1장. 머신러닝 시작하기

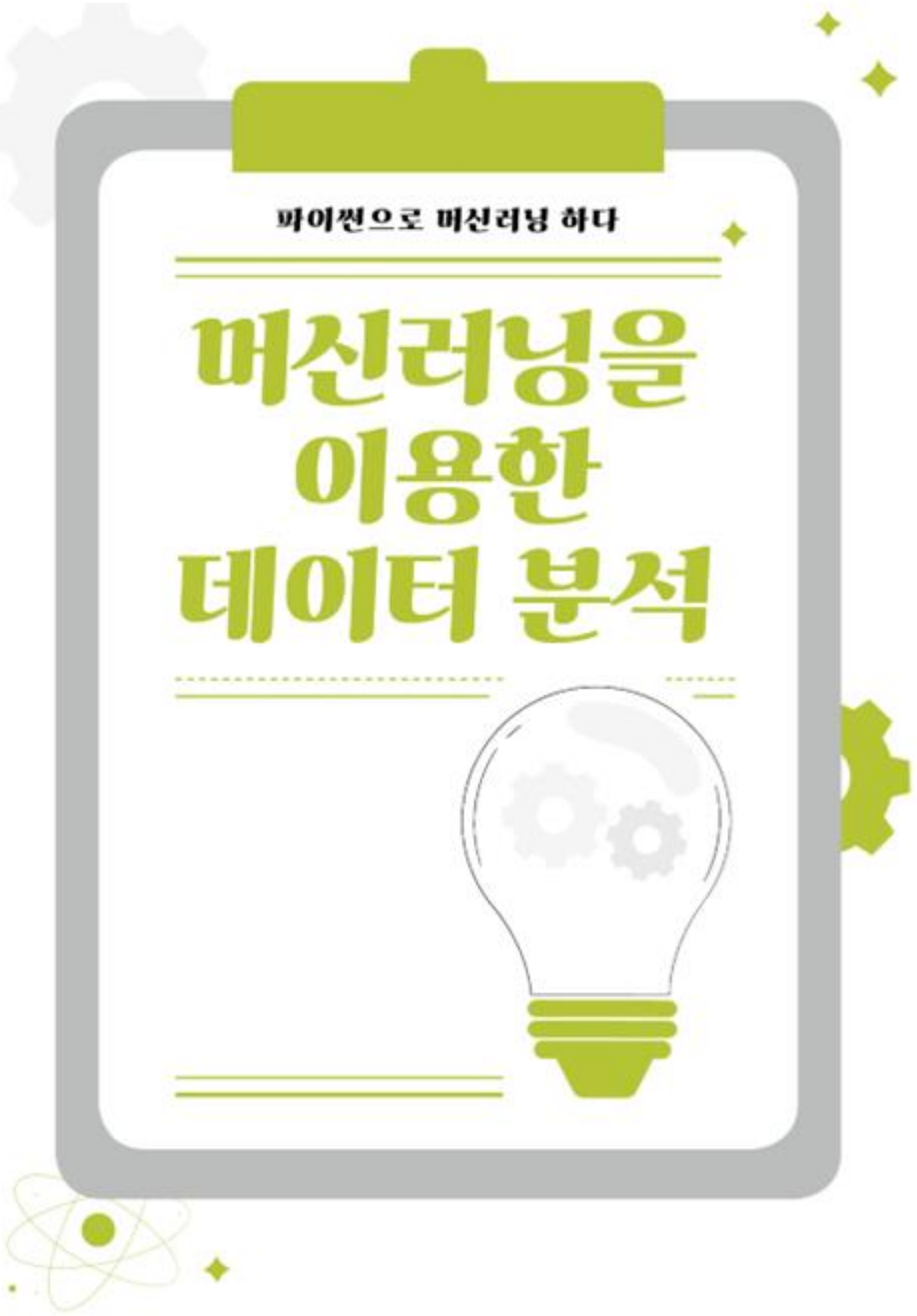
1절. 머신러닝 개요

2절. 데이터 탐색

3절. 데이터 전처리

4절. 단순 데이터 분리

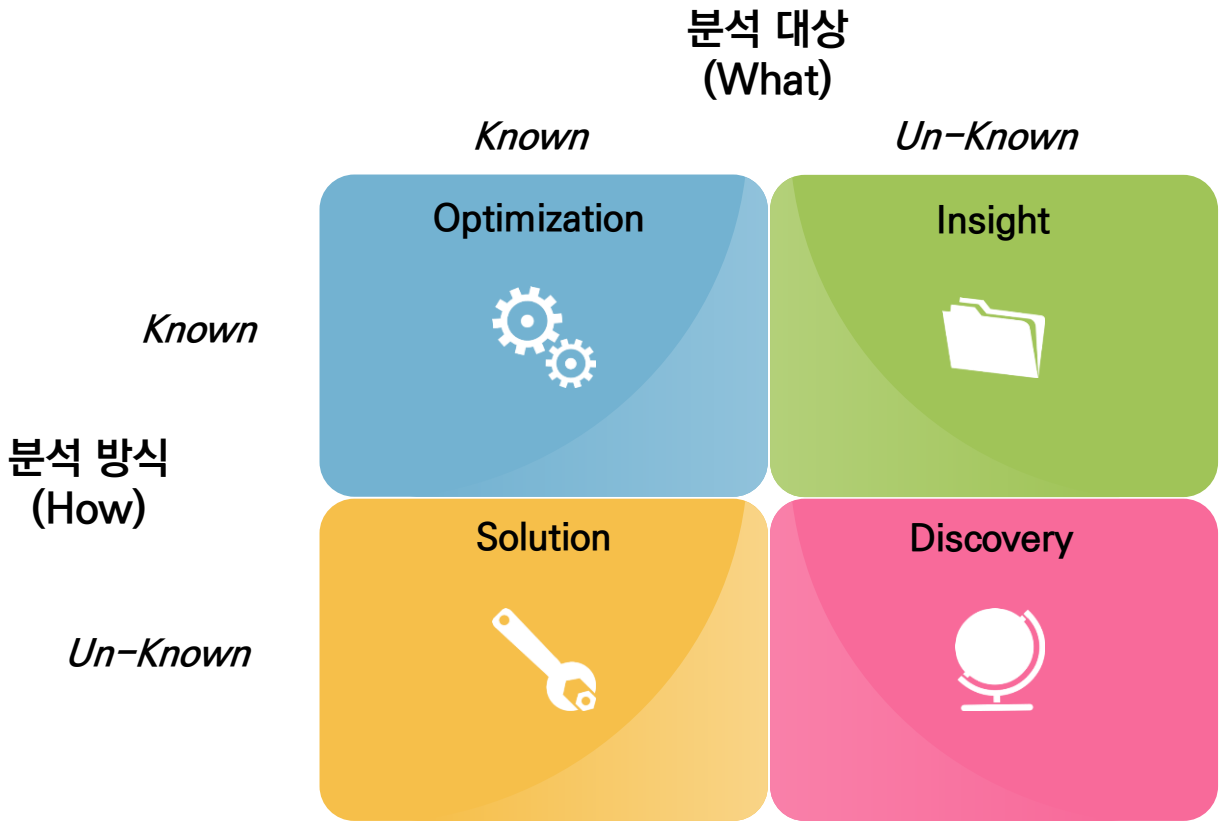
5절. 모형 생성, 예측, 평가



1장. 머신러닝 시작하기

1절. 머신러닝 개요

데이터 분석에서 가장 중요한 것은?



Insight vs. Optimization

Insight

탐색적 자료 분석(Exploratory Data Analysis)

데이터의 특징과 내재하는 구조적인 관계를 알아내기 위한 분석 기법으로 이러한 자료의 탐색 과정을 통하여 얻은 정보를 기초로 통계모형을 세울 수 있음
미지의 특성을 파악하고 자료구조를 파악할 수 있는 증거 수집의 과정
Looking at data to see what it seems to say. It's concentrates on simple arithmetic and easy-to-draw picture. *John Tukey, 1977*

Optimization

기계학습(Machine Learning)

어떻게 하면 더 빨리 학습시키고 어떻게 하면 더 정확히 예측할 수 있을까에 대한 연구를 하며 데이터 전처리, 파생 변수 추가, 모형 선택 등의 방법을 통해서 예측 모형의 평가 점수를 높이는 과정
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. *Tom. Mitchell, 1997*

분석을 잘 하려면…

EDA ML

데이터 탐색 및 시각화

의미 있는 정보는 무엇이 있을까?

- 데이터를 잘 다룰 수 있어야 함
- 데이터의 부분집합 추출 및 병합 등의 작업이 주를 이룸
- 데이터 시각화를 통해 탐색 결과를 이해하기 쉽도록 함

파이썬의 패키지

❖ Numpy & Pandas : 데이터를 다루기 위한 패키지
❖ Matplotlib & Seaborn : 데이터를 시각화하기 위한 패키지

예측력이 높은 모형 생성

어떻게 하면 모형이 예측을 잘 할 수 있을까?

- 데이터 전처리, 파생변수 추가
- 머신러닝 모형 생성 및 예측
- 모형 평가

파이썬의 패키지

❖ Scikit-learn : 기계학습 라이브러리
❖ Statsmodels : 통계 라이브러리

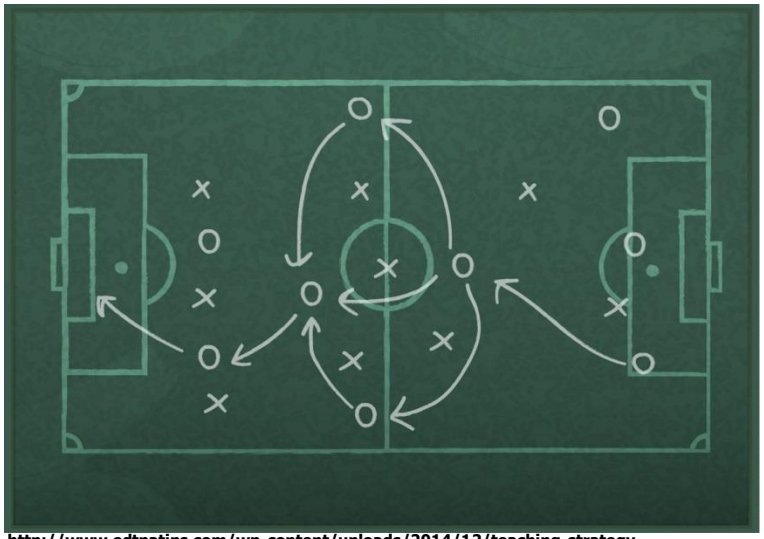
ML – What is Machine Learning?

ML studies algorithms that *improve with* experience.

learn from

Tom. Mitchell(1997,* Definition of the [*general*] *learning problem)

A computer program is said to *learn* from experience *E* with respect to some class of *tasks T* and *performance measure P*, if its performance at tasks in T, as measured by P, improves with experience E -

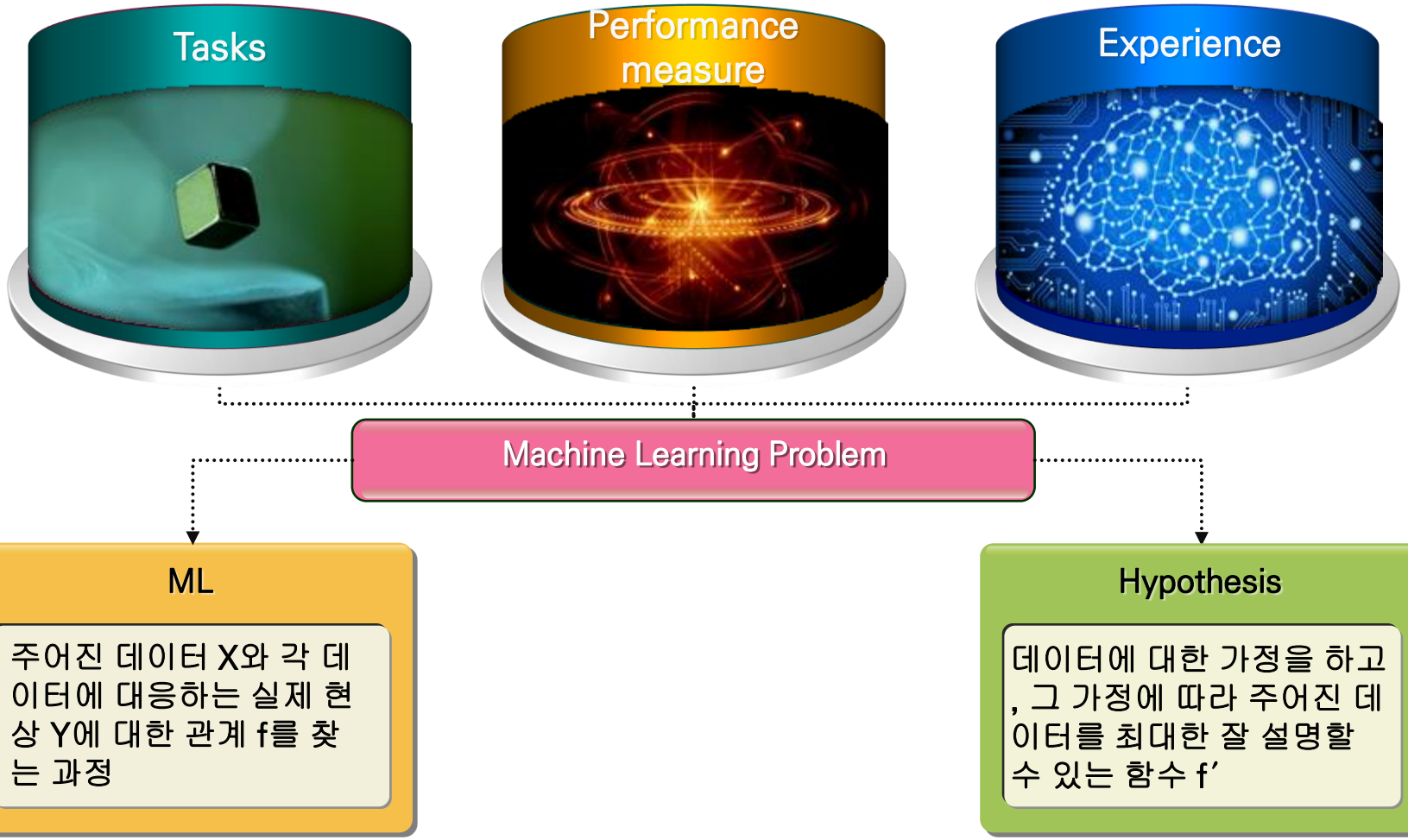


http://www.edtpatips.com/wp-content/uploads/2014/12/teaching-strategy-184317176-1440x1008.jpg

Example: A program for soccer tactics

T : Win the game
P : Goals
E : (x) Players' movements, (y) Evaluation

T를 달성하는 데 있어서 E를 통해 P를 향상시킨다.



지도학습(감독학습) vs. 비지도학습(자율학습)

Supervised Learning

Supervised Learning

Estimate an unknown mapping from known input and target output pairs

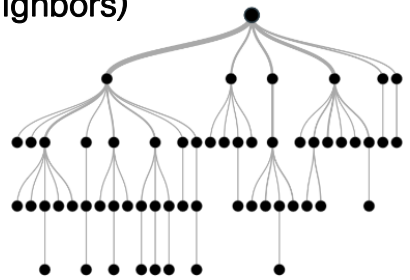
Learn f_w from training set D={(x, y)} $s.t.$ $f_w(x)$=xy = $f(y)$

- Classification : y is discrete
- Regression : y is continuous

Classification

Inputs are divided into two or more classes

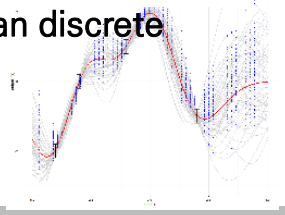
- SVM (Support Vector Machine)
- K-NN (k-Nearest Neighbors)
- Naïve Bayes
- Decision Tree
- Random Forest
- Logistic Regression
- Neural Network



Regression

Outputs are continuous rather than discrete

- Linear Regression
- K-NN
- SVM
- Random Forest



Unsupervised Learning

Unsupervised Learning

Only input values are provided

Learn f_w from D={(x)} $s.t.$ $f_w(x)$=x

- Clustering
- Association

Clustering & Dimension Reduction

A set of inputs is to be divided into groups the group are now known beforehand

- K-means
- Hierarchical clustering
- PCA(Principal Component Analysis)
- Neural Network

Association

End to end connection

- Apriori (arules package)

지도학습(감독학습) vs. 비지도학습(자율학습)

Supervised Learning

Data: (X, y)
– X is data, y is label

Goal: Learn a function to map X –〉 y

Examples
- Classification : y is discrete
- Regression : y is continuous
- Object detection
- Semantic segmentation
- Image captioning

Unsupervised Learning

Data: X
– Just data, no labels!

Goal: Learn some structure of the data

Examples
- Clustering
- Association
- Dimensionality reduction
- Feature learning
- Generative models

ML – Tasks

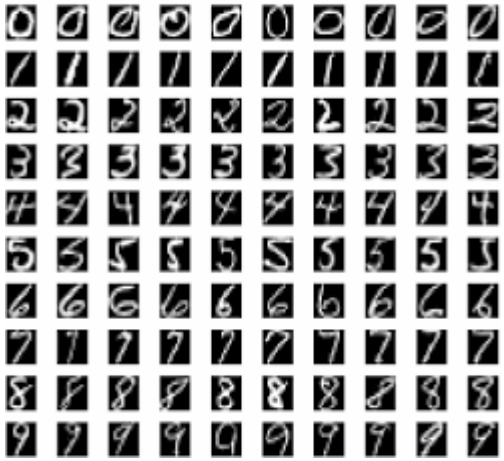
Tom. Mitchell(1997, **Definition of the [*general*] *learning problem)***

A computer program is said to *learn* from experience *E* with respect to some class of *tasks T* and *performance measure P*, if its performance at tasks in T, as measured by P, improves with experience E -

Classification

discrete target values

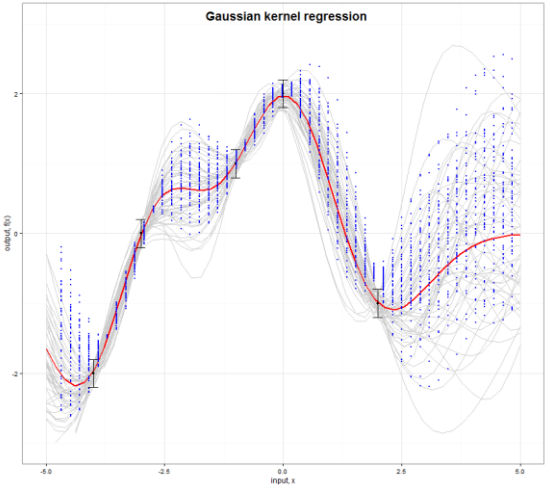
x : pixels (28*28)
y : 0,1,2,3,4,5,6,7,8,9



Regression

real target values

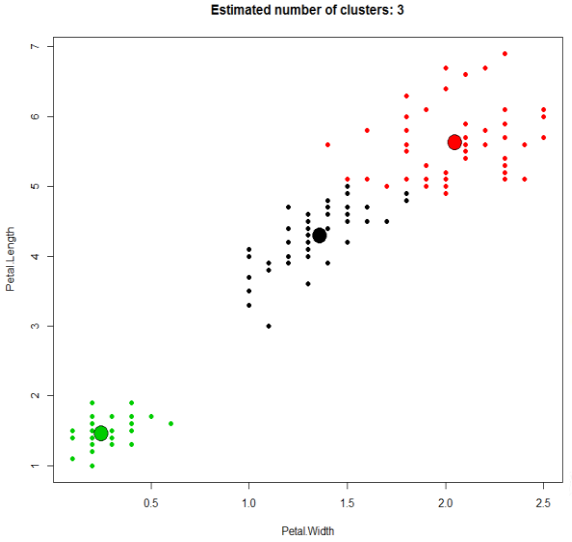
$x \in (0,100)$
y : 0,1, 2,3,⋯,9



Clustering

no target values

$x \in (-3,3) \times (-3,3)$



ML - Performance measure

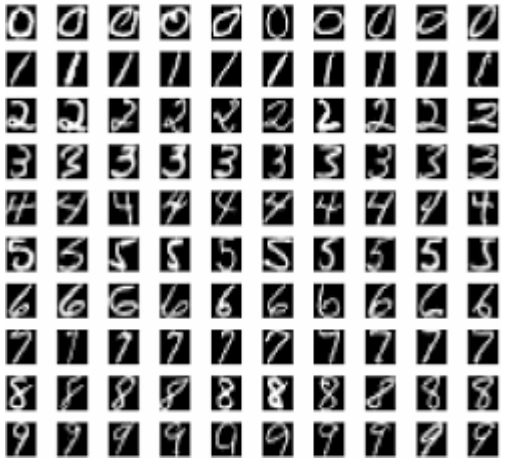
Tom. Mitchell(1997, **Definition of the [*general*] *learning problem)***

A computer program is said to *learn* from experience *E* with respect to some class of *tasks T* and *performance measure P*, if its performance at tasks in T, as measured by P, improves with experience E -

Classification

0-1 loss function

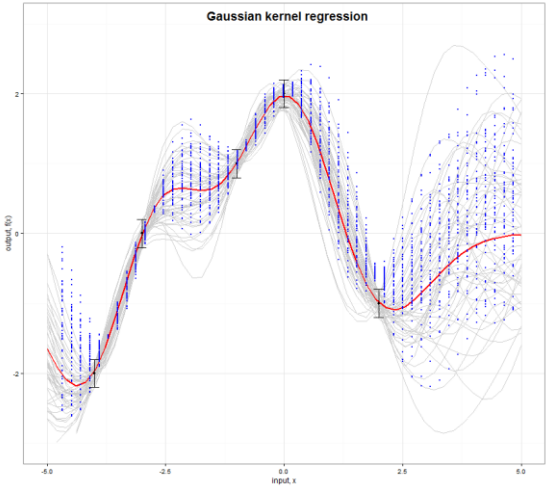
$$L(\hat{y}, y) = I(\hat{y} \neq y)$$



Regression

L2 loss function

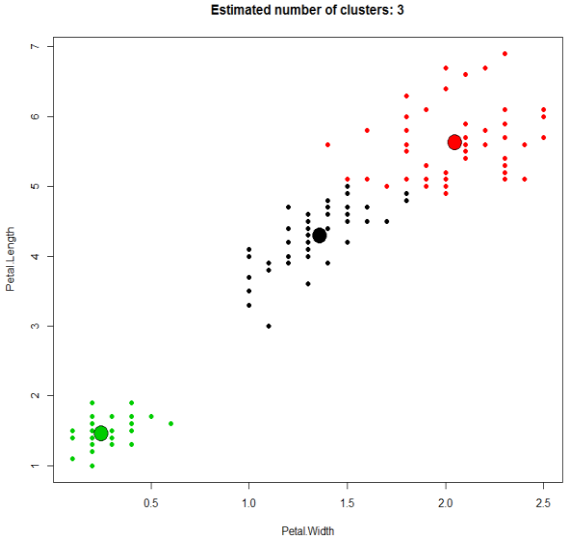
$$L(f, \hat{f}) = \|f - \hat{f}\|_2^2$$



Clustering

no target values

$$L(\hat{y}, y) = I(\hat{y} \neq y)$$
$$L(f, \hat{f}) = \|f - \hat{f}\|_2^2$$



ML – Experience

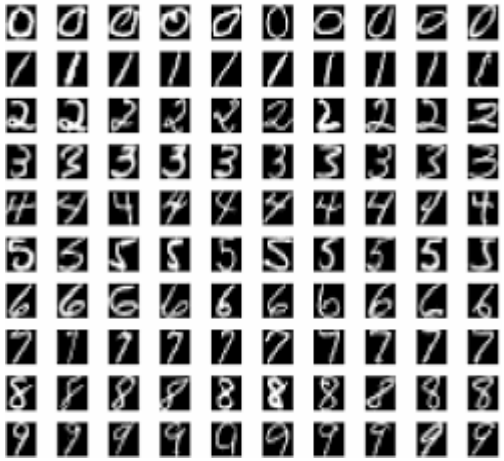
Tom. Mitchell(1997, **Definition of the [*general*] *learning problem)***

A computer program is said to *learn* from experience *E* with respect to some class of *tasks T* and *performance measure P*, if its performance at tasks in T, as measured by P, improves with experience E -

Classification

labeled data

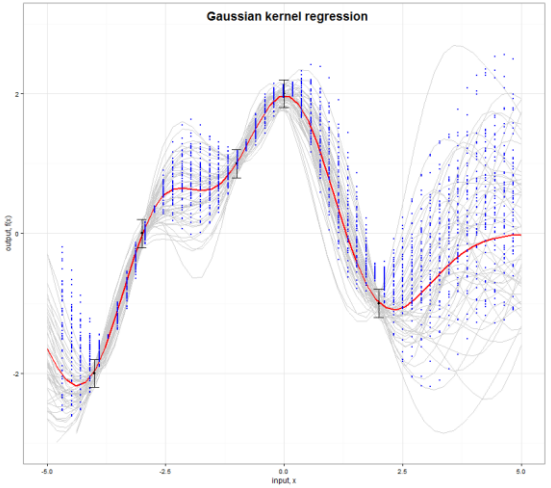
(pixels)→(number)



Regression

labeled data

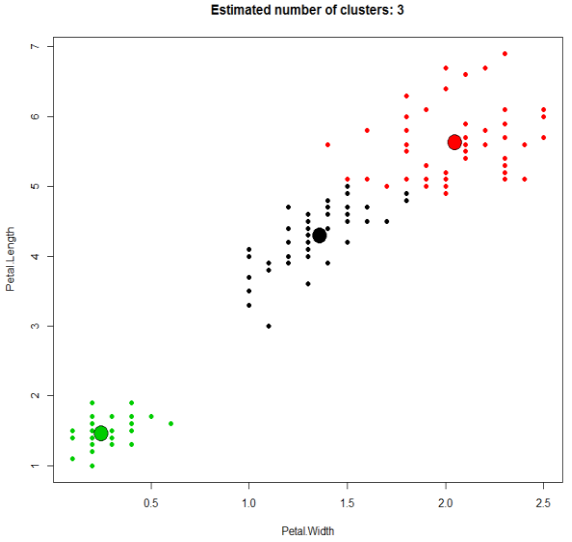
(x) → (y)



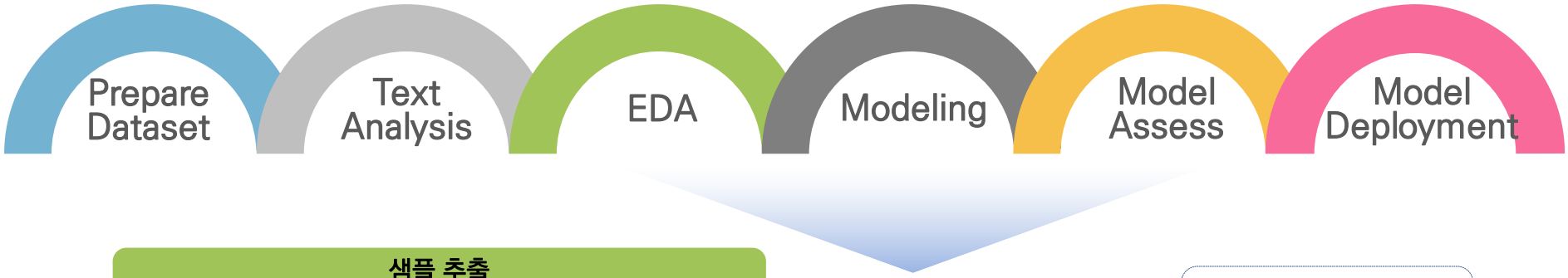
Clustering

unlabeled data

(x_1, x_2)



데이터 분석 단계에서 머신러닝



샘플 추출
Sampling

탐색적 자료분석
Exploratory Data Analysis

유의성 검정
Significance test

차원축소
Dimension Reduction

변수선택
Variable selection

주성분분석
Principal Component Analysis

자율학습
Unsupervised Learning

연관분석
Association

군집분석
Clustering

지도학습
Supervised Learning

Prediction

Forecasting

분류 분석
Classification

회귀 분석
Regression

시계열 분석
Time Series Analysis

최적화
Optimization

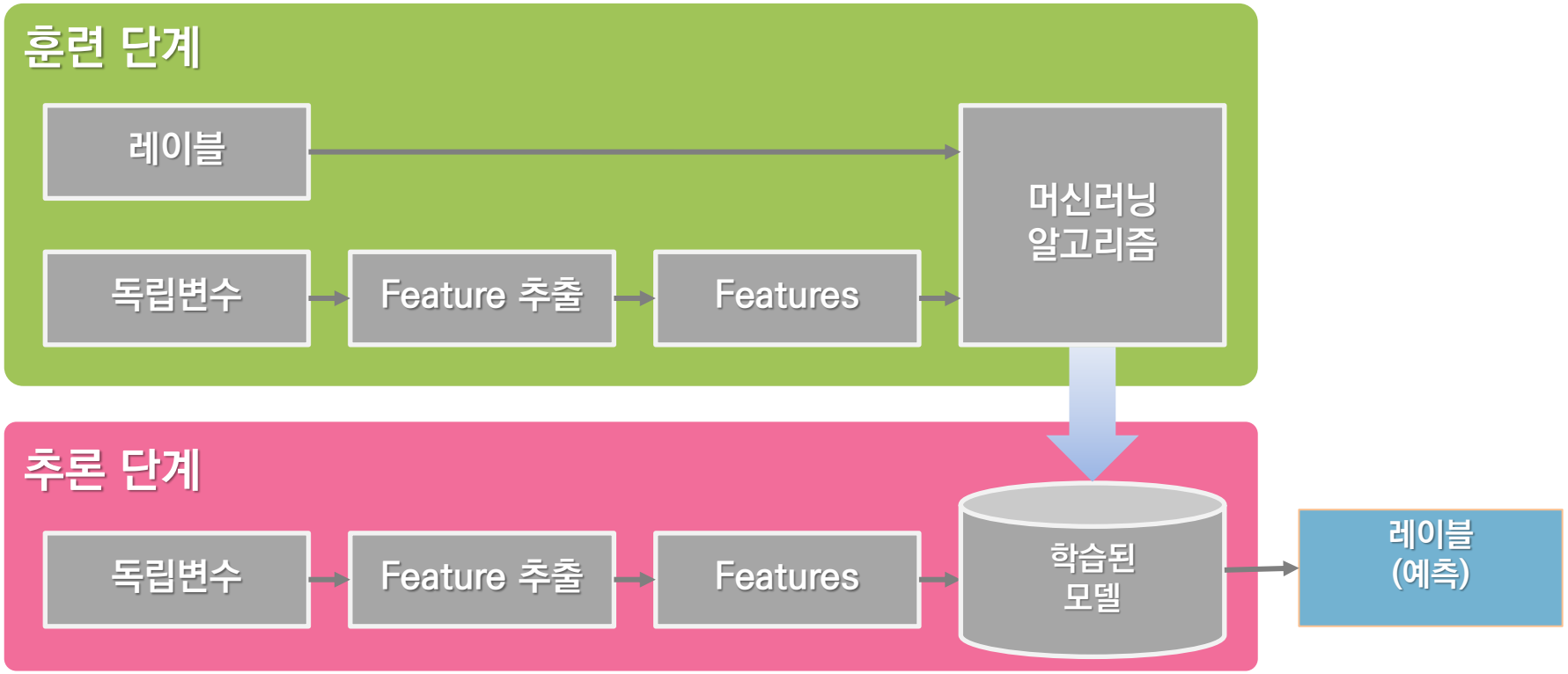
모형 평가
Assessment

적용 전 평가
Front-end

적용 후 평가
Back-end

학습(training) vs. 추론(Prediction/Inference)

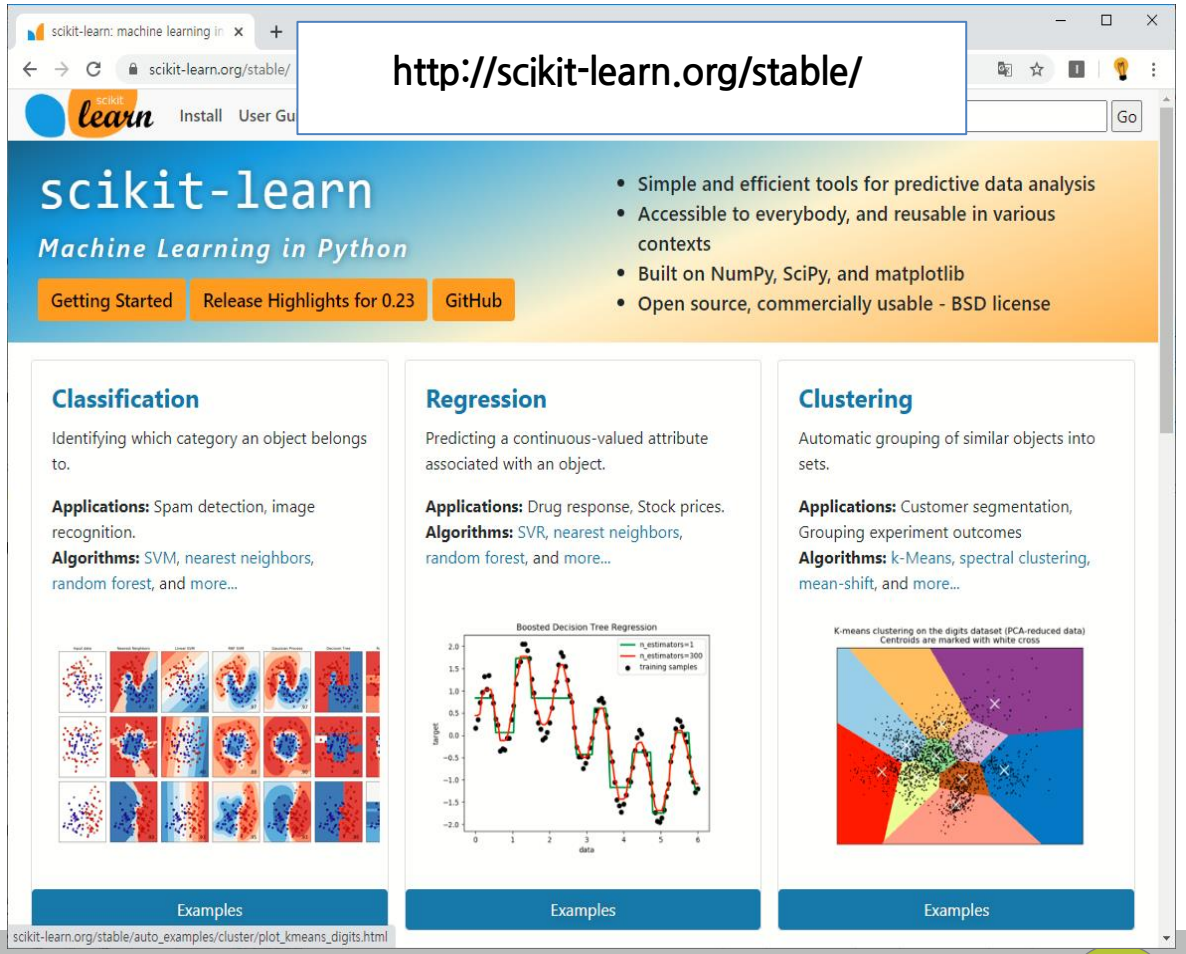
학습: 훈련 데이터를 이용하여 모델을 학습하는 과정
추론: 학습된 모델을 이용하여 미래의 새로운 데이터를 추론/예측하는 과정

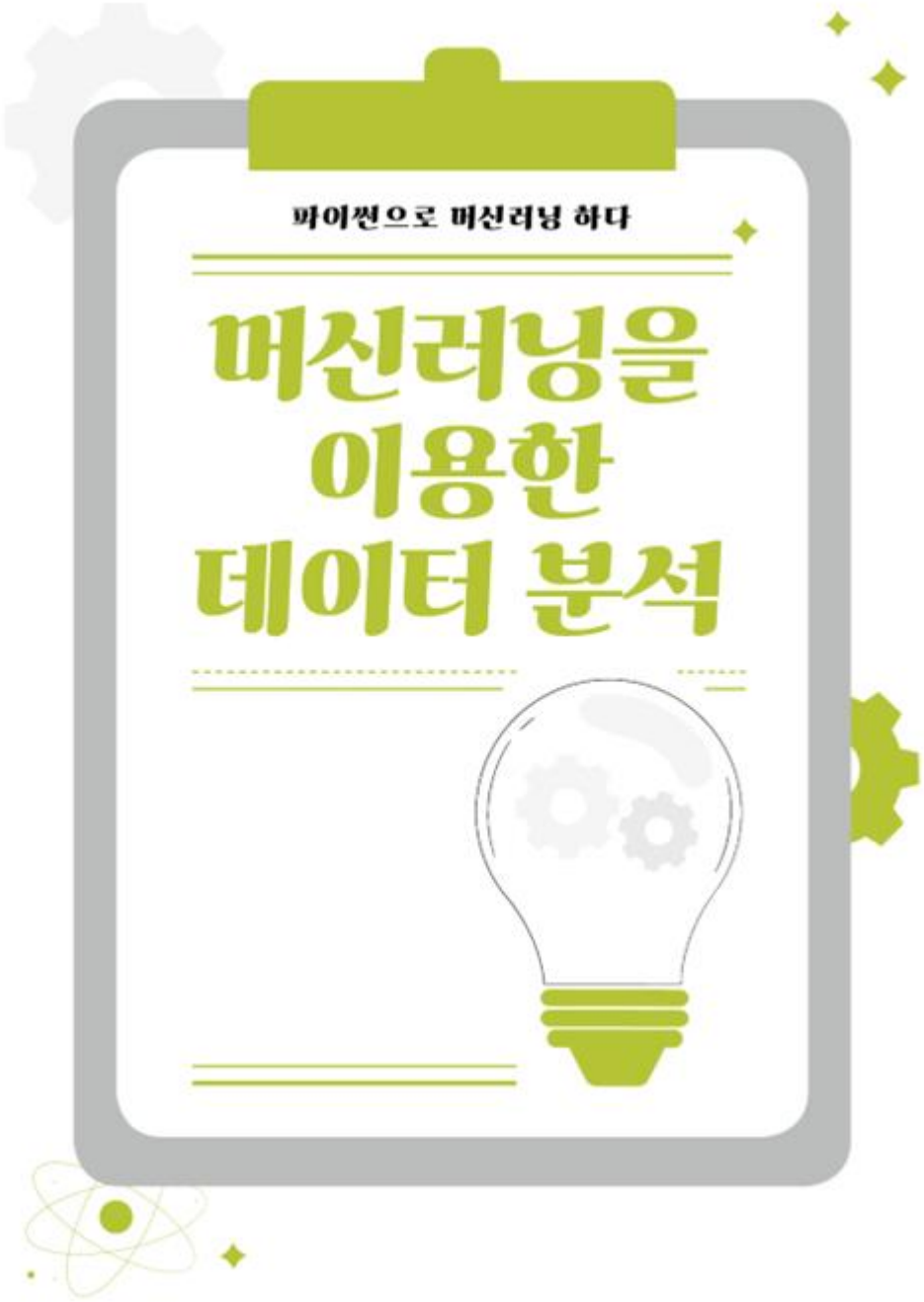


Scikit-learn 패키지

- 예측분석을 위한 간단하고 효율적인 도구
- 상업적으로 사용 가능한 오픈소스 BSD 라이센스이므로 모든 사람이 사용할 수 있음
- NumPy(넘파이), SciPy(사이파이) 및 matplotlib(맷플롯립) 기반

- 사이킷런은 분류(Classification), 회귀(Regression), 군집(Clustering) 분석을 위한 다양한 클래스들이 구현되어 있으며, 이를 통해 예측 모형을 만들 수 있습니다. 뿐만아니라 사이킷런은 차원 축소(Dimensionality reduction), 모델 선택(Model selection), 전처리(Preprocessing)를 위한 많은 기능들이 구현되어 있으므로 머신러닝을 위한 필수 패키지입니다.





1장. 머신러닝 시작하기

2절. 데이터 탐색

- 대량 데이터의 전반적인 형태를 조사하고 데이터의 변환과 축약

변수 차트	그래프	데이터
단일 변수 차트	막대 그래프(Bar chart)	이산형 변수의 빈도수 분포
	파이 그래프(Pie chart)	이산형 변수의 빈도수 분포
	히스토그램(Histogram)	연속형 변수의 빈도수 분포
두 변수 차트	선 그래프(Line graph)	X축과 Y축에 변수를 할당하여 선으로 연결
	시계열 그래프(Time series plot)	X축에 시간 변수를 할당한 Line graph
	산점도 (Scatter plot)	두 개의 연속형 변수의 관계 분석(상관관계, 산포)
다중 변수 차트	모자이크 플롯(Mosaic plot)	이산형 데이터의 다차원 도수 분포표
	스타 차트 (Star chart)	각 데이터가 변수별로 전체 데이터 중에서 최솟값과 최댓값 중 어느 정도의 위치에 있는지 확인(Radar chart)
	평행 좌표 그래프(Parallel coordinate plot)	변수의 수만큼 평행 좌표선을 만들고 각 변수의 관측값을 좌표선에 점으로 표시한 후 이를 선으로 연결
	체르노프 얼굴 그래프 (Chernoff face diagram)	사람 얼굴의 가로 너비, 세로 높이, 눈, 코, 입, 귀 등 각 부위를 변수로 대체하여 시각화

통계 요약

- 도수분포표 (Frequency table)
  - 도수, 백분율 (상대 도수), 누적 백분율, 유효 백분율 (결측값 제외), 누적 유효 백분율
- 중심 위치 측도
  - 평균 (mean, 모평균, 표본 평균), 중앙값 (median), 최빈값 (mode)
- 산포의 측도

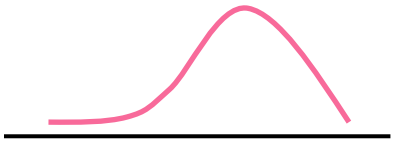
이름	설명
분산(Variance)	각 데이터의 값과 평균과의 차이를 제곱하여 합한 후 이를 데이터수로 나눈 값
표준 편차(Standard deviation)	분산의 제곱근
변이계수(Coefficient of variation)	표준편차를 평균으로 나눈 값, 변이계수간 비교가 가능
백분위수(Percentile)	데이터를 순서대로 나열할 경우 이하 값의 개수가 p%
일사분위수(1st quartile)	Q1, 25% 백분위수
이사분위수(2nd quartile), 중앙값(Median)	Q2, 50% 백분위수
삼사분위수(3rd quartile)	Q3, 75% 백분위수
범위(Range)	최댓값 – 최솟값
IQR(InterQuartile Range)	사분위수 범위, Q3 – Q1
사분위수 그래프(Box-whisker plot)	사분위수 등을 이용하여 그린 데이터의 요약 그림

왜도와 첨도

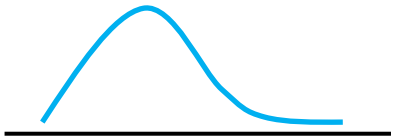
- 왜도와 첨도 (skewness and kurtosis)
  - 왜도 : 표본 데이터의 대칭성 측정, 음수. 왼 꼬리, 0. 대칭, 양수. 오른 꼬리
  - 첨도 : 음수. 낮은 봉우리, 0. 정규분포와 같은 봉우리 높이, 양수. 높은 봉우리

```
import numpy as np
from scipy.stats import kurtosis, skew
data = np.random.normal(0, 1, 10000000)
print("mean : ", np.mean(data))
print("var : ", np.var(data))
print("skew : ", skew(data))
print("kurt : ", kurtosis(data))
```

왜도(Skewness)



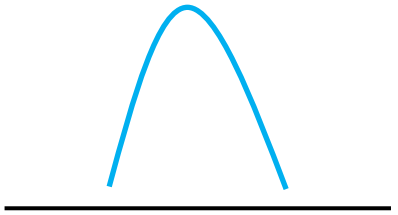
Negative Skew(S<0)
왼쪽으로 긴 꼬리
오른쪽으로 치우친 분포
예) 시험성적



Positive Skew(S>0)
오른쪽으로 긴 꼬리
왼쪽으로 치우친 분포
예) 소득자료

첨도(Kurtosis)

Negative Kurtosis(K<0)
정규분포보다 납작한 분포



Positive Kurtosis(K>0)
정규분포보다 뾰족한 분포

공분산과 상관계수

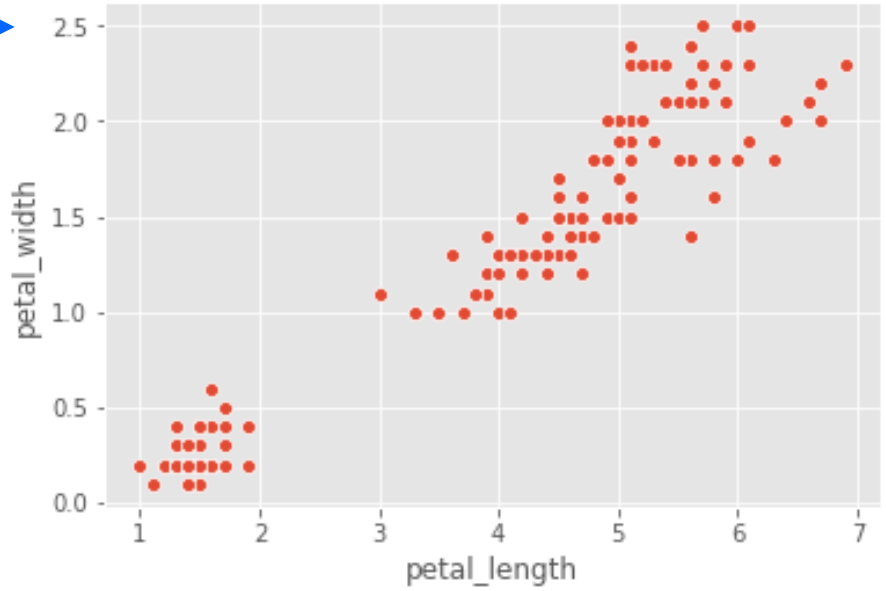
- 공분산 (Covariance) : 양수. 양의 상관관계, 음수, 음의 상관관계
  - 두 개의 확률변수의 분포가 결합된 결합확률분포의 분산
  - 방향성은 나타내지만, 결합 정도에 대한 정보로서는 유용하지 않다.
  - 공분산이 0보다 크면 두 변수는 같은 방향으로 움직이고, 0보다 작으면 다른 방향으로 움직임을 의미한다.
  - 만약 공분산이 0이라면 두 변수간에는 아무런 선형관계가 없으며 두 변수는 서로 독립적인 관계에 있음을 알 수 있다.
  - 두 변수가 독립적이라면 공분산은 0이 되지만, 공분산이 0이라고 해서 항상 독립적이라고 할 수 없다.
- 상관계수(Correlation coefficient) : -1 ~ 1, 공분산을 단위와 상관없는 형태로 가공한 값
  - 두 개의 확률변수 사이의 선형적 관계 정도를 나타내는 척도.
  - 방향성과 선형적 결합 정도에 대한 정보를 모두 포함하고 있다.
  - 두 변수의 공분산을 각 변수의 표준편차로 모두 나누어 구할 수 있으며, -1과 1사이에서 그 값이 결정된다.
  - 공분산은 원래의 단위의 곱이 되기 때문에 경우에 따라서 이를 표준화할 필요가 있으며,
  - 표준화한 결과가 상관계수가 된다.

공분산과 상관계수


```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.corr()
```

	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

```
sns.scatterplot(x="petal_length", y="petal_width", data=iris)
plt.show()
```



기초 통계량


```
iris.describe(include='all')
```

- 수치형 자료 : 최소값, 1사분위수, 중앙값, 평균, 3사분위수, 최대값 표시
- describe() 함수의 인수에 include='all'을 추가하면 숫자 자료형이 아닌 변수들의 중복을 제거한 값의 개수(unique), 가장 많은 데이터(top), 그리고 가장 많은 데이터의 수(freq)에 대한 정보를 볼 수 있음

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.057333	3.758000	1.199333	NaN
std	0.828066	0.435866	1.765298	0.762238	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

상관 분석(Correlation Analysis)

- 두 변수간에 선형적 관계가 있는지 분석

- Pearson correlation coefficient
  - 두 변수가 모두 연속형 자료일 때 두 변수간 선형적인 상관 관계의 크기를 모수적(parametric)인 방법으로 나타낸 값

- Spearman correlation coefficient
  - 상관 관계를 분석하고자 하는 두 연속형 변수의 분포가 심각하게 정규 분포(normal distribution)을 벗어난다거나 또는 두 변수가 순위 척도(ordinal scale) 자료일 때 사용하는 값

피어슨 상관계수

- 피어슨 상관계수 (Pearson Correlation Coefficient)
  - 두 변수가 모두 연속형 자료일 때
  - -1 ~ 1 사이의 값 (-1. 반비례, 1. 비례)
  - 일반적으로 0.6 이상이면 양의 상관 관계, -0.6 이하이면 음의 상관 관계

$$\rho_{X,\,Y} = cov(X, Y)/\sigma_X * \sigma_Y$$

- cov(X, Y) : X, Y의 공분산
- óX : X의 표준 편차
- óY : Y의 표준 편차

```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.corr(method='pearson')
```

	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

상관계수와 관계

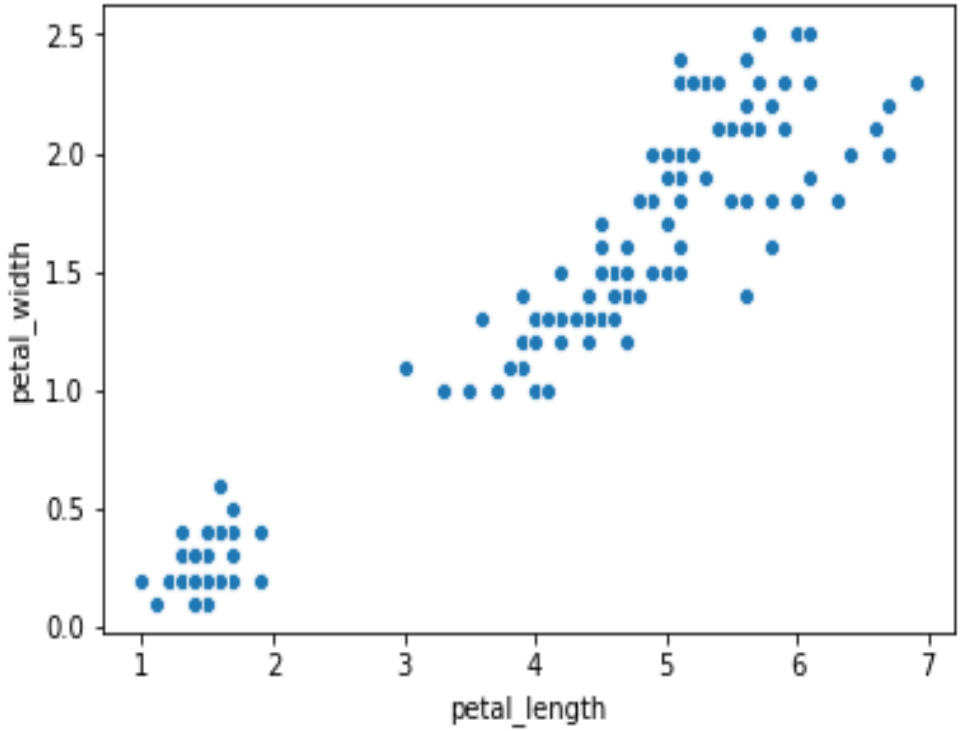
범위	관계
-1.0 ≤ r ≤ -0.7	매우 강한 음(-) 의 상관관계
-0.7 < r ≤ -0.3	강한 음(-) 의 상관관계
-0.3 < r ≤ -0.1	약한 음(-) 의 상관관계
-0.1 < r ≤ 0.1	상관관계 없음
0.1 < r ≤ 0.3	약한 양(+) 의 상관관계
0.3 < r ≤ 0.7	강한 양(+) 의 상관관계
0.7 < r ≤ 1.0	매우 강한 양(+) 의 상관관계

피어슨 상관계수


```
from scipy.stats.stats import pearsonr
pearsonr(iris.petal_length, iris.petal_width)
    (0.9628654314027961, 4.675003907327543e-86)
```

- 유의확률(p-value)이 0.05보다 작으므로 귀무가설을 기각
- 그러므로 petal_length(꽃잎 길이)와 petal_width(꽃잎 너비)는 관계가 있음

```
sns.scatterplot(iris.petal_length, iris.petal_width)
```



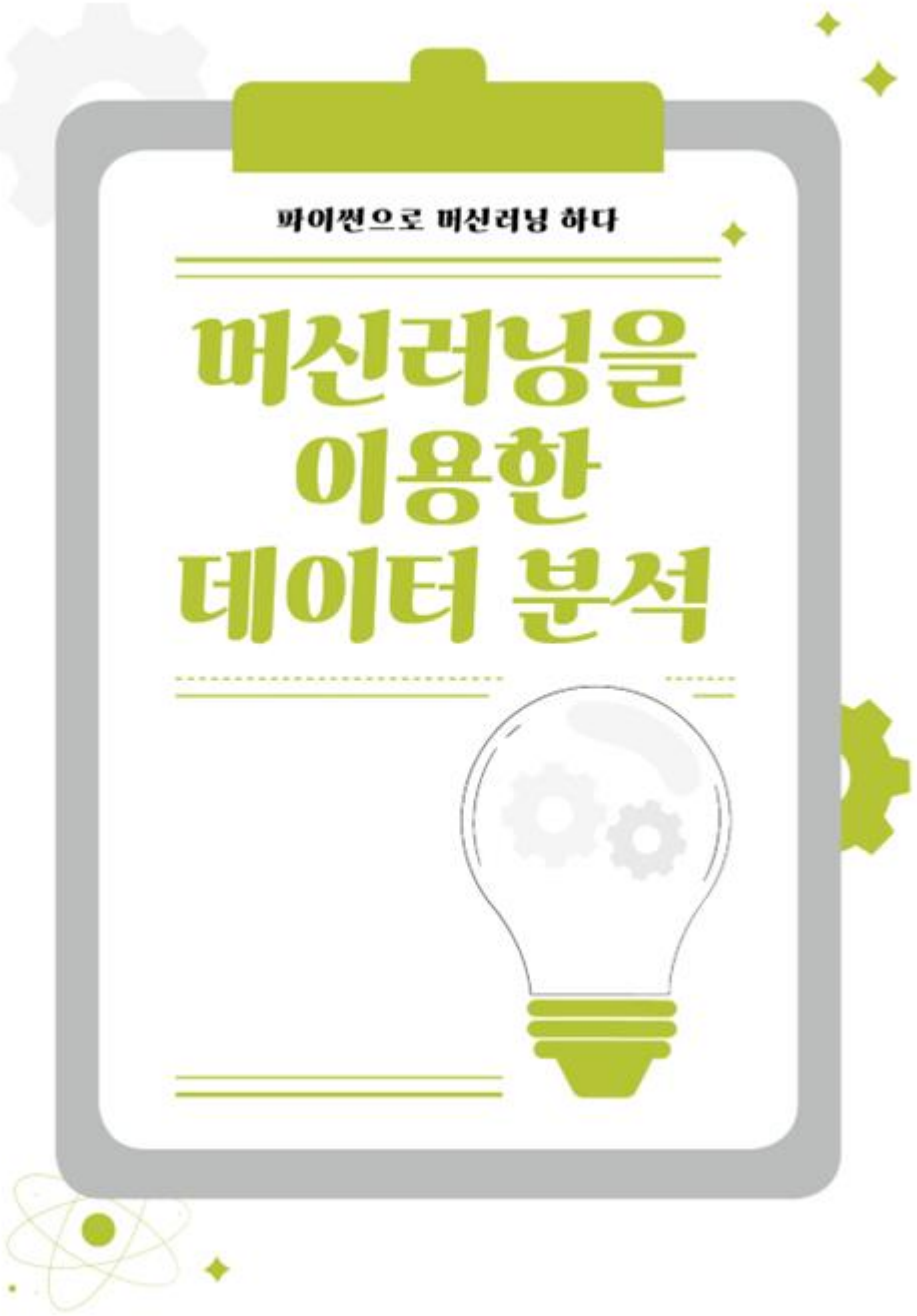
스피어만 상관계수(Spearman's Rank Correlation Coefficient)

- 데이터의 순위로 상관계수 계산 (서열척도인 변수에 사용)
- 데이터가 서열 척도인 경우, 값 대신 순위를 사용하여 계산한 상관계수

```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.corr(method='spearman')
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.166778	0.881898	0.834289
sepal_width	-0.166778	1.000000	-0.309635	-0.289032
petal_length	0.881898	-0.309635	1.000000	0.937667
petal_width	0.834289	-0.289032	0.937667	1.000000

```
from scipy.stats.stats import spearmanr
spearmanr(iris.petal_length, iris.petal_width)
SpearmanrResult(correlation=0.937668235763412, pvalue=8.156596854126675e-70)
```



1장. 머신러닝 시작하기

3절. 데이터 전처리

sklearn.preprocessing 패키지는 모델을 만들기 적합한 데이터로 변환하는 것을 도와줌

결측치 추정

Null 비율 20% 미만

연속형

이상치 제외 평균

범주형

Null 값을 미리 정의된 값으로 대체

레이블 인코딩

문자를 숫자로 변환

원-핫 인코딩

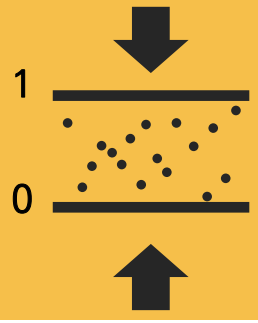
성별을 저장하는 변수가 male, female을 가진다면 원-핫 인코딩하면 sex_male, sex_female이라는 두 개의 변수가 만들어지고 male일 경우 (1, 0)값을 가지며, female일 경우 (0, 1)값을 가짐

표준화

최대값 및 최소값 정의

0~1 표준화

연속형 데이터의 경우 변수들의 Scale이 다르면 모델링 과정이 제대로 수행이 안 되기 때문에 표준화 시킴

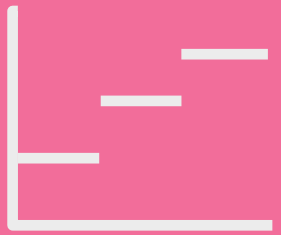


구간화

연속형 ▶ 범주형

의미부여

연속형으로 산재되어 있는 데이터를 일정 규칙에 맞추어 범주형으로 변형 함으로써 데이터에 의미 부여



데이터를 정해진 구간 사이의 값으로 표준화(Standardization)

sklearn.preprocessing 모듈의 표준화 함수들

- scale(x) : 표준 정규분포를 사용해 표준화
- robust_scale(x) : 중위수(median)와 사분위범위(interquartile range)를 사용하여 표준화
- minmax_sacle(x) :최댓값과 최솟값을 사용하여 표준화. 0~1
- maxabs_scale(x) :최대 절댓값을 사용하여 표준화. -1~1

예제에 사용할 데이터

```
import seaborn as sns
iris = sns.load_dataset("iris")
x = iris.iloc[:, :-1]
x.head()
```

	sepal_length	sepal_width	petal_length	petal_width
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

scale() 함수는 평균을 0, 표준편차를 1이 되도록 표준화

```
from sklearn.preprocessing import scale
x_scaled = scale(x)
x_scaled[:5, :]
```

```
array([[-0.90068117,  1.01900435, -1.34022653, -1.3154443 ],
       [-1.14301691, -0.13197948, -1.34022653, -1.3154443 ],
       [-1.38535265,  0.32841405, -1.39706395, -1.3154443 ],
       [-1.50652052,  0.09821729, -1.2833891 , -1.3154443 ],
       [-1.02184904,  1.24920112, -1.34022653, -1.3154443 ]])
```

```
x_scaled.mean(axis=0)
```

```
array([-2.77555756e-16, -9.69594775e-16, -8.65233811e-16, -4.6
6293670e-16])
```

robust_scale()

robust_scale() 함수는 중위수와 사분위범위(interquartile range)를 사용하여 표준화

```
from sklearn.preprocessing import robust_scale
iris_robust_scaled = robust_scale(x)
iris_robust_scaled[:5, :]
```

```
array([[-0.53846154,  1.        , -0.84285714, -0.73333333],
       [-0.69230769,  0.        , -0.84285714, -0.73333333],
       [-0.84615385,  0.4       , -0.87142857, -0.73333333],
       [-0.92307692,  0.2       , -0.81428571, -0.73333333],
       [-0.61538462,  1.2       , -0.84285714, -0.73333333]])
```

minmax_scale()

minmax_scale() 함수는 최솟값을 0, 최댓값을 1에 매핑시켜 표준화

```
from sklearn.preprocessing import minmax_scale
iris_minmax_scaled = minmax_scale(x)
iris_minmax_scaled[:5, :]
```

```
array([[0.22222222, 0.625     , 0.06779661, 0.04166667],
       [0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.19444444, 0.66666667, 0.06779661, 0.04166667]])
```

maxabs_scale()

maxabs_scale() 함수는 절댓값이 가장 큰 값을 1로 정하면서 0부터 1사이의 값에 매핑

```
from sklearn.preprocessing import maxabs_scale
iris_maxabs_scaled = maxabs_scale(x)
iris_maxabs_scaled[:5, :]
```

```
array([[0.64556962, 0.79545455, 0.20289855, 0.08      ],
       [0.62025316, 0.68181818, 0.20289855, 0.08      ],
       [0.59493671, 0.72727273, 0.1884058 , 0.08      ],
       [0.58227848, 0.70454545, 0.2173913 , 0.08      ],
       [0.63291139, 0.81818182, 0.20289855, 0.08      ]])
```

maxabs_scale()

음수는 부호를 그대로 유지하는 것이 minmax_scale() 함수와 다른 점임

```
1 import numpy as np
2 X = np.array([[ 1., -1.,  2.],
3               [ 2.,  0.,  0.],
4               [ 0.,  1., -1.]])
```

```
1 minmax_scale(X)
```

```
array([[0.5       , 0.        , 1.        ],
       [1.        , 0.5       , 0.33333333],
       [0.        , 1.        , 0.        ]])
```

```
1 maxabs_scale(X)
```

```
array([[ 0.5, -1. ,  1. ],
       [ 1. ,  0. ,  0. ],
       [ 0. ,  1. , -0.5]])
```

스케일과 스케일 백

StandardScaler, MinMaxScaler, MaxAbsScaler 클래스를 이용해서 표준화

표준화 방법은 scale(), minmax_scale(), maxabs_scale() 함수와 같음

이들 클래스를 이용하면 표준화 후 표준화한 값을 원래의 값 범위로 되돌릴 수 있음

메서드	설명
fit(X[, y])	스케일링에 사용될 평균 및 표준 편차를 계산합니다.
fit_transform(X[, y])	평균과 표준편차를 계산하고 표준화를 수행해서 데이터를 변환합니다.
get_params([deep])	파라미터를 가져옵니다.
inverse_transform(X[, copy])	데이터를 원래 표현으로 스케일 백 합니다.
partial_fit(X[, y])	스케일링을 위해 X에 대한 평균 및 표준편차를 계산합니다. (RobustScaler 클래스에는 이 메서드가 없습니다.)
set_params(**params)	매개변수를 설정합니다.
transform(X[, y, copy])	표준화를 수행해서 데이터를 변환합니다.

StandardScaler 객체를 이용해 표준화

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(x)
iris_scaled = sc.transform(x)
iris_scaled[:5, :]
```

```
array([[-0.90068117,  1.01900435, -1.34022653, -1.3154443 ],
       [-1.14301691, -0.13197948, -1.34022653, -1.3154443 ],
       [-1.38535265,  0.32841405, -1.39706395, -1.3154443 ],
       [-1.50652052,  0.09821729, -1.2833891 , -1.3154443 ],
       [-1.02184904,  1.24920112, -1.34022653, -1.3154443 ]])
```

StandardScaler 객체를 이용해 스캐일백

```
1 iris_origin = sc.inverse_transform(iris_scaled)
2 iris_origin[:5,:]
```

```
array([[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]])
```

레이블 인코딩

- 레이블 인코딩(Label Encoding)은 실제 값에 상관없이 0~K-1까지의 정수로 변환하는 것
- 지역 또는 성별 등 문자로 되어있는 데이터를 숫자로 변환해야 함
- 레이블 인코딩은 preprocessing 모듈의 LabelEncoder 클래스를 이용

메서드	설명
fit(y)	레이블 인코더 모델을 생성합니다.
fit_transform(y)	레이블 인코더 모델을 생성하고 인코드된 레이블을 반환합니다.
get_params([deep])	매개변수를 반환합니다.
inverse_transform(y)	레이블을 원래 인코딩으로 다시 변환합니다.
set_params(**params)	매개변수를 설정합니다.
transform(y)	라벨을 표준화 된 인코딩으로 변환합니다.

레이블 인코딩


```
1  from sklearn.preprocessing import LabelEncoder
2  le = LabelEncoder()
3  le.fit(y)
4  species = le.transform(y)
```

```
1  species
```

```
array([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, 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],
      dtype=int64)
```

원-핫 인코딩

원-핫(one-hot) 인코딩은 클래스의 수만큼 0 또는 1을 갖는 열을 이용해서 데이터를 표현

성별(sex)을 저장하는 변수가 male, female을 가진다면 sex_male, sex_female이라는 두 개의 변수가 만들어지고 male일 경우 두 변수가 각각 1, 0값을 가지며, female일 경우 두 변수가 각각 0, 1값을 가짐

원-핫 인코딩은 preprocessing 모듈의 OneHotEncoder 클래스를 이용

원-핫 인코딩


```
1 from sklearn.preprocessing import OneHotEncoder
2 enc = OneHotEncoder(categories='auto')
3 enc.fit(species.reshape(-1,1))
```

```
OneHotEncoder(categorical_features=None, categories='auto', drop=None,
              dtype=<class 'numpy.float64'>, handle_unknown='error',
              n_values=None, sparse=True)
```

```
1 iris_onehot = enc.transform(species.reshape(-1,1))
2 iris_onehot
```

```
<150x3 sparse matrix of type '<class 'numpy.float64'>'
        with 150 stored elements in Compressed Sparse Row format>
```

```
1 iris_onehot.toarray()[:5]
```

```
array([[1., 0., 0.],
       [1., 0., 0.],
```

pandas.get_dummies()


```
1 import pandas as pd
2 pd.get_dummies(iris.species)
```

	setosa	versicolor	virginica
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...	...	...	...
145	0	0	1
146	0	0	1
147	0	0	1
148	0	0	1
149	0	0	1

150 rows × 3 columns

평균값 인코딩

레이블에 따른 예측값의 평균은 레이블 값에 따라 달라질 수 있음

평균값 인코딩은 레이블 값을 수치적으로도 표현하면서도 서로 구분할 수 있음

이렇게 인코딩한 값은 예측값과 수치적인 면에서 연관이 있음

```
1 import seaborn as sns
2 titanic = sns.load_dataset("titanic")
3 sex_mean = titanic.groupby("sex")["survived"].mean()
4 sex_mean
```

```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

```
1 titanic['sex_mean'] = titanic['sex'].map(sex_mean)
2 titanic[['sex', 'sex_mean']].head()
```

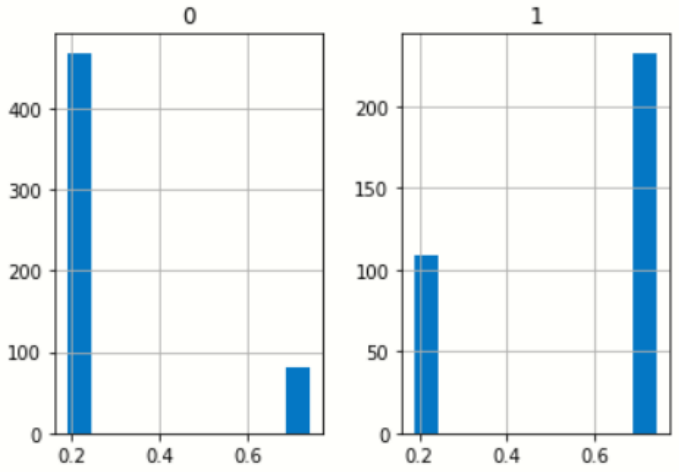
	sex	sex_mean
0	male	0.188908
1	female	0.742038
2	female	0.742038
3	female	0.742038
4	male	0.188908

평균값 인코딩


```
titanic_pivot = titanic.pivot_table(columns="survived",
                                    index=titanic.index,
                                    values='sex_mean')
titanic_pivot.head()
```

survived	0	1
0	0.188908	NaN
1	NaN	0.742038
2	NaN	0.742038
3	NaN	0.742038
4	0.188908	NaN

```
_ = titanic_pivot.hist(bins=10)
```



- 0.188908 은 남자의 인코딩된 값입니다. 예측값(target)에서 0이 더 많고 1이 적게 있음을 볼 수 있습니다. 여자의 경우는 그 반대입니다. 이것은 레이블 값과 타겟 값 사이의 일종의 Correlation이 있음을 알 수 있는 것입니다.
- 평균값 인코딩의 장점은 만들어지는 특징변수의 수가 매우 적어, One-hot의 문제였던 차원의 저주가 없습니다. 그러므로 비교적 빠른 학습이 이루어집니다. 그리고 회귀(Regression)뿐만 아니라 분류(Classification)에서도 이런 특징변수들은 예측값에 좀 더 가깝게 학습되게 합니다.

결측값 처리

데이터에 누락된 정보(결측값, Missing Value)가 있을 경우 이 값을 다른 값으로 채워야 함

impute 모듈의 SimpleImputer 클래스를 이용하면 데이터의 평균, 중앙값, 최빈값 또는 상수값 중 하나로 결측값을 채울 수 있음

sklearn.impute.SimpleImputer

- strategy : str, 기본값 'mean', 결측값을 채울 방법을 지정합니다. 결측값을 채울 방법은 평균(mean)값으로 채우기, 중위수(median)로 채우기, 최빈값(most_frequent), 그리고 상수값(constant, 문자열 또는 숫자 데이터)으로 채우기가 있음

결측값을 갖도록 nan값 추가


```
1 import seaborn as sns
2 iris = sns.load_dataset("iris")
3 iris_X = iris.iloc[:, :-1]
4 iris_y = iris.iloc[:, -1]
```

```
1 import random
2 random.seed(7902)
3 for col in range(4) :
4     iris_X.iloc[[random.sample(range(len(iris)), 20)],col] = float('nan')
```

```
1 iris_X.head()
```

	sepal_length	sepal_width	petal_length	petal_width
0	NaN	3.5	NaN	0.2
1	4.9	NaN	1.4	0.2
2	4.7	3.2	NaN	NaN
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

평균으로 채우기


```
1 iris_X.mean(axis=0)
```

```
sepal_length    5.790000
sepal_width     3.073846
petal_length    3.749231
petal_width     1.226154
dtype: float64
```

	sepal_length	sepal_width	petal_length	petal_width
0	NaN	3.5	NaN	0.2
1	4.9	NaN	1.4	0.2
2	4.7	3.2	NaN	NaN
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
1 from sklearn.impute import SimpleImputer
2 imp_mean = SimpleImputer(strategy='mean')
3 imp_mean.fit(iris_X)
```

```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='mean', verbose=0)
```

```
1 iris_new = imp_mean.transform(iris_X)
2 iris_new[:5,]
```

```
array([[5.79      , 3.5       , 3.74923077, 0.2       ],
       [4.9       , 3.07384615, 1.4       , 0.2       ],
       [4.7       , 3.2       , 3.74923077, 1.22615385],
       [4.6       , 3.1       , 1.5       , 0.2       ],
       [5.        , 3.6       , 1.4       , 0.2       ]])
```

중앙값으로 채우기


```
1 iris_X.median(axis=0)
```

```
sepal_length    5.7
sepal_width     3.0
petal_length    4.4
petal_width     1.3
dtype: float64
```

	sepal_length	sepal_width	petal_length	petal_width
0	NaN	3.5	NaN	0.2
1	4.9	NaN	1.4	0.2
2	4.7	3.2	NaN	NaN
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
1 imp_median = SimpleImputer(strategy='median')
2 imp_median.fit(iris_X)
```

```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='median', verbose=0)
```

```
1 iris_median = imp_median.transform(iris_X)
2 iris_median[:5,]
```

```
array([[5.7, 3.5, 4.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 4.4, 1.3],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

최빈값으로 채우기


```
1 iris_X.mode(axis=0)
```

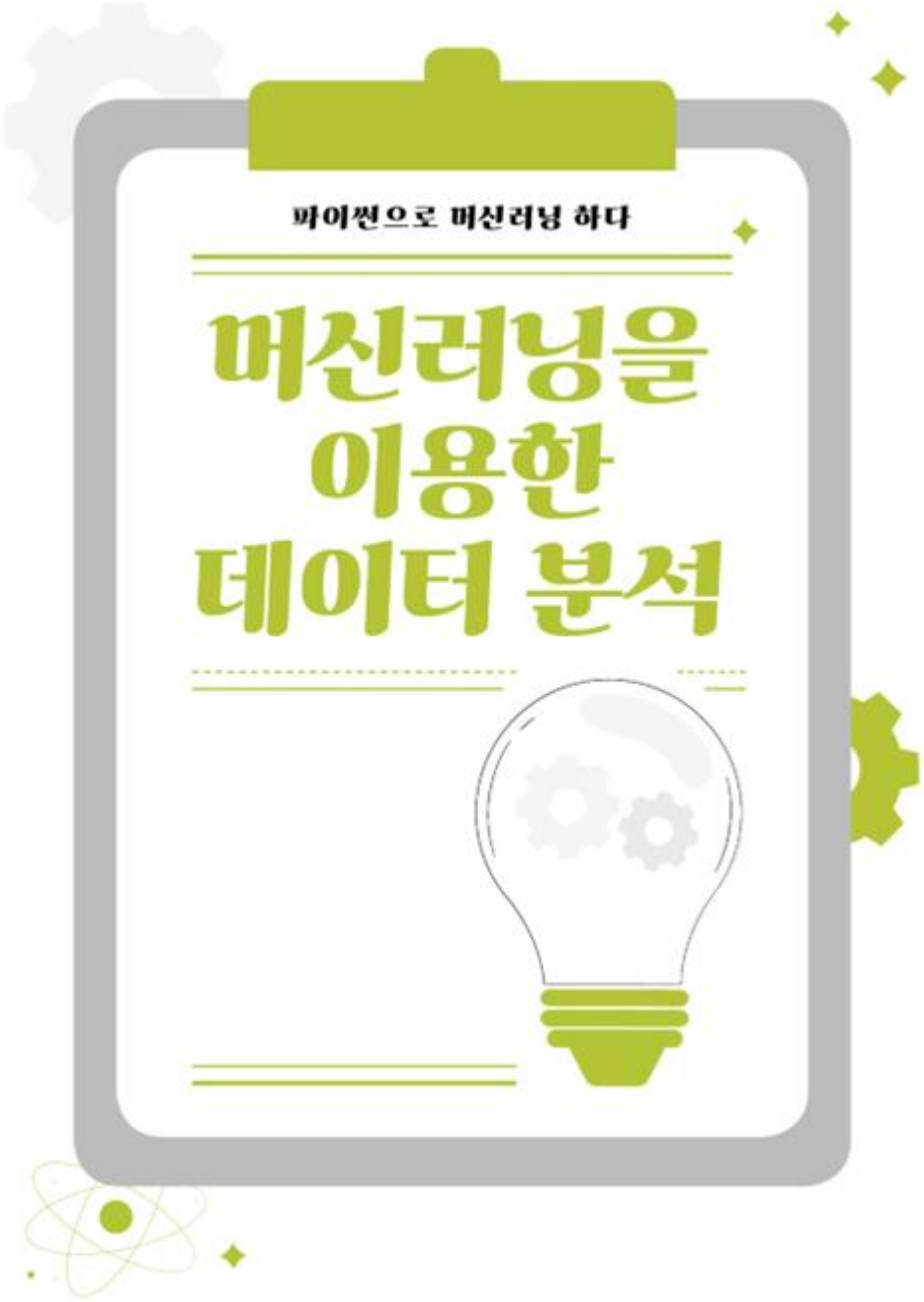
	sepal_length	sepal_width	petal_length	petal_width
0	5.0	3.0	1.4	0.2
1	5.7	NaN	1.5	NaN

```
1 imp_mostfreq = SimpleImputer(strategy='most_frequent')
2 iris_mostfreq = imp_mostfreq.fit_transform(iris_X)
```

```
1 iris_mostfreq[:5, ]
```

```
array([[5. , 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.4, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

	sepal_length	sepal_width	petal_length	petal_width
0	NaN	3.5	NaN	0.2
1	4.9	NaN	1.4	0.2
2	4.7	3.2	NaN	NaN
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

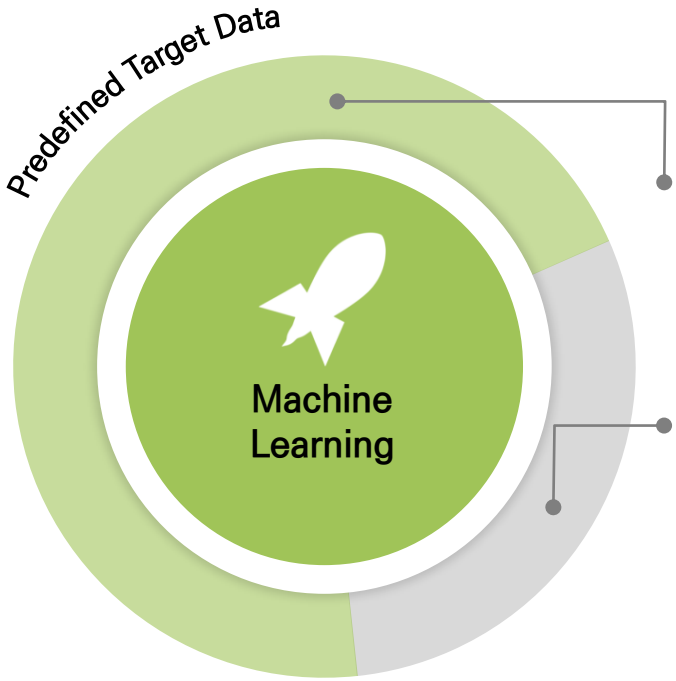


1장. 머신러닝 시작하기

4절. 단순 데이터 분리

학습용 데이터셋과 검증용 데이터셋

- 머신러닝 모형이 얼마만큼 신뢰도가 있는지 확인하기 위해서 데이터를 이용해서 예측한 후 실제의 정답과 비교
- 데이터를 학습용 데이터셋과 검증용 데이터셋으로 나누고 모형을 만들고 학습시킬 때에는 학습용 데이터셋을 사용하며, 모형을 평가하기 위해서는 검증용 데이터셋을 사용



70%

Training Set

학습 모델을 생성하기 위한 데이터 셋

30%

Test Set

검정 통계량 확인을 통하여 모델의 적합도를 확인하기 위한 데이터 셋

Training Set & Test Set

이미 분류되어 있는 데이터 셋에 대하여 트레이닝 셋과 테스트 셋으로 나누고 트레이닝 셋을 이용하여 학습 모델을 생성한다.
테스트셋과 예측 데이터를 비교한 검정 통계량 확인을 통하여 모델의 적합도를 확인한다.

random.sample()


```
1 import random
2 random.seed(10)
3 inds = random.sample(range(150), int(150*0.7))
```

```
1 import seaborn as sns
2 iris = sns.load_dataset("iris")
3 train = iris.loc[inds,:]
4 train.sort_index(inplace=True)
5 train.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
7	5.0	3.4	1.5	0.2	setosa

```
1 test = iris.loc[~iris.index.isin(train.index)]
2 test.sort_index(inplace=True)
3 test.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
6	4.6	3.4	1.4	0.3	setosa
13	4.3	3.0	1.1	0.1	setosa
16	5.4	3.9	1.3	0.4	setosa

pandas.DataFrame.sample()

- isin() 함수는 데이터프레임 또는 시리즈가 데이터를 포함하는지 여부를 알려줌
- ~는 논리 반전을 의미함(True -> False, False -> True)
- DataFrame.loc[]는 논리값을 이용해서 부분 집합을 얻을 수 있음

```
1 train = iris.sample(frac=0.7)
2 test = iris.loc[~iris.index.isin(train.index)]
```

```
1 train.shape, test.shape
```

```
((105, 5), (45, 5))
```

sklearn.model_selection.train_test_split()

학습용 데이터와 검증용 데이터를 쉽게 나눌 수 있음

- sklearn.model_selection 모듈은 Scikit-learn 0.18 버전부터 사용할 수 있습니다.

인자	타입	설명
test_size	float, int 또는 None	float인 경우 0.0과 1.0 사이여야 하며 test 데이터에 포함 할 데이터셋의 비율을 나타냅니다. int이면 test 샘플의 절대 수를 나타냅니다. None인 경우, 값은 train 크기의 보수로 설정됩니다. train_size도 None이면 0.25로 설정됩니다.
train_size	float, int 또는 None	float인 경우 0.0과 1.0 사이여야 하며 train 데이터에 포함 할 데이터셋의 비율을 나타냅니다. int이면 train 샘플의 절대 수를 나타냅니다. None인 경우, 값은 test 크기의 보수로 설정됩니다.
random_state	int, RandomState 인스턴스 또는 None	int이면 난수 생성기에서 사용하는 시드입니다. 난수 생성기로 RandomState 인스턴스를 지정할 수 있습니다. None인 경우 np.random에서 사용하는 RandomState 인스턴스가 난수 생성기로 사용됩니다.
shuffle	bool	분할하기 전에 데이터를 섞을지 여부입니다. shuffle=False인 경우 stratify=None이어야 합니다.
stratify	array like 또는 None	None이 아닌 경우 데이터를 클래스 레이블로 사용하여 계층화된 방식으로 데이터가 분할됩니다.

데이터셋 분리


```
1 import seaborn as sns
2 iris = sns.load_dataset("iris")
3 X = iris.iloc[:, :-1]
4 y = iris.iloc[:, -1]
```

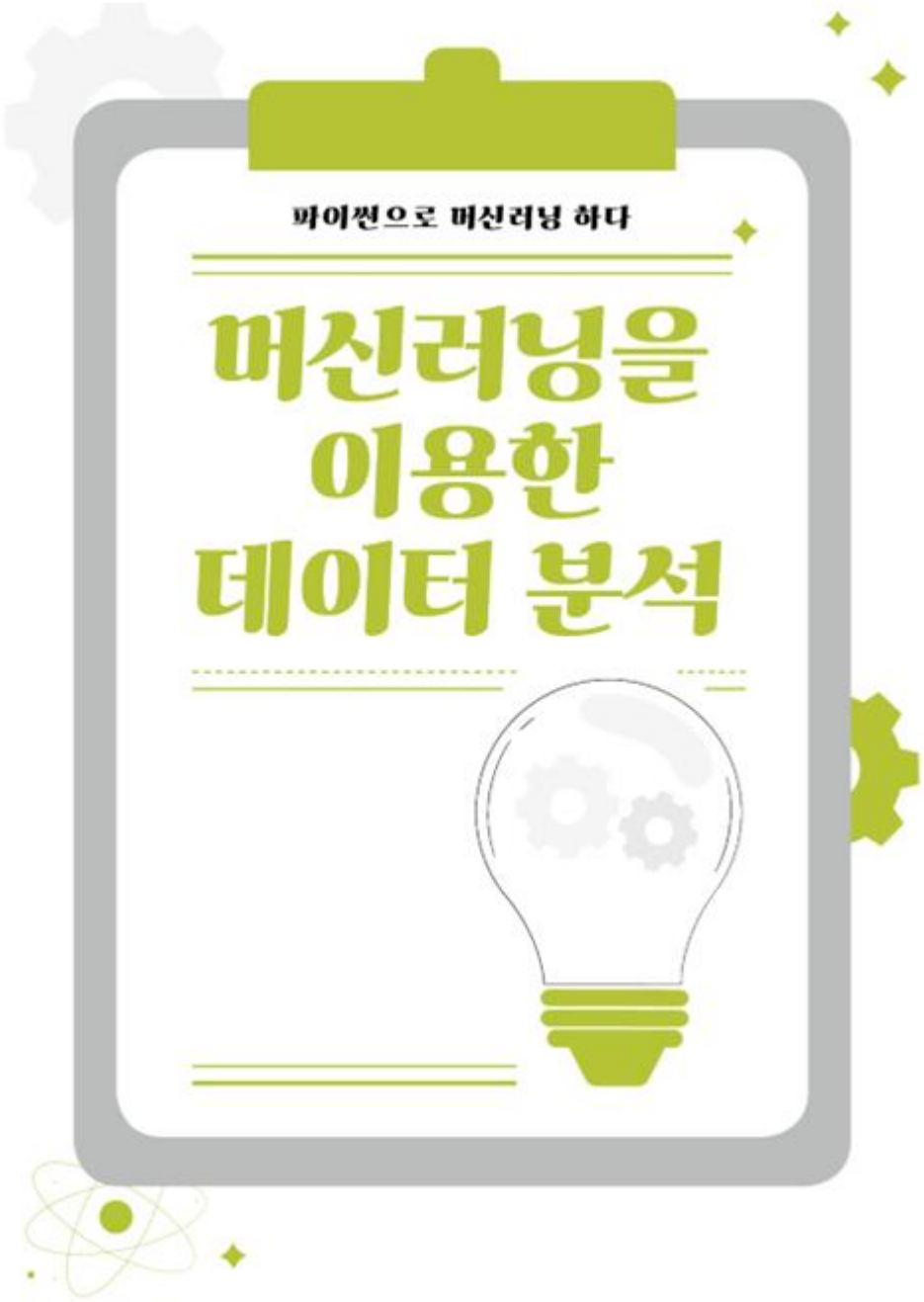
```
1 from sklearn.model_selection import train_test_split
2 train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3)
```

```
1 train_X.shape, test_X.shape, train_y.shape, test_y.shape
```

```
((105, 4), (45, 4), (105,), (45,))
```

```
1 test_y.value_counts()
```

```
setosa        19
versicolor    13
virginica     13
Name: species, dtype: int64
```



1장. 머신러닝 시작하기

5절. 모형 생성, 예측, 평가

모형 생성

- 머신러닝을 할 때 사용하는 클래스는 어떤 머신러닝 모형을 만들 것인지에 따라 다름
- Scikit-learn에서 제공하는 클래스들은 분류/회귀 모형에 따라 클래스가 정해져 있음
- 클래스를 이용해서 객체를 생성하는 것으로 모형이 만들어 짐
- 모형을 학습시키기 위해서는 훈련 데이터셋의 독립변수와 종속변수를 이용해 학습을 시킴
- 훈련 데이터셋을 이용해 학습시키기 위해 사용하는 함수는 fit()

Scikit-learn의 주요 예측 클래스

모듈명	분류 클래스	회귀 클래스
ensemble	AdaBoostClassifier	AdaBoosetRegressor
	BaggingClassifier	BaggingRegressor
	ExtraTreesClassifier	ExtraTreesRegressor
	GradientBoostingClassifier	GradientBoostingRegressor
	RandomForestClassifier	RandomForestRegressor
	VotingClassifier	VotingRegressor
linear_model		LinearRegression
		LogisticRegression
	PassiveAggressiveClassifier	PassiveAggressiveRegressor
	SGDClassifier	SGDRegressor
neural_network	MLPClassifier	MLPRegressor
neighbors	KNeighborsClassifier	KNeighborsRegressor

모형 생성


```
1 from sklearn.tree import DecisionTreeClassifier
2 dt_model = DecisionTreeClassifier(random_state=1)
3 dt_model.fit(train_X, train_y)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=1, splitter='best')
```

```
1 from sklearn.neural_network import MLPClassifier
2 mlp_model = MLPClassifier(hidden_layer_sizes=(50,50,20), max_iter=500,
3                           random_state=1)
4 mlp_model.fit(train_X, train_y)
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(50, 50, 20), learning_rate='constant',
              learning_rate_init=0.001, max_iter=500, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=1, shuffle=True, solver='adam', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

검증용 데이터(test_x)를 이용해 예측, 예측에 사용하는 함수는 predict()

```
1 dt_model.predict(test_X)
```

```
array(['setosa', 'versicolor', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'setosa', 'setosa', 'virginica',
       'versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor',
       'setosa', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'versicolor', 'versicolor', 'virginica', 'setosa', 'virginica',
       'versicolor', 'setosa', 'setosa', 'versicolor', 'virginica',
       'versicolor', 'virginica', 'versicolor', 'virginica', 'virginica',
       'setosa', 'versicolor', 'setosa', 'versicolor', 'virginica',
       'virginica', 'setosa', 'versicolor', 'virginica', 'versicolor'],
      dtype=object)
```

```
1 mlp_model.predict(test_X)
```

```
array(['setosa', 'versicolor', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'virginica', 'setosa', 'setosa', 'virginica',
       'versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor',
       'setosa', 'versicolor', 'versicolor', 'setosa', 'setosa',
       'versicolor', 'versicolor', 'versicolor', 'setosa', 'virginica',
       'versicolor', 'setosa', 'setosa', 'versicolor', 'virginica',
       'versicolor', 'virginica', 'versicolor', 'virginica', 'virginica',
       'setosa', 'versicolor', 'setosa', 'versicolor', 'virginica',
       'virginica', 'setosa', 'virginica', 'virginica', 'versicolor'],
      dtype='<U10')
```

평가

5절. 모형 생성, 예측, 평가

의사결정나무

```
1 dt_pred_y = dt_model.predict(test_X)
2 pd.crosstab(test_y, dt_pred_y)
```

col_0	setosa	versicolor	virginica
species			
setosa	14	0	0
versicolor	0	17	1
virginica	0	1	12

```
1 dt_model.score(test_X, test_y)
```

0.9555555555555556

인공신경망

```
1 mlp_pred_y = mlp_model.predict(test_X)
2 pd.crosstab(test_y, mlp_pred_y)
```

col_0	setosa	versicolor	virginica
species			
setosa	14	0	0
versicolor	0	18	0
virginica	0	0	13

```
1 mlp_model.score(test_X, test_y)
```

1.0