Machine Learning & Scikit-Learn

#1

Scikit-learn 소개

Scikit-learn

- 탄생: 2007년 구글
- 특징:
  - 파이썬으로 구현된 라이브러리 중에서 머신 러닝 교육 및 실무용으로 가장 많이 사용되고 있는 open source library
  - API가 일관되고 간결함
  - 다른 라이브러리와 호환성이 좋음
- Requirements (Scikit-learn v.0.21.2)
  - Python(>=3.5)
  - Numpy(>=1.11.0)
  - SciPy(>= 0.17.0)
  - Matplotlib (>=1.5.1)
  - Pandas (>=0.18.0)

Scikit-learn의 설치

- Anaconda 프롬프트
  - Install

```
conda install scikit-learn
```

  - Update

```
conda update scikit-learn
```

  - Uninstall

```
conda remove scikit-learn
```

- Scikit-learn에 기본으로 있는 dataset
  - 붓꽃(Iris)
  - 필기체 숫자(Digits)
  - Boston house price

설치 확인

- 데이터세트 로딩

Iris(붓꽃)

In :
```
from sklearn.datasets import load_iris
```

In :
```
load_iris??
```

```
Signature: load_iris(return_X_y=False)
Source:
def load_iris(return_X_y=False):
    """Load and return the iris dataset (classification).

    The iris dataset is a classic and very easy multi-class classification
    dataset.

    =================   ==============
    Classes                          3
    Samples per class               50
    Samples total                  150
    Dimensionality                   4
    Features            real, positive
    =================   ==============
```

```
if return_X_y:
    return data, target

return Bunch(data=data, target=target,
             target_names=target_names,
             DESCR=fdescr,
             feature_names=['sepal length (cm)', 'sepal width (cm)',
                            'petal length (cm)', 'petal width (cm)'],
             filename=iris_csv_filename)
```

설치 확인

- 데이터세트 로딩

Iris(붓꽃)

```
In :  from sklear.datasets import load_iris
      data = load_iris()
```

```
In :  data
```

```
In :  type(data)
```

```
In :  dir(data)
```

```
Out[31]: ['DESCR', 'data', 'feature_names', 'filename', 'target', 'target_names']
```

```
In :  print(data.feature_names)
```

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

```
In :  form Ipython.display import HTML
```

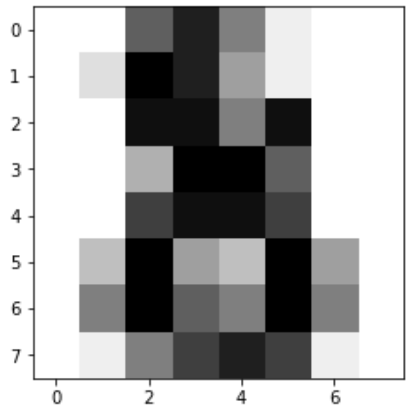
```
In :  HTML(data.DESCR.replace('\n', '<br/>'))
```

설치 확인

- 데이터세트 로딩

  Digits(필기체 숫자)

In :	`from sklearn import datasets` `digits = datasets.load_digits()`
In :	`digits.images.shape`
In :	`digits.target`
In :	`import matplotlib.pyplot as plt`
In :	`plt.imshow(digits.images[-1], cmap=plt.cm.gray_r)`



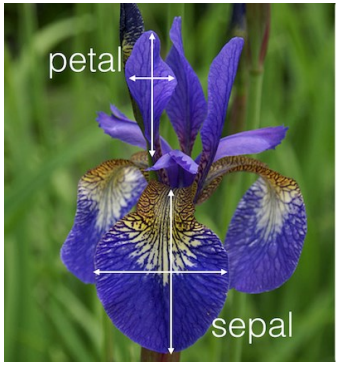
Scikit-learn에서의 데이터

- Bunch 클래스
  - 속성
    - ✓ data (필수): 독립변수 ndarray 배열
    - ✓ target (필수): 종속변수 ndarray 배열
    - ✓ feature_names (옵션): 독립 변수 이름 리스트
    - ✓ target_names (옵션): 종속 변수 이름 리스트
    - ✓ DESCR (옵션): 자료에 대한 설명

Seaborn에서 데이터 가져오기

```
import seaborn as sns
iris = sns.load_dataset('iris')
iris.head()
```

	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



Iris Versicolor | Iris Setosa | Iris Virginica

외부에서 데이터 가져오기

- 데이터 다운로드

Housing data

In :

```
import os
import tarfile
from six.moves import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-
ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL,
housing_path=HOUSING_PATH):
    if not os.path.isdir(housing_path):
        os.makedirs(housing_path)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
```

외부에서 데이터 가져오기

- 데이터 다운로드

  Housing data(계속)

In :
```
fetch_housing_data()
```

In :
```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

In :
```
housing = load_housing_data()
housing.head()
```

In :
```
housing.info()
```

In :
```
housing["ocean_proximity"].value_counts()
```

In :
```
housing.describe()
```

외부에서 데이터 가져오기

- 데이터 확인 및 저장

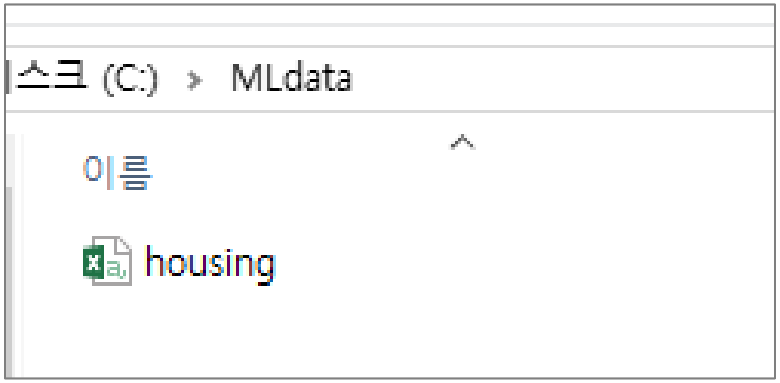
Housing data(계속)

In :

```
%matplotlib inline
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```

In :

```
import pandas as pd
housing.to_csv('c:/MLdata/housing.csv')
```



외부에서 데이터 가져오기

- 데이터 확인

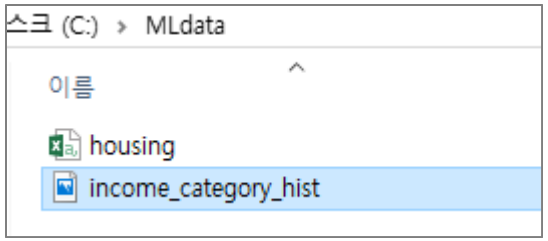
Housing data(계속)

In :
```
housing["median_income"].hist()
```

In :
```
# 소득 카테고리 개수를 제한하기 위해 1.5로 나눕니다.
housing["income_cat"] = np.ceil(housing["median_income"] / 1.5)
# 5 이상은 5로 레이블합니다.
housing["income_cat"].where(housing["income_cat"] < 5, 5.0,
inplace=True)
```

In :
```
housing["income_cat"].value_counts()
```

In :
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize =(5,3))
housing["income_cat"].hist()
fig.savefig('c:\MLdata\income_category_hist')
```



기계학습을 위한 데이터 저장소

- **Kaggle** 예측모델로 경쟁하는 플랫폼의 선두자 (https://www.kaggle.com/competitions)
- **UCI MLR** UC Irvine 기계 학습 데이터 저장소 (http://archive.ics.uci.edu/ml/index.php)
- **google.com/publicdata** - 구글이 제공하는 공개된 데이터 저장소 (http://www.google.com/publicdata/directory)
- **Freebase** 커뮤니티가 만든 유명한 사람, 장소, 물건들에 대한 데이터베이스 (https://developers.google.com/freebase/)
- **mldata.org** 업로드 검색이 가능한 기계 학습 데이터 저장소 (http://mldata.org/)
- **Infochimps** 거대한 양의 수집된 대용량 데이터 저장소 (http://www.infochimps.com/datasets)
- **Amazon Web Services** 아마존 웹 서비스가 제공하는 공개된 데이터 저장소 (https://aws.amazon.com/ko/datasets/)

#2

데이터 전처리

Data preprocessing?

- 데이터 전처리
  - 현실에서 가져오는 raw 데이터의 품질을 보완하는 작업들
- 데이터전처리의 주요 작업
  - 데이터 정제(Cleaning, Cleansing)
    - ✓ 결측값(missing value)을 채우거나, 잡음값(noisy data)을 평활화(smoothing), 이상치(outlier) 발견하여 제거, 불일치 해결
  - 데이터 통합(Integration)
    - ✓ 다수의 소스에서 얻은 데이터를 합쳐서 표현
  - 축소(Reduction)
    - ✓ 크기는 작지만 분석 결과는 동일한 데이터로 표현
  - 변환(Transformation)
    - ✓ 기계학습 알고리즘의 효율성을 극대화하기 위한 변형

Scikit-learn을 이용한 전처리

- 표준화(standardization = mean removal and variance scaling)
  - 자료에 선형 변환을 적용하여 전체 자료를 평균 0, 분산 1이 되도록 만드는 과정
  - 자료의 overflow나 underflow를 방지하기 위함
  - 제공되는 함수와 클래스

    scale(X): 표준정규분포 스케일

    robust_scale(X): meadian 사용. Outlier의 영향 최소화

    minmax_scale(X): 최대/최소값 사용

    maxabs_scale(X): 최대 절대값 사용

Scikit-learn을 이용한 전처리

- 표준화(standardization = mean removal and variance scaling)

```
from sklearn.preprocessing import scale, robust_scale, minmax_scale, maxabs_scale
```

```
import numpy as np
x = (np.arange(10, dtype=np.float) - 3).reshape(-1, 1)
```

```
x
```

```
array([[-3.],
       [-2.],
       [-1.],
       [ 0.],
       [ 1.],
       [ 2.],
       [ 3.],
       [ 4.],
       [ 5.],
       [ 6.]])
```

```
import pandas as pd
df = pd.DataFrame(np.hstack([x, scale(x), robust_scale(x), minmax_scale(x), maxabs_scale(x)]),
columns=["x", "scale(x)", "robust_scale(x)", "minmax_scale(x)", "maxabs_scale(x)"])
```

Scikit-learn을 이용한 전처리

- 표준화(standardization = mean removal and variance scaling)

```
from sklearn.datasets import load_iris
iris = load_iris()
data1 = iris.data
data2 = scale(iris.data)
```

```
print("old mean:", np.mean(data1, axis=0))
print("old std: ", np.std(data1, axis=0))
print("new mean:", np.mean(data2, axis=0))
print("new std: ", np.std(data2, axis=0))
```

```
old mean: [5.84333333 3.05733333 3.758      1.19933333]
old std:  [0.82530129 0.43441097 1.75940407 0.75969263]
new mean: [-1.69031455e-15 -1.84297022e-15 -1.69864123e-15 -1.40924309e-15]
new std:  [1. 1. 1. 1.]
```

Scikit-learn을 이용한 전처리

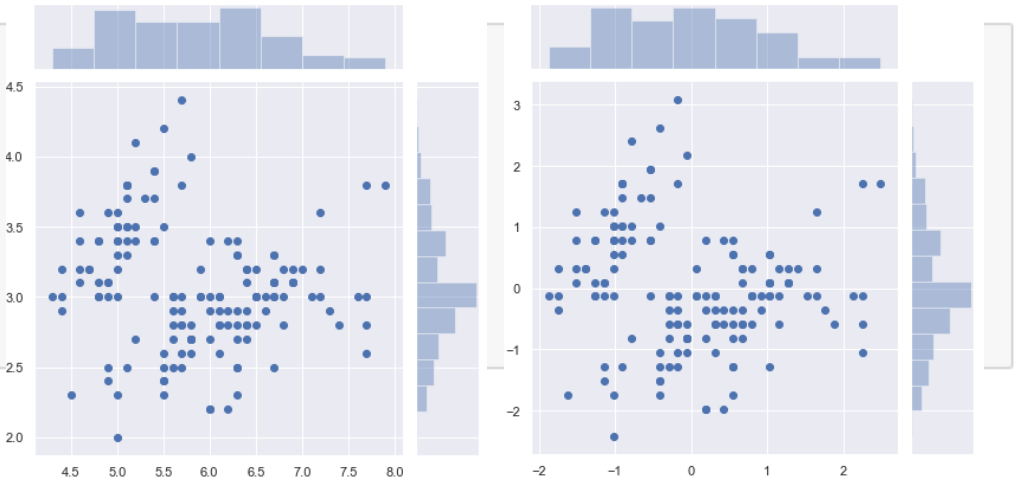
- 표준화(standardization = mean removal and variance scaling)

```
from sklearn.datasets import load_iris
iris = load_iris()
data1 = iris.data
data2 = scale(iris.data)
```

```
print("old mean:", np.mean(data1, axis=0))
print("old std: ", np.std(data1, axis=0))
print("new mean:", np.mean(data2, axis=0))
print("new std: ", np.std(data2, axis=0))
```

```
old mean: [5.84333333 3.05733333 3.758      1.19933333]
old std:  [0.82530129 0.43441097 1.75940407 0.75969263]
new mean: [-1.69031455e-15 -1.84297022e-15 -1.69864123e-15 -1.40924309e-15]
new std:  [1. 1. 1. 1.]
```

```
import seaborn as sns
sns.set();
sns.jointplot(data1[:,0], data1[:,1])
plt.show()
sns.jointplot(data2[:,0], data2[:,1])
plt.show()
```



Scikit-learn을 이용한 전처리

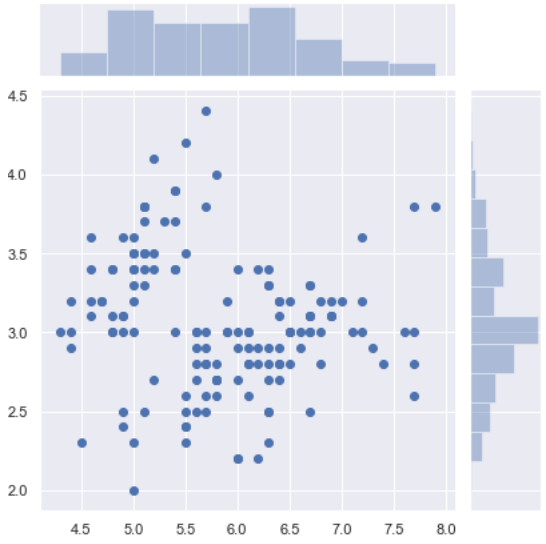
- 표준화(standardization = mean removal and variance scaling)

```
from sklearn.datasets import load_iris
iris = load_iris()
data1 = iris.data
```

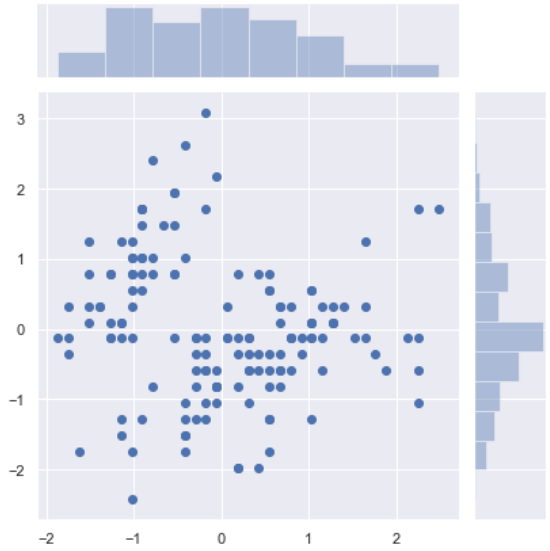
```
data4 = minmax_scale(data1)
```

```
sns.jointplot(data1[:,0], data1[:,1])
plt.show()
sns.jointplot(data4[:,0], data4[:,1])
plt.show()
```

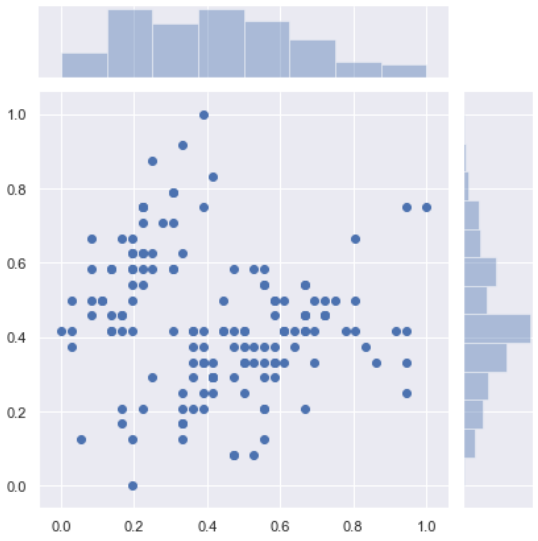
original



scale



minmax_scale



Scikit-learn을 이용한 전처리

- 정규화(normalization = scaling and centering)
  - 개별 데이터의 크기를 모두 같게 만들기 위한 변환
  - 다차원 독립변수 벡터가 있을 때, 각 벡터 원소들의 상대적인 크기만 중요한 경우에 사용
  - 관련 함수

    normalize(X)

Scikit-learn을 이용한 전처리

- 정규화(normalization = scaling and centering)

```
from sklearn.preprocessing import scale, normalize
y1 = scale(x)
y2 = normalize(x)
```

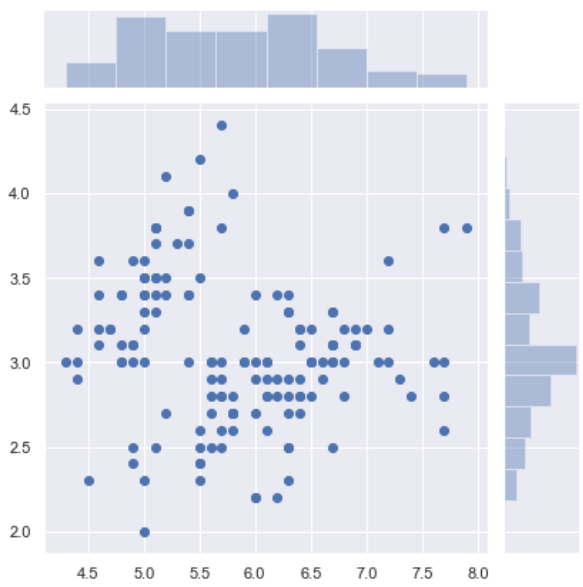
```
print("original x:\n", x)
print("scale:\n", y1)
print("norms (scale)\n", np.linalg.norm(y1, axis=1))
print("normlize:\n", y2)
print("norms (normalize)\n", np.linalg.norm(y2, axis=1))
```

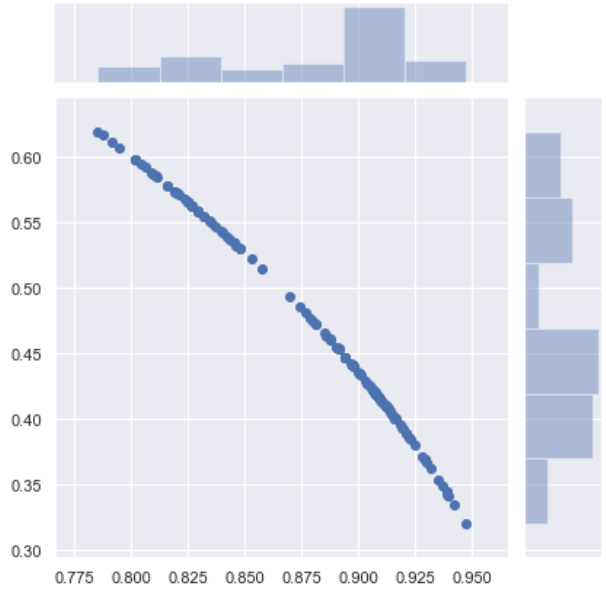
```
original x:
 [[-20.  -2.]
 [-19.  -1.]
 [-18.   0.]
 [-17.   1.]
 [-16.   2.]]
scale:
 [[-1.41421356 -1.41421356]
 [-0.70710678 -0.70710678]
 [ 0.          0.        ]
 [ 0.70710678  0.70710678]
 [ 1.41421356  1.41421356]]
norms (scale)
 [2. 1. 0. 1. 2.]
normlize:
 [[-0.99503719 -0.09950372]
 [-0.99861783 -0.05255883]
 [-1.          0.        ]
 [-0.99827437  0.05872202]
 [-0.99227788  0.12403473]]
norms (normalize)
 [1. 1. 1. 1. 1.]
```

Scikit-learn을 이용한 전처리

- 정규화(normalization = scaling and centering)

```
from sklearn.datasets import load_iris
iris = load_iris()
data1 = iris.data[:,:2]
data3 = normalize(data1)
sns.jointplot(data1[:,0], data1[:,1])
plt.show()
sns.jointplot(data3[:,0], data3[:,1])
plt.show()
```





Scikit-learn을 이용한 전처리

- 사용자 정의 변환

```
from sklearn.preprocessing import FunctionTransformer
def all_but_first_column(X):
    return X[:, :1]
X = np.arange(12).reshape(4, 3)
print(X)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
Y = FunctionTransformer(all_but_first_column).fit_transform(X)
print(Y)
```

```
[[0]
 [3]
 [6]
 [9]]
```

Scikit-learn을 이용한 전처리

- 사용자 정의 변환

```
from sklearn.preprocessing import FunctionTransformer
def all_but_first_column(X):
    return X*0.5
X = np.arange(12).reshape(4, 3)
print(X)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
Y = FunctionTransformer(all_but_first_column).fit_transform(X)
print(Y)
```

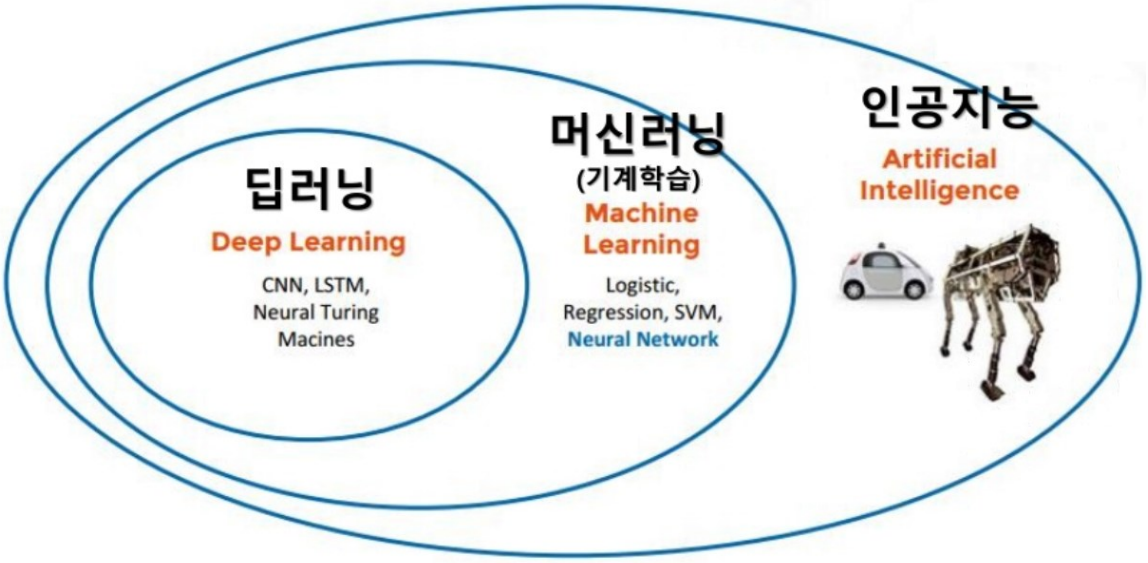
```
[[0.  0.5 1. ]
 [1.5 2.  2.5]
 [3.  3.5 4. ]
 [4.5 5.  5.5]]
```

#3. Machine Learning

기계 학습 개요

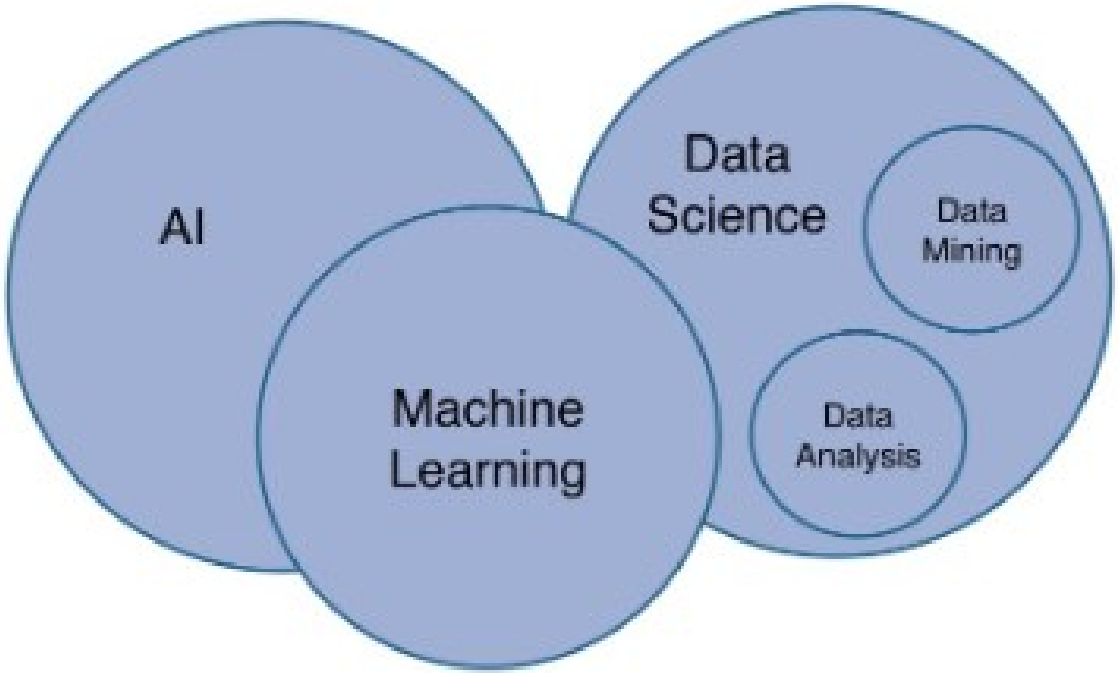
인공지능(Artificial Intelligence)

- 인공지능이란?
  - 기계가 학습과 추리 등의 인간의 지능과 비슷한 작업을 수행하는 것
- 기계학습
  - 데이터에 대한 수학적 모델 기반 인공지능 기법
- 딥러닝
  - Aritificial Neural Network 기반 인공지능 기법



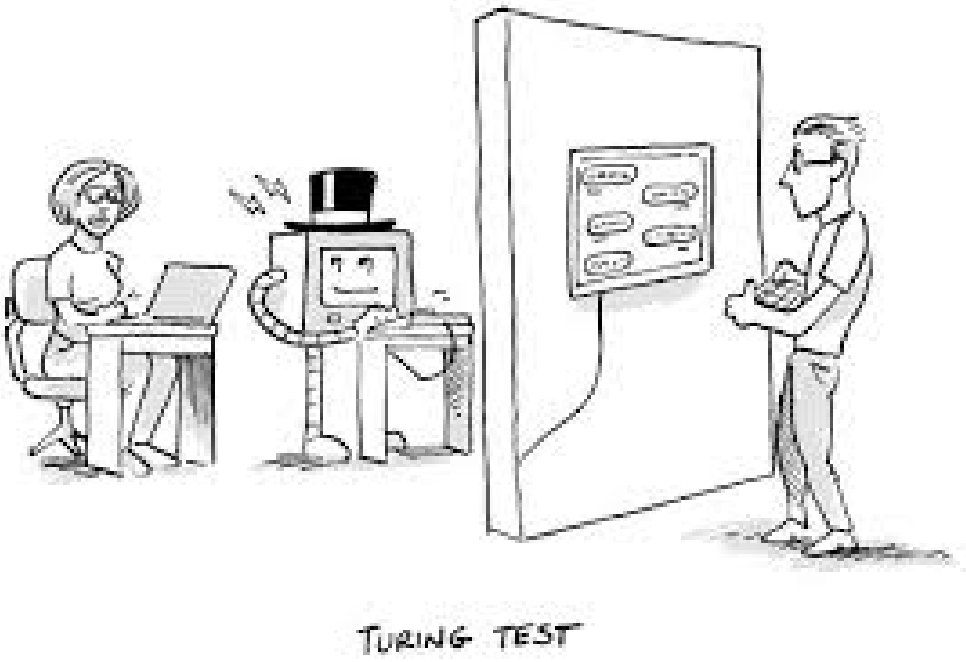
인공지능

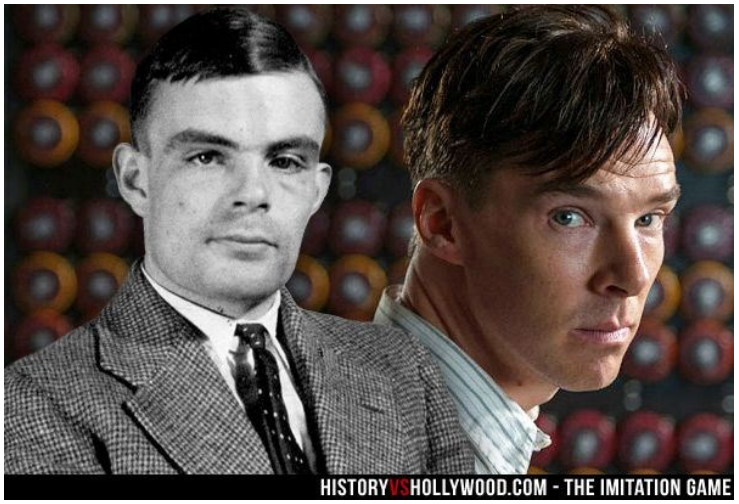
- 인공지능, 기계학습, 데이터 과학



인공지능의 정확성 테스트

- Imitation Game = Turing Test





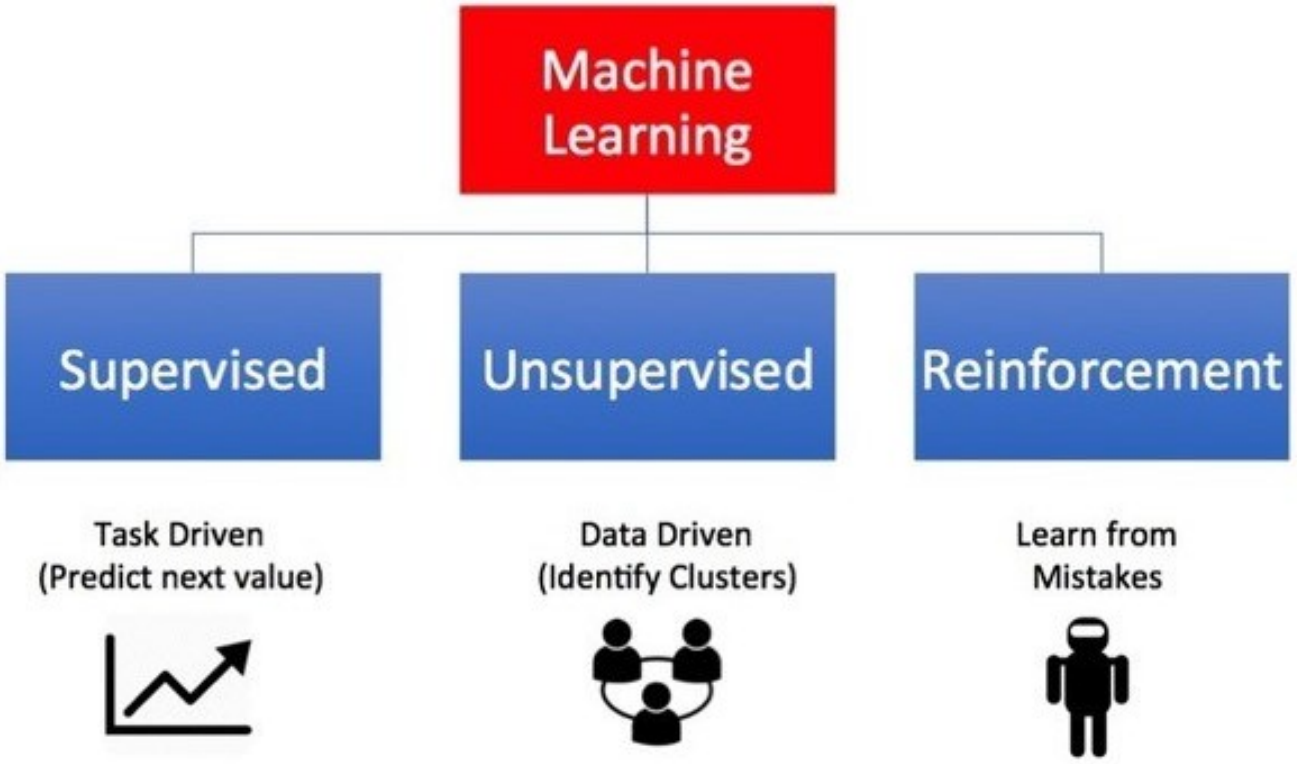
인공지능(Artificial Intelligence)의 분류

약한 인공지능 ← → 강한 인공지능

인간처럼 생각하는 시스템 • 마음뿐 아니라, 인간과 유사한 사고 및 의사결정을 내리는 시스템 • 인지 모델링 접근 방식	**합리적으로 생각하는 시스템** • 계산 모델을 통해 지각, 추론, 행동 같은 정신적 능력을 갖춘 시스템 • 사고의 법칙 접근 방식
인간처럼 행동하는 시스템 • 인간의 지능을 필요로 하는 어떤 행동을 기계가 따라하는 시스템 • 튜링 테스트 접근 방식 • 보고(Computer vision), 듣고(Voice Recognition), 움직이고(Movement), 운전하는(Driving) 행동	**합리적으로 행동하는 시스템** • 계산 모델을 통해 지능적 행동을 하는 에이전트 시스템 • 합리적인 에이전트 접근 방식

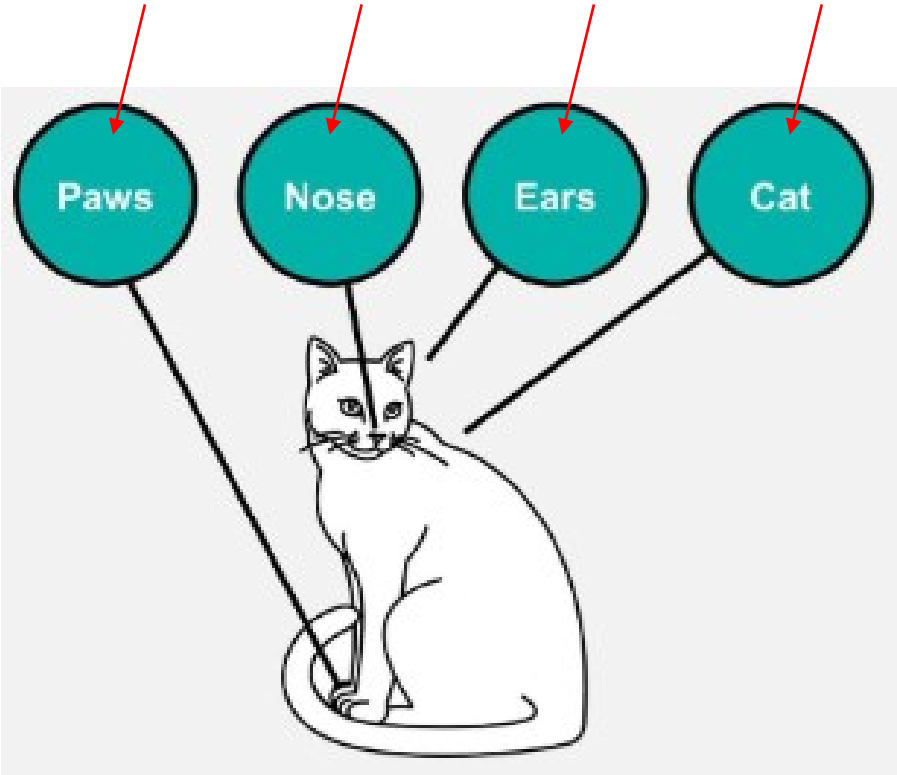
기계 학습의 범주

- 지도학습(Supervised Learning)
- 비지도학습(Unsupervised Learning)
- 강화학습(Reinforment Learning)



인공 지능 용어

- 레이블(label)

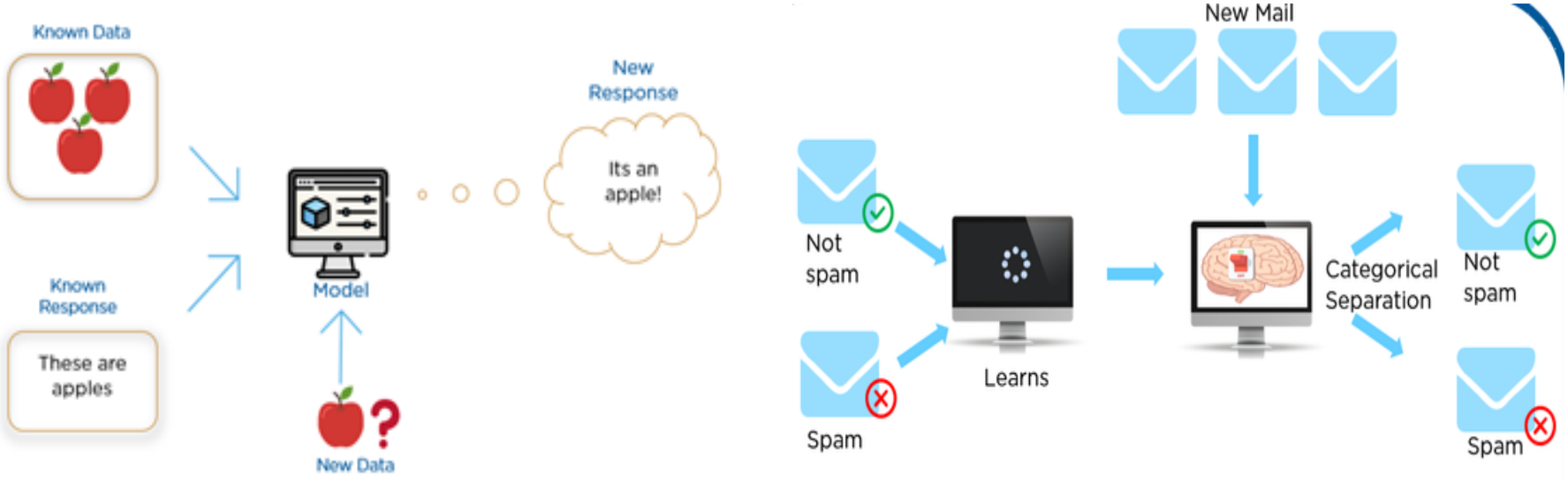


인공지능 용어

- 학습 데이터(Training set)
  - 기계 학습의 모델을 검증하며 학습(파라메터 조정)에 사용되는 데이터 세트
- 테스트 데이터(testing set, recall data set)
  - 이미 학습된 모델에 적용할 자료
- 성능평가
  - 데이터 집합을 얼마나 정확하게 예측했는 지를 알려주는 것

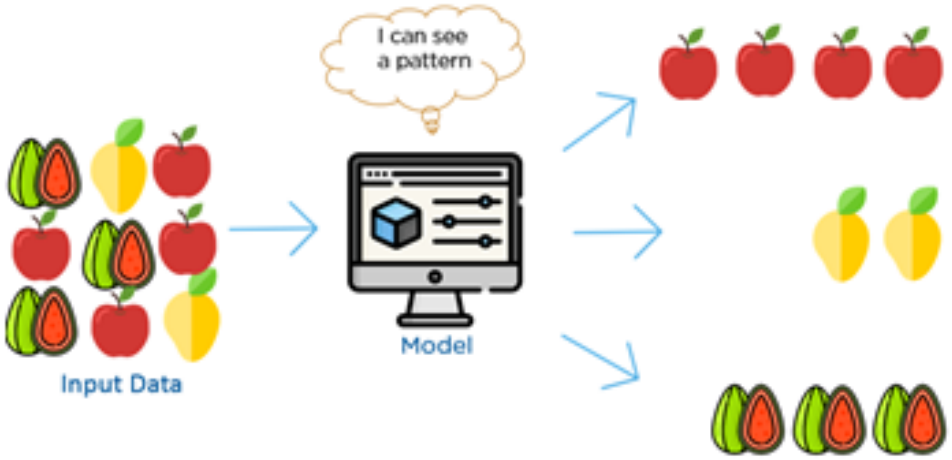
기계 학습의 범주: 지도학습

- 지도학습(Supervised Learning)
  - (특징, 결과)로 주어진 훈련 데이터로 학습함
  - 특징(feature)과 결과(label, 레이블) 사이의 관계를 모델링
  - 새 데이터에 대해 학습된 레이블로 결과로 냄
  - 분야: 분류, 회귀



기계 학습의 범주: 비지도학습

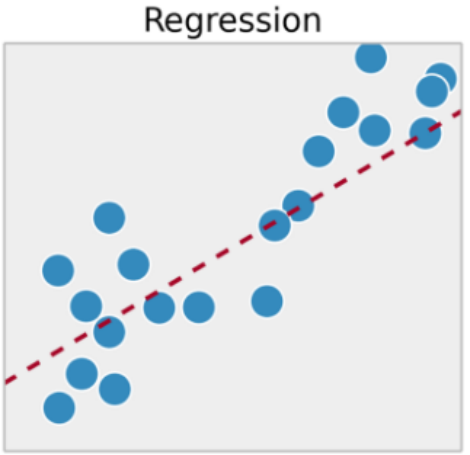
- 비지도학습(Unsupervised Learning)
  - (특징)만 주어짐
  - 데이터의 특징을 모델링('데이터 스스로가 말하게 하는 것')
  - 새로 주어진 데이터에 대해 기존 자료 중 비슷한 군에 배치
  - 분야: 군집화, 차원축소



기계 학습의 적용 분야

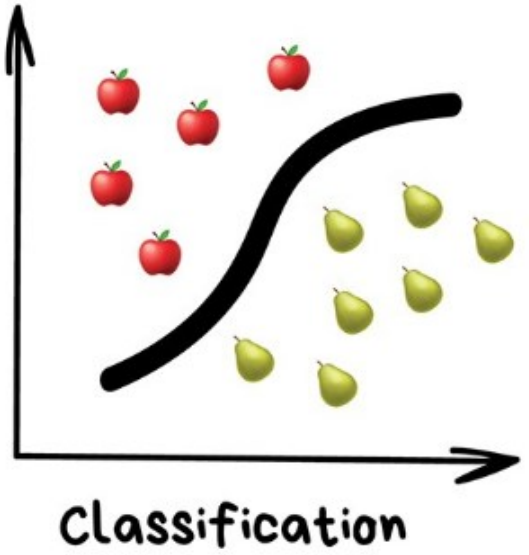
- 회귀(Regression)
  - 학습 자료에 대한 근사식을 구해

    새로운 자료에 대한 레이블을 예측하는 분야
  - 오늘날의 활용 분야
    - ✓ 주식 가격 예측
    - ✓ 주문 및 판매량 분석
    - ✓ 의료 진단
    - ✓ 어떤 값과 시간과의 상관관계 분석
  - 방법론
    - ✓ Linear regression, Logistic regression, Polynomial regression, etc.



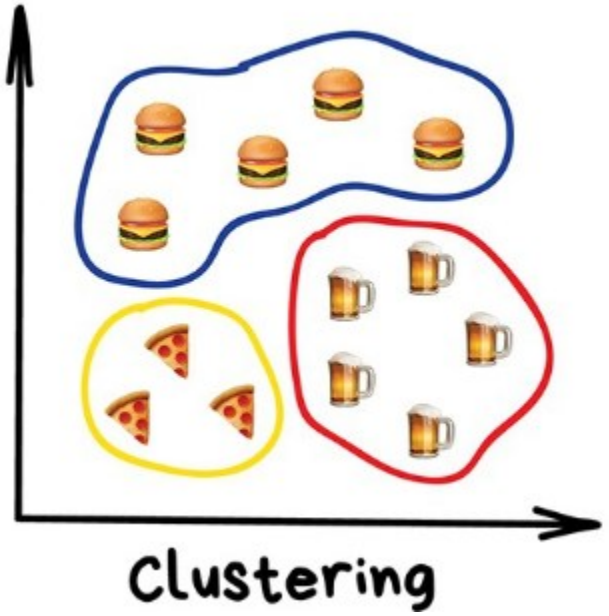
기계 학습의 적용 분야

- 분류(Classification)
  - 둘 이상의 이산적인 범주로 레이블을 예측하는 분야
  - 오늘날의 활용 분야
    - ✓ 스팸 필터링
    - ✓ 언어 분류
    - ✓ 문서 유사도 분석
    - ✓ 필기체 문자 인식
    - ✓ 사기 판단(fraud detection)
  - 방법론
    - ✓ Naive Bayes, Decision Tree, Logistic Regression, K-Nearest Neighbors, SVM(Support Vector Machine)



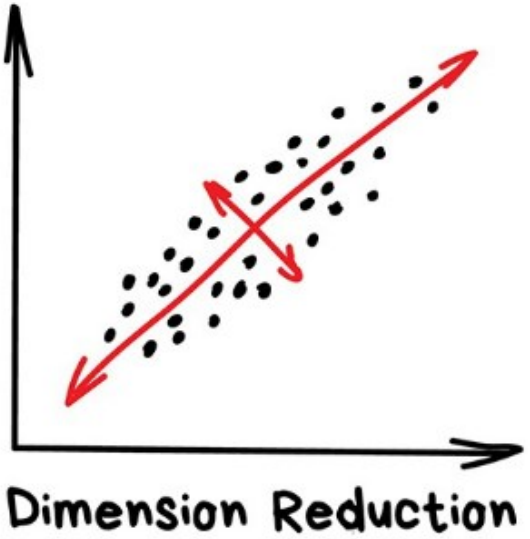
기계 학습의 적용 분야

- 군집화(Clustering
  - 레이블이 없는 데이터에 대한 레이블을 추론하는 분야
  - 오늘날의 활용 분야
    - ✓ 시장 분할
    - ✓ 지도에서 가까운 지점을 병합
    - ✓ 영상 압축
    - ✓ 자료에 새로 레이블 부여
    - ✓ 이상행동 감지
  - 방법론
    - ✓ K-means clustering, DBSCAN, etc

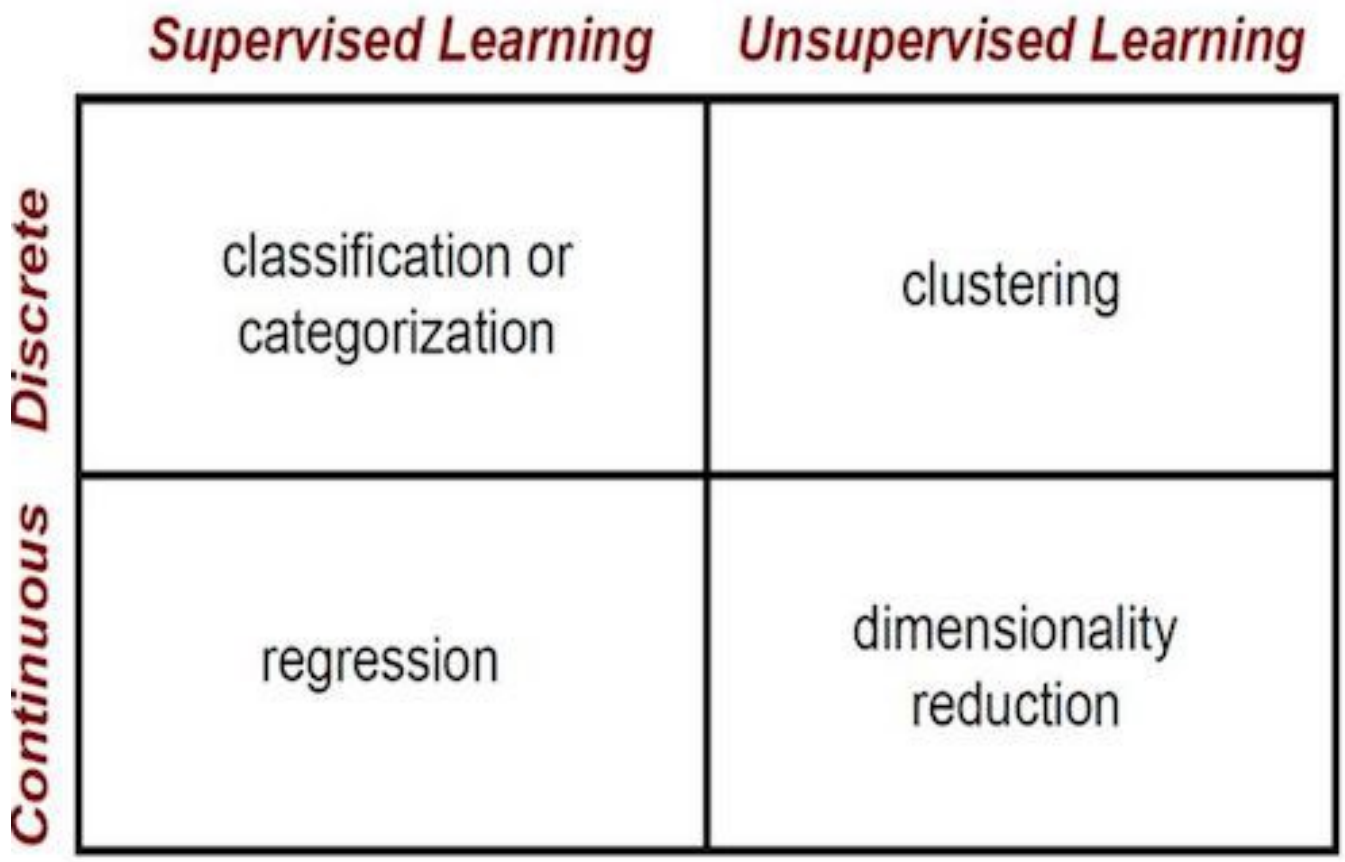


기계 학습의 적용 분야

- 차원 축소
  - 특별한 특징을 조합하여 한 차원 높은 것으로
  - 오늘날의 활용 분야
    - ✓ 추천 시스템
    - ✓ Topic modeling
    - ✓ 유사 문서 검색
    - ✓ 가짜 이미지 분석
    - ✓ 위험 관리
  - 방법론
    - ✓ PCA(Principal Component Analysis),LSA(Latent Semantic Analysis)

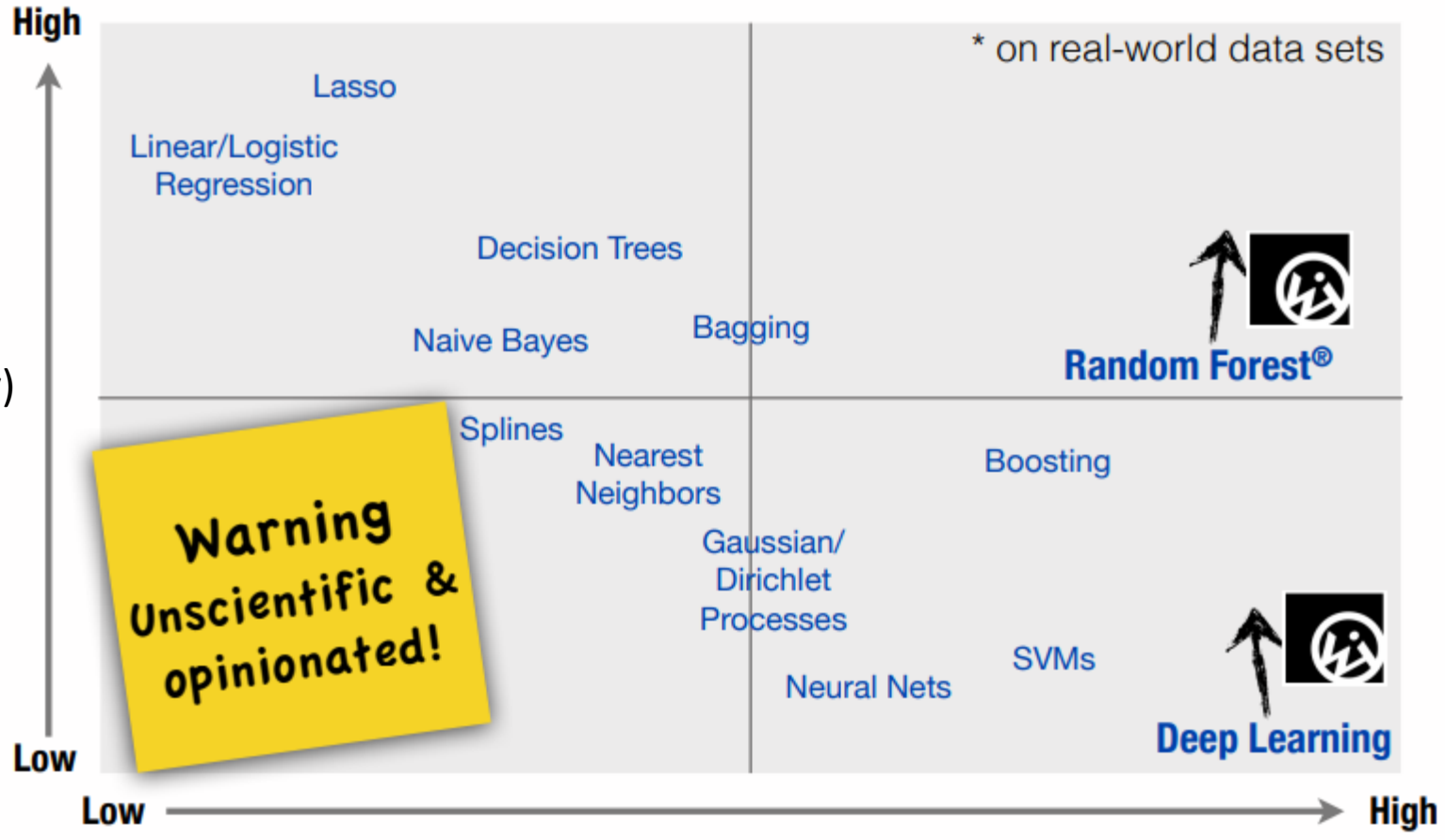


기계 학습의 적용 분야


Supervised Learning
Unsupervised Learning
Discrete
Continuous
classification or categorization
clustering
regression
dimensionality reduction

기계 학습의 방법들

과학적인
해석
가능도
(Interpreterability)



정확도(Accuracy)

#4

통계와 기계 학습

통계와 기계학습

- 머신러닝과 통계학과의 관계
  - 데이터를 수집하고 분석해 컴퓨터에게 학습 시키는 과정
  - 새로운 입력값에 대해 결과를 예측하는 과정에 확률 이론을 활용

- 기계학습에서 통계적인 방법
  - 상관관계분석
  - 회귀분석
  - 확률분석

통계모델:[1] 상관분석

- 상관분석(correlation analysis)
  - 두 변수간의 관계의 강도(얼마나 밀접하게 유지되어 있는 지)를 분석하는 것
- 상관계수(correlation coefficient) r
  - 상관계수 r의 범위 : $-1 \le r \le 1$
  - Pearson's correlation을 주로 사용
    - ✓ Pandas의 corr()함수
    - ✓ 수식
      - 측정값(x, y)에 대하여 n개의 측정값 $(x_1, y_1),(x_2,y_2),\ldots(x_n, y_n)$ 이 주어졌을 때

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)S_xS_y}$$

$\bar{x}, \bar{y}$: 각각 x, y의 평균 값
S_x , S_y : 각각 x, y의 표준편차

통계모델:[1] 상관분석

- 상관계수(correlation coefficient) r 구하기
  - ✓ 데이터 샘플 1

```
# generate related variables
from numpy import mean
from numpy import std
from numpy.random import randn
from numpy.random import seed
from matplotlib import pyplot
# seed random number generator
seed(1)
# prepare data
data1 = 20 * randn(1000) + 100
data2 = data1 + (10 * randn(1000) + 50)
# summarize
print('data1: mean=%.3f stdv=%.3f' % (mean(data1), std(data1)))
print('data2: mean=%.3f stdv=%.3f' % (mean(data2), std(data2)))
# plot
pyplot.scatter(data1, data2)
```

통계모델:[1] 상관분석

- 상관계수(correlation coefficient) r 구하기
  - 수식으로 구하기

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)S_xS_y}$$

$\bar{x}, \bar{y}$: 각각 x, y의 평균 값
S_x , S_y : 각각 x, y의 표준편차

```
mean1 =  mean(data1)
std1 = std(data1)
mean2 = mean(data2)
std2 = std(data2)
Qx = data1 - mean1
Qy = data2 - mean2
Q = Qx * Qy
```

```
Q.shape
print(len(Q))
```

```
1000
```

```
r = sum(Q)/((len(Q)-1)*std1*std2)
print('correlation coefficient:', r)
```

```
correlation coefficient: 0.8885004089885196
```

통계모델:[1] 상관분석

- 상관계수(correlation coefficient) r 구하기
  - Pandas 활용

```
import pandas as pd
rows = [int(i) for i in range(len(data1))]
s_d1 = pd.Series(data1, index=rows)
s_d2 = pd.Series(data2, index=rows)
clm = ['data1', 'data2']
d = pd.DataFrame(columns=clm)
d['data1'] = s_d1
d['data2'] = s_d2
d.corr()
```

	data1	data2
data1	1.000000	0.887612
data2	0.887612	1.000000

통계모델:[1] 상관분석

- 상관계수(correlation coefficient) r 구하기

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)S_xS_y}$$

$\bar{x}, \bar{y}$: 각각 x, y의 평균 값
S_x , S_y : 각각 x, y의 표준편차

  - Scipy 의 pearson 상관 계수

```
from scipy.stats import pearsonr
# calculate Pearson's correlation
corr1, _ = pearsonr(data1, data2)
print('Pearsons correlation between data1 and data2: %.3f' % corr1)
```

```
Pearsons correlation between data1 and data2: 0.888
```

통계모델:[1] 상관분석

- 상관계수(correlation coefficient) r 구하기
  - ✓ 데이터 샘플 2

```
seed(1)
# prepare data
data1 = 20 * randn(1000) + 100
data2 = -data1 + (10 * randn(1000) + 50)
# summarize
print('data1: mean=%.3f stdv=%.3f' % (mean(data1), std(data1)))
print('data2: mean=%.3f stdv=%.3f' % (mean(data2), std(data2)))
# plot
pyplot.scatter(data1, data2)
```

통계모델:[1] 상관분석

- 상관계수(correlation coefficient) r 구하기
  - 수식대로 구하기

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)S_xS_y}$$

$\bar{x}, \bar{y}$: 각각 x, y의 평균 값
S_x , S_y : 각각 x, y의 표준편차

```
mean1 =  mean(data1)
std1 = std(data1)
mean2 = mean(data2)
std2 = std(data2)
Qx = data1 - mean1
Qy = data2 - mean2
Q = Qx * Qy
```

```
r = sum(Q)/((len(Q)-1)*std1*std2)
print('correlation coefficient:', r)
```

```
correlation coefficient: -0.8841064512766241
```

통계모델:[1] 상관분석

- 상관계수 r 구하기
  - ✓ 데이터 샘플: Iris

```
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
data.shape
# data[:,0] : sepal length in cm\n
# data[:,1] : sepal width in cm\n
# data[:,2]:  petal length in cm\n
# data[:,3]: petal width in cm\n
```

```
(150, 4)
```

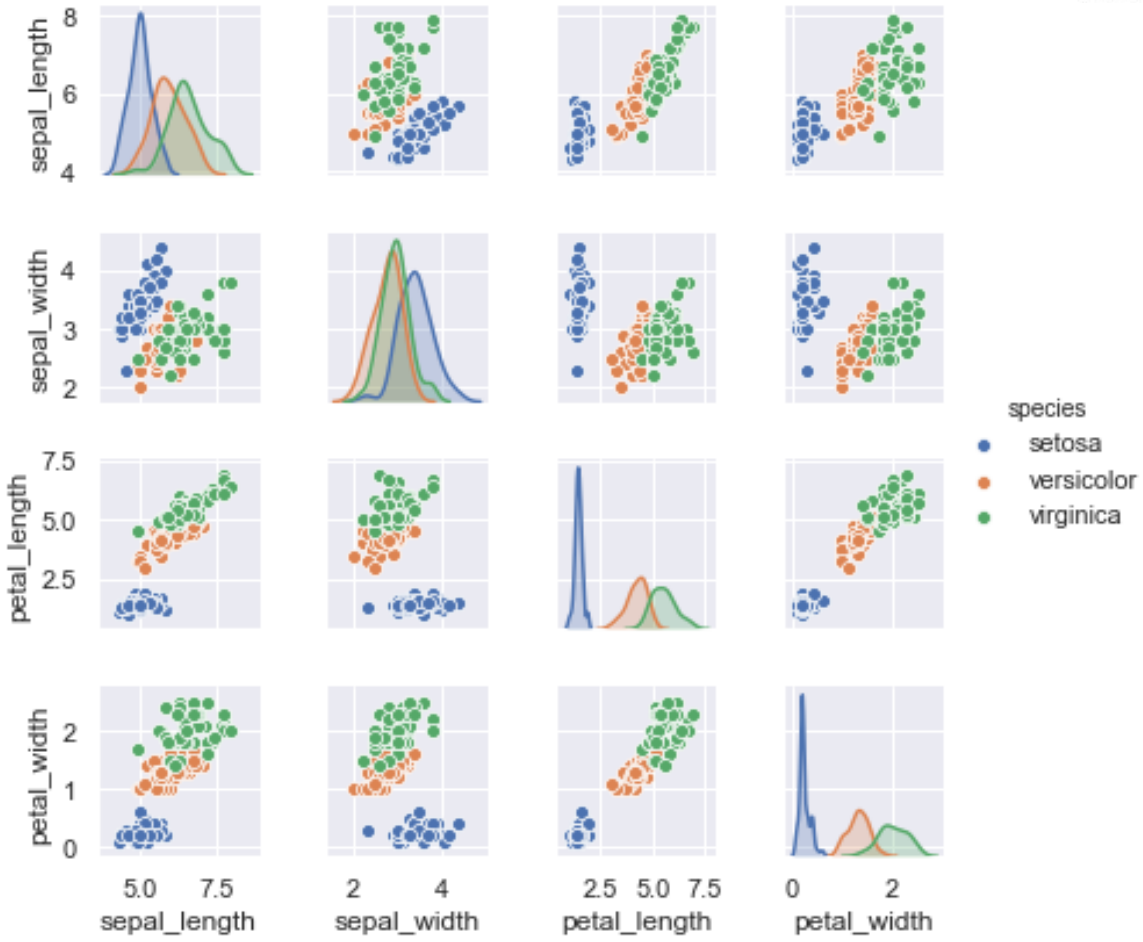
Iris 데이터

```
%matplotlib inline
import seaborn as sns; sns.set()
sns_iris = sns.load_dataset('iris')
sns.pairplot(sns_iris, hue='species', size=1.5)
```

Iris Versicolor

Iris Setosa

Iris Virginica



통계모델:[1] 상관분석

✓ Iris data 실습

```
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
data.shape
```

```
data1= data[:,1]
data2 = data[:, 3]
mean1 =  mean(data1)
std1 = std(data1)
mean2 = mean(data2)
std2 = std(data2)
Qx = data1 - mean1
Qy = data2 - mean2
Q = Qx * Qy
r = sum(Q)/((len(Q)-1)*std1*std2)
print('correlation coefficient:', r)
```

```
correlation coefficient: -0.3685831535601737
```

통계모델:[1] 상관분석

✓ Iris data 실습

```
from sklearn import datasets
iris = datasets.load_iris()
data = iris.data
data.shape
# data[:,0] : sepal length in cm\n
```

```
data1= data[:,2]
data2 = data[:, 3]
mean1 =  mean(data1)
std1 = std(data1)
mean2 = mean(data2)
std2 = std(data2)
Qx = data1 - mean1
Qy = data2 - mean2
Q = Qx * Qy
r = sum(Q)/((len(Q)-1)*std1*std2)
print('correlation coefficient:', r)
```

```
correlation coefficient: 0.9693276155061709
```

통계모델:[1] 상관분석

- ✓ Iris data: Pearson’s correlation

```
from scipy.stats import pearsonr
# calculate Pearson's correlation
corr1, _ = pearsonr(data[:,2], data[:,3])
print('Pearsons correlation between data 2 and data 3: %.3f' % corr1)
corr1, _ = pearsonr(data[:,1], data[:,3])
print('Pearsons correlation between data 1 and data 3: %.3f' % corr1)
```

```
Pearsons correlation between data 2 and data 3: 0.963
Pearsons correlation between data 1 and data 3: -0.366
```

통계모델:[1] 상관분석

- 상관계수 r 구하기
  - ✓ 데이터 샘플: housing

```
import pandas as pd
```

```
housing = pd.read_csv('c:/MLdata/housing.csv')
```

```
corr_matrix = housing.corr()
```

```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
Unnamed: 0            0.072086
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

통계모델:[1] 상관분석

- 상관계수 r 구하기
  - ✓ 데이터 샘플: housing

```
from pandas.plotting import scatter_matrix
attributes = ["median_house_value", "median_income", "total_rooms",
              "housing_median_age"]
scatter_matrix(housing[attributes], figsize=(12, 8))
```