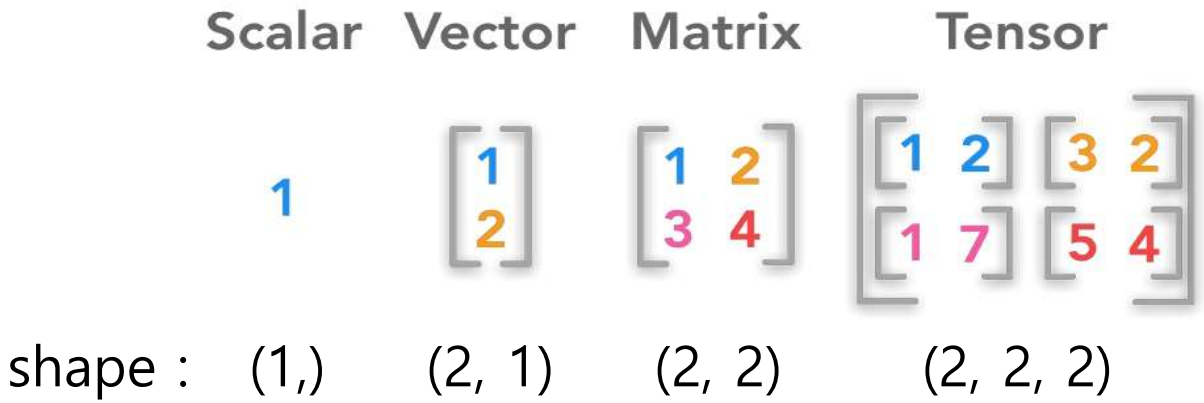
Crash Course

1. Numpy & Linear Algebra
2. Pandas
3. Matplotlib
4. Feature Scaling

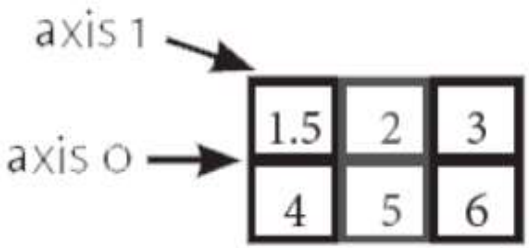
Numpy

- numpy array
  - 1 차원 - vector, 2 차원 – matrix, 3 차원 이상 - tensor
  - rank – array 의 dimension (차원)
  - shape – 각 dimension 의 size
  - dtype – tensor 의 data type

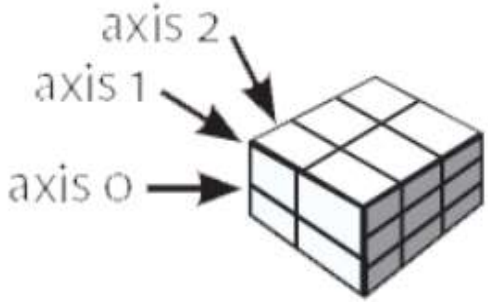


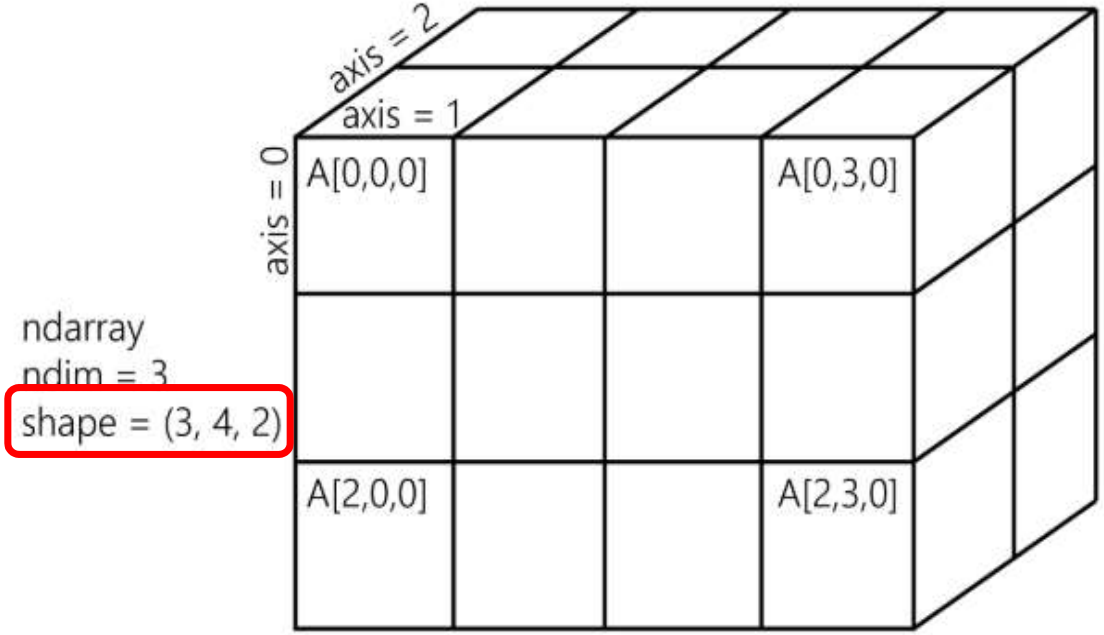
1D array

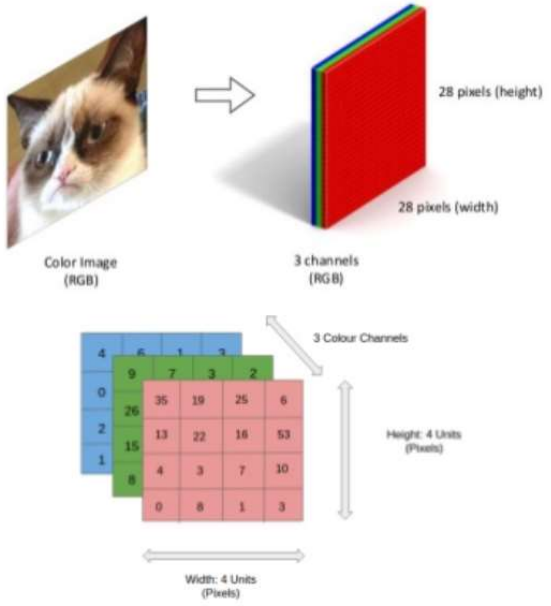
2D array



3D array





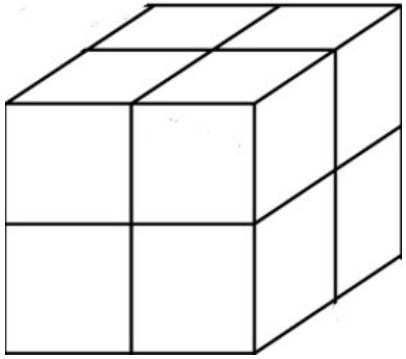


Matrix 의 numpy 표현

8	5	3
1	2	9

np.array([[8, 5, 3], [1, 2, 9]])

```
[[8 5 3]
 [1 2 9]]
```



np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

Joining & Slicing

- concatenate

```
[[1 , 2 ],
 [10, 20]]
```
+
```
[[3 , 4 ],
 [30, 40]]
```

```
[[1 , 2 ],
 [10, 20],
 [3 , 4 ],
 [30, 40]]
```
axis=0

```
[[1 , 2 , 3 , 4 ],
 [10, 20, 30, 40]]
```
axis=1

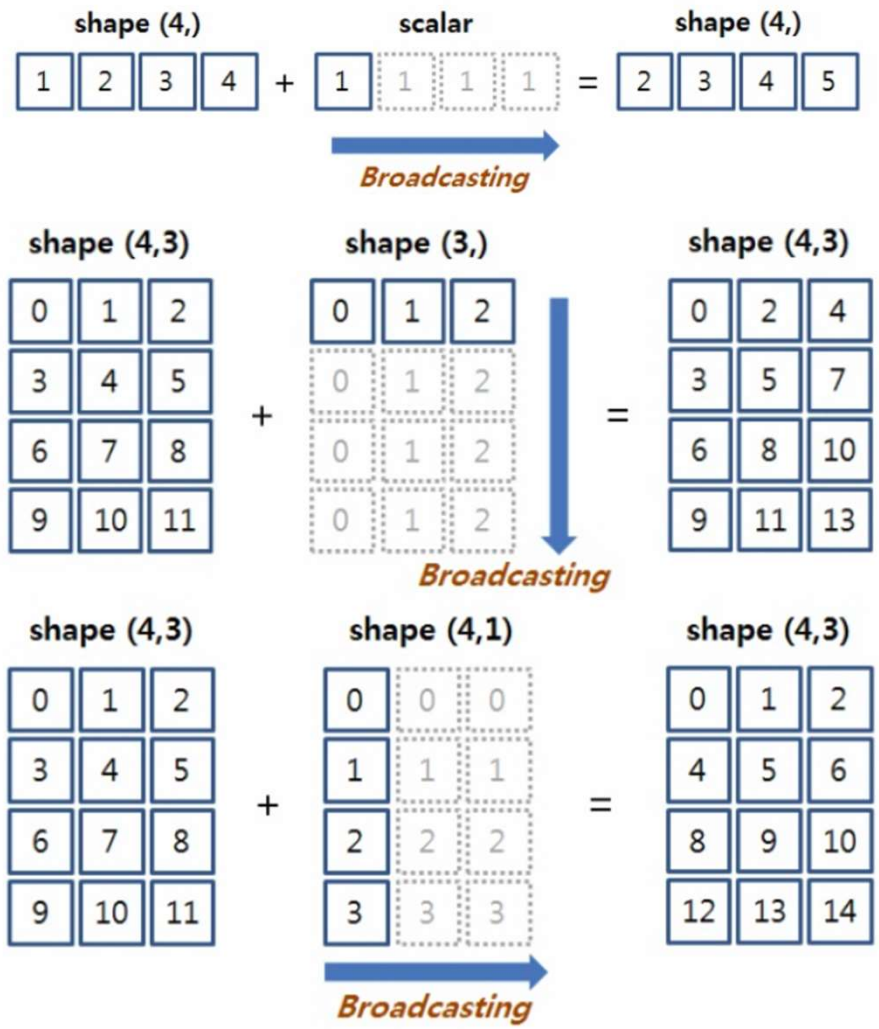
- slicing

```
a = np.array([[1,2], [3,4]])
a[1,:]
```

```
[[1 2]
 [3 4]]
```

```
[[3, 4],]
```

Broadcasting - 차원의 크기가 서로 다른 배열 간 산술연산



numpy array 간의 연산

- np.add(a, b) – 두개의 tensor 를 element-wise 더함

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} + \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = \begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}$$

- np.sub(a, b), np.multiply(a, b), np.divide(a, b)

- np.matMul(a, b) - dot product of two matrices

$$[1 \quad 2] \times \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} = [1*1+2*3 \quad 1*2+2*4] = [7 \quad 10]$$

shape : (1, 2) x (2, 2) → (1, 2)

$$\begin{bmatrix} a & b\\ c & d\end{bmatrix}\begin{bmatrix} e & f\\ g & h\end{bmatrix} = \begin{bmatrix} ae+bg & af+bh\\ ce+dg & cf+dh\end{bmatrix}$$

Transpose, Reshape

- tf.transpose() – row 와 column 을 전치

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.\text{transpose}() \rightarrow \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

- tf.reshape([m, n])

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}.\text{reshape}([3, 2]) \rightarrow \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Pandas

- Dataframe - R 을 모방하여 구현

- Tabular 형식의 Data 처리 (Excel 대체 가능)

	A	B	C	D	E	F	G	H	I
1	Order Date	OrderID	Salesperson	UK Units	UK Order Amt	USA Units	USA Order Amt	Total Units	Total Order Amt
2	1/01/2011	10392	Fuller			13	1440	13	1440
3	2/01/2011	10397	Gloucester	17	716.72			17	716.72
4	2/01/2011	10771	Bromley	18	344			18	344
5	3/01/2011	10393	Finchley			16	2556.95	16	2556.95
6	3/01/2011	10394	Finchley			10	442	10	442
7	3/01/2011	10395	Gillingham	9	2122.92			9	2122.92
8	6/01/2011	10396	Finchley			7	1903.8	7	1903.8
9	8/01/2011	10399	Callahan			17	1765.6	17	1765.6
10	8/01/2011	10404	Fuller			7	1591.25	7	1591.25
11	9/01/2011	10398	Fuller			11	2505.6	11	2505.6
12	9/01/2011	10403	Coghill	18	855.01			18	855.01
13	10/01/2011	10401	Finchley			7	3868.6	7	3868.6

Matplotlib

matplotlib
Version 3.1.1

Fork me on GitHub

home | examples | tutorials | API | contents » User's Guide »

previous | next | modules | index

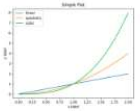
Tutorials

This page contains more in-depth guides for using Matplotlib. It is broken up into beginner, intermediate, and advanced sections, as well as sections covering specific topics.

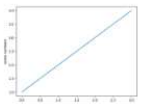
For shorter examples, see our examples page. You can also find external resources and a FAQ in our user guide.

Introductory

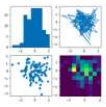
These tutorials cover the basics of creating visualizations with Matplotlib, as well as some best-practices in using the package effectively.



Usage Guide

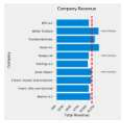


Pyplot tutorial

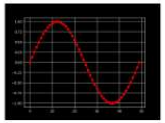


Sample plots in Matplotlib

Image tutorial



The Lifecycle of a Plot



Customizing Matplotlib with style sheets and rcParams

Quick search

Go

Table of Contents

Tutorials

- Introductory
- Intermediate
- Advanced
- Colors
- Text
- Toolkits

Related Topics

Documentation overview

- User's Guide
  - Previous: Installing
  - Next: Usage Guide

Show Page Source

Feature Scaling

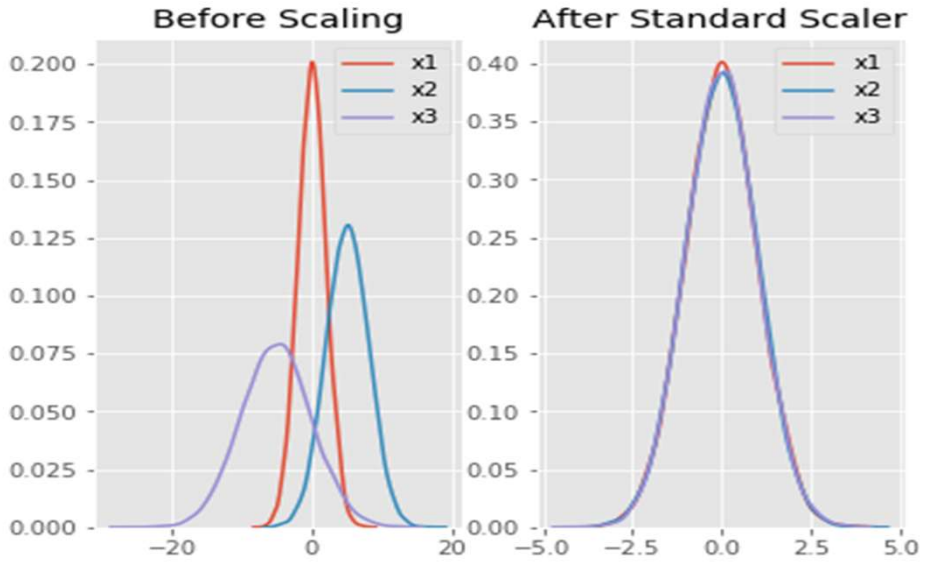
- Raw data 를 전 처리하여 input data 의 구간을 표준화

- Standard Scaling :

  $z = (x - \mu) / s$ (μ : 평균, s : 표준편차)

- Minmax Scaling :
  - 0~1 사이값

$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$



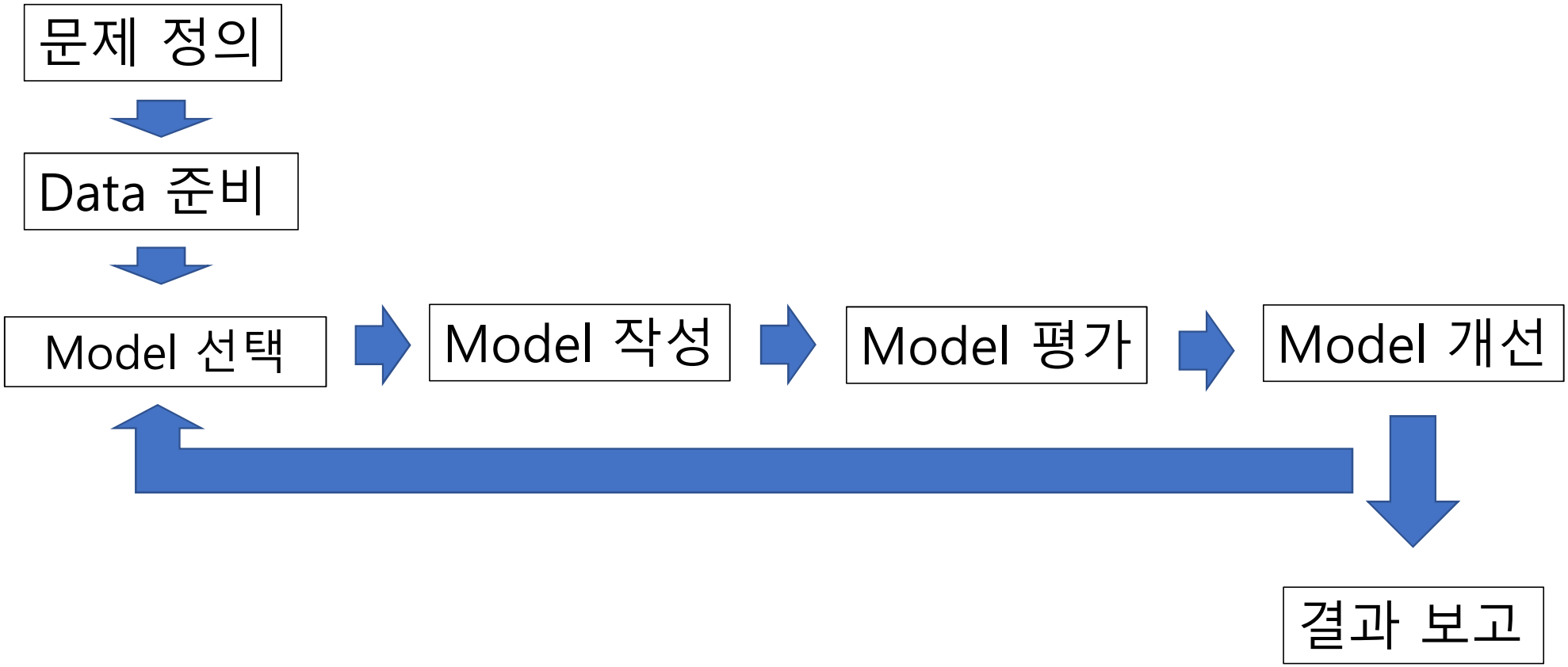
실습: sklearn 을 이용한 feature scaling

- Simple Feature Scaling
- Standard Scaling (z-score)
- MinMax Scaling

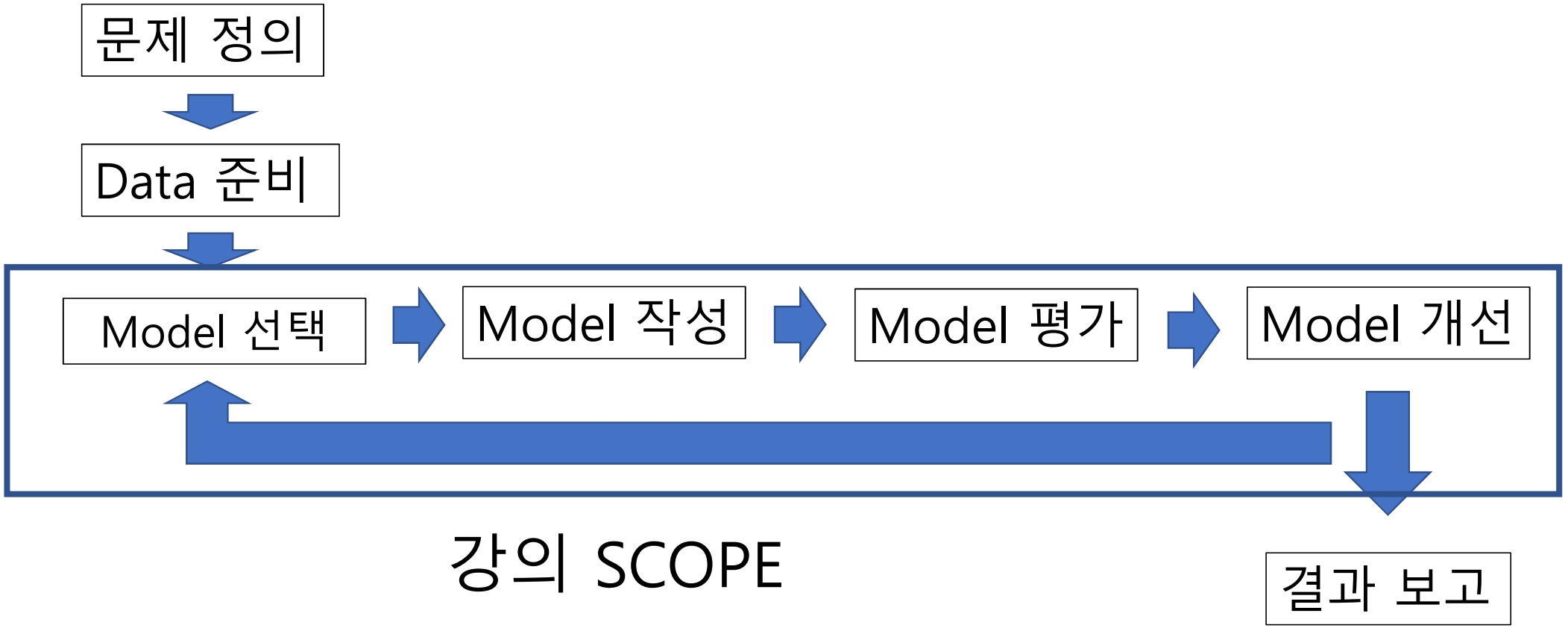
Machine Learning

End-to-End Process

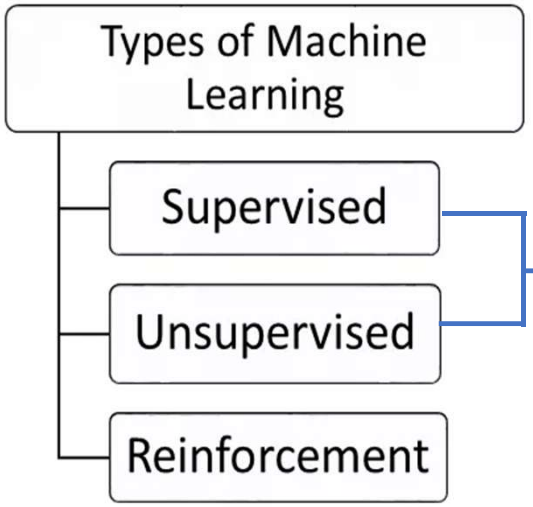
End-to-End of Machine Learning



End-to-End of Machine Learning

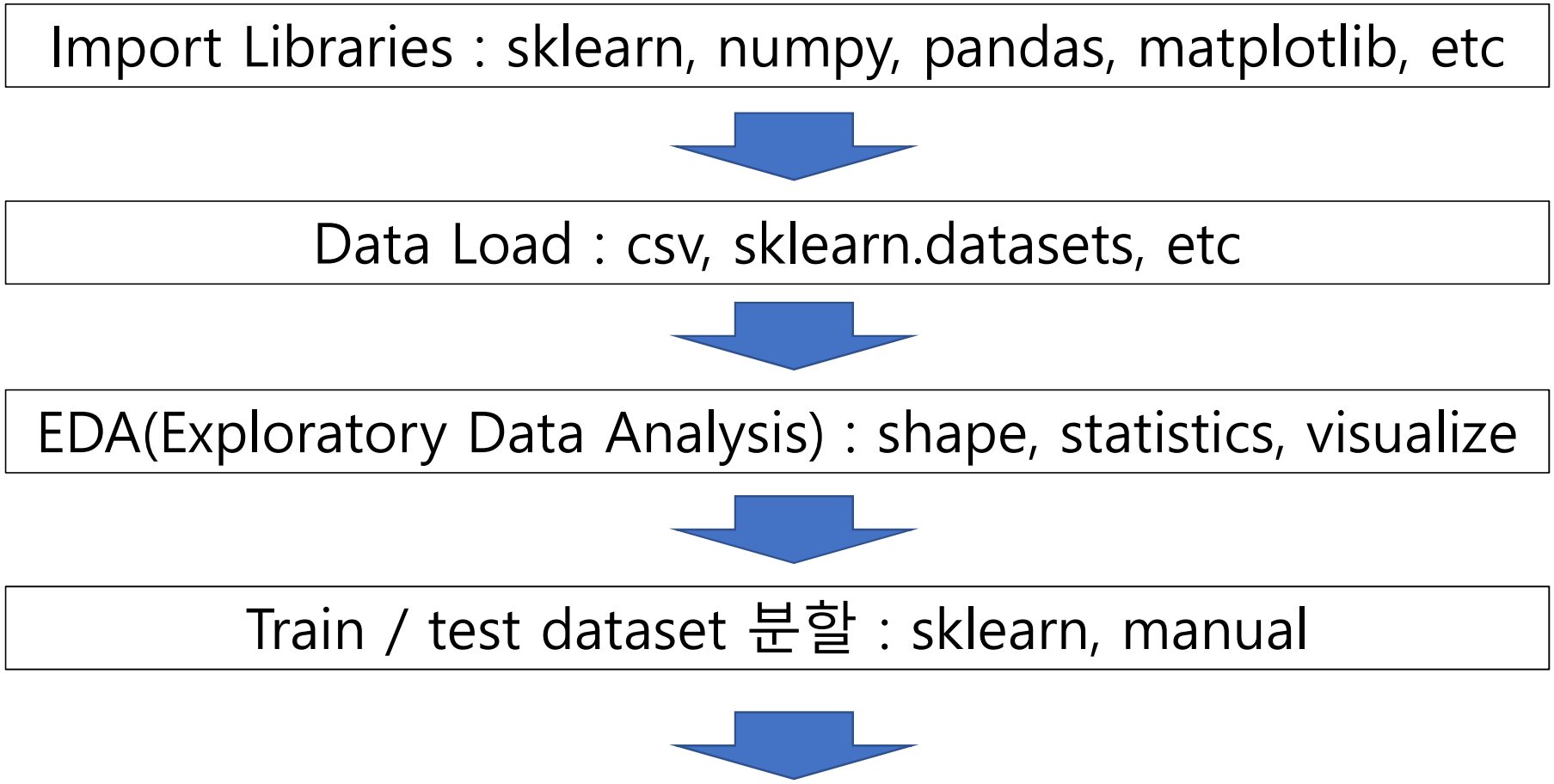


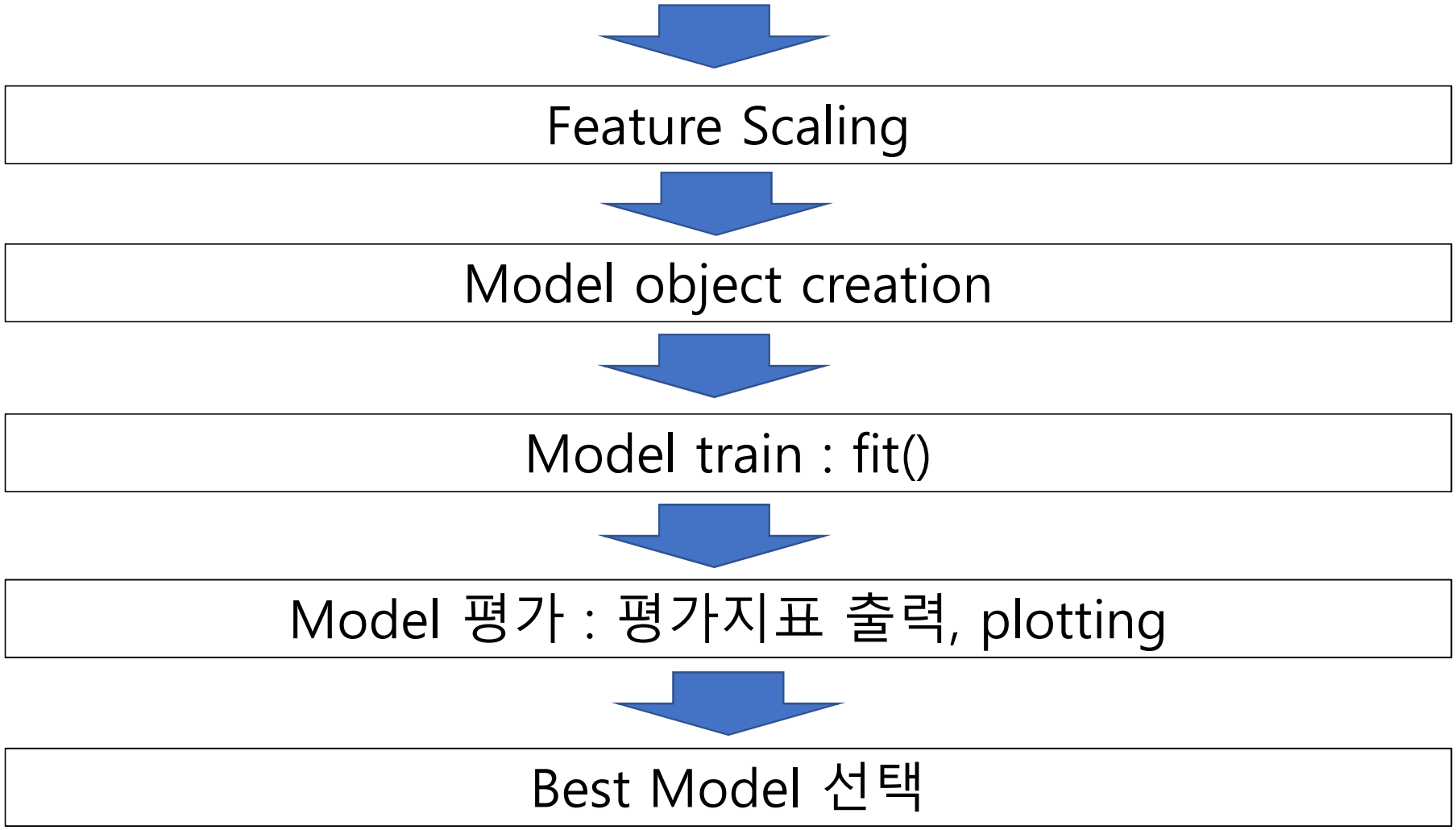
Model 선택



전통적 ML	• Linear & Logistic Regression • KNN • Decision Tree • SVM • Ensembles • K-Means Clustering • PCA
Deep Learning (Neural Network)	• FC • CNN • RNN • Auto Encoder • GAN

Model 작성 순서


Import Libraries : sklearn, numpy, pandas, matplotlib, etc
Data Load : csv, sklearn.datasets, etc
EDA(Exploratory Data Analysis) : shape, statistics, visualize
Train / test dataset 분할 : sklearn, manual

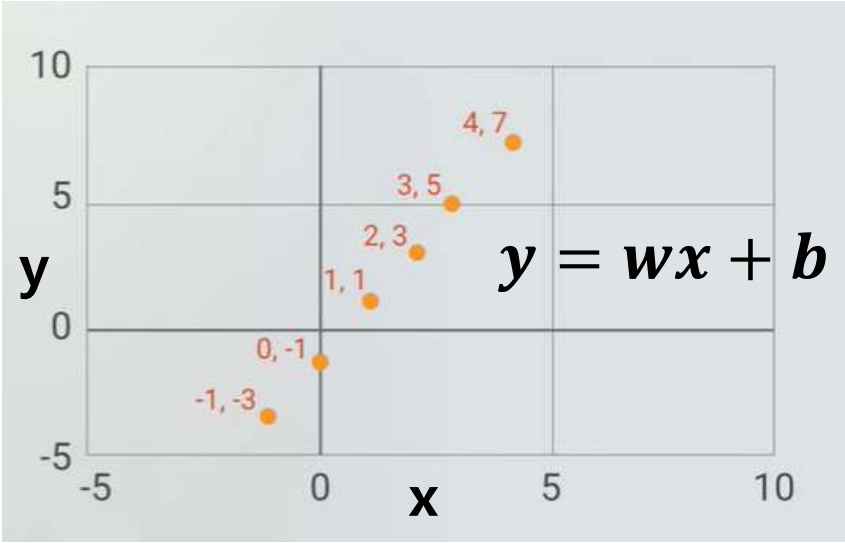

Feature Scaling
Model object creation
Model train : fit()
Model 평가 : 평가지표 출력, plotting
Best Model 선택

Linear Regression

1. Univariate Linear Regression → 변수가 한 개 (ex. 혈압)

2. Multivariate Linear Regression
   → 변수가 여러 개 (ex. 혈압, 체질량, 몸무게)

3. Polynomial Regression → 변수의 차수(degree) 증가



- x, y 가 주어지고
  w, b 가 미지수

- **w, b 를 infer (추정)**

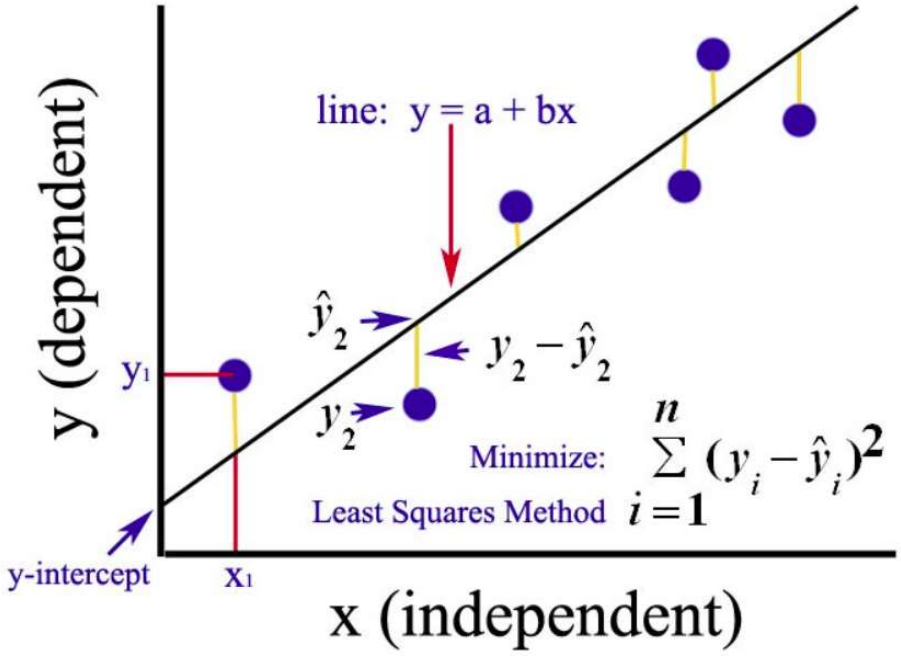
1. Univariate Linear Regression (단변수선형회귀)

y = a + bx → Hypothesis

- OLS (Ordinary Least Squares, 최소자승법)

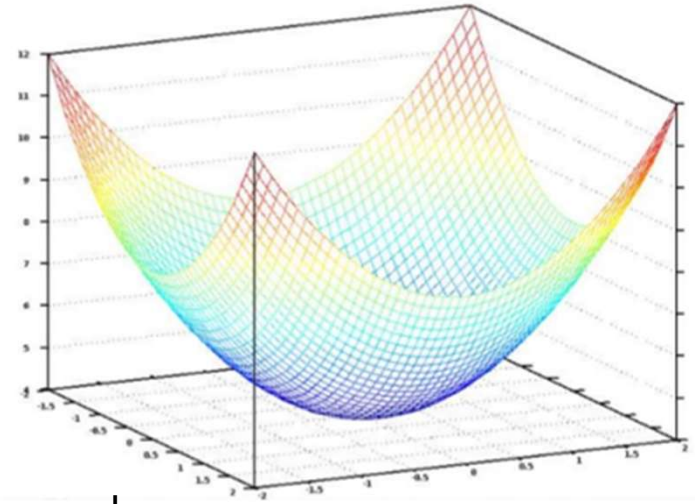
$$Minimize \sum_{i=1}^{n} (true - prediction)^2$$

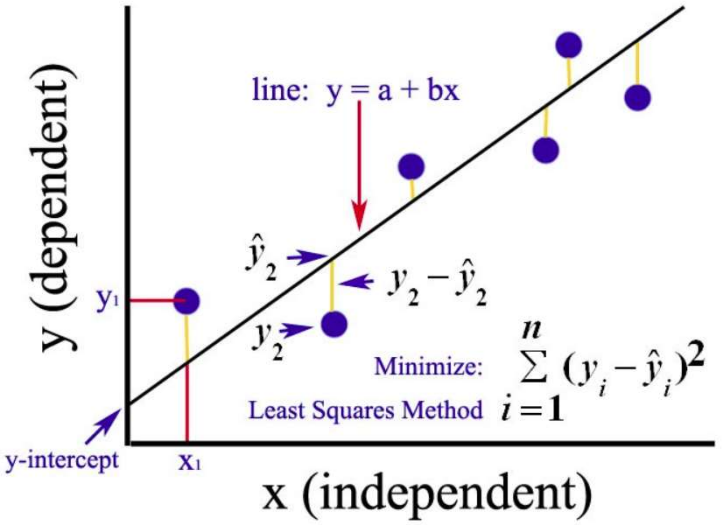
Cost Function



Cost Function - Linear Regression

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$





$$y = \theta x + b$$

MSE 를 최소화 하는
θ 와 b 를 optimize

How to solve Cost Function of θ ?

- Normal Equation (정규방정식) 이용
  - $\hat{\boldsymbol{\theta}} = \left(\boldsymbol{X}^{\boldsymbol{T}} \cdot \boldsymbol{X}\right)^{-1} \cdot \boldsymbol{X}^{\boldsymbol{T}} \cdot \boldsymbol{y}$ ($\hat{\theta}$: 기울기/절편, X: training data, y: target data)
  - Data 양이 많으면 사용할 수 없음 (시간, memory)

- **Gradient Descent** : 미분을 이용한 점근적 방법 ➔ Machine Learning 의 기본

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(\theta X^{(i)} - y^{(i)}\right)^2$$

$$\Rightarrow \quad \frac{\partial f}{\partial \theta} = \frac{2}{m}\sum_{i=1}^{m}\left(\theta X^{(i)} - y^{(i)}\right) \cdot X^{(i)}$$

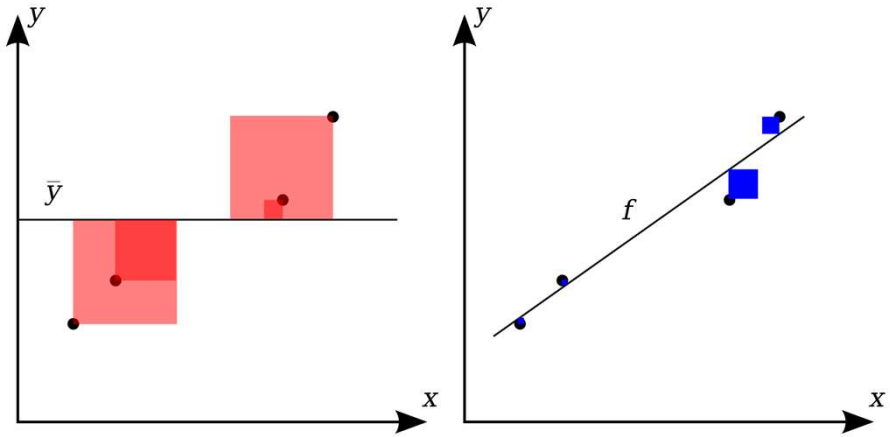
- sklean method 이용 : **sklearn.linear_model.LinearRegression()**

- 평가기준 2 : R2 score (결정계수)
  sklearn.metrics.r2_score

* 결정계수(R2) – 회귀식의 정확도 측정
  $0 \leq R2 \leq 1$

  $R2 = 1 - SSE / SST$

$$\left(\frac{Sum\ of\ Square\ Error}{Sum\ of\ Square\ Total}\right)$$



R2 ≈ 0　　R2 ≈ 1

- 체질량지수(bmi) 하나만으로 univariate linear regression
- 체질량지수(bmi) 와 혈압(bp) 두가지 변수로 multivariate linear regression

2. Multivariate Linear Regression (다변수선형회귀)

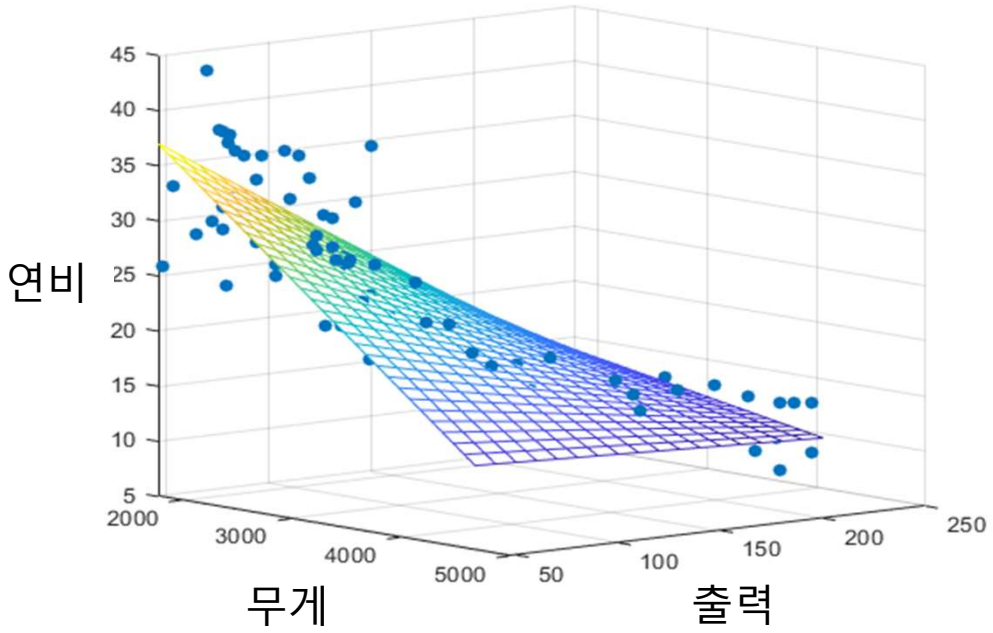
$$Yhat = \theta 0 + \theta 1X1 + \theta 2X2 + \theta 3X3 + ... + \theta nXn$$

$$Yhat = \theta X$$

sklearn.linear_model.LinearRegression()

θ = coef_

$\theta 0$ = intercept_



3. Polynomial Regression (다항회귀)

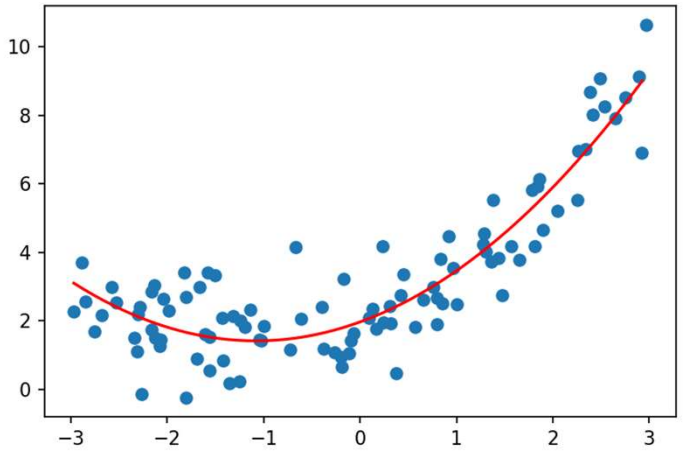
$$y = b_0 + b_1 x^3 + b_2 x^2 + b_3 x$$

sklearn.preprocessing.PolynomialFeatures

데이터 변환 → 다항식에 해당하는 features 추가 (degree 지정)

sklearn.linear_model.LinearRegression()

b1 = coef_
b0 = intercept_



실습: 당뇨병 data 를 이용한 선형회귀

- Dataset : sklearn.datasets.load_diabetes()
- Feature : 나이, 성별, 체질량지수, 혈압, 6가지 혈청 수치 → scaling 되어 있음
- Target : 1년 뒤 측정한 당뇨병의 진행률
- Model : OLS (Ordinary Least Squares)
- 평가기준 1 : MSE (Mean Squared Error)
  sklearn.metrics.mean_squared_error

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

실습: 비선형 다항회귀 (Polynomial Regression)

- Data : X = np.random.rand() 를 이용한 random data 생성

- Data 변환 : poly_features = PolynomialFeatures(degree=2, include_bias=False)
  poly_features.fit_transform(X)

- Target : y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)

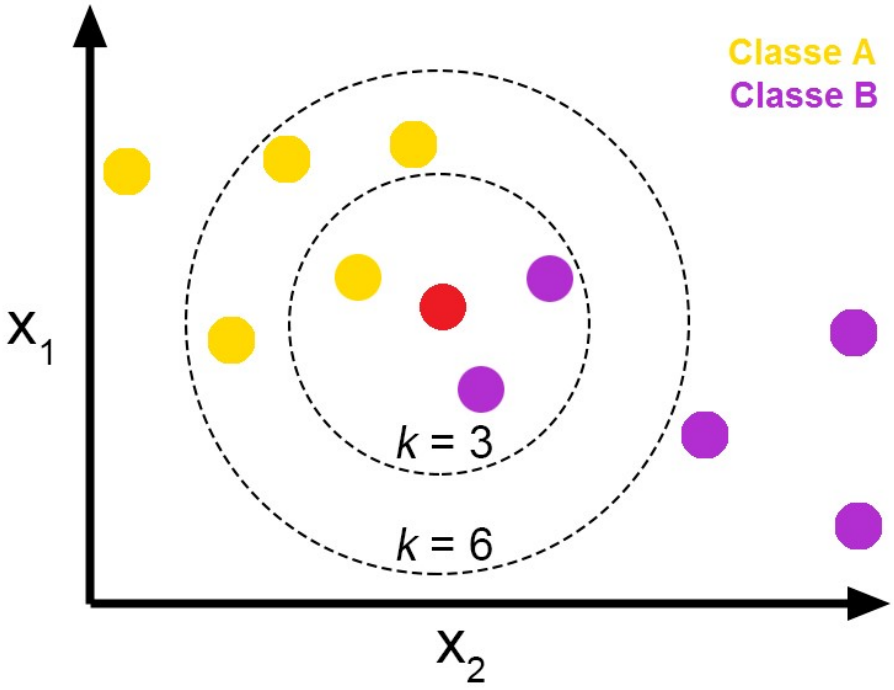
- 평가 : matplotlib 을 이용한 visualization
  coef_[0][1] – 2 차항의 계수 (coefficient)
  coef_[0][0] – 1 차항의 계수
  intercept_ - 절편

Classification

K-Nearest Neighbors

KNN (K-Nearest Neighbors, K 최근접 이웃)

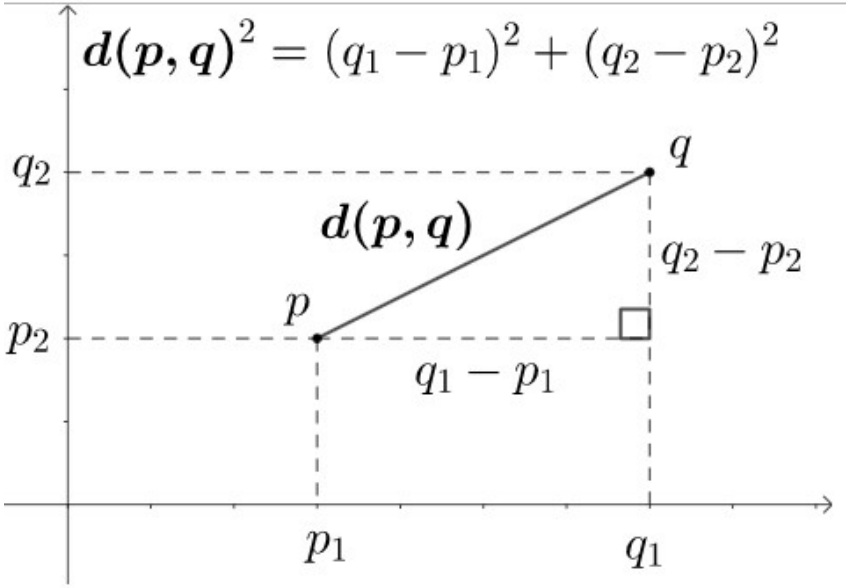
- 다른 observation (관측치, X data) 과의 유사성에 따라 분류 (classify)
- 서로 가까이 있는 data 들을 "이웃" (neighbor) 이라고 부른다.
- 가까이 있는 이웃의 label 들 중 가장 많은 것을 unknown case 의 prediction 으로 응답한다.
- 장점 : simple and easy to implement
- 단점 : dataset 이 커지면 slow.
  outlier/missing value 의 영향이 크다.



KNN 알고리즘

1. K 값을 선택한다.

2. Unknown case 와 모든 data point 간의 거리를 계산한다.



$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

3. Training dataset 에서 unknown data point 와 가장 가까이 있는 K 개의 관측치(observation) 을 선택한다.

4. K 개의 nearest neighbors 의 label 중 가장 많은 것을 unknown data point 의 **class** 로 **분류**한다 (→ classification)

   K 개의 nearest neighbors 의 label value 의 평균을 predicted **value** 로 **계산** 한다 (→ regression)

실습 :

1. sklearn 에서 제공하는 iris (붓꽃) 분류 dataset 사용 :

   꽃잎의 각 부분의 너비와 길이 등을 측정한 데이터이며
   150개의 레코드로 구성

2. Dataset 의 형식:

   data – 독립변수 (ndarray)

   target – 종속변수 (ndarray)

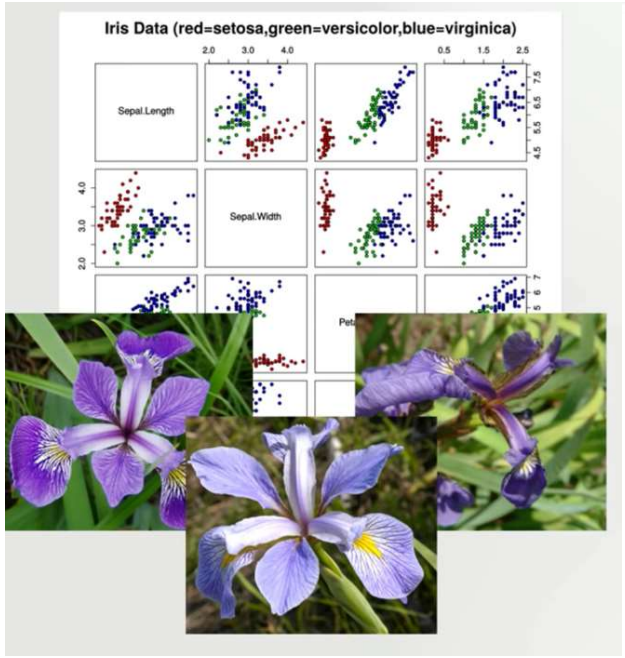
   feature_names – 독립변수 이름 list

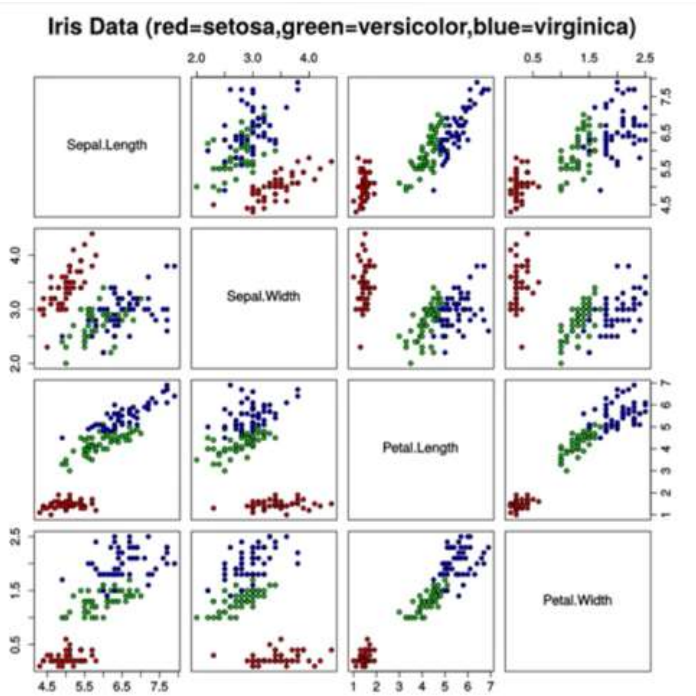
   target_names : 종속변수 이름 list

   DESCR – 자료에 대한 설명

3. Data 의 내용 :

Sepal Length : 꽃받침 길이
Sepal Width : 꽃받침 너비
Petal Length : 꽃잎 길이
Petal Width : 꽃잎 너비





4. Neighbor 개수 (K = 15)

5. 거리에 따른 가중치 parameter

uniform – neighbor 간의 거리 무시

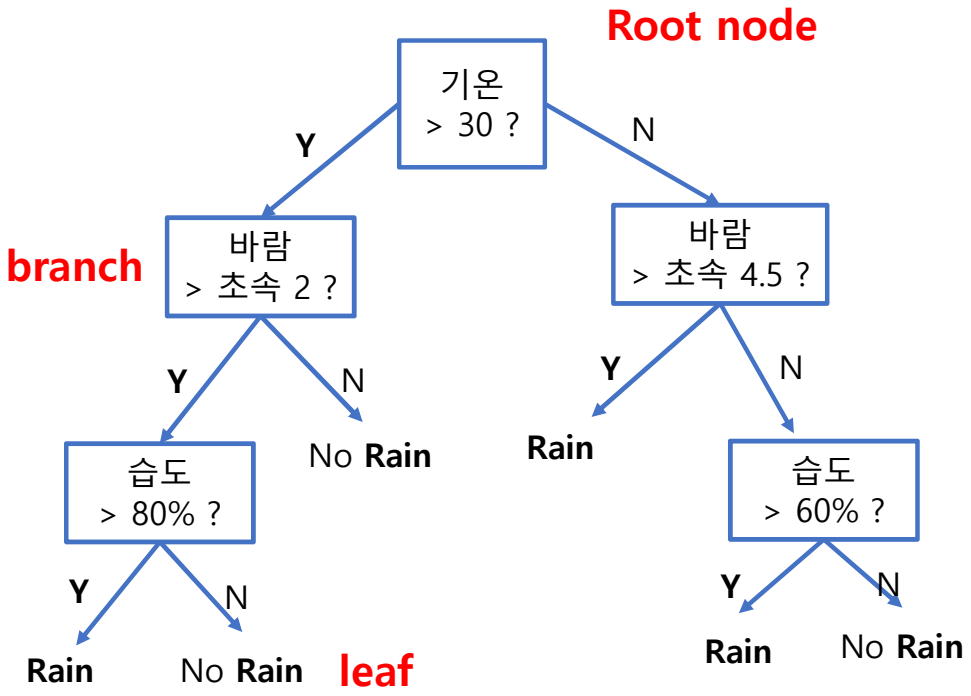
distance – 가까운 neighbor 에 더 높은 가중치 부여

6. Data 들의 실제 class 와 KNN classifier 가 예측한 class 를 color 로 구분하여 plot

Decision Tree

Decision Tree (결정나무)

- 모든 가능한 결정 경로(Decision Path)를 tree 형태로 구성
- 각 node 는 test 를 의미
- 각 branch 는 test 의 결과에 해당
- 각 leaf node 는 classification 에 해당
- 장점 : **white-box model**
  data preprocessing 불필요
- 단점 : overfitting 되기 쉽다.



Decision Tree 알고리즘의 종류

1. ID3 – 기본적 알고리즘. 정보이득(Information Gain) 을 이용한 트리 구성

2. CART (Classification and Regression Tree)

    – Gini 불순도에 기반한 트리 구성

3. C4.5, C5.0 – ID3 개선

4. 기타 – CHAID, MARS

엔트로피 (Entropy), 정보 이득(information gain)

- 엔트로피(Entropy) - 주어진 데이터 집합의 혼잡도. 즉, 우리가 가지고 있지 않은 정보의 양을 의미.

- 주어진 데이터 집합에서 서로 다른 종류의 레코드들이 섞여 있으면 엔트로피가 높고, 같은 종류의 레코드들이 섞여 있으면 엔트로피가 낮음.

- 우리가 시스템에 대해 알게 될수록 시스템의 엔트로피는 감소. 예를 들어 데이터의 통계를 알게 되면 엔트로피는 감소. 이것을 정보 이득(information gain)이라고 한다.

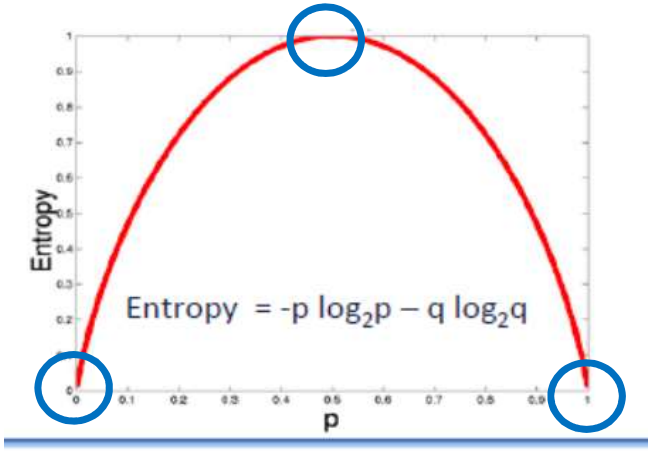
- 엔트로피 값은 0에서 1사이의 값. 가장 혼합도가 높은 상태의 값이 1이고, 반대는 0

- Decision Tree 에서는 엔트로피가 높은 상태에서 낮은 상태가 되도록 데이터를 특정 조건을 찾아 나무 모양으로 구분해 나감.

$$\boldsymbol{Entropy} = -\sum_{i=1}^{m} \boldsymbol{p_i} \log_2(\boldsymbol{p_i}) \ , \ \boldsymbol{p_i} = \frac{\boldsymbol{freq(C_i,S)}}{|\boldsymbol{S}|}$$

(S: 주어진 데이터들의 집합, C: 레코드(클래스) 값들의 집합, freq(Ci,S): S에서 Ci에 속하는 레코드의 수, |S|: 주어진 데이터들의 집합의 데이터 개수)

p = 0.5, Entropy = 1

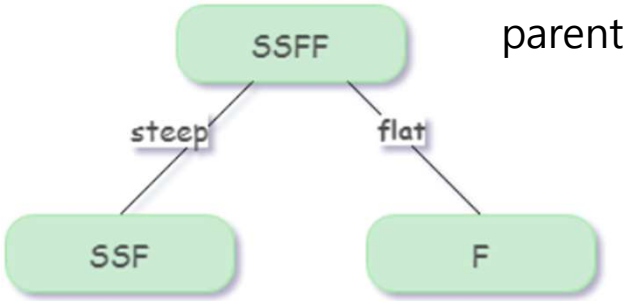


Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

- Information Gain = Entropy(Parent) – (weight) * Entropy(Child)

ex)

Feature			Label
경사도	노면상태	속도제한	속도
steep	bumpy	Yes	slow
steep	smooth	Yes	slow
flat	bumpy	No	fast
steep	smooth	No	fast



- "경사도"의 information gain 계산:

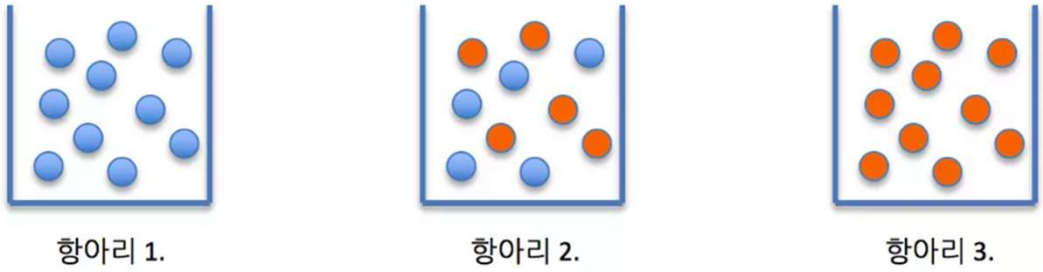
Entropy(Parent) = - {0.5 log2(0.5) + 0.5 log2(0.5)} = 1

Entropy(Child/steep) = - {0.667 log2(0.667) + 0.334 log2(0.334)} = 0.918
Entropy(Child/flat) = - {0 + 1 log2(1)} = 0
Entropy(가중평균) = ¾*0.918 + ¼*0 = 0.688

- Information Gain = 1 – 0.688 = 0.312

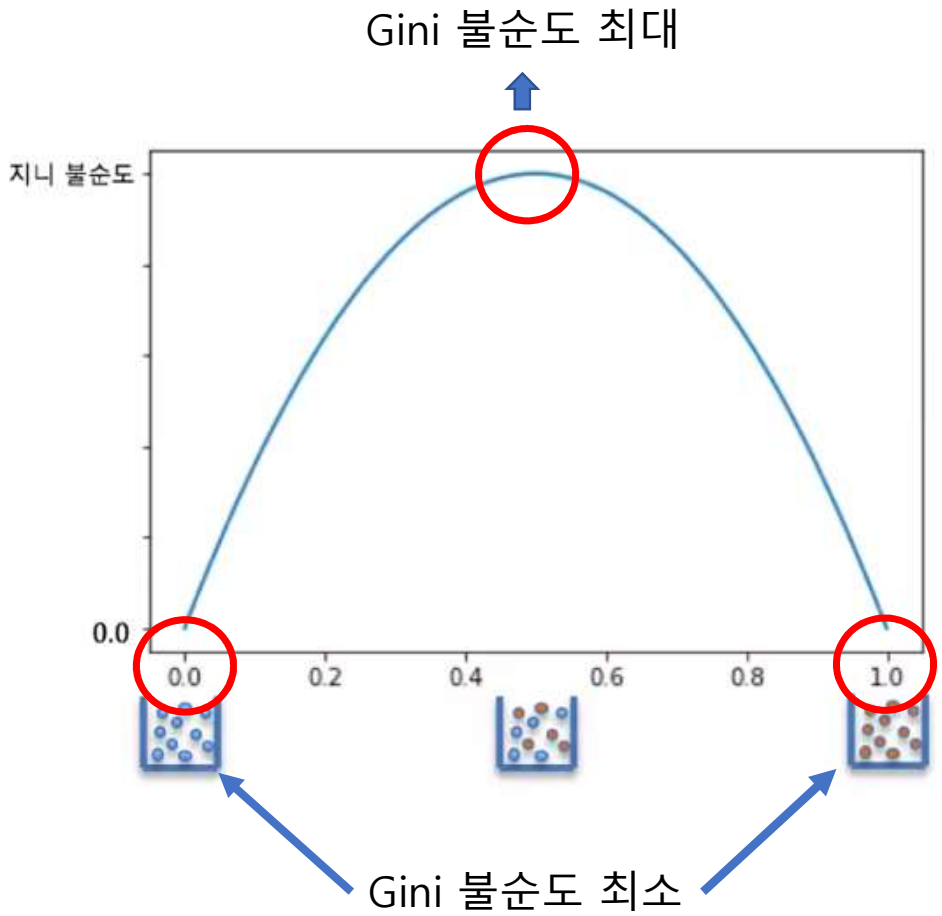
- Gini Impurity (지니불순도)

  - CART 알고리즘에서 사용



지니불순도를 최소화 하는 방향으로 Tree 를 구성

$$\sum_{j=1}^{J} p_j(1-p_j)$$



Decision Tree 알고리즘 (ID3)

1. Initial open node 를 생성하고 모든 instance 를 open node 에 넣는다.

2. Open node 가 없어질 때까지 loop
   - 분할할 open node 선택
   - information gain 이 최대인 attribute(feature) 선택
   - 선택된 attribute 의 class (Y, N) 별로 instance sort
   - sort 된 item 으로 새로운 branch 생성
   - sort 된 item 이 모두 하나의 class 인 경우 leaf node close

실습 : Decision Tree 작성 및 시각화

1. KNN 에서 사용하였던 Iris data 사용

2. Tree 의 max_depth = 2, None 으로 변경하여 test/비교

3. graphviz 를 이용한 visualization

Graphviz 설치 방법 (or Google Collab 사용)

1. Install graphviz windows (https://graphviz.gitlab.io/download/)

   Stable 2.38 Windows install packages ➔ graphviz-2.38.msi 설치

2. 환경변수 setting

   C:₩Program Files (x86)₩Graphviz2.38₩bin 경로 copy

   내PC – 속성 – 고급시스템설정 – 환경변수 – Path – 편집
   – 새로만들기 – 경로 paste

3. anaconda prompt 에서 pydotplus install

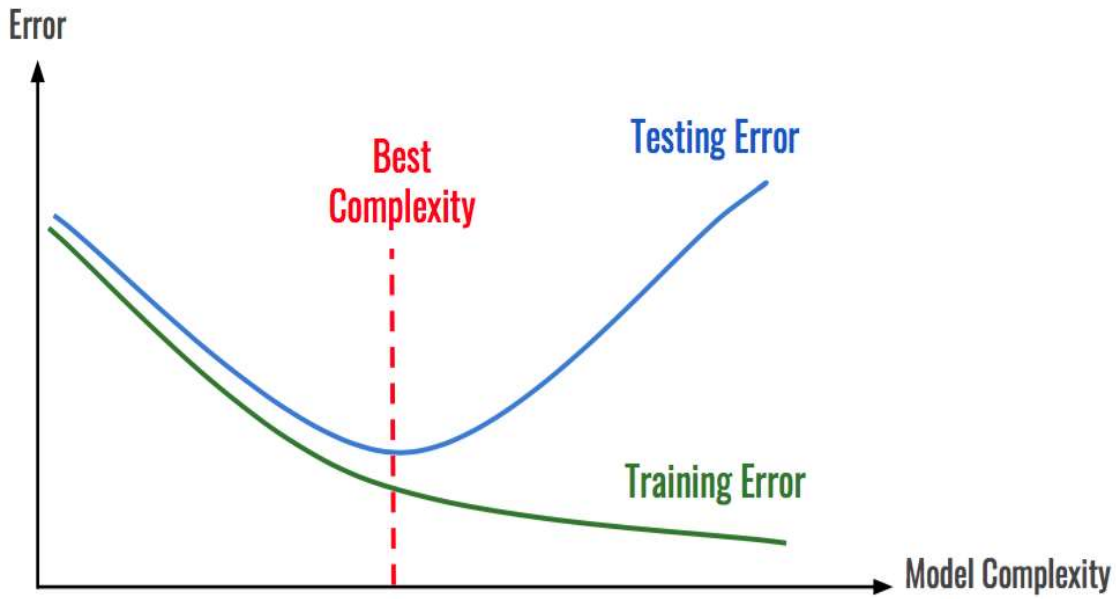
   >conda install python-graphviz
   >pip install pydotplus

Training/Test/Evaluation

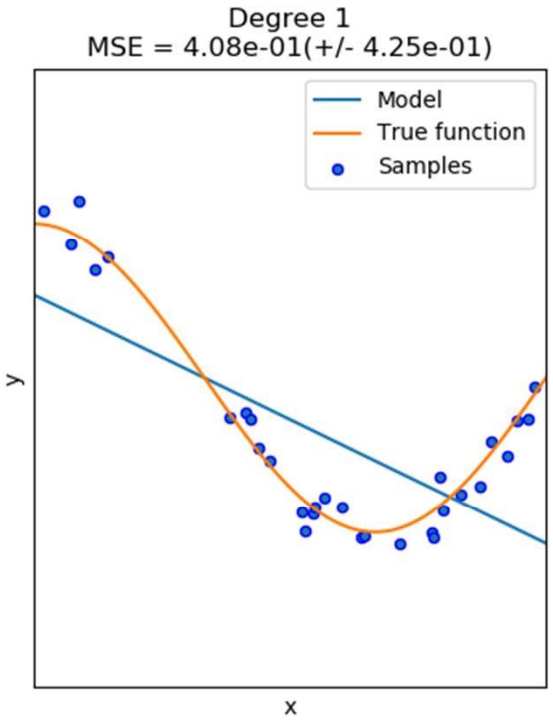
Overfitting(과적합) 과 Underfitting(과소적합)

- Training Data 에 비해 Test Data 의 ERROR 율이 높게 나타나는 경우 이를 과적합 (Overfitting) 이라고 한다.

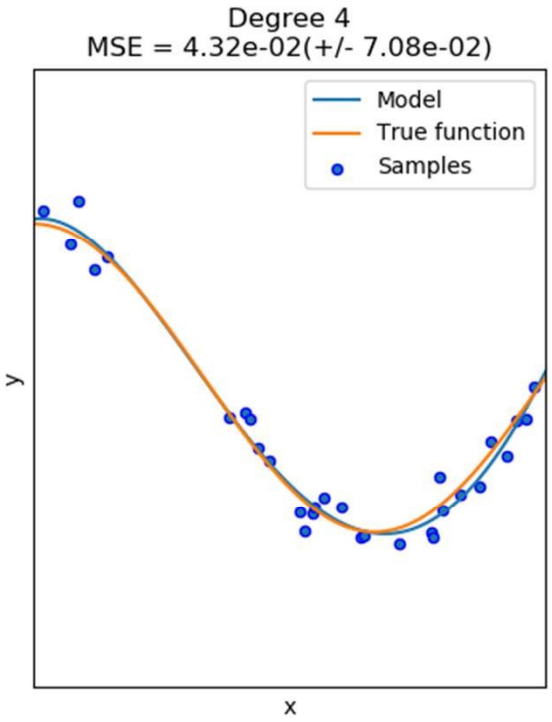
- 반대로 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못하는 경우 과소적합 (Underfitting) 이라고 한다.



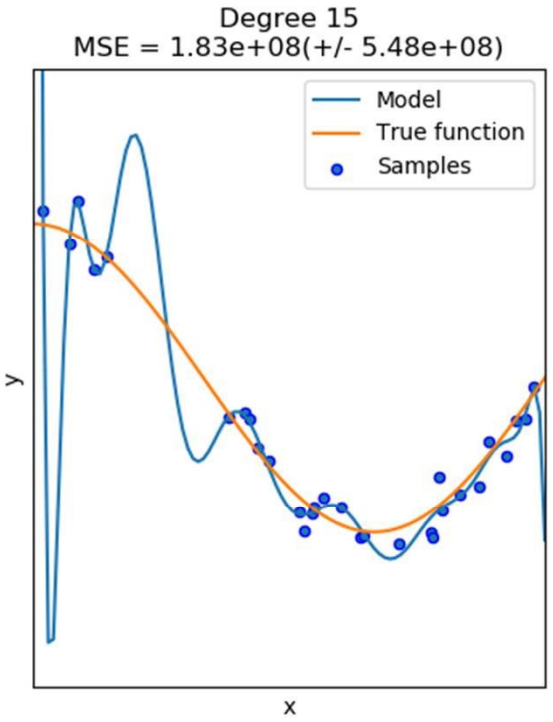
Underfitting
(과소적합)



모델이 너무 단순
(data 의 중요 부분을 놓침)
High Bias Model



Overfitting
(과대적합)



모델이 너무 복잡
(실제와 무관한 noise 까지 학습)
High Variance Model

Machine Learning 의 Source of Error

- 세가지의 Error Source

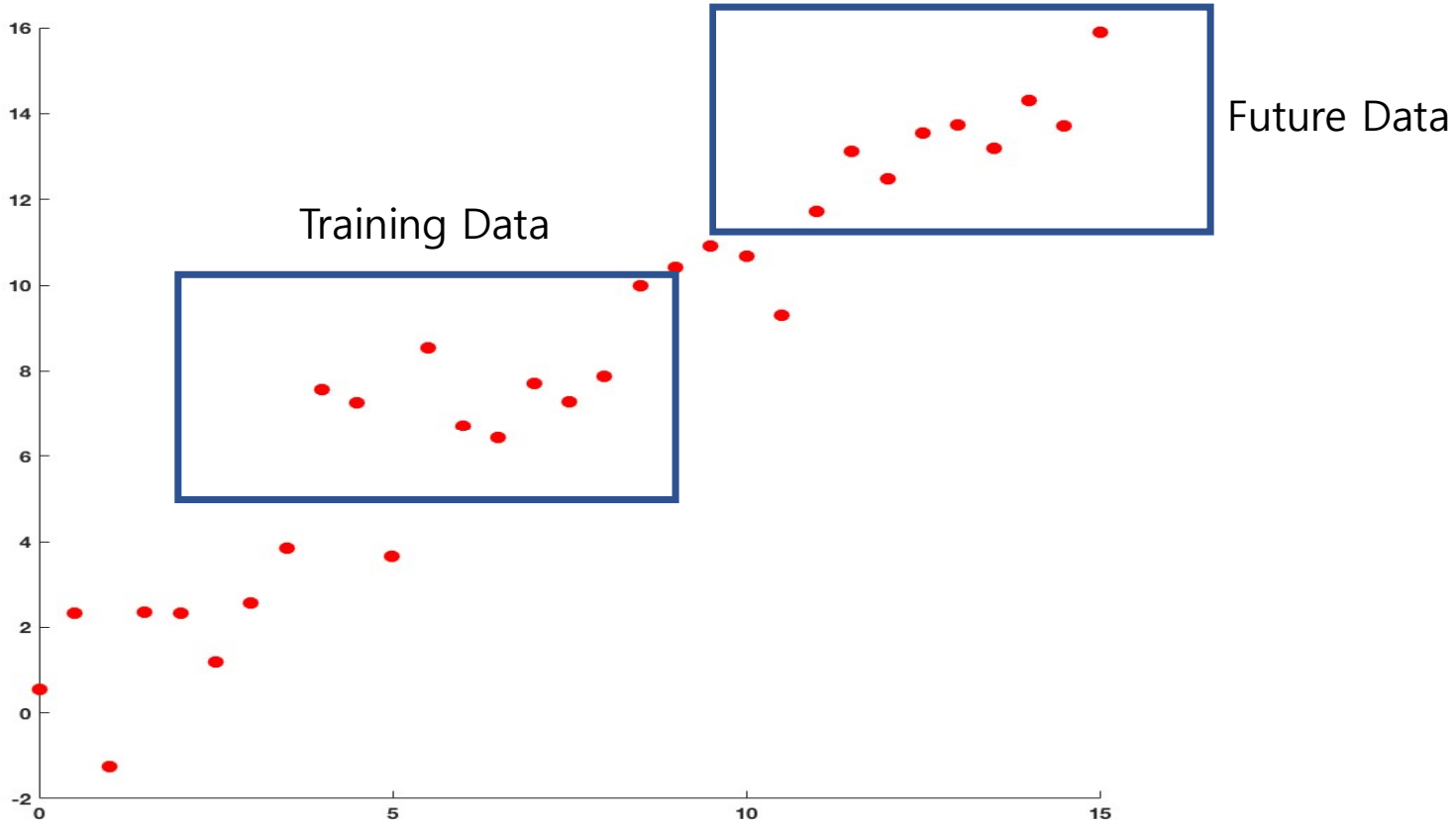
  1. 학습 Data 와 실제 data 분포의 차이에 의한 error → Variance

  2. Approximation Model 과 True Function 의 차이에 의한 error → Bias

  3. Noise 에 의한 error → 제거할 수 없음

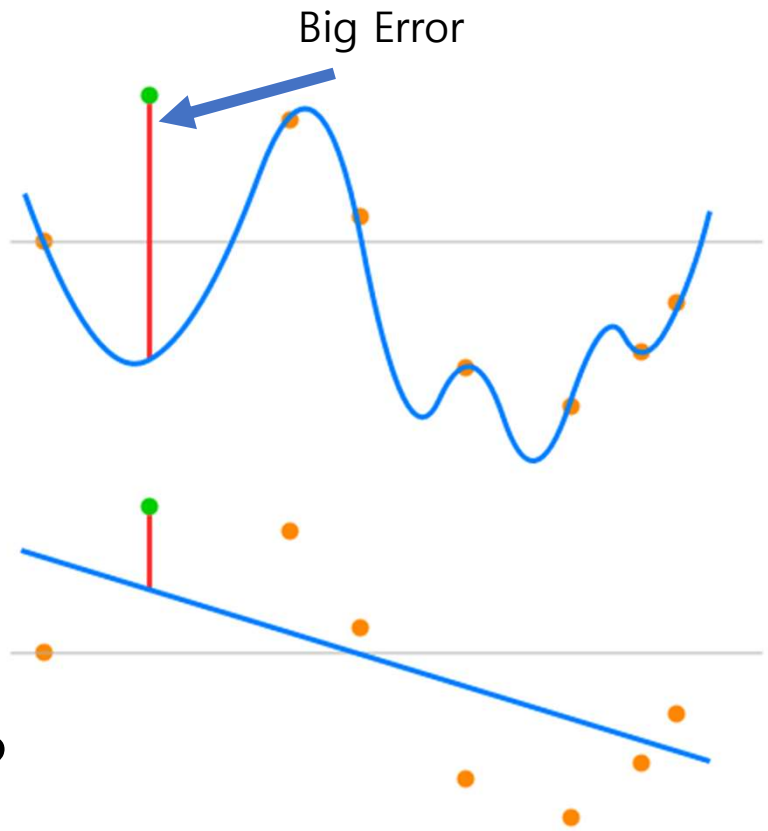
- Variance 를 줄이려면 Dataset 의 크기를 늘이고, Bias 를 줄이려면 모델의 Complexity 를 올린다.
  → Bias-Variance Dilemma (Bias-Variance Trade-off)

Variance – Training Data vs. Future Data



Bias – True & Inference Function 차이

- True function 은 Sign 함수
- Model 1 – polynomial regression
- Model 2 – Linear regression



Which one is better model ?

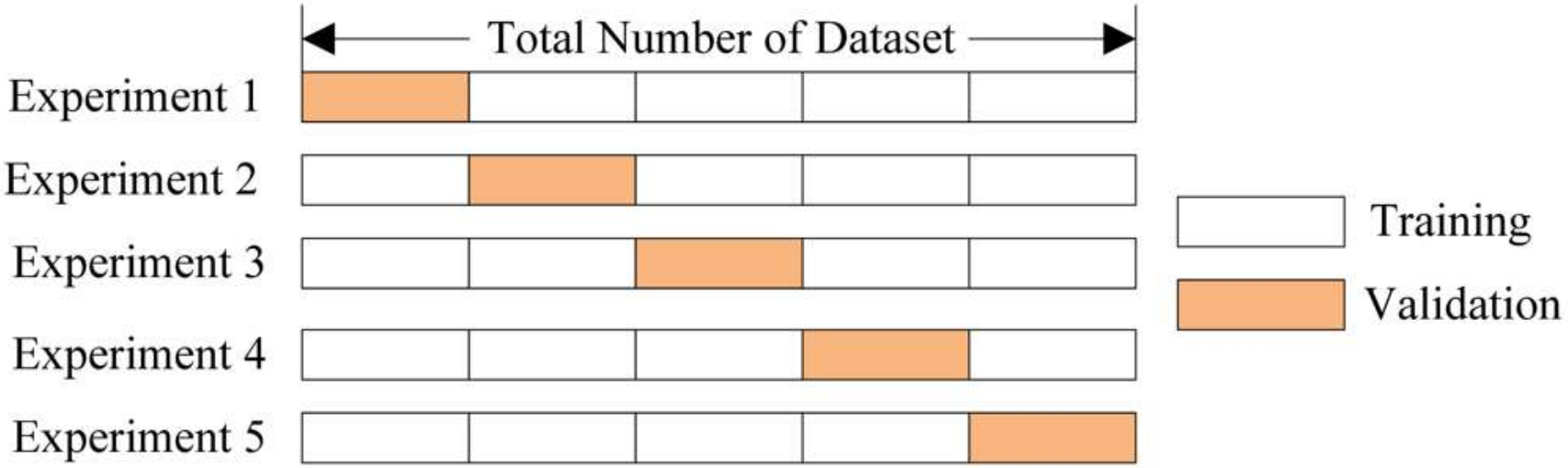
- Variance – infinite data sampling 으로 해결 가능
- Bias – true function 을 알면 해결 가능
- **BUT,** 현실에서는 infinite data sampling 도 할 수 없고 true function 도 알 수 없으므로 간접적 방법을 사용
  1. Cross Validation
  2. Precision / Recall / F1-Score

Training & Testing & Cross-Validation Set

- Training Set
  - parameter 를 inference 하는 procedure 에 사용하는 data
  - 보지 못한 Data 의 분포가 Training set 과 상이할 경우 문제
  - Training set 내에서 cross-validation set 을 구성 (Data 가 충분한 경우)
- Testing Set
  - 학습한 Machine Learning Model 을 Test 하기 위해 사용하는 data
  - 미래의 instance 인 것처럼 간주
- Training set 과 Testing set 은 섞이면 안되고 동일한 분포를 유지 해야함.

Cross Validation (교차검증)

- 훈련세트를 여러 개의 sub-set 으로 나누고 각 모델을 이 sub-set 의 조합으로 훈련시키고 나머지 부분으로 검증
- Data 의 수가 적은 경우 사용



실습: Train/Test split and K-Fold Cross-Validation

1. Training Set 과 Test set 으로 구분

- 당뇨병 dataset 을 이용하여 실습
- sklearn.model_selection 의 train_test_split method 사용
- Test set (true value) 를 x 축으로, predicted value 를 y 축으로 시각화
  - True value vs. predicted value 산점도(scatter plot) 표시
  - True value 와 predicted value 가 일치할 경우를 line 으로 표시

2. k-Fold Cross-Validation

- sklearn.model_selection 의 cross_val_score 를 이용하여 model 평가
- 데이터가 소량인 경우 혹은 여러 model 의 적합성 비교에 편리
- computing cost high

Model Performance for Biased Data

Confusion Matrix (혼동 행렬)

		True condition	
	Total population	**Condition positive**	**Condition negative**
Predicted Condition	**Predicted condition positive**	**True positive** TP	**False positive** FP
	Predicted condition negative	**False negative** FN	**True negative** TN

- Classification (분류) 성능의 정확성 측정

  TP – 1 을 1 로 제대로 분류, FP – 0 을 1 로 잘못 분류
  FN – 0 을 1 로 잘못 분류, TN – 0 을 0 으로 제대로 분류

- Classification rate (Accuracy) = (TP + TN) / (TP+TN+FP+FN)
  ➔ 단순 정확성. 전체 데이터 중에서, 제대로 분류된 데이터의 비율

Confusion Matrix 를 이용한 분류 모델 성능 평가

- Precision = TP / (TP + FP)
  - 정밀성. **Positive로 예측한 내용 중에, 실제 Positive**의 비율
  - positive 분류의 정확성 측정 (1 에 가까울 수록 좋음)

		True condition	
	Total population	**Condition positive**	**Condition negative**
Predicted Condition	**Predicted condition positive**	**True positive**	**False positive**
	Predicted condition negative	**False negative**	**True negative**

Confusion Matrix 를 이용한 분류 모델 성능 평가

- Recall (Sensitivity/ True positive Rate) = TP / (TP + FN)
  - 민감도. **전체 Positive 데이타 중에서 Positive로 분류**한 비율 (1 에 가까울 수록 좋음)
  - Positive case 를 놓치고 싶지 않은 경우의 성능 측정

		True condition	
	Total population	**Condition positive**	**Condition negative**
Predicted Condition	**Predicted condition positive**	**True positive**	**False positive**
	Predicted condition negative	**False negative**	**True negative**

Precision / Recall 활용 방법

- Confidence 수준을 올리고 싶으면 Precision 을 높이고 Recall 을 낮추도록 Threshold 조정. 너무 많은 case 를 놓치고 싶지 않은 경우 Recall 을 높이고, Precision 을 낮춘다.

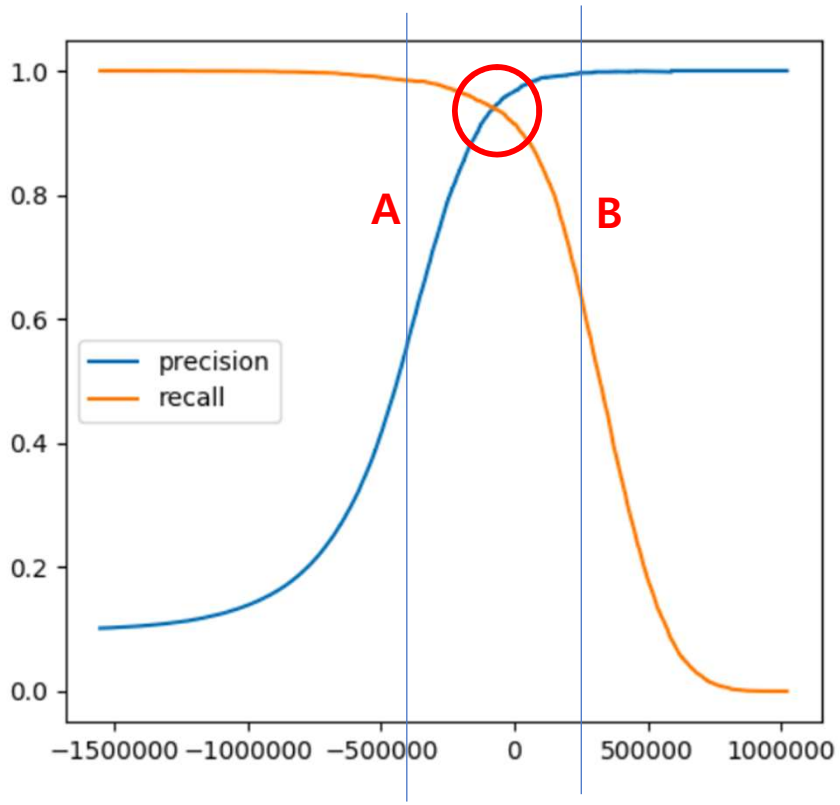
- 전체적 성능 측정에 활용 (조화평균)

$$\text{F1-Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

F1-Score = 0 ---> Poor (P=0 or R=0)
F1-Score = 1 ---> Perfect (P=1 or R=1)

A – High Recall / Low Precision

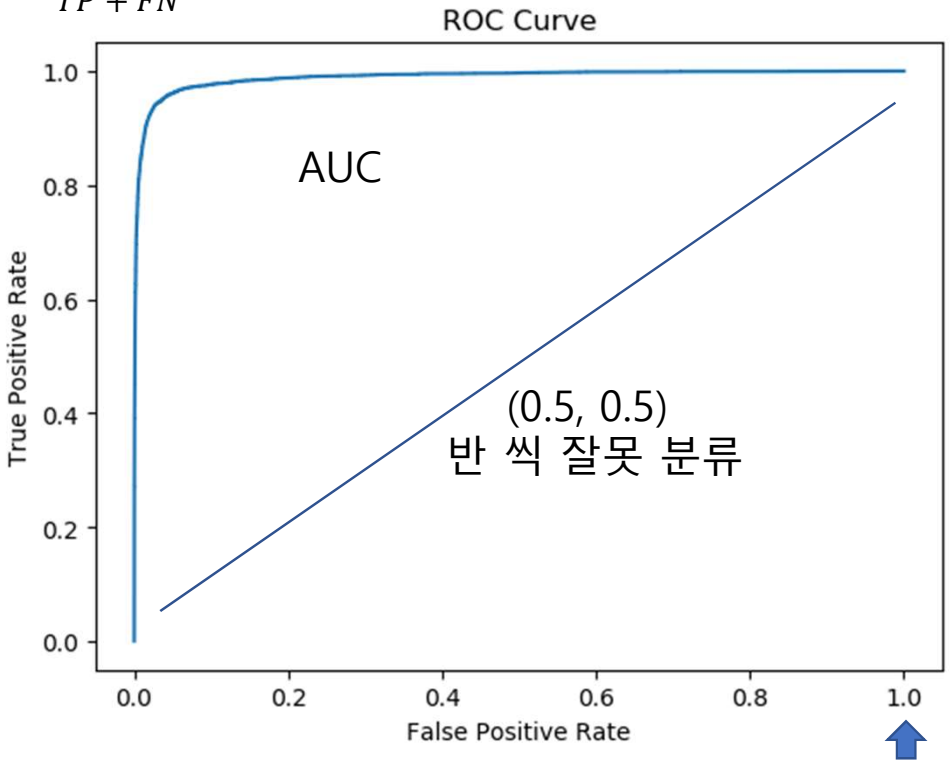
B – High Precision / Low Recall



ROC Curve (수신자 조작 특성 곡선)

$$TPR = \frac{TP}{TP + FN}$$

모든 Positive 를
Positive 로 정확히 분류



$$FPR = \frac{FP}{FP + TN}$$

모든 Negative 가
Positive 로 잘못 분류

- ROC (Receiver Operating Characteristic) curve 는 radar 상의 적기 탐지를 위해 개발되었던 분석 기법
- ROC_AUC (Area Under the Receiver Operating Characteristic Curve) 라고도 함
- roc_auc_score 를 이용하여 **분류기 간의 성능 비교**를 할 수 있다.

Logistic Regression

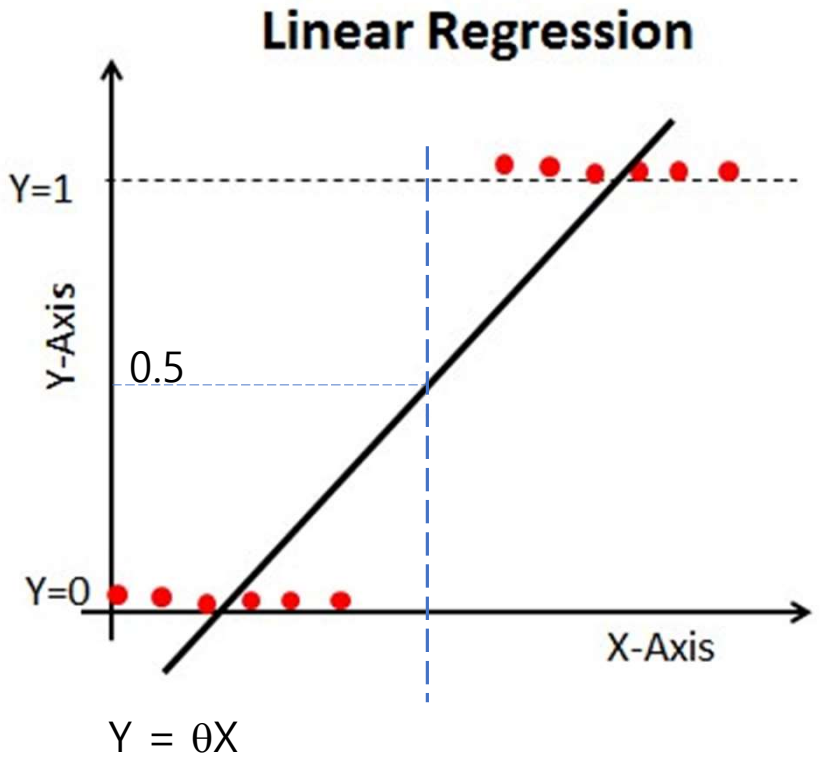
Logistic Regression (로지스틱 회귀)

- 선형회귀를 분류 문제에 적용 가능 ?

$\hat{y}$ = 0 if $\theta X < 0.5$

1 if $\theta X \geq 0.5$

- 0 과 1 로 구성된 분류 문제에 적합한 함수 ?
- 0 과 1 에 속할 확률값을 return
- 미분가능한 성질



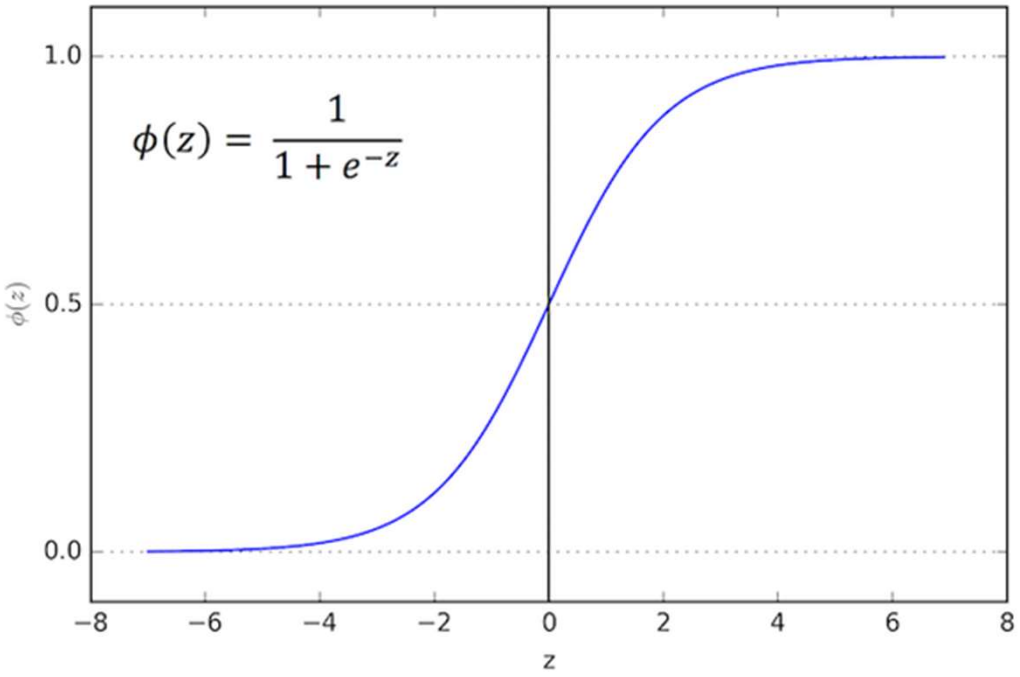
Sigmoid 함수

- S curve 형성
- Logistic 함수

$$f(z) = \frac{1}{1+e^{-z}} \quad (z = \theta X)$$

$$\hat{y} = 0 \text{ if } f(z) < 0.5$$
$$1 \text{ if } f(z) \geq 0.5$$

- [0, 1] 로 bound 되어 있음
- 미분가능
- 0.5 부근에서 급격히 변화



Logistic Regression (로지스틱 회귀) classifier

1. 가장 단순한 분류기 → binary-class

2. 구현이 간단하고 모든 classification problem 의 기초

3. Logistic Regression 의 기본 concept 은 Deep Learning 에도 적용

4. 독립변수와 종속변수 간의 관계를 찾아내고 평가함

5. sklearn.linear_model.LogisticRegression(solver='lbfgs')

   * solver – optimization algorithm

실습: Logistic Regression 을 이용한 Binary Classification

1. 특정 사용자가 구매를 할지 여부를 예측 (구매: 1, 구매 않음: 0)

2. Dataset 구성
   사용자 id, 성별, 연령, 추정급여, 구매여부
   이중 연령, 추정급여 두가지 feature 를 이용하여 구매여부 예측

3. Train / Test dataset 은 8 : 2 로 분리

4. Feature Scaling 실시 (standard scaling)

5. Model evaluation by Confusion Matrix, f1-score

6. Visualization – ROC curve, 결정경계 시각화

Multi-class classification

Binary Classification

예) 고객이 구매할지 여부
0 - 비구매
1 - 구매

Titaninc 호 생존 여부
1 - 생존
0 - 사망

Multiclass Classification

예) 유권자의 정치적 성향
1. 보수적
2. 진보적
3. 중도적

붓꽃의 종류
1. Iris
2. Setosa
3. Virginica

실습: Hand-write digit 을 이용한 Multi-class Classification

1. from sklearn.datasets import load_digits

2. 0 – 9 까지의 손글씨체 분류

3. multi_class='ovr : One-vs-Rest

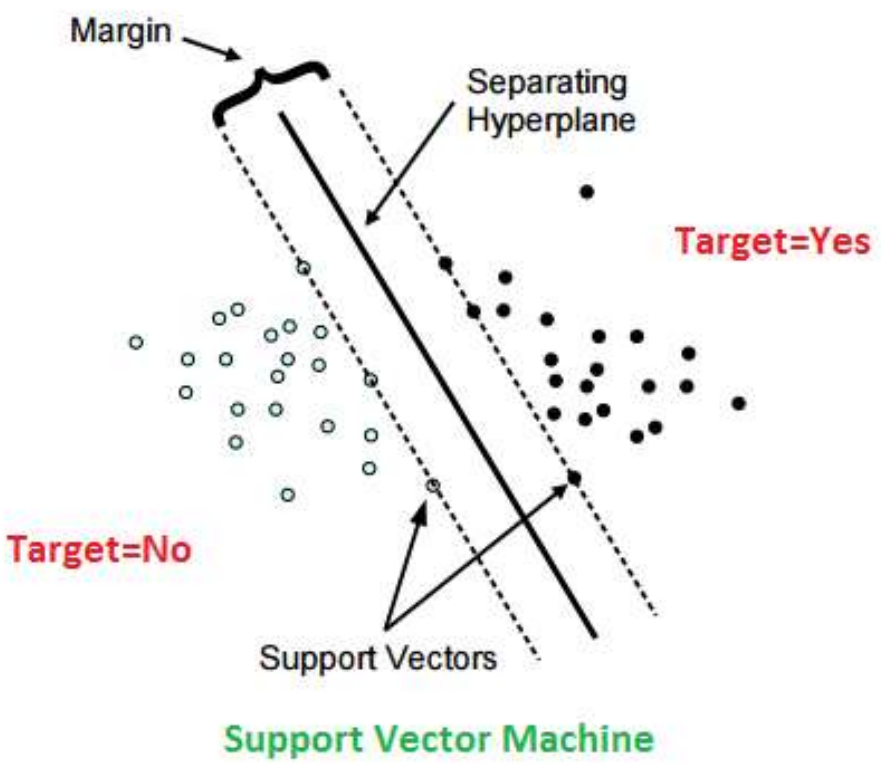
   K개의 클래스가 존재하는 경우, 각각의 클래스에 대해 표본이 속하는지(y=1) 속하지 않는지(y=0)의 이진 분류 문제를 푼다.

Support Vector Machine

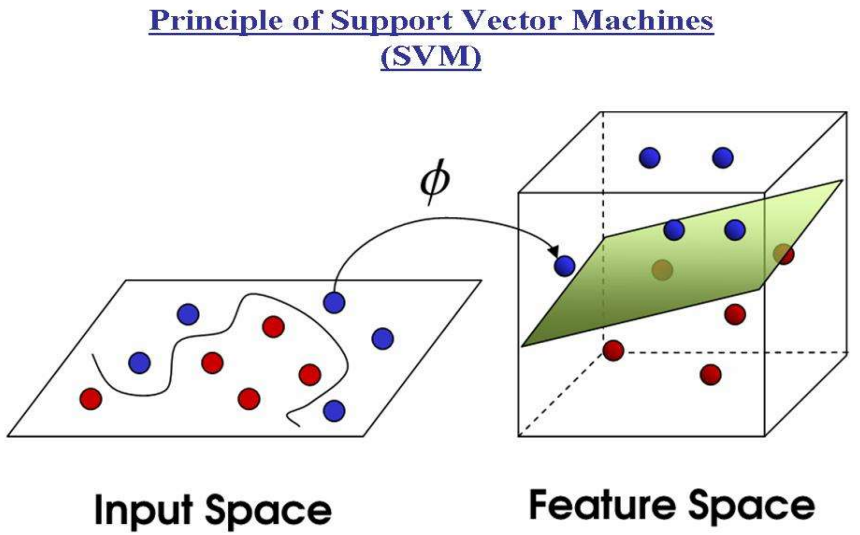
SVM (Support Vector Machine)

- Data 를 High Dimension feature space 에 mapping 하여 separate
- Support Vector 를 이용하여Data 사이를 분리하는 hyper-plane 유추

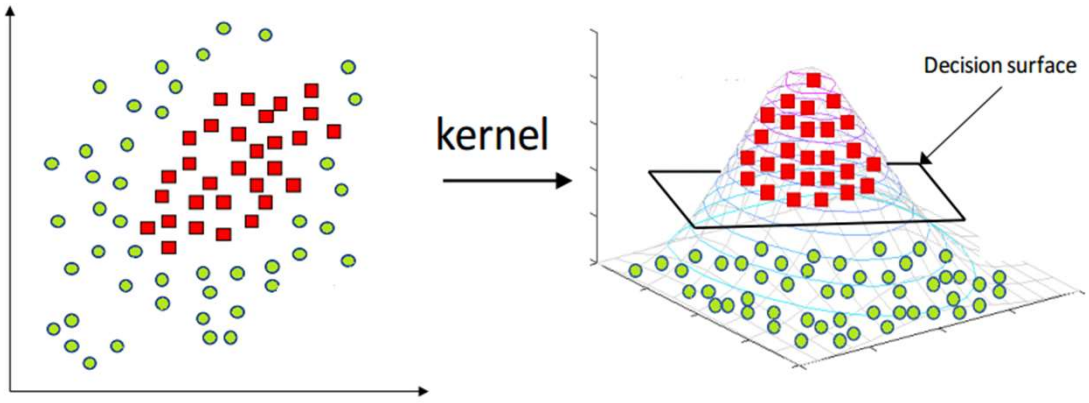
- 장점 – High Dimensional Space 에서 정확.
  Memory efficient.
- 단점 – Overfitting 되기 쉽다.
  No Probability Estimation (확률모델이 아님)
  Small Dataset 에 적합.



- Kernel Trick
  - Non-Linearly separable dataset 을 차원 변경하여 separating hyper-plane 생성



- Kernel 의 종류
  - 다항식 kernel
  - Gaussian (RBF) kernel
  - Hyperbolic Tangent kernel, etc



실습: Logistic Regression 과 동일한 Dataset 을 SVM 적용

1. 특정 사용자가 구매를 할지 여부를 예측 (구매: 1, 구매 않음: 0)

2. Dataset 구성
   사용자 id, 성별, 연령, 추정급여, 구매여부
   이중 연령, 추정급여 두가지 feature 를 이용하여 구매여부 예측

3. Train / Test dataset 은 8 : 2 로 분리

4. Feature Scaling 실시 (standard scaling)

5. Model evaluation by Confusion Matrix, f1-score

6. Visualization – 결정경계 시각화 ➔ **Logistic Regression 과 비교**

Ensemble Learning

Random Forest/Gradient Boosting

What is Ensemble Learning ?

- 다수의 약한 학습기 (weak learner) 를 조합(Ensemble) 하여 더 높은 성능 추출

- Decrease Variance by **Bagging** (Bootstrap Aggregating)

- Decrease Bias by **Boosting**

What is Bagging (Bootstrap Aggregating) ?

- Training sample 의 sub-set 을 무작위로 추출하여 classifier 훈련

- Bootstrap – 중복을 허용하는 random sampling 방법 (통계학)

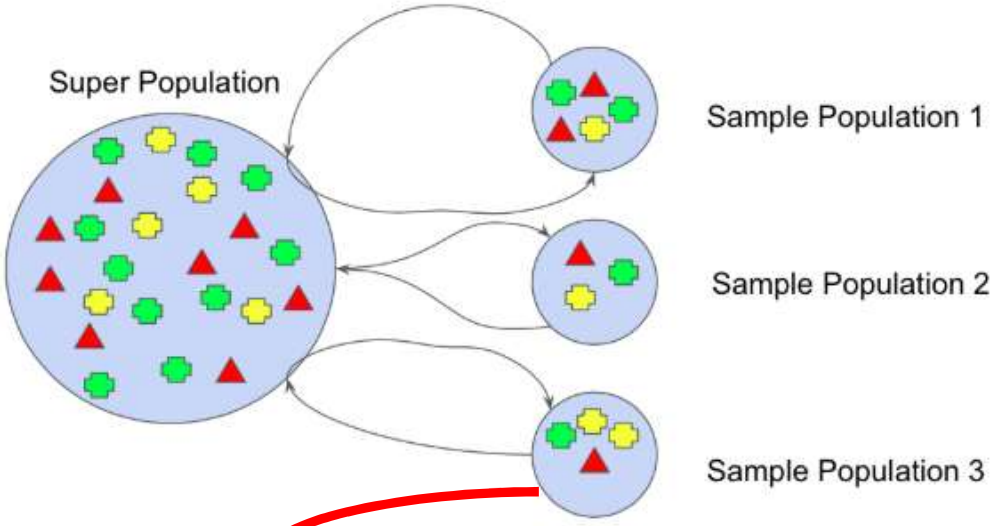
  ➔ b 개의 independent training set 을 평균하면,

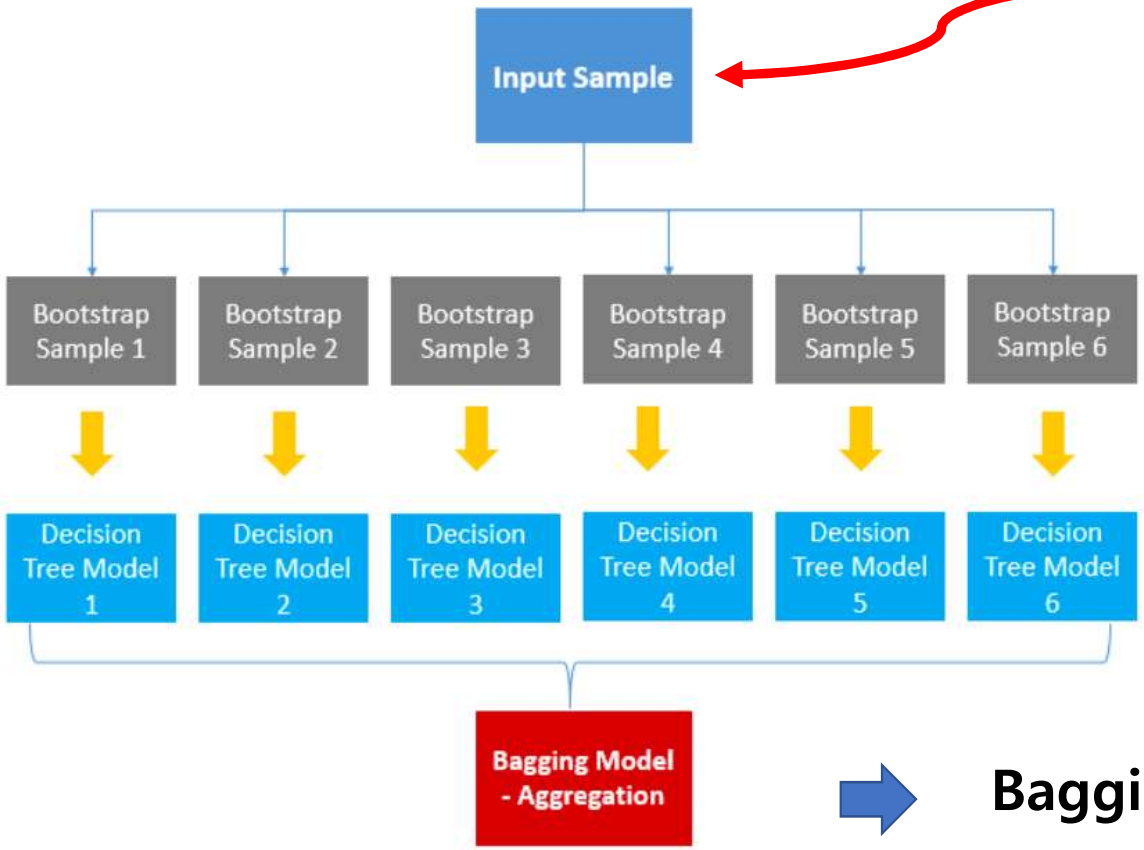
  variance → $\frac{\sigma^2}{b}$ 로 감소, mean → 동일

- Aggregating – voting for classification

- **Random Forest** 가 대표적인 method

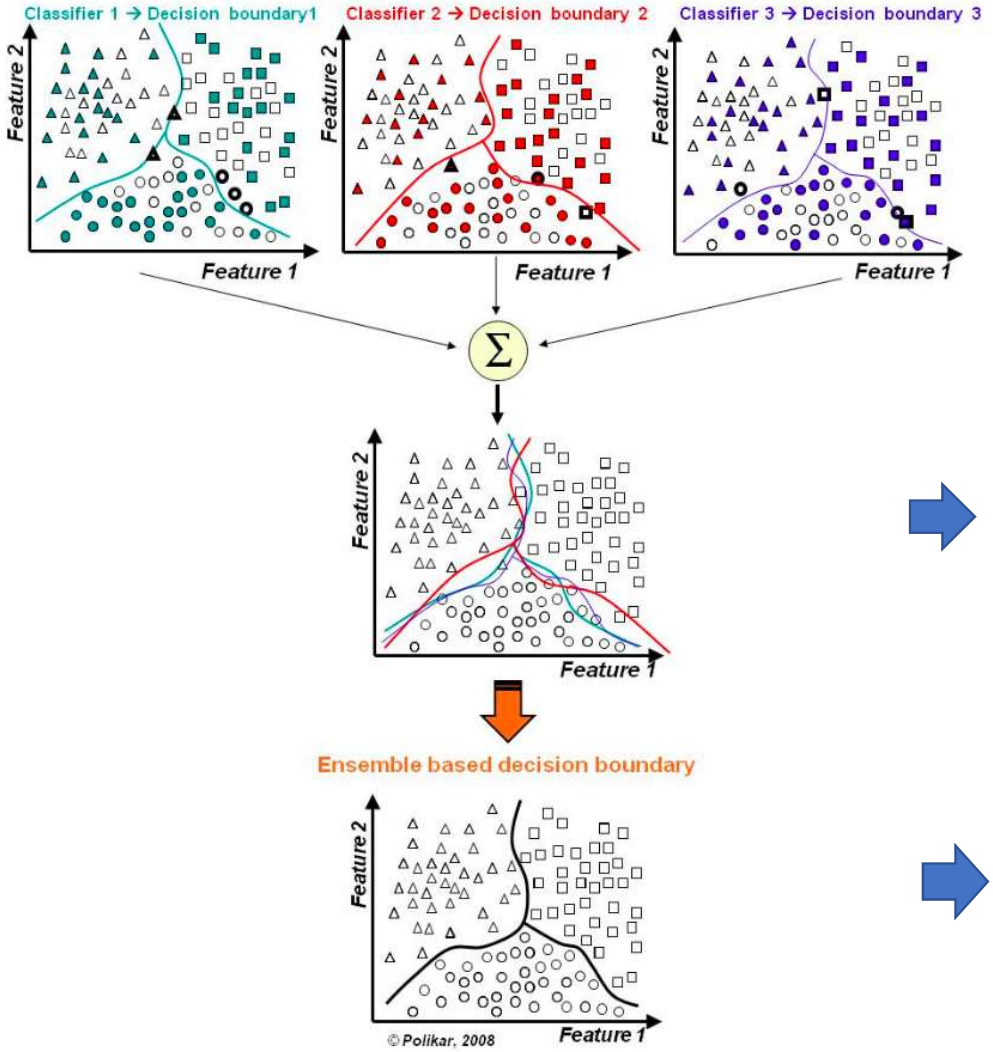
Bootstrapping (복원추출)





Decision Tree

Bagging Tree (Random Forest)



Decision Trees

Majority Voting

Random Forest

Bagging Tree 알고리즘

1. training data 에서 random sampling 하여 X_b, y_b 를 고름
   (ex. 1,000 개의 data 중 100 개 sampling)

2. X_b, y_b 를 이용하여 **ID3 알고리즘**으로 **decision tree** 구성

3. B 개의 tree 가 만들어질 때까지 1, 2 반복

4. B 개의 모든 tree 를 이용하여 classification 한 후 majority vote 로 결정

5. Bias 를 유지하며 Variance 를 줄인다.

   ➔ bagging 은 random sampling 에 의해 model 을 fit 하므로 **bias (model complexity) 가 커지지 않으면서 variance 를 줄일 수 있는 특징**이 있다.

Random Forest

- Bagging Tree

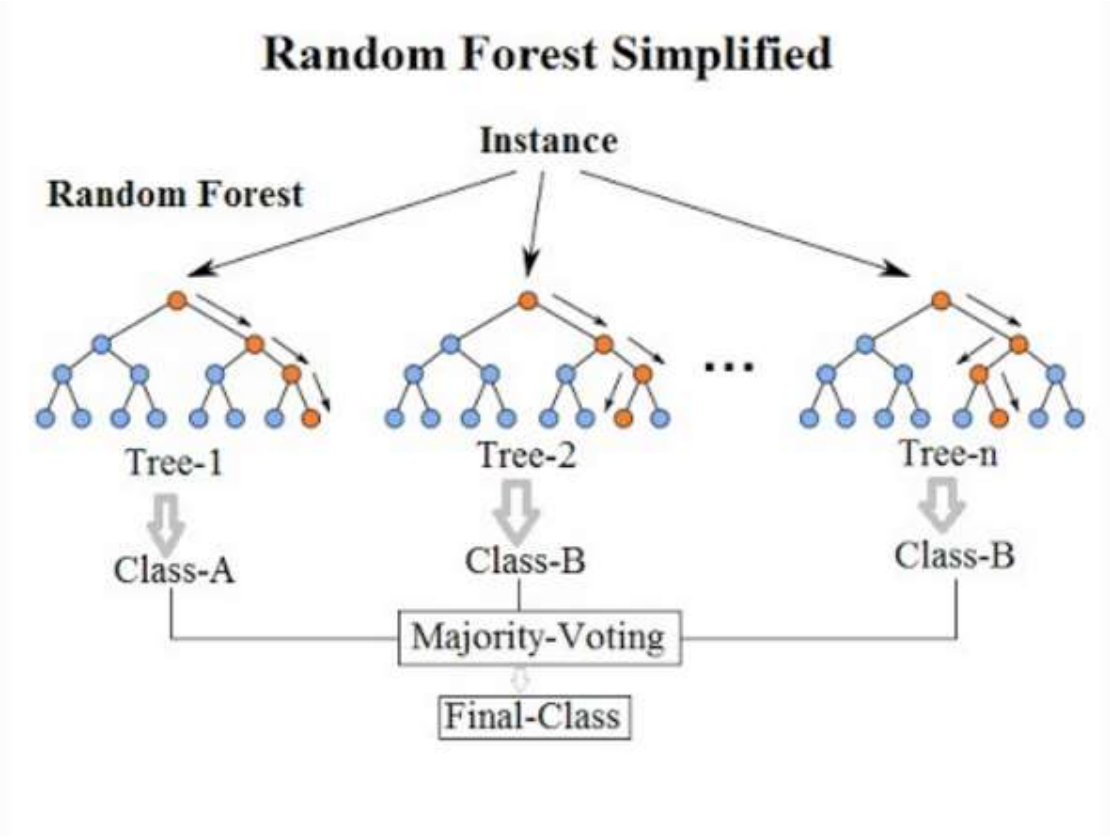
  ➔ Random Forest 로 발전

- **Randomly Choose Attributes**

- bootstrapping 에 의한 복원추출

  +

  attribute 의 random 선택에 따른 independent trees 구성



Random Forest Algorithm

1. Decision Tree 에 포함될 attribute 들을 random 하게 선정

   ➔ 모든 attribute 를 가지고 Tree 를 만들 경우 매우 강한 attribute 가 모든 tree 에 항상 포함되는 것을 막기 위한 방법
   (ex. 30 개 attribute 중 10 개만 random selection)

   ➔ 좀 더 Random 하고 독립적인 classifier (Tree) 들을 생성시킬 수 있으므로 Random Forest 로 명명

2. Tree-based model 이므로 white box 특징을 유지

3. High prediction accuracy

4. 병렬적으로 생성 가능하므로 속도가 빠르다

5. 각 tree 는 매우 deep 하게 생성된다.
   ➔ 인위적인 prune 을 하지 않으나 Ensemble 을 하면 low bias, low variance 가 된다.

What is Boosting ?

- Misclassified data 에 더 높은 weight 를 부여하여 다음 번 model 의 sampling 에 포함될 확률을 높이는 것

- 다수의 weak learner (small decision tree) 를 훈련 시켜 majority voting 하는 것은 Bagging 과 동일

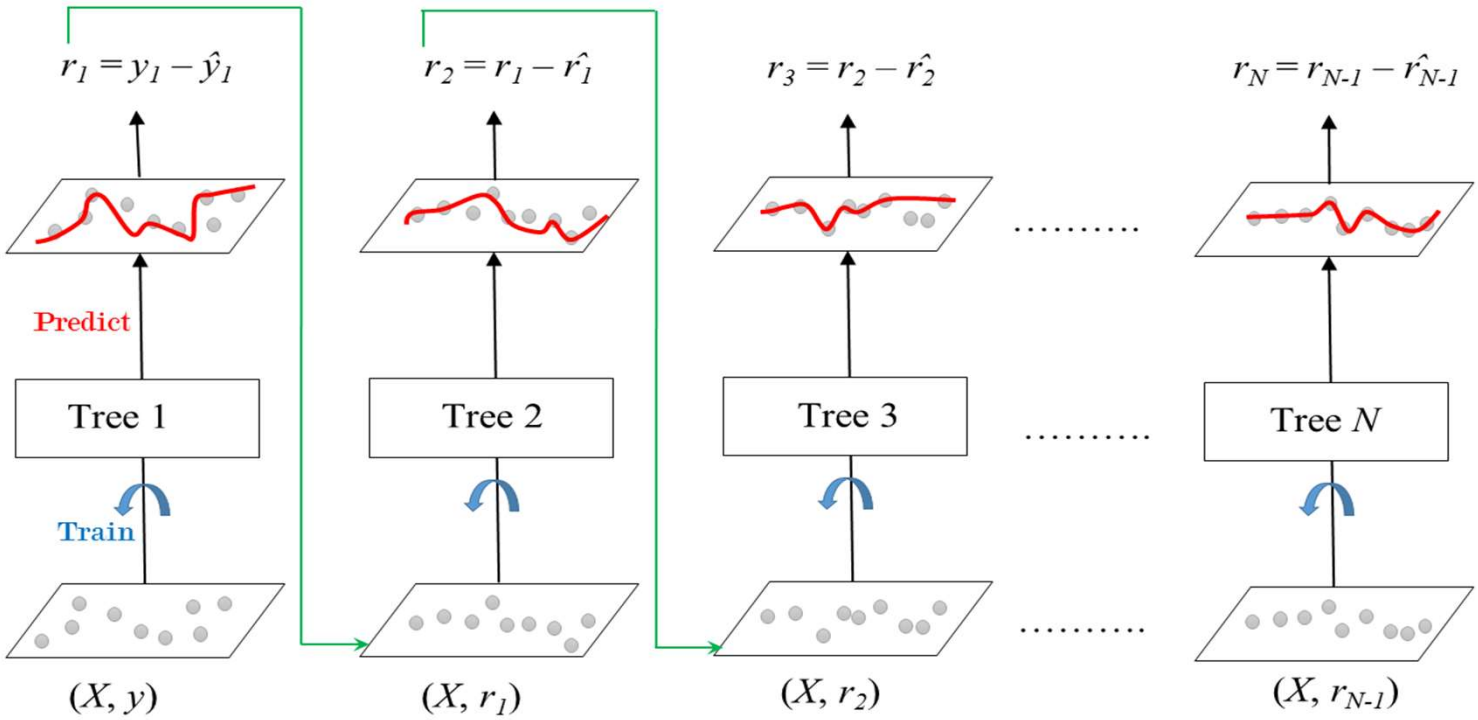
- **AdaBoost, Gradient Boost (XGBoost)** 가 대표적인 method

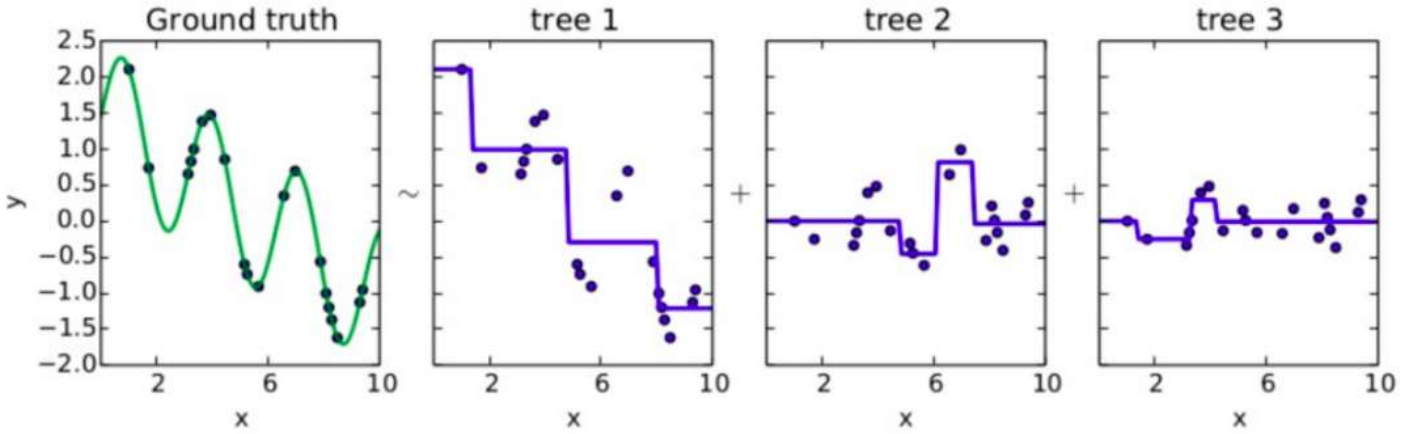
Gradient Boost 알고리즘

- Weak learner - random choice 보다 약간 더 나은 성능의 모델
  ➔ Decision Tree 사용

- 기존의 weak learner 가 생성한 residual error 를 감소시키는 추가 tree 를 더 이상의 감소 효과가 없을 때까지 생성

- Gradient Descent (경사하강법)에 의해 loss function (ex. MSE) 을 optimize





- 이전 tree 에서 발생한 잔차 (residual error) 를 next tree 에서 보정

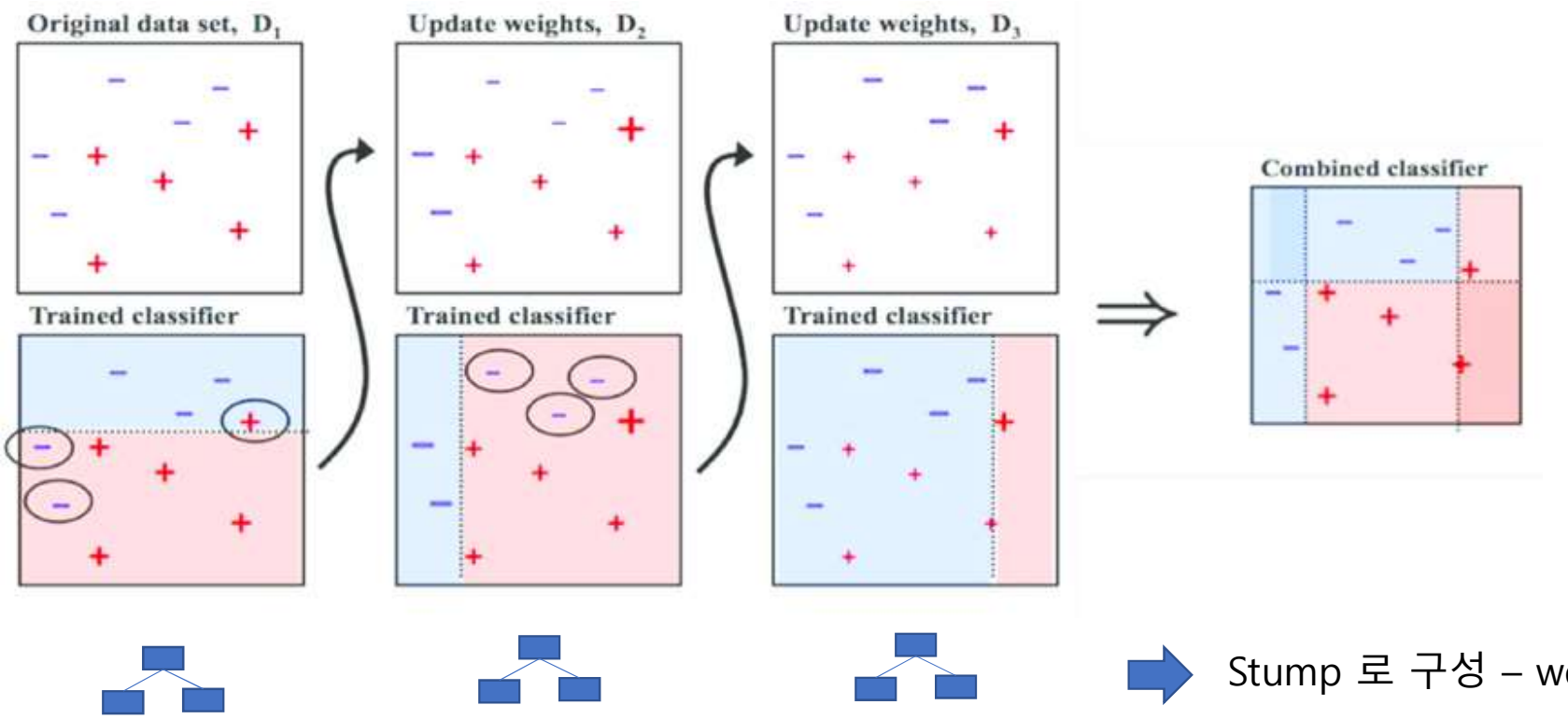
- 학습이란 손실함수를 최소화하는 파라미터를 찾는 일

- 최적의 파라미터를 찾는 방법 중 하나가 Gradient Descent

- 손실함수를 파라미터로 미분해서 기울기를 구하고, 값이 작아지는 방향으로 파라미터를 움직이다 보면 손실함수가 최소화되는 지점에 도달. Gradient Boosting 손실함수를 파라미터가 아니라 현재까지 학습된 모델 함수로 미분

$$f_{i+1} = f_i - \rho \frac{\partial J}{\partial f_i}$$

(참고) AdaBoost (Adaptive Boosting)



Stump 로 구성 – weak learner

이전 stump 에서 잘 못 분류한 example 이 다음 sampling 에 포함되도록 가중치를 높이는 기법

Tree Models 비교

Model	비 유
Tree	개별 면접관들이 각자의 평가 기준 (학력, 경력, 인터뷰 성적 등)에 따라 선발
Bagging (Bootstrap Aggregating)	Tree 방식의 면접관들이 패널을 구성하여 민주적으로 투표하여 선발
Random Forest	Bagging 방식과 유사하나 평가 기준은 random 하게 일부분만 선택. (면접관 별로 기술부분만 평가, 혹은 인성 부분만 평가)
Boosting	이전 면접관의 피드백에 따라 다음 면접관이 dynamic 하게 평가 기준을 변경해가며 평가
Gradient Boosting	Boosting 에 Gradient Descent 알고리즘 적용.
XGBoost, LGBM	Gradient Boosting 의 속도 개선

- 실습: Social_Network_Ads data 를 이용

1. Random Forest 를 이용한 분류

2. Gradient Boosting 을 이용한 분류

3. 이전 모델과 동일한 요령

Overfitting 방지 기법 (Regularization)

- Linear Regression (선형회귀)

$$J(W) = MSE_{train}(W) + \lambda\Omega(W)$$

손실함수 Train loss

λ : regularization 강도
Ω : regularizer

대표적 regularizer

1. L2 regularization : $\Omega(W) = ||W||_2$ (ridge regularization)
2. L1 regularization : $\Omega(W) = ||W||_1$ (lasso regularization)
3. Elastic net regularization : L1 + L2

Overfitting 방지 기법 (Regularization)

- Logitstic Regression (이진 분류)

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] + \lambda\frac{1}{2m}\sum_{i=0}^{m}\theta_j^2$$

regularizer

Binary cross-entropy 함수

λ : regularization 강도

Generalization

- No Free Lunch Theorem
  - 모든 문제를 하나의 model 로 해결할 수 없다. 즉, 하나의 문제에 잘 맞는 모델이 다른 문제에도 잘 맞는 것은 아님. 따라서, 다양한 model 를 try 하여 가장 잘 맞는 모델을 선정.

- Speed, accuracy, complexity 의 trade-off 로 model 과 알고리즘 선택
  - 모든 data 를 구할 수 없고, True Function 을 아는 것은 불가능하므로 machine learning 은 단지 확률과 통계에 기반하여 approximate 하는 것

- 간단한 model ➔ 복잡한 model 순으로 Try
  - Bias-variance Trade-off 원칙을 잊지 말 것

Feature Engineering

머신러닝을 위한 Feature Engineering

- ✓ Missing Values 처리
- ✓ Data Formatting
- ✓ 편향(skewed data) 처리
- ✓ Data Normalization
- ✓ Binning
- ✓ categorical 변수의 수치화

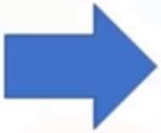
Data Formatting

City
NY
New York
N.Y
N.Y

City
New York
New York
New York
New York

Data Normalization

age	income
20	100000
30	20000
40	500000



age	income
0.2	0.2
0.3	0.04
0.4	1

Binning

price
13495
16500
18920
41315
5151
6295
...

price	price-binned
13495	Low
16500	Low
18920	Medium
41315	High
5151	Low
6295	Low
...	...

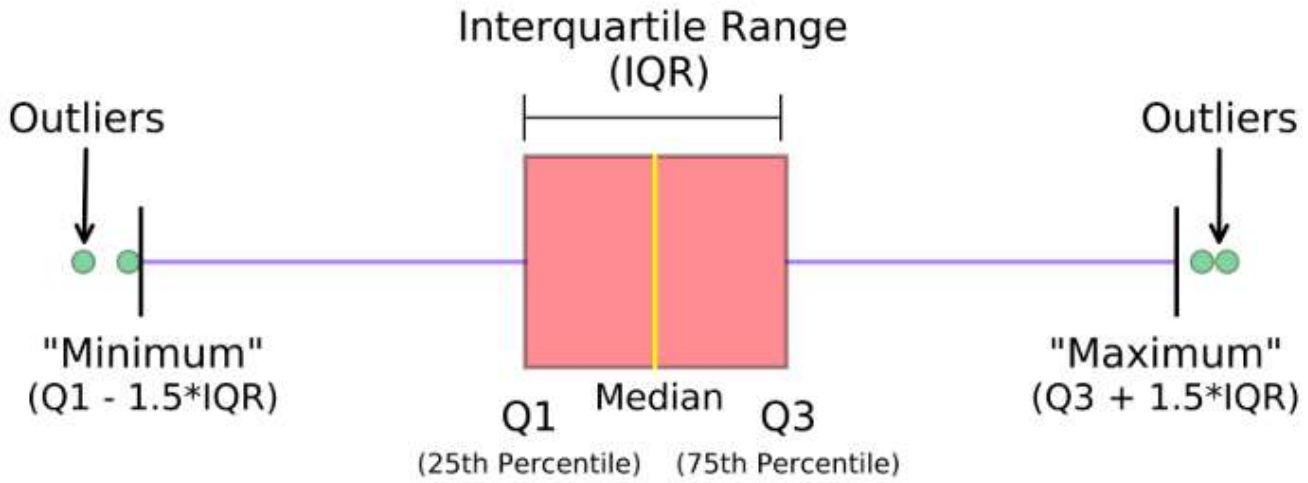
Categorical 변수의 수치화

fuel
gas
diesel
gas
gas

gas	diesel
1	0
0	1
1	0
1	0

Boxplot

- 4 분위수



실습: Titanic 호 data 를 이용한 Feature Engineering 과 Modeling

- 이전 모델과 동일한 dataset 이용
- Survival 예측 model 작성
- Feature Engineering
- Grid Search 를 통한 최적 model 및 parameter tunning

Clustering

Clustering 이란 ?

- 비슷한 object 들끼리 모으는 것

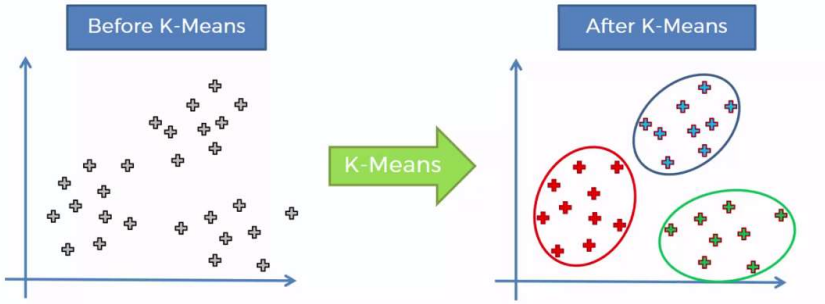
- label data 가 없는 것이 classification 과 차이

  ➔ unsupervised machine learning

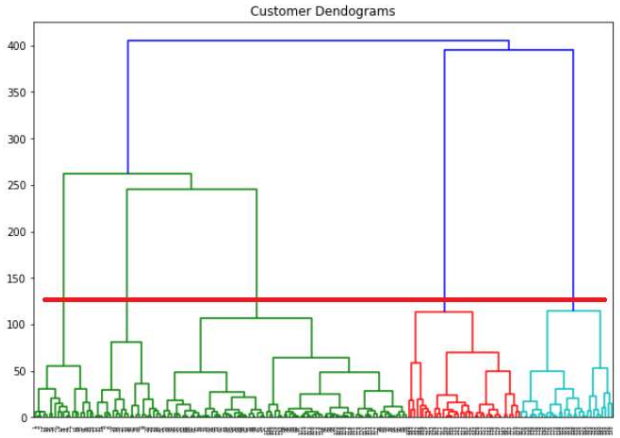
- 적용 사례
  - 고객의 구매 형태별 분류
  - 고객의 취향에 맞는 책, 동영상 등의 추천
  - 신용카드 사용의 fraud detection
  - 뉴스 자동 분류 및 추천
  - 유전자 분석 등

Clustering 알고리즘의 종류

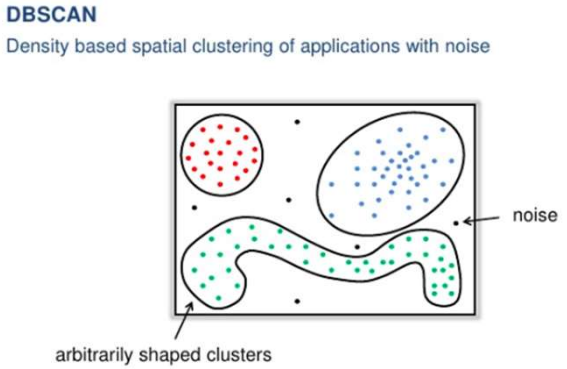
- K-Means Clustering



- Hierarchical Clustering (dendrogram)



- Density-based Clustering (DBSCAN)



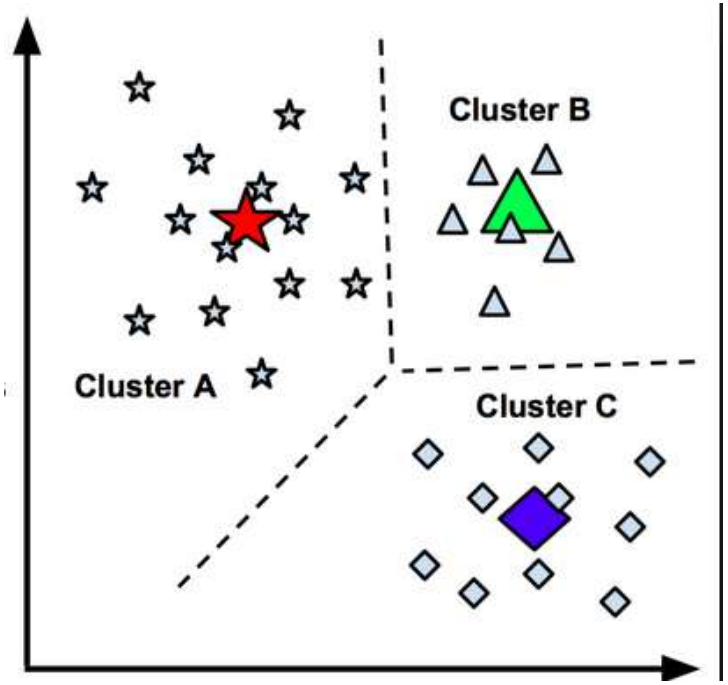
K-Means Clustering 알고리즘 – Distance 계산

Distance = Euclidean Distance

$$= \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

Ex)

고객	나이	수입	교육	
1	54	190	3	→ x1
2	50	200	8	→ x2



Distance(x1, x2) = $\sqrt{(54-50)^2 + (190-200)^2 + (3-8)^2}$ = 11.87

K-Means Clustering 알고리즘

1. Random 하게 k 개의 centroid (중심점) 를 정한다.

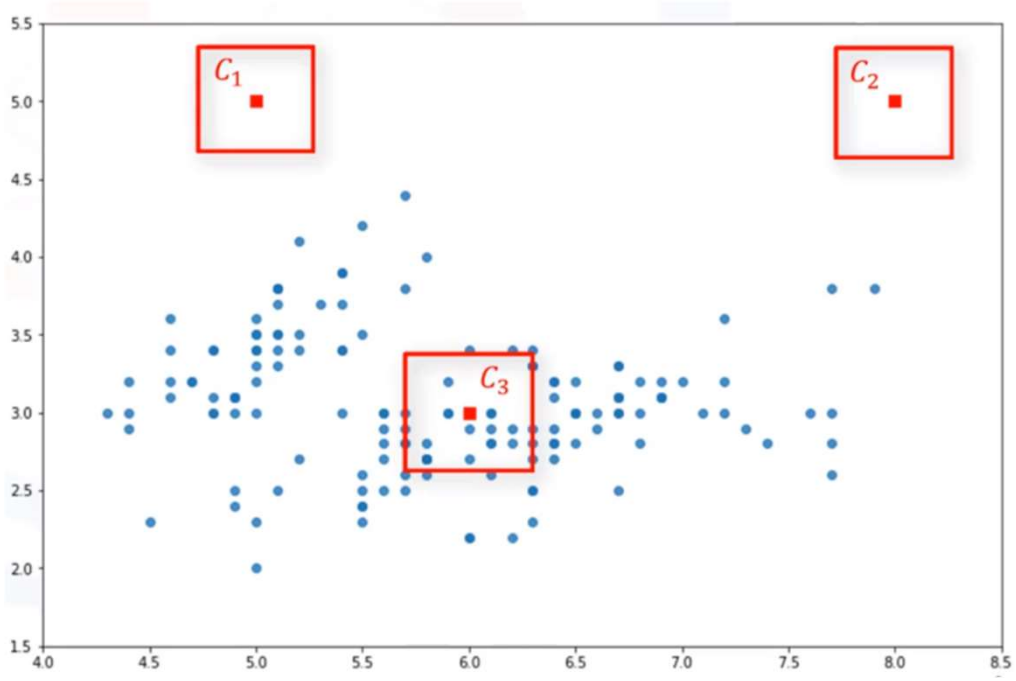
2. 각 centroid 로 부터 각 data point 까지의 거리를 계산.

3. 각 data point 를 가장 가까운 centroid 에 할당하여 cluster 를 생성.

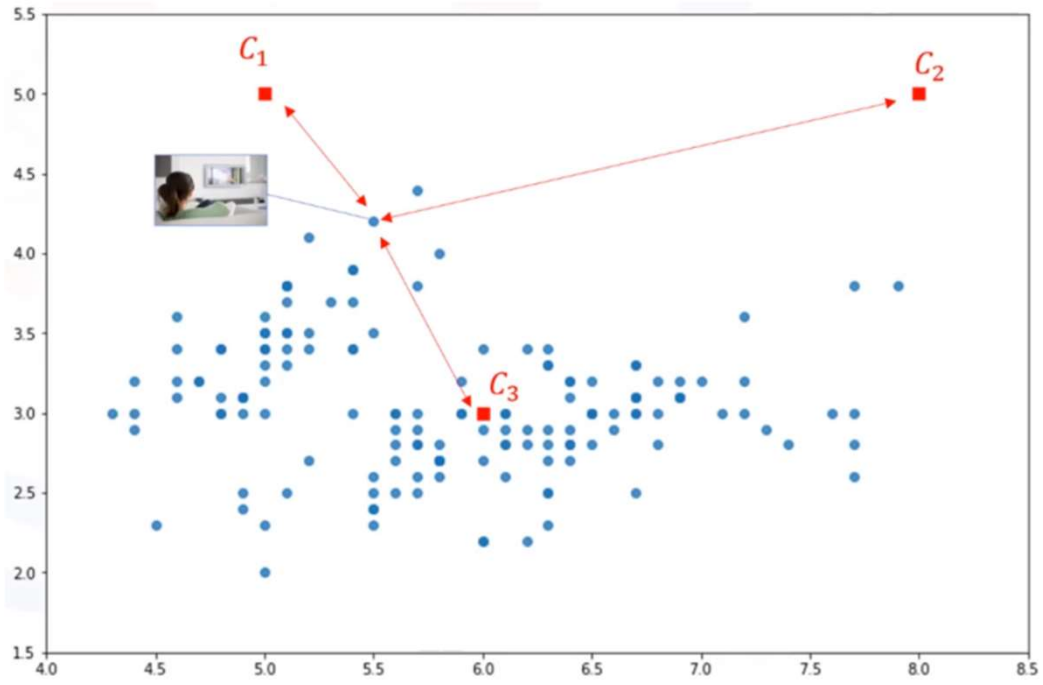
4. K centroid 의 위치를 다시 계산

5. centroid 가 더 이상 움직이지 않을 때까지 2-4 단계를 반복

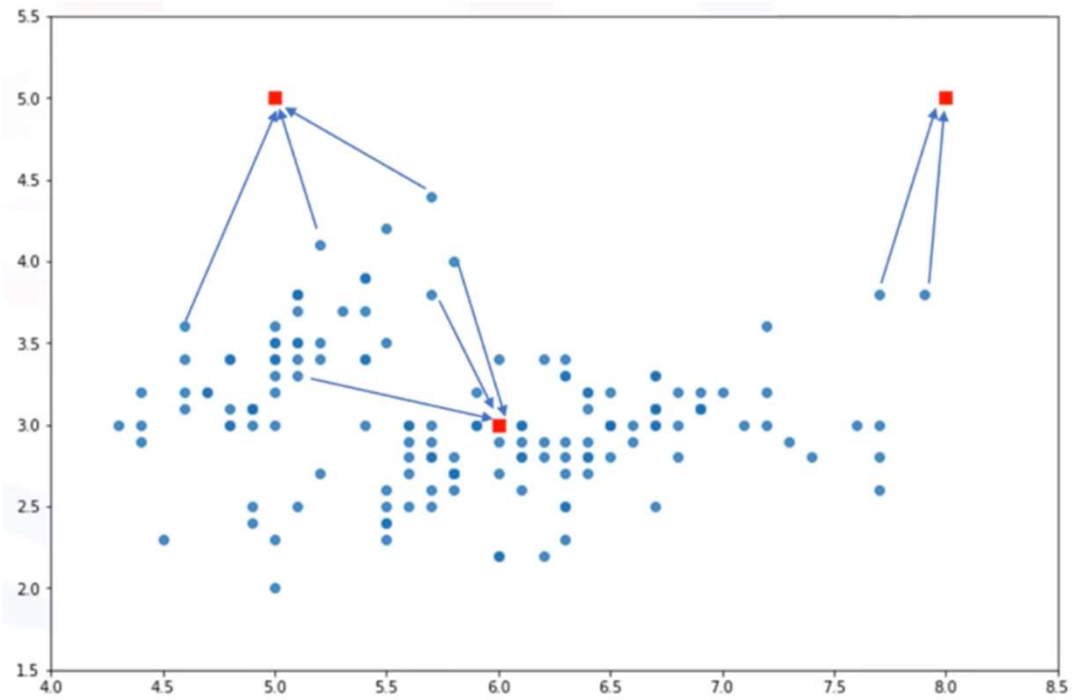
1. 임의의 centroid 선정 : k=3

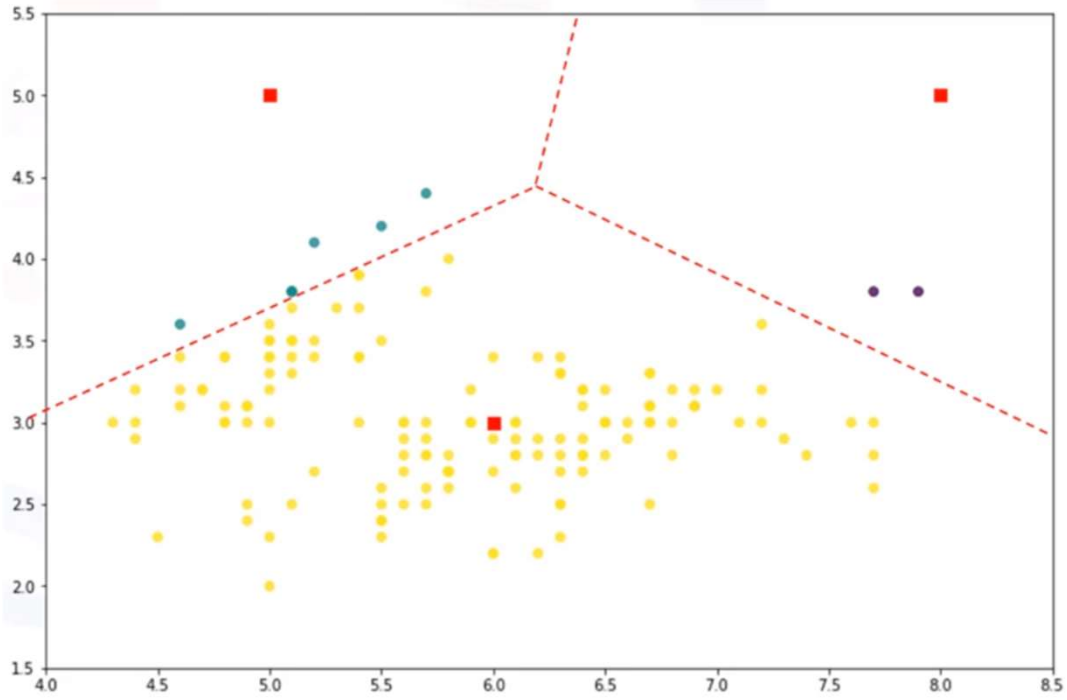


2. 거리 계산 for each data point

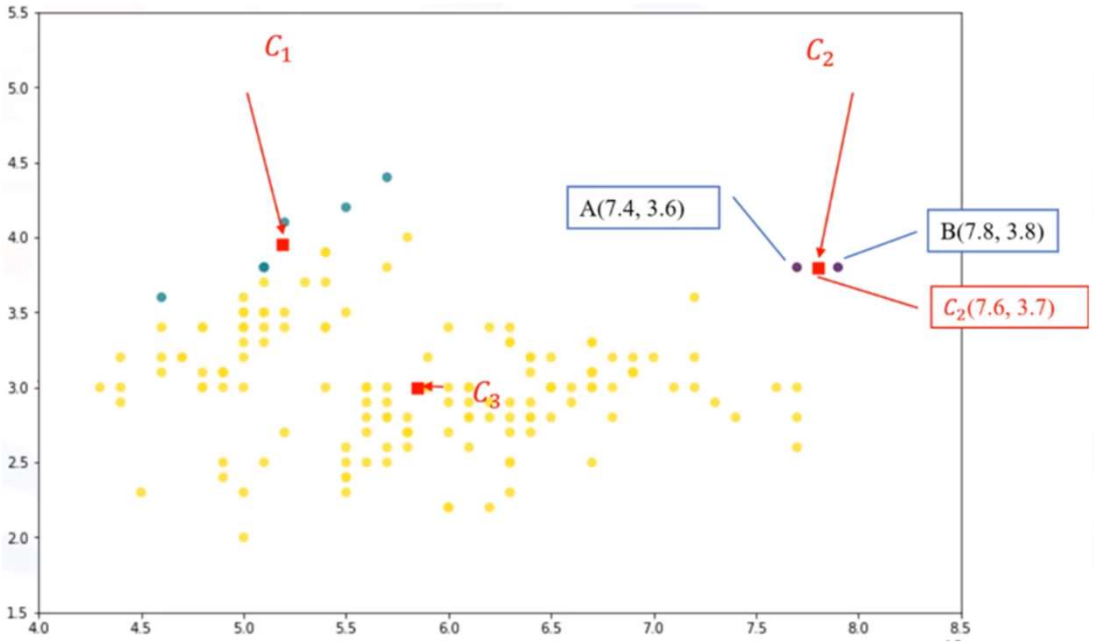


3. 가장 가까운 centroid 로 각 data point 할당

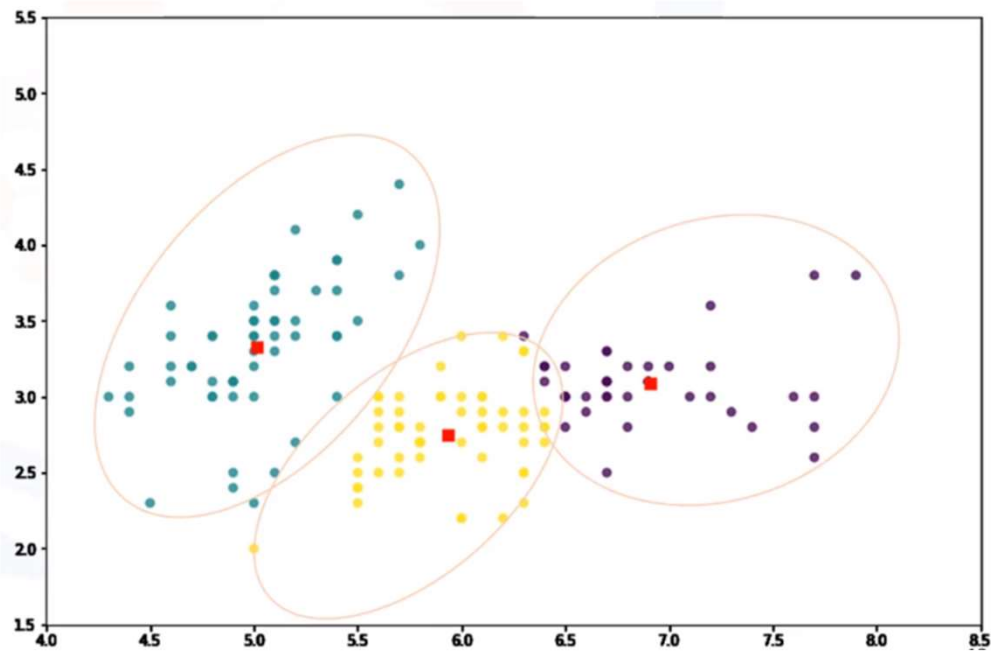




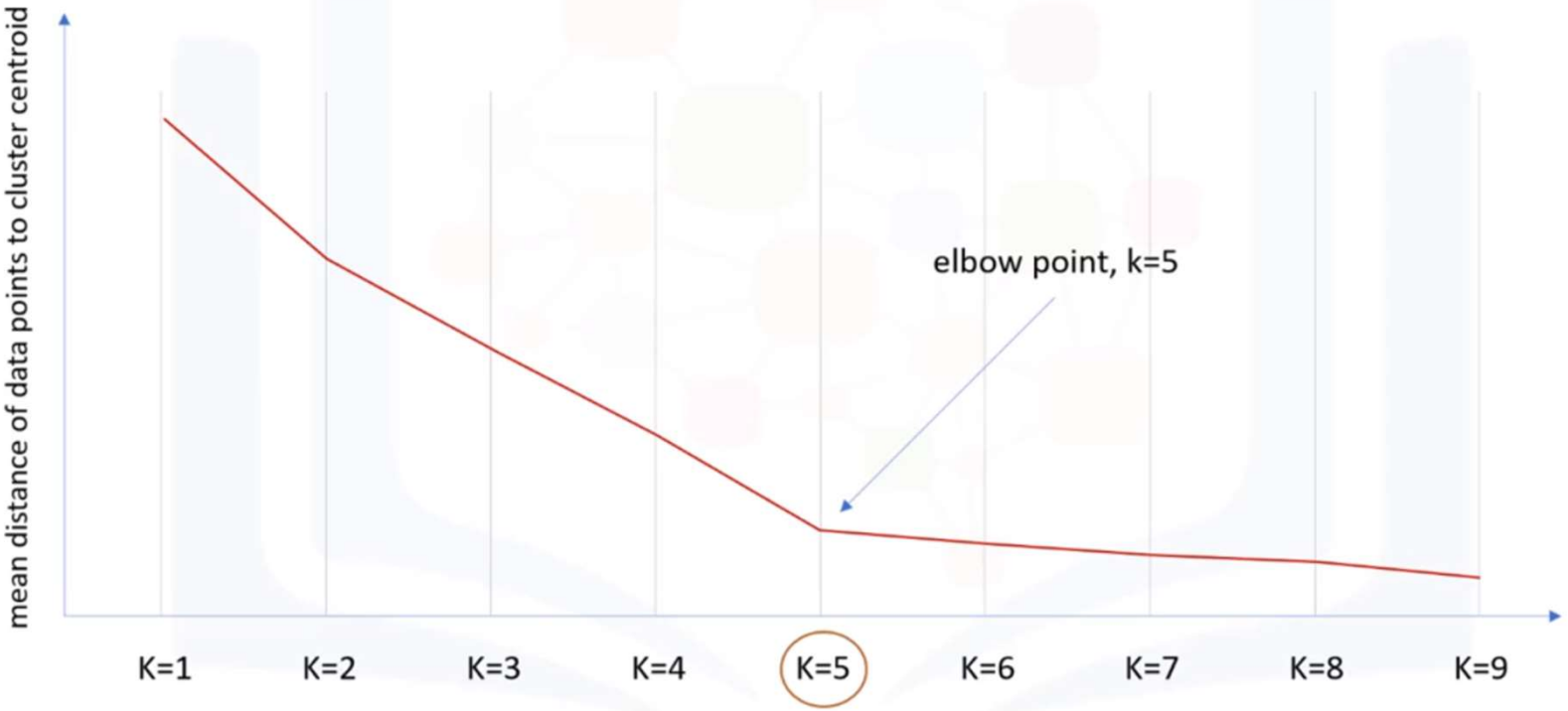
4. 각 cluster 의 new centroid 계산



5. Centroid 변화 없을 때까지 반복



Choosing k



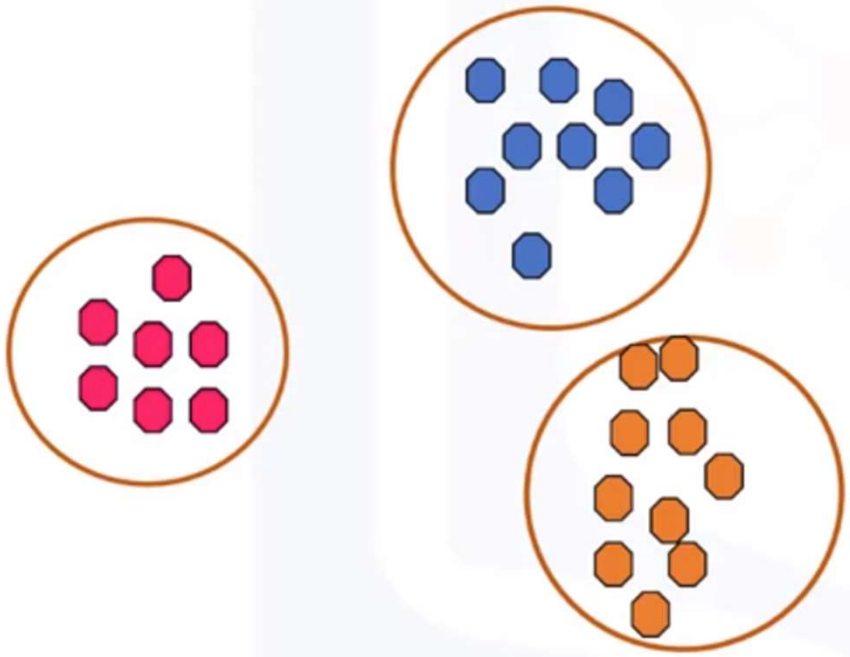
K 를 잘 정하는 것이 중요하다.

DBSCAN 알고리즘

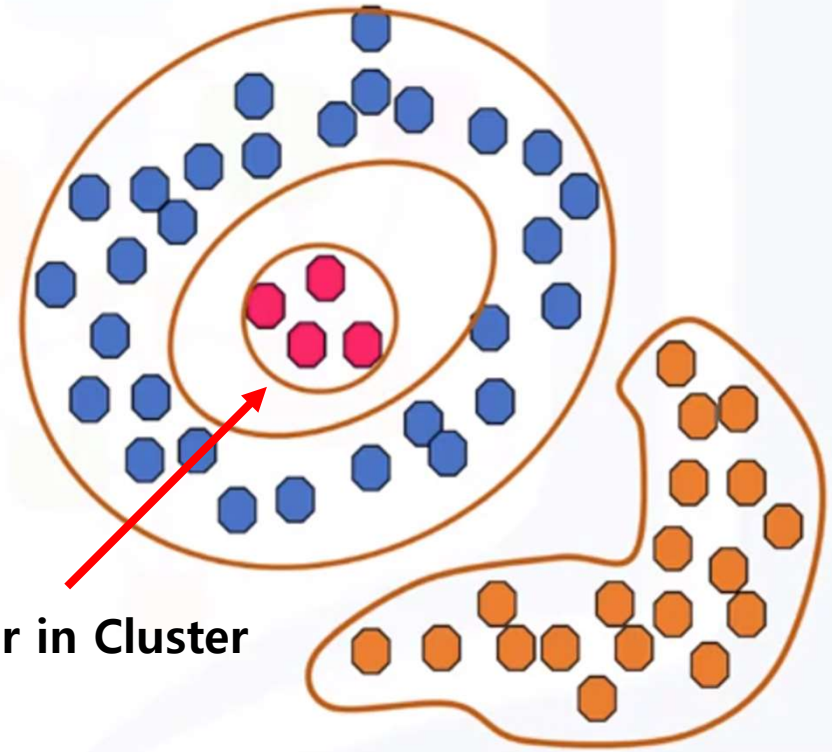
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- K-Means 의 경우 임의로 cluster 지정하므로 same cluster 내의 data point 들이 실제로는 유사하지 않을 수 있다.
- DBSCAN 은 밀도가 높은 지역과 낮은 지역을 서로 분리 (밀도 – 특정 반경내의 data point 숫자)
- Outlier 의 영향을 적게 받고, cluster 숫자를 미리 정해주지 않아도 되는 것이 장점
- Slower than K-Means
- 밀도 차이가 큰 경우 잘 작동 않음

K-Means vs. DBSCAN

- Spherical-shape clusters



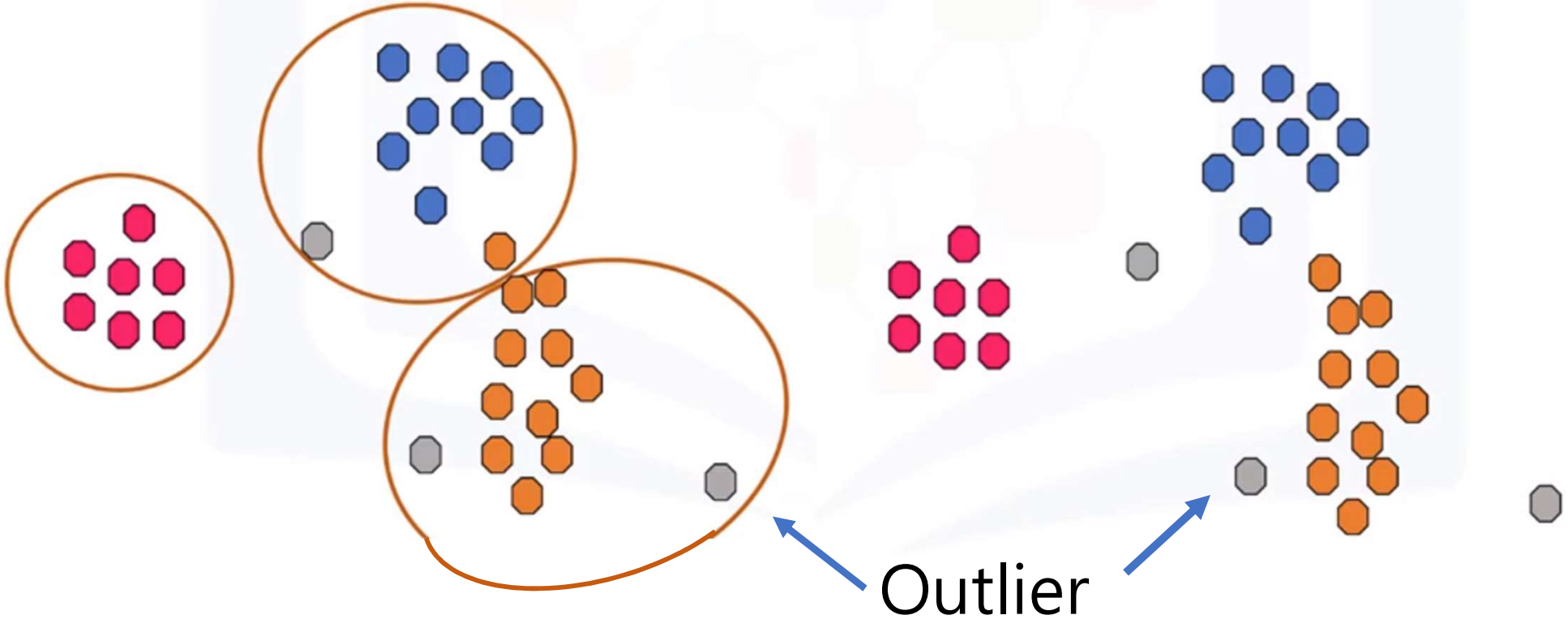
- Arbitrary-shape clusters



K-Means vs. DBSCAN

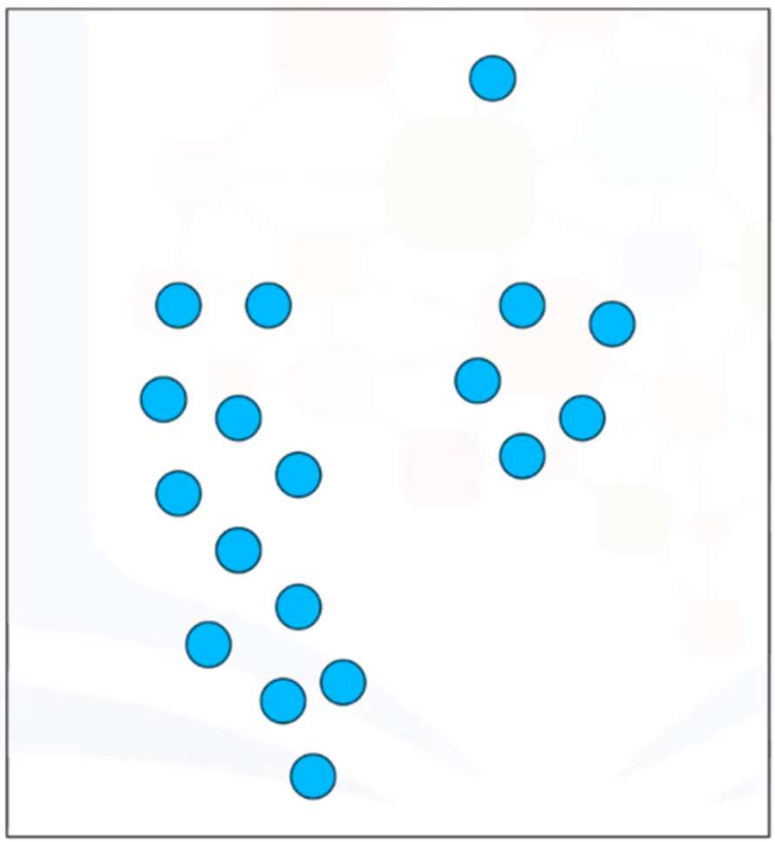
Anomaly detection (이상감지) 불가능

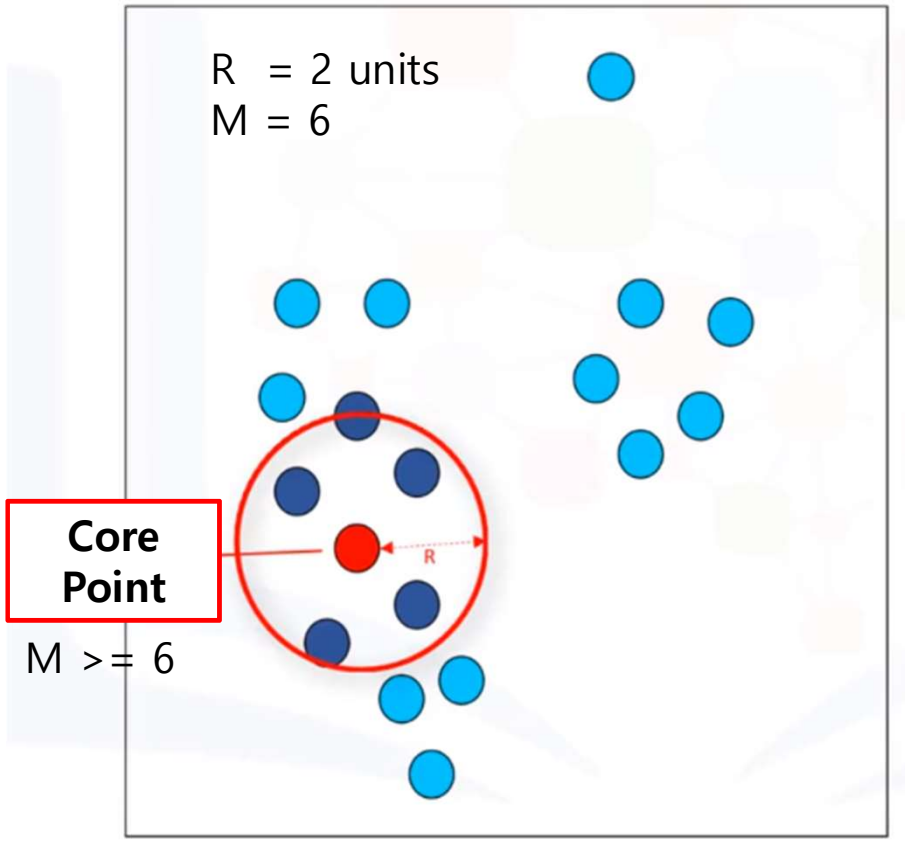
Outlier 를 쉽게 detect



DBSCAN 알고리즘

1. Radius (R) , Minimum Neighbor number (M) 지정

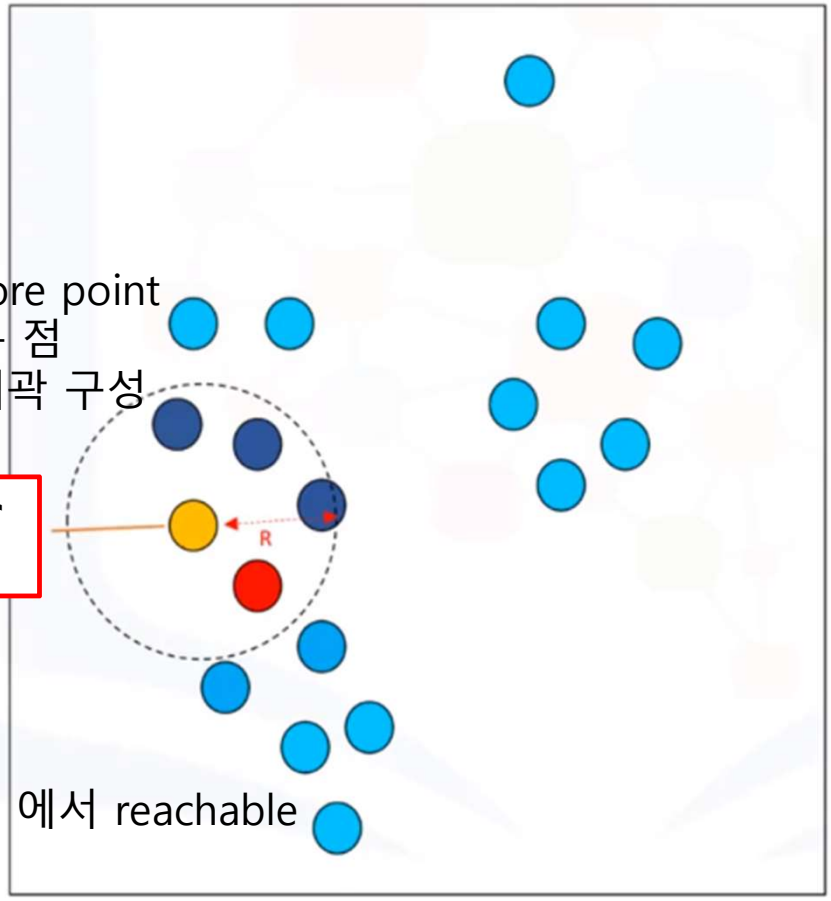


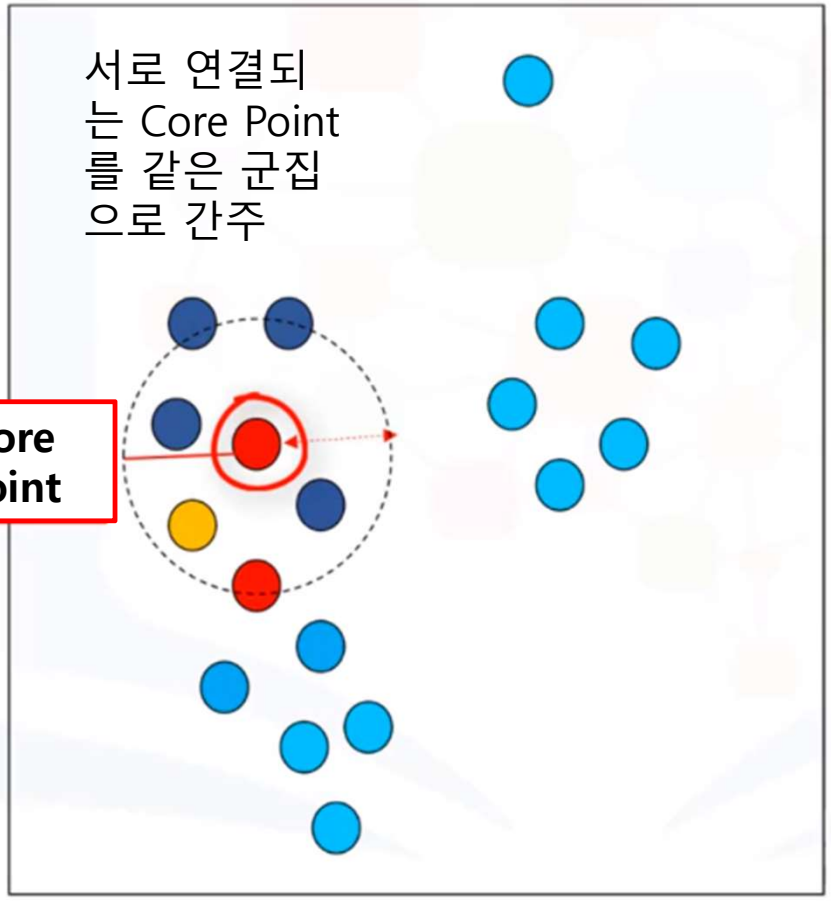


2. 각 point 를 Core, Border, Outlier 로 구분

R = 2 units
M = 6

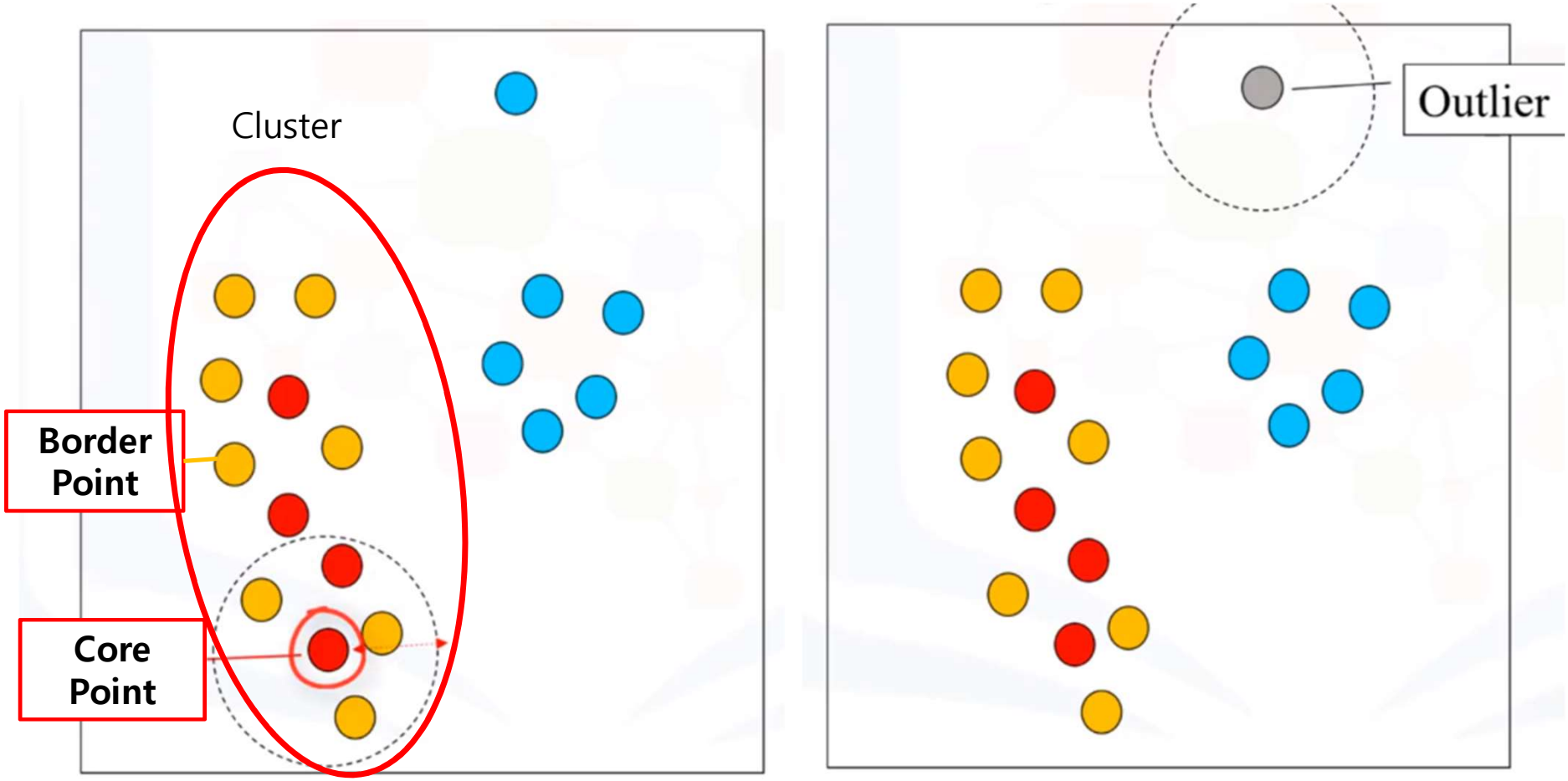
- 스스로 core point 가 안되는 점
- 군집의 외곽 구성





3. 모든 point 에 대해 동일한 과정 반복

R = 2 units
M = 6



실습: K-Means and DBSCAN

1. sklearn.cluster.Kmeans & sklearn.cluster.DBSCAN

2. Dataset : sklearn.dataset.samples_generator.make_blobs 사용

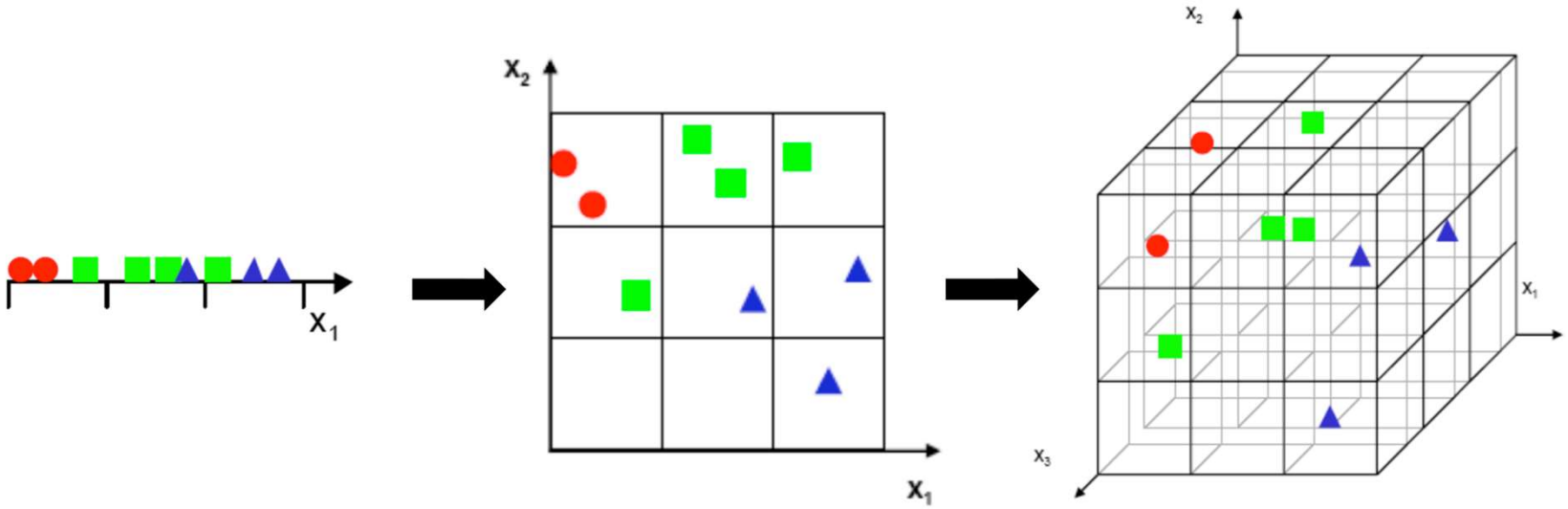
3. Matplotlib 을 이용하여 시각화

차원축소

차원의 저주 (Curse of Dimensionality)

- 차원이 증가함에 따라 vector 공간내의 space 도 증가하는데 데이터의 양이 적으면 빈공간이 많이 발생하여 예측의 정확도가 떨어진다.

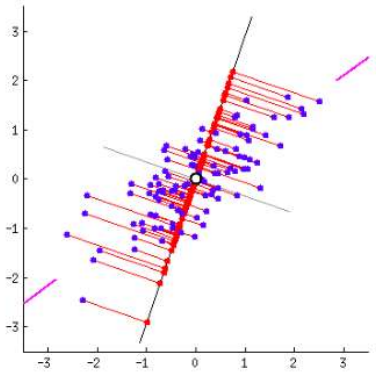
- 유사한 성격의 feature (예, 키, 신장, 앉은키, 기온, 수도관 동파, 빙판길 미끄러짐 사고 등) 는 하나의 새로운 feature 로 성분을 합칠 수 있음



PCA(Principal Component Analysis) 의 수학적 기초

- 정사영

- N 차원 공간에 있는 데이터들을 하나의 주성분(PC)을 새로운 기저로 하여 **선형변환**



핑크색 표시가 돼 있는 사선 축이 원 데이터의 **분산을 최대한 보존**하는 (=데이터가 가장 많이 흩뿌려져 있는) 새 기저

➔ 공분산 행렬의 Eigenvector (고유벡터)

https://i.imgur.com/Uv2dlsH.gif (예. 2차원을 1차원으로 선형변환)

공분산 행렬 (Covariance Matrix)

- 다른 변수 (feature) 사이의 관계를 수치적으로 표현

  → 어떤 변수가 평균으로 부터 증가, 감소 형태를 보일 때, 이런 경향을 다른 변수가 따라하는 정도를 수치화

$$\begin{bmatrix} VAR(X_1) & COV(X_1, X_2) & \cdots & COV(X_1, X_N) \\ COV(X_2, X_0) & VAR(X_2) & \cdots & COV(X_2, X_N) \\ \vdots & \vdots & \ddots & \vdots \\ COV(X_N, X_1) & COV(X_N, X_2) & \cdots & VAR(X_N) \end{bmatrix}$$

	X	Y	Z
X	**2.0**	-0.86	-0.15
Y	-0.86	**3.4**	0.48
Z	-0.15	0.48	**0.82**

- 평균(μ) – 확률변수의 기대값
- 분산(variance) – 확률변수가 평균으로부터 얼마나 넓게 퍼져 있는지의 정도 (σ^2)
- 표준편차(σ) - 분산의 제곱근

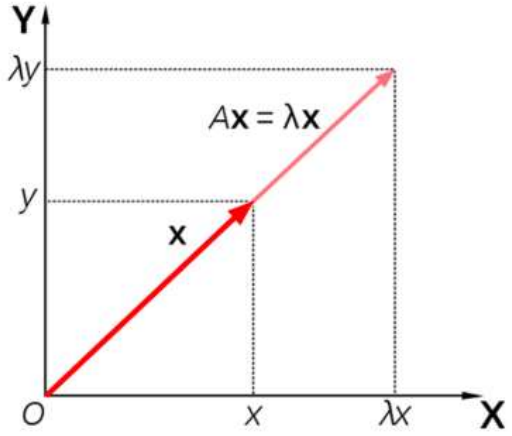
- 고유벡터(eigenvector) / 고유값(eigenvalue)

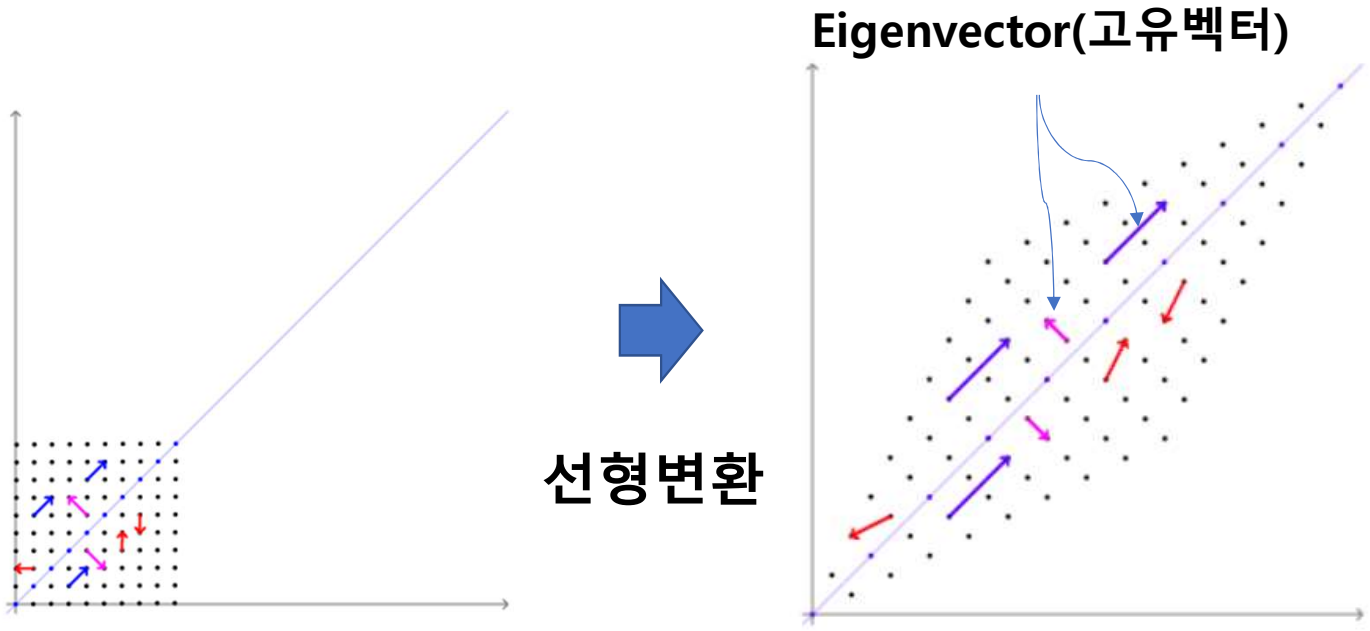
  - 선형변환 A에 의한 변환 결과가 자기 자신의 상수배가 되는 0이 아닌 벡터를 고유벡터(eigenvector)라 하고 이 상수배 값을 고유값(eigenvalue)라 한다.

  - 행렬(선형변환) A의 **고유벡터**는 선형변환 A에 의해 **방향은 보존**되고 스케일(scale)만 변화되는 **방향**

$$Av = \lambda v$$

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \lambda \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$





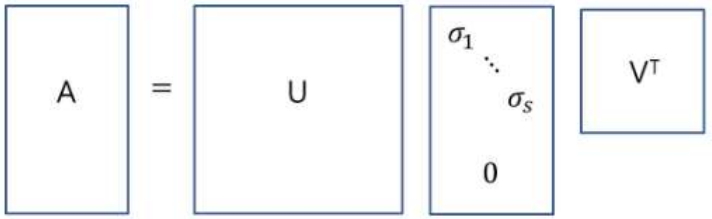
크기만 변하고 방향은 일정

- Vector - 방향과 크기로 결정되는 벡터 공간'(Vector space)의 원소
- 행렬(matrix) – vector 의 집합 (행벡터, 열벡터)

- 특이값 분해 (Singular Value Decomposition, SVD)

  - 모든 m x n 행렬에 대해 적용 가능 (고유값 분해는 정방행렬에만 제한)
  - 어떤 m x n 행렬 A 는 다음과 같은 형태의 세가지 행렬의 곱으로 분해 할 수 있다.

$$A_{mXn} = U_{mXm} \sum_{mXn} V^{T}_{nXn}$$



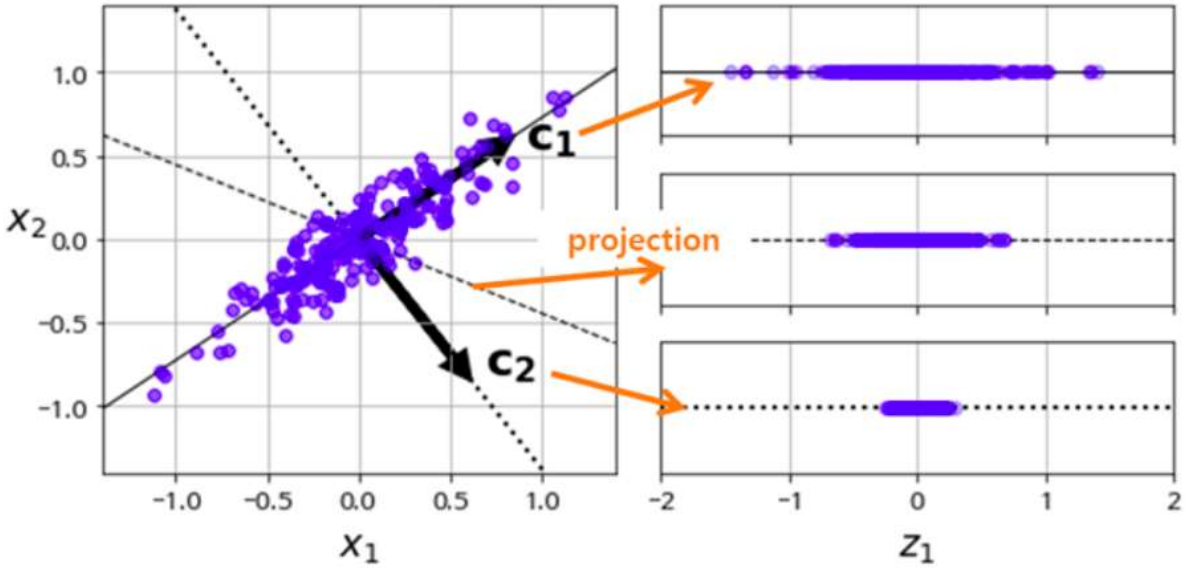
U : AA^T를 고유값 분해(eigendecomposition)해서 얻어진 m x m 직교행렬(orthogonal matrix)

V : A^TA를 고유값 분해해서 얻어진 n x n 직교행렬(orthogonal matrix) - **주성분 column 들로 구성**

Σ : AA^T, A^TA를 고유값 분해해서 나오는 고유값(eigenvalue)들의 square root($\sqrt{\lambda_i}$)를 대각원소로 하는 m x n 직사각 대각행렬로 그 대각원소들을 A의 특이값(singular value)이라 부른다

PCA (Principal Component Analysis) - 주성분 분석

- **선형대수학의 SVD** 를 이용하여 **분산이 최대인 축**을 찾음
- 데이터의 **분산(variance)**을 최대한 보존하면서 서로 직교하는 새 기저(축)를 찾아, 고차원 공간의 표본들을 선형 연관성이 없는 저차원 공간으로 변환



분산을 최대한 보존

실습: PCA

1. sklearn.decomposition 의 PCA 를 이용하여 차원 축소
2. Dataset : 통신회사 고객 이탈 (Churn) data 이용
3. 27 개의 feature 를 2 개로 reduce 한 후 2 개의 feature 를 이용하여 Logistic Regression
4. Matplotlib 을 이용하여 시각화