14장. 베스트 모델 만들기

(데이터 입력)

```
[ ]  1 from keras.models import Sequential
     2 from keras.layers import Dense
     3 from keras.callbacks import ModelCheckpoint, EarlyStopping
     4
     5 import pandas as pd
     6 import numpy
     7 import tensorflow as tf
     8 import matplotlib.pyplot as plt
```

```
[ ]  1 # seed 값 설정
     2 seed = 0
     3 numpy.random.seed(seed)
     4 tf.random.set_seed(3)
     5
     6 # 데이터 입력
     7 df_pre = pd.read_csv('wine.csv', header=None)
     8 df = df_pre.sample(frac=1)
```

(데이터 확인)

```
1 print(df.head(5)) #처음 다섯줄 출력
2 print(df.info())
```

```
      0     1     2     3      4     5   ...       7     8     9     10  11  12
5316  6.3  0.18  0.24   3.4  0.053  20.0  ...  0.99373  3.11  0.52   9.2   6   0
5210  6.8  0.14  0.18   1.4  0.047  30.0  ...  0.99164  3.27  0.54  11.2   6   0
3518  7.3  0.22  0.50  13.7  0.049  56.0  ...  0.99940  3.24  0.66   9.0   6   0
1622  7.6  0.67  0.14   1.5  0.074  25.0  ...  0.99370  3.05  0.51   9.3   5   0
2443  7.3  0.21  0.29   1.6  0.034  29.0  ...  0.99170  3.30  0.50  11.0   8   0

[5 rows x 13 columns]
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6497 entries, 5316 to 2732
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       6497 non-null   float64
 1   1       6497 non-null   float64
 2   2       6497 non-null   float64
 3   3       6497 non-null   float64
 4   4       6497 non-null   float64
 5   5       6497 non-null   float64
 6   6       6497 non-null   float64
 7   7       6497 non-null   float64
 8   8       6497 non-null   float64
 9   9       6497 non-null   float64
 10  10      6497 non-null   float64
 11  11      6497 non-null   int64
```

(실행)

```
dataset = df.values
X = dataset[:,0:12]
Y = dataset[:,12]

# 모델 설정
model = Sequential()
model.add(Dense(30,  input_dim=12, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

#모델 컴파일
model.compile(loss='binary_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])

# 모델 실행
model.fit(X, Y, epochs=200, batch_size=200)

# 결과 출력
print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

(결과)

```
Epoch 191/200
33/33 [==============================] - 0s 2ms/step - loss: 0.0484 - accuracy: 0.9
Epoch 192/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0449 - accuracy: 0.9
Epoch 193/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0560 - accuracy: 0.9
Epoch 194/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0410 - accuracy: 0.9
Epoch 195/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0501 - accuracy: 0.9
Epoch 196/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0459 - accuracy: 0.9
Epoch 197/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0545 - accuracy: 0.9
Epoch 198/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0480 - accuracy: 0.9
Epoch 199/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0491 - accuracy: 0.9
Epoch 200/200
33/33 [==============================] - 0s 1ms/step - loss: 0.0573 - accuracy: 0.9
204/204 [==============================] - 0s 950us/step - loss: 0.0470 - accuracy:

 Accuracy: 0.9861
```

프레임워크 완성

• 실습 : 모델의 정확도를 기록하며 저장

```
import os
```

```
# 모델 저장 폴더 설정
MODEL_DIR = './model/'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)

# 모델 저장 조건 설정
modelpath="./model/{epoch:02d}-{val_loss:.4f}.hdf5"
checkpointer = ModelCheckpoint(filepath=modelpath,
                               monitor='val_loss', verbose=1,
                               save_best_only=True)

# 모델 실행 및 저장
model.fit(X, Y, validation_split=0.2, epochs=200,
          batch_size=200, verbose=0, callbacks=[checkpointer])
```

↳ 앞서 저장한 모델보다 나아졌을 때만 저장

(결과)

```
Epoch 00192: val_loss did not improve from 0.03946

Epoch 00193: val_loss did not improve from 0.03946

Epoch 00194: val_loss did not improve from 0.03946

Epoch 00195: val_loss did not improve from 0.03946

Epoch 00196: val_loss did not improve from 0.03946

Epoch 00197: val_loss did not improve from 0.03946

Epoch 00198: val_loss did not improve from 0.03946

Epoch 00199: val_loss did not improve from 0.03946

Epoch 00200: val_loss did not improve from 0.03946
<tensorflow.python.keras.callbacks.History at 0x7fdf72c95f90>
```

마지막200!

• 실습 : 그래프 표현

```
import matplotlib.pyplot as plt
```

```
# 모델 실행 및 저장
history = model.fit(X, Y, validation_split=0.33, epochs=3500, batch_size=500)

# y_vloss에 테스트셋으로 실험 결과의 오차 값을 저장
y_vloss=history.history['val_loss']

# y_acc 에 학습 셋으로 측정한 정확도의 값을 저장
y_acc=history.history['accuracy']

# x값을 지정하고 정확도를 파란색으로, 오차를 빨간색으로 표시
x_len = numpy.arange(len(y_acc))
plt.plot(x_len, y_vloss, "o", c="red", markersize=3)
plt.plot(x_len, y_acc, "o", c="blue", markersize=3)

plt.show()
```

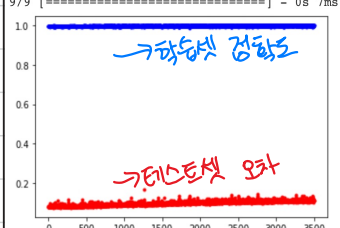
→ 'acc'는 버전 충돌 때문에 안됨

→ 오차

→ 정확도

(결과1) → fmc=1 일때

```
Epoch 3497/3500
9/9 [==============================] - 0s 8ms/step - loss: 0.0032 - accuracy: 0.998
Epoch 3498/3500
9/9 [==============================] - 0s 7ms/step - loss: 0.0043 - accuracy: 0.998
Epoch 3499/3500
9/9 [==============================] - 0s 8ms/step - loss: 0.0034 - accuracy: 0.998
Epoch 3500/3500
9/9 [==============================] - 0s 7ms/step - loss: 0.0031 - accuracy: 0.999
```



(학습의 자동 중단) → 학습이 진행되어도 테스트셋 오차가 줄지 않으면 학습을 멈추게 함.

```
from keras.callbacks import EarlyStopping
```

```
# 자동 중단 설정
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=100)

# 모델 실행
model.fit(X, Y, validation_split=0.2, epochs=2000,
          batch_size=500, callbacks=[early_stopping_callback])

# 결과 출력
print("\n Accuracy: %.4f" % (model.evaluate(X, Y)[1]))
```

(결과)

```
Epoch 240/2000
2/2 [==============================] - 0s 41ms/step - loss: 0.0043 - accuracy: 0.99
Epoch 241/2000
2/2 [==============================] - 0s 44ms/step - loss: 0.0046 - accuracy: 0.99
Epoch 242/2000
2/2 [==============================] - 0s 42ms/step - loss: 0.0053 - accuracy: 0.99
Epoch 243/2000
2/2 [==============================] - 0s 42ms/step - loss: 0.0042 - accuracy: 0.99
Epoch 244/2000
2/2 [==============================] - 0s 48ms/step - loss: 0.0041 - accuracy: 0.99
Epoch 245/2000
2/2 [==============================] - 0s 44ms/step - loss: 0.0043 - accuracy: 0.99
31/31 [==============================] - 0s 874us/step - loss: 0.0187 - accuracy: 0

 Accuracy: 0.9969
```