

ch01 머신러닝 시작하기

- Machine Learning with sklearn @ DJ,Lim
- date : 2021/02

01 기본 개념 이해하기

- 샘플(sample) or 데이터 포인트(data point) : 하나의 개체 혹은 행을 말한다.
- 특성 or 속성(feature) : 샘플의 속성, 즉 열을 가르킨다.

02 기본 라이브러리 이해하기

scikit-learn(사이킷 런)

- 오픈 소스입니다.
- 매우 인기 높고 독보적인 파이썬 머신러닝 라이브러리입니다.
- url : <http://scikit-learn.org/stable/documentation> (<http://scikit-learn.org/stable/documentation>)
- 사용자 가이드 : https://scikit-learn.org/stable/user_guide.html (https://scikit-learn.org/stable/user_guide.html)

Numpy

- 파이썬으로 과학 계산을 하기 위한 꼭 필요한 패키지
- 다차원 배열을 위한 기능
- 선형 대수 연산 기능
- 푸리에 변환 같은 고수준 수학 함수와 유사 난수 생성기 기능
- url : <https://www.numpy.org/> (<https://www.numpy.org/>)

SciPy

- SciPy(<https://www.scipy.org/scipylib> (<https://www.scipy.org/scipylib>)) 과학 계산을 함수를 모아놓은 파이썬 패키지.
- 고성능 선형대수 기능, 함수 최적화, 신호 처리, 특수한 수학 함수와 통계 분포 등
- 희소 행렬 기능

Matplotlib

- 파이썬 대표적인 과학 계산을 그래프 라이브러리

Pandas

- 데이터 처리와 분석을 위한 파이썬 라이브러리

- url : <https://pandas.pydata.org/> (<https://pandas.pydata.org/>).
- R의 data.frame을 본떠 설계한 **Dataframe**이라는 데이터 구조를 기반으로 만들어짐.
- SQL 처럼 테이블 쿼리나 조인을 수행 가능함.
- xls, csv등의 다양한 파일과 데이터베이스에서 데이터를 읽어들일 수 있음.
- 참고 도서 : 파이썬 라이브러리를 활용한 데이터 분석
- https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html
(https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html)

mglearn

- 깃허브에 있는 코드와 함께 작성.

In [37]:

```
from IPython.display import display, Image
```

03 라이브러리 소프트웨어 버전 확인

```
라이브러리이름.__version__
라이브러리이름.version
```

In [38]:

```
import sys
print("파이썬 버전 :", sys.version)
```

파이썬 버전 : 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]

In [39]:

```
import pandas as pd
print("판다스 버전 :", pd.__version__)
```

판다스 버전 : 1.0.1

In [40]:

```
import matplotlib
import numpy as np
import scipy as sp
```

직접 해보기1

- matplotlib, numpy, seaborn, scipy의 각각의 버전을 확인해 보자.
- 추가 라이브러리를 설치할 때 명령
 - pip install [라이브러리명]

04 머신러닝 모델을 위한 iris 데이터를 준비

데이터 : 붓꽃

- 종류 : setosa, versicolor, virginica
- 데이터 내용 : 붓꽃의 꽃잎과 꽃받침
- 우리가 해결하려고 하는 문제 : 데이터를 주고 붓꽃의 종류 예측하기

In [41]:

```
display(Image(filename='img/iris_setosa01.png'))
```



SETOSA



virginica



Verisicolor

[Wiki 참조](#)

용어 이해하기

- 클래스(class) : 출력될 수 있는 값들. 붓꽃의 종류들, 붓꽃의 종류는 세 클래스 중 하나에 속한다.
- 레이블(label) : 데이터 포인트 하나(붓꽃 하나)에 대한 기대 출력. **특정 데이터 포인트에 대한 출력**

데이터 준비

- load_iris가 반환한 iris객체는 파이썬의 딕셔너리(Dictionary)와 유사한 Bunch 클래스의 객체

7

In [44]:



```
# iris 데이터 셋의 설명 확인
iris['DESCR']
```

Out[44]:

```
'.. _iris_dataset: Iris plants dataset-----**Data Set Charac
teristics:**
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!)
=====
Missing Attribute Values: None
Class Distribution: 33.3% for each of 3 classes.
Creator: R.A. Fisher
Donor: Michael Marshall (MARS HALL%PLU@io.arc.nasa.gov)
Date: July, 1988
The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.
This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.
.. topic:: References
- Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to Mathematical Statistics" (John Wiley, N Y, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis. (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.
- Dasarthy, B.V. (1980) "Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al's AUTOCLASS II conceptual clustering system finds 3 classes in the data.
- Many, many more ...'
```

In [45]:



```
# iris 데이터 셋의 행열 확인
print( iris['data'].shape )
print( iris['feature_names'])
print( iris['data'][:5])      # 5개의 데이터 확인
print( iris['target_names'][:5])
print( iris['target'][:5])
```

```
(150, 4)
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
['setosa' 'versicolor' 'virginica']
[0 0 0 0 0]
```

데이터의 크기 확인

- 데이터의 사이즈 확인 : 데이터.shape
- 데이터의 자료형 확인 : type(데이터자료형)

In [46]:



```
print(iris['target'].shape) # 타겟
print(iris['data'].shape)
print(type(iris['target']), type(iris['data']))
```

```
(150,)
(150, 4)
<class 'numpy.ndarray'> <class 'numpy.ndarray'>
```

05 데이터를 훈련 데이터와 테스트 데이터로 나누기

- 학습용 데이터 : 실제 공부를 위한 데이터 셋(실제 문제지의 문제)
- 테스트 데이터 : 공부 후, 실제 잘 동작하는지 확인하기 위한 데이터 (모의고사시험)
- 내용 : 모델을 새 데이터에 적용하기 전에 우리가 만든 모델이 잘 동작하는지 확인하기 위해 테스트 데이터를 활용하여 평가한다.

훈련 데이터 셋(training set) : 머신러닝 모델을 만들 때 쓰는 데이터 셋

테스트 데이터 셋(test set) : 모델이 얼마나 잘 작동하는지 쓰는 데이터 셋

- 테스트 데이터 셋을 또는 홀드아웃(hold-out set)이라 한다.
- scikit-learn은 데이터 셋을 나눠주기 위해 train_test_split 함수를 이용.
- train_test_split 함수는 기본적으로 75% 훈련 세트, 25%의 테스트 세트

In [47]:



```
from sklearn.model_selection import train_test_split

X = iris['data']
y = iris['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

In [48]:



```
# 데이터 사이즈
print(X_train.shape) # 훈련 데이터 셋 사이즈
print(X_test.shape)  # 테스트 데이터 셋 사이즈
print(y_train.shape) # 훈련 데이터 레이블 사이즈
print(y_test.shape)  # 테스트 데이터 레이블 사이즈
```

```
(112, 4)
(38, 4)
(112,)
(38,)
```

06. 데이터 살펴보기 - 시각화

- 머신러닝 모델을 만들기 전 머신러닝 없이도 풀 수 있는 문제는 아닌지, 혹은 필요한 정보가 누락이 없는지 확인
- 산점도(SCATTER PLOT)를 이용하여 확인.
 - 한특성을 x축에 놓고 다른 하나는 y축에 놓아 데이터를 점으로 나타내는 그래프
- 2개의 변수만 사용 가능하여 산점도 행렬(scatter matrix)를 사용

In [49]:



```
import seaborn as sns
```

In [50]:



```
print(iris['feature_names']) # 붓꽃의 꽃잎과 꽃받침의 feature 이름
```

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

In [51]:



```
iris_df = pd.DataFrame(X_train, columns=iris.feature_names)
iris_df['species'] = y_train
iris_df['species'] = iris_df['species'].astype('category') # 자료형 변환
iris_df.head()
```

Out[51]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.9	3.0	4.2	1.5	1
1	5.8	2.6	4.0	1.2	1
2	6.8	3.0	5.5	2.1	2
3	4.7	3.2	1.3	0.2	0
4	6.9	3.1	5.1	2.3	2

In [52]:



```
print(iris_df.shape)
print(iris_df.info())
```

(112, 5)

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 112 entries, 0 to 111

Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
0	sepal length (cm)	112 non-null	float64
1	sepal width (cm)	112 non-null	float64
2	petal length (cm)	112 non-null	float64
3	petal width (cm)	112 non-null	float64
4	species	112 non-null	category

dtypes: category(1), float64(4)

memory usage: 3.8 KB

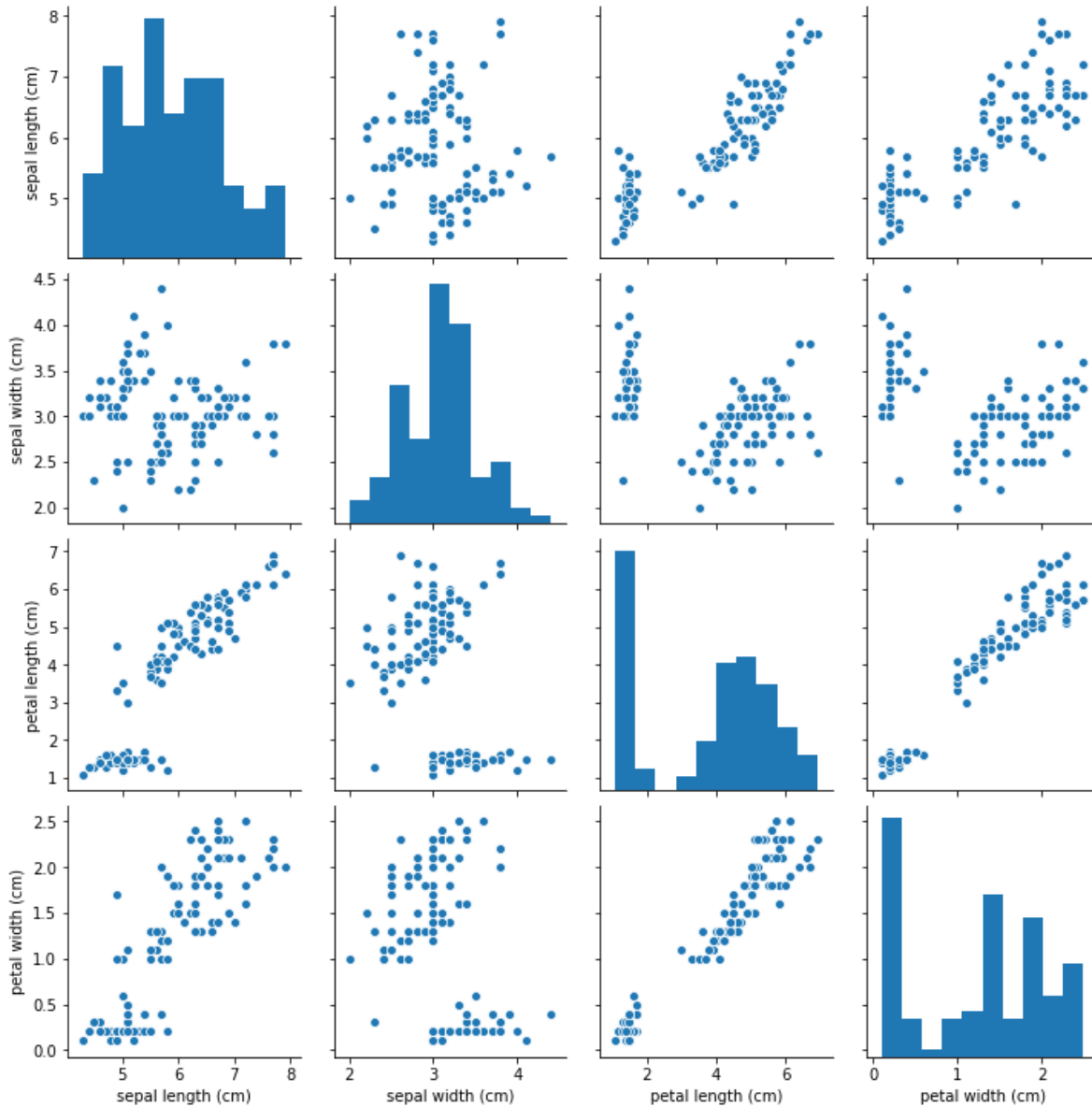
None

In [53]:

```
sns.pairplot(iris_df.iloc[ : ,0:4]) # 1~4열 선택
```

Out[53]:

<seaborn.axisgrid.PairGrid at 0x262d10d1d88>



In [54]:

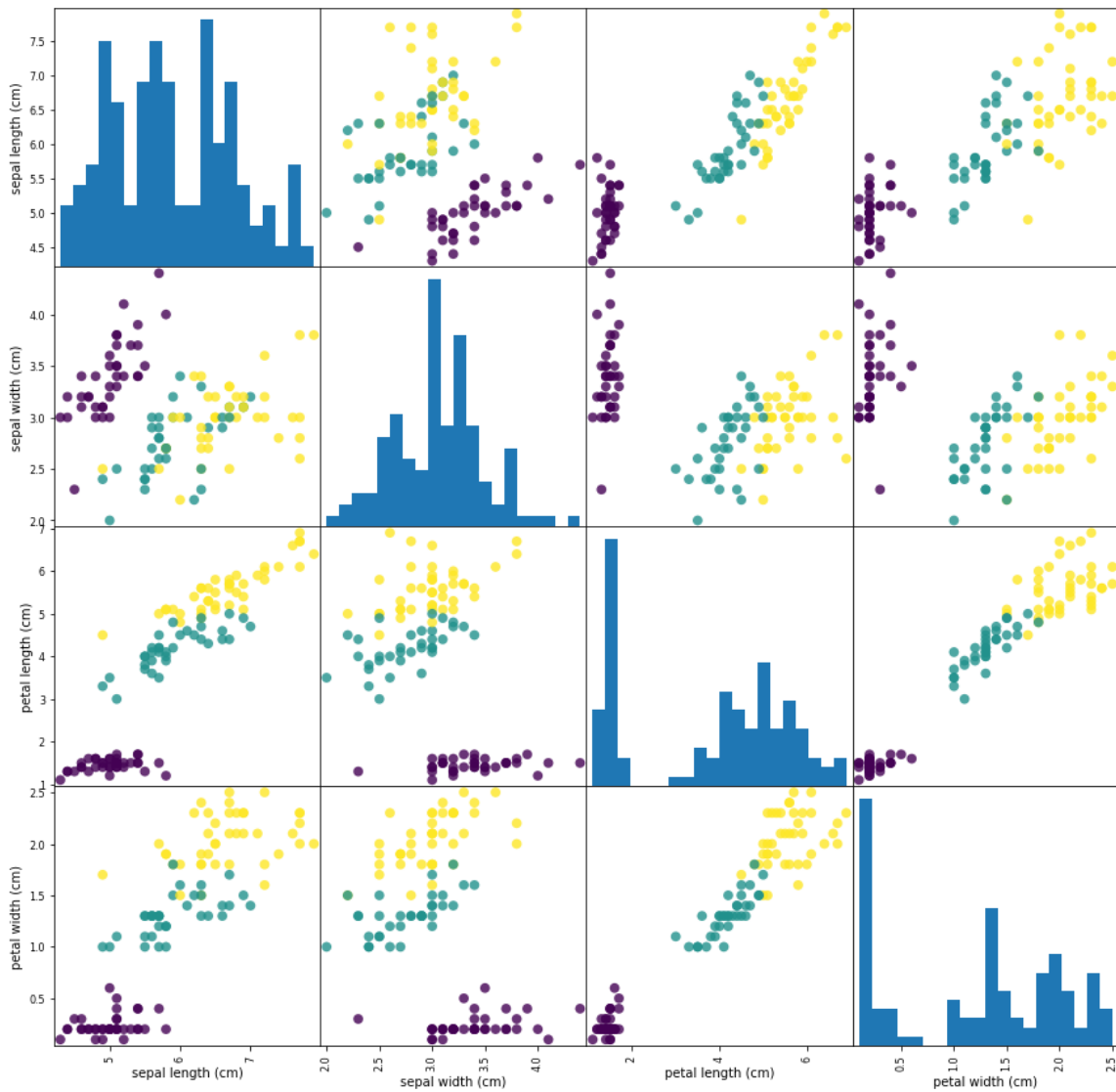


```
pd.plotting.scatter_matrix(iris_df, c=y_train,      # 색
                             figsize=(15,15),      # 크기
                             marker='o',           # 표시
                             hist_kwds={'bins':20}, # 막대의 개수
                             s=60,                # size
                             alpha=0.8 ) # 투명도
```

Out [54]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000262D7A35908>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D7D08DC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D7F46308>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D7F7CCC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000262D7FB5708>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D7FF0108>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D8025848>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D805ECC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000262D806A888>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D80A2A48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D810E048>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D81440C8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000262D817E1C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D81B6308>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D81EF448>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000262D8228548>]],
      dtype=object)
```





07 첫번째 머신러닝 모델 만들기

- k-최근접 이웃(k-nearest neighbors, k-NN) 알고리즘 :
 - 훈련 데이터에서 새로운 데이터 포인트에 가장 가까운 'k개'의 이웃을 찾는다.
 - 이웃들의 클래스 중 빈도가 가장 높은 클래스를 예측값으로 사용

모델 만들기

In [55]:

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=2)
```

모델 학습시키기

In [56]:

```
model.fit(X_train, y_train)
```

Out [56]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                     weights='uniform')
```

하나의 데이터로 예측해 보기

In [57]:

```
X_new = np.array([[5, 2.9, 1, 0.2]])
```

In [58]:

```
### 예측시키기
pred = model.predict(X_new)
pred_targetname = iris['target_names'][pred]
print("예측 : ", pred)
print("예측한 타겟의 이름: ", pred_targetname)
```

```
예측 : [0]
예측한 타겟의 이름: ['setosa']
```

08 내가 만든 모델 평가하기

In [35]:

```
y_pred = model.predict(X_test)
print("예측값 : \n", y_pred)
```

```
예측값 :
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
```

In [36]:

```
print("테스트 세트의 정확도 : {:.2f}".format(np.mean(y_pred == y_test)))
```

```
테스트 세트의 정확도 : 0.97
```

실습해 보기

- titanic 데이터 셋을 활용하여 knn 모델을 구현해 보자

REF

sklearn score 매개변수 : https://scikit-learn.org/stable/modules/model_evaluation.html (https://scikit-learn.org/stable/modules/model_evaluation.html)

History

- 내용 확인 및 업데이트 2021/02/16 v11

In []:

