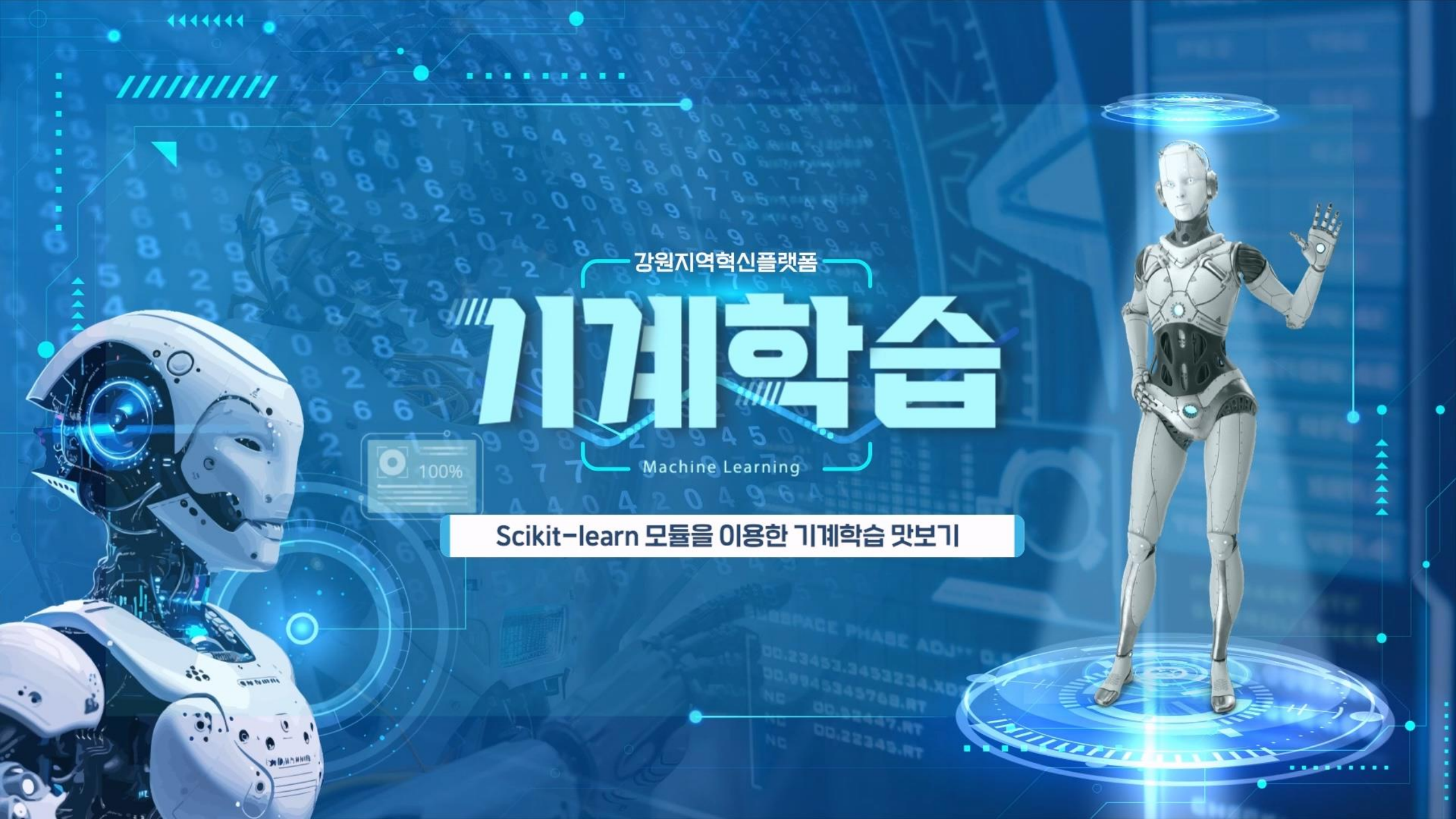
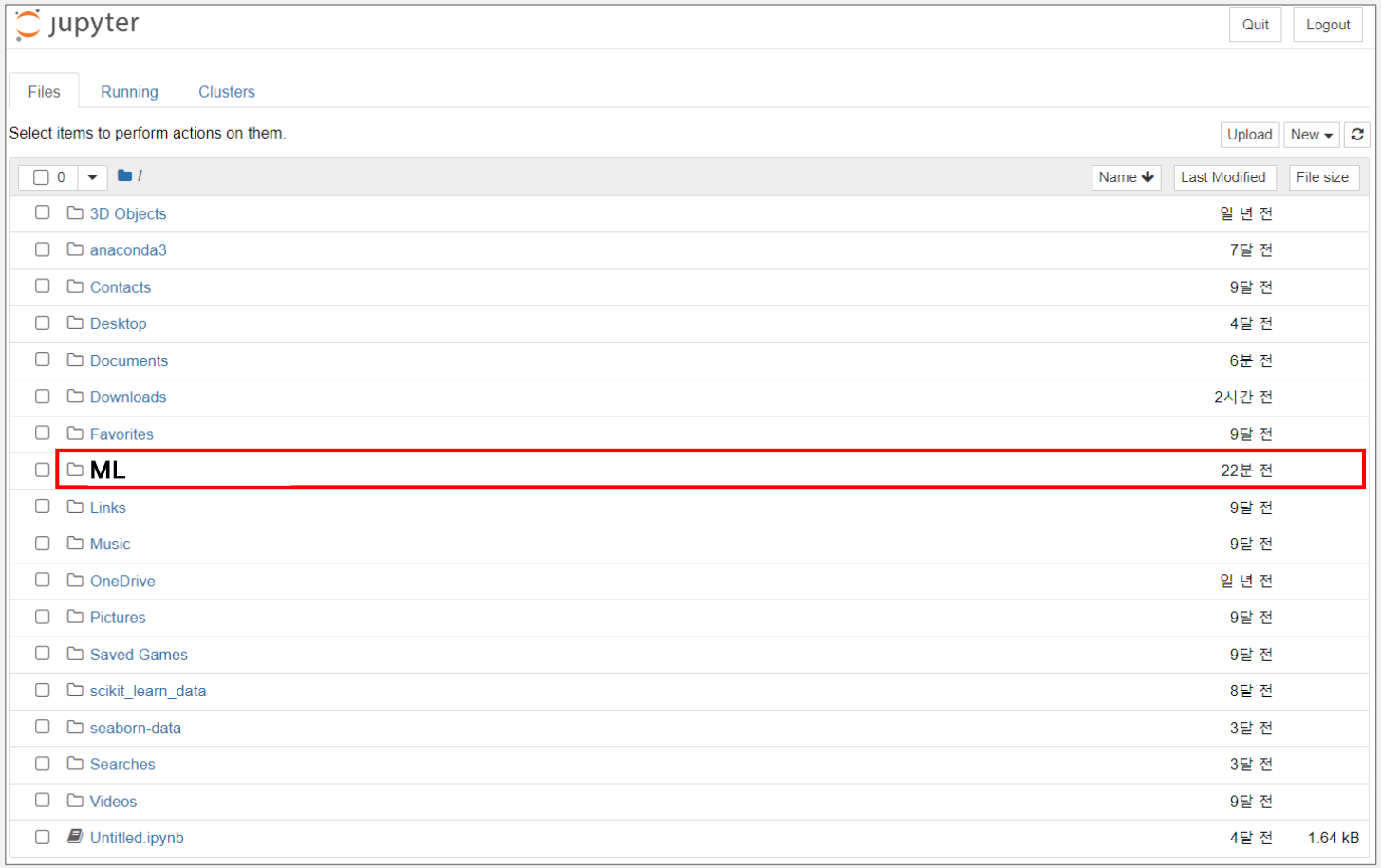

강원지역혁신플랫폼
기계학습
Machine Learning
Scikit-learn 모듈을 이용한 기계학습 맛보기

## 학습목표

Scikit-learn 모듈을 이용한
기계학습 구현을 실행할 수 있습니다.

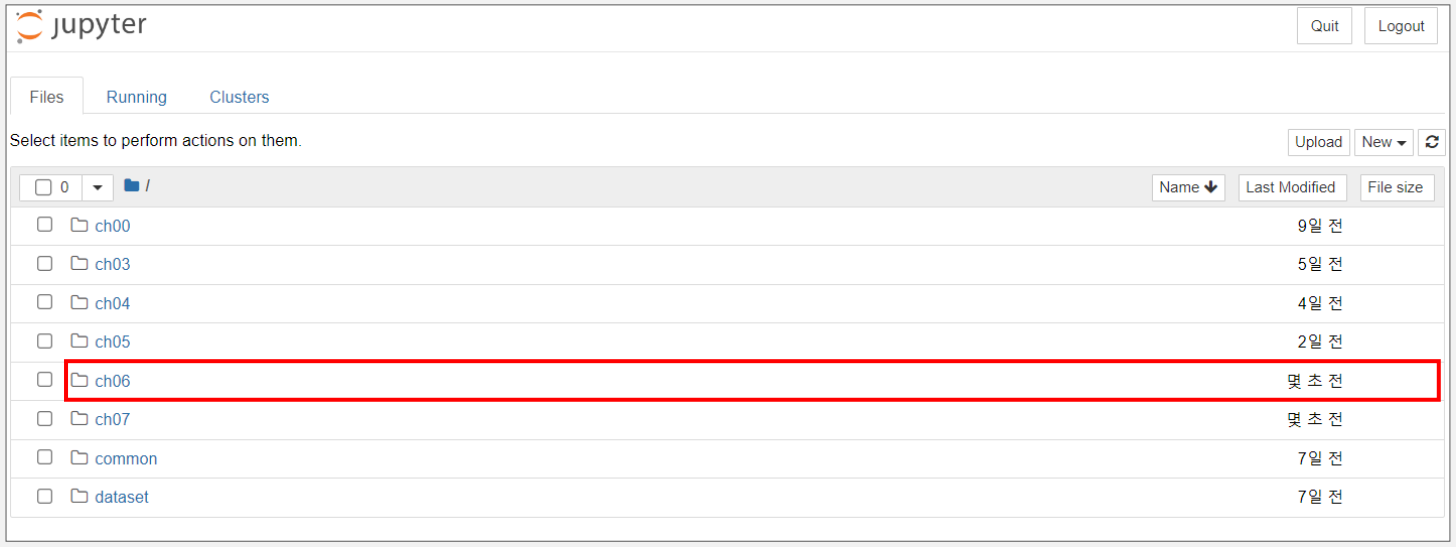
# 01 | ML 폴더

## ML 폴더를 클릭하기



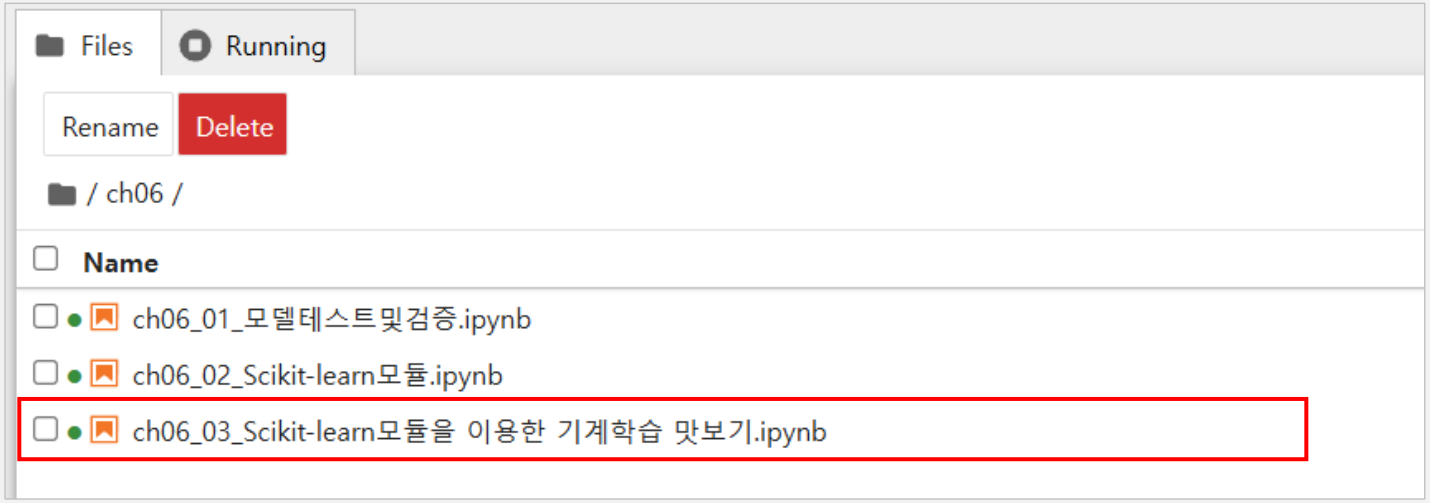
# 02 | ch06 폴더

## ch06 폴더 클릭하기



# 03 | ch06_03_Scikit-learn모듈을 이용한 기계학습 맛보기.ipynb

✦ ch06_03_Scikit-learn모듈을 이용한 기계학습 맛보기.ipynb 파일 클릭하기



## Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 아이리스(iris) 데이터셋을 읽어오는 코드이다.
  - sklearn.utils.Bunch 객체는 몇 가지의 key를 제공하며 이를 통해 데이터의 정보를 쉽게 확인할 수 있음

```
data = load_iris()
print(type(data))     # <class 'sklearn.utils._bunch.Bunch'>
data.keys()           # key를 통해 데이터의 정보를 쉽게 확인
```

```
<class 'sklearn.utils._bunch.Bunch'>
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 아이리스 데이터셋 객체 data에서 DESCR속성을 통해 아이리스 데이터 정보를 확인하는 코드이다.

- 실행결과에서 관측치가 150개, 4개의 독립 변수, 1개의 종속 변수로 구성된 것을 볼 수 있음

```
print(data.DESCR)
```

```
**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 아이리스 데이터셋 객체 data에서 독립 변수 이름과 종속 변수의 레이블을 확인하는 코드이다.
  - 독립 변수 이름은 'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)' 인 것을 볼 수 있음
    - 종속 변수 레이블은 'setosa', 'versicolor', 'virginica' 인 것을 볼 수 있음

```
# 독립 변수 이름
print(data.feature_names)

# 종속변수 레이블
print(data.target_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 분석을 용이하게 하기 위해 아이리스 데이터셋의 독립 변수로 데이터프레임을 생성하는 코드이다.
  - 아래와 같이 데이터 형상이 (150, 4)인 것을 볼 수 있음

```
iris = pd.DataFrame(data=data.data, columns=data.feature_names)
print(iris.shape)  # (150, 4)
iris
```

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **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 |
| **...** | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 |

150 rows × 4 columns

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 아이리스 데이터셋의 종속 변수를 범주형으로 변환하여 판다스의 시리즈로 생성하는 코드이다.
  - 종속 변수가 숫자(int64) 범주형으로 변환되고, 레이블이 0, 1, 2인 것을 볼 수 있음

```
target = pd.Series(data.target, dtype="category")
target
```

```
0       0
1       0
2       0
3       0
4       0
       ..
145     2
146     2
147     2
148     2
149     2
Length: 150, dtype: category
Categories (3, int64): [0, 1, 2]
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 종속 변수의 숫자 범주형을 종속 변수의 레이블로 변환하는 코드이다.

- 종속 변수가 레이블이 'setosa', 'versicolor', 'virginica'으로 변환된 것을 볼 수 있음

```
target = target.cat.rename_categories(data.target_names)
print(target)
```

```
0          setosa
1          setosa
2          setosa
3          setosa
4          setosa
          ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Length: 150, dtype: category
Categories (3, object): ['setosa', 'versicolor', 'virginica']
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 독립 변수로 구성된 데이터프레임에 종속 변수 속성을 추가하는 코드이다.

- 아래와 같이 데이터 형상이 (150, 5)로 변환된 것을 볼 수 있음

```
iris["species"] = target
print(iris.shape)       # (150, 5)
iris
```

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체의 변수 이름을 변경하는 코드이다.

아래와 같이 변수 이름이 변경된 것을 볼 수 있음

```
iris.rename({"sepal length (cm)":"sepal_length", "sepal width (cm)": "sepal_width",
             "petal length (cm)":"petal_length", "petal width (cm)":"petal_width"}, axis=1, inplace=True)
iris
```

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica |

150 rows × 5 columns

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체의 결측값을 확인하는 코드이다.

- 아래와 같이 모든 변수에서 결측값이 없는 것을 볼 수 있음

```
iris.isna().sum(axis=0)
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체 정보를 확인하는 코드이다.

- 아래와 같이 독립 변수 4개는 모두 실수 타입, 종속 변수는 범주형으로 구성된 것을 볼 수 있음

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    category
dtypes: category(1), float64(4)
memory usage: 5.1 KB
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체에서 숫자 타입 변수의 기초 통계량을 확인하는 코드이다.

- 아래와 같이 독립 변수 4개의 기초 통계량이 계산된 것을 볼 수 있음

```
iris.describe()
```

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 5.843333 | 3.057333 | 3.758000 | 1.199333 |
| **std** | 0.828066 | 0.435866 | 1.765298 | 0.762238 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 iris 데이터프레임 객체에서 숫자 타입 변수간의 상관 계수를 계산하는 코드이다.
  - 아래와 같이 독립 변수 4개의 상관 계수가 계산된 것을 볼 수 있음
    - petal_width와 petal_length 변수간의 상관 계수가 약 0.962로 가장 높은 것을 볼 수 있음

```
iris.corr(numeric_only=True)
```

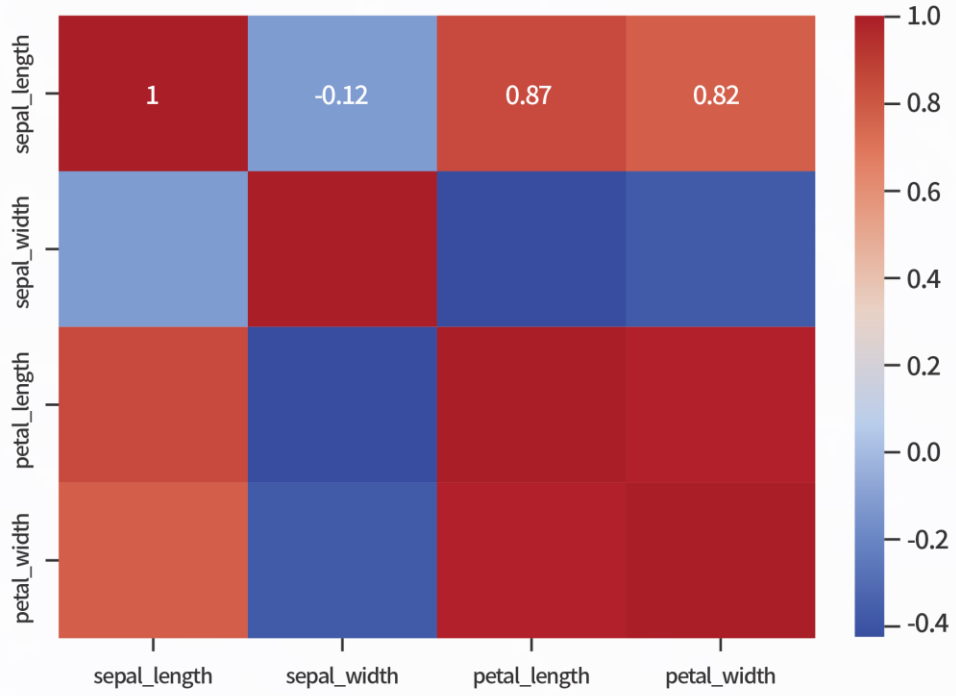
| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| **sepal_length** | 1.000000 | -0.117570 | 0.871754 | 0.817941 |
| **sepal_width** | -0.117570 | 1.000000 | -0.428440 | -0.366126 |
| **petal_length** | 0.871754 | -0.428440 | 1.000000 | 0.962865 |
| **petal_width** | 0.817941 | -0.366126 | 0.962865 | 1.000000 |

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체에서 숫자 타입 변수간의 상관 계수로 히트맵을 그리는 코드이다.

- 아래와 같이 독립 변수 4개의 상관 계수로 히트맵을 볼 수 있음

```
sns.heatmap(iris.iloc[:,:4].corr(), cmap="coolwarm", annot=True, annot_kws={"fontsize":8})
plt.tight_layout()
plt.show()
```



# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 iris 데이터프레임 객체에서 종속 변수의 각 레이블 빈도를 확인하는 코드이다.
  - 아래와 같이 3개 레이블이 각 50개의 빈도로 구성된 것을 볼 수 있음

```
# groupby 메서드를 이용해 종속 변수의 각 레이블 빈도 확인

iris.groupby("species").size()
```

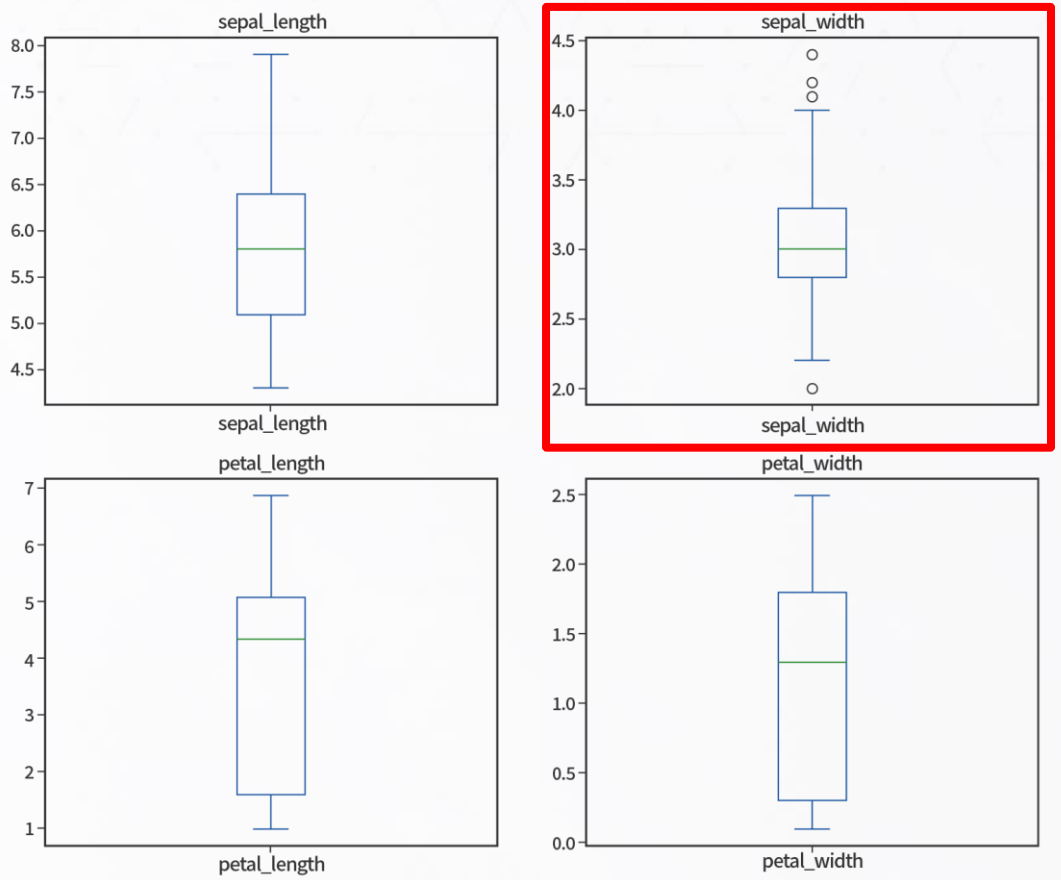
```
species
setosa        50
versicolor    50
virginica     50
dtype: int64
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체에서 독립 변수의 이상치 탐지를 위해 박스플롯(boxplot) 시각화하는 코드이다.

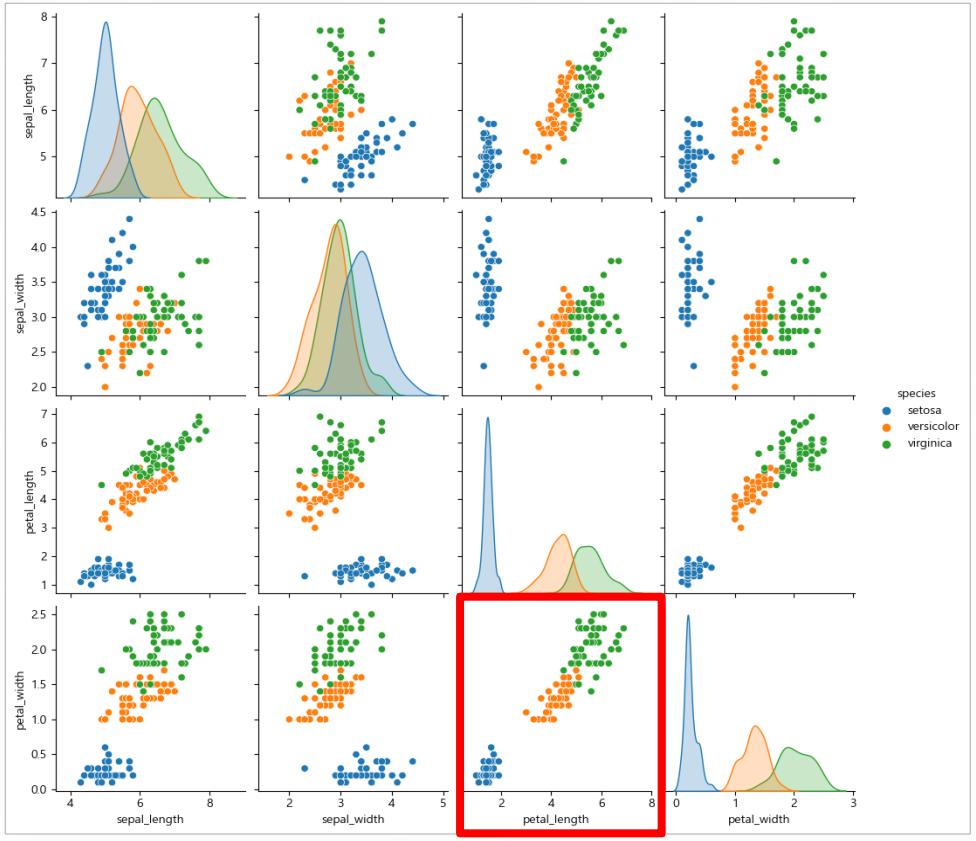
- 아래와 같이 sepal_width 변수에서 이상치가 있는 것을 볼 수 있음

```
def boxplot_iris(feature_names, dataset):
    plt.figure(figsize=(11, 9))
    for i, col in enumerate(feature_names, 1):
        plt.subplot(2, 2, i)
        dataset[col].plot(kind="box")
        plt.title(col)
    plt.show()
boxplot_iris(iris.columns[:-1], iris)
```



# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

- 다음은 iris 데이터프레임 객체에서 독립변수 간의 상관관계 및 데이터 분포를 시각화하는 코드이다.
  - 아래와 같이 petal_width와 petal_length 변수의 상관관계가 높은 것을 볼 수 있음



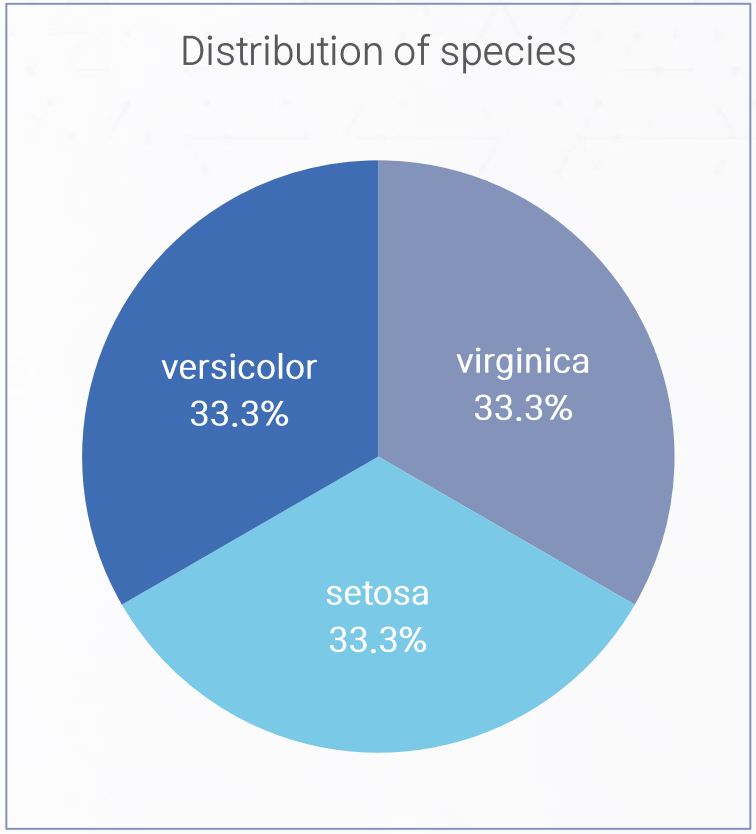

```
sns.pairplot(iris, hue="species")
plt.show()
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 iris 데이터프레임 객체에서 종속 변수의 클래스 비율을 파이차트로 시각화는 코드이다.

- 아래와 같이 종속 변수의 각 레이블의 비율이 33.3%로 동일한 것을 볼 수 있음

```
def piechart_iris(target, dataset):
    plt.figure(figsize=(6, 4))
    labels = []
    sizes = []
    df = dataset.groupby(target).size()
    for key in df.keys():
        labels.append(key)
        sizes.append(df[key])
    plt.pie(sizes, labels=labels,
autopct="%1.1f%%", shadow=True,
startangle=140)
    plt.axis('equal')
    plt.title('Distribution of ' + target)
    plt.show()
piechart_iris('species', iris)
```



# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

학습 및 검증을 수행해 보자.

1. 홀드아웃 검증
   - sklearn의 train_test_split()함수로 훈련 데이터와 테스트 데이터를 7:3 비율로 나눔
   - 여기서는 의사결정나무 모델을 사용함
   - 모델 성능지표는 정확도(accuray)를 사용함

1. 10겹 교차 검증
   - sklearn의 StratifiedShuffleSplit()함수로 훈련 데이터와 테스트 데이터를 7:3 비율로 나눔
   - 여기서는 의사결정나무 모델을 사용함
   - 모델 성능지표는 정확도(accuray)를 사용함

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

## 1 홀드아웃 검증

다음은 train_test_split() 함수로 아이리스 데이터셋을 훈련 데이터와 테스트 데이터를 7:3 비율로 분리하는 코드이다.

- 아래와 같이 데이터를 섞어서 종속 변수의 레이블을 기준으로 7:3 비율로 훈련 데이터와 테스트 데이터가 분리된 것을 볼 수 있음

```
X_train, X_test, y_train, y_test = train_test_split(iris.iloc[:, :-1],
                                   iris.iloc[:, -1], test_size=0.3,
                                   shuffle=True, stratify=iris['species'],
                                   random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(y_train.value_counts())
print(y_test.value_counts())
```

```
(105, 4) (45, 4) (105,) (45,)
setosa        35
versicolor    35
virginica     35
Name: species, dtype: int64
setosa        15
versicolor    15
virginica     15
Name: species, dtype: int64
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 의사결정나무 모델을 생성하고 훈련 데이터로 학습을 수행하는 코드이다.

- 아래와 같이 모델이 훈련 데이터와 훈련 데이터 정답으로 학습된 것을 알 수 있음

```
model = DecisionTreeClassifier(random_state=1234) # 모델 생성
model.fit(X_train, y_train)                       # 학습
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

다음은 학습된 모델에 테스트 데이터로 모델 성능을 평가하는 코드이다.

- 모델 성능 평가결과 정확도가 약 97.78%인 것을 알 수 있음

```
pred = model.predict(X_test)
accuracy = np.round(accuracy_score(y_test, pred), 4)
print('\n## 검증 정확도:', accuracy)    # 검증 정확도: 0.9778
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

## 2 10겹 교차 검증

- 다음은 StratifiedShuffleSplit()함수로 아이리스 데이터셋을 훈련 데이터와 테스트 데이터를 7:3 비율로 분리하고, 의사결정나무 모델로 10겹 교차검증을 수행하는 코드이다.

  - 아래와 같이 데이터를 섞어서 종속 변수의 레이블을 기준으로 7:3 비율로 훈련 데이터와 테스트 데이터가 분리된 것을 볼 수 있음

  - 10겹 교차 검증의 평균 정확도는 약 95.34%인 것을 볼 수 있음

```
------------------------------------------------------------
(105,) (45,)
n_iter= 10
 [('setosa', 35), ('virginica', 35), ('versicolor', 35)] [('virginica', 15), ('versicolor', 15), ('setosa', 15)]
교차 검증 정확도 :0.9333, 학습 데이터 크기: 105, 검증 데이터 크기: 45
검증 세트 인덱스 :[103 120  50   5  14  12  46  22 107  94  43 108 141  62  33  87 114  63
  81 122  44  45   4  41  73  67 134 128 135 119  51  98   0  79 124 102
  49  61   1 132  89  71 123  10  77]
------------------------------------------------------------
검증 정확도
 [0.9556, 0.9333, 0.9556, 0.9556, 0.9556, 0.9556, 0.9333, 0.9778, 0.9778, 0.9333]

## 평균 검증 정확도: 0.9534
```

# 04 | Scikit-learn 모듈을 이용한 기계학습 맛보기

```
sfld = StratifiedShuffleSplit(n_splits=10, test_size=0.3, random_state=0)
cv_accuracy=[]     # KFold 별 정확도 저장
n_iter = 0          # 반복횟수

for train_index, test_index in sfld.split(iris.iloc[:,:-1],iris['species']):
    print(train_index.shape, test_index.shape)
    X_train = iris.iloc[:,:-1].iloc[train_index]
    X_test = iris.iloc[:,:-1].iloc[test_index]
    y_train = iris['species'].iloc[train_index]
    y_test = iris['species'].iloc[test_index]
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    n_iter += 1
    label_train = iris['species'].iloc[train_index]
    label_test = iris['species'].iloc[test_index]
    print("n_iter=",n_iter,"\n",count_frequency(label_train), count_frequency(label_test))

    accuracy = np.round(accuracy_score(y_test, pred), 4)
    train_size = X_train.shape[0]
    test_size = X_test.shape[0]
    print('교차 검증 정확도 :{0}, 학습 데이터 크기: {1}, 검증 데이터 크기: {2}'.format(accuracy,train_size, test_size))
    print('검증 세트 인덱스 :{0}'.format(test_index))
    print('-----------------------------------------------------------')
    cv_accuracy.append(accuracy)
# 개별 iteration별 정확도를 합하여 평균 정확도 계산
print('검증 정확도 \n', cv_accuracy)
print('\n## 평균 검증 정확도:', np.round(np.mean(cv_accuracy), 4))  # 평균 검증 정확도: 0.9534
```