

Introduction to Machine Learning with Python

1. Introduction

Hongdae Machine Learning Study Epoch #2

Contacts

Haesun Park

Email : haesunpark@gmail.com

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

Facebook : <https://facebook.com/haesunpark>

Blog : <https://tensorflow.blog>

Book

파이썬 라이브러리를 활용한 머신러닝, 박해선.

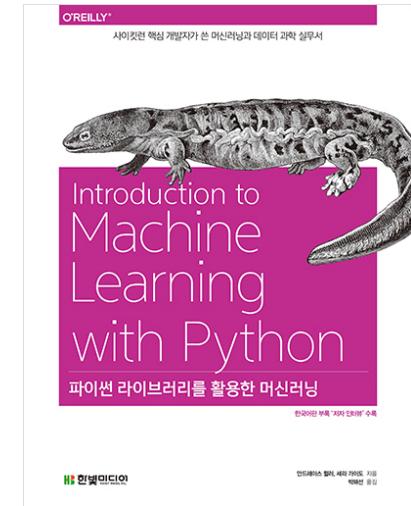
(Introduction to Machine Learning with Python, Andreas Muller & Sarah Guido의 번역서입니다.)

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 작업입니다.(vs 데이터 마이닝)

통계학, 인공지능, 컴퓨터 과학이 어우러진 연구분야입니다.

예측 분석predictive analytics이나 통계적 머신러닝statistical learning으로도 불립니다.

위키피디아의 소개: “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**”

명확한 해결 방법을 모르거나 해결 방법을 찾기 어려운 문제를 다룹니다.

어플리케이션

영화, 쇼핑, 음악 추천

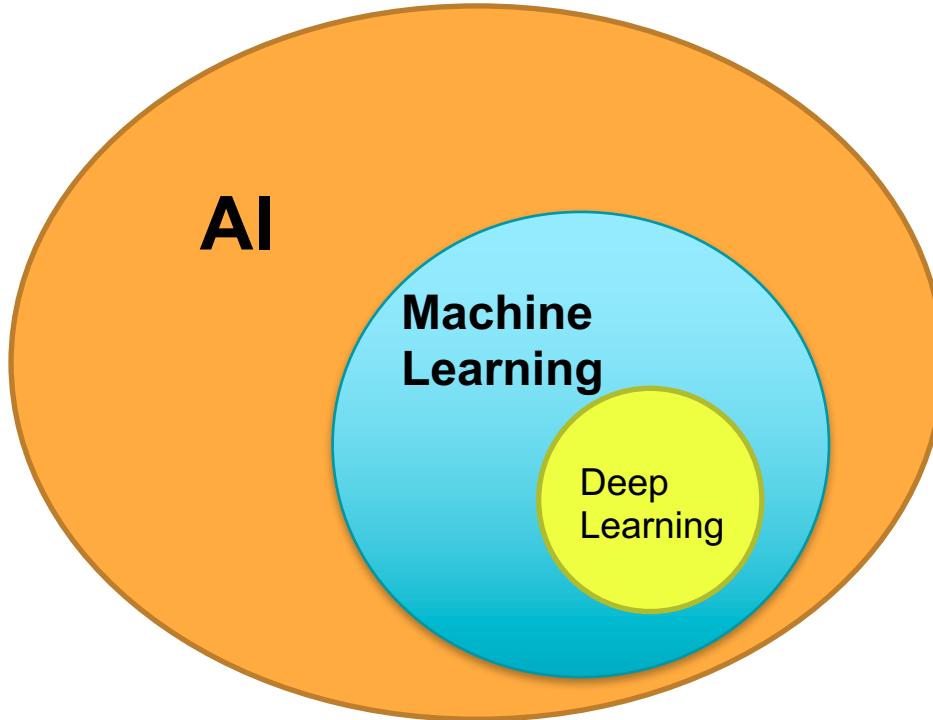
사진 분류, 검색, 친구 추천, 광고, 번역, 음성 인식

의학, 자율주행 자동차, 천문, 양자역학

유럽입자물리연구소
James Beacham



AI ⊃ 머신러닝 ⊃ 딥러닝



왜 머신러닝인가?

규칙기반 전문가 시스템^{rule based expert system}의 문제점(고전적인 스팸 필터)

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

규칙을 설계하려면 분야의 전문가들의 결정 방식에 대해 잘 알아야 합니다.

2001년 이전까지 이미지에서 얼굴을 인식하는 문제를 풀지 못했음

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

머신러닝을 사용해 많은 얼굴 이미지를 제공하면 얼굴을 특정하는 요소를 찾을 수
있습니다.

지도 학습 Supervised Learning

알고리즘에 입력과 기대하는 출력을 제공합니다.

알고리즘은 입력으로부터 기대하는 출력을 만드는 방법을 찾습니다. 혹은 기대하는 출력이 나오도록 알고리즘을 가르칩니다.

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

지도 학습의 예

손글씨 숫자 판별: 손글씨 스캔 이미지(입력), 우편번호(기대 출력)

데이터 수집에 수작업이 많음. 비교적 쉽고 적은 비용 소모.

의료 영상에 기반한 암진단: 영상(입력), 종양 여부(기대 출력)

도덕적/개인정보 문제 고가의 장비, 전문가 의견 필요

신용카드 부정거래 감지: 신용카드 거래내역(입력), 부정거래 여부(기대 출력)
고객에게 신고가 올 때까지 기다리면 됨.

비지도 학습Unsupervised Learning

알고리즘에 입력은 주어지지만 출력은 제공되지 않습니다.

따라서 비지도 학습의 이해하거나 평가하기 어렵습니다.

비지도 학습의 예

블로그 글의 주제 구분: 사전에 어떤 주제가 있는지 얼마나 많은 주제가 있는지 모릅니다.

고객들을 취향에 따라 그룹으로 묶기: 부모, 독서광, 게이머 등 어떤 그룹이 있는지 미리 알 수 없고 얼마나 많이 있는지 모릅니다.(평가하기 어렵습니다)

비정상적 웹사이트 접근 탐지: 비정상적인 패턴은 각기 다를 수 있고 가지고 있는 비정상 데이터가 없을 수 있습니다. 현재 트래픽만 관찰할 수 있습니다.

데이터, 특성



좋은 입력 데이터를 만들어 내는 일을 특성 추출 feature extraction, 특성 공학 feature engineering이라고 합니다.

문제와 데이터 이해

머신러닝 프로세스에서 데이터를 이해하고 해결할 문제와 어떤 관련이 있는지 이해하는 것이 가장 중요합니다.

알고리즘마다 잘 들어맞는 데이터나 문제의 종류가 다릅니다.

알고리즘이나 방법론은 문제를 푸는 전체 과정 중 일부일 뿐입니다.

머신러닝 솔루션을 만들 동안 마음에 새겨야 할 질문들

어떤 질문에 대답을 원하는가? 원하는 답을 만들 수 있는 데이터를 가지고 있는가?

머신러닝의 문제로 가장 잘 기술할 수 있는 방법은 무엇인가?

문제를 풀기에 충분한 데이터가 있는가?

추출한 데이터의 특성은 무엇이고 좋은 예측을 위한 특성을 가지고 있는가?

애플리케이션의 성과를 어떻게 측정할 것인가?

다른 연구나 제품과 어떻게 협력할 수 있는가?

Why Python?

아직도 이런 질문을?

파이썬(Python)

과학 분야를 위한 표준 프로그래밍 언어가 되어 가고 있습니다.(vs Julia)

MATLAB, R 같이 도메인에 특화된 언어와 Java, C 같은 범용 언어의 장점을 모두 가지고 있습니다.

통계, 머신러닝, 자연어, 이미지, 시각화 등을 포함한 풍부한 라이브러리가 있습니다: Scikit-Learn, pandas, Numpy, Scipy, matplotlib, ...

브라우저 기반 인터랙티브 프로그래밍 환경인 주피터 노트북 Jupyter Notebook 0| 있습니다.

파이썬 주도 딥러닝 라이브러리: TensorFlow, Keras, PyTorch, Caffe2, CNTK, MXNet, Theano, ...

Scikit-Learn

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

소스 코드를 보고 어떻게 알고리즘이 구현되어 있는지 확인할 수 있습니다.

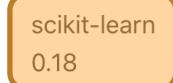
회귀, 분류, 군집, 차원축소, 특성공학, 전처리, 교차검증, 파이프라인 등 머신러닝에 필요한 도구를 두루 갖추고 있습니다.

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

학교, 산업 현장에서 널리 사용됩니다.

폭넓은 커뮤니티와 많은 튜토리얼, 예제 코드가 있습니다.

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 합니다.

대화식 환경을 위해서는 IPython 커널과 Jupyter Notebook 설치가 필요합니다.

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 + 문서 + 결과
(텍스트, 그래프)를 위한 대화식 개발
환경입니다.

탐색적 데이터 분석에 유리하여
많은 과학자, 엔지니어들이
사용합니다.

<https://jupyter.org>



KNeighborsRegressor 분석

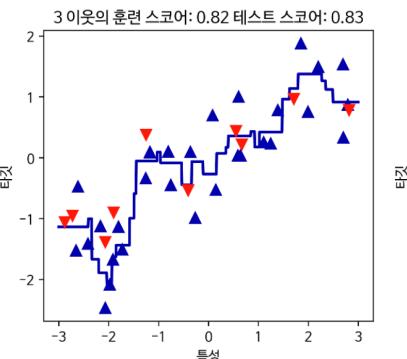
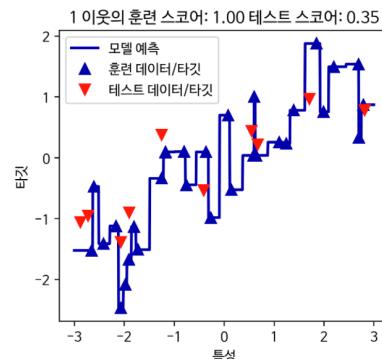
In [25]:

```
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=mlearn.cm2(0), markersize=8)
    ax.plot(X_test, y_test, 'v', c=mlearn.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("점수")
    axes[0].legend(["모델 예측", "훈련 데이터/타깃", "테스트 데이터/타깃"], loc="best")
```

Out[25]:

<matplotlib.legend.Legend at 0x114152b00>



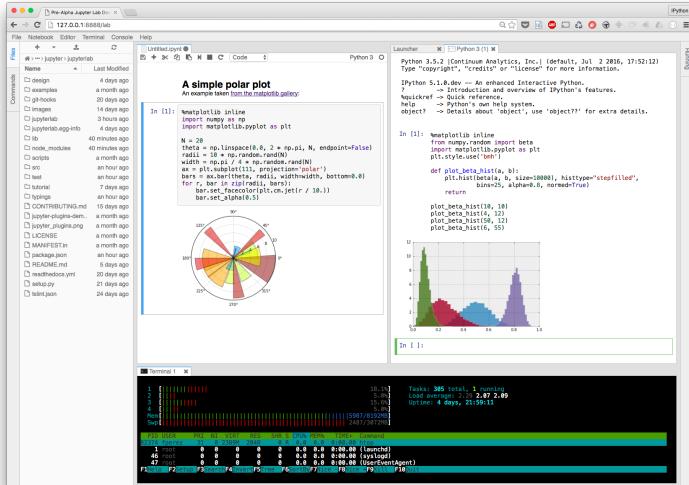
IDE for Jupyter Notebook

PyCharm: 파이썬 IDE로 노트북 파일을 지원합니다.

Rodeo(<https://www.yhat.com/products/rodeo>) : 데이터 분석용 전문 IDE입니다.

JupyterLab(<https://github.com/jupyter/jupyterlab>) : 차세대 주피터 노트북 환경

Colaboratory(<https://colab.research.google.com>) : 구글 드라이브와 연동되는 연구와 교육 목적의 노트북 환경



A screenshot of the PyCharm IDE interface. On the left is a file tree showing a directory structure for a 'jupyter' project. In the center, a Jupyter notebook cell displays Python code for creating a polar plot using matplotlib. The plot shows concentric rings of colored segments. At the bottom, a terminal window shows system logs and task status.

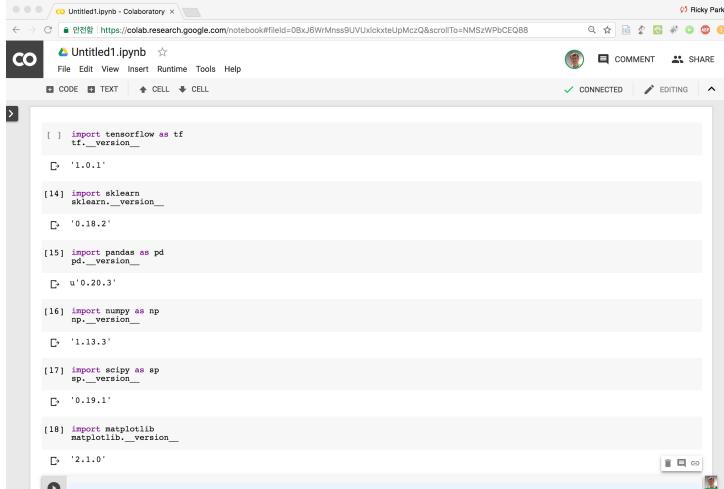
```
In [1]: %matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

N = 20
theta = np.linspace(0, 2 * np.pi, N, endpoint=False)
radii = 10 + np.random.randn(N) * 4
widths = 10 * np.random.rand(N)
ax = plt.subplot(111, projection='polar')
ax.set_theta_offset(np.pi / 4)
ax.set_theta_direction(-1)
bars = ax.bar(theta, radii, widths=widths, bottom=0)
for bar in bars:
    bar.set_facecolor(bar.get_cmap()(bar.theta / 18.0))
    bar.set_alpha(0.5)
```

```
In [1]:
```

```
18:56 Tasks: 385 total, 1 running, 0 pending, 384 idle, 0 stopped
Uptime: 4 days, 21:59:13
[5887/653292]
```

```
1 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 [launched] [exit] [stop] [restartAgent]
```



A screenshot of the Google Colaboratory interface. It shows a Jupyter notebook cell with Python code for plotting histograms of beta distributions. The plots show three distinct bell-shaped curves. The interface includes a sidebar for file management and a top bar for navigation.

```
In [1]: %matplotlib inline
from numpy import random
import tensorflow as tf
import sklearn
import pandas as pd
import numpy as np
```

```
In [1]:
```

```
14:00 Untitled1.ipynb - Colaboratory x
Untitled1.ipynb
File Edit View Insert Runtime Tools Help
CO File Edit View Insert Runtime Tools Help
Untitled1.ipynb
CODE TEXT CELL EDITING
CONNECTED ✓ COMMENT SHARE
```

```
In [1]: %matplotlib inline
from numpy import random
import tensorflow as tf
tf.__version__
plt.figure(figsize=(10, 6))

def plot_beta_hist(b):
    plt.hist(beta.betaprime(b, b, size=10000), histtype='stepfilled',
            bins=100, alpha=0.6, normed=True)
    return bins, alpha, normed
```

```
In [2]: plot_beta_hist(10)
plot_beta_hist(100)
plot_beta_hist(1000)
plot_beta_hist(10000)
plot_beta_hist(100000)
plot_beta_hist(1000000)
```

```
In [3]:
```

```
In [4]: import tensorflow as tf
tf.__version__
'Deep Dive into TensorFlow 2.0'
```

```
In [5]: import sklearn
sklearn.__version__
'0.18.2'
```

```
In [6]: import pandas as pd
pd.__version__
'u'0.20.3'
```

```
In [7]: import numpy as np
np.__version__
'1.13.3'
```

```
In [8]: import scipy as sp
sp.__version__
'0.19.1'
```

```
In [9]: import matplotlib
matplotlib.__version__
'2.1.0'
```

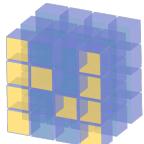
NumPy

다차원 배열을 위한 기능과 선형 대수를 비롯해 고수준 수학 함수와 유사 난수 생성기를 포함하고 있습니다.

scikit-learn의 기본 데이터 구조입니다.(TensorFlow 도)

<http://www.numpy.org>

<http://www.scipy-lectures.org/> 의 1장



NumPy

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

모듈의 별칭을 만듦

ndarray → x:
 [[1 2 3]
 [4 5 6]]

동일한 데이터 타입

SciPy

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

scikit-learn은 알고리즘 구현에 SciPy에 많이 의존하고 있습니다.

0이 많이 포함된 행렬을 효율적으로 표현하기 위한 희소 행렬 sparse matrix
scipy.sparse 패키지를 사용합니다.

<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

밀집 배열로 부터 희소 행렬을 생성
(메모리 부족 가능성)

In [4]: # NumPy 배열을 CSR 포맷의 SciPy 희소 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

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

Coordinate Format

인덱스를 지정하여 희소 행렬을 생성

matplotlib

과학 계산용 그래프 라이브러리입니다.

선, 히스토그램, 산점도 등 다양한
그래프를 그릴 수 있으며 출판 수준의
고품질을 제공합니다.

노트북에 그래프 포함: %matplotlib inline

<https://matplotlib.org/>



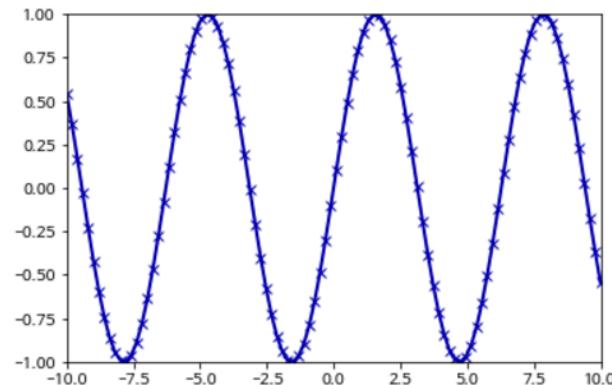
그외 대표적인 그래프 라이브러리: Bokeh, Seaborn,
Plotly

별칭 이름

```
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

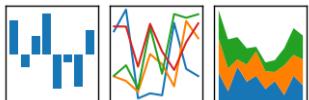
엑셀과 비슷하게 이종 데이터 포함 가능합니다.(numpy와 크게 다른 점)

웨스 매킨니 Wes Makinney의 “파이썬 라이브러리를 활용한 데이터 분석”

<http://pandas.pydata.org>

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



별칭 이름

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

CSV, SQL, 엑셀 등에서
데이터 읽을 수 있음

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

Python 2 vs Python 3

어떤 것을 써야할지 잘 모르겠다면 Python 3이 적절합니다.

이미 파이썬 2로 작성한 코드가 많이 있거나 사용하고 있는 라이브러리가 파이썬 2 밖에 지원하지 않는 경우에는 파이썬 2를 사용합니다.

six 패키지를 사용해 호환된 코드를 만들 수 있습니다. (<https://pythonhosted.org/six/>)

Python 2.7은 2020년까지만 지원합니다.

사용하는 Version

Python 3.6

scikit-learn 0.19.x

matplotlib 2.0.x

NumPy 1.13.x

SciPy 0.19.x

pandas 0.20.x

Iris Dataset

붓꽃의 품종 분류

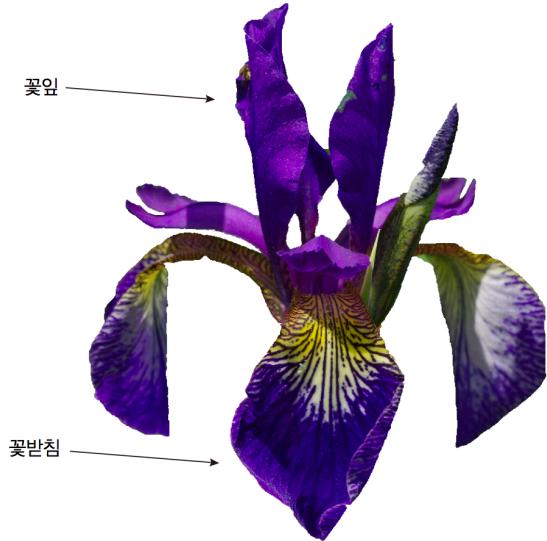
setosa, versicolor, virginica 품종(타깃)을 분류합니다.

특성은 꽃잎 petal, 꽃받침 sepal의 폭과 길이입니다.

사전에 이미 분류한 데이터를 이용하므로 지도 학습 supervised learning이고 3개의 붓꽃 품종에서 고르는 분류 classification 문제입니다.

클래스 class: 가능한 출력값. 즉 세개의 붓꽃 품종

레이블 label: 데이터 포인트 하나에 대한 출력



데이터 적재

붓꽃 데이터: `sklearn.datasets.load_iris()`

그외 `load_digits()`, `load_boston()`, `load_breast_cancer()`, `load_diabetes()` 도 있습니다.

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

딕셔너리와 유사한
scikit-learn의 Bunch 클래스 반환

```
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.DESCRIPTOR[:193] 도 가능

붓꽃 품종의 이름

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

타깃의 이름: ['setosa' 'versicolor' 'virginica']

특성 설명

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]))
```

샘플

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]		

후련 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:

훈련 데이터와 테스트 데이터

훈련에 사용한 데이터는 테스트(일반화)에 사용하지 않습니다: 낙관적인 추정 방지

데이터 분리: 훈련 세트, 테스트 세트(홀드아웃 holdout 세트)

X: 2차원 배열(행렬), 대문자 표기, y: 1차원 배열(벡터), 소문자 표기

```
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)
```

유사 난수 생성

```
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))
```

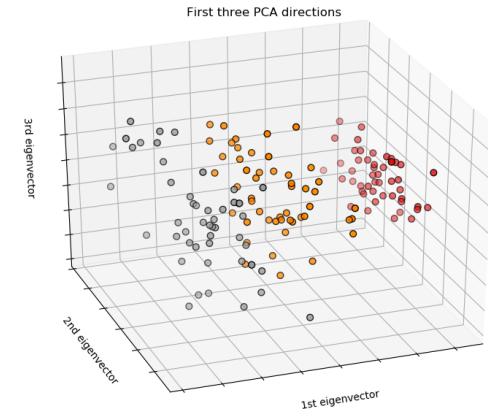
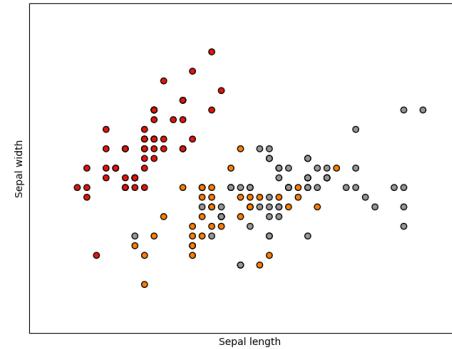
X_test 크기: (38, 4)
y_test 크기: (38,)

기본 3:1 비율 (test_size 매개변수로 변경 가능)

산점도 Scatter Plot

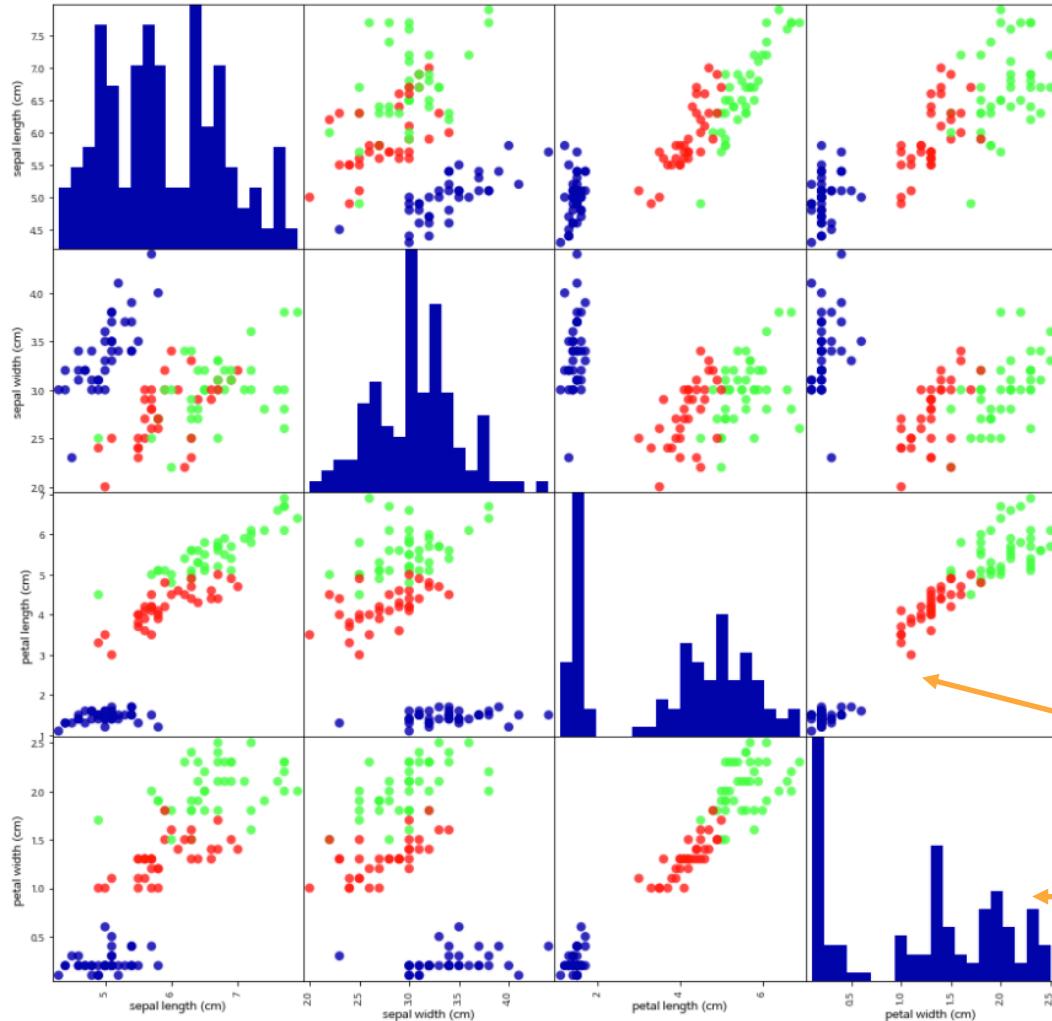
두 개의 특성을 이용 점으로 데이터 표시(3차원 이상은 표현이 어려움)

각 특성의 조합에 대해 모두 그리는 pandas의 산점도 행렬 Scatter Matrix 이용합니다.



pandas 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)
```



특성이 많을 경우 산점도 행렬을
그리기 어렵습니다.

한 그래프에 모든 특성이 나타나는
것이 아니므로 서로 나뉘어진 그래프
사이에 중요한 성질이 있을 수
있습니다.

머신러닝으로 잘 구분할
수 있을 것 같음

자기 자신(주대각선)은
히스토그램으로 표시

First ML Application

k-최근접 이웃 k-Nearest Neighbors, k-NN

이 알고리즘은 훈련 데이터를 저장하는 것이 학습의 전부입니다.

새 데이터 포인트에서 가장 가까운 훈련 데이터 포인트(k개)를 찾아서 가장 높은 빈도의 클래스를 예측으로 사용합니다.

대표적인 인스턴스 기반 instance based 학습 알고리즘(vs model based)

```
In [25]: from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier(n_neighbors=1) 이웃의 개수: 기본값 5
```

```
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')
```

fit() 메소드에서도
knn 객체 반환

훈련 데이터 주입

예측

학습된 모델(이전 슬라이드의 knn 객체)을 사용해 새로운 붓꽃의 품종을 분류합니다.

가상으로 새로운 데이터 생성

```
In [27]: X_new = np.array([[5, 2.9, 1, 0.2]])
print("X_new.shape: {}".format(X_new.shape))

X_new.shape: (1, 4)
```

sepal length, width
petal length, width

특성은 항상 2차원 배열

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

숫자로 된 레이블이
반환됩니다.

```
In [28]: prediction = knn.predict(X_new)
print("예측: {}".format(prediction))
print("예측한 타깃의 이름: {}".format(
    iris_dataset['target_names'][prediction]))
```

예측: [0]

예측한 타깃의 이름: ['setosa']

예측

실제로 이 값이 맞을까요?

평가

모델의 성능(정확도)을 평가하기 위해 훈련에 사용하지 않았던 테스트 세트를 활용합니다.

테스트 세트에 대한 예측(타깃이 없습니다)

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

예측 값을 실제 레이블과 비교합니다. 0 또는 1의 배열

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]

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

평균

테스트 세트의 정확도: 0.97

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

새로운 데이터에 대해 97% 정확도를 기대할 수 있습니다.

테스트 세트의 정확도: 0.97

요약

붓꽃 어플리케이션은 분류해 놓은 품종 데이터를 사용한 지도 학습입니다.

세개의 품종(클래스)을 구분하는 분류^{classification} 문제(다중 분류)입니다.

특성 데이터를 담고 있는 **X**(2차원)와 기대 출력(레이블)을 가진 **y**(1차원)를 NumPy 배열로 다루었습니다.

훈련 세트에서 학습(훈련)하고 테스트 세트로 만들어진 모델을 평가했습니다.

```
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)))  
  
predict(X_new)
```

테스트 세트의 정확도: 0.97