데이터 분석(Python / R)

삼성전자공과대학교 3학년 3학기

Naïve Bayesian classifier

❑ Naïve Bayesian theorem

- ❖ 가정 : 주머니에 빨간공 60%, 파란공 40%, 빨간공의 20%는 깨졌고, 파란공의 30%는 깨졌다. 임의로 꺼낸 공이 깨졌을때 이 공이 빨간색일 확률은?
  - P(깨진공) : prior probability, 사전확률
  - P(파란색/깨진공) : posterior, 우도확률
  - P(깨진공/파란색) : likelihood, 사후확률
- ❖ 깨진공이 파란색인지 빨간색인지 알아내는 방법
  - Maximum A posterior : 깨진 공이 빨간색일 확률과 깨진 공이 파란색일 확률을 비교해서 더 확률 높은 쪽을 선택
  - Maximum Likehood : 파란색이 깨진공일 확률과 빨간색이 깨진공일 확률을 계산해서 더 확률 높은 쪽을 선택
- ❖ 가정 : 빨간공 60개 파란공 40개, 빨간공 12개 깨짐, 파란공 12개 깨졌다. 임의로 꺼낸 공이 깨졌을때 이 공이 빨간색일 확률은?

❑ Naïve Bayesian theorem

조건부 확률 $$P(B|A) = \frac{P(A \cap B)}{P(A)} \qquad P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$$

Bayesian theorem $$p(Y = k|X = x) = \frac{P(X = \chi|y = k)P(Y = k)}{p(X = x)}$$

- P(빨간색/깨진공) = P(깨진공/빨간색) * P(빨간색) / P(깨진공)
  = P(깨진공/빨간색) * P(빨간색) / (P(깨진공/빨간색) * P(빨간색) + P(깨진공/파란색) * P(파란색))
  = 0.2 * 0.6 / (0.2 * 0.6 + 0.3 * 0.4)
- 확장 : 주머니에 빨간공 35%, 파란공 55%, 흰공 : 10%, 빨간공의 15%는 깨졌고, 파란공의 20%는 깨졌고, 흰공의 35%는 깨짐 임의로 꺼낸 공이 깨졌을때 이 공이 빨간색일 확률은?
- P(빨간색/깨진공) = P(깨진공/빨간색) * P(빨간색) / P(깨진공)
  = P(깨진공/빨간) * P(빨간색) / (P(깨진공/빨간색)* P(빨간색) + P(깨진공/파란색)* P(파란색) + P(깨진공/흰색) * P(흰색))
  = 0.15 * 0.35 / (0.15*0.35 + 0.2*0.55 + 0.35*0.1)

❑ Naïve Bayesian classifier

no	words	class
1	fun, couple, love, love	comedy
2	fast, furious, shoot	action
3	couple, fly, fast, fun, fun	comedy
4	furious, shoot, shoot, fun	action
5	fly, fast, shoot, love	action
6	fast, furious, fun	?

no	class	fun	couple	love	fast	furious	shoot	fly
1	comedy	1	1	2	0	0	0	0
2	action	0	0	0	1	1	1	0
3	comedy	2	1	0	1	0	0	1
4	action	1	0	0	0	1	2	0
5	action		0	1	1	0	1	1
6	?	1	0	0	1	1	0	0

$$P(C_1 \mid x_1, x_2, \ldots, x_n) = \frac{P(x_1, x_2, \ldots, x_n \mid C_1)P(C_1)}{P(x_1, x_2, \ldots, x_n)}$$

$$P(C_2 \mid x_1, x_2, \ldots, x_n) = \frac{P(x_1, x_2, \ldots, x_n \mid C_2)P(C_2)}{P(x_1, x_2, \ldots, x_n)}$$

$$P(x_1, x_2, \ldots, x_n \mid C_1) = P(x_1 \mid C_1) * P(x_2 \mid C_1) * \ldots P(x_n \mid C_1)$$

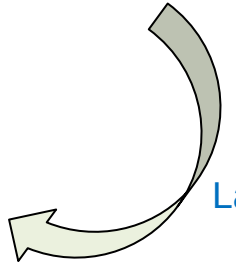
✓ Feature 들은 서로 독립이라고 가정

❑ Naïve Bayesian classifier

no	words	class
1	fun, couple, love, love	comedy
2	fast, furious, shoot	action
3	couple, fly, fast, fun, fun	comedy
4	furious, shoot, shoot, fun	action
5	fly, fast, shoot, love	action
6	fast, furious, fun	?

P(comedy/fast,furious,fun) = P(fast/comedy) * P(furious/comedy)
* P(fun/comedy) * P(comedy)
= 1/9 * 0/9 * 3/9 * 2/5 = 0

P(action/fast,furious,fun) = P(fast/action) * P(furious/action)
* P(fun/action) * P(action)
= 2/11 * 2/11 * 1/11 * 3/5 = 0.0018



Laplace smoothing

P(comedy/fast,furious,fun) = P(fast/comedy) * P(furious/comedy)
* P(fun/comedy) * P(comedy)
= (1+1)/(9+7) * (0+1)/(9+7) * (3+1)/(9+7) * 2/5 = 0.00078

P(action/fast,furious,fun) = P(fast/action) * P(furious/action)
* P(fun/action) * P(action)
= (2+1)/(11+7) * (2+1)/(11+7) * (1+1)/(11+7) * 3/5 = 0.0018

❑ Naïve Bayesian classifier

❖ 장점

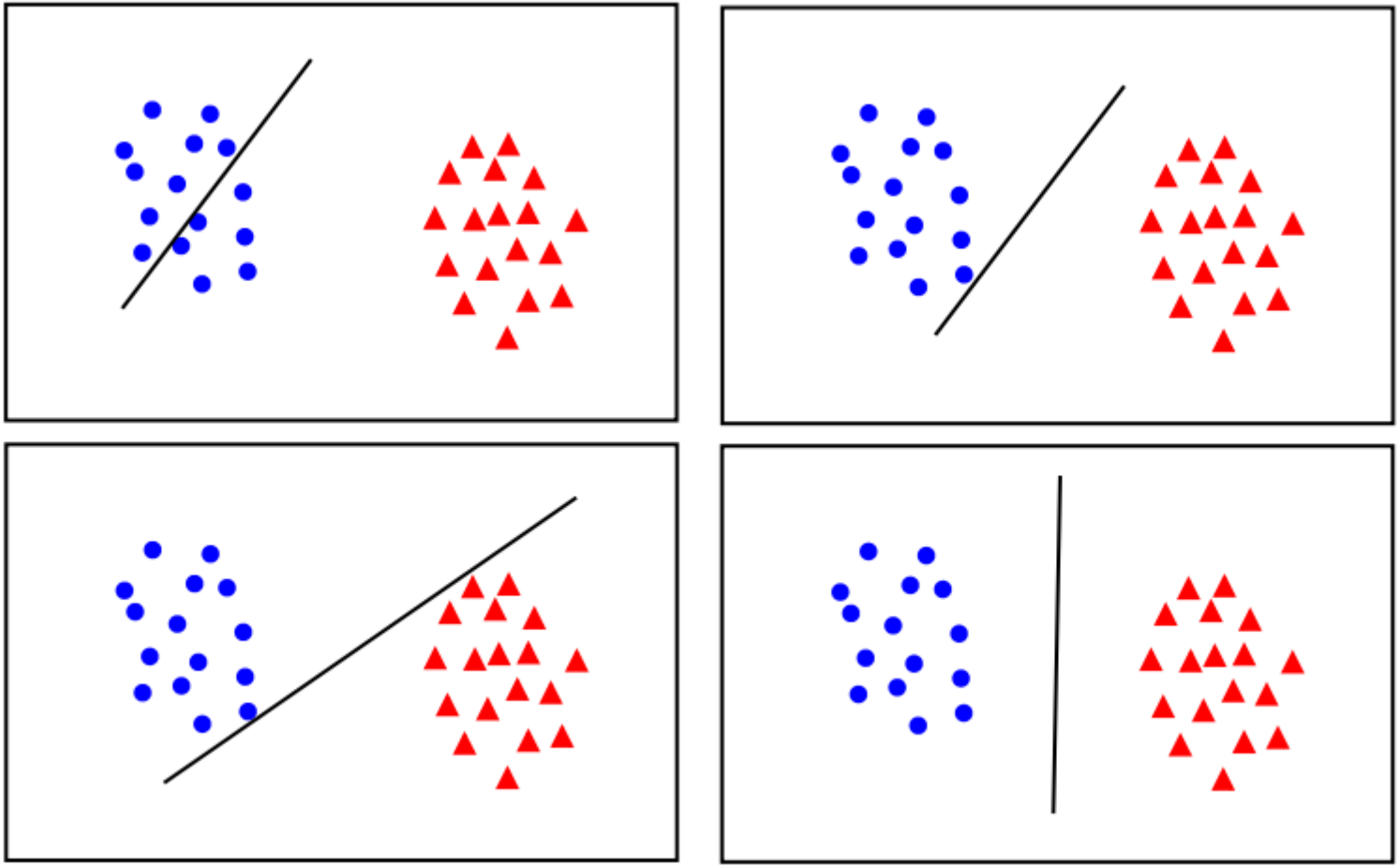
- noise에 민감하지 않음
- 무관한 입력 변수들에 사용하기 적합
- Missing value를 무시할 수 있음
- DTM(document term matrix)와 같은 sparse data에 대해서도 좋은 성능을 보임

❖ 단점

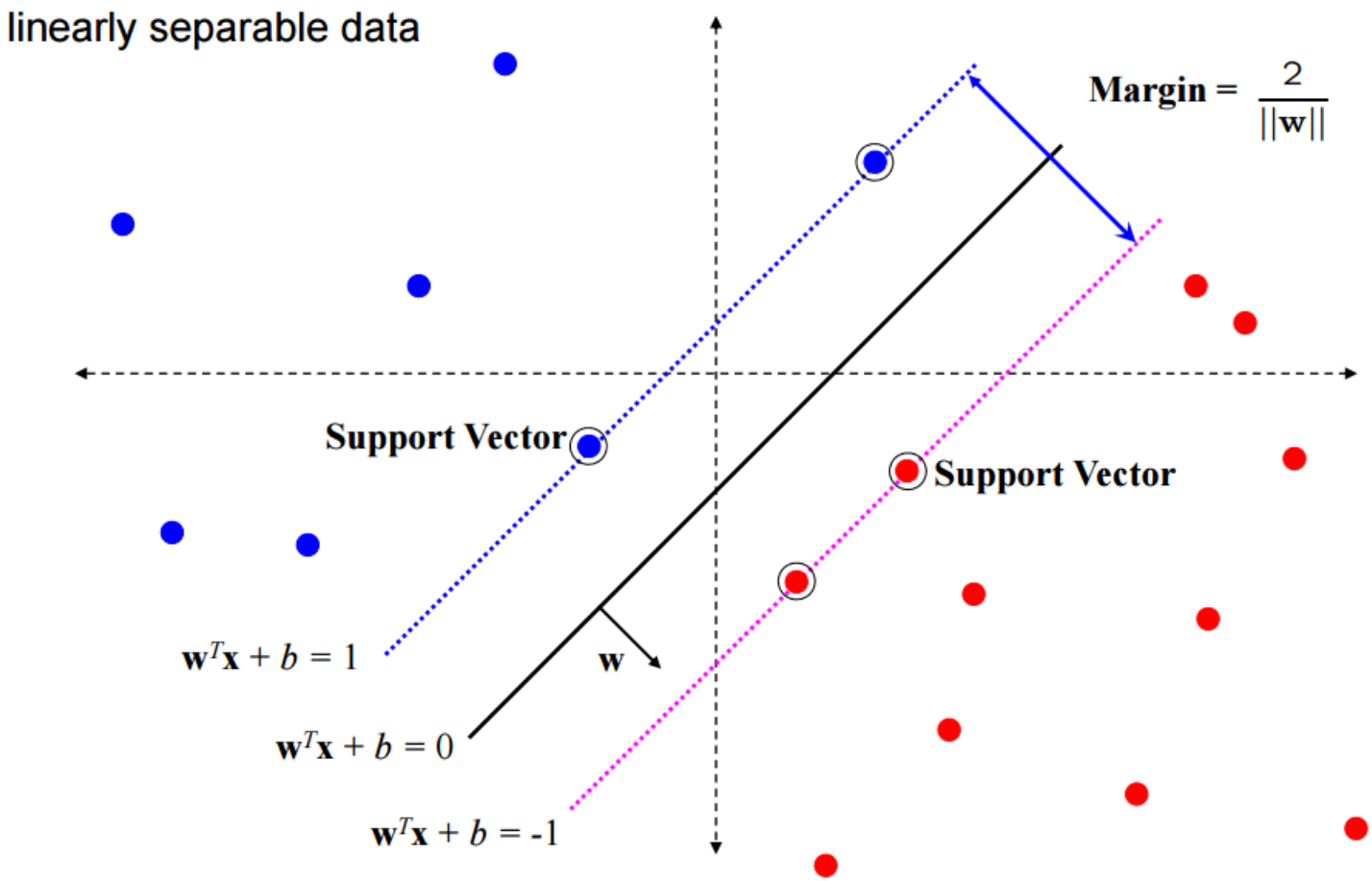
- 변수들 간의 상관관계가 있는 데이터
- 큰 biased result
- 충분히 많은 자료 필요

Support Vector Machine

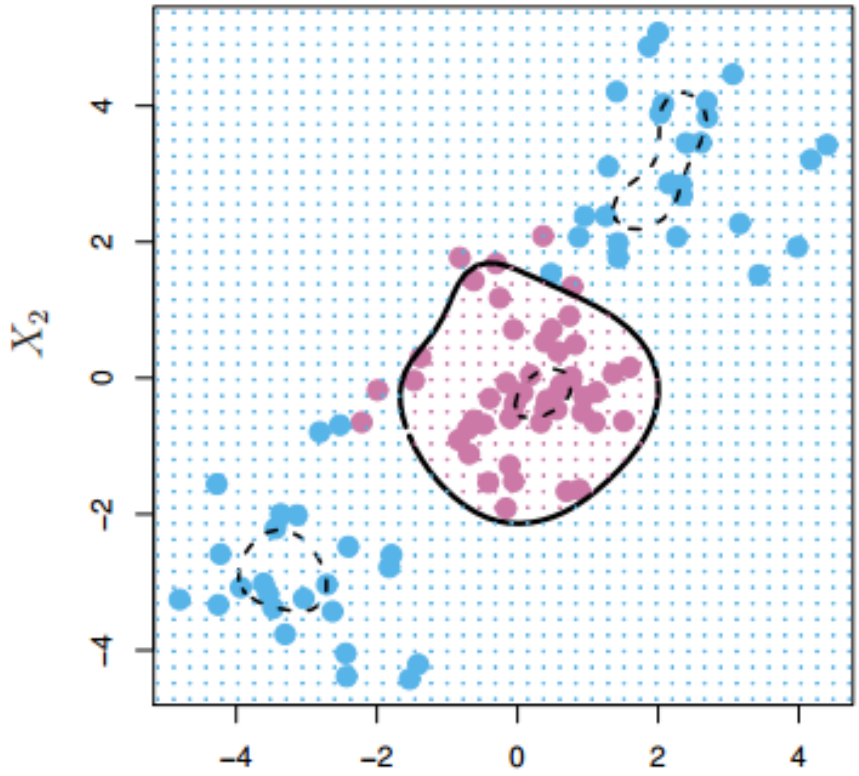
❑ SVM



❑ SVM


linearly separable data
Margin = 2 / ||w||
Support Vector
Support Vector
w
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

❑ SVM



K-nearest neighbor(KNN)

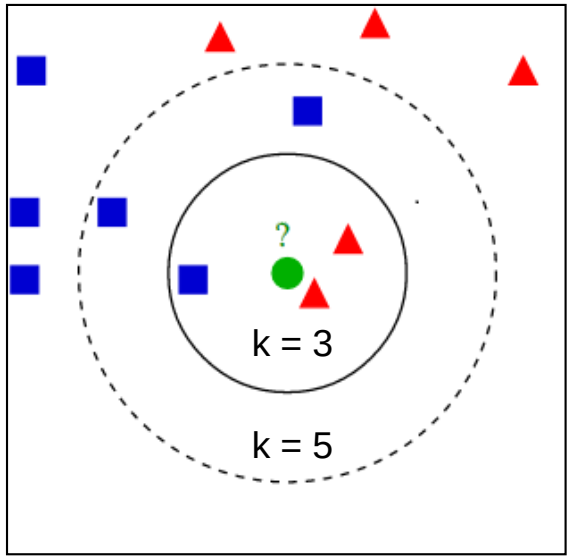
❑ KNN

❖ training data로 부터 식별 함수를 학습하는 대신 데이터를 기억하는 방식, 학습 과정동안 비용이 들지 않는다(lazy learning)

1. k 에 해당하는 숫자와 distance metric 선택
2. 분류하고자 하는 데이터에 대한 k개의 nearest neighbor를 찾는다
3. 다수결 투표 방식으로 분류 레이블 할당

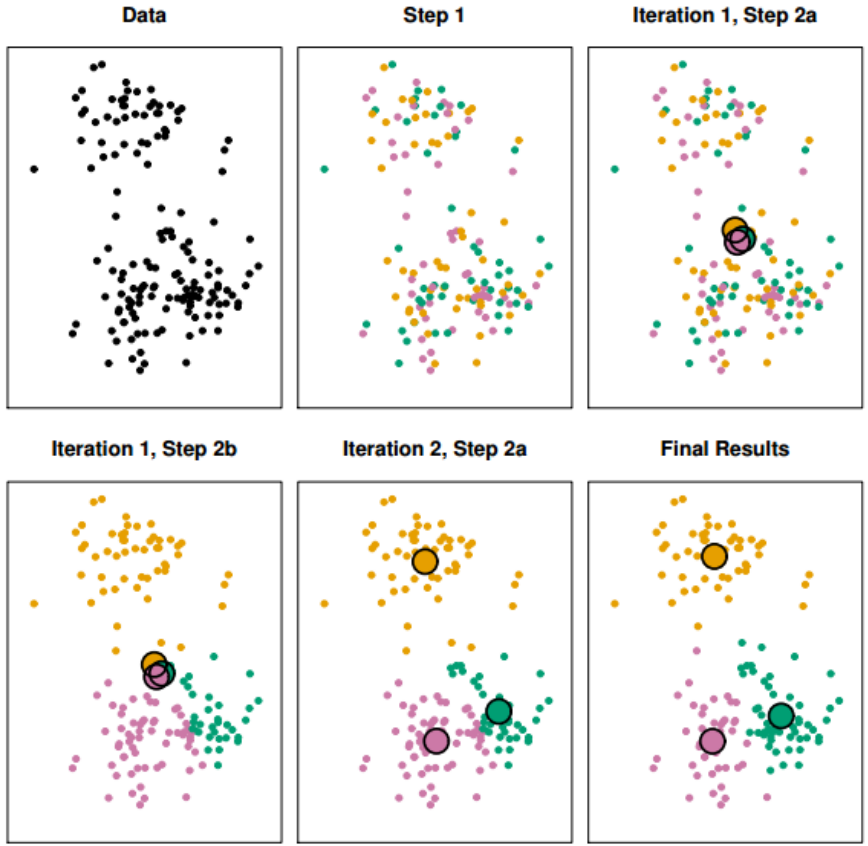
❖ 메모리 기반 방식이므로 큰 데이터로 작업하는 경우 스토리지 공간이 중요

❖ k, distance metric 선택이 중요



Clustering

❑ K-means



1. 임의로 k 개의 centroid 선택
2. 각각의 데이터를 가장 가까운 centroid에 할당
3. 2단계에서 할당된 데이터의 중심으로 centroid 변경
4. Cluster 할당이 변하지 않을 때 까지 또는 사용자 정의의 허용오차나 최대 반복 회수에 다다를 때까지 2,3 단계 반복

❑ K-means clustering

- ❖ Cluster : C1 ~ Ck
  - ▪ 모든 데이터는 적어도 하나의 클러스터에 속한다
  - ▪ 하나 이상의 클러스터에 속하는 데이터는 없다
- ❖ WCV(within cluster variation) 이 최소화 되도록 클러스터 조정

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \text{WCV}(C_k) \right\}.$$

- ❖ K개의 클러스터의 WCV 의 총합이 최소가 되도록 데이터를 K 개의 클러스터로 나눔

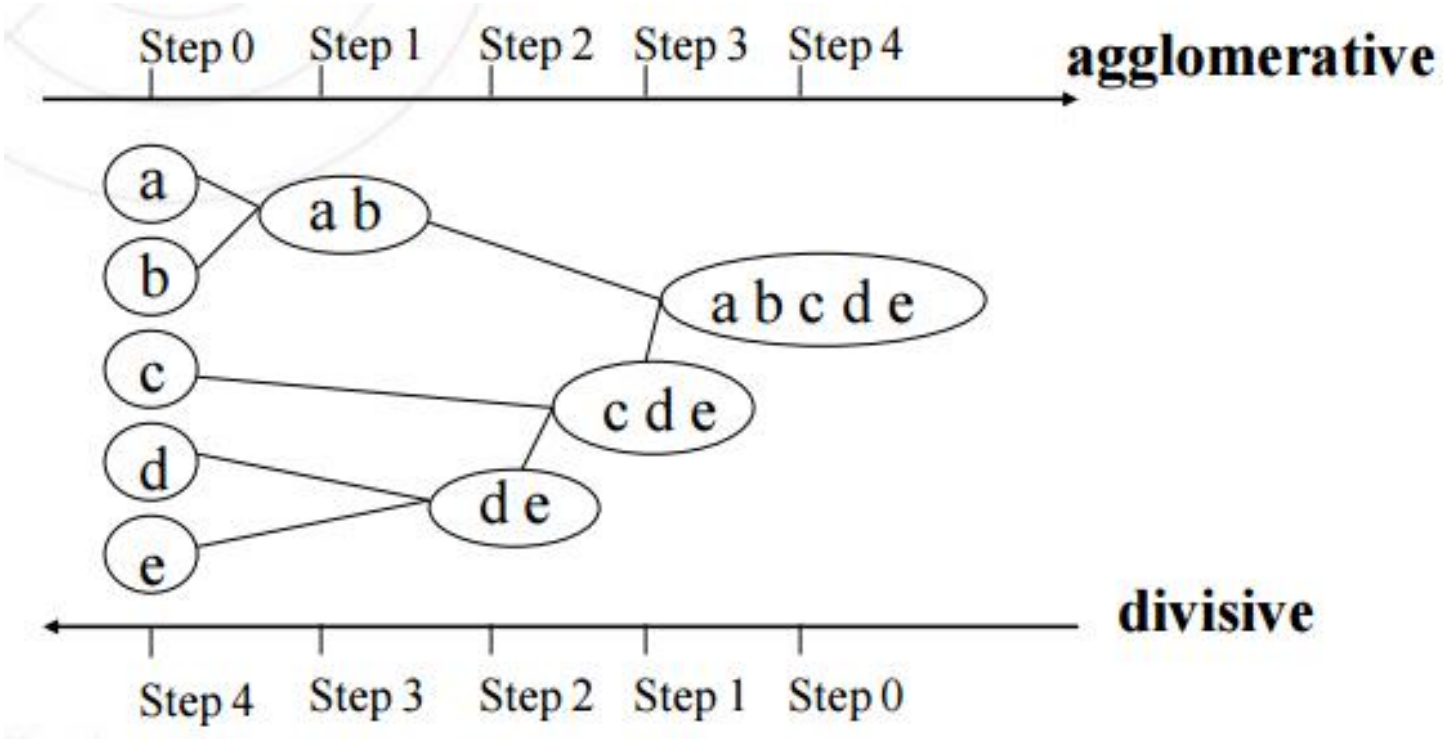
$$\text{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, \qquad \underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

❑ K-means++

❖ 초기 centroid를 서로 가능한 멀리 지정하여 성능 향상

1. 공집합 M을 초기화
2. 데이터들 로부터 첫번째 centroid μ^j 를 임의로 선택하여 M에 할당
3. M내에 있지 않은 각각의 데이터 x^i 에 대해 M내의 centroid 중 모든 centroid에 대한 최소 제곱거리 $d(x^i, M)^2$ 를 찾는다
4. 다음 centroid μ^p 를 임의로 선택할 때, $\frac{d(\mu^p, M)^2}{\sum d(x^i, M)^2}$ 와 같은 가중확률분포 사용
5. K 개의 centroid 가 선택될 때 까지 2,3 단계 반복
6. K-mean 수행

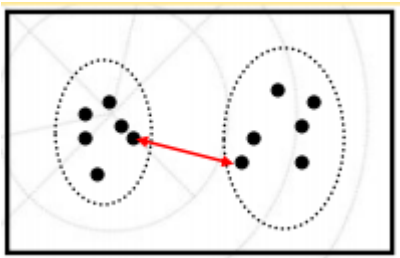
❑ Hierarchical clustering



❑ Hierarchical clustering

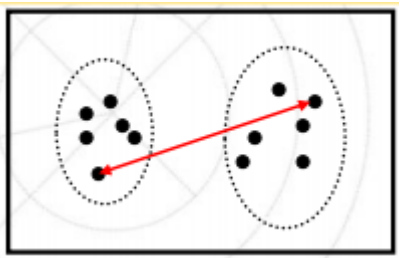
❖ 클러스터간의 유사도 측정 방법

- Single Link



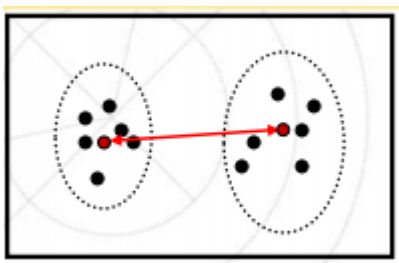
가장 가까운 점의 거리
(neighbouring joining)

- Complete Link



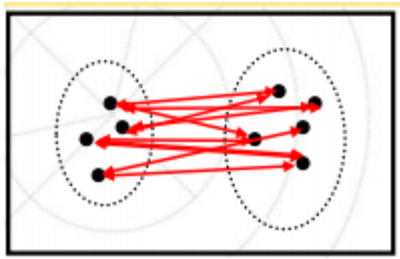
가장 먼 점의 거리

- Median



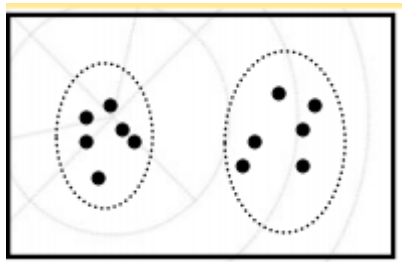
중앙값 거리
(centroid)

- Average Link



모든 점들의 평균 거리

- Ward Link



점에서 중심까지의 편차에
대한 제곱을 합한 것

❑ Hierarchical clustering

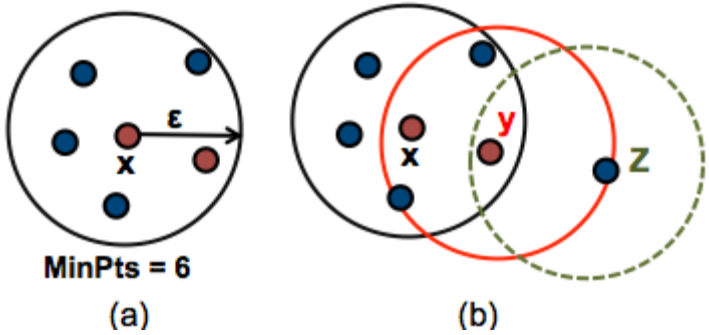
❖ Similarity measures between clusters : Lance-Williams formula

$$d(i+j,k) = a_i d(i,k) + a_j d(j,k) + b d(i,j) + c\left|d(i,k) - d(j,k)\right|$$

Single-link	$a_i = a_j = 0.5\ ;\ b = 0\ ;\ c = -0.5 \qquad d(i+j,k) = \min\{d(i,k), d(j,k)\}$
Complete-link	$a_i = a_j = 0.5\ ;\ b = 0\ ;\ c = 0.5 \qquad d(i+j,k) = \max\{d(i,k), d(j,k)\}$
Centroid	$a_i = \frac{n_i}{n_i + n_j} \quad a_j = \frac{n_j}{n_i + n_j} \quad b = -\frac{n_i n_j}{(n_i + n_j)^2} \quad c = 0 \quad d(i+j,k) = d(\mu_{i+j}, \mu_k)$
Median	$a_i = a_j = 0.5\ ;\ b = -0.25\ ;\ c = 0$
(Average link)	$a_i = \frac{n_i}{n_i + n_j} \quad a_j = \frac{n_j}{n_i + n_j} \quad b = c = 0 \quad d(C_i, C_j) = \frac{1}{n_i n_j} \sum_{a \in C_i, b \in C_j} d(a,b)$
Ward's Method (minimum variance)	$a_i = \frac{n_k + n_i}{n_k + n_i + n_j} \quad a_j = \frac{n_k + n_j}{n_k + n_i + n_j} \quad b = -\frac{n_k}{n_k + n_i + n_j} \quad c = 0$

❑ DBSCAN(Density-based spatial clustering of application with Noise)

- ❖ 노이즈가 있는 어플리케이션에 대한 밀도 기반의 희소 클러스터링
  - ▪ 밀도(density) : 특정 반지름 내 점들의 개수
- ❖ Neighbor : 한 점(point)에서 특정 반지름 ε 내에 위치한 다른 점들
- ❖ Core point(x) : 하나의 점(point)로 부터 거리 ε 내에 위치한 다른 점들의 수가 MinPts 보다 클 경우 하나의 cluster를 생성하며 중심이 된 점을 core point로 분류
- ❖ Border point : core point로 부터 거리 ε 내에 위치해서 같은 cluster로 분류되나, 그 자체로는 core point 가 아닌, cluster의 외곽을 형성하는 점
- ❖ Noise point : 거리 ε 이내의 MinPts 개 미만의 점들이 있으며 그 점들 모두 core point가 아닐 경우, 어떠한 cluster에도 속하지 않고 Noise point 로 분류

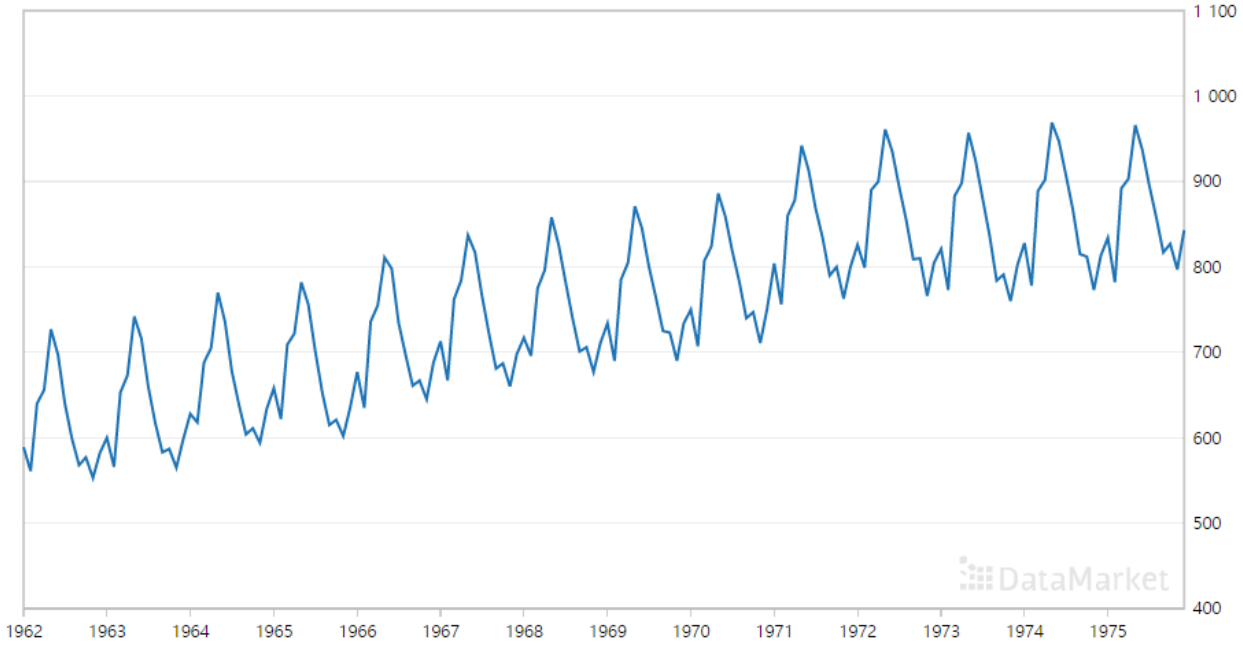


Time series analysis

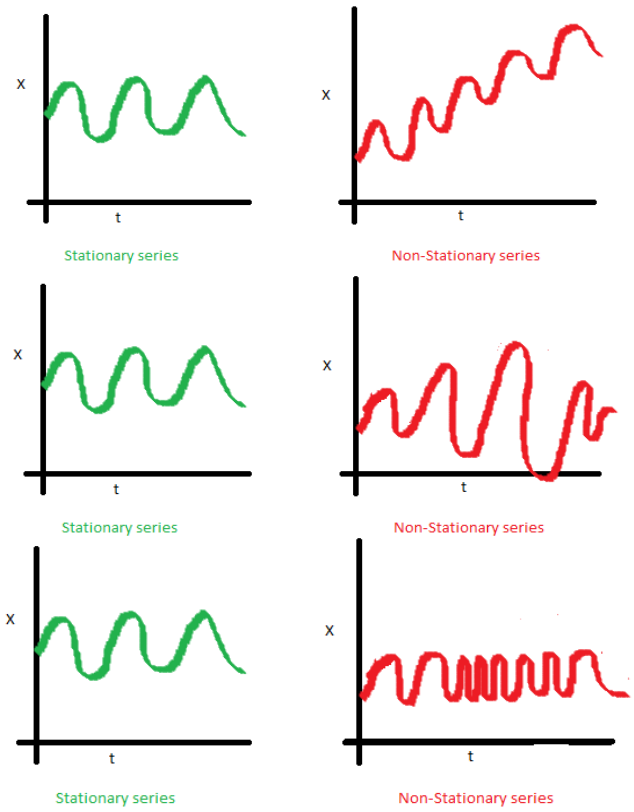
❑ Time series data

- ❖ https://datamarket.com/data/list/?q=provider:tsdl

"Month","Monthly milk production"
"1962-01",589
"1962-02",561
"1962-03",640
"1962-04",656
"1962-05",727
"1962-06",697
"1962-07",640
"1962-08",599
"1962-09",568
"1962-10",577
"1962-11",553
"1962-12",582
"1963-01",600
"1963-02",566
"1963-03",653



❑ Stationarity(정상성)

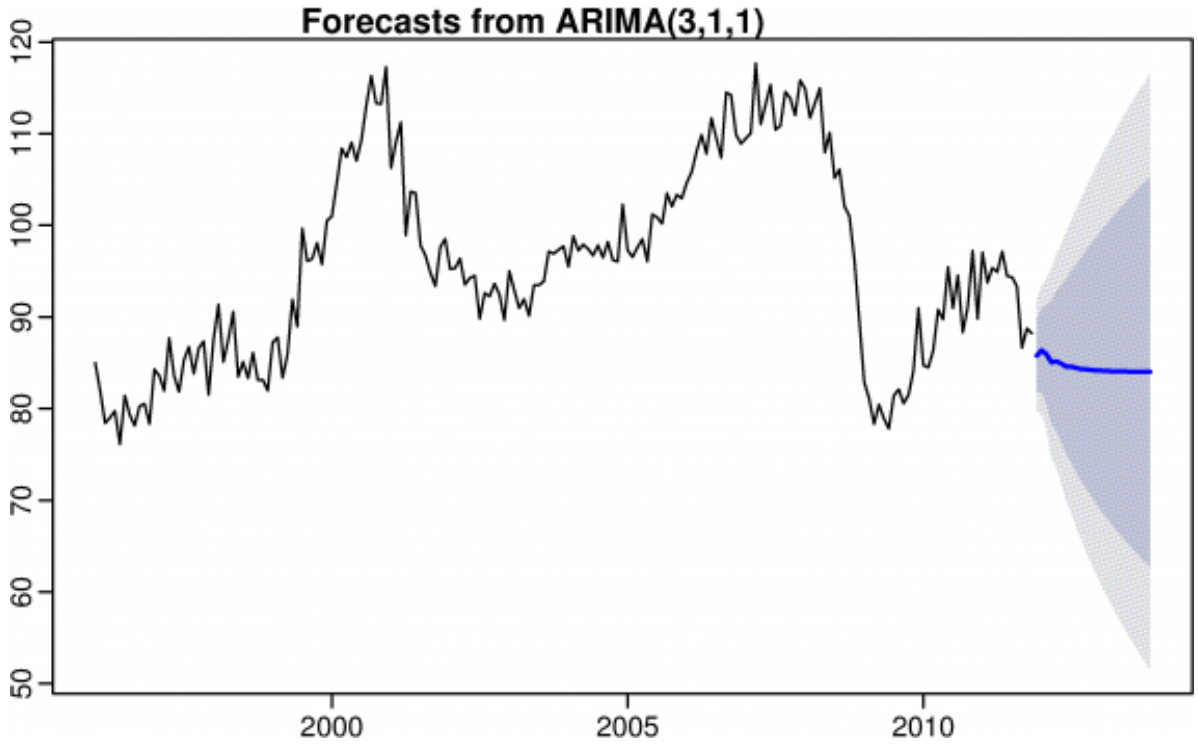


- ❖ 데이터의 평균이 함수가 아니라 상수, 시간축에 평형
- ❖ 데이터의 분산이 함수가 아니라 상수, 시간축에 평형
- ❖ 시계열의 변동이 시간의 흐름에 따라 일정, 시간이 지나도 분산이 일정
- ❖ 시계열의 확률적인 성질들이 시간의 흐름에 따라 변하지 않는다
- ❖ 평균과 분산 등 체계적인 변화가 없고 주기적인 변화가 없다

❑ Non-Stationarity

- ❖ trend : 시간의 흐름에 따라 평균이 변함
- ❖ Seasonality : 특정 구간 별로 변형이 일어 남
- ❖ 비정상성 데이터를 정상성 데이터로 변환
  - ▪ 변형을 통한 Trend 제거 : log, square root, cube root
  - ▪ Aggregation, smoothing, polynomial filtering(regression model) 등을 통한 예측 값 사용
  - ▪ 이동평균(Moving Average) 사용
  - ▪ Trend, Seasonality 제거 : Differencing, Decomposition

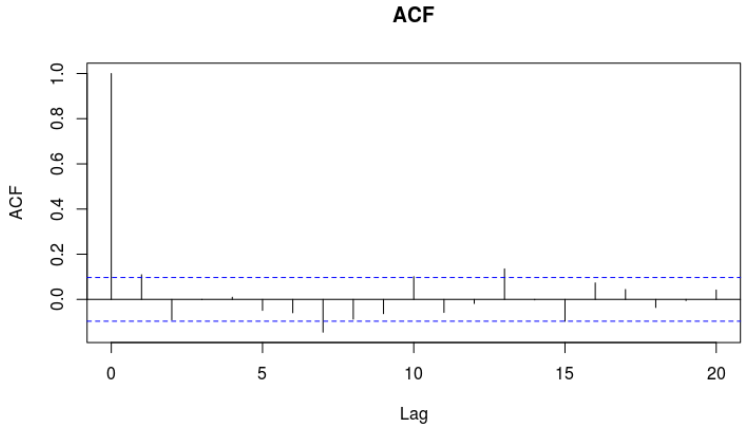
❑ ARIMA(Autoregressive Integrated Moving Average)



❑ ARIMA

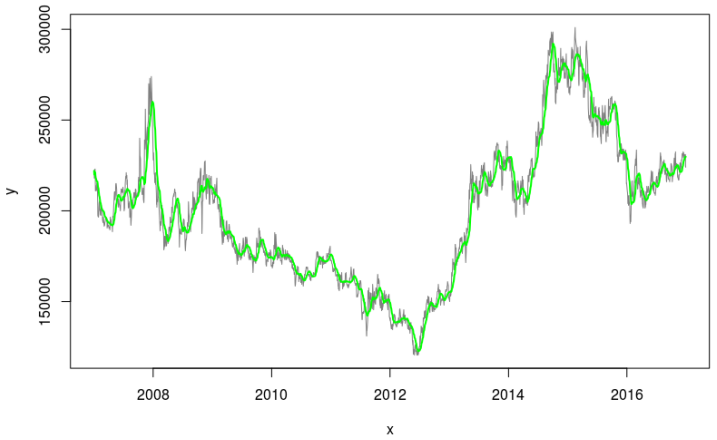
❑ AR(Autoregressive)

$$y_t = \alpha_1 y_{t-1} + \ldots + \alpha_p y_{t-p} + \varepsilon_t$$



❑ MA(Moving Average)

$$y_t = \mu + \beta_0 \varepsilon_t + \beta_1 \varepsilon_{t-1} + \ldots + \beta_q \varepsilon_{t-q}$$



❑ ARIMA

❑ ARMA(Autoregressive Moving Averag)

$$y_t = \mu + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p} + \beta_0 \varepsilon_t + \beta_1 \varepsilon_{t-1} + \ldots + \beta_q \varepsilon_{t-q}$$

❑ ARIMA(Autoregressive Integrated Moving Averag)

$$\Delta^d y_t = \mu + \alpha_1 \Delta^d y_{t-1} + \alpha_2 \Delta^d y_{t-2} + \ldots + \alpha_p \Delta^d y_{t-p} + \beta_0 \varepsilon_t + \beta_1 \varepsilon_{t-1} + \ldots + \beta_q \varepsilon_{t-q}$$

- 이상의 모형들은 관련 시계열 변수가 (약)정상적이라는 전제하에 사용될 수 있음
- ARIMA(p,d,q) : 대부분의 경제 시계열 변수는 비정상적이며, d 차 누적되어 있는 I(d) 시계열을 d 차 차분하여 정상 시계열로 만들어 ARMA(p,q)에 적용
- 관측된 시계열을 발생시킨 것으로 해석될 수 있는 통계적 모형을 확인하고 추정
- 예측의 목적으로 사용하기 위해서는 이 모형의 통계적 특성(특히 평균, 분산)이 시간이 흐름에도 불구하고 일정해야 함 (정상시계열의 사용이 필요함)

❑ Generalized additive regression model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

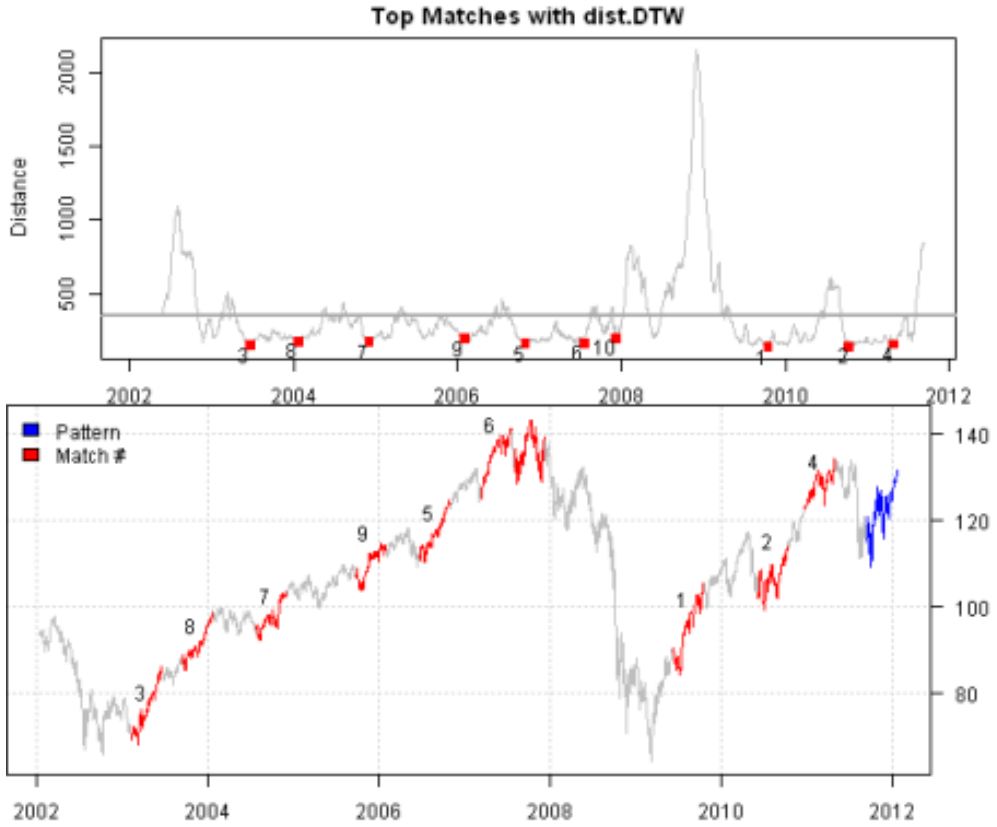
$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

$$= \beta_0 + \underbrace{f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip})}_{\text{비선형 함수}} + \epsilon_i.$$

- Nonparametric regression
- additive model : 각각의 x_j에 대해 분리된 f_j 를 계산한 다음 그것들의 모든 기여도를 한데 모으는 모델

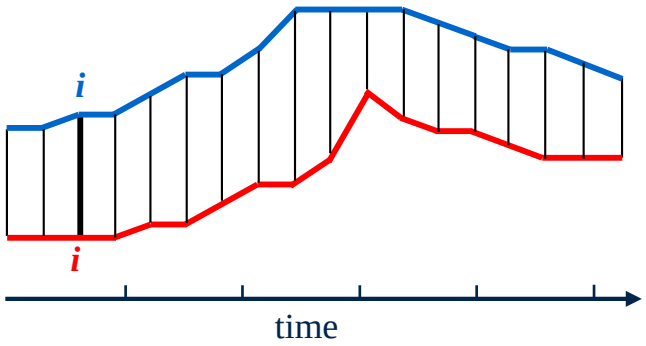
- 표준적인 선형 회귀가 놓치는 비선형 관계를 자동으로 모형화
  비선형 적합은 좀 더 정확한 예측을 잠재적으로 가능하게 한다
  모형이 가산적 이므로 y_j 에 대한 x_j 각각의 영향을 다른 변수들은 모두 고정하고 개별적으로 조사 가능, 추론에 효과적
- 많은 변수들에 대해, 중요한 상호작용이 누락될 수 있다

❑ DTW(Dynamic Time Warping)

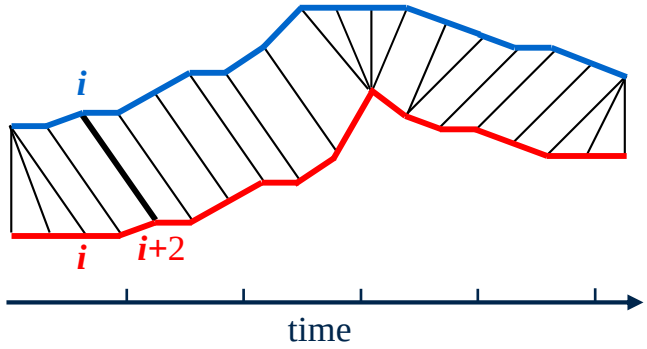


❑ DTW

❑ Euclidean Distance



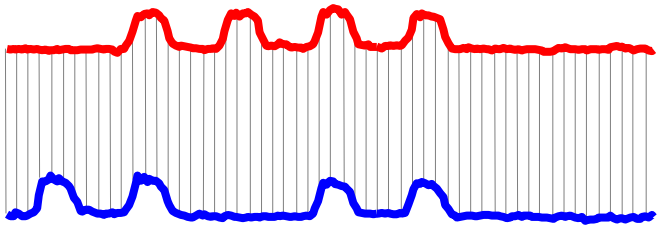
❑ DTW

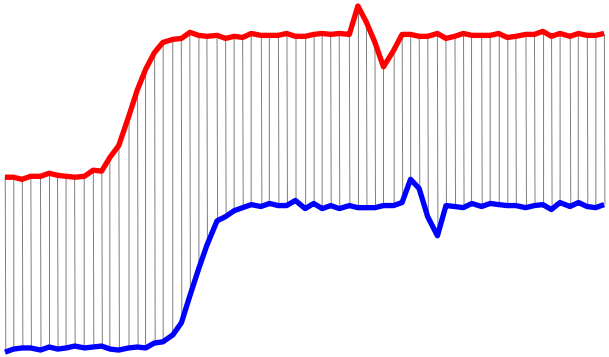


- 시간순서상의 여러 개의 연속적인 데이터를 비교하여 유사도 판별
- 시퀀스의 길이를 고려하지 않고 인식

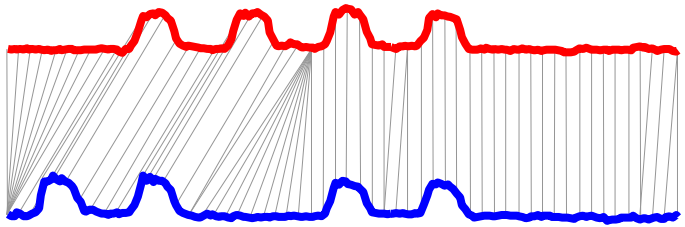
❑ DTW

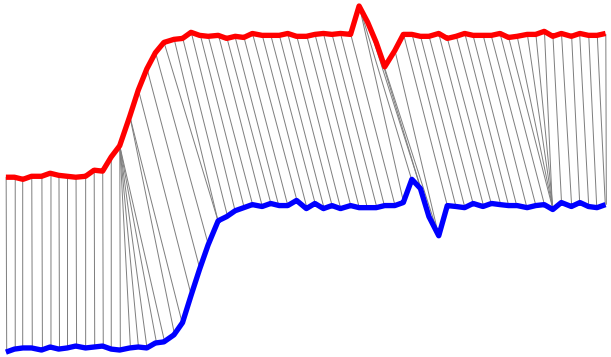
❑ Euclidean Distance



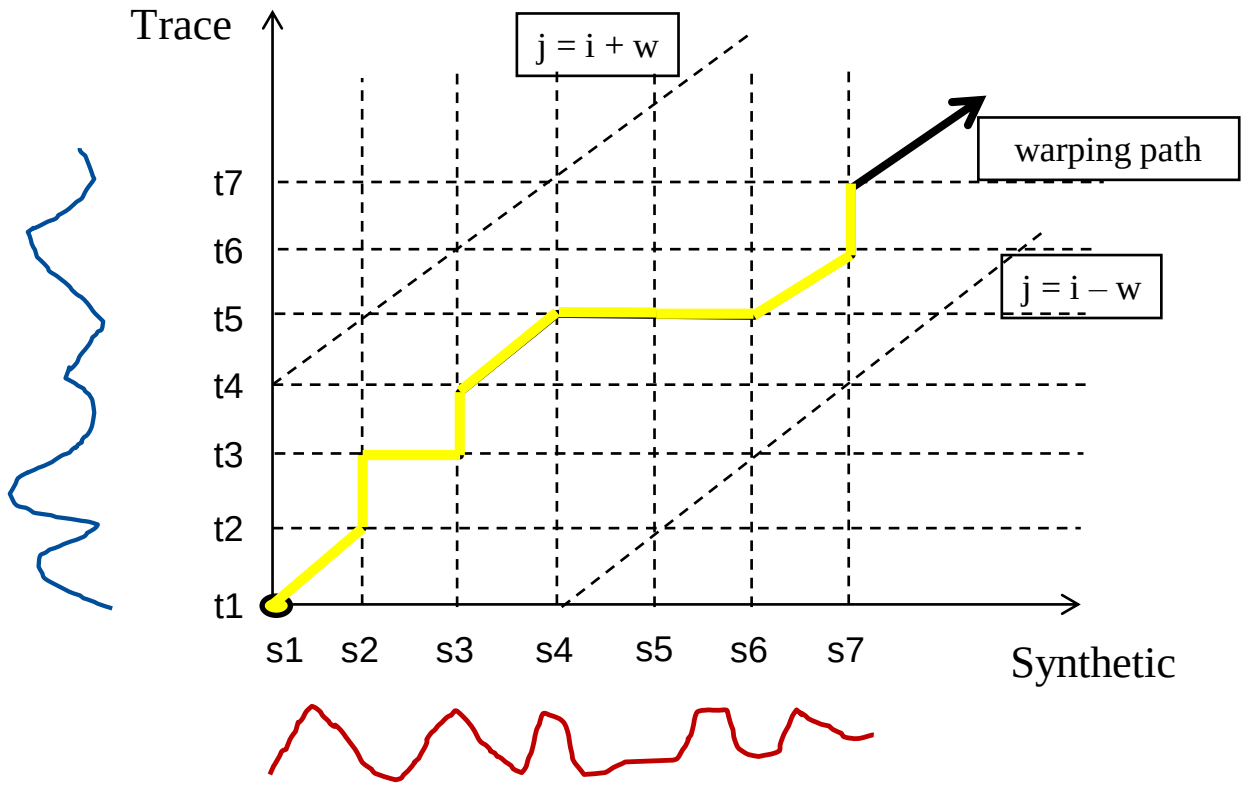


❑ DTW





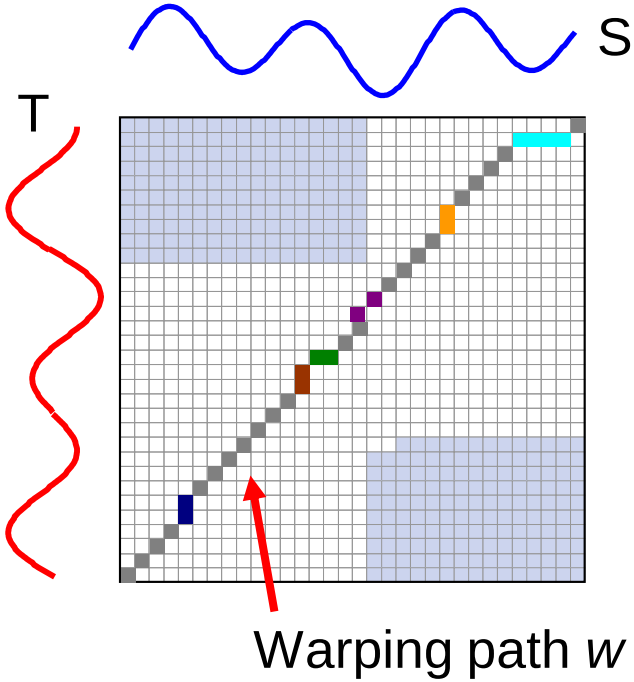
❑ DTW

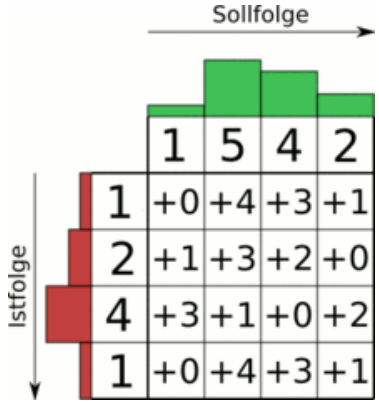


S_warped =	s1	s2	s2	s3	s3	s4	s5	s6	s7	s7
T_warped =	t1	t2	t3	t3	t4	t5	t5	t5	t6	t7

❑ DTW

$$DTW(S,T) = \min\left\{ \sqrt{\sum_{k=1}^{K} w_k} \Big/ K \right.$$





❑ Wavelet transform

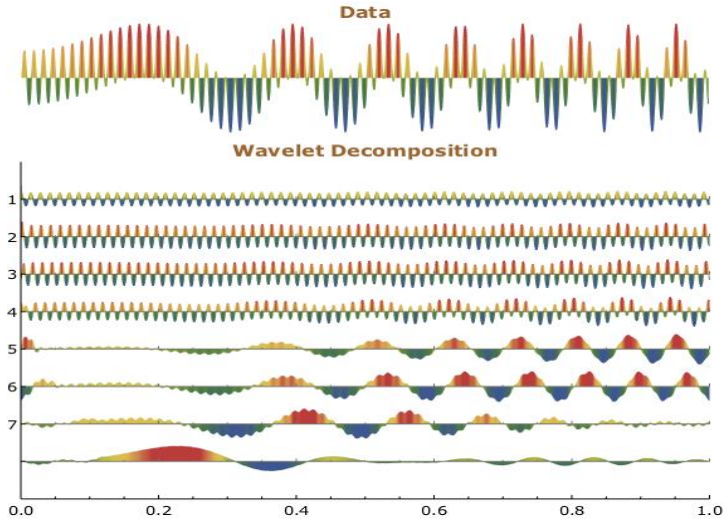
❖ Fourier Transform

- Time domain의 함수를 frequency domain의 함수로 변환
- 데이터에 불연속성, 고주파 성분(이상치)이 포함될 경우 데이터의 특징 분석이 어려움
- 모든 시간에 모든 주파수가 혼합되어져 있는 형태, 데이터의 시간 정보와 주파수 정보를 동시에 파악하기 어려움

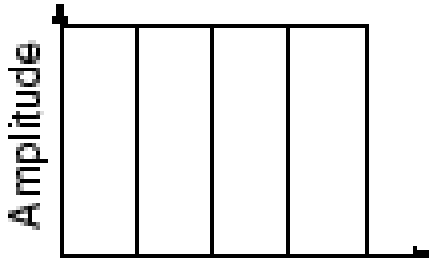
$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(t) \cdot e^{2\pi i \omega t} dt$$

❖ Wavelet Transform

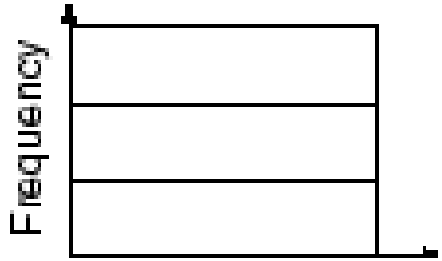
- 데이터를 일정한 크기의 window로 잘라서 특정한 wavelet을 넣어 그 창을 scale화 하여 그 window 안의 데이터가 정상성 데이터가 될 때 각 시간 별 주파수로 변환
- 입력 데이터를 특정기저함수의 집합으로 분리하는 과정



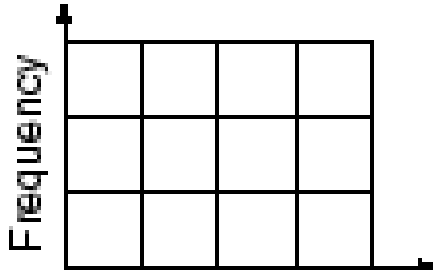
❑ Wavelet transform



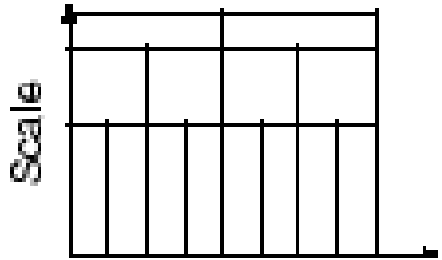
Time Domain (Shannon)



Frequency Domain (Fourier)



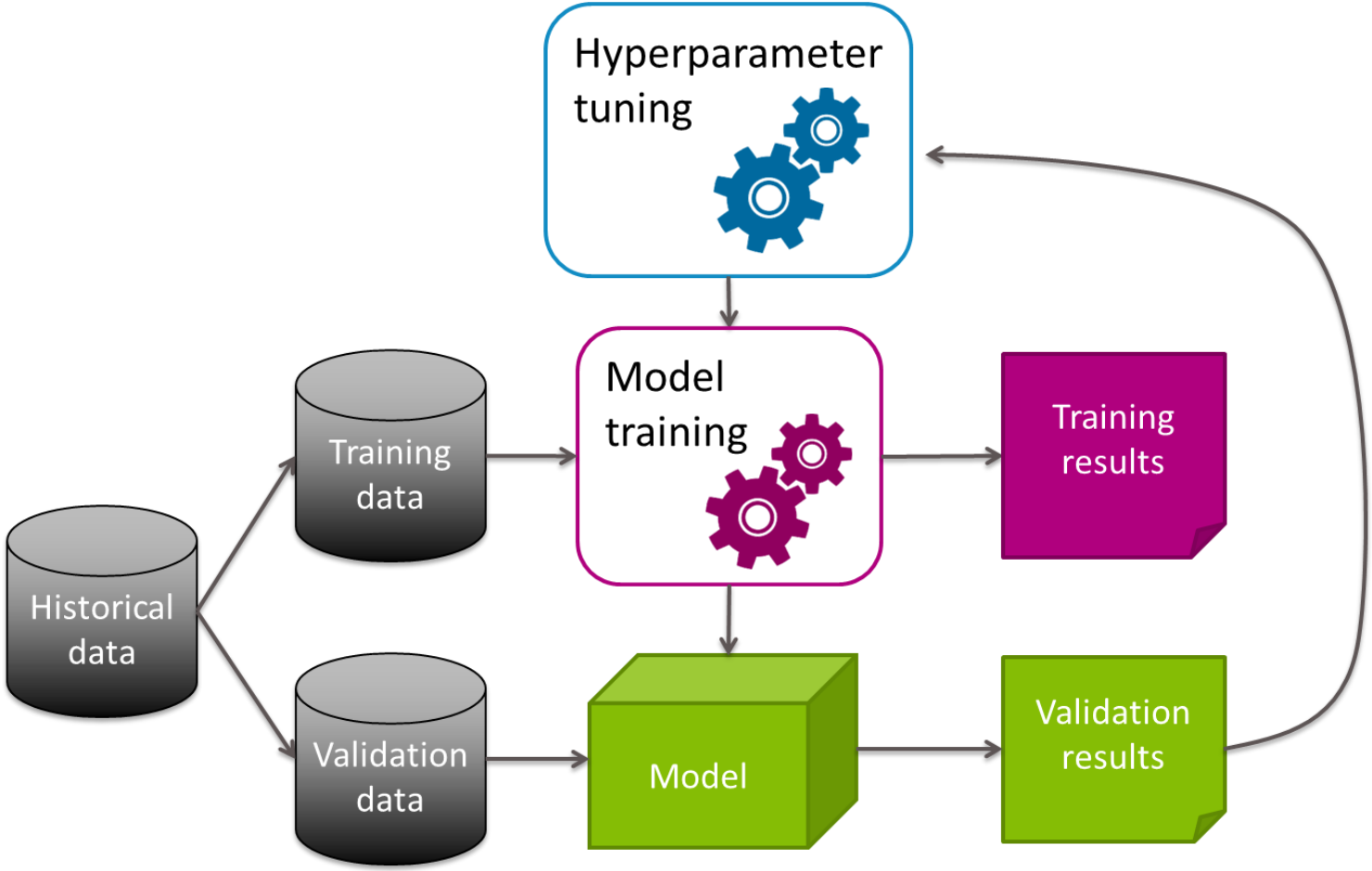
STFT (Gabor)



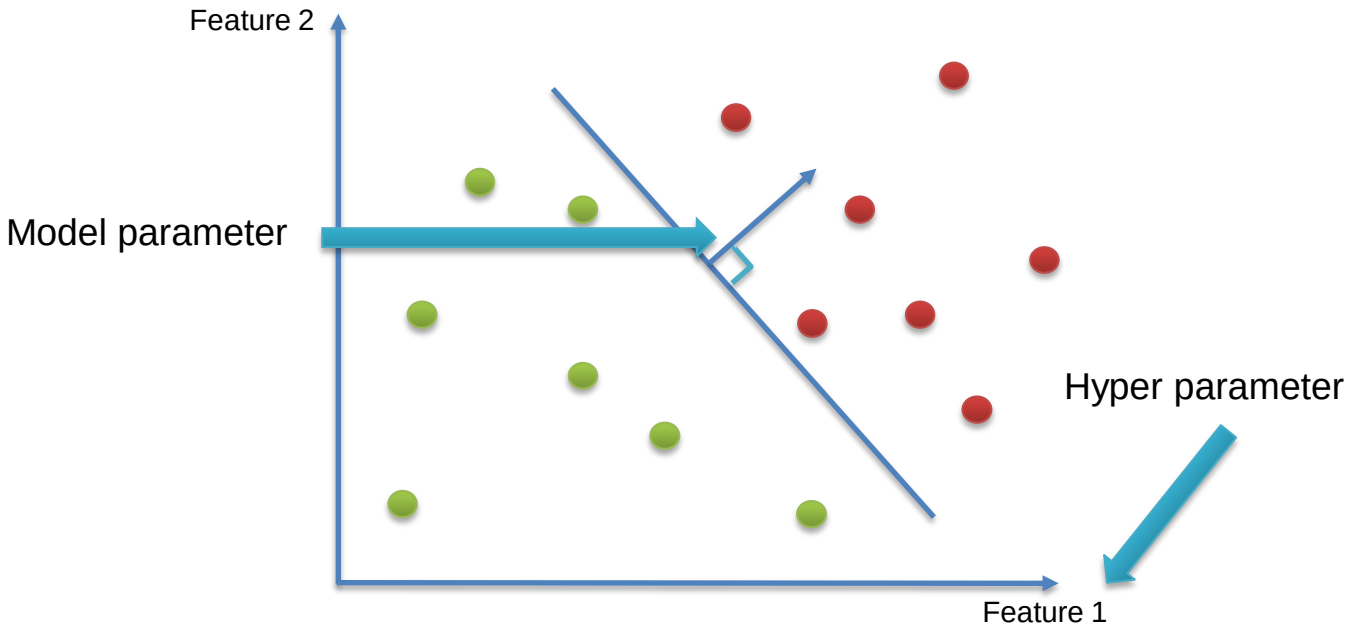
Wavelet Analysis

Model selection

❑ Model selection & tuning

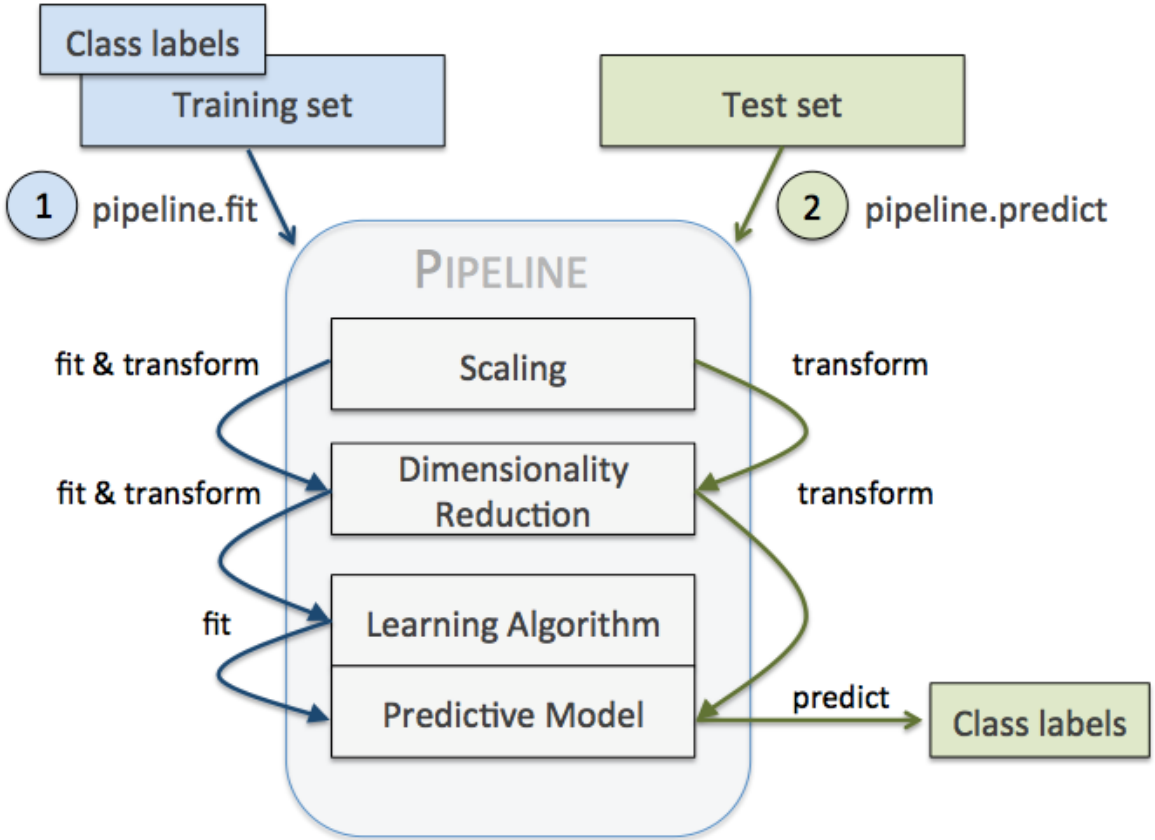


❑ Model parameter training vs. Hyper parameter tuning

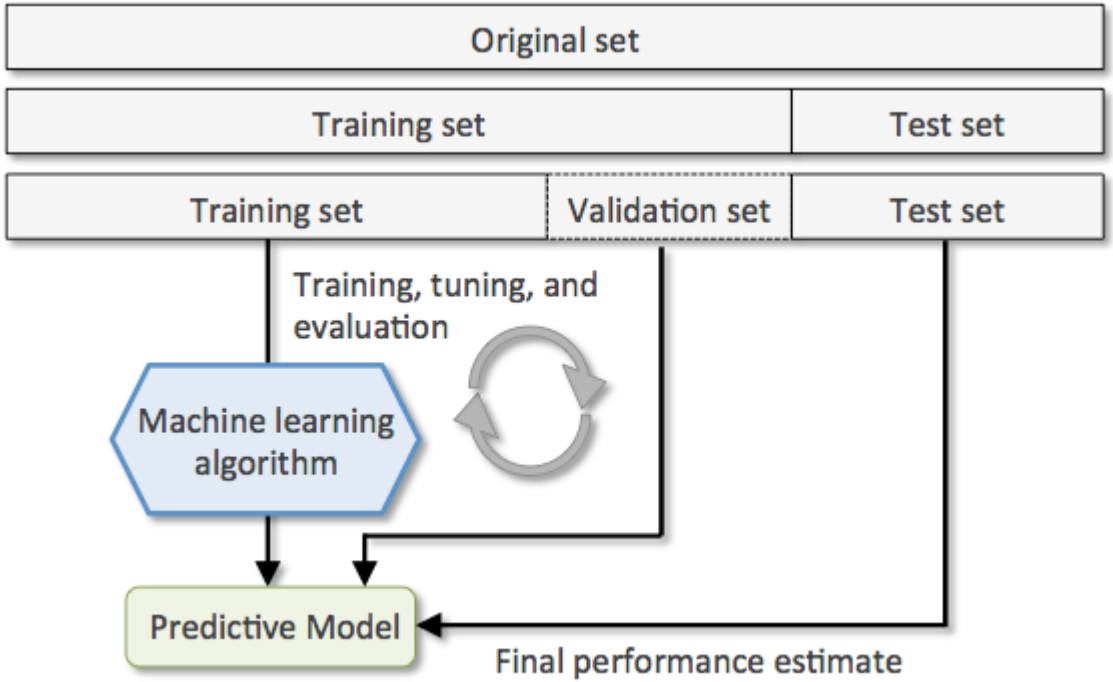


- ❖ model parameter training : weight
- ❖ hyper-parameter tuning : regularization parameter, decision tree depth

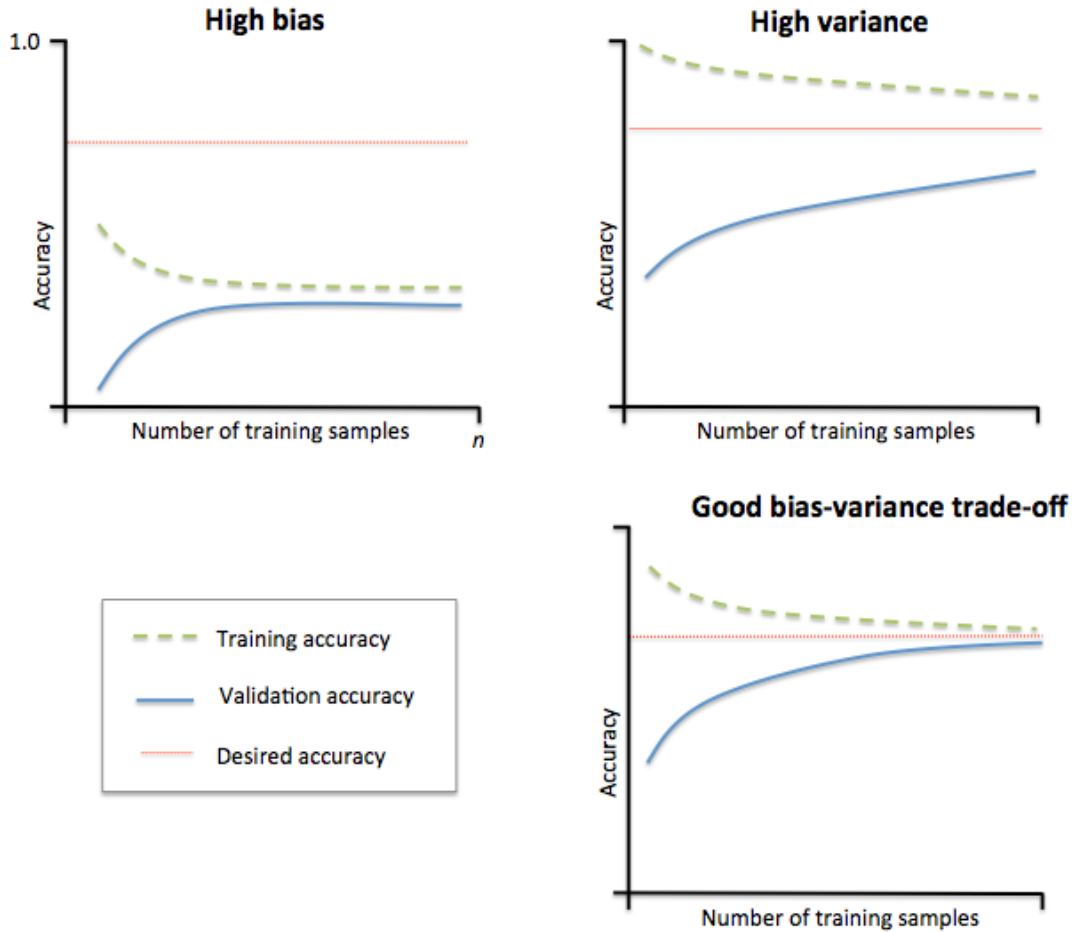
❑ pipeline


Class labels
Training set
Test set
1
pipeline.fit
2
pipeline.predict
PIPELINE
fit & transform
Scaling
transform
fit & transform
Dimensionality Reduction
transform
fit
Learning Algorithm
Predictive Model
predict
Class labels

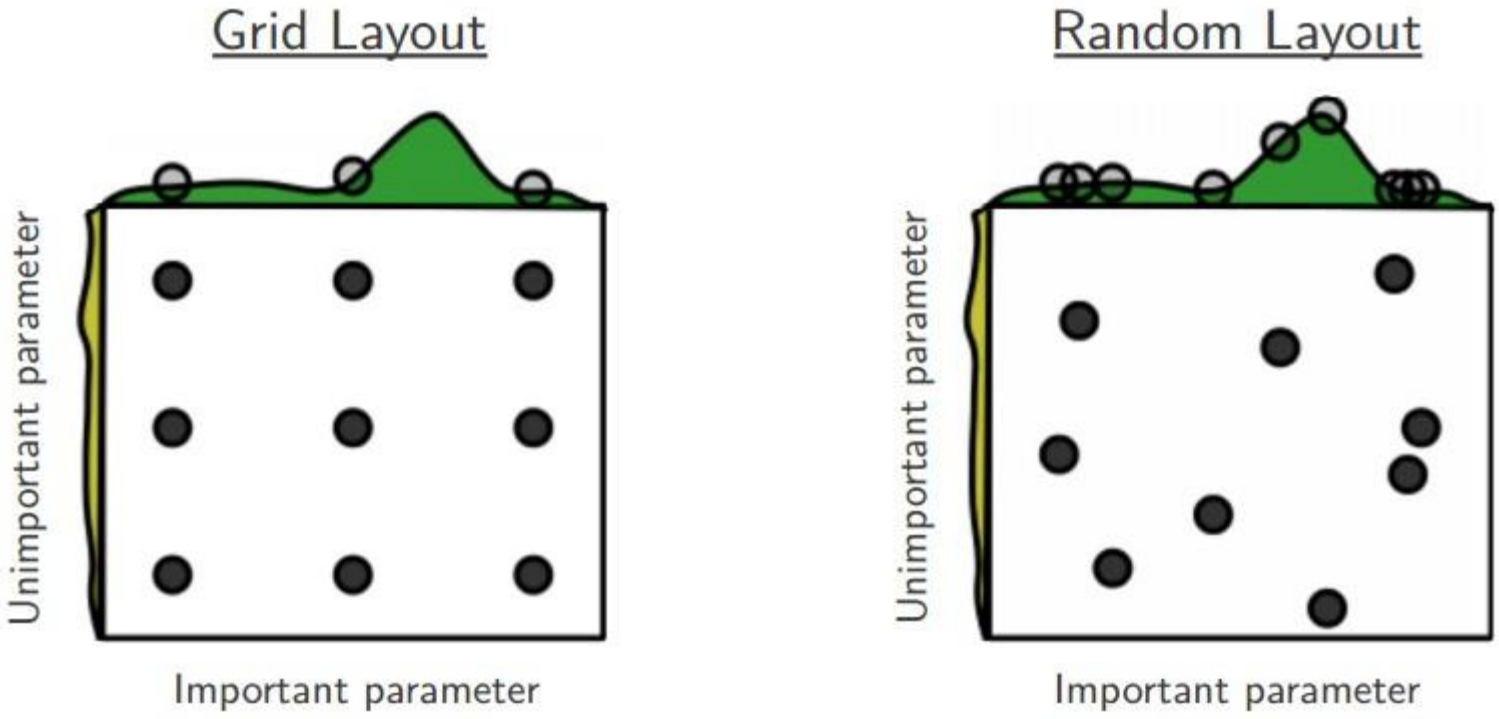
❑ Cross validation



❑ Learning curve



❑ Hyperparameter search



❑ Nested cross validation