Machine Learning with Python

Introduction

Contacts

Haesun Park

Email : haesunrpark@gmail.com

Meetup: https://www.meetup.com/Hongdae-Machine-Learning-Study/

Facebook : https://facebook.com/haesunrpark

Blog : https://tensorflow.blog

Book

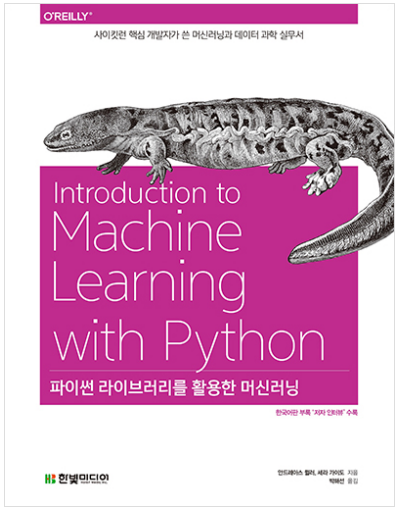
파이썬 라이브러리를 활용한 머신러닝, 박해선.

(Introduction to Machine Learning with Python, Andreas Muller & Sarah Guido의 번역서입니다.)

번역서의 1장과 2장은 블로그에서 무료로 읽을 수 있습니다.

원서에 대한 프리뷰를 온라인에서 볼 수 있습니다.

Github:
https://github.com/rickiepark/introduction_to_ml_with_python/



소개

머신러닝이란

데이터에서 지식을 추출하는 작업

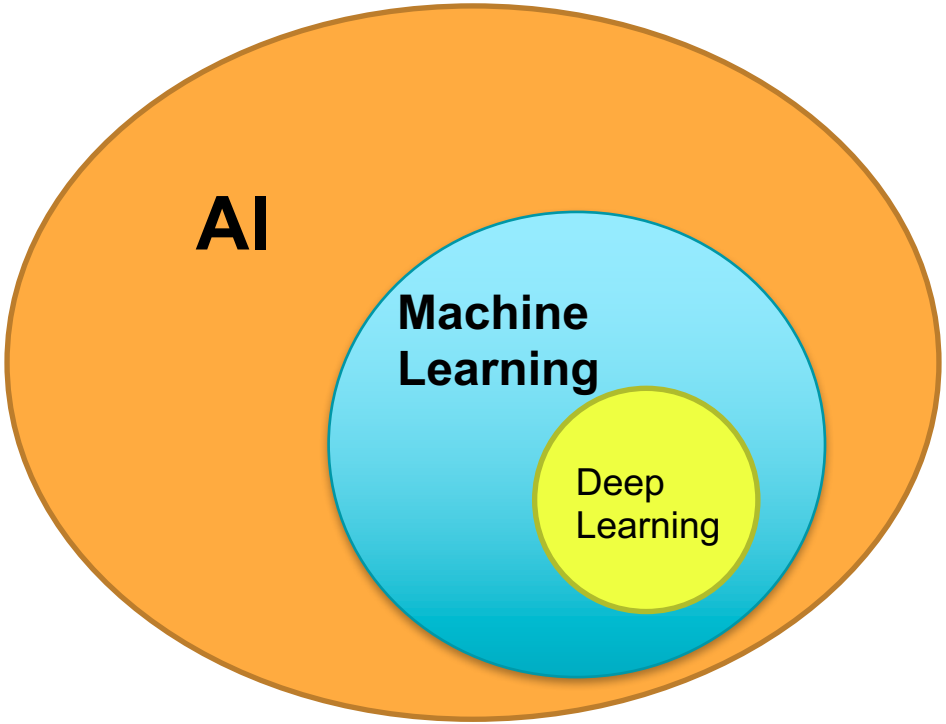
통계학, 인공지능, 컴퓨터 과학이 어우러진 연구분야

예측 분석이나 통계적 머신러닝으로도 불림

위키피디아 “인공지능의 한 분야로 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야”

Arthur Samuel “Field of study that gives computers the ability to learn without being explicitly programmed”

AI ⊃ 머신러닝 ⊃ 딥러닝



왜 머신러닝인가?

규칙기반 전문가 시스템(Rule based expert system)의 문제점

결정에 필요한 로직이 한 분야나 작업에 국한, 작업 변경에 따라 시스템 개발을 다시 수행할 수도 있음

규칙을 설계하려면 분야의 전문가들의 결정 방식에 대해 잘 알아야 함

2001년 이전까지 얼굴 인식 문제를 풀지 못함

컴퓨터의 픽셀 단위는 사람이 인식하는 방식과 다름, 얼굴이 무엇인지 일련의 규칙을 만들기 어려움

머신러닝으로 많은 얼굴 이미지를 제공하면 얼굴을 특정하는 요소를 찾을 수 있음

지도 학습(Supervised Learning)

알고리즘에 입력과 기대하는 출력을 제공

알고리즘은 입력으로 부터 기대하는 출력을 만드는 방법을 찾음

스팸 문제의 경우 이메일(입력)과 스팸 여부(기대 출력)을 제공해야 함

지도 학습의 예

손글씨 숫자 판별: 데이터 수집에 수작업이 많음. 비교적 쉽고 적은 비용 소모.

의료 영상에 기반한 암진단: 도덕적/개인정보 문제 고가의 장비, 전문가 의견 필요

신용카드 부정거래 감지: 고객에게 신고가 올 때까지 기다리면 됨.

비지도 학습(Unsupervised Learning)

알고리즘에 입력은 주어지지만 출력은 제공되지 않음

따라서 비지도 학습의 성공을 평가하기 어려움

비지도 학습의 예

- 블로그 글의 주제: 사전에 어떤 주제가 있는지 얼마나 많은 주제가 있는지 모름
- 고객의 취향 그룹: 부모, 독서광, 게이머 등 어떤 그룹이 얼마나 많이 있는지 모름
- 웹사이트 비정상적 접근: 부정행위나 버그 감지를 위한 비정상적인 패턴은 각기 다르고 가지고 있는 비정상 데이터가 없을 수 있음.

데이터, 특성

특성 추출, 특성 공학: 입력 특성을 만들어 내는 일

문제와 데이터 이해

알고리즘마다 잘 들어맞는 데이터나 문제의 종류가 다름

- 어떤 질문에 대답을 원하는가? 원하는 답을 만들 수 있는 데이터를 가지고 있는가?
- 어떻게 머신러닝의 문제로 가장 잘 기술할 수 있는가?
- 충분한 데이터가 있는가?
- 좋은 예측을 위한 특성을 가지고 있는가?
- 애플리케이션의 성과를 어떻게 측정할 것인가?
- 다른 연구나 제품에 어떤 영향이 있는가?

Why Python?

파이썬(Python)

과학 분야를 위한 표준 프로그래밍 언어

MATLAB, R 같은 도메인 특화 언어와 Java, C 같은 범용 언어의 장점을 갖춤

통계, 머신러닝, 자연어, 이미지, 시각화 등을 포함한 풍부한 라이브러리

브라우저 기반 인터랙티브 프로그래밍 환경인 Jupyter Notebook

파이썬 주도 딥러닝 라이브러리: TensorFlow, PyTorch, Theano, ...

Scikit-Learn

오픈소스: https://github.com/scikit-learn/scikit-learn

회귀, 분류, 군집, 차원축소, 특성공학, 전처리, 교차검증, 파이프라인 등 머신러닝에 필요한 도구를 두루 갖춤

풍부한 문서 (영문): http://scikit-learn.org/stable/documentation

학교, 산업 현장에서 널리 사용됨

폭 넓은 커뮤니티

Apple’s Core ML Support

Model type	Supported models	Supported tools
Neural networks	Feedforward, convolutional, recurrent	Caffe Keras 1.2.2+
Tree ensembles	Random forests, boosted trees, decision trees	scikit-learn 0.18 XGBoost 0.6
Support vector machines	Scalar regression, multiclass classification	scikit-learn 0.18 LIBSVM 3.22
Generalized linear models	Linear regression, logistic regression	scikit-learn 0.18
Feature engineering	Sparse vectorization, dense vectorization, categorical processing	scikit-learn 0.18
Pipeline models	Sequentially chained models	scikit-learn 0.18

Scikit-Learn 설치

NumPy, SciPy를 기반으로 함

대화식 환경을 위해서는 IPython 커널과 Jupyter Notebook 설치 필요

Anaconda(https://www.continuum.io/anaconda-overview)

무료, 과학전문 파이썬 배포판, 수백개의 패키지, 맥/윈도우/리눅스 지원, 인텔 MKL 라이브러리 포함

Enthought Canopy(https://www.enthought.com)

과학전문 파이썬 배포판, 학생, 교육 기관에 무료(scikit-learn 미포함)

Python(x, y)

윈도우즈 전용 과학 파이썬 배포판

필수 라이브러리

Jupyter Notebook

프로그램 코드 + 결과 + 문서를 위한 대화식 개발 환경

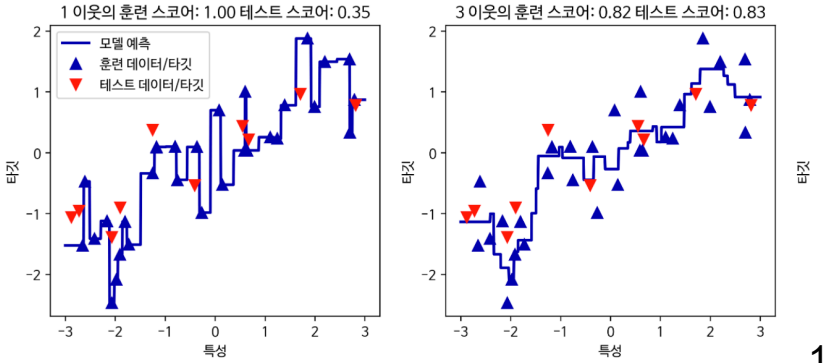
탐색적 데이터 분석에 유리하여 많은 과학자 엔지니어들이 사용

https://jupyter.org

KNeighborsRegressor 분석

In [25]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 4))
# -3 과 3 사이에 1,000 개의 데이터 포인트를 만듭니다
line = np.linspace(-3, 3, 1000).reshape(-1, 1)
for n_neighbors, ax in zip([1, 3, 9], axes):
    # 1, 3, 9 이웃을 사용한 예측을 합니다
    reg = KNeighborsRegressor(n_neighbors=n_neighbors)
    reg.fit(X_train, y_train)
    ax.plot(line, reg.predict(line))
    ax.plot(X_train, y_train, '^', c=mglearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mglearn.cm2(1), markersize=8)

    ax.set_title(
        "{} 이웃의 훈련 스코어: {:.2f} 테스트 스코어: {:.2f}".format(
            n_neighbors, reg.score(X_train, y_train), reg.score(X_test, y_test)
    ax.set_xlabel("특성")
    ax.set_ylabel("타깃")
axes[0].legend(["모델 예측", "훈련 데이터/타깃", "테스트 데이터/타깃"], loc="best")
```

Out[25]: <matplotlib.legend.Legend at 0x114152b00>



NumPy

다차원 배열, 선형 대수, 다양한 수학 함수, 난수 생성기 포함

scikit-learn의 기본 데이터 구조

http://www.numpy.org

http://www.scipy-lectures.org/ 의 1장


```
In [2]: import numpy as np

        x = np.array([[1, 2, 3], [4, 5, 6]])
        print("x:\n{}".format(x))
```

```
x:
[[1 2 3]
 [4 5 6]]
```

SciPy

선형 대수, 최적화, 통계 등 많은 과학 계산 함수를 모아놓은 파이썬 패키지

scikit-learn은 알고리즘 구현에 SciPy에 많이 의존함

0이 많이 포함된 행렬을 효율적으로 표현하기 위한 희소 행렬 scipy.sparse 패키지 주요하게 사용

https://www.scipy.org/scipylib

http://www.scipy-lectures.org/
의 2.5절

```
In [3]: from scipy import sparse

        # 대각선 원소는 1이고 나머지는 0인 2차원 NumPy 배열을 만듭니다.
        eye = np.eye(4)
        print("NumPy 배열:\n{}".format(eye))
```

```
NumPy 배열:
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

단위 행렬

SciPy

In [4]:
```
# NumPy 배열을 CSR 포맷의 SciPy 희박 행렬로 변환합니다.
# 0이 아닌 원소만 저장됩니다.
sparse_matrix = sparse.csr_matrix(eye)
print("\nSciPy의 CSR 행렬:\n{}".format(sparse_matrix))
```

```
SciPy의 CSR 행렬:
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
```

Compressed Sparse Row Format

대각 행렬의 위치

Coordinate Format

In [5]:
```
data = np.ones(4)
row_indices = np.arange(4)
col_indices = np.arange(4)
eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))
print("COO 표현:\n{}".format(eye_coo))
```

```
COO 표현:
  (0, 0)        1.0
  (1, 1)        1.0
  (2, 2)        1.0
  (3, 3)        1.0
```

matplotlib

과학 계산용 그래프 라이브러리

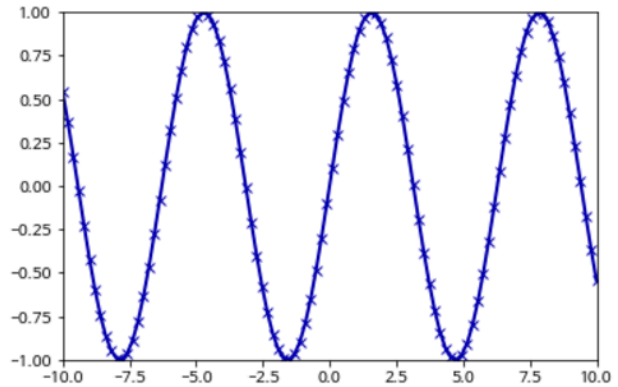
선, 히스토그램, 산점도 등 다양한 그래프

출판 수준의 고품질

%matplotlib inline

https://matplotlib.org/


```
In [6]: %matplotlib inline
        import matplotlib.pyplot as plt

        # -10에서 10까지 100개의 간격으로 나뉘어진 배열을 생성합니다.
        x = np.linspace(-10, 10, 100)
        # 사인 함수를 사용하여 y 배열을 생성합니다.
        y = np.sin(x)
        # plot 함수는 한 배열의 값을 다른 배열에 대응해서 선 그래프를 그립니다.
        plt.plot(x, y, marker="x")

Out[6]: [<matplotlib.lines.Line2D at 0x110403b00>]
```



pandas

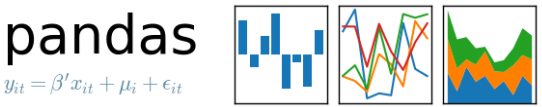
데이터 처리와 분석을 위한 라이브러리

DataFrame inspired by R's data.frame

엑셀과 비슷

numpy와 달리 이종 데이터 포함 가능

http://pandas.pydata.org



In [7]:
```
import pandas as pd

# 회원 정보가 들어간 간단한 데이터셋을 생성합니다.
data = {'Name': ["John", "Anna", "Peter", "Linda"],
        'Location' : ["New York", "Paris", "Berlin", "London"],
        'Age' : [24, 13, 53, 33]
       }

data_pandas = pd.DataFrame(data)
# IPython.display는 주피터 노트북에서 Dataframe을 미려하게 출력해줍니다.
display(data_pandas)
```

	Age	Location	Name
0	24	New York	John
1	13	Paris	Anna
2	53	Berlin	Peter
3	33	London	Linda

In [8]:
```
# Age 열의 값이 30 이상인 모든 행을 선택합니다.
display(data_pandas[data_pandas.Age > 30])
```

	Age	Location	Name
2	53	Berlin	Peter
3	33	London	Linda

Version

Python 3

scikit-learn 0.18.x

matplotlib 2.0.x

NumPy 1.12.x

SciPy 0.19.x

pandas 0.20.x

Iris Dataset

붓꽃의 품종 분류

setosa, versicolor, virginica 종 분류

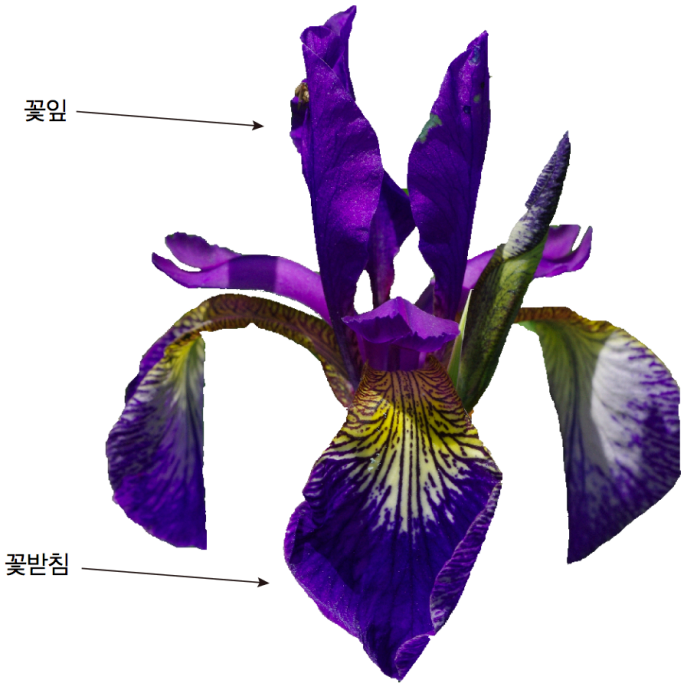
꽃잎petal, 꽃받침sepal의 폭과 길이

사전에 준비한 데이터를 이용하므로 지도 학습

3개의 붓꽃 품종에서 고르는 분류classification

클래스class: 가능한 출력값. 즉 세개의 붓꽃 품종

레이블label: 데이터 포인트 하나에 대한 출력



데이터 적재

붓꽃 데이터: sklearn.datasets.load_iris()

load_digits(), load_boston(), load_breast_cancer(), load_diabetes()

Bunch 클래스

```
In [10]: from sklearn.datasets import load_iris
         iris_dataset = load_iris()

In [11]: print("iris_dataset의 키: {}".format(iris_dataset.keys()))
iris_dataset의 키: dict_keys(['target_names', 'feature_names', 'data', 'target', 'DESCR'])

In [12]: print(iris_dataset['DESCR'][:193] + "\n...")
Iris Plants Database
====================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive att
...
```

데이터셋에 대한 설명
iris_dataset.DESCR[:193] 도 가능

붓꽃 품종의 이름

특성 설명

```
In [13]: print("타깃의 이름: {}".format(iris_dataset['target_names']))
타깃의 이름: ['setosa' 'versicolor' 'virginica']

In [14]: print("특성의 이름: {}".format(iris_dataset['feature_names']))
특성의 이름: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

In [15]: print("data의 타입: {}".format(type(iris_dataset['data'])))
data의 타입: <class 'numpy.ndarray'>

In [16]: print("data의 크기: {}".format(iris_dataset['data'].shape))
data의 크기: (150, 4)
```

데이터 크기
(꽃잎 길이/폭, 꽃받침 길이/폭)

```
In [17]: print("data의 처음 다섯 행:\n{}".format(iris_dataset['data'][:5]))
```

```
data의 처음 다섯 행:
[[ 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]]
```

샘플

훈련 데이터

특성

```
In [18]: print("target의 타입: {}".format(type(iris_dataset['target'])))
```

```
target의 타입: <class 'numpy.ndarray'>
```

```
In [19]: print("target의 크기: {}".format(iris_dataset['target'].shape))
```

```
target의 크기: (150,)
```

타깃 데이터

```
In [20]: print("타깃:\n{}".format(iris_dataset['target']))
```

```
타깃:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

훈련 데이터와 테스트 데이터

훈련에 사용한 데이터는 테스트 (일반화) 에 사용하지 않음

데이터 분리: 훈련 세트, 테스트 세트(홀드아웃holdout 세트),

X(2차원 배열-행렬, 대문자), y(1차원 배열-벡터, 소문자)

```
In [21]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(
             iris_dataset['data'], iris_dataset['target'], random_state=0)
```

3:1 비율 (test_size 매개변수에서 변경 가능)

```
In [22]: print("X_train 크기: {}".format(X_train.shape))
         print("y_train 크기: {}".format(y_train.shape))
```

```
X_train 크기: (112, 4)
y_train 크기: (112,)
```

```
In [23]: print("X_test 크기: {}".format(X_test.shape))
         print("y_test 크기: {}".format(y_test.shape))
```

```
X_test 크기: (38, 4)
y_test 크기: (38,)
```

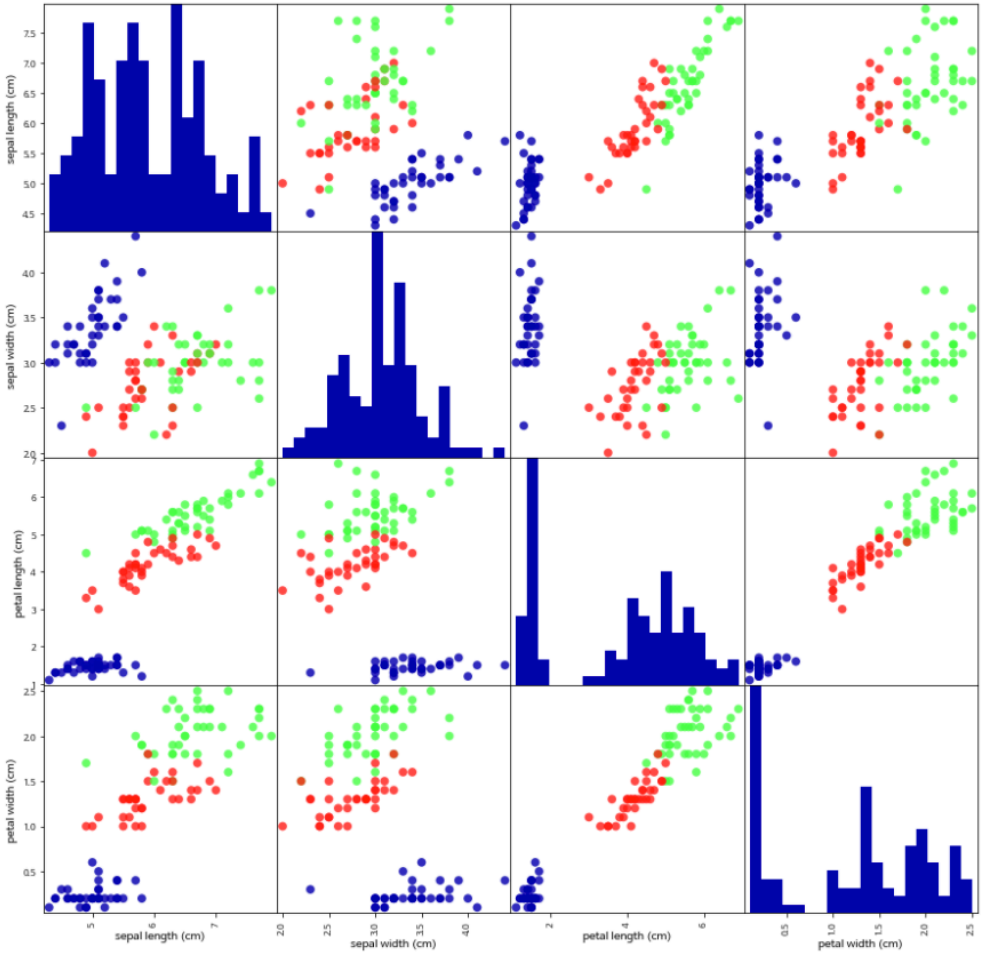
산점도 Scatter Plot

두 개의 특성을 이용 점으로 데이터 표시(3차원 이상은 표현이 어려움)

각 특성의 조합에 대해 모두 그리는 pandas의 산점도 행렬Scatter Matrix 이용

데이터프레임으로 변경

In [24]:
```
# X_train 데이터를 사용해서 데이터프레임을 만듭니다.
# 열의 이름은 iris_dataset.feature_names에 있는 문자열을 사용합니다.
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
# 데이터프레임을 사용해 y_train에 따라 색으로 구분된 산점도 행렬을 만듭니다.
pd.plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15), marker='o',
                           hist_kwds={'bins': 20}, s=60, alpha=.8, cmap=mglearn.cm3)
```


sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)

First ML Application

k-최근접 이웃Nearest Neighbors

훈련 데이터를 저장하는 것이 학습의 전부

새 데이터 포인트에 대해 가장 가까운 훈련 데이터 포인트(k개)를 찾는 것이 예측임

대표적인 인스턴스 기반 학습 알고리즘

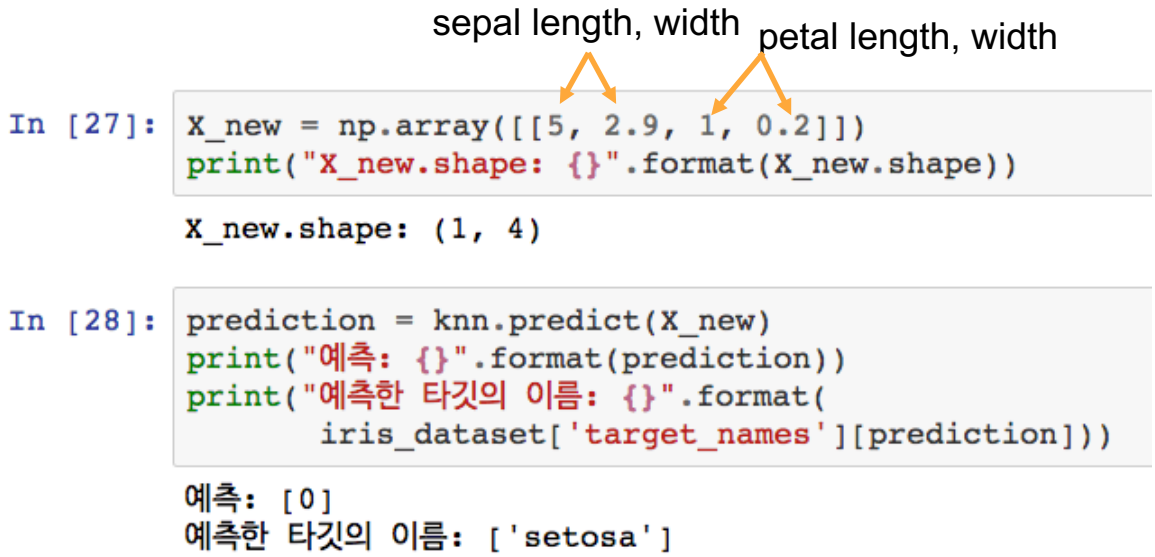
```
In [25]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors=1)

In [26]: knn.fit(X_train, y_train)

Out[26]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                              metric_params=None, n_jobs=1, n_neighbors=1, p=2,
                              weights='uniform')
```

기본값: 5

knn 객체 리턴

예측

학습된 모델로 새로운 붓꽃의 품종을 분류



평가

모델의 성능(정확도)을 평가하기 위해 테스트 세트를 활용

```
In [29]: y_pred = knn.predict(X_test)
         print("테스트 세트에 대한 예측값:\n {}".format(y_pred))
```

```
테스트 세트에 대한 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]
```

0 또는 1의 배열

```
In [30]: print("테스트 세트의 정확도: {:.2f}".format(np.mean(y_pred == y_test)))
```

```
테스트 세트의 정확도: 0.97
```

평균

```
In [31]: print("테스트 세트의 정확도: {:.2f}".format(knn.score(X_test, y_test)))
```

```
테스트 세트의 정확도: 0.97
```

요약

붓꽃 데이터셋을 사용한 지도 학습

세개의 품종(클래스)을 구분하는 분류classification 문제 (다중 분류)

특성 데이터를 담고 있는 X(2차원)와 기대 출력(레이블)을 가진 y(1차원)

훈련 세트(학습)와 테스트 세트(평가)

```
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)

print("테스트 세트의 정확도: {:.2f}".format(knn.score(X_test, y_test)))
```

```
테스트 세트의 정확도: 0.97
```