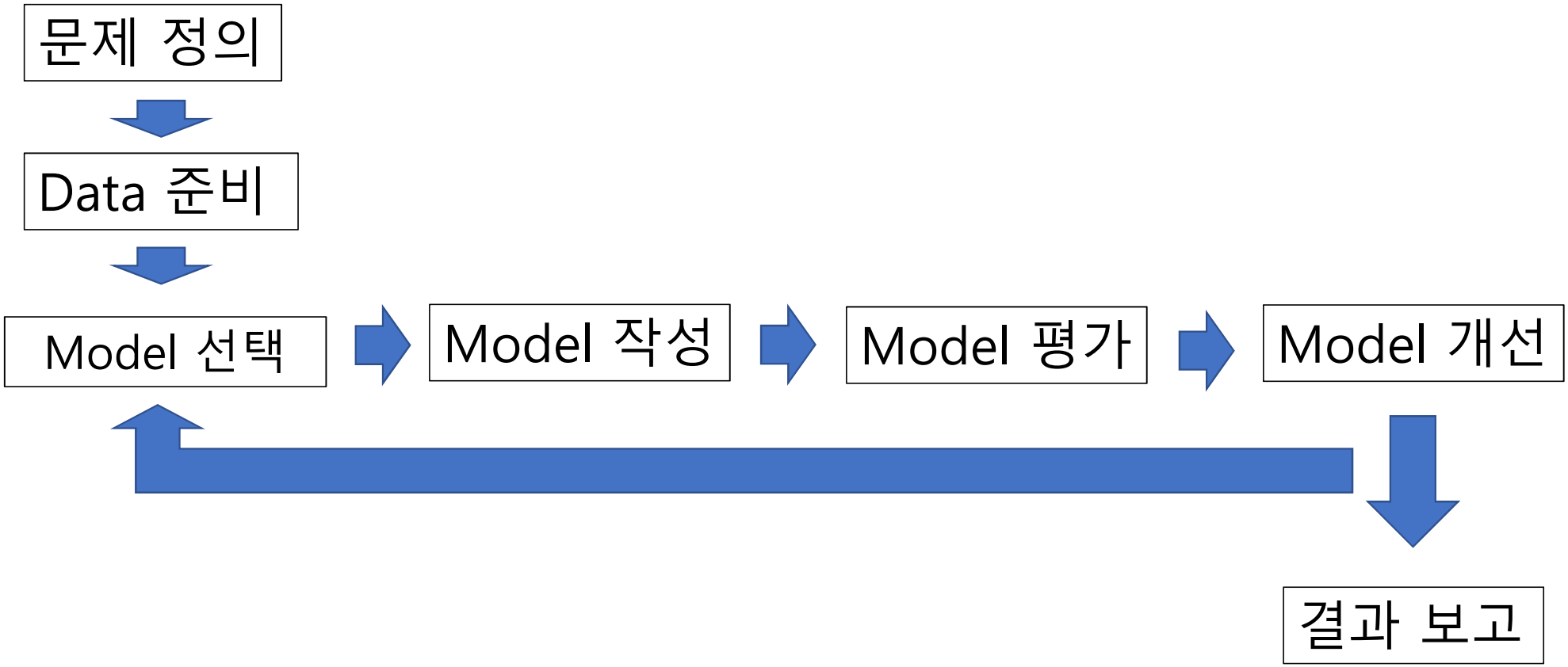
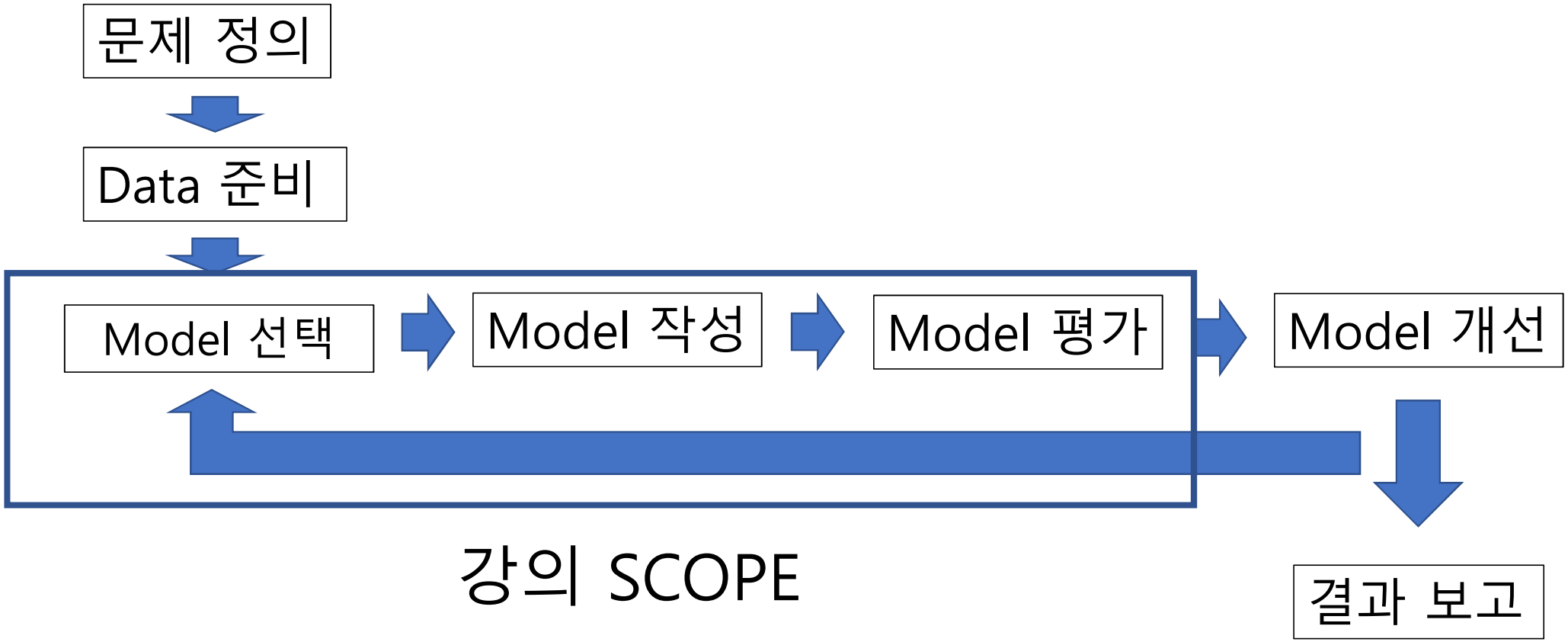
Machine Learning

End-to-End Process

End-to-End of Machine Learning

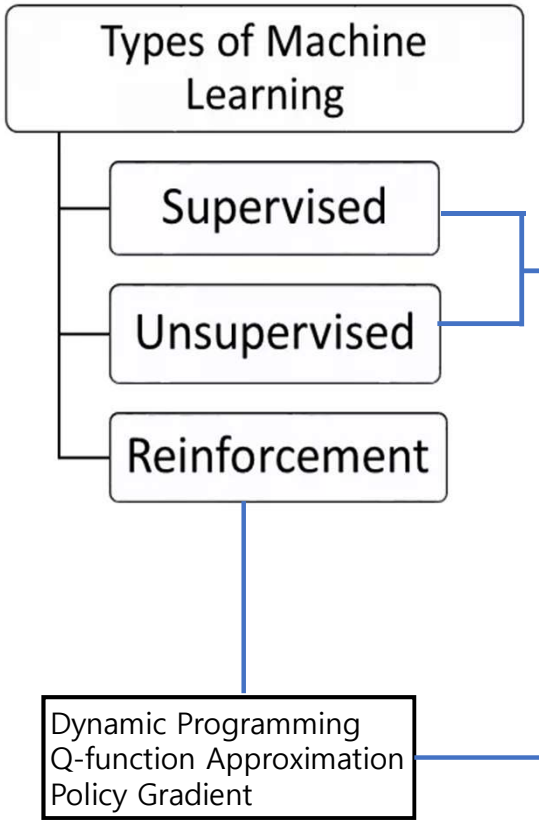


End-to-End of Machine Learning



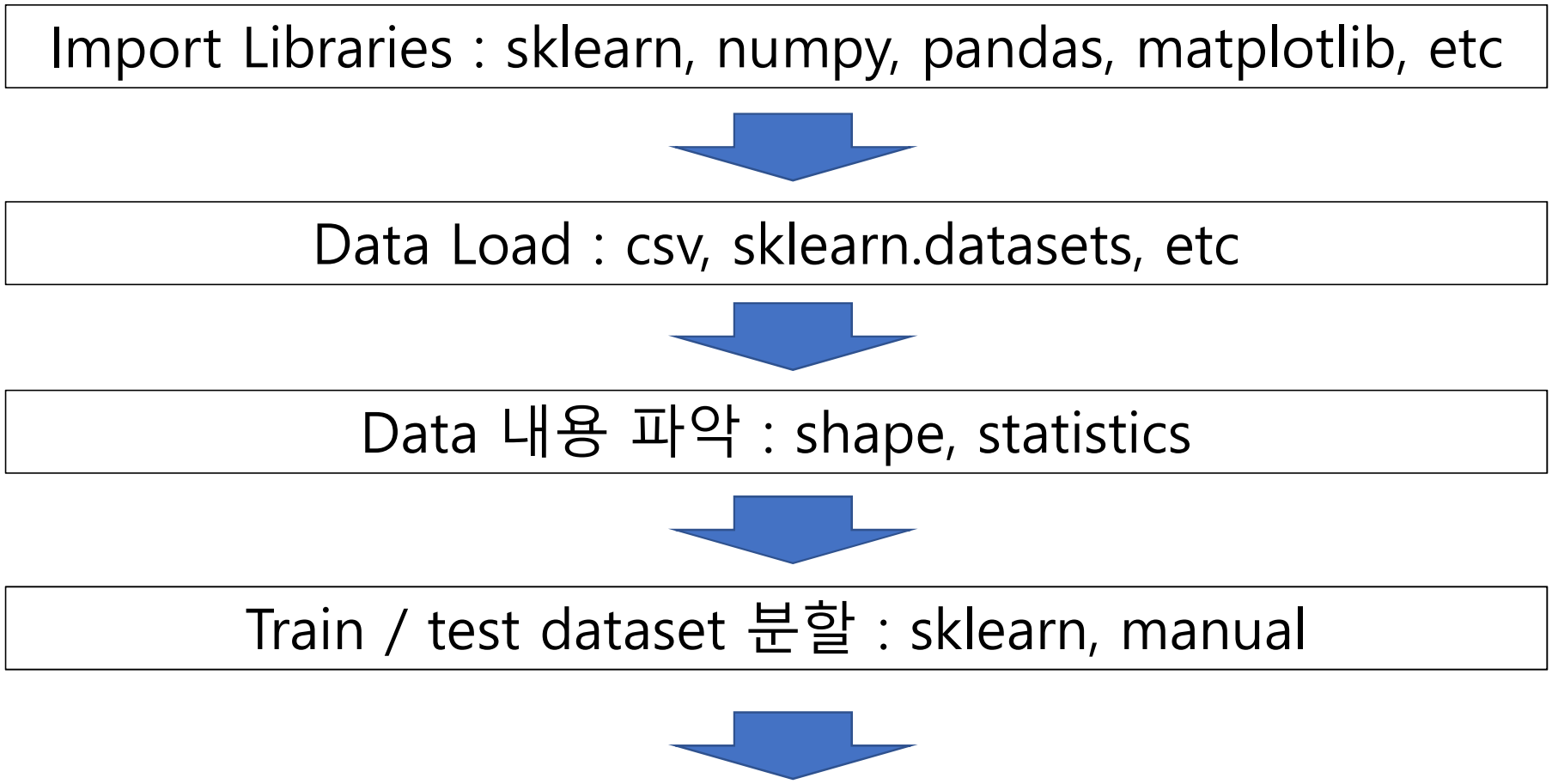
강의 SCOPE

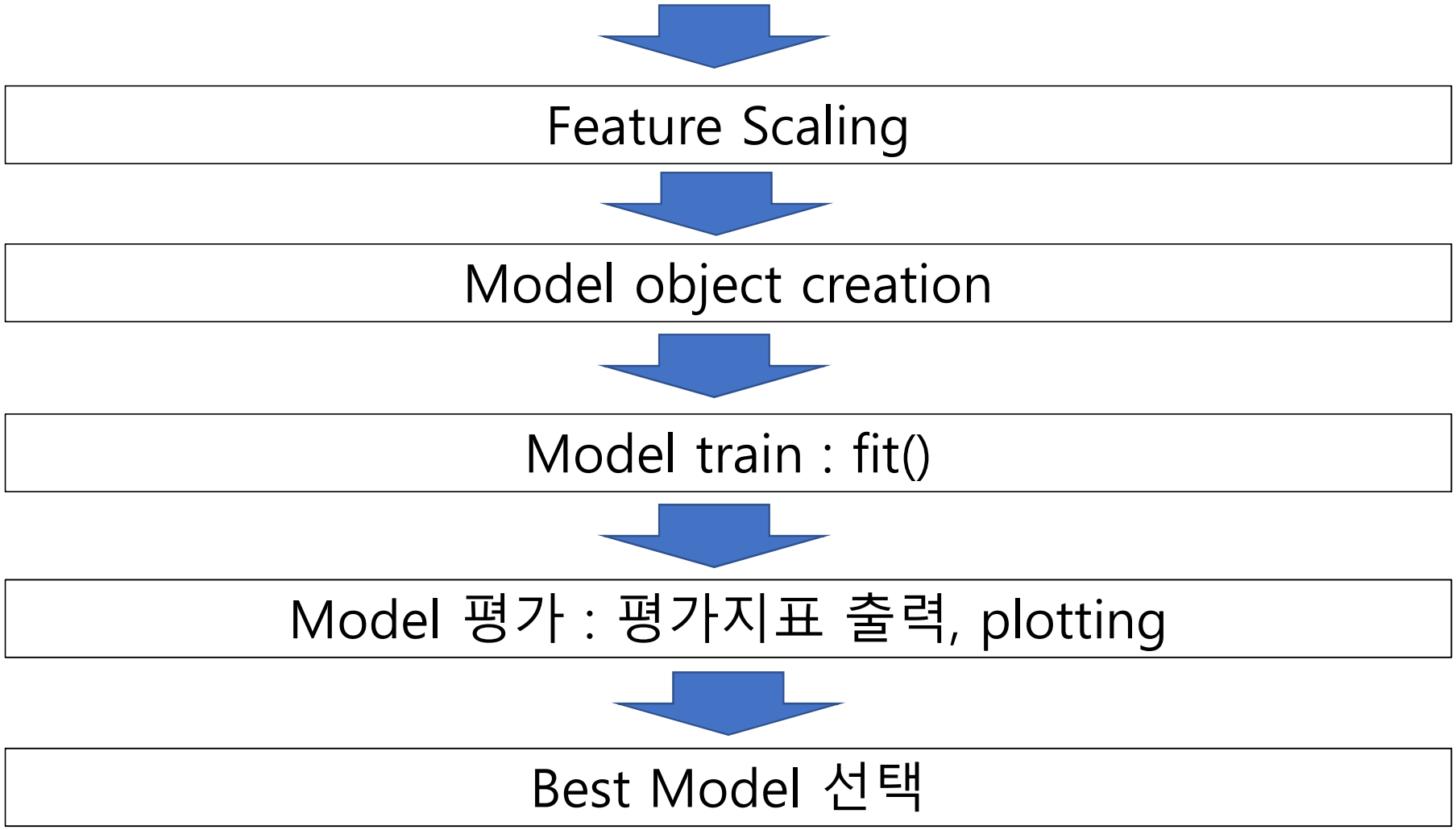
Model 선택



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
Data 내용 파악 : shape, statistics
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
2. Multivariate Linear Regression
3. Polynomial Regression

1. Univariate Linear Regression (단변수선형회귀)

y = mx + b → Hypothesis

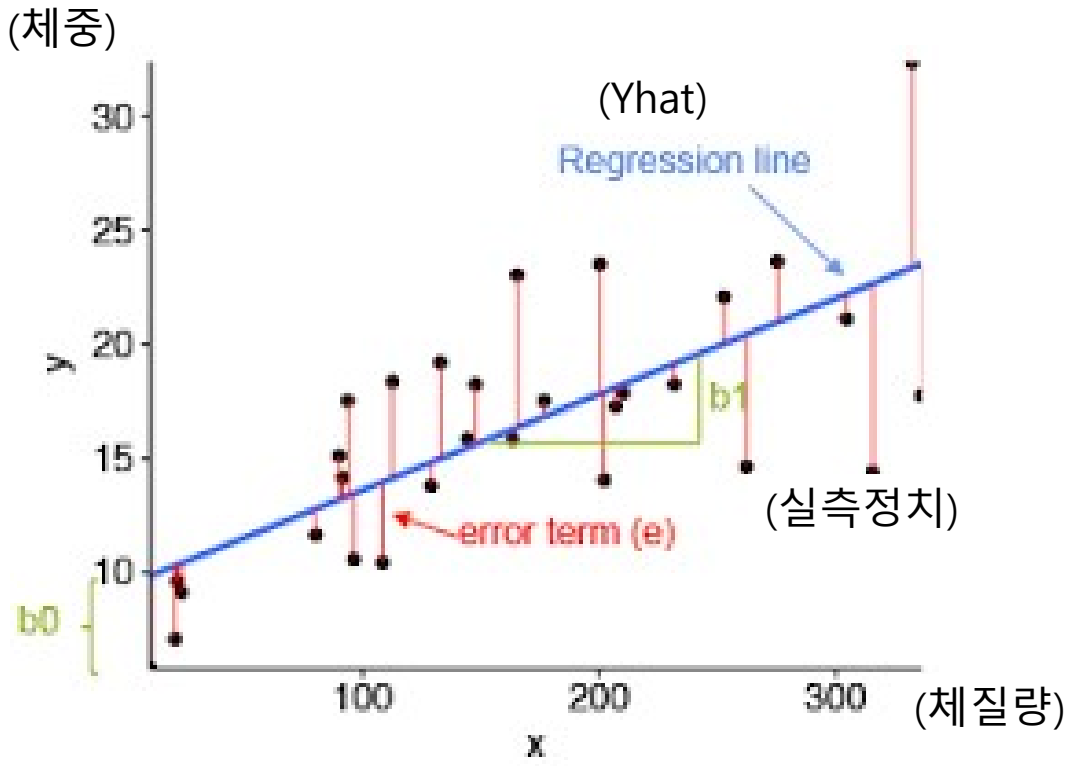
OLS (Ordinary Least Squares, 최소자승법)

Minimize sum(prediction – actual)

$$\min sum(|yhat - y|^2)$$

sklearn.linear_model.LinearRegression()

m = coef_
b = intercept_



2. Multivariate Linear Regression (다변수선형회귀)

$Yhat = \theta_0 + \theta_1 X1 + \theta_2 X2 + \theta_3 X3 + ... + \theta_n Xn$

$Yhat = \theta X$

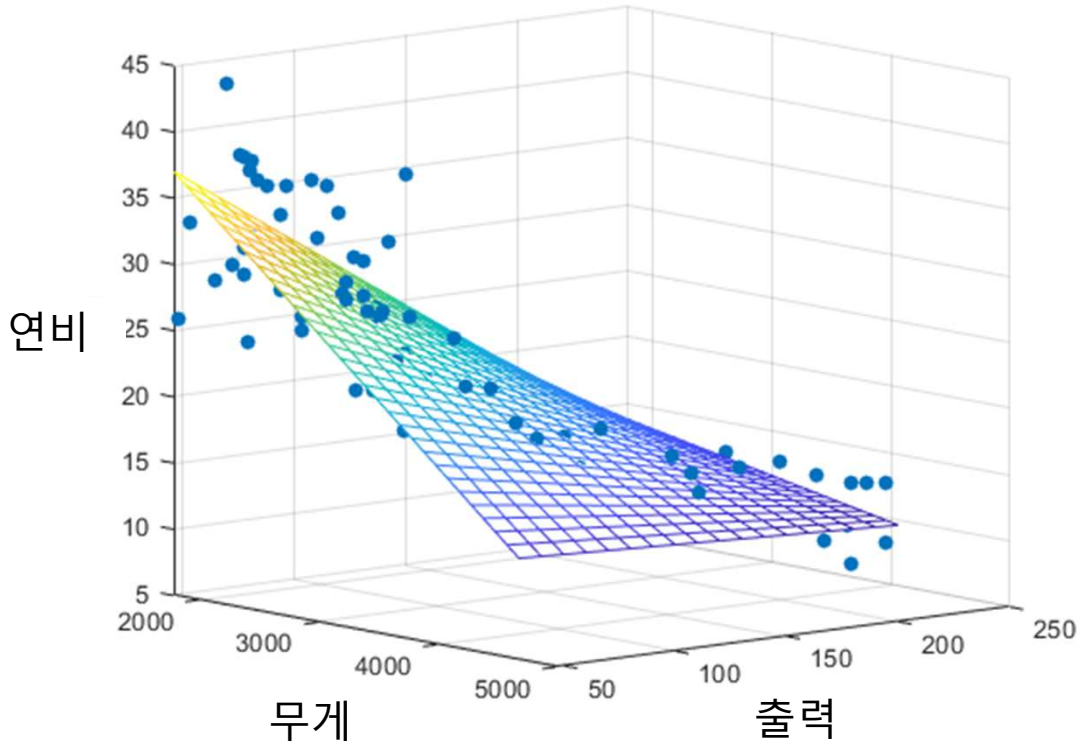
Minimize(prediction – actual)

$\min\ sum(|yhat - y|^2)$

sklearn.linear_model.LinearRegression()

θ = coef_

θ_0 = intercept_



3. Polynomial Regression (다항회귀)

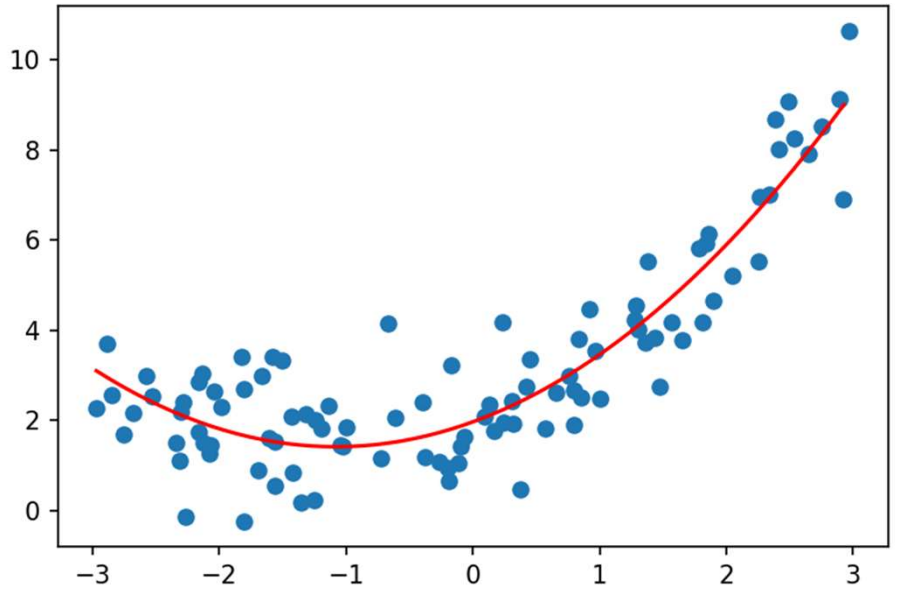
$$y = b_0 + b_1 x^3 + b_2 x^2 + b_3 x$$

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
- Target : 1년 뒤 측정한 당뇨병의 진행률
- Model : OLS (Ordinary Least Squares)
- 평가기준 1 : MSE (Mean Squared Error)
  sklearn.metrics.mean_squared_error

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

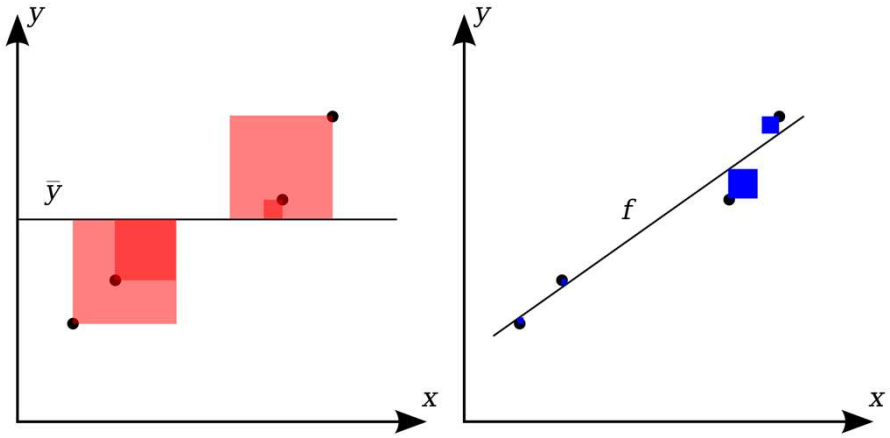
실습: 당뇨병 data 를 이용한 선형회귀

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

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

  $R2 = 1 - SSE / SST$

  (Sum of Square Error / Sum of Square Total)



R2 ≈ 0　　R2 ≈ 1

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

실습: toy data 를 이용한 비선형회귀

- 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)

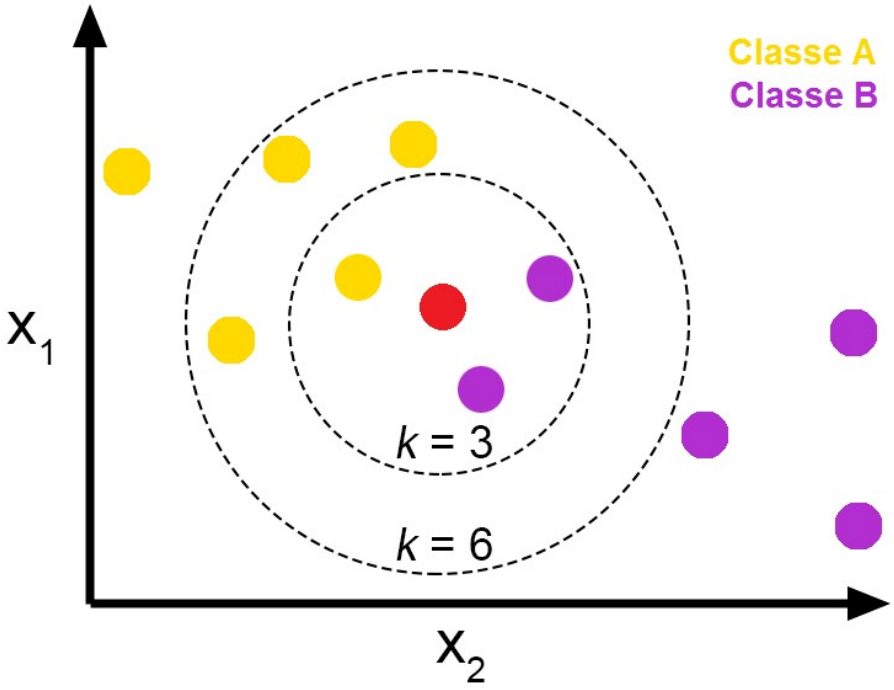
- Target : y = 0.5 * X**2 + X + 2 + np.random.randn(m, 1)

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

K-Nearest Network

4. 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.

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 분류한다.

실습 :

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

   꽃잎의 각 부분의 너비와 길이등을 측정한 데이터이며 150개의 레코드로 구성 되어 있다

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

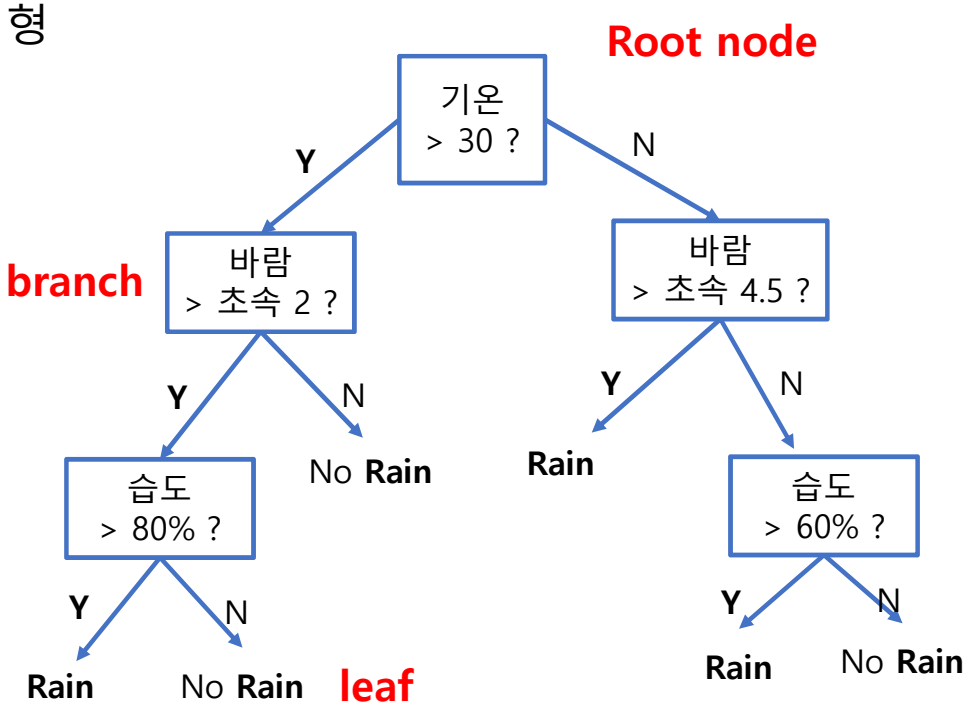
distance – 가까운 neighbor 에 더 높은 가중치 부여

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

Decision Tree

4. 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

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

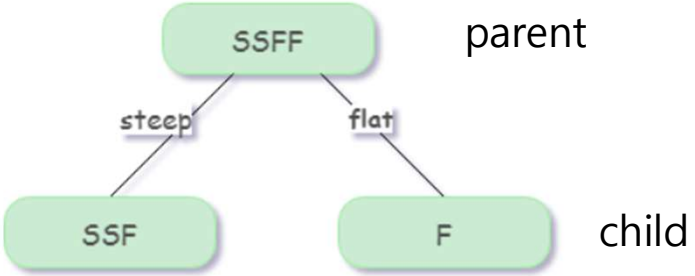
ex)

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

경사도의 information gain 계산:

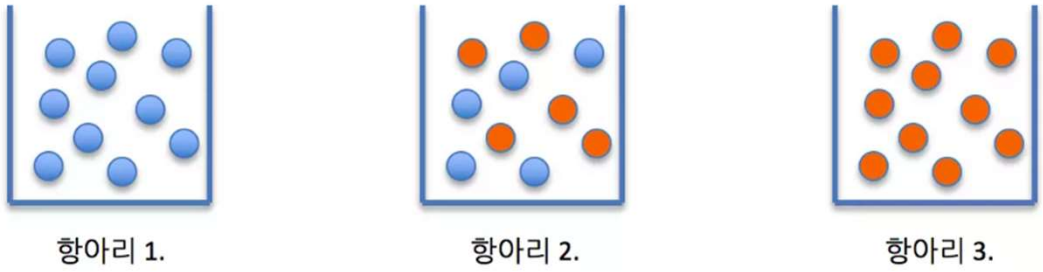
Entropy(Parent) = - {0.5 log2(0.5) + 0.5 log2(0.5)} = 1
Entropy(Child) = - {0.667 log2(0.667) + 0.334 log2(0.334)}
(가중평균) = ¾*0.918 + ¼*0

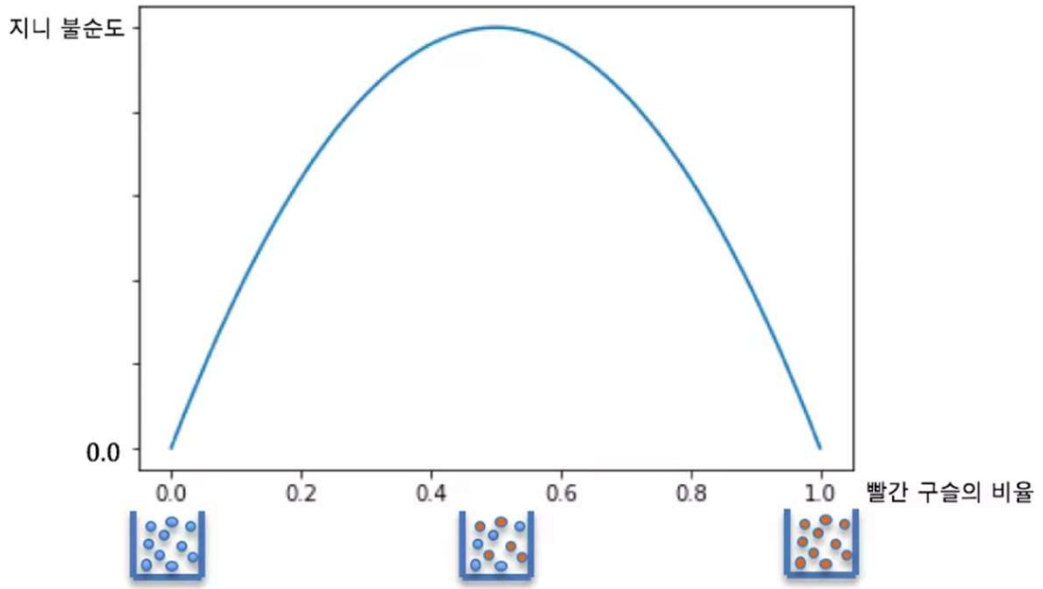
Information Gain = 1 – 0.688 = 0.312



- 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
   - 분류의 불확실성이 최소화 되는 (information gain 이 최대인) attribute 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 변경 test

3. graphviz 를 이용한 visualization

Graphviz 설치 방법

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

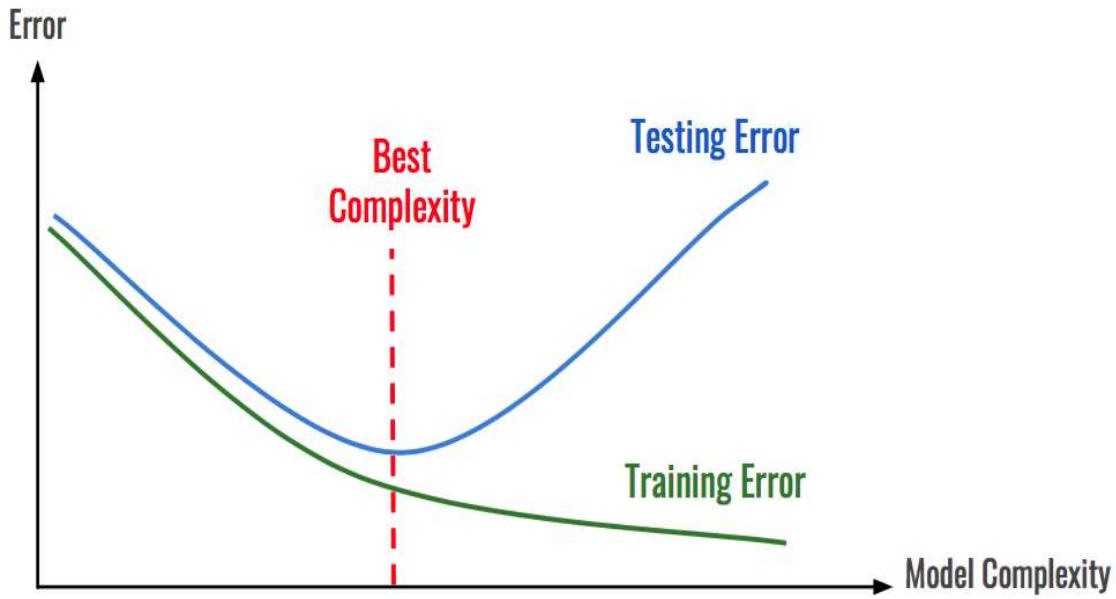
   >pip install pydotplus

Training/Test/Evaluation

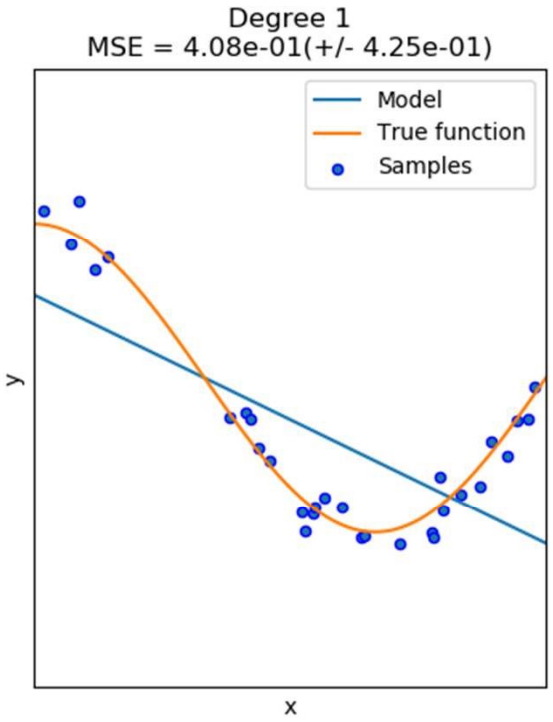
Overfitting(과적합) 과 Underfitting(과소적합)

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

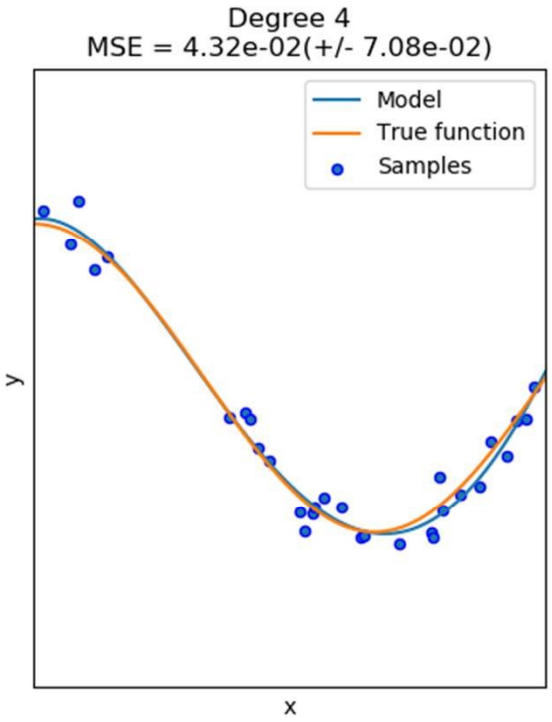
- 반대로 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못하는 경우 과소적합 (Underfitting) 이라고 한다.



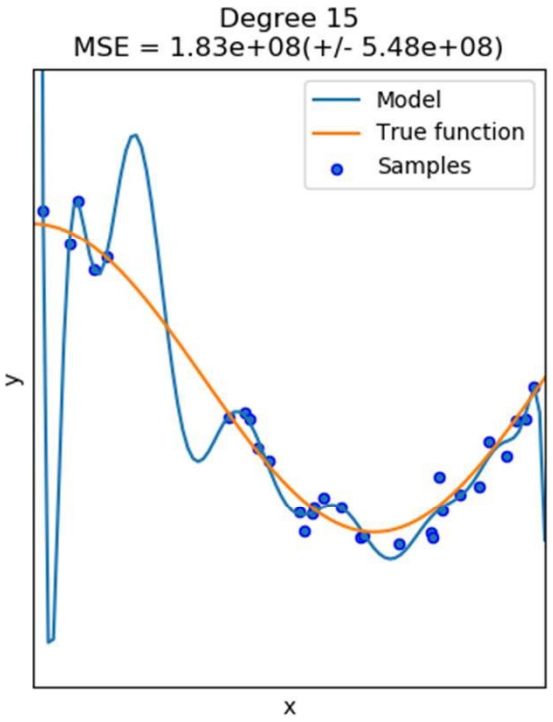
Underfitting
(과소적합)



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

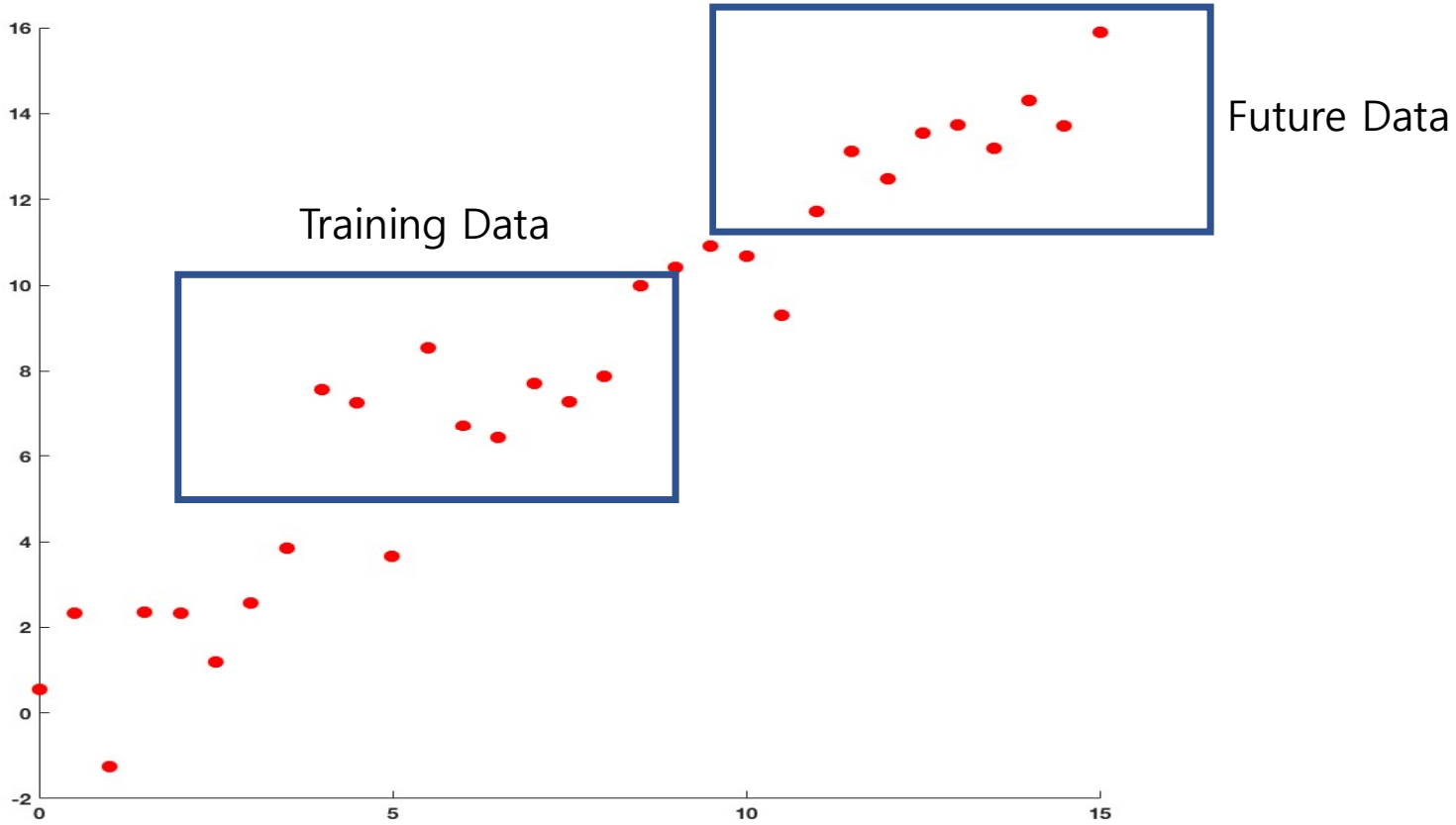
  1. 학습 Data 와 실제 data 분포의 차이에 의한 error → Variance

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

  3. Noise 에 의한 error → 제거할 수 없음

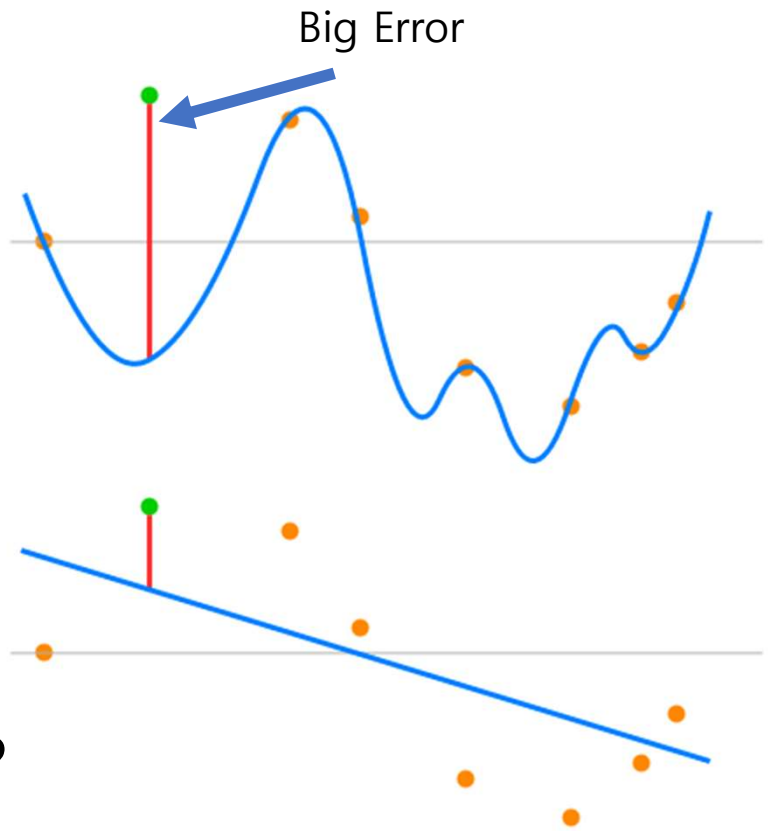
- Variance 를 줄이려면 Dataset 의 크기를 늘이고, Bias 를 줄이려면 모델의 Complexity 를 올린다.
  → Bias-Variance Dilemma (Bias-Variance Trade-off)

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

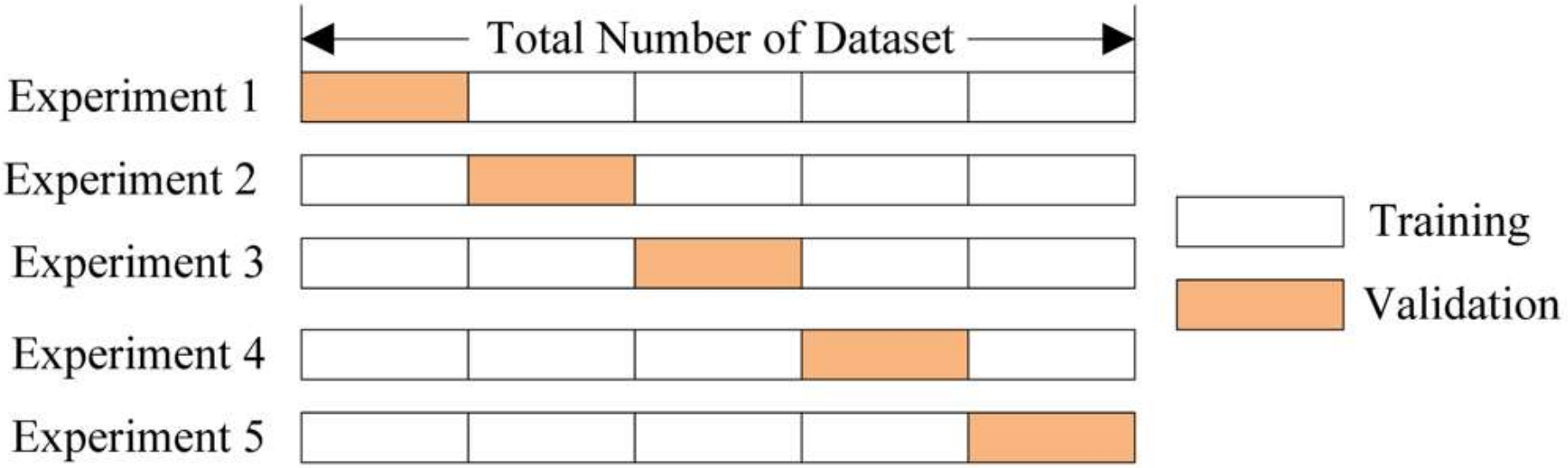
- Training
  - Parameter 를 inference 하는 procedure
  - 보지 못한 Data 의 분포가 Training set 과 상이할 경우 문제
  - Training set 내에서 cross-validation set 을 구성 (Data 가 충분한 경우)

- Testing
  - 학습한 Machine Learning Model 을 Test
  - 미래의 instance 인 것처럼 간주

- Training set 과 Testing set 은 섞이면 안되고 동일한 분포를 유지해야함.

Cross Validation (교차검증)

- 훈련세트를 여러 개의 sub-set 으로 나누고 각 모델을 이 sub-set 의 조합으로 훈련시키고 나머지 부분으로 검증



Confusion Matrix

		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 (정밀도)

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

Recall (재현율)

		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

- Classification rate = (TP + TN) / (TP+TN+FP+FN)

- Precision = TP / (TP + FP) → positive 분류의 정확성
  (ex. Spam 으로 예측한 것 중 실제 spam의 비율, Security check 통과 여부)

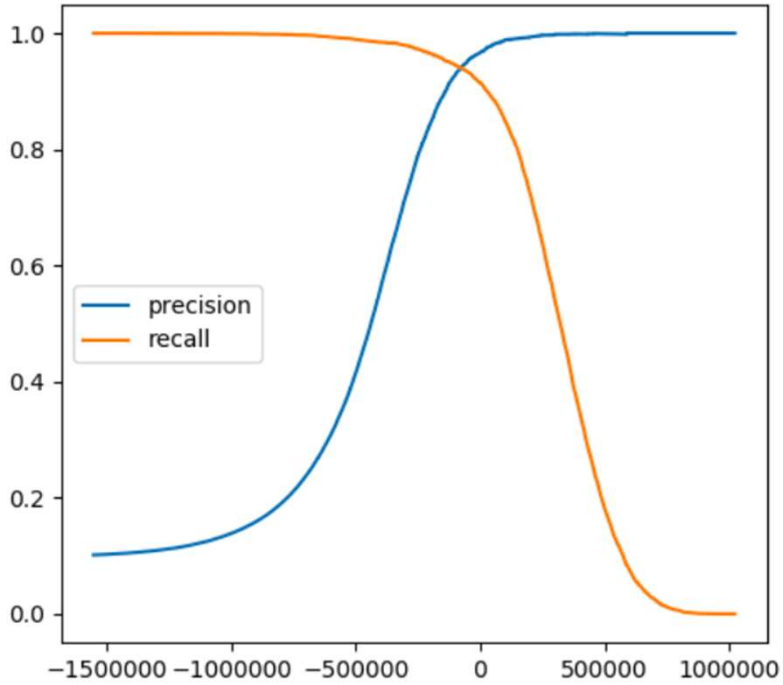
- Recall = TP / (TP + FN) → positive 구분 성능
  (ex. 실제 악성 샘플 중 악성으로 예측한 비율, VIP 고객 자동 선정)

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

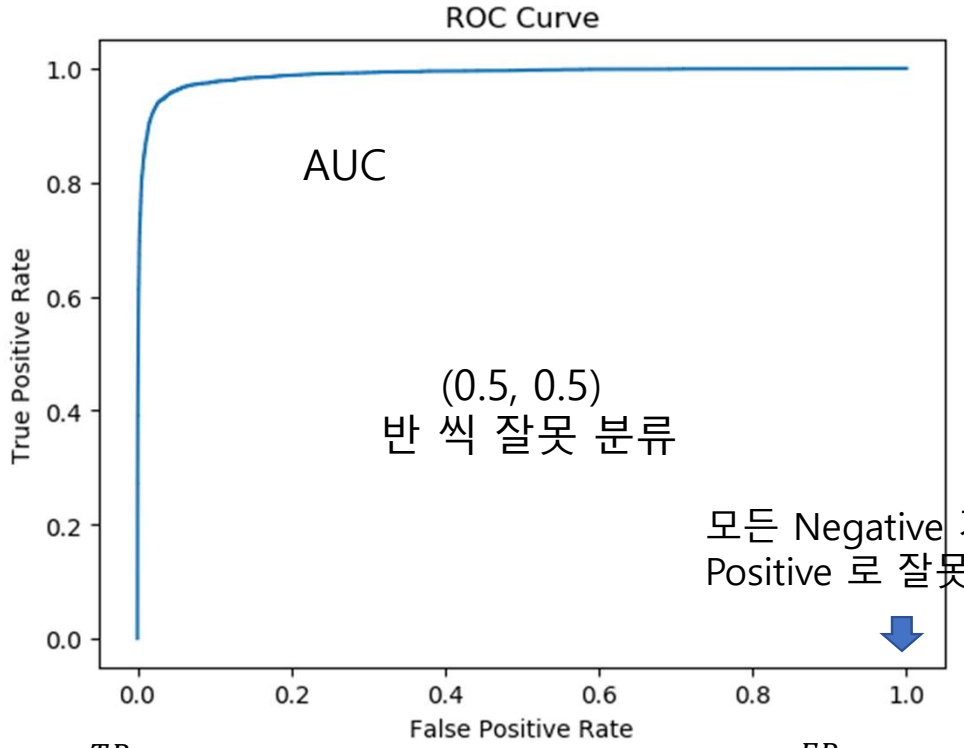
- F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

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

ROC Curve (수신자 조작 특성 곡선)



모든 Positive 를
Positive 로 정확히 분류



모든 Negative 가
Positive 로 잘못 분류

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

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

- 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 비교를 할 수 있다.

Overfitting 방지 기법 (Regularization)

- Linear Regression

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

손실함수 Train loss

λ : regularization 강도

Ω : regularizer

대표적 regularizer

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

Overfitting 방지 기법 (Regularization)

- Logitstic Regression

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

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

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

1. Training Set 과 Test set 으로 구분
   - 가장 일반적인 방법
   - sklearn.model_selection 의 train_test_split method 사용
   - sklearn.metrics 의 mean_squared_error, r2_score 로 평가

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

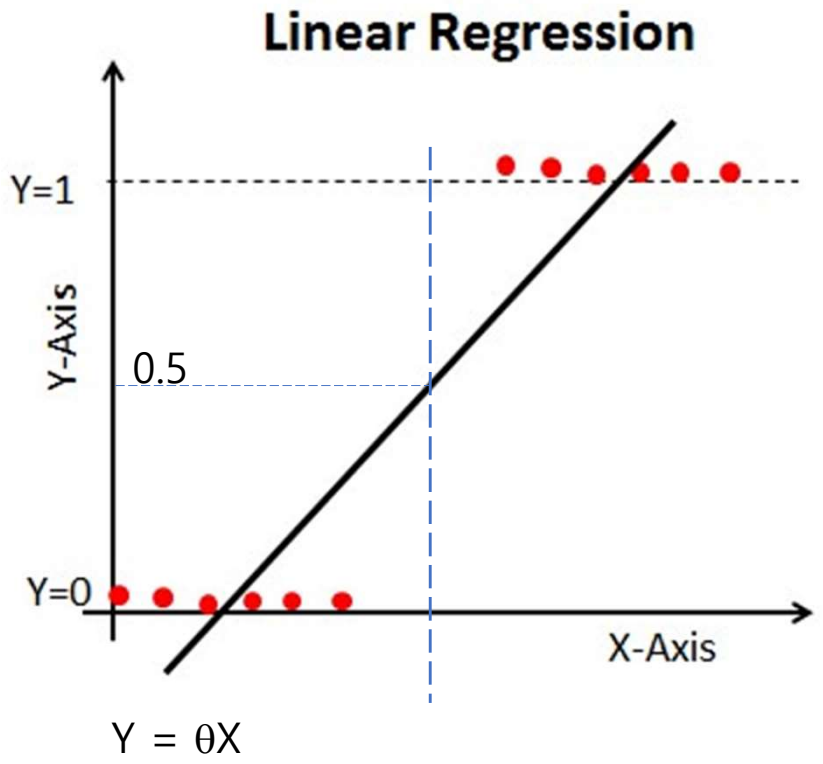
Logistic Regression

5. Logistic Regression (로지스틱 회귀)

- 선형회귀를 분류 문제에 적용 가능 ?

  yhat = 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

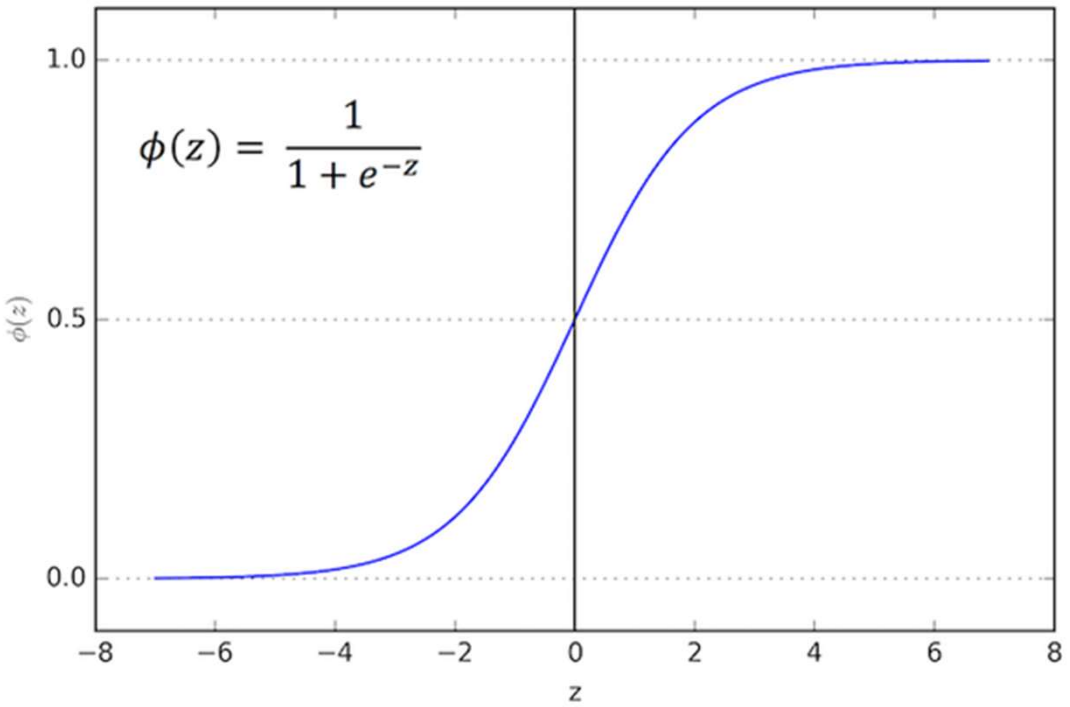


- Sigmoid 함수 : S curve 형성
- Logistic 함수

f(z) = 1 / (1+exp(-z)), z = θX

yhat = 0 if f(z) < 0.5
1 if f(z) ≥ 0.5

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



- Logistic Regression (로지스틱 회귀) classifier

1. 가장 단순한 분류기 → two-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

- 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

Support Vector Machine

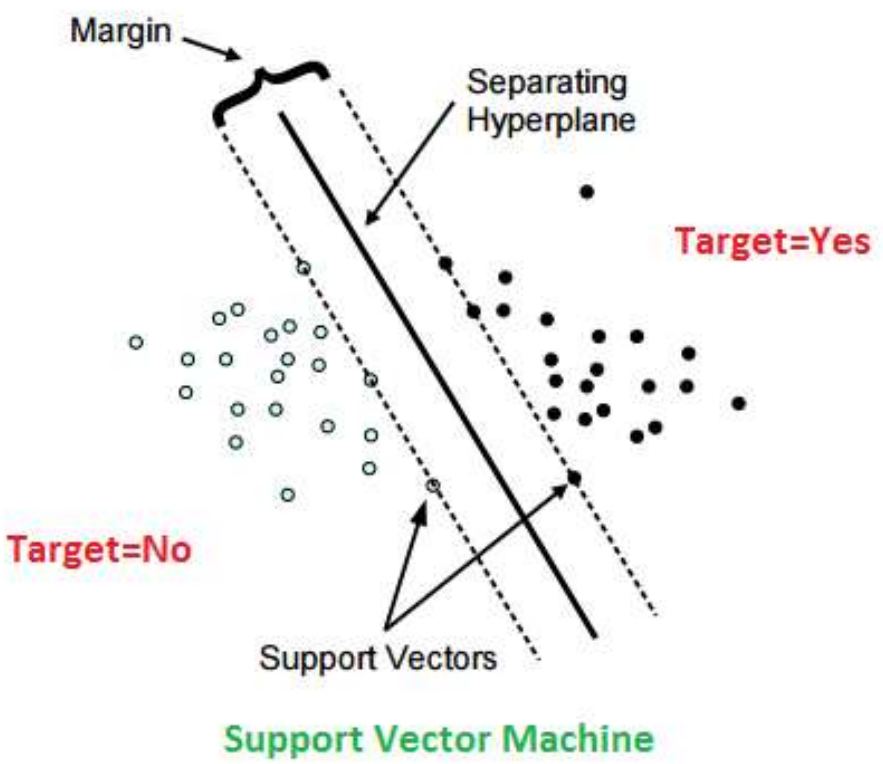
6. SVM (Support Vector Machine)

- Data 를 High Dimension feature space 에 mapping 하여 separate

- Support Vector 를 이용하여Data 사이를 분리하는 hyper-plane 유추

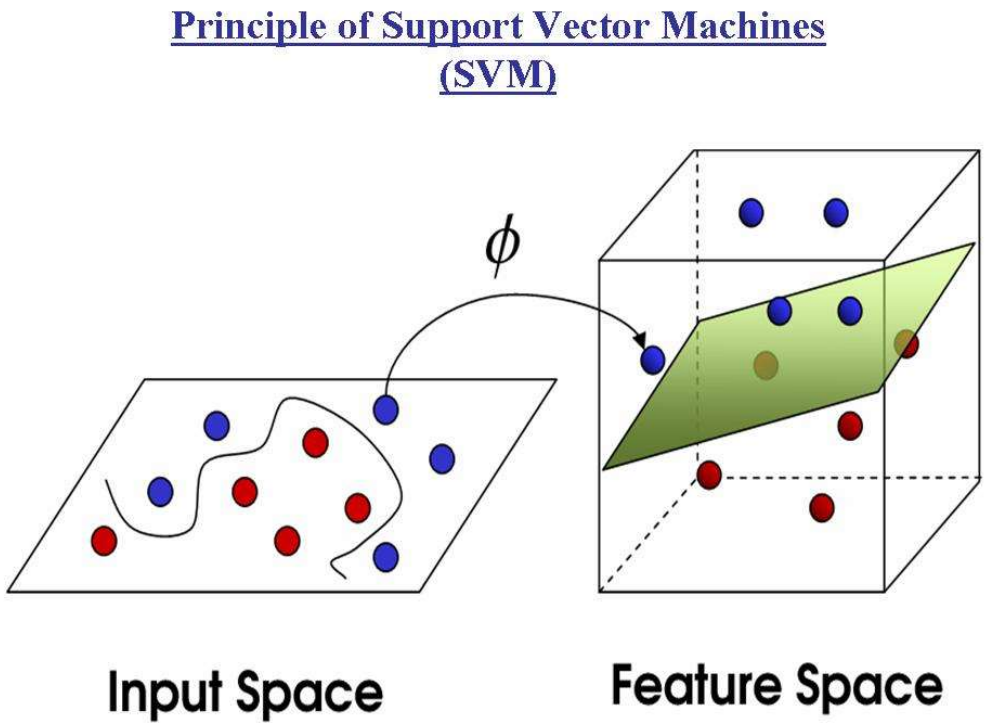
- 장점 – High Dimensional Space 에서 정확
  Memory efficient

- 단점 – Overfitting 되기 쉽다.
  No Probability Estimation (확률모델이 아님)
  Small Dataset 에 적합

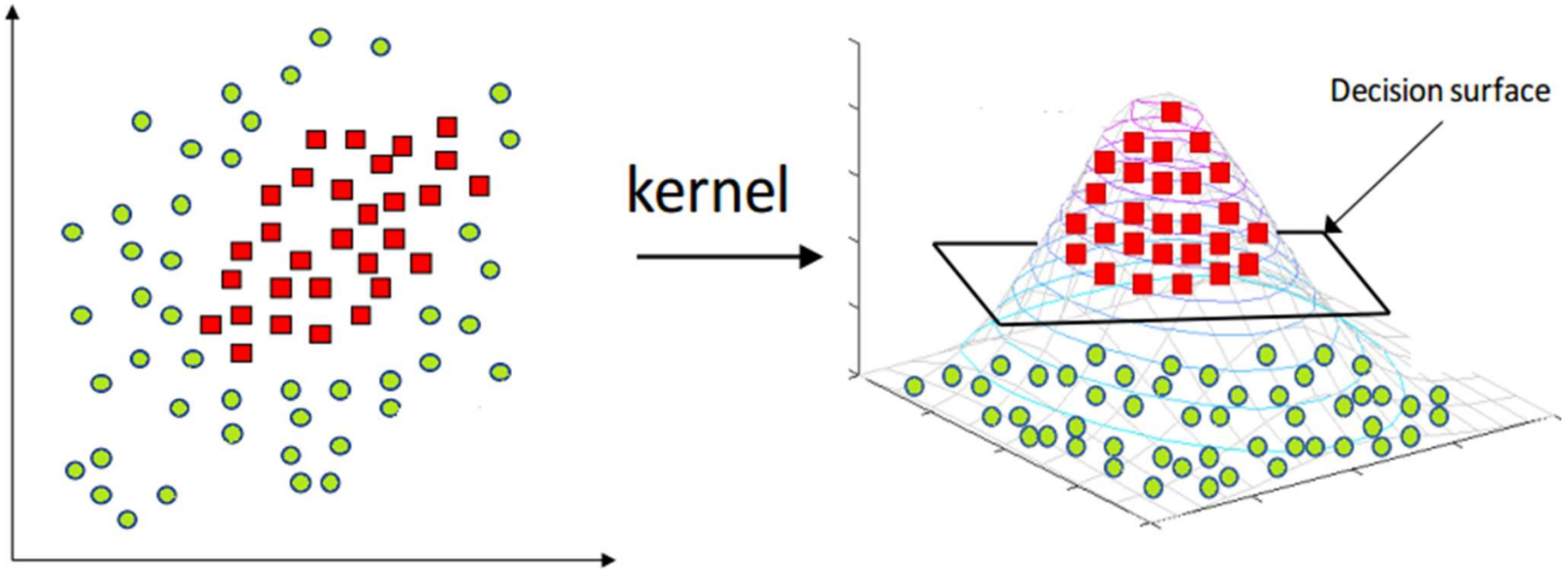


- Non-Linearly separable dataset
  ➔ kernel trick 이용하여 차원 변경

- Kernal 의 종류
  다항식
  가우시안 RBF
  Hyperbolic Tanget, etc



Kernel Function 의 역할



- 실습: Logistic Regression 과 동일한 Dataset 사용

1. 특정 사용자가 구매를 할지 여부를 예측 (구매: 1, 구매않음: 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

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

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

Bagging Tree 알고리즘

1. b = 1,….B 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 알고리즘으로 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 특징이 있다.

7. Random Forest

- Bagging Tree

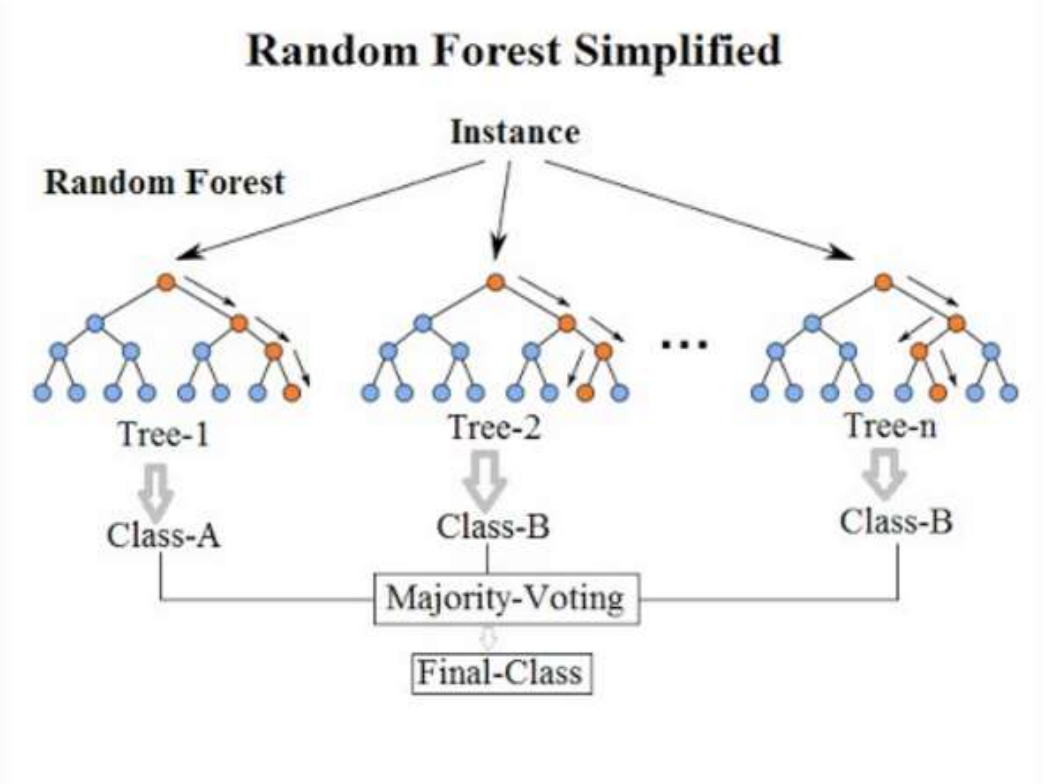
  ➔ Random Forest 로 발전

- Randomly Choose Attributes

- bootstrapping 에 의한 복원추출

  +

  attribute 의 random 선택에 따른 independent trees 구정



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

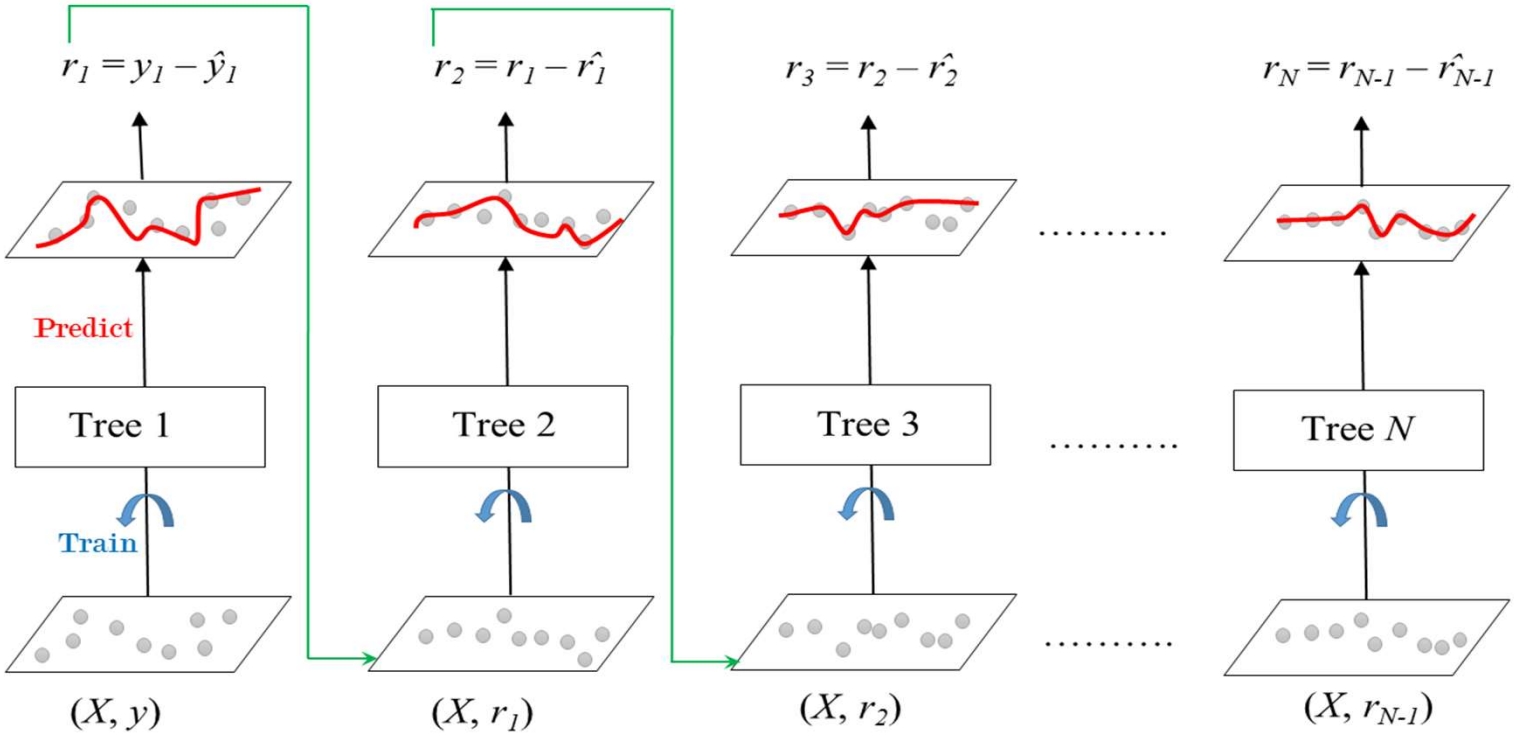
Random Forest Algorithm

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.

8. Gradient Boosting



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

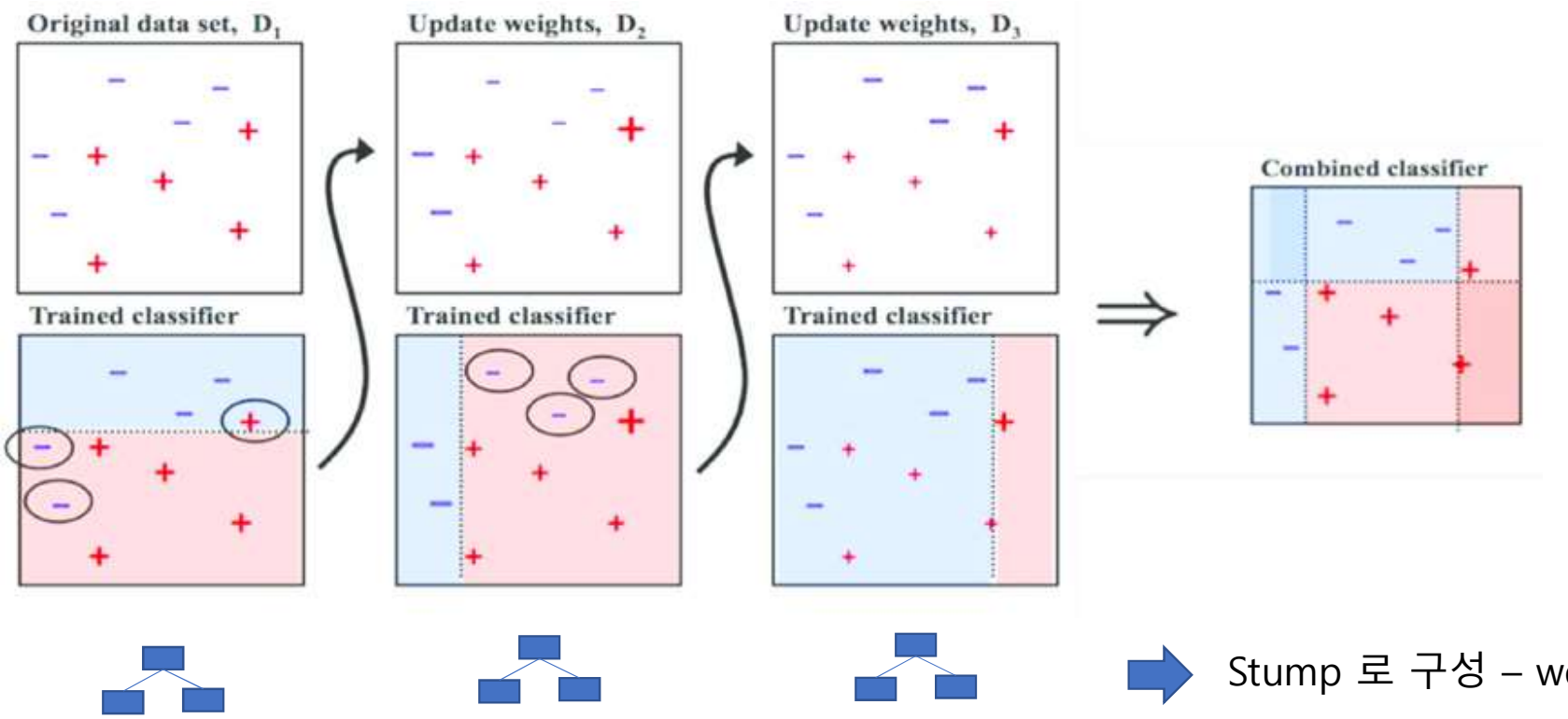
Gradient Boost 알고리즘

- 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

(참고) AdaBoost (Adaptive Boosting)



Stump 로 구성 – weak learner

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

- 실습: Social_Network_Ads data 를 이용

1. Random Forest 를 이용한 분류

2. Gradient Boosting 을 이용한 분류

3. 이전 실습과 동일한 요령

머신러닝을 위한 Data 전처리

- ✓ Missing Values 처리
- ✓ Data Formatting
- ✓ Data Normalization
- ✓ Binning
- ✓ categorical 변수의 수치화

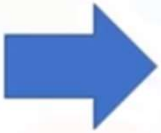
Data Formatting

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

Clustering

Clustering 이란 ?

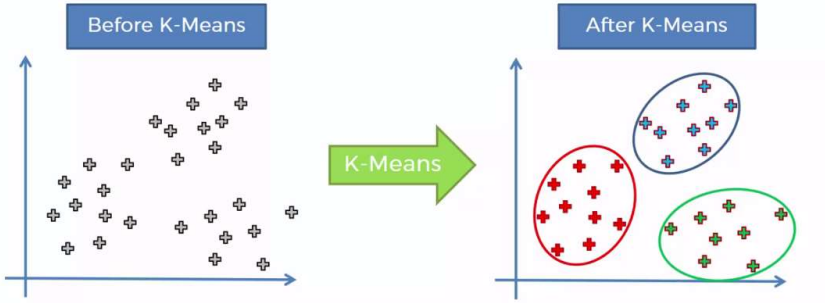
- 비슷한 object 들끼리 모으는 것

- label data 가 없는 것이 classification 과 차이 ➔ unsupervised machine learning

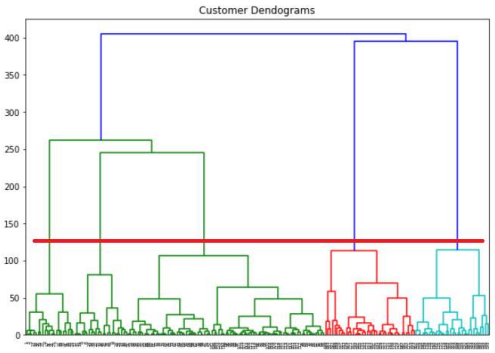
- 적용 사례
  - 고객의 구매 형태별 분류
  - 고객의 취향에 맞는 책, 동영상 등의 추천
  - 신용카드 사용의 fraud detection
  - 뉴스 자동 분류 및 추천
  - 유전자 분석 등

Clustering 알고리즘의 종류

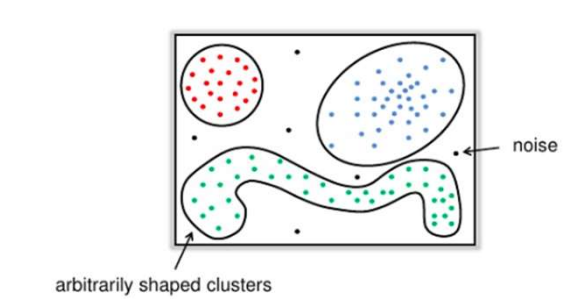
- K-Means Clustering



- Hierarchical Clustering
  (dendrogram)



- Density-based Clustering (DBSCAN)



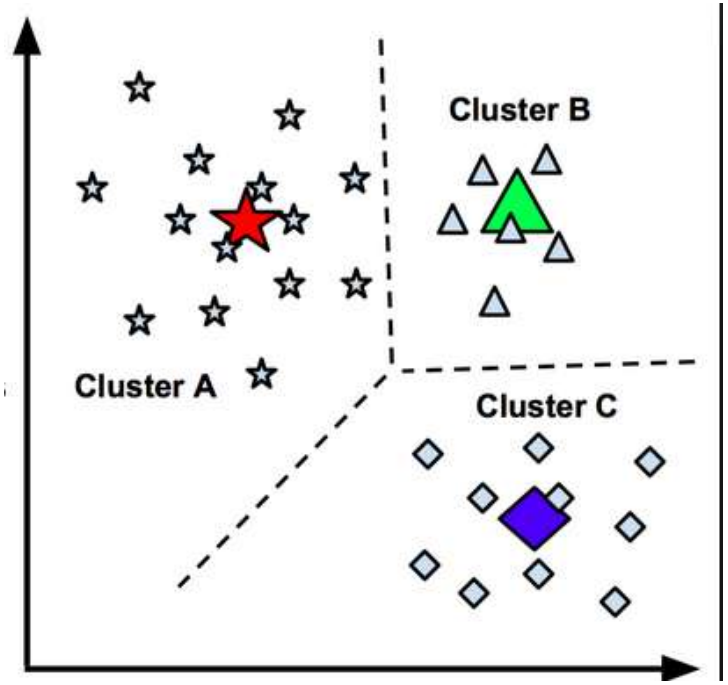
K-Means Clustering 알고리즘 – Distance 계산

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

9-1. K-Means Clustering 알고리즘

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

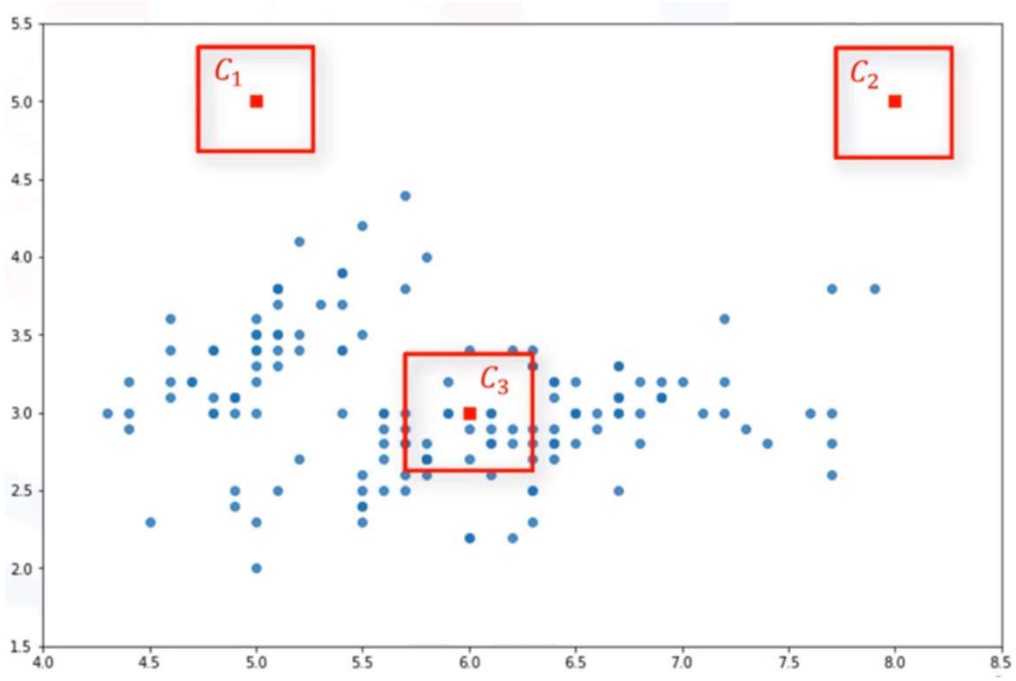
2. 각 centroid 로 부터 각 data point 까지의 거리를 계산.

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

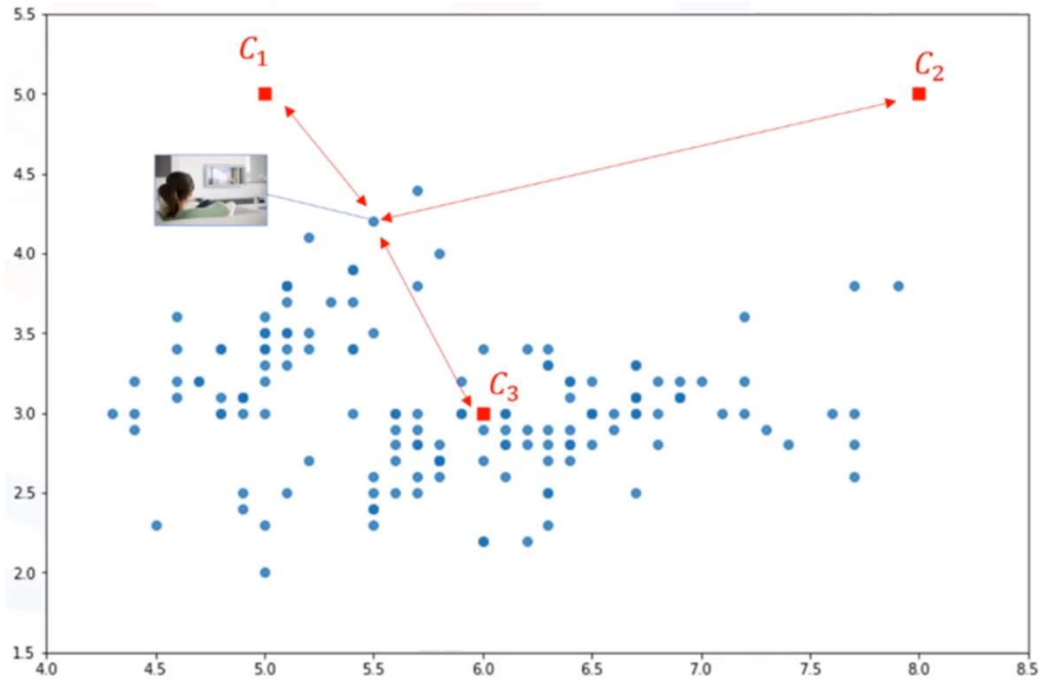
4. K centroid 의 위치를 다시 계산

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

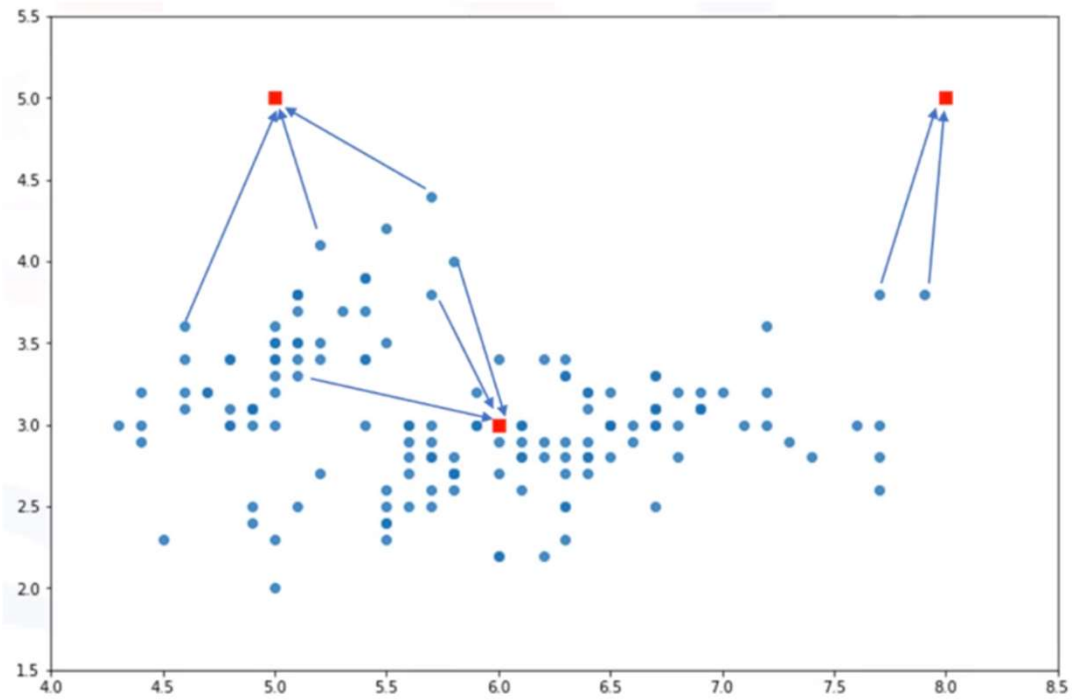
1. 임의의 centroid 선정 : k=3

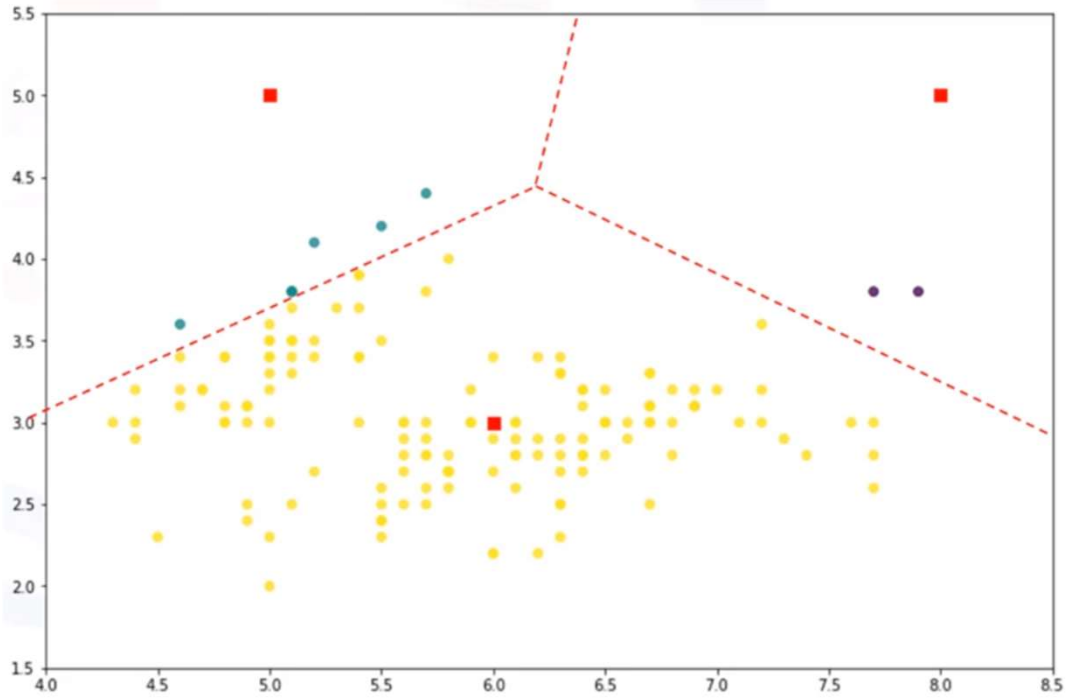


2. 거리 계산 for each data point

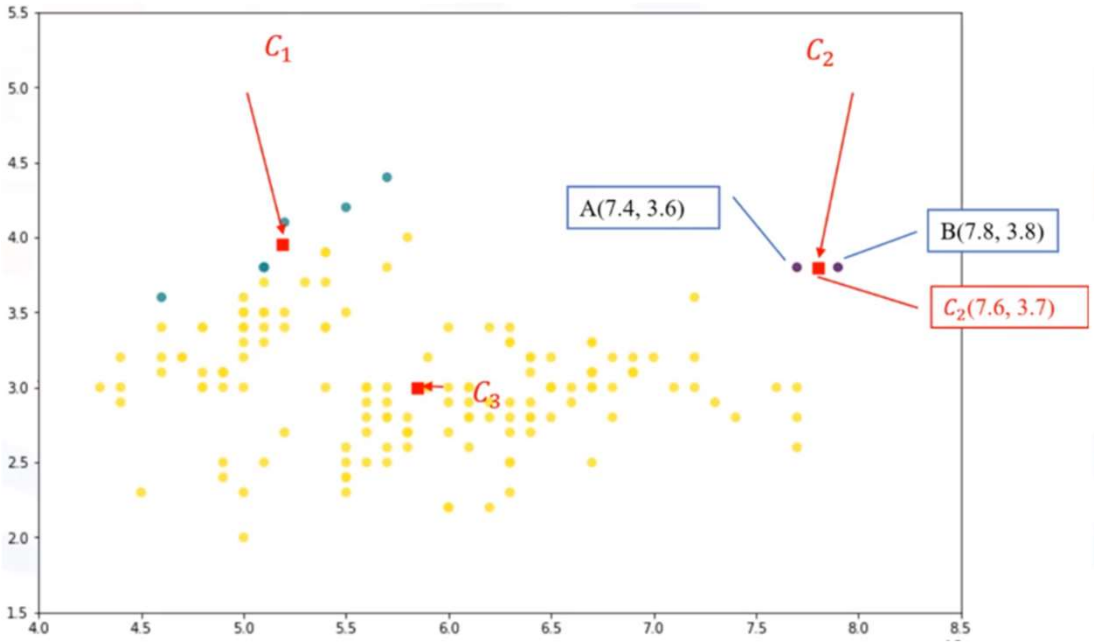


3. 가장 가까운 centroid 로 각 data point 할당

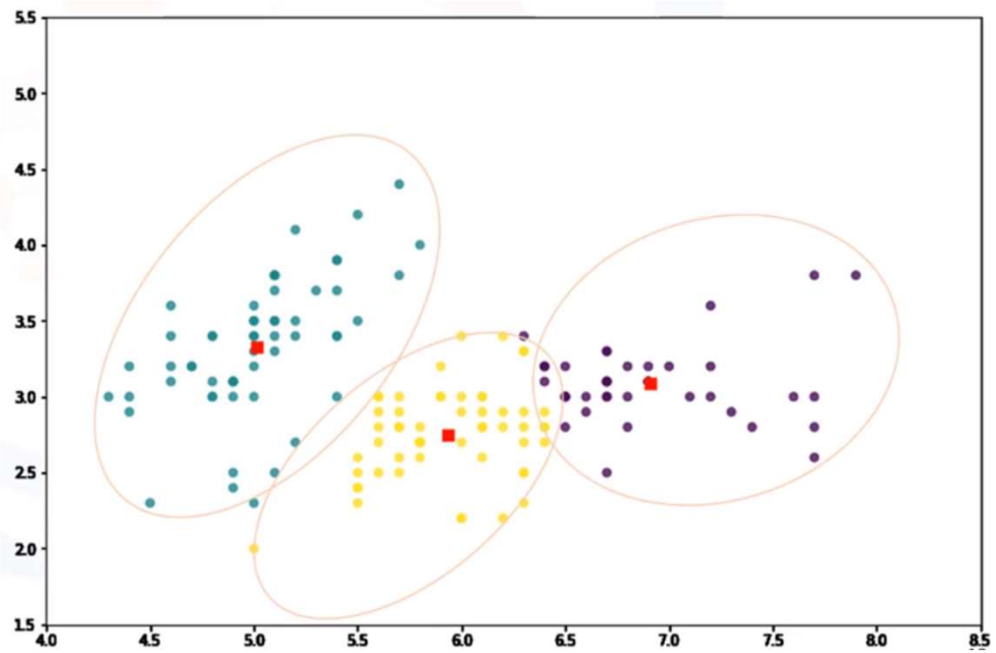




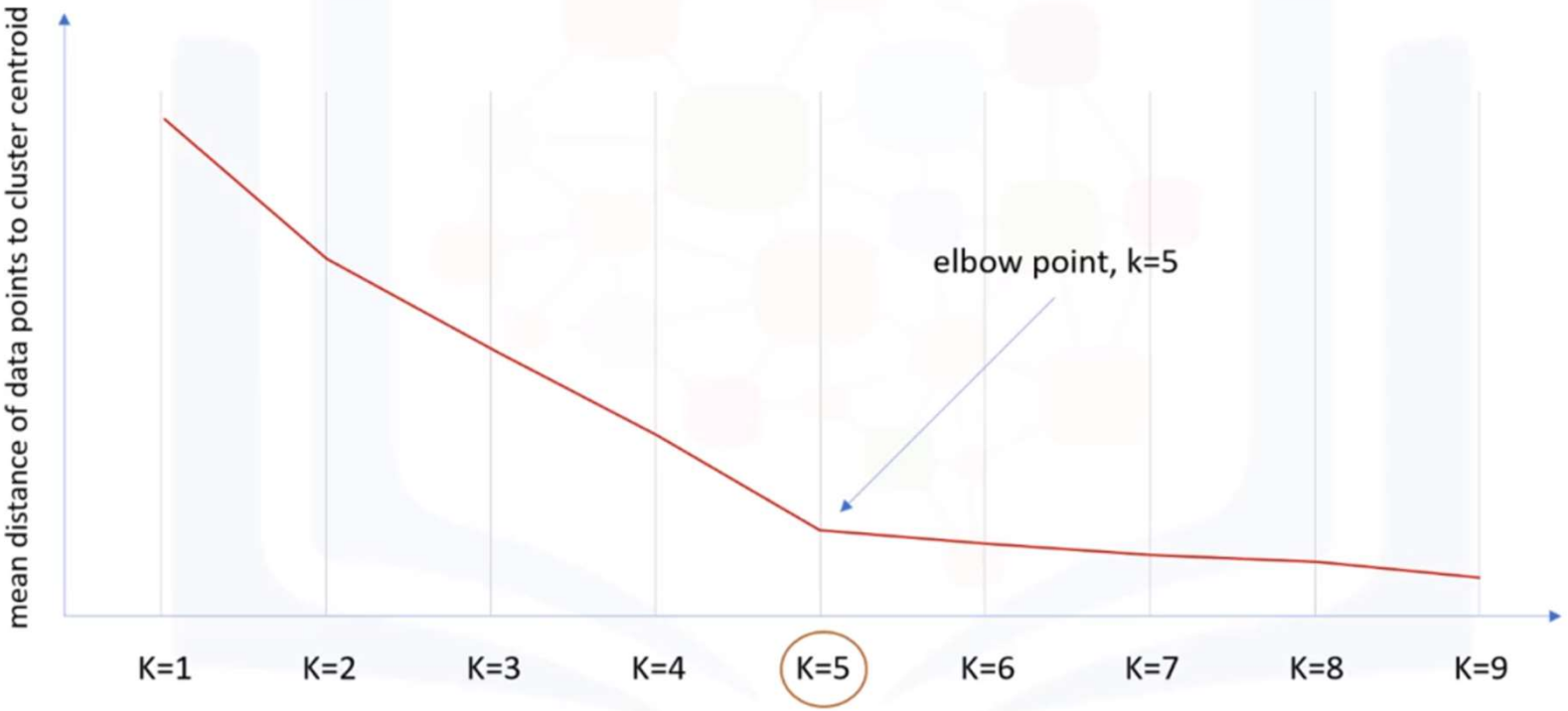
4. 각 cluster 의 new centroid 계산



5. Centroid 변화 없을 때까지 반복



Choosing k



K 를 잘 정하는 것이 중요하다.

9-2. DBSCAN 알고리즘

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

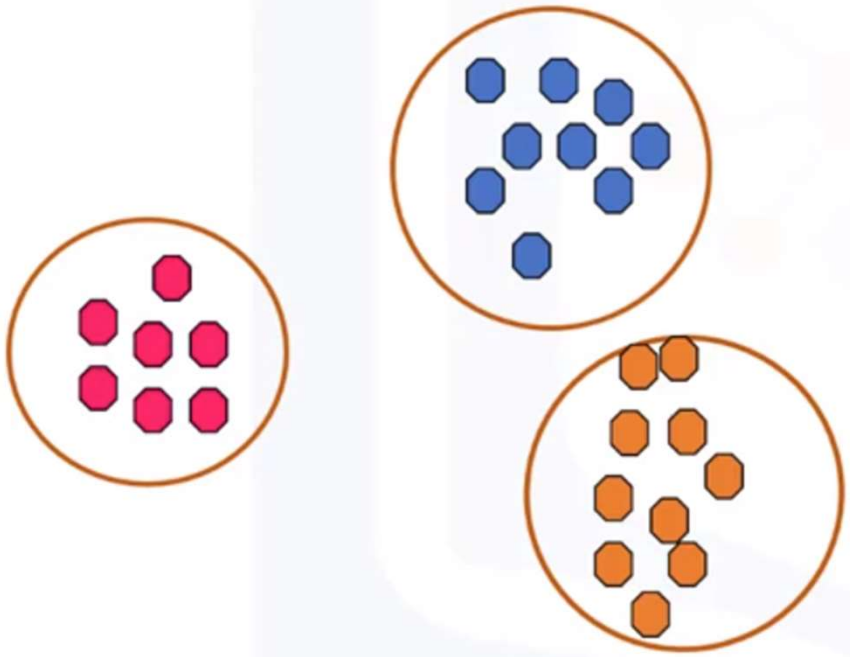
- K-Means 의 경우 임의로 cluster 지정하므로 same cluster 내의 data point 들이 실제로는 유사하지 않을 수 있다.

- DBSCAN 은 밀도가 높은 지역과 낮은 지역을 서로 분리
  (밀도 – 특정 반경내의 data point 숫자)

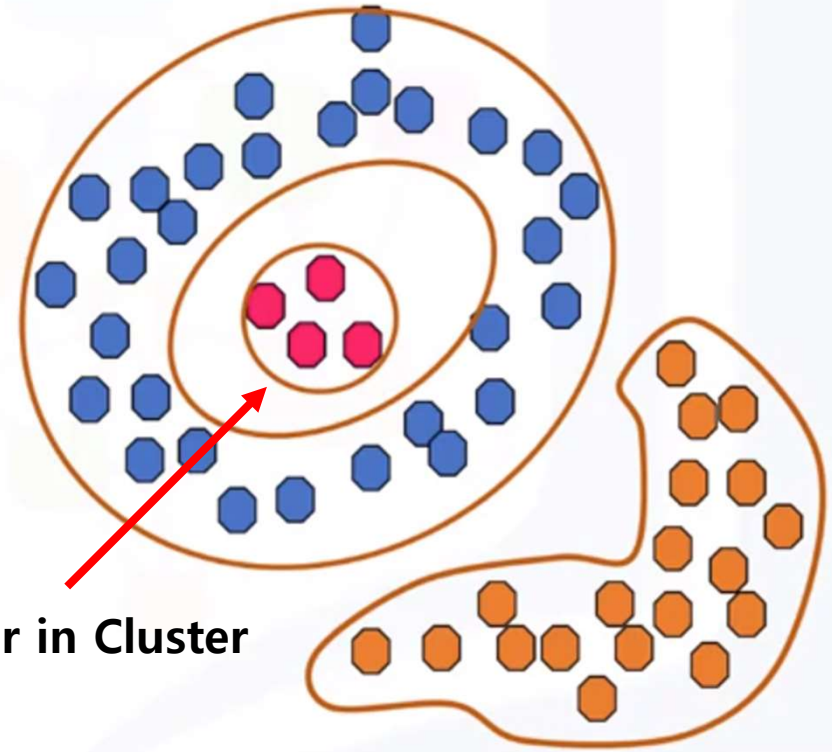
- Outlier 의 영향을 적게 받고, cluster 숫자를 미리 정해주지 않아도 되는 것이 장점

K-Means vs. DBSCAN

- Spherical-shape clusters



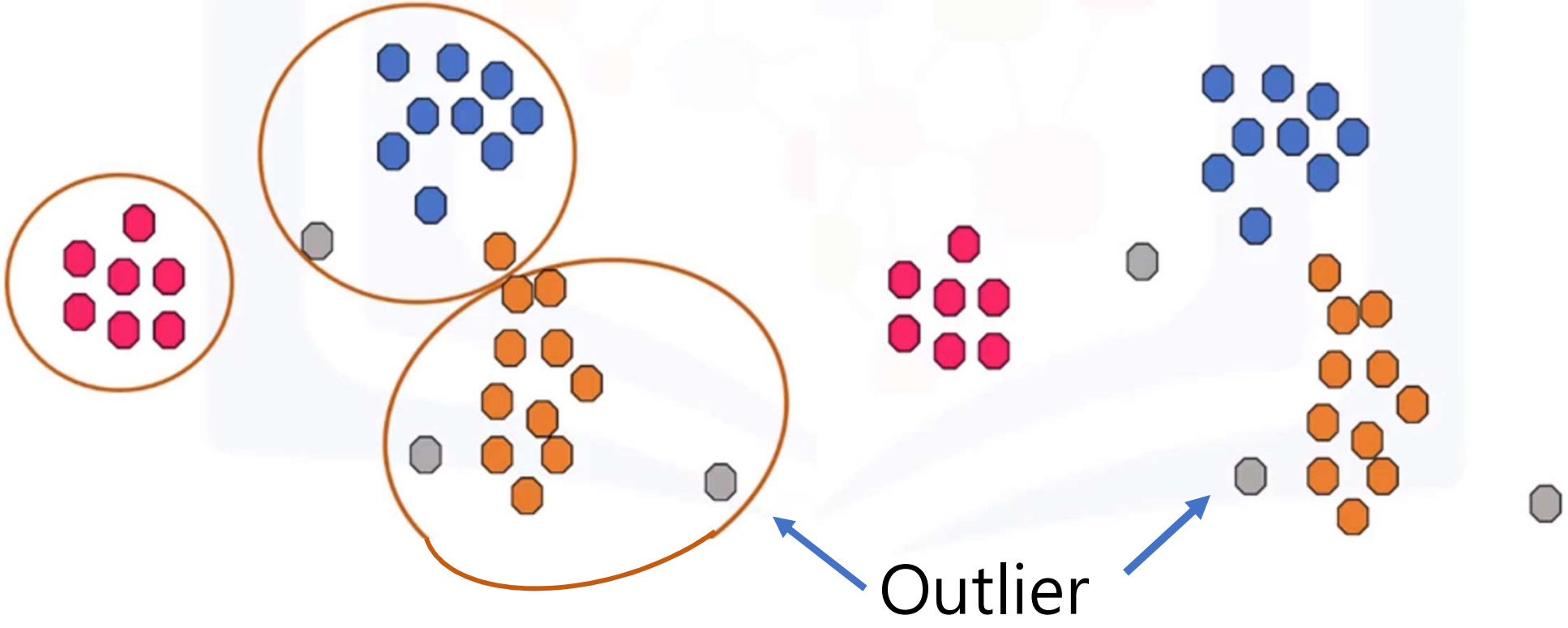
- Arbitrary-shape clusters



K-Means vs. DBSCAN

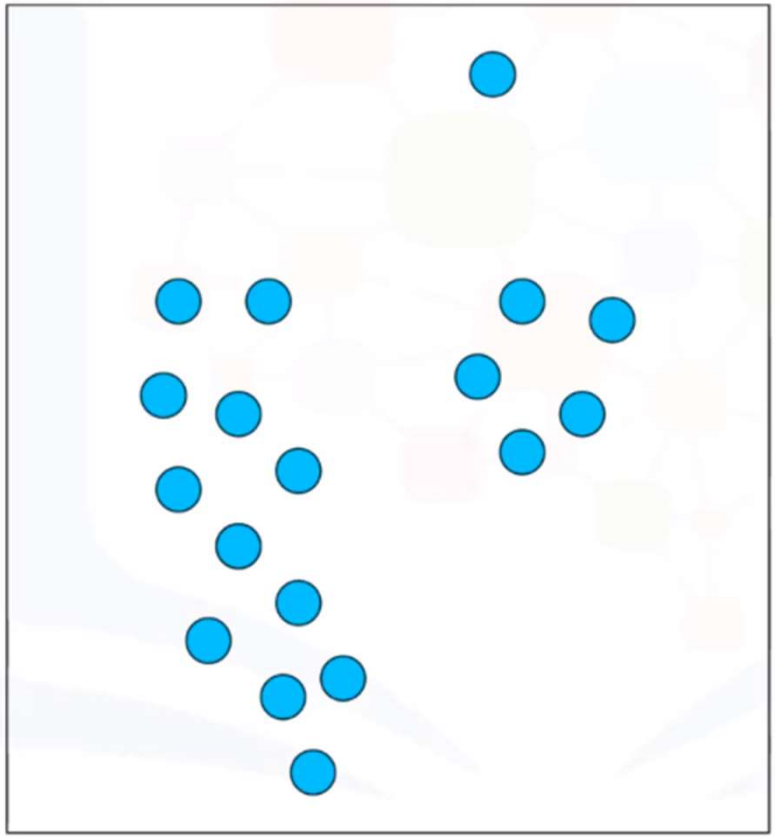
Anomaly detection (이상감지) 불가능

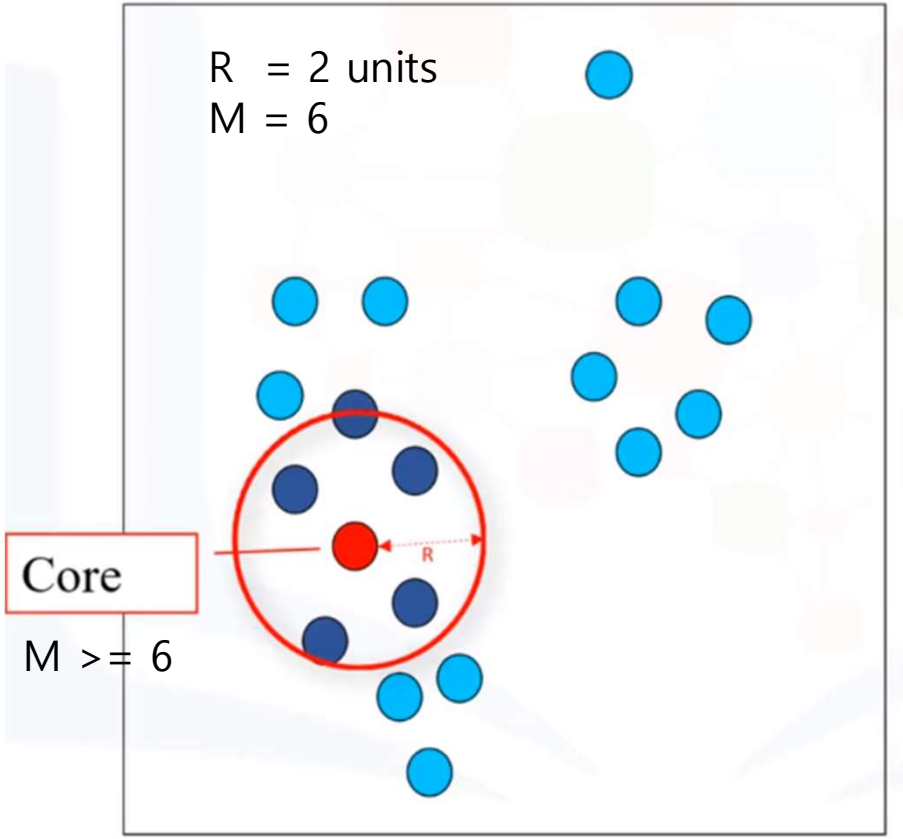
Outlier 를 쉽게 detect



DBSCAN 알고리즘

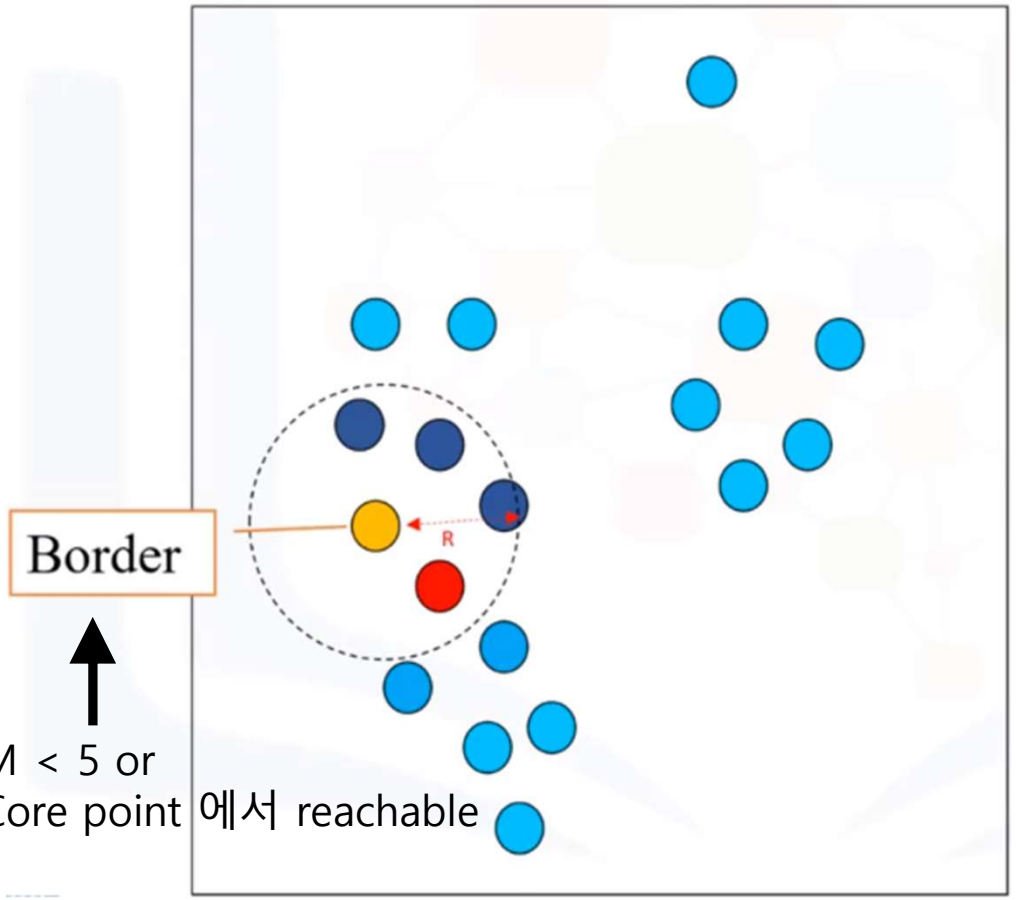
1. Radius , Minimum Neighbor number 지정

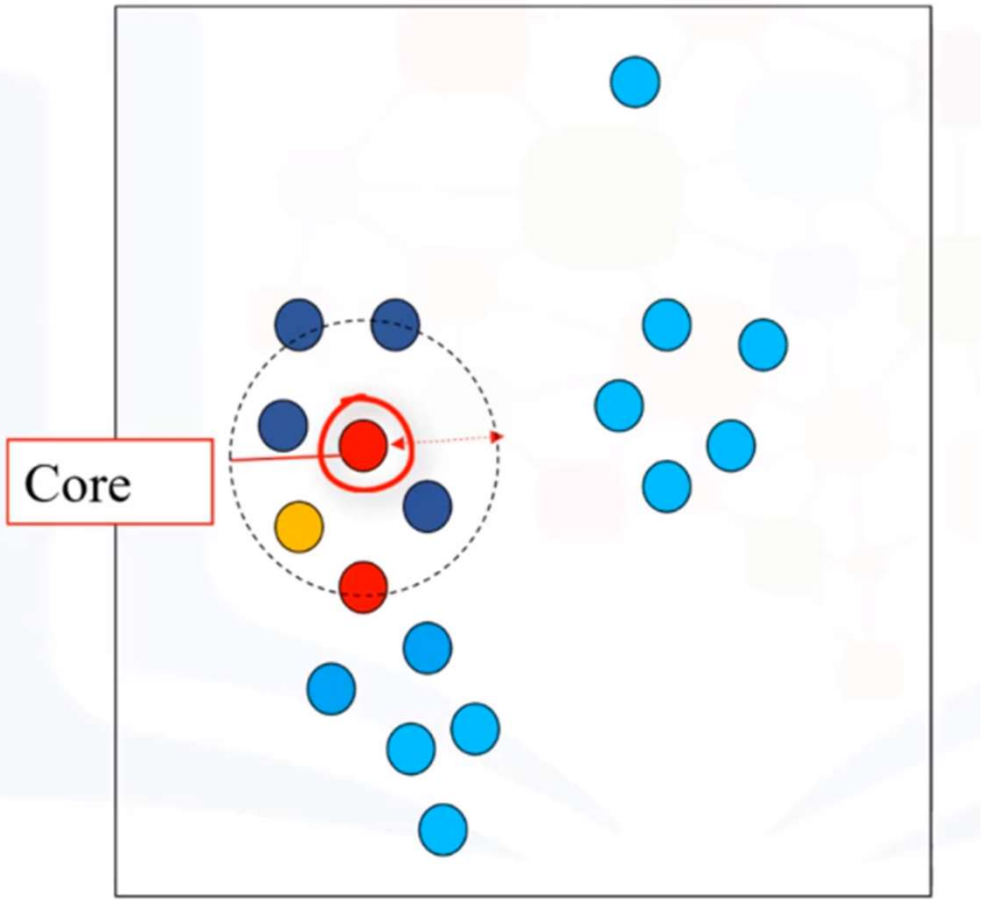




2. 각 point 를 Core, Border, Outlier 로 구분

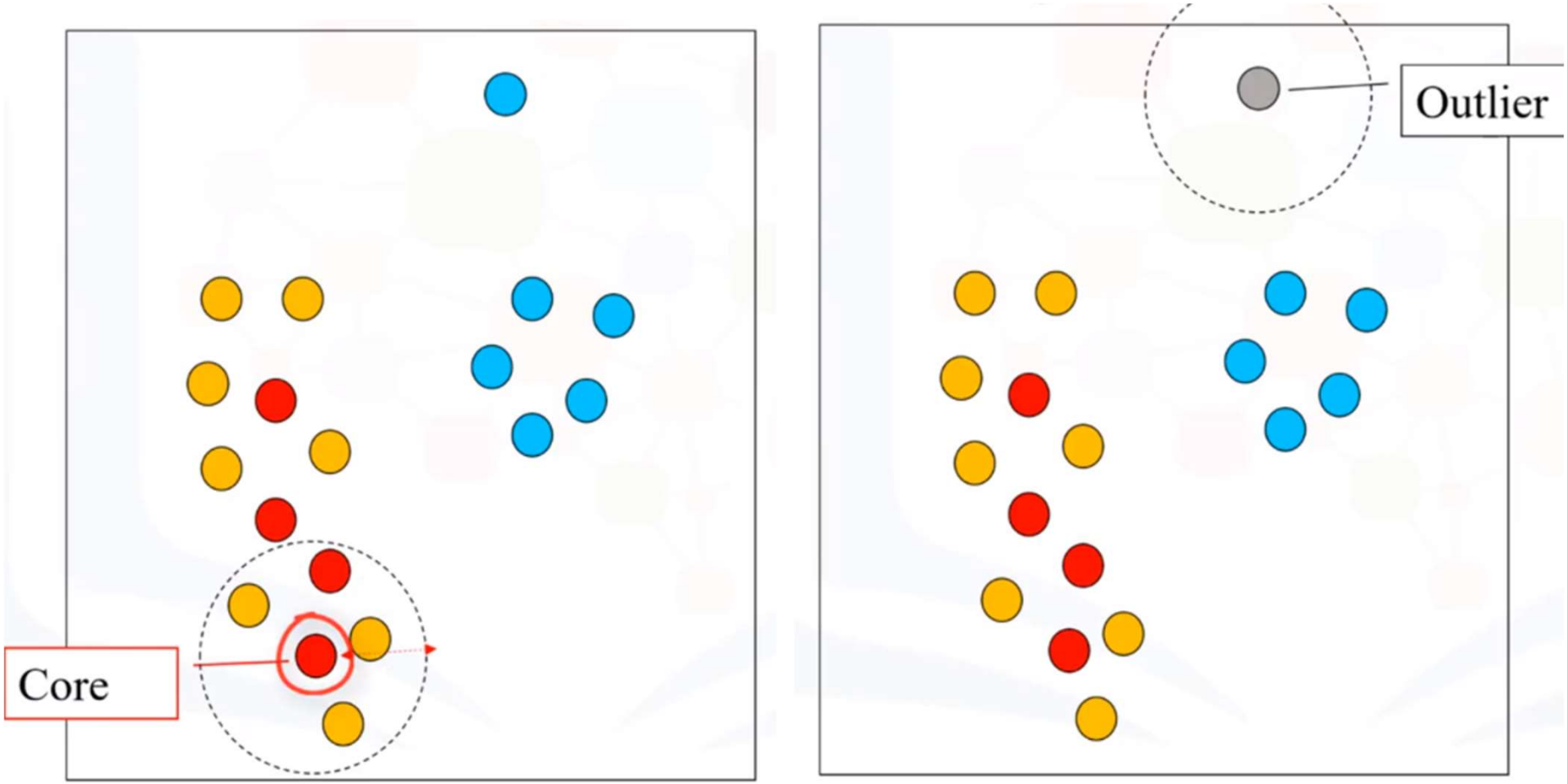
R = 2 units
M = 6





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

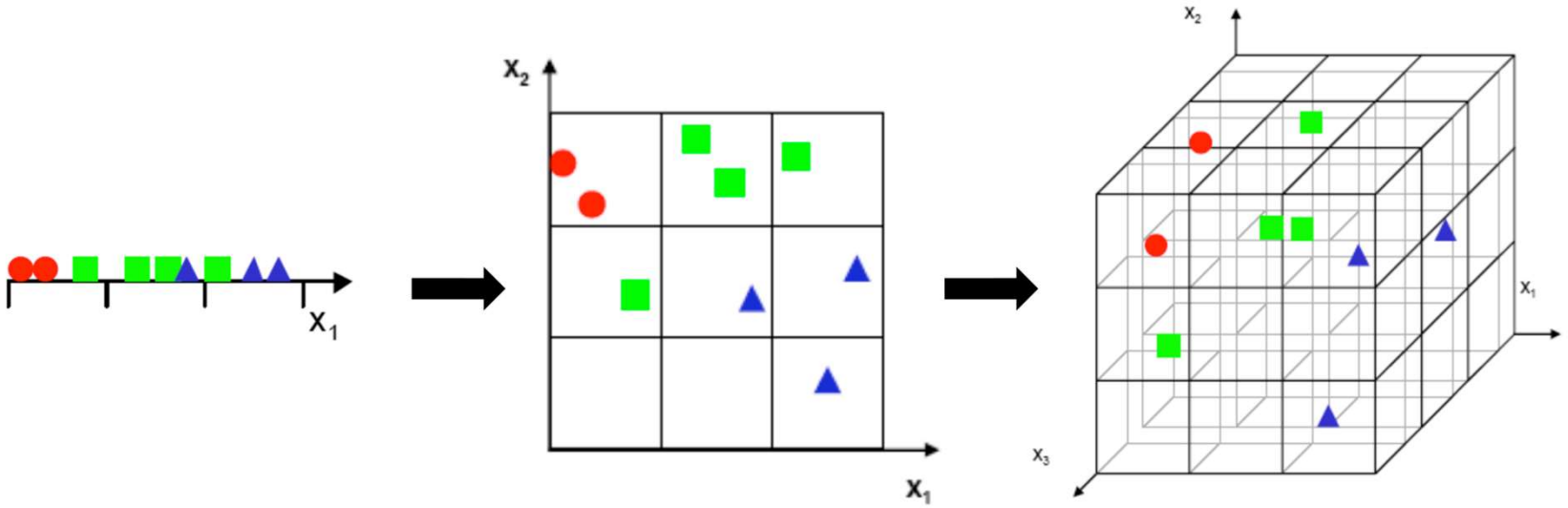
3. Matplotlib 을 이용하여 visualize

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)



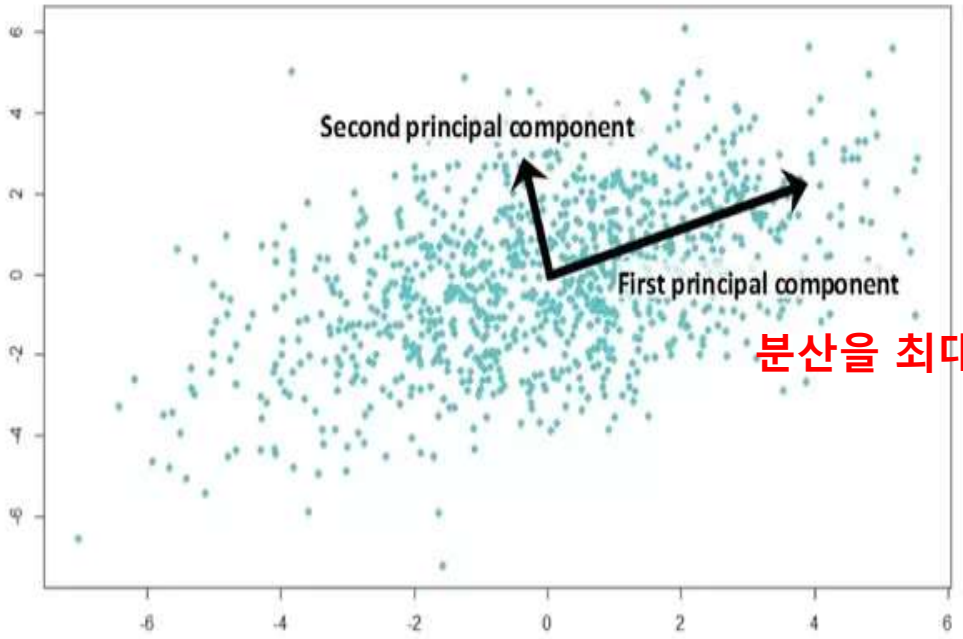
10. PCA (Principal Component Analysis) - 주성분 분석

- **선형대수학의** SVD(Singluar Value Decomposition, 특이값분해) 를 이용하여 **분산이 최대인 축**을 찾음

- 어떤 m X n 행렬 A 는 다음과 같은 형태의 세가지 행렬의 곱으로 분해할 수 있다.

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

U : nXn 직교행렬
V : mXm 직교행렬 - 주성분 column 들로 구성
Σ : nXm 직사각대각행렬



분산을 최대한 보존

sklearn.decomposition.PCA

pca.components_

- 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 visualize