



아이리스 품종 예측



목차

A table of Contents

#1, Decision Tree Classifier

#2, 아이리스 품종

#3, 머신러닝 아이리스 품종 예측

#4, 딥러닝 아이리스 품종 예측

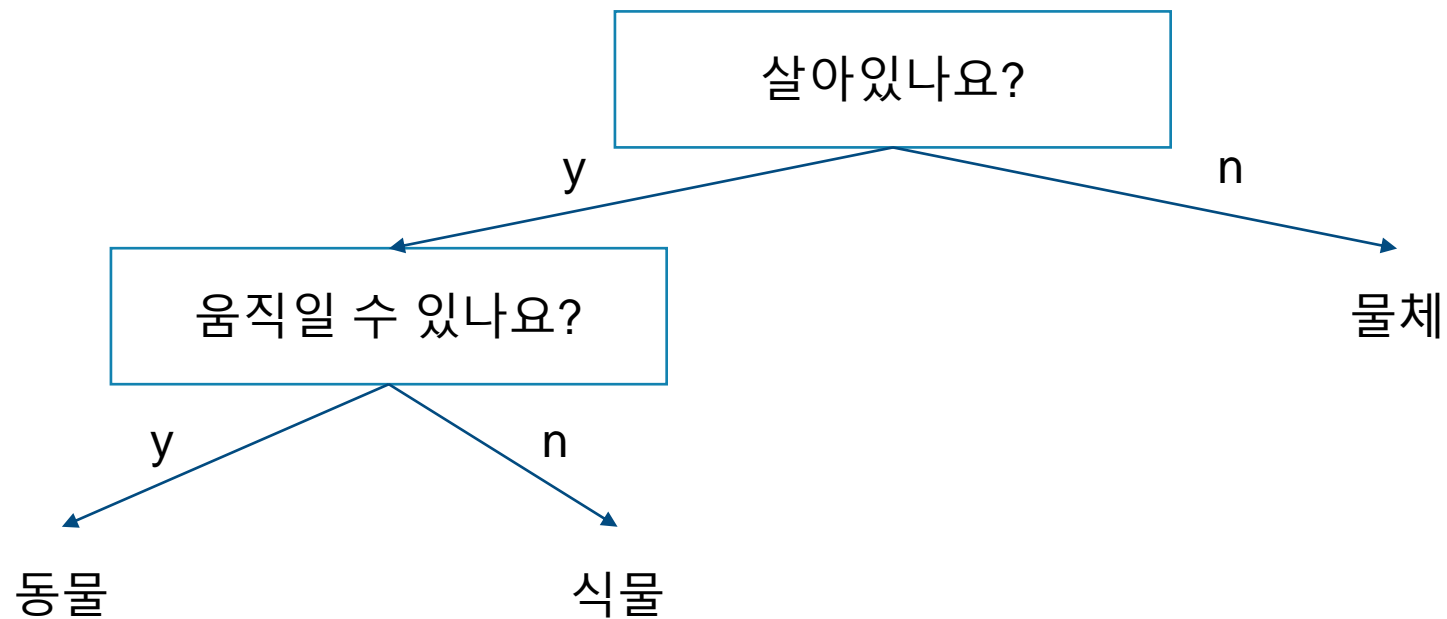
Decision Tree Classifier

Decision Tree Classifier

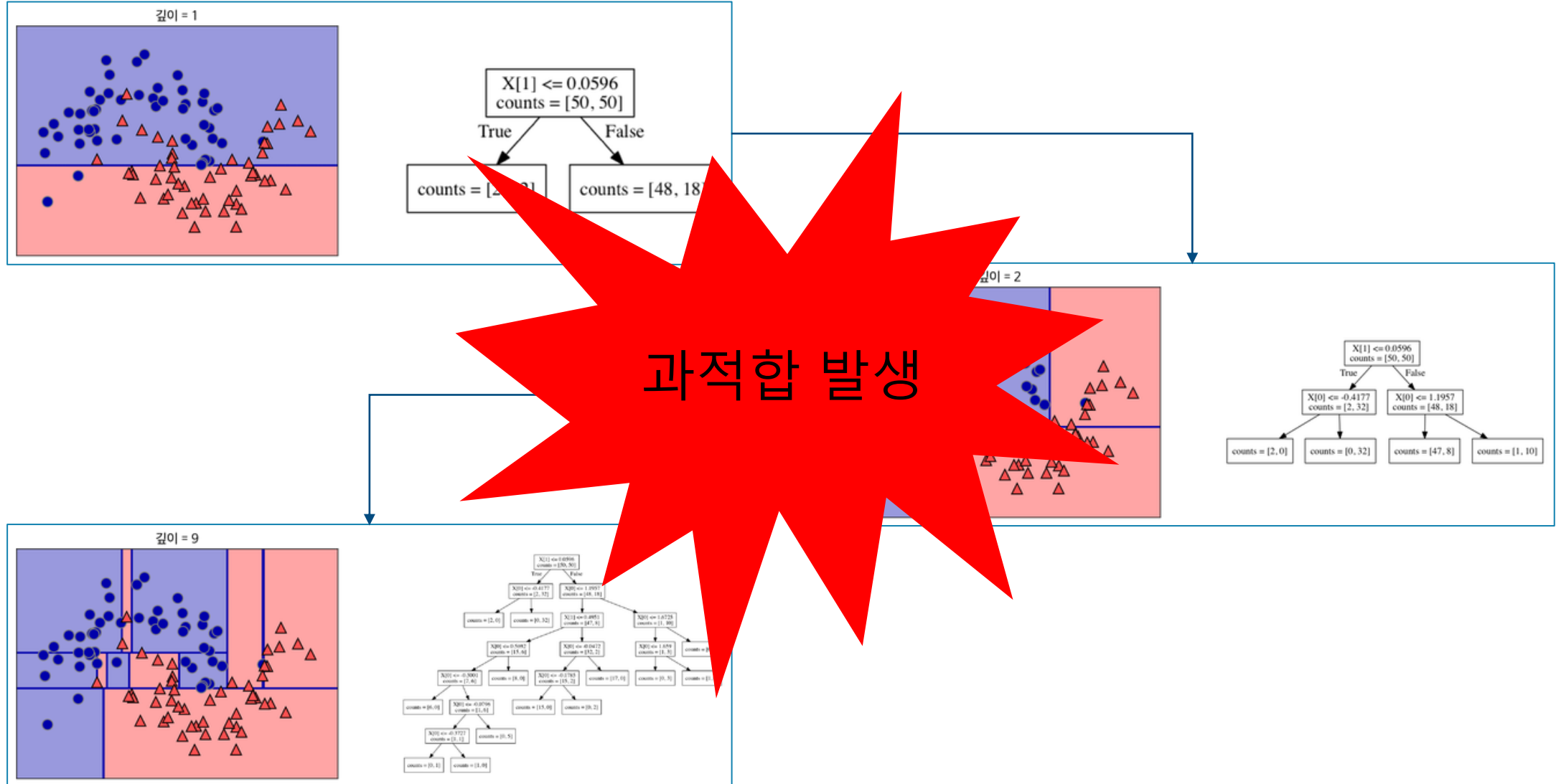
Decision Tree란?

의사 결정 트리

스무고개처럼 예/아니요 질문을 이어가며 학습



Decision Tree Classifier



Decision Tree Classifier

하이퍼파라미터로 과적합을 방지할 수 있음

파라미터 명	설명
min_samples_split	<ul style="list-style-type: none"> - 노드를 분할하기 위한 최소한의 샘플 데이터수 → 과적합을 제어하는데 사용 - Default = 2 → 작게 설정할 수록 분할 노드가 많아져 과적합 가능성 증가
min_samples_leaf	<ul style="list-style-type: none"> - 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수 - min_samples_split과 함께 과적합 제어 용도 - 불균형 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 작게 설정 필요
max_features	<ul style="list-style-type: none"> - 최적의 분할을 위해 고려할 최대 feature 개수 - Default = None → 데이터 세트의 모든 피처를 사용 - int형으로 지정 → 피처 갯수 / float형으로 지정 → 비중 - sqrt 또는 auto : 전체 피처 중 $\sqrt{(\text{피처개수})}$ 만큼 선정 - log : 전체 피처 중 $\log_2(\text{전체 피처 개수})$ 만큼 선정
max_depth	<ul style="list-style-type: none"> - 트리의 최대 깊이 - default = None → 완벽하게 클래스 값이 결정될 때 까지 분할 또는 데이터 개수가 min_samples_split보다 작아질 때까지 분할 - 깊이가 깊어지면 과적합될 수 있으므로 적절히 제어 필요
max_leaf_nodes	리프노드의 최대 개수

Decision Tree Classifier

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
X, y = load_breast_cancer(return_X_y=True)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, stratify=y)

from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier(max_depth=6, random_state=1).fit(X_train, y_train)
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

Decision Tree Classifier

```
model = DecisionTreeClassifier(max_depth=6, random_state=1).fit(X_train, y_train)
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

```
1.0
0.9300699300699301
```

```
model = DecisionTreeClassifier(max_depth=5, random_state=1).fit(X_train, y_train)
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

```
0.9976525821596244
0.9300699300699301
```

```
model = DecisionTreeClassifier(max_depth=4, random_state=1).fit(X_train, y_train)
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

```
0.9859154929577465
0.9300699300699301
```

```
model = DecisionTreeClassifier(max_depth=3, random_state=1).fit(X_train, y_train)
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

```
0.971830985915493
0.9440559440559441
```

```
model = DecisionTreeClassifier(max_depth=2, random_state=1).fit(X_train, y_train)
print(model.score(X_train, y_train))
print(model.score(X_test, y_test))
```

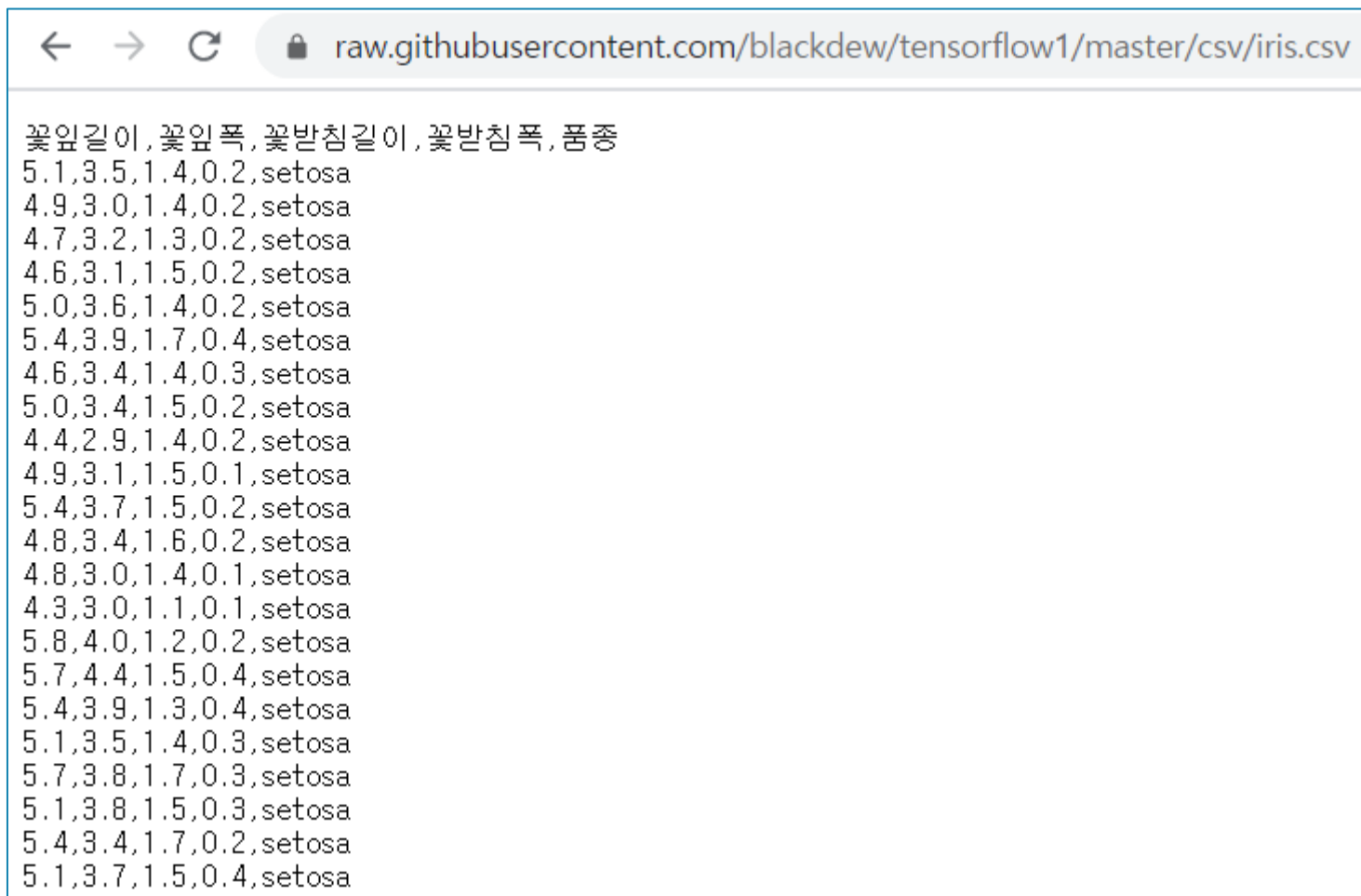
```
0.9647887323943662
0.9370629370629371
```


아이리스 품종

아이리스 품종

꽃잎 길이, 꽃잎 폭, 꽃받침 길이, 꽃받침 폭



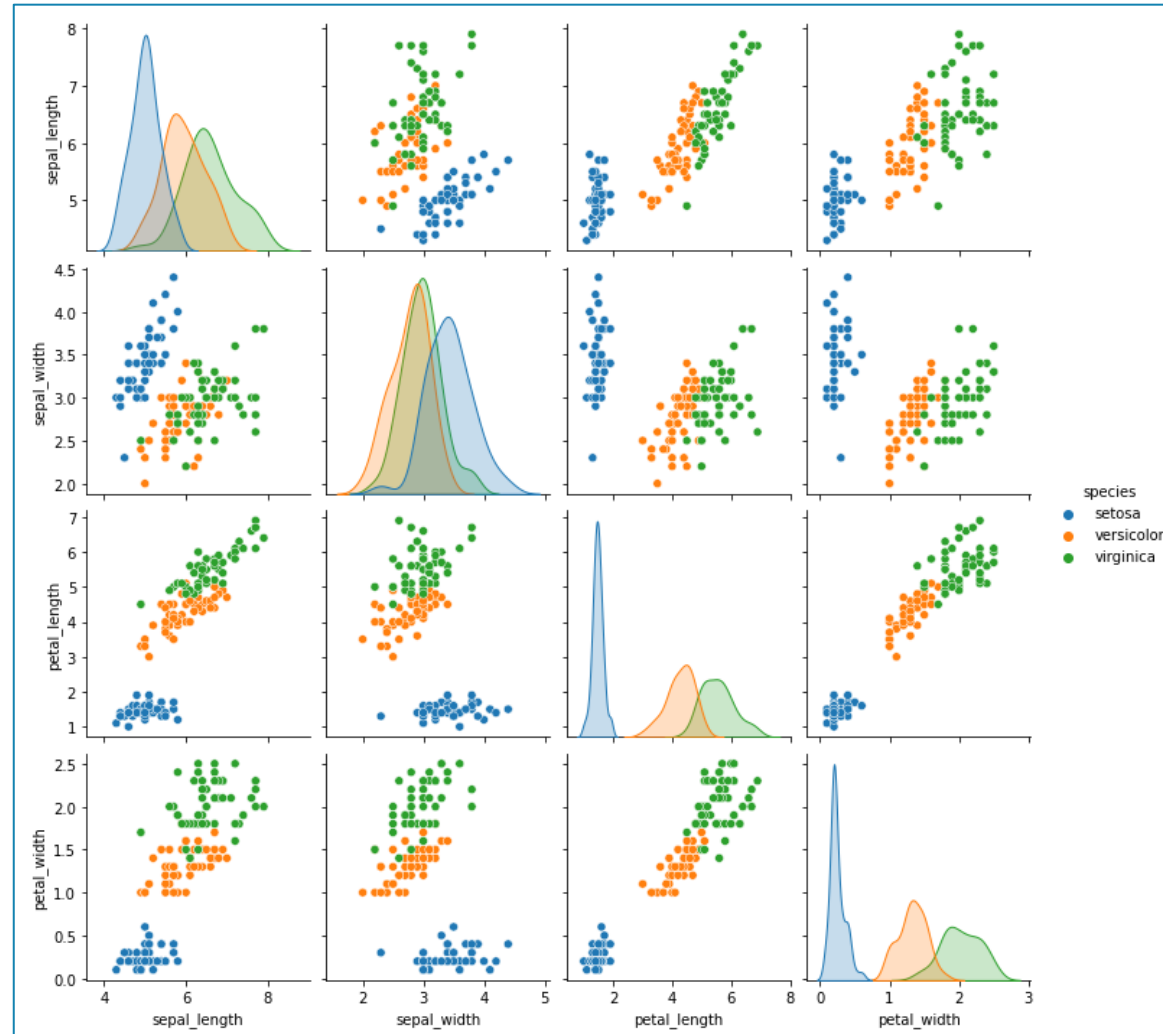


A screenshot of a web browser displaying the raw CSV file for the Iris dataset. The address bar shows the URL: `raw.githubusercontent.com/blackdew/tensorflow1/master/csv/iris.csv`. The page content lists the species names and their corresponding feature values.

꽃잎길이	꽃잎폭	꽃받침길이	꽃받침폭	품종
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa

```
import seaborn as sns
import matplotlib.pyplot as plt
iris = sns.load_dataset("iris")
sns.pairplot(iris, hue='species');
plt.show()
```

아이리스 품종



머신러닝 아이리스 품종 예측

머신러닝 아이리스 품종 예측

```
In [20]: import sklearn
```

```
In [21]: from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

```
In [22]: import pandas as pd

iris = load_iris()

iris_data = iris.data
iris_label = iris.target
```

```
In [23]: X_train, X_test, y_train, y_test = train_test_split(iris_data, iris_label, test_size=0.2, random_state=11)
```

```
In [24]: dt_clf = DecisionTreeClassifier(random_state=11)

dt_clf.fit(X_train, y_train)
```

```
Out[24]: DecisionTreeClassifier(random_state=11)
```

```
In [25]: pred = dt_clf.predict(X_test)

pred
```

```
Out[25]: array([2, 2, 1, 1, 2, 0, 1, 0, 0, 1, 1, 1, 1, 2, 2, 0, 2, 1, 2, 2, 1, 0,
                0, 1, 0, 0, 2, 1, 0, 1])
```

```
In [26]: from sklearn.metrics import accuracy_score
print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test, pred)))
```

예측 정확도: 0.9333

딥러닝 아이리스 품종 예측

딥러닝 아이리스 품종 예측

```
from keras.models import Sequential
from keras.layers.core import Dense
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
```

```
import pandas as pd
import numpy
import tensorflow as tf
```

```
seed = 0
numpy.random.seed(seed)
tf.random.set_seed(seed)
```

```
df = pd.read_csv('../PerfectGuide-master/PerfectGuide-master/2장/iris.csv', names = ["sepal_length", "sepal_width",
                                                                                       "petal_length", "petal_width", "species"])
print(df)
```

	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
...
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

```
[150 rows x 5 columns]
```

딥러닝 아이리스 품종 예측

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

```
e = LabelEncoder()  
e.fit(Y_obj)  
Y = e.transform(Y_obj)  
Y_encoded = np_utils.to_categorical(Y)  #원핫인코딩
```

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

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

원-핫 인코딩

여러 개의 Y 값을 0과 1로만 이루어진 형태로 바꿔주는 기법

```
e = LabelEncoder()
print(Y_obj)
e.fit(Y_obj)
Y = e.transform(Y_obj)
print(Y)
Y_encoded = np_utils.to_categorical(Y) #원-핫인코딩
print(Y_encoded)
```

```
[ 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
  'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
  'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
  'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
  'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
  'setosa' 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
  'versicolor' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
  'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
  'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica'
  'virginica' 'virginica' 'virginica' 'virginica' 'virginica' 'virginica']
```

20

딥러닝 아이리스 품종 예측

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

```
Epoch 1/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0672 - accuracy: 0.9733  
Epoch 2/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0698 - accuracy: 0.9600  
Epoch 3/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0636 - accuracy: 0.9800  
Epoch 4/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0730 - accuracy: 0.9667  
Epoch 5/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0670 - accuracy: 0.9800  
Epoch 6/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0698 - accuracy: 0.9733  
Epoch 7/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0664 - accuracy: 0.9667  
Epoch 8/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0691 - accuracy: 0.9600  
Epoch 9/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0746 - accuracy: 0.9733  
Epoch 10/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0675 - accuracy: 0.9733  
Epoch 11/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0699 - accuracy: 0.9667  
Epoch 12/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0647 - accuracy: 0.9800  
Epoch 13/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0721 - accuracy: 0.9800  
Epoch 14/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0693 - accuracy: 0.9600  
Epoch 15/50  
150/150 [=====] - 0s 2ms/step - loss: 0.0687 - accuracy: 0.9667
```

딥러닝 아이리스 품종 예측

```
Epoch 48/50
150/150 [=====] - 0s 2ms/step - loss: 0.0636 - accuracy: 0.9800
Epoch 49/50
150/150 [=====] - 0s 2ms/step - loss: 0.0605 - accuracy: 0.9800
Epoch 50/50
150/150 [=====] - 0s 2ms/step - loss: 0.0591 - accuracy: 0.9667

<tensorflow.python.keras.callbacks.History at 0x21aaf7250a0>

print("\n Accuracy: %0.4f" %(model.evaluate(X, Y_encoded)[1]))

5/5 [=====] - 0s 3ms/step - loss: 0.0575 - accuracy: 0.9867

Accuracy: 0.9867
```

머신러닝	딥러닝
93.00%	98.67%
1 번의 학습	epoch 수만큼 학습(50)



머신러닝과 딥러닝에 대해 더 많은 공부를 한 뒤
두 모델 모두 개선하여 비교해볼 예정



감사합니다