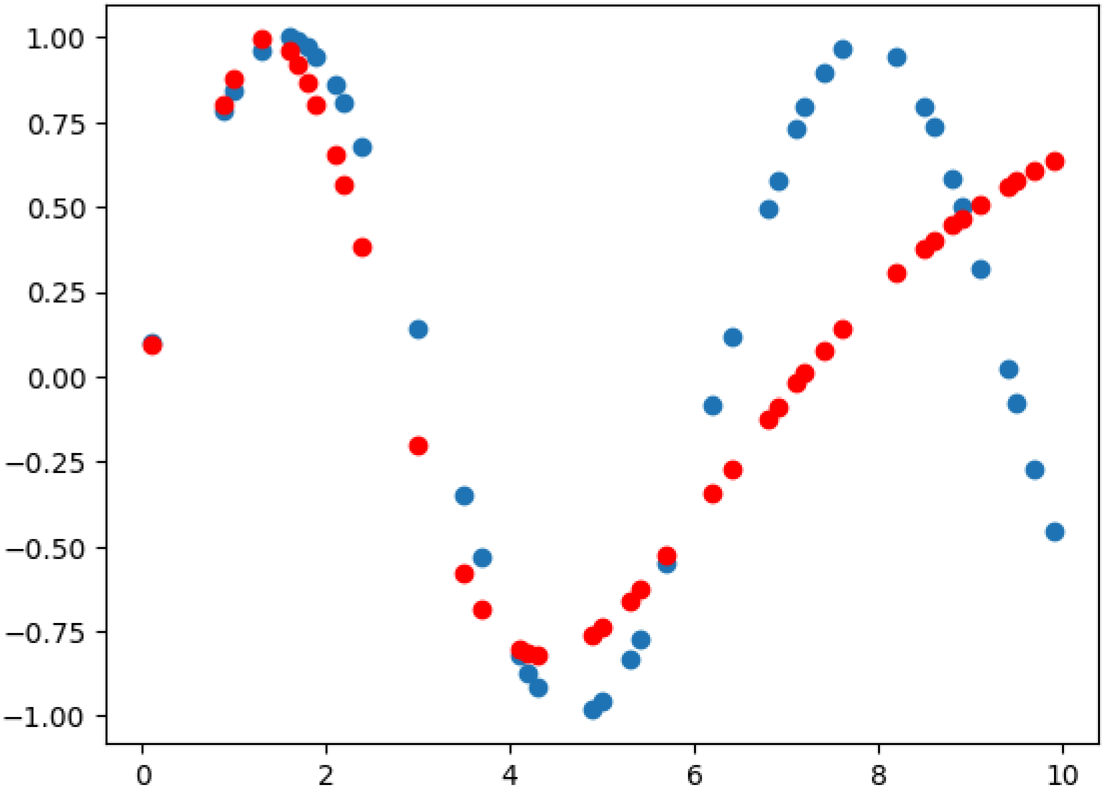
In [122…

```
rst = model1.predict(test_x)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
```

In [123…

```
plt.scatter(test_x, test_y)
plt.scatter(test_x, rst, color = 'r')
plt.show()
```



틀어진 그래프를 조정하여 맞추는 방법

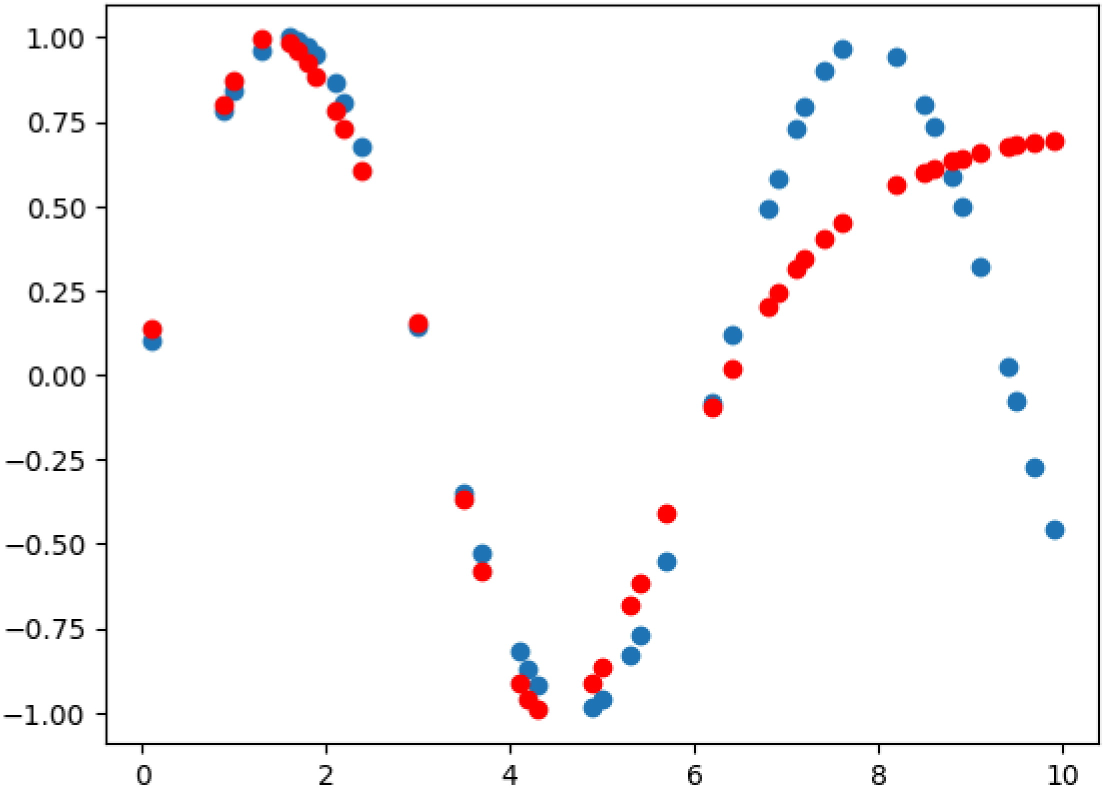
In [124…

```
def fit_one_more(model, train_x, train_y, test_x, test_y, batch_size = 20):
    start_time = time.time()
    model1.fit(train_x, train_y, epochs=1000, verbose=0, batch_size=batch_size)
    print("학습 진행 : {}".format(time.time() - start_time))

    rst = model.predict(test_x)

    plt.scatter(test_x, test_y)
    plt.scatter(test_x, rst, color='r')
    plt.show()

def fit_n_times(model, train_x, train_y, test_x, test_y, n):
    for i in range(n):
        print(f"{i} 번째 학습중...")
        fit_one_more(model, train_x, train_y, test_x, test_y)
```

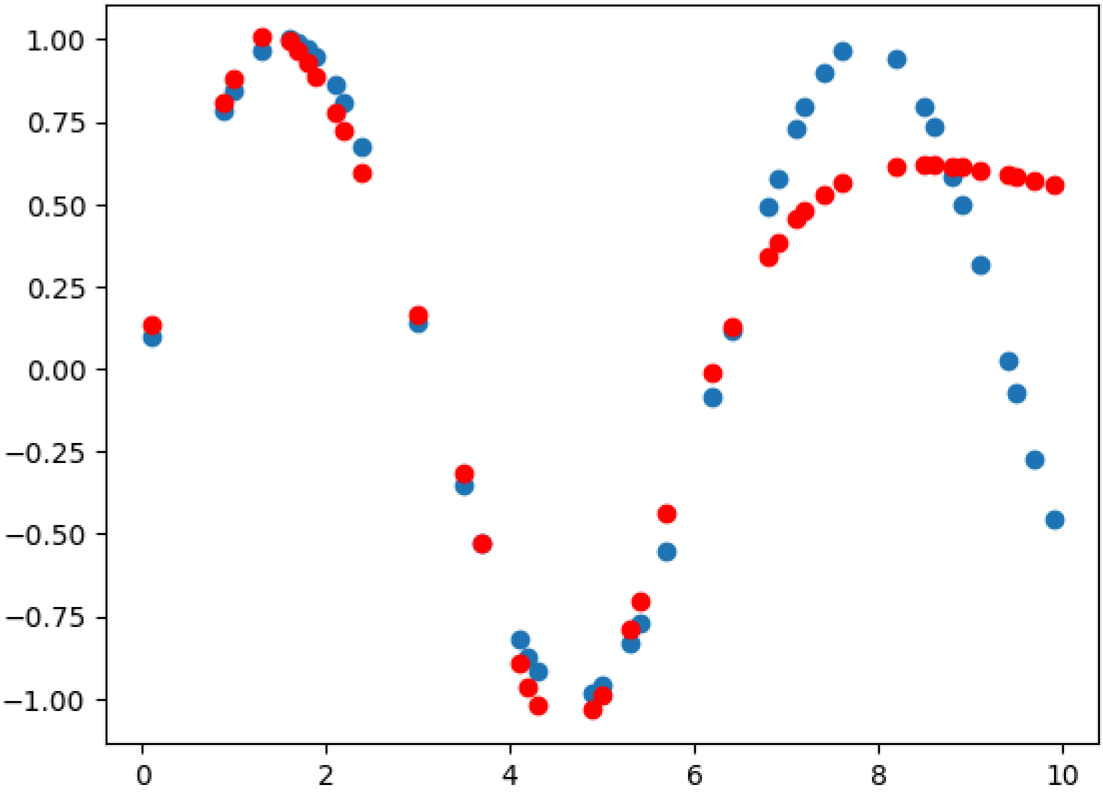
In [125…

```
# 1000번씩 10번 학습하는 과정을 그래프로 표기
fit_n_times(model1, train_x, train_y, test_x, test_y, 10)
```

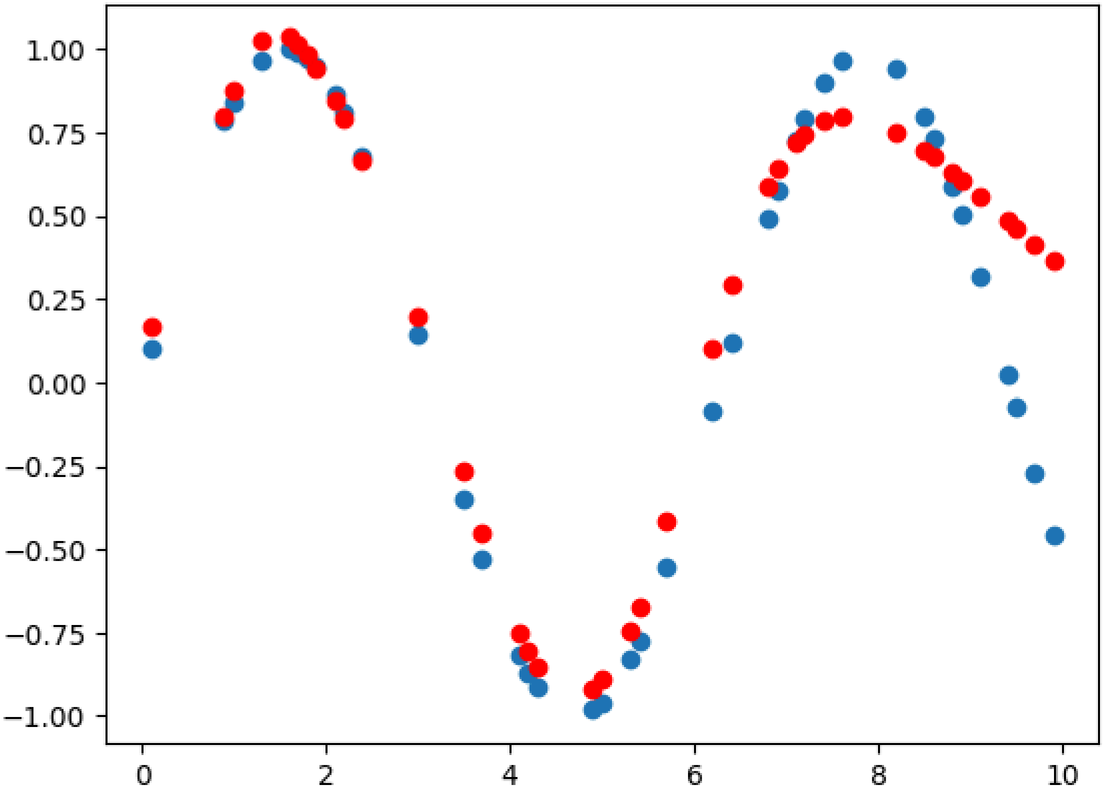
```
0 번째 학습중...
학습 진행 : 22.368649005889893
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```



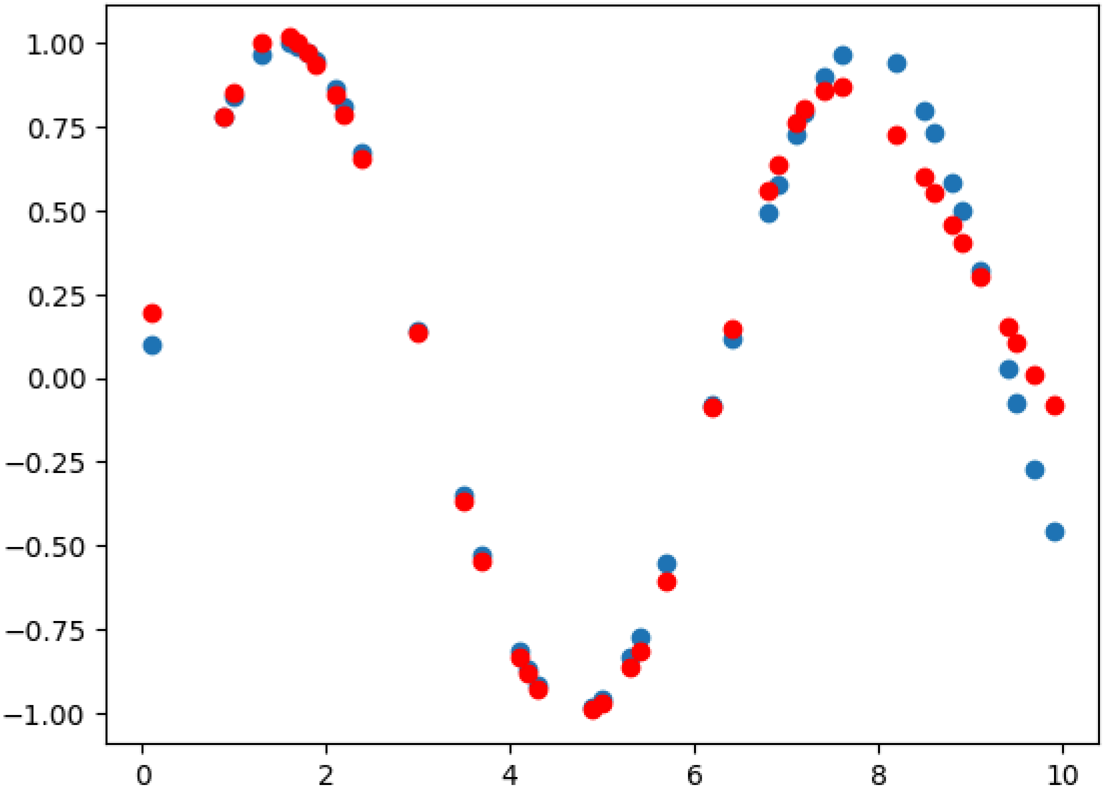
```
1 번째 학습중...
학습 진행 : 23.540597200393677
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
```



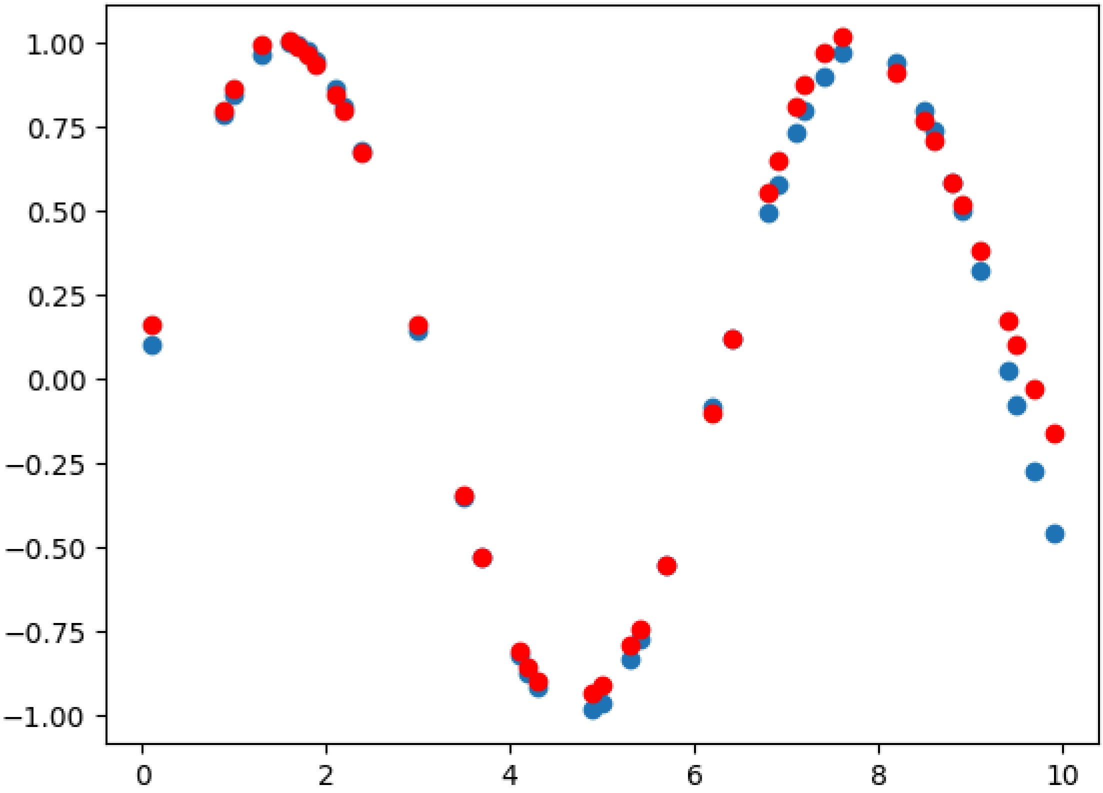
```
2 번째 학습중...
학습 진행 : 23.690056562423706
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
```



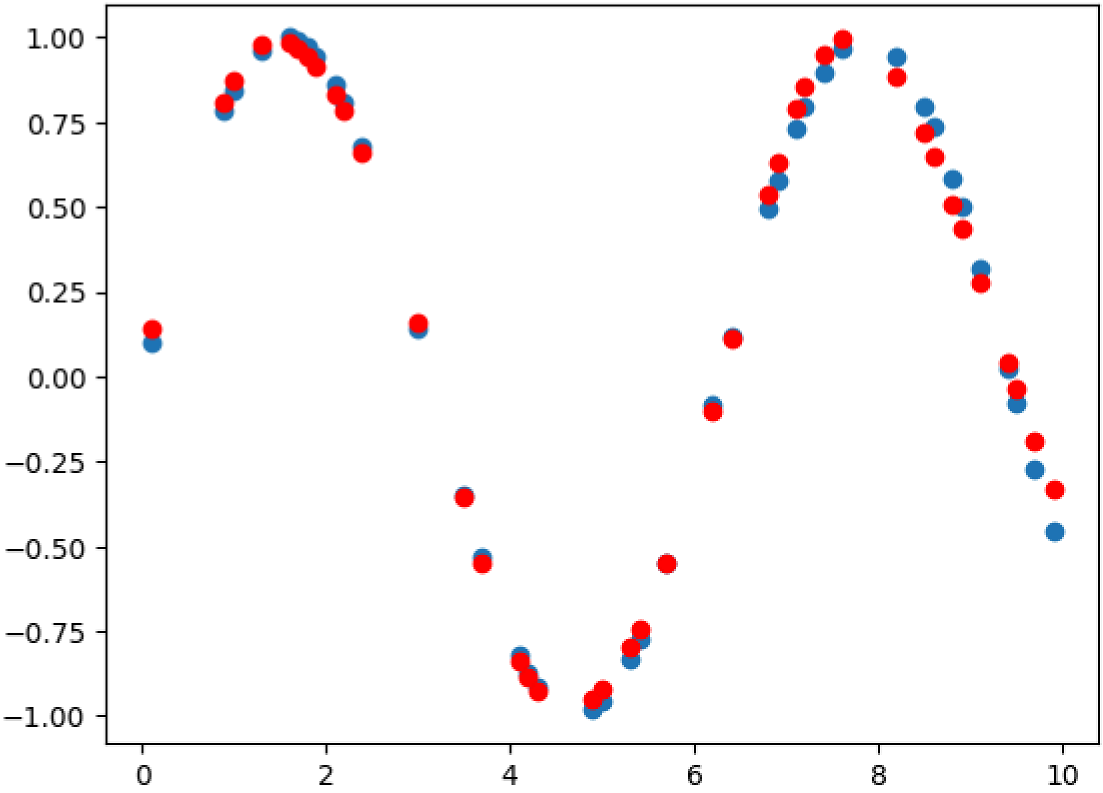
```
3 번째 학습중...
학습 진행 : 23.385226249694824
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
```



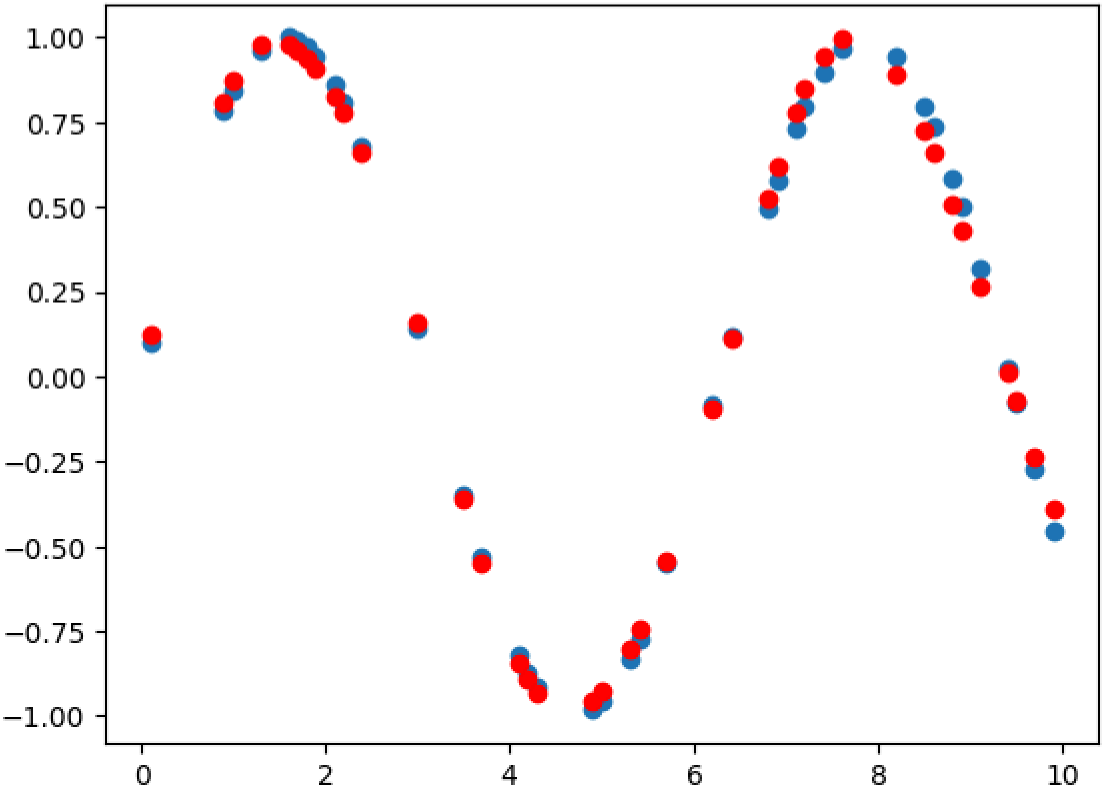
```
4 번째 학습중...
학습 진행 : 24.66965937614441
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```



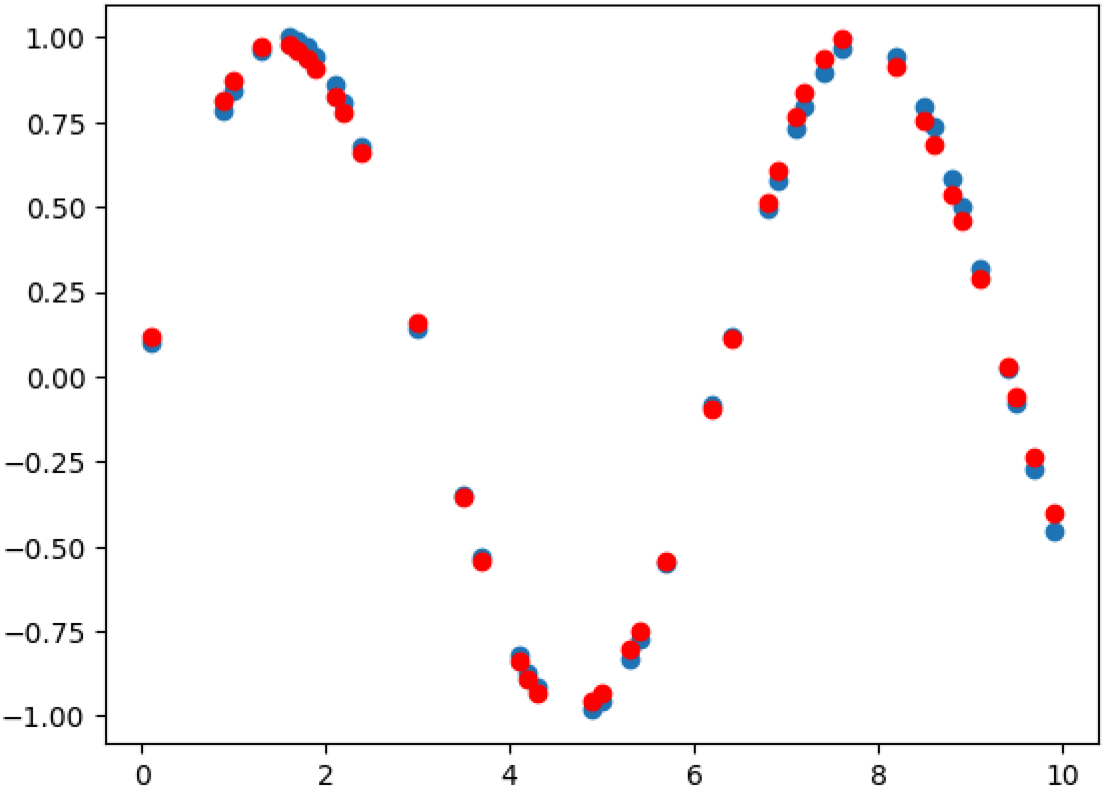
```
5 번째 학습중...
학습 진행 : 25.43707299232483
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
```



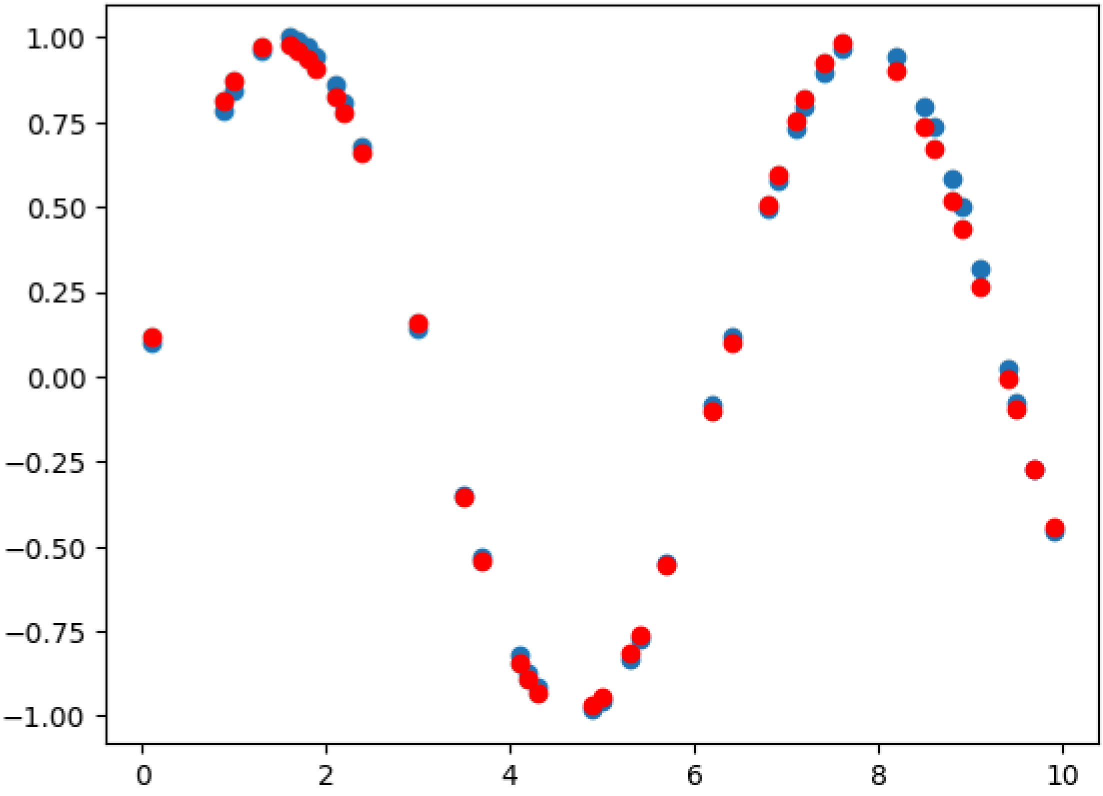
```
6 번째 학습중...
학습 진행 : 25.635162115097046
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
```



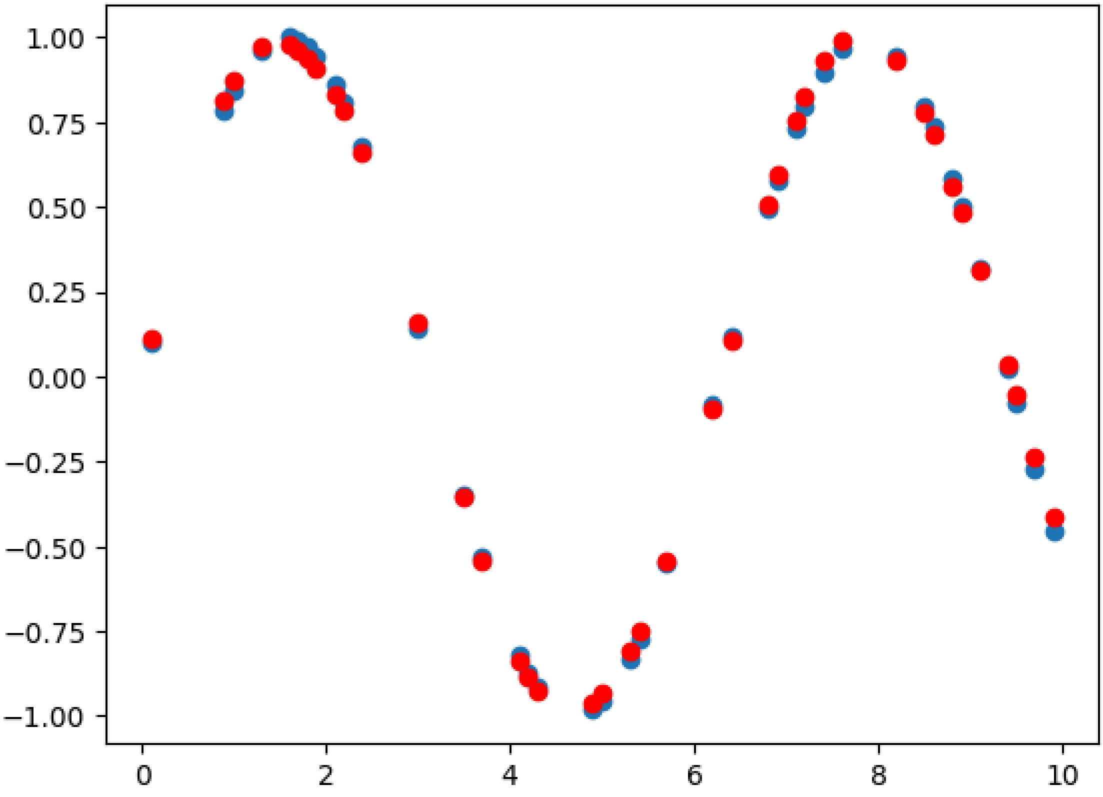
```
7 번째 학습중...
학습 진행 : 25.663201332092285
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
```



```
8 번째 학습중...
학습 진행 : 25.653672218322754
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 0s/step
```



```
9 번째 학습중...
학습 진행 : 24.819753408432007
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```



GPU가 있는 경우[batch 개수가 1, 2, 5, 10, 20, 50, 100]

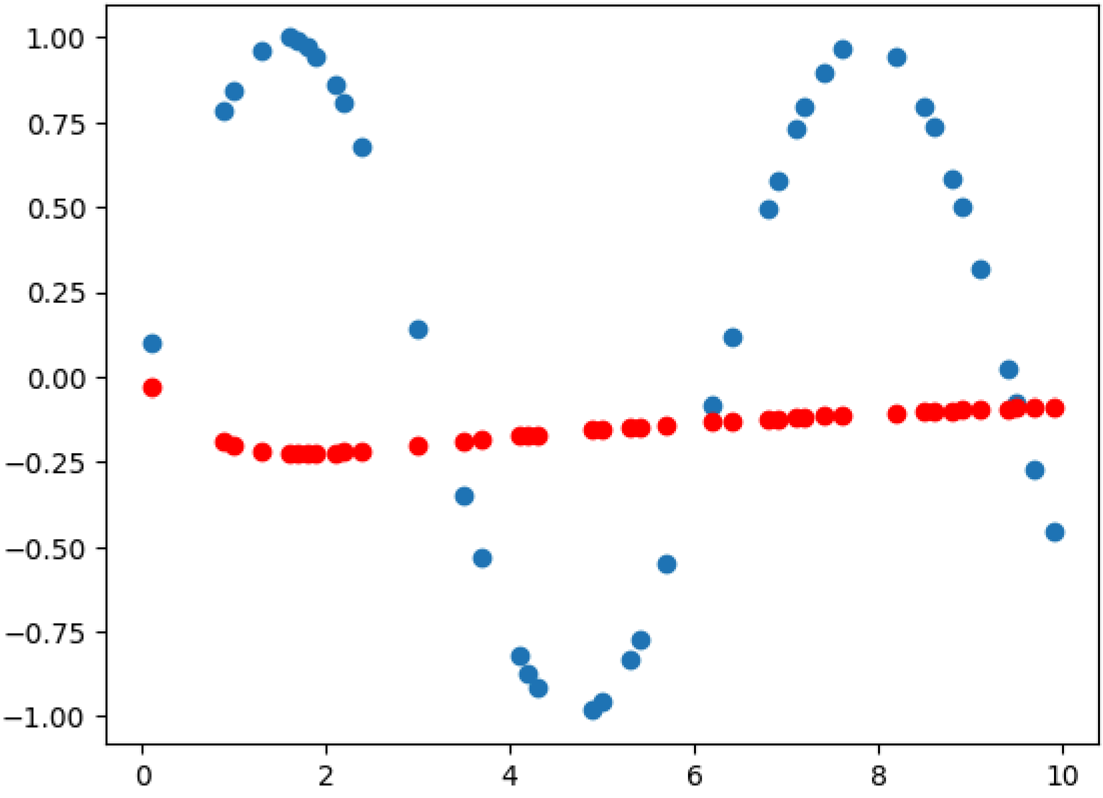
In [138…

```
def build_model():
    model = keras.Sequential()
    model.add(Dense(10, activation='tanh', input_shape=(1,)))
    model.add(Dense(10, activation='tanh'))
    model.add(Dense(1))

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

    return model

def fit_batch_size(train_x, train_y, test_x, test_y, batch_sizes):
    for batch in batch_sizes:
        model = build_model()
        print(f"batch_size 크기 : {batch}")
        fit_one_more(model, train_x, train_y, test_x, test_y, batch)
```

In [139…

```
fit_batch_size(train_x, train_y, test_x, test_y, batch_sizes=[1, 2, 5, 10, 20, 5
```

```
batch_size 크기 : 1
학습 진행 : 64.80388951301575
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
```



```
batch_size 크기 : 2
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[139], line 1
----> 1 fit_batch_size(train_x, train_y, test_x, test_y, batch_sizes=[1, 2, 5, 1
0, 20, 50, 100])

Cell In[138], line 15, in fit_batch_size(train_x, train_y, test_x, test_y, batch_
sizes)
     13 model = build_model()
     14 print(f"batch_size 크기 : {batch}")
---> 15 fit_one_more(model, train_x, train_y, test_x, test_y, batch)

Cell In[124], line 3, in fit_one_more(model, train_x, train_y, test_x, test_y, ba
tch_size)
      1 def fit_one_more(model, train_x, train_y, test_x, test_y, batch_size = 2
0):
      2     start_time = time.time()
----> 3     model1.fit(train_x, train_y, epochs=1000, verbose=0, batch_size=batch
_size)
      4     print("학습 진행 : {}".format(time.time() - start_time))
      6     rst = model.predict(test_x)

File ~\anaconda3\envs\p310_cnn\lib\site-packages\keras\src\utils\traceback_utils.
py:117, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\anaconda3\envs\p310_cnn\lib\site-packages\keras\src\backend\tensorflow\tra
iner.py:312, in TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose, ca
llbacks, validation_split, validation_data, shuffle, class_weight, sample_weight,
initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validati
on_freq)
    310 callbacks.on_epoch_begin(epoch)
    311 with epoch_iterator.catch_stop_iteration():
--> 312     for step, iterator in epoch_iterator.enumerate_epoch():
    313         callbacks.on_train_batch_begin(step)
    314         logs = self.train_function(iterator)

File ~\anaconda3\envs\p310_cnn\lib\site-packages\keras\src\backend\tensorflow\tra
iner.py:645, in TFEpochIterator.enumerate_epoch(self)
    643         yield step, self._current_iterator
    644 else:
--> 645     iterator = iter(self._distributed_dataset)
    646     if self.num_batches:
    647         for step in range(
    648             0, self.num_batches, self.steps_per_execution
    649         ):

File ~\anaconda3\envs\p310_cnn\lib\site-packages\tensorflow\python\data\ops\datas
et_ops.py:501, in DatasetV2.__iter__(self)
    499 if context.executing_eagerly() or ops.inside_function():
    500   with ops.colocate_with(self._variant_tensor):
--> 501     return iterator_ops.OwnedIterator(self)
    502 else:
    503   raise RuntimeError("`tf.data.Dataset` only supports Python-style "
    504                      "iteration in eager mode or within tf.function.")
```

```
File ~\anaconda3\envs\p310_cnn\lib\site-packages\tensorflow\python\data\ops\itera
tor_ops.py:705, in OwnedIterator.__init__(self, dataset, components, element_spe
c)
    701   if (components is not None or element_spec is not None):
    702     raise ValueError(
    703         "When `dataset` is provided, `element_spec` and `components` must
"
    704         "not be specified.")
--> 705   self._create_iterator(dataset)
    707 self._get_next_call_count = 0

File ~\anaconda3\envs\p310_cnn\lib\site-packages\tensorflow\python\data\ops\itera
tor_ops.py:744, in OwnedIterator._create_iterator(self, dataset)
    741   assert len(fulltype.args[0].args[0].args) == len(
    742       self._flat_output_types)
    743   self._iterator_resource.op.experimental_set_type(fulltype)
--> 744 gen_dataset_ops.make_iterator(ds_variant, self._iterator_resource)

File ~\anaconda3\envs\p310_cnn\lib\site-packages\tensorflow\python\ops\gen_datase
t_ops.py:3478, in make_iterator(dataset, iterator, name)
   3476 if tld.is_eager:
   3477   try:
-> 3478     _result = pywrap_tfe.TFE_Py_FastPathExecute(
   3479         _ctx, "MakeIterator", name, dataset, iterator)
   3480     return _result
   3481   except _core._NotOkStatusException as e:

KeyboardInterrupt:
```

데이터 크기가 영향을 주는지 확인

In [ ]:

```
(train_x, train_y), (test_x, test_y) = get_sin_data(start=0, end=10, step=0.1)
```

In [143…

```
# 모델 신규 생성
model2 = keras.Sequential()
model2.add(Dense(10, activation='tanh', input_shape=(1,)))
model2.add(Dense(10, activation='tanh'))
model2.add(Dense(1))

model2.compile(optimizer='SGD', loss='mse', metrics=['mse'])

start_time = time.time()
history = model2.fit(train_x, train_y, epochs=100, verbose=2, batch_size=20)
print("학습 시간 : {}".format(time.time() - start_time))
```

```
Epoch 1/100
3/3 - 0s - 114ms/step - loss: 1.4838 - mse: 1.4838
Epoch 2/100
3/3 - 0s - 9ms/step - loss: 0.6483 - mse: 0.6483
Epoch 3/100
3/3 - 0s - 11ms/step - loss: 0.4821 - mse: 0.4821
Epoch 4/100
3/3 - 0s - 11ms/step - loss: 0.4426 - mse: 0.4426
Epoch 5/100
3/3 - 0s - 10ms/step - loss: 0.4261 - mse: 0.4261
Epoch 6/100
3/3 - 0s - 6ms/step - loss: 0.4219 - mse: 0.4219
Epoch 7/100
3/3 - 0s - 6ms/step - loss: 0.4225 - mse: 0.4225
Epoch 8/100
3/3 - 0s - 11ms/step - loss: 0.4172 - mse: 0.4172
Epoch 9/100
3/3 - 0s - 28ms/step - loss: 0.4151 - mse: 0.4151
Epoch 10/100
3/3 - 0s - 9ms/step - loss: 0.4159 - mse: 0.4159
Epoch 11/100
3/3 - 0s - 11ms/step - loss: 0.4101 - mse: 0.4101
Epoch 12/100
3/3 - 0s - 11ms/step - loss: 0.4116 - mse: 0.4116
Epoch 13/100
3/3 - 0s - 11ms/step - loss: 0.4081 - mse: 0.4081
Epoch 14/100
3/3 - 0s - 11ms/step - loss: 0.4051 - mse: 0.4051
Epoch 15/100
3/3 - 0s - 11ms/step - loss: 0.4035 - mse: 0.4035
Epoch 16/100
3/3 - 0s - 11ms/step - loss: 0.4044 - mse: 0.4044
Epoch 17/100
3/3 - 0s - 11ms/step - loss: 0.4006 - mse: 0.4006
Epoch 18/100
3/3 - 0s - 11ms/step - loss: 0.3980 - mse: 0.3980
Epoch 19/100
3/3 - 0s - 11ms/step - loss: 0.4018 - mse: 0.4018
Epoch 20/100
3/3 - 0s - 10ms/step - loss: 0.3972 - mse: 0.3972
Epoch 21/100
3/3 - 0s - 10ms/step - loss: 0.3929 - mse: 0.3929
Epoch 22/100
3/3 - 0s - 6ms/step - loss: 0.3922 - mse: 0.3922
Epoch 23/100
3/3 - 0s - 7ms/step - loss: 0.3904 - mse: 0.3904
Epoch 24/100
3/3 - 0s - 11ms/step - loss: 0.3910 - mse: 0.3910
Epoch 25/100
3/3 - 0s - 7ms/step - loss: 0.3927 - mse: 0.3927
Epoch 26/100
3/3 - 0s - 7ms/step - loss: 0.3874 - mse: 0.3874
Epoch 27/100
3/3 - 0s - 8ms/step - loss: 0.3903 - mse: 0.3903
Epoch 28/100
3/3 - 0s - 11ms/step - loss: 0.3846 - mse: 0.3846
Epoch 29/100
3/3 - 0s - 11ms/step - loss: 0.3945 - mse: 0.3945
Epoch 30/100
3/3 - 0s - 10ms/step - loss: 0.3854 - mse: 0.3854
```

```
Epoch 31/100
3/3 - 0s - 6ms/step - loss: 0.3858 - mse: 0.3858
Epoch 32/100
3/3 - 0s - 6ms/step - loss: 0.3824 - mse: 0.3824
Epoch 33/100
3/3 - 0s - 12ms/step - loss: 0.3852 - mse: 0.3852
Epoch 34/100
3/3 - 0s - 11ms/step - loss: 0.3821 - mse: 0.3821
Epoch 35/100
3/3 - 0s - 11ms/step - loss: 0.3773 - mse: 0.3773
Epoch 36/100
3/3 - 0s - 11ms/step - loss: 0.3854 - mse: 0.3854
Epoch 37/100
3/3 - 0s - 11ms/step - loss: 0.3783 - mse: 0.3783
Epoch 38/100
3/3 - 0s - 6ms/step - loss: 0.3743 - mse: 0.3743
Epoch 39/100
3/3 - 0s - 6ms/step - loss: 0.3728 - mse: 0.3728
Epoch 40/100
3/3 - 0s - 7ms/step - loss: 0.3733 - mse: 0.3733
Epoch 41/100
3/3 - 0s - 11ms/step - loss: 0.3743 - mse: 0.3743
Epoch 42/100
3/3 - 0s - 11ms/step - loss: 0.3772 - mse: 0.3772
Epoch 43/100
3/3 - 0s - 6ms/step - loss: 0.3736 - mse: 0.3736
Epoch 44/100
3/3 - 0s - 10ms/step - loss: 0.3739 - mse: 0.3739
Epoch 45/100
3/3 - 0s - 6ms/step - loss: 0.3688 - mse: 0.3688
Epoch 46/100
3/3 - 0s - 6ms/step - loss: 0.3728 - mse: 0.3728
Epoch 47/100
3/3 - 0s - 11ms/step - loss: 0.3681 - mse: 0.3681
Epoch 48/100
3/3 - 0s - 11ms/step - loss: 0.3669 - mse: 0.3669
Epoch 49/100
3/3 - 0s - 11ms/step - loss: 0.3656 - mse: 0.3656
Epoch 50/100
3/3 - 0s - 10ms/step - loss: 0.3672 - mse: 0.3672
Epoch 51/100
3/3 - 0s - 11ms/step - loss: 0.3649 - mse: 0.3649
Epoch 52/100
3/3 - 0s - 6ms/step - loss: 0.3696 - mse: 0.3696
Epoch 53/100
3/3 - 0s - 6ms/step - loss: 0.3651 - mse: 0.3651
Epoch 54/100
3/3 - 0s - 12ms/step - loss: 0.3649 - mse: 0.3649
Epoch 55/100
3/3 - 0s - 12ms/step - loss: 0.3626 - mse: 0.3626
Epoch 56/100
3/3 - 0s - 10ms/step - loss: 0.3649 - mse: 0.3649
Epoch 57/100
3/3 - 0s - 10ms/step - loss: 0.3690 - mse: 0.3690
Epoch 58/100
3/3 - 0s - 5ms/step - loss: 0.3614 - mse: 0.3614
Epoch 59/100
3/3 - 0s - 11ms/step - loss: 0.3654 - mse: 0.3654
Epoch 60/100
3/3 - 0s - 4ms/step - loss: 0.3647 - mse: 0.3647
```

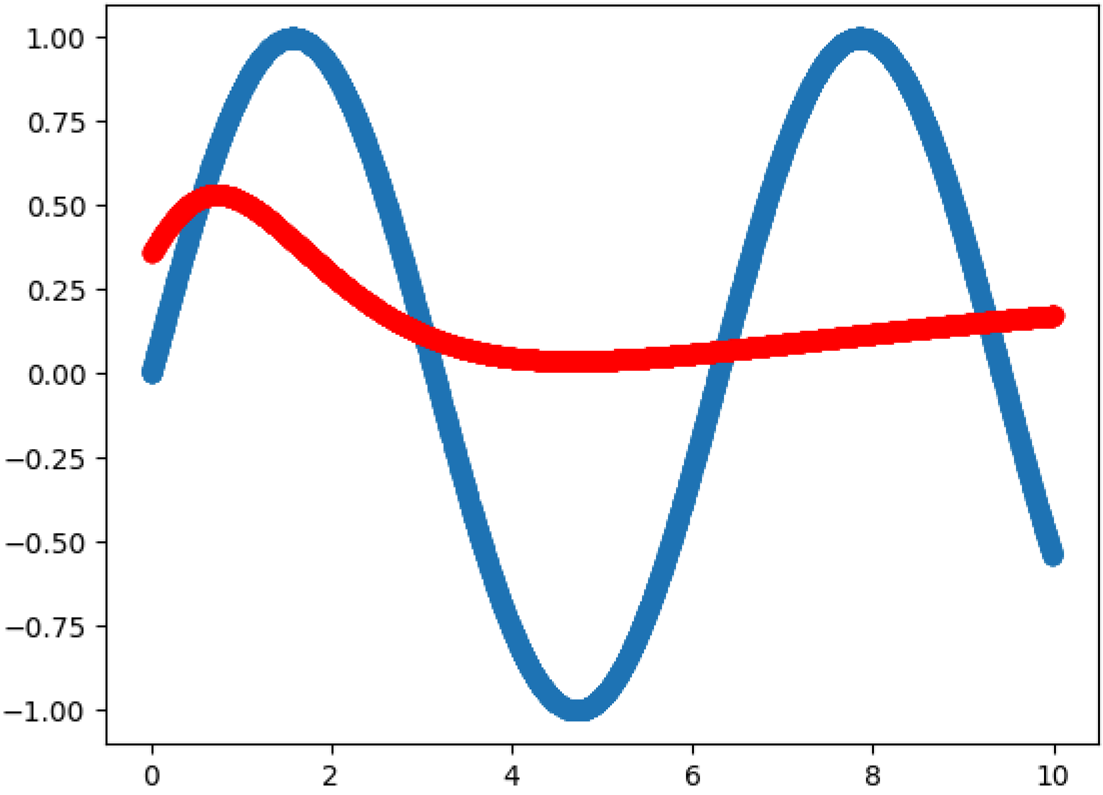
```
Epoch 61/100
3/3 - 0s - 11ms/step - loss: 0.3647 - mse: 0.3647
Epoch 62/100
3/3 - 0s - 11ms/step - loss: 0.3620 - mse: 0.3620
Epoch 63/100
3/3 - 0s - 4ms/step - loss: 0.3606 - mse: 0.3606
Epoch 64/100
3/3 - 0s - 6ms/step - loss: 0.3598 - mse: 0.3598
Epoch 65/100
3/3 - 0s - 12ms/step - loss: 0.3589 - mse: 0.3589
Epoch 66/100
3/3 - 0s - 4ms/step - loss: 0.3580 - mse: 0.3580
Epoch 67/100
3/3 - 0s - 6ms/step - loss: 0.3579 - mse: 0.3579
Epoch 68/100
3/3 - 0s - 12ms/step - loss: 0.3554 - mse: 0.3554
Epoch 69/100
3/3 - 0s - 4ms/step - loss: 0.3620 - mse: 0.3620
Epoch 70/100
3/3 - 0s - 6ms/step - loss: 0.3568 - mse: 0.3568
Epoch 71/100
3/3 - 0s - 12ms/step - loss: 0.3554 - mse: 0.3554
Epoch 72/100
3/3 - 0s - 4ms/step - loss: 0.3550 - mse: 0.3550
Epoch 73/100
3/3 - 0s - 24ms/step - loss: 0.3603 - mse: 0.3603
Epoch 74/100
3/3 - 0s - 13ms/step - loss: 0.3548 - mse: 0.3548
Epoch 75/100
3/3 - 0s - 5ms/step - loss: 0.3520 - mse: 0.3520
Epoch 76/100
3/3 - 0s - 6ms/step - loss: 0.3526 - mse: 0.3526
Epoch 77/100
3/3 - 0s - 11ms/step - loss: 0.3524 - mse: 0.3524
Epoch 78/100
3/3 - 0s - 11ms/step - loss: 0.3574 - mse: 0.3574
Epoch 79/100
3/3 - 0s - 11ms/step - loss: 0.3506 - mse: 0.3506
Epoch 80/100
3/3 - 0s - 5ms/step - loss: 0.3506 - mse: 0.3506
Epoch 81/100
3/3 - 0s - 11ms/step - loss: 0.3506 - mse: 0.3506
Epoch 82/100
3/3 - 0s - 11ms/step - loss: 0.3506 - mse: 0.3506
Epoch 83/100
3/3 - 0s - 4ms/step - loss: 0.3486 - mse: 0.3486
Epoch 84/100
3/3 - 0s - 64ms/step - loss: 0.3541 - mse: 0.3541
Epoch 85/100
3/3 - 0s - 23ms/step - loss: 0.3511 - mse: 0.3511
Epoch 86/100
3/3 - 0s - 13ms/step - loss: 0.3486 - mse: 0.3486
Epoch 87/100
3/3 - 0s - 28ms/step - loss: 0.3594 - mse: 0.3594
Epoch 88/100
3/3 - 0s - 4ms/step - loss: 0.3525 - mse: 0.3525
Epoch 89/100
3/3 - 0s - 6ms/step - loss: 0.3471 - mse: 0.3471
Epoch 90/100
3/3 - 0s - 11ms/step - loss: 0.3460 - mse: 0.3460
```

```
Epoch 91/100
3/3 - 0s - 11ms/step - loss: 0.3475 - mse: 0.3475
Epoch 92/100
3/3 - 0s - 4ms/step - loss: 0.3457 - mse: 0.3457
Epoch 93/100
3/3 - 0s - 6ms/step - loss: 0.3513 - mse: 0.3513
Epoch 94/100
3/3 - 0s - 11ms/step - loss: 0.3448 - mse: 0.3448
Epoch 95/100
3/3 - 0s - 5ms/step - loss: 0.3443 - mse: 0.3443
Epoch 96/100
3/3 - 0s - 11ms/step - loss: 0.3495 - mse: 0.3495
Epoch 97/100
3/3 - 0s - 8ms/step - loss: 0.3458 - mse: 0.3458
Epoch 98/100
3/3 - 0s - 5ms/step - loss: 0.3432 - mse: 0.3432
Epoch 99/100
3/3 - 0s - 11ms/step - loss: 0.3474 - mse: 0.3474
Epoch 100/100
3/3 - 0s - 5ms/step - loss: 0.3456 - mse: 0.3456
학습 시간 : 3.764402389526367
```

In [148…

```python
rst = model2.predict(test_x)

plt.scatter(train_x, train_y)
plt.scatter(test_x, rst, color = 'r')
plt.show()
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 760us/step
```



In [144…

```python
(train_x, train_y), (test_x, test_y) = get_sin_data(start=0, end=10, step=0.001)
```

In [145…

```python
# 모델 신규 생성
model3 = keras.Sequential()
model3.add(Dense(10, activation='tanh', input_shape=(1,)))
model3.add(Dense(10, activation='tanh'))
```

```
model3.add(Dense(1))

model3.compile(optimizer='SGD', loss='mse', metrics=['mse'])

start_time = time.time()
history = model3.fit(train_x, train_y, epochs=100, verbose=2, batch_size=20)
print("학습 시간 : {}".format(time.time() - start_time))
```

```
Epoch 1/100
300/300 - 1s - 2ms/step - loss: 0.4249 - mse: 0.4249
Epoch 2/100
300/300 - 0s - 849us/step - loss: 0.3738 - mse: 0.3738
Epoch 3/100
300/300 - 0s - 819us/step - loss: 0.3455 - mse: 0.3455
Epoch 4/100
300/300 - 0s - 834us/step - loss: 0.3162 - mse: 0.3162
Epoch 5/100
300/300 - 0s - 848us/step - loss: 0.2820 - mse: 0.2820
Epoch 6/100
300/300 - 0s - 816us/step - loss: 0.2525 - mse: 0.2525
Epoch 7/100
300/300 - 0s - 828us/step - loss: 0.2272 - mse: 0.2272
Epoch 8/100
300/300 - 0s - 811us/step - loss: 0.2048 - mse: 0.2048
Epoch 9/100
300/300 - 0s - 801us/step - loss: 0.1839 - mse: 0.1839
Epoch 10/100
300/300 - 0s - 821us/step - loss: 0.1708 - mse: 0.1708
Epoch 11/100
300/300 - 0s - 796us/step - loss: 0.1564 - mse: 0.1564
Epoch 12/100
300/300 - 0s - 770us/step - loss: 0.1392 - mse: 0.1392
Epoch 13/100
300/300 - 0s - 759us/step - loss: 0.1291 - mse: 0.1291
Epoch 14/100
300/300 - 0s - 851us/step - loss: 0.1196 - mse: 0.1196
Epoch 15/100
300/300 - 0s - 748us/step - loss: 0.1117 - mse: 0.1117
Epoch 16/100
300/300 - 0s - 809us/step - loss: 0.1039 - mse: 0.1039
Epoch 17/100
300/300 - 0s - 747us/step - loss: 0.0960 - mse: 0.0960
Epoch 18/100
300/300 - 0s - 778us/step - loss: 0.0894 - mse: 0.0894
Epoch 19/100
300/300 - 0s - 812us/step - loss: 0.0843 - mse: 0.0843
Epoch 20/100
300/300 - 0s - 800us/step - loss: 0.0795 - mse: 0.0795
Epoch 21/100
300/300 - 0s - 809us/step - loss: 0.0729 - mse: 0.0729
Epoch 22/100
300/300 - 0s - 802us/step - loss: 0.0706 - mse: 0.0706
Epoch 23/100
300/300 - 0s - 858us/step - loss: 0.0594 - mse: 0.0594
Epoch 24/100
300/300 - 0s - 749us/step - loss: 0.0567 - mse: 0.0567
Epoch 25/100
300/300 - 0s - 803us/step - loss: 0.0506 - mse: 0.0506
Epoch 26/100
300/300 - 0s - 771us/step - loss: 0.0484 - mse: 0.0484
Epoch 27/100
300/300 - 0s - 806us/step - loss: 0.0417 - mse: 0.0417
Epoch 28/100
300/300 - 0s - 821us/step - loss: 0.0367 - mse: 0.0367
Epoch 29/100
300/300 - 0s - 943us/step - loss: 0.0327 - mse: 0.0327
Epoch 30/100
300/300 - 0s - 953us/step - loss: 0.0275 - mse: 0.0275
```

```
Epoch 31/100
300/300 - 0s - 882us/step - loss: 0.0262 - mse: 0.0262
Epoch 32/100
300/300 - 0s - 771us/step - loss: 0.0239 - mse: 0.0239
Epoch 33/100
300/300 - 0s - 833us/step - loss: 0.0190 - mse: 0.0190
Epoch 34/100
300/300 - 0s - 839us/step - loss: 0.0181 - mse: 0.0181
Epoch 35/100
300/300 - 0s - 794us/step - loss: 0.0174 - mse: 0.0174
Epoch 36/100
300/300 - 0s - 785us/step - loss: 0.0143 - mse: 0.0143
Epoch 37/100
300/300 - 0s - 832us/step - loss: 0.0139 - mse: 0.0139
Epoch 38/100
300/300 - 0s - 873us/step - loss: 0.0112 - mse: 0.0112
Epoch 39/100
300/300 - 0s - 867us/step - loss: 0.0109 - mse: 0.0109
Epoch 40/100
300/300 - 0s - 843us/step - loss: 0.0097 - mse: 0.0097
Epoch 41/100
300/300 - 0s - 803us/step - loss: 0.0081 - mse: 0.0081
Epoch 42/100
300/300 - 0s - 778us/step - loss: 0.0102 - mse: 0.0102
Epoch 43/100
300/300 - 0s - 792us/step - loss: 0.0081 - mse: 0.0081
Epoch 44/100
300/300 - 0s - 864us/step - loss: 0.0100 - mse: 0.0100
Epoch 45/100
300/300 - 0s - 862us/step - loss: 0.0092 - mse: 0.0092
Epoch 46/100
300/300 - 0s - 919us/step - loss: 0.0088 - mse: 0.0088
Epoch 47/100
300/300 - 0s - 947us/step - loss: 0.0069 - mse: 0.0069
Epoch 48/100
300/300 - 0s - 881us/step - loss: 0.0075 - mse: 0.0075
Epoch 49/100
300/300 - 0s - 844us/step - loss: 0.0076 - mse: 0.0076
Epoch 50/100
300/300 - 0s - 1ms/step - loss: 0.0080 - mse: 0.0080
Epoch 51/100
300/300 - 0s - 897us/step - loss: 0.0065 - mse: 0.0065
Epoch 52/100
300/300 - 0s - 777us/step - loss: 0.0092 - mse: 0.0092
Epoch 53/100
300/300 - 0s - 937us/step - loss: 0.0057 - mse: 0.0057
Epoch 54/100
300/300 - 0s - 893us/step - loss: 0.0067 - mse: 0.0067
Epoch 55/100
300/300 - 0s - 837us/step - loss: 0.0081 - mse: 0.0081
Epoch 56/100
300/300 - 0s - 889us/step - loss: 0.0063 - mse: 0.0063
Epoch 57/100
300/300 - 0s - 827us/step - loss: 0.0053 - mse: 0.0053
Epoch 58/100
300/300 - 0s - 780us/step - loss: 0.0095 - mse: 0.0095
Epoch 59/100
300/300 - 0s - 833us/step - loss: 0.0042 - mse: 0.0042
Epoch 60/100
300/300 - 0s - 778us/step - loss: 0.0077 - mse: 0.0077
```

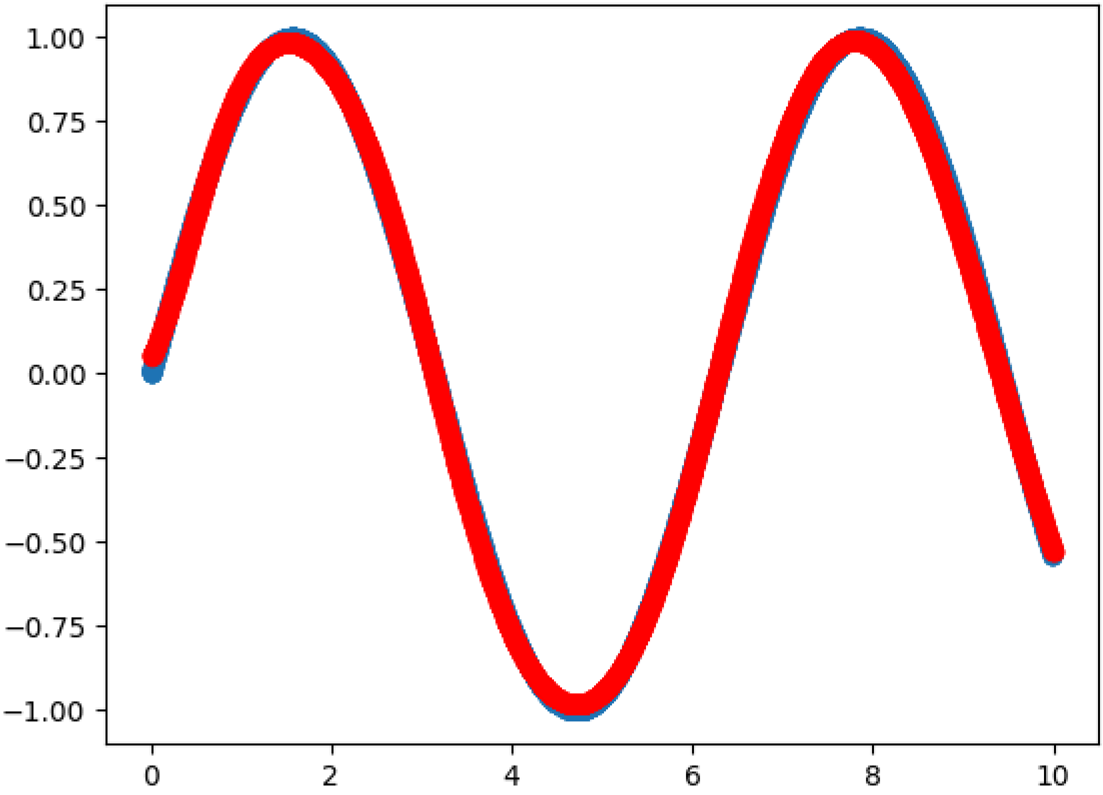
```
Epoch 61/100
300/300 - 0s - 834us/step - loss: 0.0040 - mse: 0.0040
Epoch 62/100
300/300 - 0s - 773us/step - loss: 0.0053 - mse: 0.0053
Epoch 63/100
300/300 - 0s - 845us/step - loss: 0.0031 - mse: 0.0031
Epoch 64/100
300/300 - 0s - 990us/step - loss: 0.0048 - mse: 0.0048
Epoch 65/100
300/300 - 0s - 778us/step - loss: 0.0068 - mse: 0.0068
Epoch 66/100
300/300 - 0s - 833us/step - loss: 0.0021 - mse: 0.0021
Epoch 67/100
300/300 - 0s - 912us/step - loss: 0.0045 - mse: 0.0045
Epoch 68/100
300/300 - 0s - 974us/step - loss: 0.0049 - mse: 0.0049
Epoch 69/100
300/300 - 0s - 900us/step - loss: 0.0059 - mse: 0.0059
Epoch 70/100
300/300 - 0s - 893us/step - loss: 0.0056 - mse: 0.0056
Epoch 71/100
300/300 - 0s - 944us/step - loss: 0.0032 - mse: 0.0032
Epoch 72/100
300/300 - 0s - 816us/step - loss: 0.0038 - mse: 0.0038
Epoch 73/100
300/300 - 0s - 905us/step - loss: 0.0027 - mse: 0.0027
Epoch 74/100
300/300 - 0s - 735us/step - loss: 0.0045 - mse: 0.0045
Epoch 75/100
300/300 - 0s - 935us/step - loss: 0.0060 - mse: 0.0060
Epoch 76/100
300/300 - 0s - 826us/step - loss: 0.0058 - mse: 0.0058
Epoch 77/100
300/300 - 0s - 781us/step - loss: 0.0027 - mse: 0.0027
Epoch 78/100
300/300 - 0s - 828us/step - loss: 0.0023 - mse: 0.0023
Epoch 79/100
300/300 - 0s - 849us/step - loss: 0.0038 - mse: 0.0038
Epoch 80/100
300/300 - 0s - 862us/step - loss: 0.0034 - mse: 0.0034
Epoch 81/100
300/300 - 0s - 951us/step - loss: 0.0014 - mse: 0.0014
Epoch 82/100
300/300 - 0s - 1ms/step - loss: 0.0036 - mse: 0.0036
Epoch 83/100
300/300 - 0s - 776us/step - loss: 0.0044 - mse: 0.0044
Epoch 84/100
300/300 - 0s - 784us/step - loss: 0.0052 - mse: 0.0052
Epoch 85/100
300/300 - 0s - 683us/step - loss: 0.0065 - mse: 0.0065
Epoch 86/100
300/300 - 0s - 929us/step - loss: 0.0022 - mse: 0.0022
Epoch 87/100
300/300 - 0s - 1ms/step - loss: 0.0029 - mse: 0.0029
Epoch 88/100
300/300 - 0s - 974us/step - loss: 0.0056 - mse: 0.0056
Epoch 89/100
300/300 - 0s - 733us/step - loss: 0.0021 - mse: 0.0021
Epoch 90/100
300/300 - 0s - 891us/step - loss: 8.8256e-04 - mse: 8.8256e-04
```

```
Epoch 91/100
300/300 - 0s - 813us/step - loss: 0.0029 - mse: 0.0029
Epoch 92/100
300/300 - 0s - 792us/step - loss: 0.0026 - mse: 0.0026
Epoch 93/100
300/300 - 0s - 843us/step - loss: 0.0016 - mse: 0.0016
Epoch 94/100
300/300 - 0s - 827us/step - loss: 0.0017 - mse: 0.0017
Epoch 95/100
300/300 - 0s - 777us/step - loss: 0.0016 - mse: 0.0016
Epoch 96/100
300/300 - 0s - 718us/step - loss: 0.0033 - mse: 0.0033
Epoch 97/100
300/300 - 0s - 728us/step - loss: 9.5142e-04 - mse: 9.5142e-04
Epoch 98/100
300/300 - 0s - 837us/step - loss: 0.0022 - mse: 0.0022
Epoch 99/100
300/300 - 0s - 804us/step - loss: 0.0016 - mse: 0.0016
Epoch 100/100
300/300 - 0s - 976us/step - loss: 0.0025 - mse: 0.0025
학습 시간 : 26.097890615463257
```

In [149…

```python
rst = model3.predict(test_x)

plt.scatter(train_x, train_y)
plt.scatter(test_x, rst, color = 'r')
plt.show()
```

```
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```



OverFitting, DropOut, BatchNormalization, Regularization 처리 방법

In [153…

```
from sklearn.datasets import make_moons
```

In [163…

```
x, y = make_moons(n_samples=200, noise=0.2, random_state=64)
```

In [166…

```
x
```

```
Out[166…  array([[-4.33284954e-01,  8.88269469e-01],
                 [ 1.53188463e+00, -8.60258594e-01],
                 [ 7.68390388e-02,  9.43501750e-01],
                 [ 1.26055482e+00, -2.83812966e-01],
                 [ 9.44396636e-01,  3.05758496e-01],
                 [-1.07959399e+00,  6.80776115e-01],
                 [-1.55950332e-01,  1.66758238e-01],
                 [ 2.25677990e+00,  1.42337091e-01],
                 [ 4.79020666e-01,  4.87739166e-01],
                 [ 2.57481915e-02, -2.87038322e-01],
                 [ 6.53979329e-02,  5.05101831e-03],
                 [ 1.76540212e-01,  2.32520097e-01],
                 [ 5.96452590e-01,  4.85328698e-01],
                 [-6.61744846e-01,  8.72541996e-01],
                 [ 9.78353954e-01, -3.45460620e-01],
                 [-1.49240912e-01,  3.95425244e-01],
                 [-9.11042906e-01,  3.68264620e-01],
                 [ 2.08749991e-01,  7.65399236e-01],
                 [-1.39403831e-01, -9.10057366e-02],
                 [ 1.73188313e+00, -1.79502035e-02],
                 [ 1.32259213e+00, -6.74350818e-01],
                 [ 3.66456957e-01,  2.38721131e-01],
                 [-1.98612888e-01,  1.37121507e+00],
                 [ 2.68957817e-01, -1.26943573e-01],
                 [-7.22386050e-01,  5.06072715e-01],
                 [ 2.31962658e+00, -1.38028163e-02],
                 [-7.64398689e-01,  6.19138207e-01],
                 [ 1.11096327e+00, -5.95667383e-01],
                 [ 1.52221460e+00,  9.87119938e-02],
                 [ 9.69672539e-01, -7.57809374e-01],
                 [-9.00103059e-01,  3.81588555e-01],
                 [-6.47423772e-01,  4.42581064e-01],
                 [ 2.11390562e+00,  5.87579840e-01],
                 [-1.39590555e-01,  8.21202741e-01],
                 [ 1.59705472e+00, -1.06712221e-01],
                 [ 2.19613753e+00,  2.76922618e-02],
                 [ 1.98965627e+00,  2.12061671e-01],
                 [ 6.09612208e-01, -9.03809337e-01],
                 [ 8.17913685e-01,  6.45094100e-02],
                 [ 2.14106618e-01,  2.49679495e-02],
                 [ 4.04404054e-01, -1.52146205e-01],
                 [ 4.16276196e-01, -3.01986349e-01],
                 [ 1.33610331e+00,  3.26360657e-01],
                 [ 4.35454421e-01,  9.57419159e-01],
                 [ 2.03652271e+00,  8.14401060e-01],
                 [ 1.00883260e+00,  7.00822966e-01],
                 [ 5.86780222e-01, -5.28178362e-01],
                 [ 7.70183189e-01, -3.16487241e-01],
                 [ 4.59666334e-02, -1.36123731e-01],
                 [-8.38995184e-01, -1.36846391e-01],
                 [ 7.30528628e-01,  9.26976809e-01],
                 [ 3.53199121e-01, -3.03167649e-01],
                 [ 9.87029280e-01, -3.89893709e-01],
                 [ 7.30707581e-01,  6.05374122e-01],
                 [ 1.90047340e+00,  2.95716867e-01],
                 [-6.89357728e-01,  9.61933390e-01],
                 [ 1.61188236e+00, -2.51148653e-01],
                 [ 4.23094632e-01,  1.30222517e+00],
                 [-2.87662999e-01,  1.26952002e+00],
                 [-9.72363446e-01,  6.78533999e-01],
                 [ 4.43915596e-01,  6.39806875e-02],
                 [ 9.98396652e-01,  3.67731061e-01],
                 [-4.23360762e-01,  9.66326203e-01],
                 [ 1.63061546e+00,  4.48337153e-01],
                 [ 1.81789247e+00,  3.22973053e-02],
                 [ 3.74161042e-01,  7.46182158e-02],
                 [ 1.48027234e+00, -4.39588721e-01],
                 [ 6.52167066e-01,  4.86129994e-02],
                 [ 2.18473594e-02,  4.55345743e-01],
                 [ 2.06560139e+00,  4.90355427e-01],
                 [ 2.16487029e+00,  1.13408405e-01],
                 [ 1.03182638e+00, -5.54627275e-01],
                 [-9.07049467e-01,  1.03836302e-01],
                 [ 1.81187704e-02,  1.07435991e+00],
                 [ 9.63806950e-01, -2.43544920e-01],
                 [-8.26438896e-01,  3.89139449e-01],
                 [-5.99593602e-01,  7.55290542e-01],
                 [ 3.11294505e-02,  1.04320768e+00],
                 [ 2.85972662e-01, -2.08408320e-01],
                 [ 7.57956705e-01, -6.86417908e-01],
                 [-1.32898807e-01,  8.25573251e-01],
                 [ 7.38056351e-01,  3.89823716e-01],
                 [ 1.02530325e+00, -2.20282232e-01],
                 [ 1.64861892e+00,  2.86685849e-01],
                 [ 3.11412600e-01, -2.72106866e-01],
                 [ 8.39807387e-01,  3.33248345e-01],
                 [ 1.94060057e-01, -3.53391262e-01],
                 [ 8.88856975e-01, -3.08840167e-01],
                 [ 2.68878823e-02,  1.08875944e+00],
                 [ 1.05661960e+00,  1.08647690e-01],
                 [-2.08265183e-01,  9.52369441e-01],
                 [ 1.42263799e+00, -6.94002418e-01],
                 [-4.97752107e-01, -5.57613464e-02],
                 [-4.95935989e-01,  3.06377119e-01],
                 [ 1.92743081e+00, -4.07629823e-01],
                 [-8.54349451e-01,  7.06116514e-01],
                 [-7.21370941e-01,  7.93453525e-01],
                 [-2.84408368e-01,  7.62668678e-01],
                 [-2.30837183e-01,  9.42361864e-01],
                 [ 8.04985165e-01, -8.08551565e-01],
                 [ 3.93340222e-01,  1.14481643e+00],
                 [ 7.78048154e-01, -1.40476882e-01],
                 [ 9.71001913e-01,  6.98590711e-02],
                 [ 1.43477168e-01, -4.93625253e-01],
                 [-7.34271959e-01,  5.19449114e-01],
                 [-8.75422251e-01,  7.34859742e-01],
                 [ 8.07598796e-02,  1.14320850e+00],
                 [ 1.20994860e+00, -6.24453261e-01],
                 [ 9.43574946e-04, -1.46541854e-02],
                 [-4.53583895e-02,  4.60308725e-01],
                 [ 1.88872605e+00, -3.94287836e-01],
                 [-1.02697233e+00, -1.26353524e-01],
                 [ 1.09018867e+00, -5.23729791e-01],
                 [-7.50397319e-02,  1.17448456e+00],
                 [-3.43730526e-01,  1.22501659e+00],
                 [ 2.66884066e-02, -8.59109405e-03],
                 [ 1.05088908e+00, -3.77953497e-01],
                 [ 1.04116690e+00, -4.91806372e-01],
                 [ 1.22940000e+00,  2.34391969e-01],
                 [ 1.91113444e+00,  1.16484238e-01],
                 [-6.57689971e-01,  5.96969077e-01],
                 [-6.19495064e-01,  5.45347386e-01],
                 [ 1.45870852e-01,  4.43844161e-01],
                 [ 1.92492372e+00, -3.07116443e-02],
                 [ 7.51428404e-01,  1.05199706e-01],
                 [ 7.12154418e-01, -6.18269234e-01],
                 [ 1.66279230e+00, -1.73727506e-01],
                 [-1.31059990e+00,  2.44763381e-01],
                 [ 1.21810546e+00,  4.81698095e-01],
                 [ 1.07935388e+00, -6.32503730e-01],
                 [ 8.03926133e-02,  6.64580752e-01],
                 [ 1.75612551e+00, -1.53300980e-01],
                 [-1.07366635e+00,  5.27551951e-01],
                 [ 1.30530066e+00,  2.37174211e-01],
                 [ 8.16579273e-01, -3.85833395e-01],
                 [ 1.87961409e+00, -2.29708743e-01],
                 [ 3.28152451e-01,  5.43189292e-01],
                 [ 2.04516315e+00,  1.72994567e-01],
                 [ 1.30992188e+00, -4.43528348e-01],
                 [-9.08647132e-02,  4.17332715e-01],
                 [ 1.13296993e+00, -4.17948711e-01],
                 [ 1.49537268e+00, -7.84104903e-01],
                 [ 1.64407755e+00, -3.38747327e-01],
                 [-1.31881642e+00,  3.64818542e-01],
                 [ 1.43460743e+00, -2.38977141e-01],
                 [ 1.67934795e+00, -2.28353288e-01],
                 [ 2.23207541e+00,  3.91650305e-02],
                 [-1.06933154e+00,  2.77618188e-01],
                 [-7.41592979e-01,  5.82221999e-01],
                 [ 8.21238430e-02,  1.08471896e+00],
                 [-4.11687623e-01,  5.56207840e-01],
                 [ 1.79889161e-01,  2.69674614e-02],
                 [ 8.84409380e-01,  4.99976085e-01],
                 [ 1.73337447e+00,  6.17018551e-01],
                 [ 2.12456024e+00, -1.30226213e-01],
                 [-5.06498035e-01,  4.72815325e-01],
                 [ 7.23226290e-01,  8.07326131e-01],
                 [-2.17365568e-01,  1.04214031e+00],
                 [ 8.66961820e-01, -1.69499918e-02],
                 [ 6.29465815e-01,  2.56446573e-01],
                 [ 1.70785360e-01,  8.75557740e-01],
                 [ 7.43975777e-01,  6.87940470e-01],
                 [-9.35239113e-01,  1.15693184e-01],
                 [ 5.70625850e-01,  7.72164792e-01],
                 [-3.69758037e-01,  6.74632717e-01],
                 [ 9.23880486e-01,  6.05204421e-01],
                 [ 1.63705057e+00, -4.20325406e-01],
                 [ 1.81197803e+00,  6.90139166e-01],
                 [ 1.02193865e+00, -4.63475103e-01],
                 [-6.38516053e-01,  5.03702612e-01],
                 [-4.51596362e-01,  6.87078444e-01],
                 [ 1.53109026e-01,  1.10134564e-01],
                 [ 1.07846581e+00,  4.85887846e-02],
                 [ 8.82412220e-01, -5.57996381e-01],
                 [ 8.55747270e-01,  8.52722620e-01],
                 [ 4.93376235e-01,  1.04904414e+00],
                 [-5.60318741e-01,  1.00885645e+00],
                 [ 2.57338609e-01,  5.93609004e-01],
                 [-3.74768923e-02, -7.34758502e-02],
                 [-1.10757409e+00, -4.24261210e-02],
                 [ 7.35609751e-01, -6.10107693e-01],
                 [ 5.13808802e-01, -2.13313280e-01],
                 [ 9.75584531e-01,  5.54569878e-01],
                 [-1.74654125e-01,  7.11722134e-01],
                 [ 1.07651384e-01,  9.47924001e-01],
                 [-2.31256074e-01,  2.08529077e-01],
                 [ 7.27937093e-01,  5.00378785e-01],
                 [ 4.02904360e-01,  9.61232384e-01],
                 [ 5.03581919e-01, -2.47993302e-01],
                 [-4.51224968e-02, -2.35163206e-01],
                 [ 7.70492052e-01, -4.25284804e-01],
                 [ 3.46684046e-01, -2.54299204e-01],
                 [ 7.12924331e-01,  6.12666262e-01],
                 [ 1.18329757e-01,  1.34124338e+00],
                 [-7.96815281e-01,  4.78488058e-01],
                 [ 1.44640809e-01,  9.74635732e-01],
                 [-7.64637107e-02,  8.47381039e-01],
                 [ 1.38953588e+00, -2.83712673e-01],
                 [ 4.65793072e-01,  8.33144066e-03],
                 [ 1.62361566e+00,  3.23457766e-01]])
```

In [182…

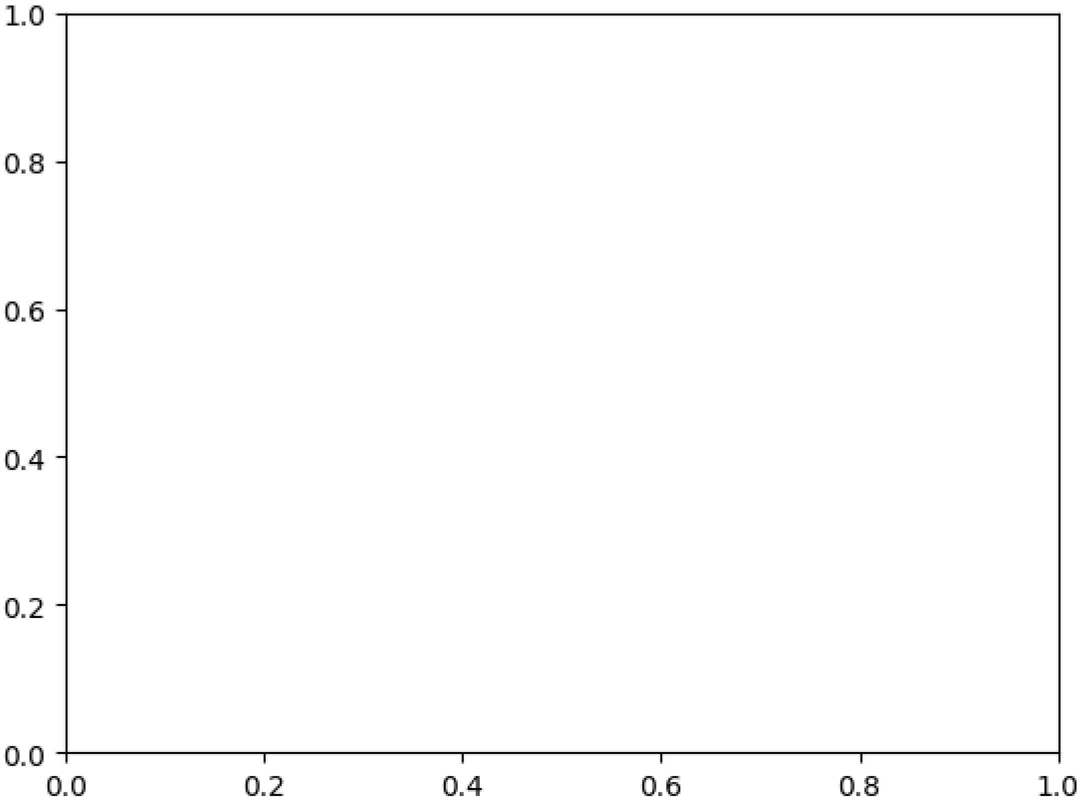
```
df = pd.DataFrame(dict(x = x[:,0], y = x[:, 1], label = y))
```

데이터를 이미지로 표시하는 시각화

In [195…

```
fig, ax = plt.subplots()
group = df.groupby('label')
colors = ['red', 'green']

for idx, group in groups:
    group.plot(ax = ax, kind = 'scatter', y = 'y', label = index, color = colors

plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[195], line 6
      3 colors = ['red', 'green']
      5 for idx, group in groups:
----> 6     group.plot(ax = ax, kind = 'scatter', y = 'y', label = index, color = 
colors[idx])
      8 plt.show()

NameError: name 'index' is not defined
```



데이터 Train, test 분리

In [171…

```
split_index = int(len(x)*0.7)
train_x, test_x = x[:split_index], x[split_index:]
train_y, test_y = y[:split_index], y[split_index:]
```

In [172…

```
(train_x.shape, train_y.shape), (test_x.shape, test_y.shape)
```

Out[172…

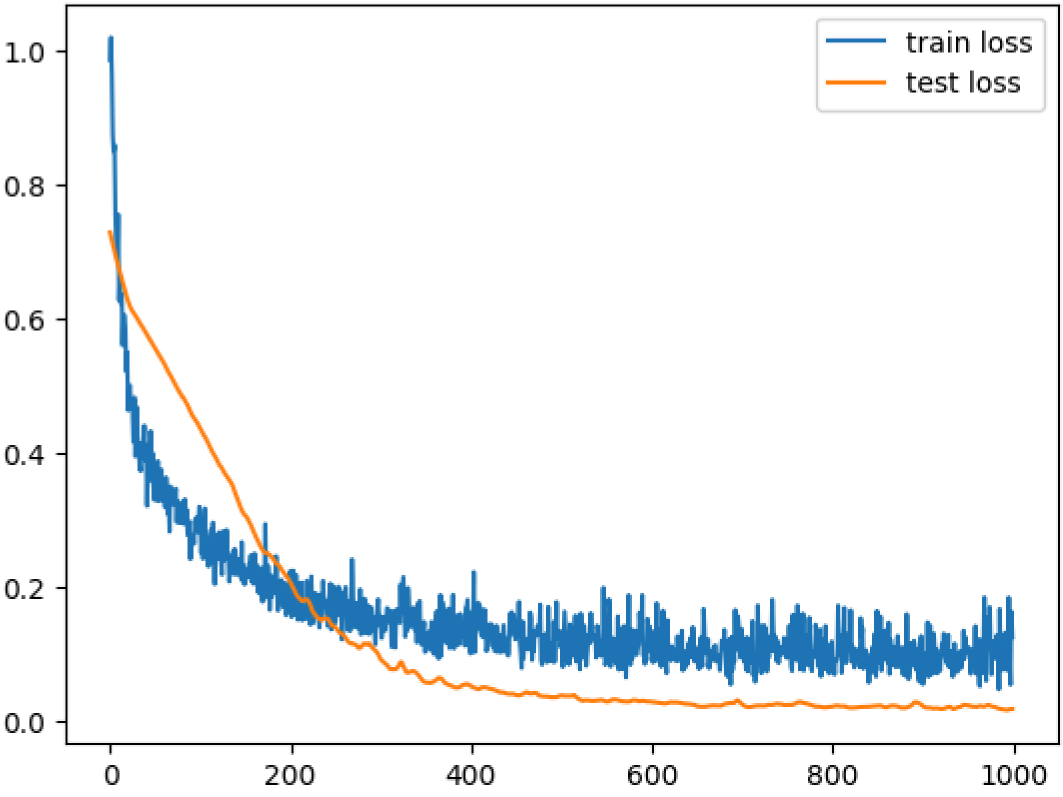
```
(((140, 2), (140,)), ((60, 2), (60,)))
```

In [190…

```
# Dropout
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.regularizers import l1, l2

model5 = keras.Sequential()
model5.add(Dense(20, input_shape=(2,), activation='relu', kernel_regularizer=l2(
model5.add(Dropout(0.3)) # 6을 넘어가면 안됨
model5.add(BatchNormalization())
model5.add(Dense(20, activation='relu', kernel_regularizer=l2(0.001)))
model5.add(Dropout(0.3)) # 시그모이드 하기 전가지 히든 레이어에서 사용
model5.add(BatchNormalization())
model5.add(Dense(1, activation='sigmoid'))

model5.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'

history = model5.fit(train_x, train_y, epochs = 1000, verbose = 0, batch_size =
```

In [191…

```
plt.plot(history.history['loss'], label = 'train loss')
plt.plot(history.history['val_loss'], label = 'test loss')
# plt.savefig('data_tf/Dropout.png')
# plt.savefig('data_tf/BatchNo.png')
plt.savefig('data_tf/Regularization.png')

plt.legend()
plt.show()
```



콜백 함수

In [218…

```
# 학습중 다양한 명령을 수행하고 싶을 때 사용하는 함수
from tensorflow.keras.callbacks import Callback
```

In [219…

```
x = np.arange(-1, 1, 0.01)
np.random.shuffle(x)
y = x ** 2
```

In [220…

```
split_index = int(x.shape[0]*0.6)
train_x, test_x = x[:split_index], x[split_index:]
train_y, test_y = y[:split_index], y[split_index:]
```

In [221…

```
(train_x.shape, test_x.shape), (train_y.shape, test_y.shape)
```

Out[221…

```
(((120,), (80,)), ((120,), (80,)))
```

In [222…

```
# 콜백 함수 생성
def train_callbacks(callbacks):
    model = keras.Sequential()
    model.add(Dense(10, activation='tanh', input_shape=(1,)))
    model.add(Dense(10, activation='tanh'))
    model.add(Dense(1))
    model.compile(optimizer='SGD', loss='mse', metrics=['mse'])

    start_time = time.time()
    model.fit(train_x, train_y, epochs=1000, verbose=0, batch_size=20, validatic
    print("학습 시간 : {}".format(time.time() - start_time))
```

In [223…

```
from tensorflow.keras.callbacks import ModelCheckpoint
```

In [224…

```
check_point = ModelCheckpoint("./best_model_h5.keras", monitor="val_loss", mode
```

In [225…

```
train_callbacks([check_point])
```

```
학습 시간 : 56.1355504989624
```

디스플레이 처리

In [226…

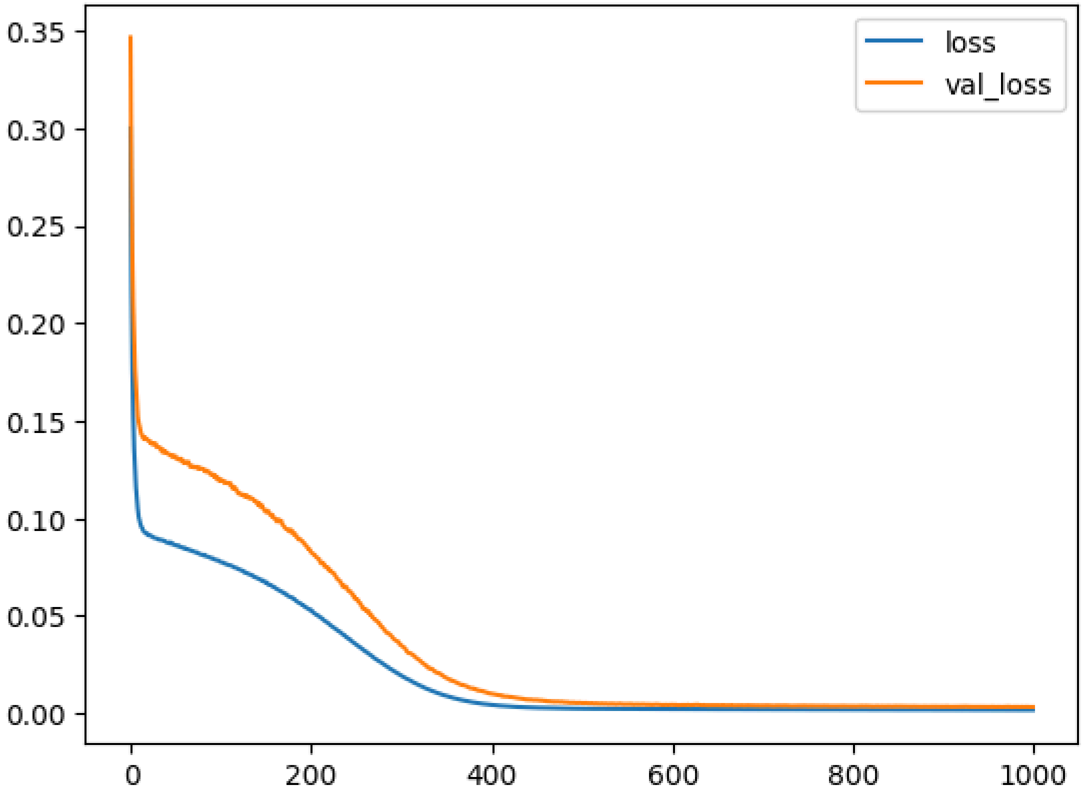
```
from IPython.display import clear_output
```

In [228…

```
class PlotLosses(Callback):
  def on_train_begin(self, logs={}):

      self.i = 0
      self.x = []
      self.losses = []
      self.val_losses = []
      self.fig = plt.figure()
      self.logs = []

  def on_epoch_end(self, epoch, logs={}):
      self.logs.append(logs)
      self.x.append(self.i)
      self.losses.append(logs.get('loss'))
      self.val_losses.append(logs.get('val_loss'))
      self.i += 1
      clear_output(wait=True)
      plt.plot(self.x, self.losses, label="loss")
      plt.plot(self.x, self.val_losses, label="val_loss")
      plt.legend()
      plt.show();
      print("loss = ", self.losses[-1], ", val_loss = ", self.val_losses[-1])
```

In [229…

```
plt_loss = PlotLosses()
train_callbacks([plt_loss])
```



```
loss =  0.0016853249398991466 , val_loss =  0.0032138151582330465
학습 시간 : 204.19889426231384
```