다중 분류 문제 해결하기

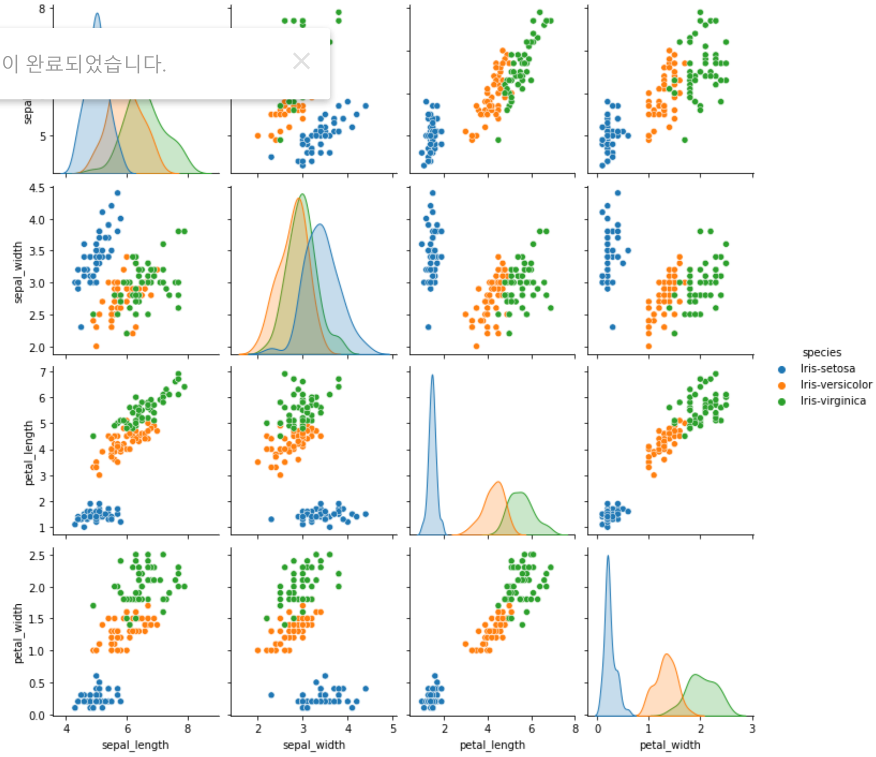
2. 상관도 그래프

```
import pandas as pd

# 아이리스 데이터를 불러옵니다.
df = pd.read_csv('./data/iris3.csv')

# 첫 5줄을 봅니다.
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
import seaborn as sns
import matplotlib.pyplot as plt

# 그래프로 확인해 봅시다.
sns.pairplot(df, hue='species');
plt.show()
```

저장이 완료되었습니다.



3. 원-핫 인코딩

```
# 속성을 X, 클래스를 y로 저장합니다.
X = df.iloc[:,0:4]
y = df.iloc[:,4]

# X와 y의 첫 5줄을 출력해 보겠습니다.
print(X[0:5])
print(y[0:5])
```

```
   sepal_length  sepal_width  petal_length  petal_width
0           5.1          3.5           1.4          0.2
1           4.9          3.0           1.4          0.2
2           4.7          3.2           1.3          0.2
3           4.6          3.1           1.5          0.2
4           5.0          3.6           1.4          0.2
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: species, dtype: object
```

```
# 원-핫 인코딩 처리를 합니다.
y = pd.get_dummies(y)

# 원-핫 인코딩 결과를 확인합니다.
print(y[0:5])
```

```
   Iris-setosa  Iris-versicolor  Iris-virginica
0            1                0               0
1            1                0               0
2            1                0               0
3            1                0               0
4            1                0               0
```

저장이 완료되었습니다. ×

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 모델 설정
model = Sequential()
model.add(Dense(12,  input_dim=4, activation='relu'))
model.add(Dense(8,  activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

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

# 모델 실행
history=model.fit(X, y, epochs=30, batch_size=5)
```

```
30/30 [==============================] - 0s 3ms/step - loss: 0.6911 - accuracy: 0.6667
Epoch 9/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6716 - accuracy: 0.6667
Epoch 10/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6547 - accuracy: 0.6667
Epoch 11/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6393 - accuracy: 0.6667
Epoch 12/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6253 - accuracy: 0.6667
Epoch 13/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6127 - accuracy: 0.6667
Epoch 14/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6005 - accuracy: 0.6667
Epoch 15/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5898 - accuracy: 0.6667
Epoch 16/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5801 - accuracy: 0.6667
Epoch 17/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5714 - accuracy: 0.6667
Epoch 18/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5633 - accuracy: 0.6667
Epoch 19/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5556 - accuracy: 0.6667
Epoch 20/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5487 - accuracy: 0.6667
Epoch 21/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5424 - accuracy: 0.6667
Epoch 22/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5370 - accuracy: 0.6667
Epoch 23/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5315 - accuracy: 0.6667
Epoch 24/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5267 - accuracy: 0.6667
Epoch 25/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5221 - accuracy: 0.6667
Epoch 26/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5185 - accuracy: 0.6667
Epoch 27/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5145 - accuracy: 0.6667
Epoch 28/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5112 - accuracy: 0.6733
Epoch 29/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5082 - accuracy: 0.6800
Epoch 30/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5053 - accuracy: 0.6800
```

저장이 완료되었습니다. ×

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# 아이리스 데이터를 불러옵니다.
df = pd.read_csv('./data/iris3.csv')

# 속성을 X, 클래스를 y로 저장합니다.
X = df.iloc[:,0:4]
y = df.iloc[:,4]

# 원-핫 인코딩 처리를 합니다.
y = pd.get_dummies(y)

# 모델 설정
model = Sequential()
model.add(Dense(12,  input_dim=4, activation='relu'))
model.add(Dense(8,  activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()
```

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

# 모델 실행
history=model.fit(X, y, epochs=30, batch_size=5)
```

```
Epoch 2/30
30/30 [==============================] - 0s 1ms/step - loss: 0.9257 - accuracy: 0.3333
Epoch 3/30
30/30 [==============================] - 0s 1ms/step - loss: 0.8802 - accuracy: 0.3600
Epoch 4/30
30/30 [==============================] - 0s 2ms/step - loss: 0.8403 - accuracy: 0.3600
Epoch 5/30
30/30 [==============================] - 0s 1ms/step - loss: 0.8072 - accuracy: 0.6733
Epoch 6/30
30/30 [==============================] - 0s 1ms/step - loss: 0.7768 - accuracy: 0.7867
Epoch 7/30
30/30 [==============================] - 0s 2ms/step - loss: 0.7498 - accuracy: 0.7800
Epoch 8/30
30/30 [==============================] - 0s 1ms/step - loss: 0.7287 - accuracy: 0.7800
Epoch 9/30
30/30 [==============================] - 0s 2ms/step - loss: 0.7127 - accuracy: 0.8000
Epoch 10/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6970 - accuracy: 0.8400
Epoch 11/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6765 - accuracy: 0.8667
Epoch 12/30
30/30 [==============================] - 0s 1ms/step - loss: 0.6584 - accuracy: 0.9267
Epoch 13/30
30/30 [==============================] - 0s 1ms/step - loss: 0.6433 - accuracy: 0.9000
Epoch 14/30
30/30 [==============================] - 0s 1ms/step - loss: 0.6279 - accuracy: 0.9200
Epoch 15/30
30/30 [==============================] - 0s 1ms/step - loss: 0.6128 - accuracy: 0.9267
Epoch 16/30
30/30 [==============================] - 0s 2ms/step - loss: 0.6002 - accuracy: 0.9267
Epoch 17/30
30/30 [==============================] - 0s 2ms/step - loss: 0.5875 - accuracy: 0.9333
Epoch 18/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5777 - accuracy: 0.9467
Epoch 19/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5588 - accuracy: 0.9267
Epoch 20/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5469 - accuracy: 0.9333
Epoch 21/30
====] - 0s 1ms/step - loss: 0.5320 - accuracy: 0.9533
====] - 0s 1ms/step - loss: 0.5198 - accuracy: 0.9600
Epoch 23/30
30/30 [==============================] - 0s 1ms/step - loss: 0.5077 - accuracy: 0.9533
Epoch 24/30
30/30 [==============================] - 0s 1ms/step - loss: 0.4999 - accuracy: 0.9800
Epoch 25/30
30/30 [==============================] - 0s 1ms/step - loss: 0.4810 - accuracy: 0.9533
Epoch 26/30
30/30 [==============================] - 0s 2ms/step - loss: 0.4705 - accuracy: 0.9667
Epoch 27/30
30/30 [==============================] - 0s 2ms/step - loss: 0.4620 - accuracy: 0.9600
Epoch 28/30
30/30 [==============================] - 0s 2ms/step - loss: 0.4152 - accuracy: 0.9800
Epoch 29/30
30/30 [==============================] - 0s 2ms/step - loss: 0.3172 - accuracy: 0.9600
Epoch 30/30
30/30 [==============================] - 0s 1ms/step - loss: 0.2777 - accuracy: 0.9800
```

저장이 완료되었습니다.

```
score=model.evaluate(X, y)
print('Test accuracy:', score[1])
```

```
5/5 [==============================] - 0s 3ms/step - loss: 0.2627 - accuracy: 0.9600
Test accuracy: 0.9599999785423279
```

✓ 1초 오전 8:44에 완료됨 ● ×

저장이 완료되었습니다. ×