

학습 사고 : 파이썬과 딥러닝

## 12장 다중 분류 문제 해결하기

이 찬 우

# 학습 내용 다중 분류 문제 해결하기

---

- 1 | 다중 분류 문제
- 2 | 상관도 그래프
- 3 | 원-핫 인코딩
- 4 | 소프트맥스
- 5 | 아이리스 품종 예측 실행

# 다중 분류 문제 해결하기

- 실습 데이터 아이리스 품종 예측

✓ dataset/iris.csv

## 1 | 다중 분류 문제

- 아이리스는 꽃잎의 모양과 길이에 따라 여러 가지 품종으로 나뉨
- 사진을 보면 품종마다 비슷해 보이는데 과연 딥러닝을 사용하여 이들을 구별해 낼 수 있을까?



Iris-virginica



Iris-setosa



Iris-versicolor

그림 12-1 아이리스의 품종

# 1 | 다중 분류 문제

- 아이리스 품종 예측 데이터는 책과 함께 제공하는 예제 파일의 dataset 폴더에서 찾을 수 있음  
(dataset/iris.csv)

속성					클래스	
샘플	정보 1	정보 2	정보 3	정보 4	품종	
	1번째 아이리스	5.1	3.5	4.0	0.2	Iris-setosa
	2번째 아이리스	4.9	3.0	1.4	0.2	Iris-setosa
	3번째 아이리스	4.7	3.2	1.3	0.3	Iris-setosa
	...	...	...	...	...	...
	150번째 아이리스	5.9	3.0	5.1	1.8	Iris-virginica

표 12-1 아이리스 데이터의 샘플, 속성, 클래스 구분

## 1 | 다중 분류 문제

- 샘플 수: 150
- 속성 수: 4
  - 정보 1: 꽃받침 길이 (sepal length, 단위: cm)
  - 정보 2: 꽃받침 너비 (sepal width, 단위: cm)
  - 정보 3: 꽃잎 길이 (petal length, 단위: cm)
  - 정보 4: 꽃잎 너비 (petal width, 단위: cm)
- 클래스: Iris-setosa, Iris-versicolor, Iris-virginica

## 1 | 다중 분류 문제

- 속성을 보니 우리가 앞서 다루었던 것과 중요한 차이는 바로 클래스가 2개가 아니라 3개임
- 즉, 참(1)과 거짓(0)으로 해결하는 것이 아니라,  
여러 개 중에 어떤 것이 답인지를 예측하는 문제임
- 다중 분류 (multi classification) :  
여러 개의 답 중 하나를 고르는 분류 문제
- 다중 분류 문제는 둘 중에 하나를 고르는 이항 분류(binary classification)와는 접근 방식이 조금 다름

## 2 | 상관도 그래프

- 먼저 데이터의 일부를 불러와 내용을 보자

```
import pandas as pd
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",
"sepal_width", "petal_length", "petal_width", "species"])
print(df.head( ))
```



## 2 | 상관도 그래프

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3	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	3.6	1.4	0.2	Iris-setosa

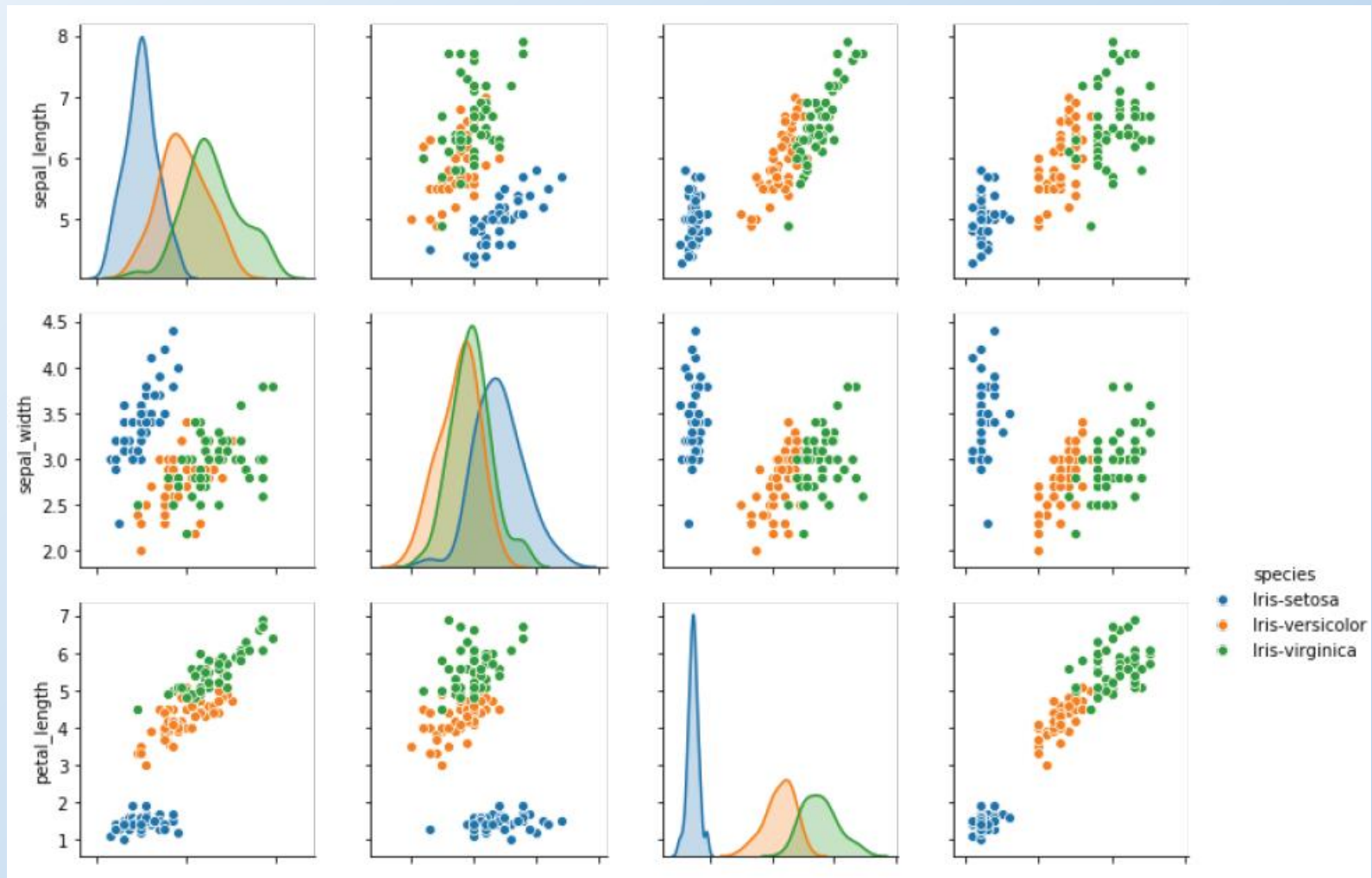
## 2 | 상관도 그래프

- 이번에는 pairplot() 함수를 써서 데이터 전체를 한번에 보는 그래프를 다음과 같이 출력

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

sns.pairplot(df, hue='species');
plt.show()
```

## 2 | 상관도 그래프



## 2 | 상관도 그래프

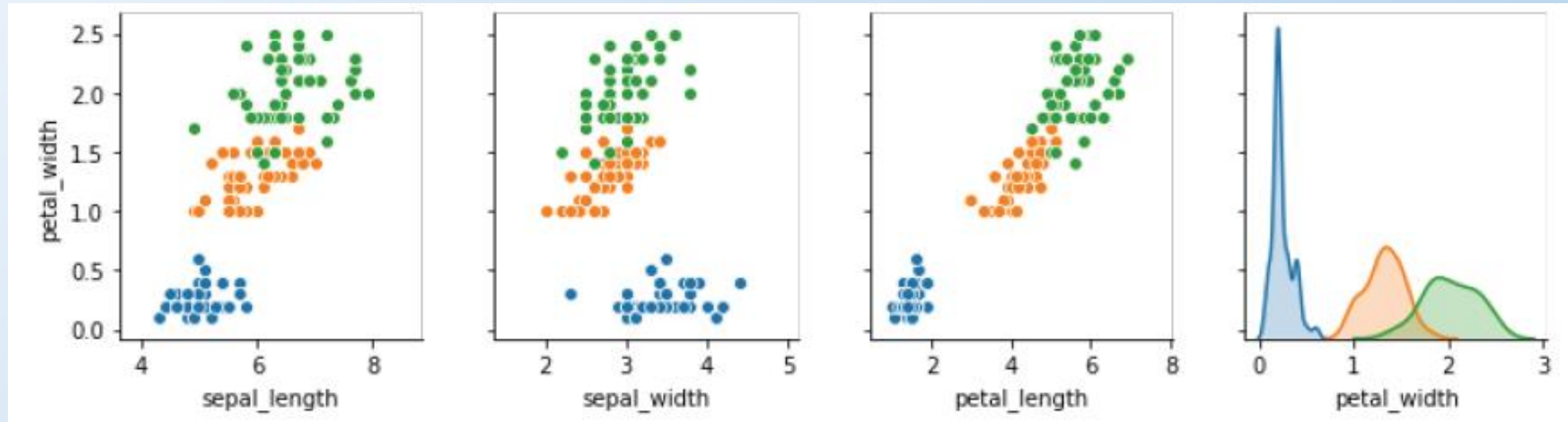


그림 12-2 pairplot 함수로 데이터 한번에 보기

## 2 | 상관도 그래프

- 그래프를 보니, 사진으로 볼 때는 비슷해 보이던  
꽃잎과 꽃받침의 크기와 너비가 품종 별로 차이가 있음을 알 수 있음
- 속성별로 어떤 연관이 있는지를 보여 주는 상관도 그래프를 통해  
프로젝트의 감을 잡고 프로그램 전략을 세울 수 있음

### 3 | 원-핫 인코딩

- 이제 케라스를 이용해 아이리스의 품종을 예측해 보자
- Iris-setosa, Iris-virginica 이터 안에 문자열이 포함되어 있음
- numpy보다는 pandas로 데이터를 불러와 X와 Y값을 구분하는 것이 좋음

```
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",  
"sepal_width", "petal_length", "petal_width", "species"])  
  
dataset = df.values  
X = dataset[:,0:4].astype(float)  
Y_obj = dataset[:,4]
```

## 참고: 원-핫 인코딩의 개념 I

- 원핫(One-Hot) 인코딩이라는 말처럼 이 기술은 데이터를 수많은 0과 한개의 1의 값으로 데이터를 구별하는 인코딩
- 예를 들어, 글자가 적힌 종이를 0부터 9까지의 숫자값으로 분류를 하는 작업을 한다면 숫자값 들은 다음과 같은 값들로 이루어질 수 있을 것이다. 출처: <https://needjarvis.tistory.com/565>

```
[1,0,0,0...0]
```

```
[0,1,0,0...0]
```

```
[0,0,1,0...0]
```

```
...
```

## 참고: 원-핫 인코딩의 개념 II

- 위와같이 0으로 이루어진 벡터에 단 한개의 1의 값으로 해당 데이터의 값을 구별하는 것이 원핫 인코딩
- 원핫 인코딩은 매우 기초적이면서도 아주 흔히 사용되는 기법으로 딥러닝 뿐만 아니라 데이터 마이닝, 자연어 처리 등 아직도 많은 분야에서 사용되고 있는 기법

출처 : <https://needjarvis.tistory.com/565>



### 3 | 원-핫 인코딩

- 또한, Y값이 이번에는 숫자가 아니라 문자열임
- 문자열을 숫자로 바꿔 주려면 클래스 이름을 숫자 형태로 바꿔 주어야 함
- 이를 가능하게 하는 함수가 sklearn 라이브러리의 LabelEncoder( ) 함수임

```
from sklearn.preprocessing import LabelEncoder  
  
e = LabelEncoder()  
e.fit(Y_obj)  
Y = e.transform(Y_obj)
```

### 3 | 원-핫 인코딩

- `array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'])`가 `array([1,2,3])`로 바뀜
- 활성화 함수를 적용하려면 Y 값이 숫자 0과 1로 이루어져 있어야 함
- 이 조건을 만족시키려면 `tf.keras.utils.categorical( )` 함수를 적용해야 함
- 이에 따라 Y 값의 형태는 다음과 같이 변형됨

```
from tensorflow.keras.utils import np_utils
```

```
Y_encoded = tf.keras.utils.to_categorical(Y)
```

### 3 | 원-핫 인코딩

- `array([1,2,3])`가 다시 `array([[1., 0., 0.], [0., 1., 0.],[0., 0., 1.]])`로 바뀜
- 원-핫 인코딩 (one-hot-encoding) :  
여러 개의 Y 값을 0과 1로만 이루어진 형태로 바꿔 주는 기법

## 4 | 소프트맥스

- 이제 모델을 만들어 보자

```
model = Sequential()  
model.add(Dense(16, input_dim=4, activation='relu'))  
model.add(Dense(3, activation='softmax'))
```

## 소프트맥스 ?

- 분류해야하는 정답지(클래스)의 총 개수를  $k$ 라고 할 때,  
     $k$ 차원의 벡터를 입력 받아 각 클래스에 대한 확률을 추정합니다
- $K$  차원의 벡터에서  $i$  번째 원소를  $z_i$ ,  $i$  번째 클래스가 정답일 확률을  $p_i$ 로 나타낸다고 하였을 때 소프트맥스 함수는  $p_i$  를 다음과 같이 정의

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \text{ for } i = 1, 2, \dots, k$$

위에서 풀어야하는 문제에 소프트맥스 함수를 차근차근 적용해봅시다. 위에서 풀어야하는 문제의 경우  $k=3$ 이므로 3차원 벡터  $z = [z_1 \ z_2 \ z_3]$ 의 입력을 받으면 소프트맥스 함수는 아래와 같은 출력을 리턴합니다.

$$\text{softmax}(z) = \left[ \frac{e^{z_1}}{\sum_{j=1}^3 e^{z_j}} \quad \frac{e^{z_2}}{\sum_{j=1}^3 e^{z_j}} \quad \frac{e^{z_3}}{\sum_{j=1}^3 e^{z_j}} \right] = [p_1, p_2, p_3] = \hat{y} = \text{예측값}$$

$p_1, p_2, p_3$  각각은 1번 클래스가 정답일 확률, 2번 클래스가 정답일 확률, 3번 클래스가 정답일 확률을 나타내며 각각 0과 1사이의 값으로 총 합은 1이 됩니다. 여기서 분류하고자하는 3개의 클래스는 virginica, setosa, versicolor이므로 이는 결국 주어진 입력이 virginica일 확률, setosa일 확률, versicolor일 확률을 나타내는 값을 의미합니다. 여기서는  $i$ 가 1일 때는 virginica일 확률을 나타내고, 2일 때는 setosa일 확률, 3일때는 versicolor일 확률이라고 지정하였다고 합시다. 이 지정 순서는 문제를 풀고자 하는 사람의 무작위 선택입니다. 이에따라 식을 문제에 맞게 다시 쓰면 아래와 같습니다.

$$\text{softmax}(z) = \left[ \frac{e^{z_1}}{\sum_{j=1}^3 e^{z_j}} \quad \frac{e^{z_2}}{\sum_{j=1}^3 e^{z_j}} \quad \frac{e^{z_3}}{\sum_{j=1}^3 e^{z_j}} \right] = [p_1, p_2, p_3] = [p_{\text{virginica}}, p_{\text{setosa}}, p_{\text{versicolor}}]$$

다소 복잡해보이지만 어려운 개념이 아닙니다. 분류하고자 하는 클래스가  $k$ 개일 때,  $k$ 차원의 벡터를 입력받아서 모든 벡터 원소의 값을 0과 1사이의 값으로 값을 변경하여 다시  $k$ 차원의 벡터를 리턴한다는 내용을 식으로 기재하였을 뿐입니다. 방금 배운 개념을 그림을 통해 다시 설명하면서 더 깊이 들어가보겠습니다.

## 4 | 소프트맥스

- 소프트맥스 :

그림 12-3에서와 같이 총합이 1인 형태로 바뀌서 계산해 주는 함수

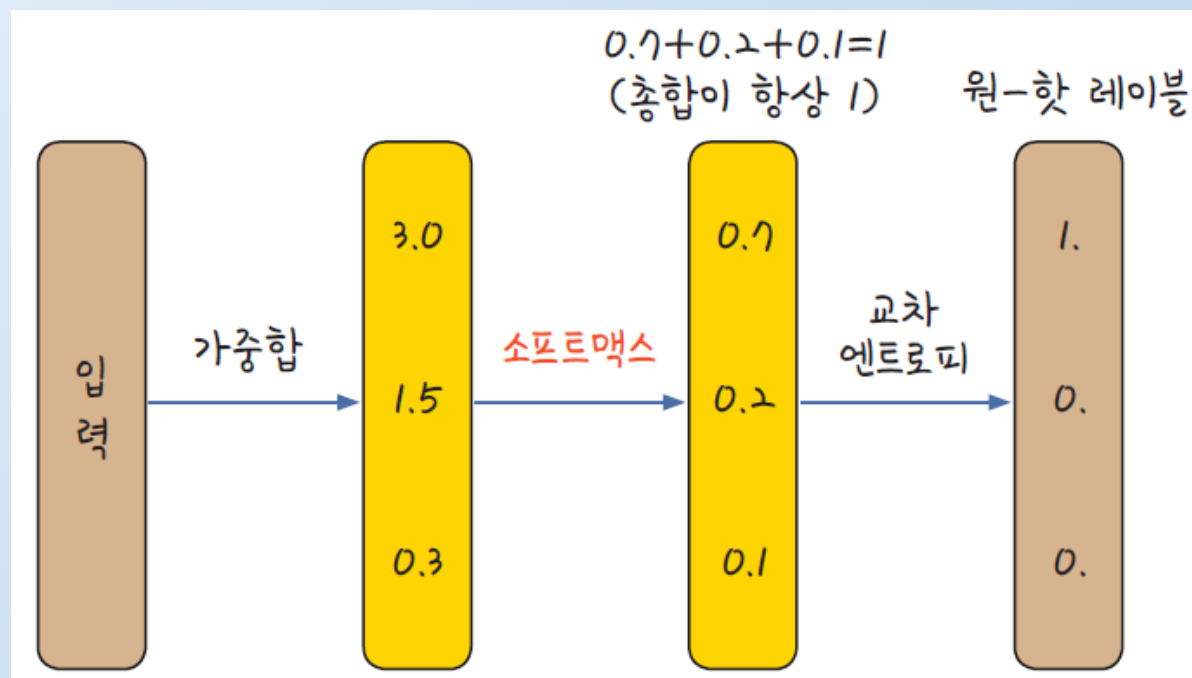


그림 12-3 소프트맥스 함수의 원리

## 5 | 아이리스 품종 예측 실행

- 다중 분류에 적절한 오차 함수인 categorical\_crossentropy를 사용하고, 최적화 함수로 adam을 사용함
- 전체 샘플이 50회 반복될 때까지 실험을 진행하되 한 번에 입력되는 값은 1개로 함

### 코드 12-1 아이리스 품종 예측하기

- 예제 소스: run\_project/03\_Iris\_Multi\_Classfication.ipynb

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.preprocessing import LabelEncoder
```



## 5 | 아이리스 품종 예측 실행

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

# seed 값 설정
np.random.seed(3)
tf.random.set_seed(3)

# 데이터 입력
df = pd.read_csv('../dataset/iris.csv', names = ["sepal_length",
"sepal_width", "petal_length", "petal_width", "species"])
```

## 5 | 아이리스 품종 예측 실행

# 그래프로 확인

```
sns.pairplot(df, hue='species');  
plt.show()
```

# 데이터 분류

```
dataset = df.values  
X = dataset[:,0:4].astype(float)  
Y_obj = dataset[:,4]
```

# 문자열을 숫자로 변환

```
e = LabelEncoder()  
e.fit(Y_obj)  
Y = e.transform(Y_obj)  
Y_encoded = tf.keras.utils.to_categorical(Y)
```

## 5 | 아이리스 품종 예측 실행

# 모델의 설정

```
model = Sequential(✓)
model.add(Dense(16, input_dim=4, activation='relu'))
model.add(Dense(3, activation='softmax'))
```

# 모델 컴파일

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

## 5 | 아이리스 품종 예측 실행

# 모델 실행

```
model.fit(X, Y_encoded, epochs=50, batch_size=1)
```

# 결과 출력

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

## 5 | 아이리스 품종 예측 실행

실행  
결과



Train on 150 samples

Epoch 1/50

150/150 [=====] - 0s 3ms/sample - loss: 1.2541

- accuracy: 0.3733

Epoch 2/50

150/150 [=====] - 0s 1ms/sample - loss: 0.9932

- accuracy: 0.4200

Epoch 3/50

150/150 [=====] - 0s 1ms/sample - loss: 0.7919

- accuracy: 0.6400

(중략)

## 5 | 아이리스 품종 예측 실행

Epoch 47/50

150/150 [=====] - 0s 1ms/sample - loss: 0.0926

- accuracy: 0.9867

Epoch 48/50

150/150 [=====] - 0s 993us/sample - loss:

0.0924 - accuracy: 0.9733

Epoch 49/50

150/150 [=====] - 0s 1ms/sample - loss: 0.0884

- accuracy: 0.9667

Epoch 50/50

150/150 [=====] - 0s 999us/sample - loss:

0.0894 - accuracy: 0.9733

...

Accuracy: 0.9867

# 정리학습 : 다중 분류 문제 해결하기

- 1 | 다중 분류 문제
- 2 | 상관도 그래프
- 3 | 원-핫 인코딩
- 4 | 소프트맥스
- 5 | 아이리스 품종 예측 실행

**다음 수업**

**과적합 피하기**